NTNU – Trondheim
Norwegian University of
Science and Technology

High Precision Deployment of Wireless Sensors from Unmanned Aerial Vehicles

Siri Holthe Mathisen

Master of Science in Cybernetics and Robotics
Submission date: May 2014
Supervisor: Tor Arne Johansen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Work Description

High precision deployment of wireless sensors from unmanned aerial vehicles.

The task consists of several steps:

1. Given a known end position for the sensor, calculate the optimal position and course of the UAV at the sensor deployment point. Assume that wind speed and direction is known, UAV speed and altitude are given.
2. Develop an algorithm that is used to control the UAV trajectory to the optimal deployment point.
3. Implement an embedded computer system for execution of sensor deployment, based on the following principles.
   (a) Use Piccolo SL autopilot in a Penguin B UAV and associated ground control station with Command Center mission planning and execution software.
   (b) Implement the custom software for coordination of sensor deployment on a Pandaboard onboard computer. Use waypoint lists or turning rate as the primary command to the Piccolo SL, use navigation sensors in the Piccolo SL autopilot, and assume that the end position of the sensor is fixed and given. Use DUNE for integration of the new functionality in the onboard computer.
   (c) Implement on the Pandaboard an interface to a simple sensor deployment mechanism that is integrated as a payload on the Penguin B UAV.
   (d) Develop a user interface with DUNE for execution and monitoring of the sensor deployment functionality. Normally the UAV is operated from the Piccolo Command Center. When a sensor deployment target is set, the command can be given to the onboard system to take over the command of the UAV to initiate and execute the deployment. After deployment is completed, command is given back to the Piccolo Command Center. The UAV operator must be able to abort the deployment and take command of the UAV at any time.
   (e) Wind speed and direction estimates from the Piccolo autopilot may be used, or a dedicated manoeuvre (like flying a circle) can be executed in order to get a reliable wind estimate before the above algorithms are executed.
4. Test the system using hardware-in-the-loop simulation setup in the lab.
5. Test the system with field experiments, to the degree that available equipment and vehicle permits it.

Preface

The relevance of a Master Thesis can be a great motivation when working on it. To see a purpose to the project does not only provide a reason for working on it in the first place, it also helps making the choices that must inevitably be made during the work. Another drive to perform in this project is the joy of working together with another Master Student. Although it might sometimes complicate the process, as what is intuitive to one is not necessarily understandable by another, the benefit is vast. The result is greater than the sum of the two separate achievements, and working together is more fun than doing it alone.

This is a Master Thesis concerning work done in the spring of 2014 at the Department of Engineering Cybernetics. The project is taken at NTNU in cooperation with the Centre for Autonomous Marine Operations and Systems (AMOS). The initiative to the Master Project came from AMOS as part of AMOS' Project 3: Autonomous unmanned vehicle systems, and the work, but not the thesis, is done in cooperation with Master student Simen Fuglaas.

The objectives described in the Work Description will be distributed between the author and Simen Fuglaas as follows:

1. Simen Fuglaas will calculate the optimal position and course of the UAV at the sensor deployment point.

2. Simen Fuglaas will do research on the subject of finding an algorithm that is used to control the UAV trajectory to the optimal deployment point. He will then present the possible solutions, after which he and the author together will find a suited solution.

3. The author will implement the embedded computer system for execution of sensor deployment, together with Simen Fuglaas on some parts:

    - The author will get familiar with programming in C++ with the use of DUNE.
    - The author will get familiar with the Pandaboard, with cross compiling and with running programs on the Pandaboard.
    - Simen Fuglaas will implement in C++ the calculations for an optimal position and course of the UAV at the sensor deployment point.
    - Simen Fuglaas will implement in C++ the algorithm that is used to control the UAV trajectory to the optimal deployment point.
    - The author will design a hardware architecture for the payload mount of the Penguin B and implement it.
    - The author and Simen Fuglaas will together choose a sensor deployment method for the UAV.
    - The author will implement the drop mechanism that is fit for the sensor deployment method including all features needed to make this work.

- The author will create a user interface with DUNE that communicates with the software on the UAV.

4. Simen Fuglaas and the author will together test the system using hardware-in-the-loop simulation setup in the lab.

5. Simen Fuglaas and the author will together test the system with field experiments, if time and equipment permits it.

Trondheim, Monday 26th May, 2014

Siri Holthe Mathisen

Acknowledgment

I would like to thank my supervisor, Professor Tor Arne Johansen, for his tireless efforts during the whole semester. I would also like to thank my co-supervisor Mariann Merz for her patient guidance and meetings. Other great contributors are Frederik Stendahl Leira, Kristian Klausen, Torkel Hansen, Lars Semb and Carl Erik Stephansen, all of which have ensured that the practical aspects of the project have gone well.

A great thank goes to my co-worker Simen Fuglaas, who has put up with me during long work hours and testing sessions.

My Vegard Gulaker has been a fabulous support to me throughout the semester as always, and has helped proofreading my Master Thesis. Geir Mathisen has given me valuable feedback on the report, for which I am very grateful.

S.H.M.

Summary

AMOS - Centre for Autonomous Marine Operations and Systems - is a research center at the Norwegian University of Science and Technology. One out of the 9 projects in AMOS researches how basic unmanned aerial vehicle (UAV) operations can be performed, and one of these operations is high precision deployment of a payload from a UAV. UAVs are normally used for tasks that are either too dangerous, too inaccessible or too repetitive for humans. These tasks may include sensors to be placed somewhere inaccessible or first aid equipment to be delivered to disaster areas.

This Master Thesis describes the development of a high precision deployment system of wireless sensors from a UAV. The UAV will be controlled in a trajectory from an arbitrary start position to the point for deployment of the sensor. The Master Thesis contains description of the development and implementation of hardware and equipment that is placed on a UAV called Penguin B, as well as software to control the system. The software is implemented on a Pandaboard, an embedded computer situated on the payload mount of the UAV. The software communicates with the UAV's autopilot through an interface in DUNE, which is an open source software solution delivered by the LSTS research group. DUNE offers easy-to-use interfaces between the control software, the autopilot and all peripheral units.When the UAV reaches its optimal deployment point, it signals a drop mechanism to let go of the sensor.

The system was tested with a hardware in the loop simulator, and with a field test where the UAV was taxing on the landing field. The results of the tests are that both the hardware and software of the system work as they should. The trajectories to the deployment point have been tested with success, and one trajectory is found to be better than the other. It can be concluded that this system has contributed a lot on the subject of high precision sensor deployment, and suggestions for further development are given at the end of the thesis.

Sammendrag

Ubemannede luftfartøyer (UAV) eller droner blir vanligvis brukt til oppgaver som er enten for farlige, for utilgjengelige eller for kjedelige for mennesker. Slike oppgaver kan være å foreta målinger som krever at en sensor er plassert på en utilgjengelig plass, eller å levere nødhjelpsutstyr til katastroferammede områder. AMOS - Senter for Autonome Marine Operasjoner og Systemer - er et senter for fremragende forskning ved Norges Teknisk-Naturvitenskapelige Universitet. Ett av prosjektene det jobbes med i AMOS er å forske på forskjellige grunnleggende operasjoner som droner må kunne utføre. Høypresisjonsslipp av en nyttelast fra en drone er en av disse grunnleggende operasjonene.

Denne masteroppgaven beskriver arbeidet med å utvikle et system for slipp av trådløse sensorer med høy presisjon fra en drone. Dronen skal styres i en bane fra et vilkårlig startsted til et optimalt slippsted, der sensoren løslates. Oppgaven rommer utvikling og implementering av maskinvare og utstyr som plasseres på en drone kalt Penguin B, samt programvare for å styre systemet. Programvaren er implementert på et Pandaboard, en innebygget datamaskin plassert i dronens nyttelastrom. Programvaren kommuniserer med dronens autopilot via et grensesnitt i DUNE, som er en fritt tilgjengelig programvareløsning levert av forskningsgruppen LSTS. DUNE tilbyr enkle grensesnitt mellom kontrollprogrammet, autopiloter og alle periferienheter. Når dronen når det optimale slippunktet signaliserer den til en slippmekanisme som lar sensoren falle til bakken.

Systemet ble testet med et *hardware in the loop*-simuleringssystem, samt utprøvd i praksis på en drone som kjørte på bakken på en rullebane. Resultatene av testene er at både maskinvaren og programvaren i systemet virker som de skal. Flybanene til slippunktet er blitt testet med suksess, of en bane har vist at den er mer korrekt enn den andre. Det kan konkluderes at dette systemet bidrar mye til arbeid med høypresisjonsslipp av sensorer, og forslag til videre utvikling er presentert i slutten av oppgaven.

Acronyms and Explanations of Words Used

NTNU Norges Teknisk Naturvitenskapelige Universitet - Norwegian University of Science and Technology

GPS Global Positioning System

UAV Unmanned Aerial Vehicle

AUV Autonomous Underwater Vehicle

LSTS Laboratório de Sistemas e Technologias Subaquáticas - Underwater Systems and Technology Laboratory

MPC Model Predictive Control

DUNE Unified Navigation Environment

IMC Inter-Module Communications

PWM Pulse Width Modulated

IDE Integrated Development Environment

LAN Local Area Network

DHCP Dynamic Host Configuration Protocol

IP Internet Protocol

SSID Service Set Identifier

POE Power Over Ethernet

HIL Hardware In the Loop

IAS Indicated Air Speed

SD Secure Digital

PoI Point of Impact

PoR Point of Release

GPIO General Purpose Input and Output

CAN Controller Area Network

TCP Transmission Control Protocol

Contents

Chapter 1

Introduction

In this chapter, the Master Thesis is presented. Its background and motivation is explained, the problem is formulated and described in separate steps. The work distribution is also described, as is the structure of the rest of the Thesis.

1.1 Background and Motivation

The Arctic area can be defined in different ways: As the area north of the polar circle, as the area on the northern hemisphere where the average temperature is below 10^oC in the summer months, or as the area north of the northern tree line (Wikipedia, 2014a). Though these definitions do not define the exact same area, the picture is still clear: The Arctic is a very cold place. And in cold places, icebergs appear. Everyone who has seen the film "Titanic" knows what an unfortunate situation an undetected iceberg can cause, and it is therefore a great advantage to the shipping industry to know where the icebergs are.

With the present technology, an iceberg could be equipped with a global positioning system (GPS) transmitter and send its position to a receiving station at all times. However, the problem with icebergs is that they are seldom easily reachable as they float on the water, are never standing still and consist of ice. This makes it difficult to place a GPS-transmitter on the iceberg. It is a risky sport to climb on icebergs, and definitely if one has to enter the iceberg from a ship. An easier solution would be to place the sensor on the iceberg without having to climb on the mountain, by dropping it from the air. This drop would have to be of high precision, as the dropped package should land on the iceberg and not into the water.

An unmanned aerial vehicle (UAV) is smaller, lighter and cheaper than manned aerial vehicles like helicopters or air crafts. There are two types of UAVs: the fixed-winged UAVs and the rotorcraft designs. A rotorcraft UAV is like a small helicopter and would be easy to land on the iceberg, but the fixed-wing UAV has a longer range and is therefore more suitable to tough missions like placement of a sensor on a far-away iceberg.

Other possible applications for high precision deployment of a payload from a UAV is the

Figure 1.1: A UAV deploying a sensor on an iceberg (graphics from UAV Factory (2014), SiriusXM (2014))

deployment of supplies to devastated areas, commercial package delivery or sensors on other places that are as unreachable as an iceberg. UAVs are used for tasks that are either so demanding or so repetitive and dull that human beings do not want to do it.

To have UAVs conduct tasks that are dangerous or unsuited for humans is not new. It saves money as it does not require a crew on the UAV, and the UAV demands less fuel and costs less than a larger aircraft. Tuna et al. (2012) published an article about how a UAV can be used to drop payloads and sensors in post-disaster areas. They wanted to find out how effective the localization and navigation of their system for a wireless sensor network was, and concluded that it is possible to deploy a wireless sensor network on a predetermined location from UAVs. Corke et al. (2004) explains in his article how unmanned helicopters can deploy sensors, also if there are several sensors that should be dropped in series. Wuest and Benney (2005) describes different methods that are currently in use for deploying payloads from air crafts with the use of parachutes. The article presents, among other air delivery technologies, five different payload deployment methods, all of which use parachutes to safely deliver the load to the ground. McGill et al. (2011) describe research done on the field of dropping sensors on icebergs. A remotely controlled UAV is launched from a ship nearby the iceberg, releases the sensor on the iceberg and returns to the ship. The article explains that the challenge was to return the UAV safely to the ship without crashing, and that the sensors released over the iceberg were operating well in approximately half of the cases.

The research centre AMOS - Centre for Autonomous Marine Operations and Systems - is a Centre of Excellence at NTNU. It is a cooperation between the Departments of Engineering Cybernetics and Marine Technology with ties to the industry, and works with operations on UAV among other systems. One out of the 9 projects in AMOS is called *Autonomous unmanned ve-*

hicle systems and is concentrated on how to perform basic operations with a UAV, like dropping a payload or picking it up again. This Master Thesis is grounded on that research project. It is conducted in cooperation with another Master Student, Simen Fuglaas, and is closely related to a series of other projects that work on the *Autonomous unmanned vehicle systems.*

1.2 Problem Formulation

In this Master Thesis, the problem formulation is as follows: Develop high precision deployment of wireless sensors from unmanned aerial vehicles (UAVs). The UAVs should be autonomous.

1.3 Objectives

The task consists of several steps:

1. Given a known end position for the sensor, calculate the optimal position and course of the UAV at the sensor deployment point. Assume that wind speed and direction is known, UAV speed and altitude are given.

2. Develop an algorithm that is used to control the UAV trajectory to the optimal deployment point.

3. Implement an embedded computer system for execution of sensor deployment, based on the following principles.

   (a) Use Piccolo SL autopilot in a Penguin B UAV and associated ground control station with Command Center mission planning and execution software.

   (b) Implement the custom software for coordination of sensor deployment on a Pandaboard onboard computer. Use waypoint lists or turning rate as the primary command to the Piccolo SL, use navigation sensors in the Piccolo SL autopilot, and assume that the end position of the sensor is fixed and given. Use DUNE for integration of the new functionality in the onboard computer.

   (c) Implement on the Pandaboard an interface to a simple sensor deployment mechanism that is integrated as a payload on the Penguin B UAV.

   (d) Develop a user interface with DUNE for execution and monitoring of the sensor deployment functionality. Normally the UAV is operated from the Piccolo Command Center. When a sensor deployment target is set, the command can be given to the onboard system to take over the command of the UAV to initiate and execute the deployment. After deployment is completed, command is given back to the Piccolo Command Center. The UAV operator must be able to abort the deployment and take command of the UAV at any time.

   (e) Wind speed and direction estimates from the Piccolo autopilot may be used, or a dedicated manoeuvre (like flying a circle) can be executed in order to get a reliable wind estimate before the above algorithms are executed.

4. Test the system using hardware-in-the-loop simulation setup in the lab.

5. Test the system with field experiments.

1.4 Delimitations

As mentioned in Section 1.3, many factors in this Thesis have already been decided and a lot is assumed before my work started. A Penguin B fixed-wing UAV will be the vehicle used in this task, controlled by a Piccolo SL autopilot. A Pandaboard embedded computer will hold the software needed onboard the UAV, and it will use the DUNE framework. DUNE will also be used on the ground station for a user interface. Another delimitation is that the choice of radio communication between the Pandaboard on the UAV and the ground station is already done.

As this work is only intended to be carried out in the course of 20 weeks for two students, it may not be possible to develop an optimal solution without any flaws. Thus, a trajectory optimization using MPC, which would give the optimal path from the start point of the UAV to the deployment point, is omitted in this Master Thesis. The *high precision deployment* is also a goal, although the term *high precision* is a question of definitions and the precision may not be as high as possible.

A simplification that is done is to only look on waypoints in two dimensions, ignoring the height. This is done because the UAV pilot does not want the safety of the UAV to be put at risk by lowering the UAV too much, and because it reduces the workload of the task. That means that the height of the UAV is constant throughout this Master Thesis, and that the deployment of the sensor has to be done at that given height.

Another assumption is that the UAV pilot will launch the UAV, and control it until a given point. This given point is assumed to be the point of impact for the sensor. Although the point of impact is a position on the ground, the UAV will loiter above the ground, at the same hight as the rest of the flight will take place. At this given point where the Pilot releases the control, the system described in this Master Thesis will take over and guide the UAV towards the deployment point.

1.5 Structure of the Thesis

The rest of this Master Thesis is divided into parts to keep the structure as tidy as possible. In the first part called *Background and Theory*, Chapters 2, 3 and 4 are placed. Chapter 2 is concerned with the background information and the literature search used to prepare this project. An annotated bibliography in Appendix C contains the relevant articles. In Chapter 3, the equipment and tools that are used in this thesis are explained and described to give some background information about them. Chapter 4 is concerned with the theory, methods and equations about the physics of a drop, a theory used to find the path from one directed point to another, and other useful theory for UAV navigation.

Part II is called *System Description, Planning and Implementation.* In this part, the preparations are described along with the implementation of the systems and the necessary choices that should be taken. Chapter 5 describes the system and gives an overview that sets the next chapters in a context. Chapter 6 debates the choices of a trajectory calculation method and a drop method, based on the theory described in Chapter 4. Chapter 7 debates the choice of a drop mechanism used to release the sensor from the UAV, the technicalities around this drop mechanism and its implementation on the UAV. Chapter 8 describes the software architecture on the Pandaboard while Chapter 9 describes the hardware architecture of the Pandaboard.

Part III is called *Experimental Procedure* and consists of a single chapter. Chapter 10 describes the verification of the developed hardware parts, the hardware-in-the-loop testing done in the UAV-laboratory as well as field testing done in Agdenes.

Part IV is called *Results and Conclusions* and consists of the results, the discussion, the future work and the conclusion of this Master Thesis. Chapter 11 contains the results of the testing from Chapter 10. Chapter 12 discusses the results found in Chapter 11, the decisions that have influenced the project and the significance of the project. Chapter 13 contains suggestions to future work on the project described in this Master Thesis. Chapter 14 describes what has been performed and achieved in this project, and what can be concluded.

Part I

Background and Theory

Chapter 2

Literature Study

The purpose of this literature study was to investigate previous work done on the field of sensor deployment from a fixed-wing air plane, either unmanned or manned, and on the field of the system architecture onboard a UAV that uses an autopilot.

A lot of information was found on the precise details of trajectory generation and tracking, and all the articles containing information about this was given to Simen Fuglaas, as that was his field of investigation. The most relevant articles for the author's part of the work were the articles by Corke et al. (2004), Williams and Trivailo (2006), Ducote and Speelman (1966) and Wuest and Benney (2005) for giving suggestions to a deployment method for the sensors. The article by McGill et al. (2011) also gave valuable experience on the field of dropping a sensor on an iceberg with a fixed-wing UAV. The advantages and disadvantages of the different deployment methods are described in Section 6.1

The other field of investigation, the system architecture on a UAV with an interface to an autopilot, was useful as the articles described their solution. The articles that explain research done with the LSTS tool chain (which will be explained in Chapter 3), written by Santamaria et al. (2007), Pinto et al. (2012), Oliveira et al. (2011) and Pinto et al. (2012), were particularly interesting as they described the mode of operation of the LSTS tool chain. LSTS is a Portuguese abbreviation that means *Underwater Systems and Technology Laboratory*. The scientists working in that group have developed a tool chain that they have called the LSTS tool chain. This will be further explained in Section 3.1.

2.1 Search Words for the Literature Study

Ballistic Deployment on Scopus.com 09.01.2014: 182 document results in the field of engineering. 5 abstracts read. No articles read

Software Architecture UAV on Scopus.com 09.01.2014: 95 document results. 4 abstracts read. 0 articles read.

System Architecture UAV Payload on Scopus.com 09.01.2014: 10 document results. 2 abstracts read. 0 article read.

Software Architecture Payload Autopilot on Scopus.com 09.01.2014: 4 document results. 2 abstracts read. 1 article read (Santamaria et al., 2007).

Optimal Trajectory UAV on Scopus.com 13.01.2014: 236 document results. 19 abstracts read. 2 articles read (Bousson and Machado, 2013), (Lai et al., 2011).

Onboard Architecture UAV on Scopus.com 13.01.2014: 83 document results, 10 abstracts read. 1 articles read (Dong et al., 2007).

System Architecture Design UAV on Scopus.com 15.01.2014: 296 document results, 8 abstracts read. 2 articles read (Tang and Li, 2011), (Omari et al., 2013).

UAV on Scopus.com 23.01.2014: 76 document results. 2 abstracts read. 0 articles read.

Deployment Unmanned Aerial Vehicle on Scopus.com 23.01.2014: 387 document results. 18 abstracts read. 3 articles read (Tuna et al., 2012), (Corke et al., 2004), (Maza et al., 2010).

In addition to this, two papers about the LSTS tool chain ((Pinto et al., 2012), (Pinto et al., 2012)) have been given to me by Remus Barbatei, who worked on the subject last semester. Three papers ((Williams and Trivailo, 2006), (Ducote and Speelman, 1966), (Wuest and Benney, 2005)) about high precision payload drop from air planes and one paper ((Oliveira et al., 2011) about the LSTS tool chain have been given to me by Mariann Merz. An additional paper about deployment of sensors on icebergs using an UAV ((McGill et al., 2011)) was found by Tor Arne Johansen. An annotated bibliography is found in Appendix C.

2.2 Payload Deployment Methods

Several payload deployment methods have been discussed in the literature. A few of which will be presented here and discussed in Chapter 6.

Free Fall without Parachute

In their Arctic research, McGill et al. (2011) used a UAV to drop sensors down to an iceberg. No parachutes were used to soften the landing, but the sensors were wrapped in a soft toy football for protection with four rods pointing out. The rods were meant to prevent the sensor from sliding off the iceberg(McGill et al., 2011). In the article, the researches described that the UAV was controlled by a pilot standing on a ship some hundred meters away, aided by a camera. Three out of the four dropped sensors survived the fall and sent reports back to the receiving station, but only one of the four non-functioning sensors was for certain delivered on the iceberg(McGill et al., 2011). The article also described that the landing of the UAVs was problematic.

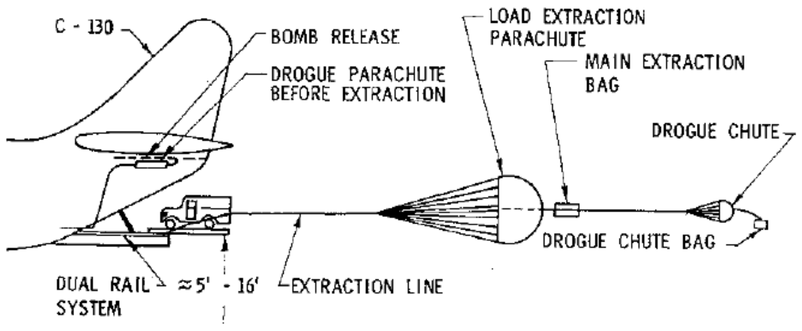


Figure 2.1: Drop of a vehicle with parachute and drogue parachute, by Ducote and Speelman (1966)

Free Fall with Parachute

Wuest and Benney (2005) describes a free fall with parachutes in their article *Precision Airdrop*. The system called Sherpa consists of a drogue parachute in addition to a main canopy connected to the payload. Another system described in their article is the SCREAMER, a parachute delivery system using a ram air drogue in addition to round, larger canopies. The DRAGONFLY system described in the same article is another parachute delivery system, as are the Capewell and Vertigo AGAS, the Atair ONYX and SPADES. Wuest and Benney (2005) conclude that there have been many advances in the precision airdrop technologies and that these changes will continue in foreseeable future.

Ducote and Speelman (1966) present other research done on parachute guided delivery of payloads from aircrafts, in an article called *U.S. Air Force Concepts for Accurate Delivery of Equipment and Supplies*. They explain how the aerial delivery altitude can influence the choice of delivery methods, and recommend a free fall for objects that are released from under 20 Ft.

Gravity Drop with parachute is illustrated in Figure 2.1 with an illustration from Ducote and Speelman (1966).

Guided fall with a Spiralling Wire Slide

A guided fall is here the opposite of a free fall: The payload is guided to the ground instead of disconnecting it from the vehicle and letting it fall freely. This method is one of a family where an aircraft flies in circles and a payload is released to the ground with a wire. Normally, the payload is attached to the tip of the wire, and the stability depends on the weight of the payload - the heavier, the more stable. However, Williams and Trivailo (2006) used the wire as a slide that delivered the payload in circles to a point on the ground. A constant heavy weight was placed on the cable tip and this was preferably anchored to the ground.

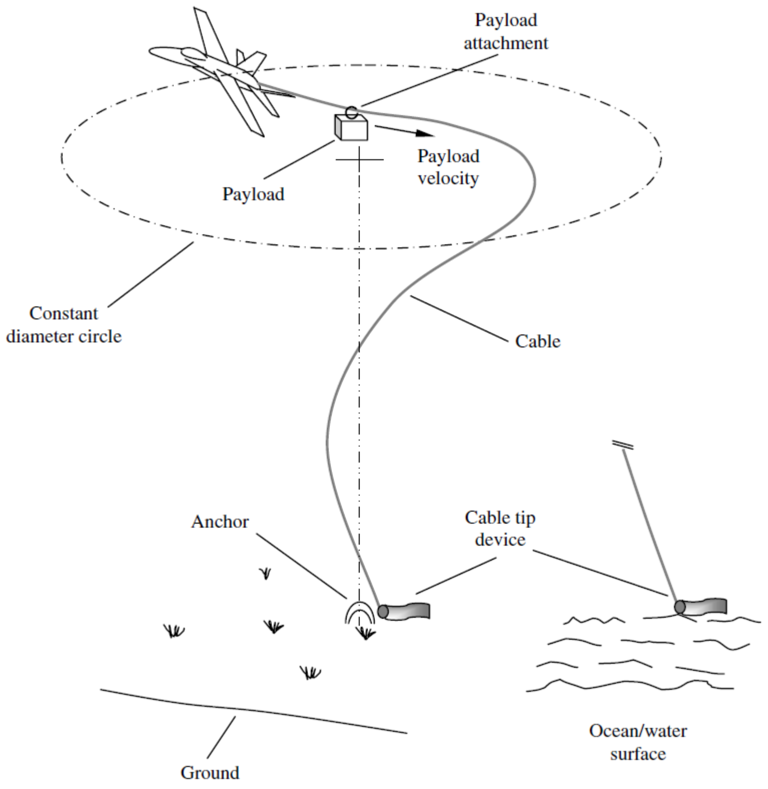


Figure 2.2: The Spiraled Slide illustrated by Williams and Trivailo (2006)

Williams and Trivailo (2006) demonstrated that if it is possible to anchor the cable tip to the ground, this will guarantee that the payload reaches the correct point of impact. However, the speed it gets along the wire would have to be braked down to reduce the damages to the payload upon the impact on the ground. The method is illustrated in Figure 2.2 with a figure from (Williams and Trivailo, 2006).

Guided Fall on a Wired Coil

This method is described by Corke et al. (2004). Its main purpose is to deliver a sensor network, finding a way to deploy many sensors after each other with a deterministic distance between them. The deployment method is to use a wired coil that is controlled by a servo. The payloads are attached with hooks to this coil at given intervals. When the servo rotates the coil, the payloads fall off, one by one. The details on how this is supposed to happen are not given, and no good illustration of the mechanism is found in the article. The method is attempted illustrated in Figure 2.3.

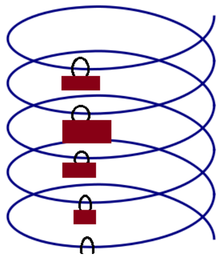


Figure 2.3: Illustration of the payload deployment with a wired coil. The payloads hang with a constant interval on the coil, and as it is rotated, they fall off one by one.

2.3 The LSTS Toolchain

Pinto et al. (2012) and Pinto et al. (2012) describe the LSTS toolchain in their articles *Implementation of a Control Architecture for Networked Vehicle Systems* and *Experiments with Deliberative planning on Autonomous Underwater Vehicles*. The LSTS group consists of interdisciplinary scientists mainly based in Portugal who research marine technology and applications for unmanned vehicles in or near the sea (Laboratório de Sistemas e Tecnologias Subaquáticas, 2013). The LSTS toolchain consists of one communication protocol called IMC, one software framework called DUNE and one graphical user interface called Neptus, which all work together to create an interface for control of unmanned vehicles.

The article by Pinto et al. (2012) introduced all these three features and describes what their areas of applications are. An overview is given, as well as a detailed description of the qualities of DUNE, Neptus and the IMC protocol. The article by Pinto et al. (2012) explains how this toolchain can be combined with other tools. The article contains experiences with the LSTS toolchain combined with the Teleo-Reactive Executive (T-REX), which is a plan execution framework.

2.4 Summary

The literature study in this Master Thesis was thorough and comprehensive considering the amount of knowledge about the system on that time. All thinkable search word combinations were tried, but with poor results: Out of 1379 document results of the searches, only 70 abstracts were read and only 9 articles were read. However, this was a very systematic way of structuring the information on the fields of study, and it was useful to the process as it catalogued the searches on the way. This way, the search word combinations are ploughed through and no searching had to be done twice on the same search words. The articles that are presented in this chapter contain useful ideas and information that is used in this Master Thesis. In the annotated bibliography in Appendix C, all articles that were read are summed up briefly together with their relevance.

Very useful information was found on the field of payload deployment methods and on the

LSTS tool chain, and this information will be used in later chapters.

In hindsight, searches on *Arctic research* would also be relevant. The article by McGill et al. (2011) was very relevant as background information, and it did not appear in any of the searches on the Scopus search engine mentioned above. The searches on the software and system architecture should also not have been emphasized that much, as they served as inspiration more than as guidelines. However, they served their purpose as they showed a diversity of architectures and demonstrated that many solutions can lead to a great design. And besides, a lot of the read articles gave useful information, like the use of the LSTS tool chain and the sensor deployment methods. And although many of the other articles were of little use for the author's part of the work, they were useful to Simen Fuglaas and to the collective good.

Chapter 3

Tools and Equipment

This Master Thesis uses a lot of equipment that works together, both software and hardware. Figure 3.1 shows an overview of how the technology is supposed to cooperate: The LSTS Software Toolchain is supposed to be a tool that facilitates the communication with the peripherals and autopilot onboard an unmanned vehicle and with the communication between the unmanned vehicle and the ground station. The Pandaboard is a computer that is mounted inside the unmanned vehicle and the Cloud Cap Technology manages the autopilot functionality of the unmanned vehicle. Ubiquity Networks supplies the system with radio equipment for the communication between the vehicle and the ground station that does not carry autopilot information.

3.1 The LSTS Software Toolchain

The Underwater Systems and Technology Laboratory (LSTS), connected to the Department of Electronic and Computational Engineering of the University of Porto, is a research group specialized on unmanned vehicles operating under water, on the water surface or in the air. The LSTS has developed a software tool-chain consisting of the programs Neptus, DUNE and the IMC communications protocol. Neptus is a user interface used for mission control and mission planning. It communicates with the onboard environment DUNE by the use of IMC messages.(Laboratório de Sistemas e Tecnologias Subaquáticas, 2013). DUNE is an interface to all other peripherals of the embedded computer on which DUNE is sited.

The LSTS tool-chain works for all Autonomous Underwater Vehicles (AUV), Unmanned Aerial Vehicles (UAV), Autonomous Surface Vehicles (ASV) and Remotely Operated Vehicles (ROV). The tool-chain is independent of the operating system it is used on (Pinto et al., 2012), but prefers to be used on Linux. The LSTS tool-chain is open-source technology available on the distributed revision control and source code management system Git (Wikipedia, 2014c). That means that it can not only be used in a project, but also modified to fit the project better.

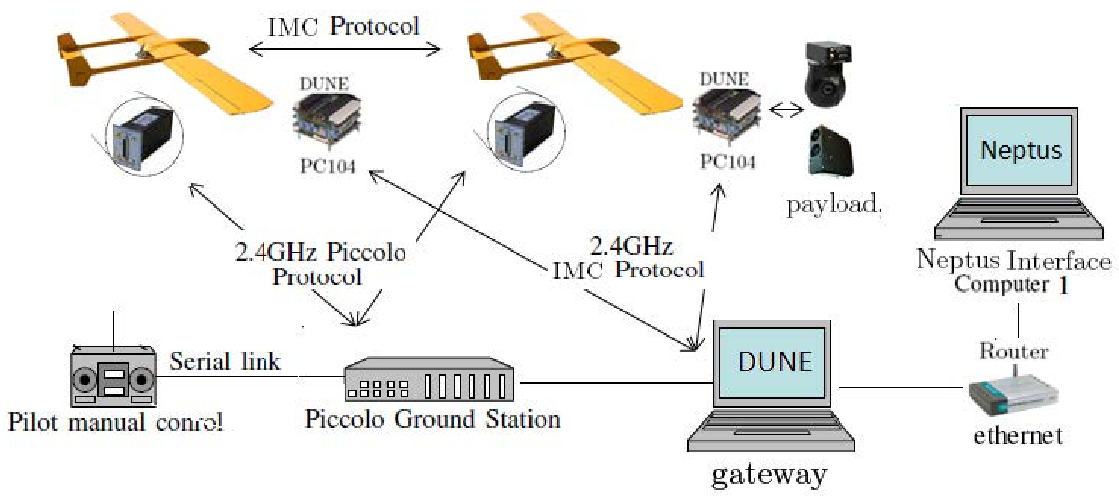


Figure 3.1: How the LSTS toolchain is supposed to work together with the Cloud Cap Technology (source: (Oliveira et al., 2011))

3.1.1 DUNE

The DUNE Unified Navigation Environment is a software that runs on-board the vehicle on an embedded computer. DUNE is responsible for the coordination of the payload processes on the vehicle, and between the payload processes and the autopilot. It consists of several tasks, each of which controls one element onboard the vehicle. The tasks communicate with each other using IMC messages, which transfer data packages between a sending task and a receiving one.

DUNE should control all peripheral units of this computer to ensure a clear chain of command and good cooperation. These peripherals are normally sensors, the autopilot and a wireless radio frequency sending device, which communicates with Neptus (Pinto et al., 2012). The control of peripherals use various protocols, to which DUNE can translate messages by including the correct task.

When DUNE is running, it takes as input an *.ini*-file. This file includes all of the tasks that should be used. The tasks could be compared to a pantry full of available ingredients, while the *.ini*-files are receipts that use these ingredients. It is possible to say: I want to bake a chocolate cake! and then the correct ingredients are used. It is the same with the *.ini*-file: If DUNE is run with lauv-seacon as argument, the correct tasks are used.

3.1.2 Neptus

The Neptus Command and Control Software is the ground station of the tool-chain. It contains a human interface and communicates with the DUNE units in the network using the IMC protocol. The human interface offers several functionalities, like mission planning, simulation, monitoring and replay. Neptus sends a mission plan via IMC messages to the DUNE nodes in the network, and receives logging information in return.

3.1.3 IMC

The Inter-Module Communication Protocol is used both between the different DUNE modules as well as between the vehicle and the ground station. The protocol consists of a large number of message definitions, and the protocol can be extended by adding new message definitions. DUNE can be an interface to other protocols like TCP or UDP. Then the IMC messages, that are sent from one DUNE task, are translated in a translation task and sent on to the correct peripheral on the hardware. Corresponding, the incoming messages are translated to IMC messages. IMC messages are available for all connected tasks, not restricted by physical limits, as long as the messages are sent between the physical units with some communication link, including the translating task.

3.2 Pandaboard

The Pandaboard is a general purpose microprocessor board. The board in use is a Pandaboard ES with an OMAP4460 processor using dual-core ARM Cortex-A9 architecture (Pandaboard.org, 2014). It communicates with the host computer through a serial port using the RS232 protocol or through Ethernet (see figure 3.2). It is also possible to connect a keyboard, a mouse and a screen to the Pandaboard and use it as a regular personal computer. Pandaboards have room for an SD Memory Card, where an operating system can be kept. The Pandaboard in this project has a Linux operating system called Wheezy on its 32 GB secure digital (SD) memory card.

The Pandaboard has several ports enabling a lot of peripherals to connect to it. They can be investigated in Figure 3.2. The Pandaboard requires a 5V power supply, preferably with 4A available (OMAPpedia, 2012). However, it is possible to power it with a battery, as long as it can provide the required voltage and manage current peaks of 1.2 A.

3.3 Penguin B

The UAV in use is a Penguin B from UAV Factory, as shown in figure 3.3. It has a propeller behind the wings, a payload module that is mounted beneath the body of the air plane, two tailboom assemblies attached to the wing and a tail joint attached to these, see Appendix A for drawings. The mass of the Penguin B is 10 kg without fuel or payload, and it can take a payload that, included the fuel, can be up to 10 kg (UAV Factory, 2014).

Although it is able to take off on a runway with 30 m run, it can also take off from a catapult. That is the take off method that will be used in this project. Once it is in the air, it may stay flying for at least 20 hours and has a cruise speed of 22 m/s and a max level speed of 36 m/s. It is petrol-powered and generates 80 W power onboard, with which it can power the payload. (UAV

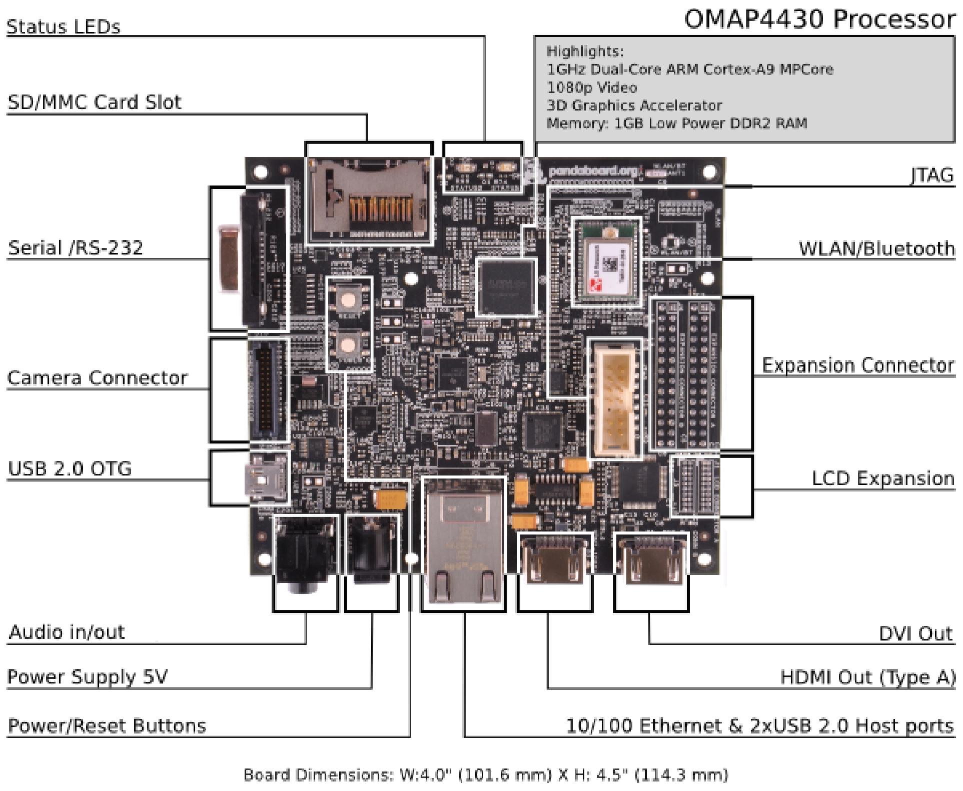


Figure 3.2: Pandaboard setup (source: Pandaboard.org (2014))

Factory, 2014).

The Penguin B has a removable payload mount that forms the bottom of the nose fuselage. The gross length of the payload mount is 42 cm and the net length is 34 cm. Drawings of the payload mount are attached in Appendix A.

The Penguin B is a fixed-wing UAV, as opposed to a rotorcraft design. Its benefits are its long reach and the ability to carry a large payload.

3.4 Piccolo SL Autopilot and Cloud Cap Technology

Piccolo SL

Piccolo SL is an autopilot from Cloud Cap Technology. It has 14 configurable general purpose input and output (GPIO) lines, a controller area network (CAN) interface, three RS232 payload interfaces and several radio options for wireless communication. It has GPS sensors that makes it able to measure its position. It supports waypoint navigation with up to 100 waypoints saved in the autopilot as well as turning rate input. Its mass is 110 grams and it takes 4 W power. The Piccolo SL supports both software-in-the-loop testing and hardware-in-the-loop testing. The

Figure 3.3: Penguin B from UAV Factory (source: UAV Factory (2014))

autopilot is placed on the UAV, and counts as part of the payload (Cloud Cap Technology, 2014b).

Piccolo Ground Station

To communicate with the Piccolo autopilot, a Piccolo ground station is needed, see Figure 3.4. This device receives the wireless signals that the autopilot transmits, and sends back directions to the autopilot. It is connected to a computer with the Piccolo Command Centre that provides a user interface (Cloud Cap Technology, 2014a).

Piccolo Command Center

This is the user interface to the Piccolo Autopilot. With Piccolo Command Center, the user may configure the autopilot, define missions, watch the vehicle track its route and save the data of the mission. The flight can be directed statically, which means that the complete voyage is pre-programmed, or dynamically, which means that the Piccolo Command Center can send way-points or heading directions to the Piccolo SL during the voyage. The Piccolo Command Center is also used for hardware-in-the-loop or software-in-the-loop simulation, which is treated like any other mission. It also contains pressure sensors and inertial sensors (Cloud Cap Technology, 2014c).

3.5 Equipment from Ubiquity Networks

Rocket M5

The Rocket M5 is a radio that transmits and receives data at high speed and over a wide range. The Rocket M5 in this project will be used as a radio station. The Rocket M5 is part of the Ubiquity Networks *airMAX* series, which use the multiple-input-multiple-output (MIMO) time division multiple access (TDMA) protocol. The different Rocket M models have different operating

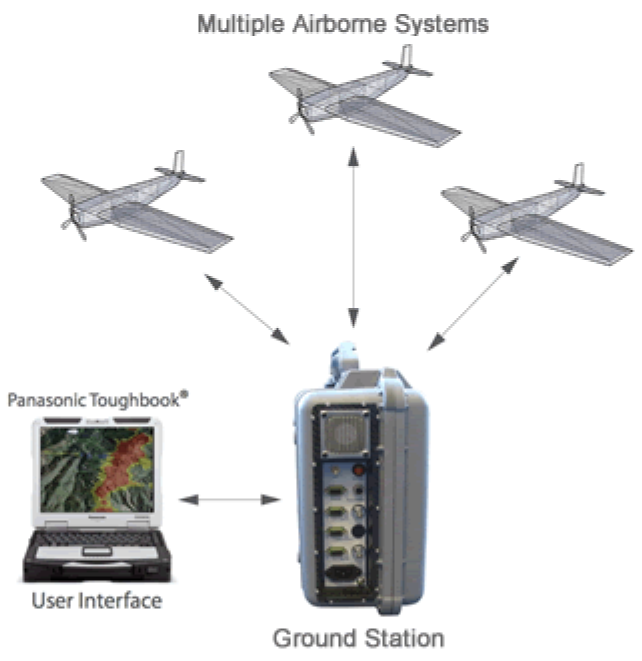


Figure 3.4: The Piccolo ground station communicates with the autopilots onboard the unmanned vehicles using radios, and with the user interface either wired or wireless. (source: Cloud Cap Technology (2014a)).

frequencies, and the Rocket M5 has an operating frequency of 5470-5825 MHz with an output power of 27 dBm. The Rocket M5 has a user interface can be reached from any internet browser by an initial static internet protocol (IP) address, and the settings of the Rocket M5 can be adjusted in the browser.

The Rocket M5 together with another device from the *airMAX* series act as an extension of the local area network (LAN), creating a wireless link. The Rocket M5 is powered by POE - Power over Ethernet. More detailed specifications can be found on Ubiquity Networks (2014)

NanoStation M5

The NanoStation M5 is another part of the *airMAX* series by Ubiquity. Like the Rocket M5, it has an output power of 27 dBm and 5 GHz transmitting and receiving frequency. The NanoStation M5 is also a radio node, and the NanoStation M5 in this project will be used as an access point. In contrast to a station, an access point in a wireless network connects many stations as an extension to the network. The NanoStation M5 and the Rocket M5 used in this project will be used in a bridge, extending the Ethernet connection. A sketch that illustrates how the Nanostation M5 can communicate with the Rocket M5 is shown in Figure 3.5. The figure shows how the radios make the upper computer an extension of the LAN, and the two computers can communicate over the internet just like they would if both were connected to the LAN. More detailed specifications can be found on Ubiquity Networks (2014)

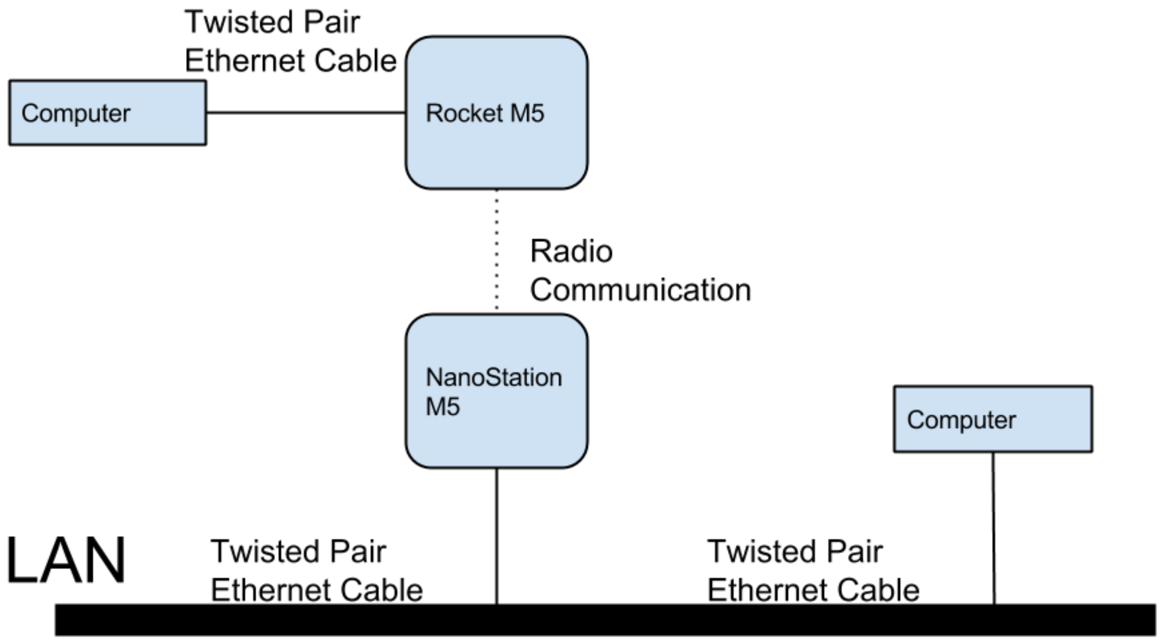


Figure 3.5: The upper computer is connected to the Rocket M5 with a twisted pair Ethernet cable. The Rocket M5 communicates with the NanoStation M5 using radio communication and the NanoStation is connected to the LAN with a twisted pair Ethernet Cable. The lower computer is also connected to the LAN.

Chapter 4

Precision Air Drop Theory

Precision drop of an object from an airplane, either unmanned or manned, is interesting in many fields of research. The topic has evolved greatly since the first times bombs were dropped from the air by Austrians in 1849 (HistoryOrb, 2014) and precision air drop is now an interesting subject for civilians as well as for the military. It is therefore a lot of information available on both methods to deliver the payload safely to the ground and methods to get to the deployment point effectively. This chapter is concerned with some of the theory that exists on these disciplines and with the theory behind the physics of a throw. The last section gives an alternative way of accepting waypoints in automatic tracking. Although the calculation of an optimal release point for the sensor that should be dropped, as well as the calculation of an optimal path to the point of deployment, are areas belonging to Simen Fuglaas' Master Thesis, theory on the subjects are presented here. However, the theory is superficial and further details can be found in Simen Fuglaas' Master Thesis.

4.1 A Simple Throw

The classical ballistics example is the simplified throw. The forces involved in a throw of a particle without any air resistance are investigated and unknown variables are calculated.

Given a particle with mass m and initial velocity v_0. This particle is released at the height h. This is illustrated in Figure 4.1. Assuming no air resistance and only the gravity g influencing the particle, then Netwon's Second Law will say that:

$$\sum \vec{F} = m \times \vec{a} \tag{4.1}$$

$$\sum \vec{F} = \vec{g} = m \times \vec{a} \tag{4.2}$$

$$\vec{a} = \frac{\vec{g}}{m} \tag{4.3}$$

Decomposing Equation 4.3 gives:

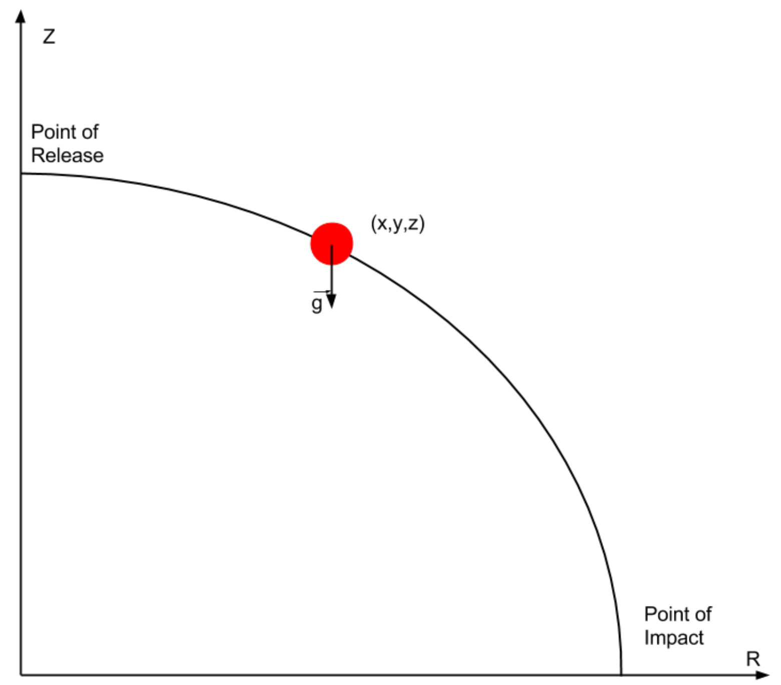


Figure 4.1: A simple throw without air resistance.

$$a_x = \frac{g_x}{m} \tag{4.4}$$

$$a_y = \frac{g_y}{m} \tag{4.5}$$

$$a_z = \frac{g_z}{m} \tag{4.6}$$

$$g_x = g_y = 0 \rightarrow a_x = a_y = 0, g_z = -g \tag{4.7}$$

Given a constant initial velocity $V_0(x, y, z) = (v_x, v_y, v_z)$. Integrating Equations 4.6 and 4.7 twice then gives the positions $P(x, y, z)$:

$$\int a_z = v_z = \int \frac{g_z}{m} = \frac{g_z}{m} t + v_{0z} \tag{4.8}$$

$$\int v_z = z = \int (\frac{-g}{m} t + v_{0z}) = \frac{1}{2} \frac{-g}{m} t^2 + v_{0z} t + z_0 \tag{4.9}$$

$$\int a_x = v_x = v_{0x} \tag{4.10}$$

$$\int v_x = x = v_{0x} t + x_0 \tag{4.11}$$

$$\int a_y = v_y = v_{0y} \tag{4.12}$$

$$\int v_y = y = v_{0y}t + y_0 \tag{4.13}$$

Which gives:

$$P(x,y,z) = (v_{0x}t + x_0, v_{0y}t + y_0, -\frac{g}{2m}t^2 + v_{0z}t + z_0) \tag{4.14}$$

Given $v_{0z} = 0$, Equation 4.14 gives:

$$P(x,y,z) = (v_{0x}t + x_0, v_{0y}t + y_0, z_0 - \frac{g}{2m}t^2) \tag{4.15}$$

At $t = 0$, $P_0(x,y,z) = (x_0, y_0, z_0)$. At the time of impact, $t = t_i$, $P_i(x,y,z) = (x_i, y_i, z_i)$. To find the release position of the particle that makes it fall a given point of impact, the point of impact is set to $P_i(x,y,z) = (0,0,0)$. Given a release height, $z_0 = h$, then:

$$v_{0x}t_i + x_0 = x_i = 0 \rightarrow x_0 = -t_i v_{0x} \tag{4.16}$$

$$v_{0y}t_i + y_0 = y_i = 0 \rightarrow y_0 = -t_i v_{0y} \tag{4.17}$$

$$h - \frac{g}{2m}{t_i}^2 = z_i = 0 \rightarrow t_i = \pm\sqrt{\frac{2mh}{g}} \tag{4.18}$$

As the time is always positive, that means:

$$x_0 = -t_i v_{0x} = -\sqrt{\frac{2mh}{g}}v_{0x} \tag{4.19}$$

$$y_0 = -t_i v_{0y} = -\sqrt{\frac{2mh}{g}}v_{0x} \tag{4.20}$$

If he size of V_0 is constant, but the direction is arbitrary, then the velocity can be written with polar coordinates, using a size and an angle. The size is given and the angle can change. We can then easy see that x_0 and y_0 can take any points of a circle around the point of impact with a radius given by Equation 4.23, where the velocity is directed towards the point of impact (see Figure 4.2).

$$|(x_0, y_0)| = \sqrt{(-\sqrt{\frac{2mh}{g}}v_{0x})^2 + (-\sqrt{\frac{2mh}{g}}v_{0x})^2} \tag{4.21}$$

$$= \sqrt{\frac{2mh}{g}v_{0x}^2 + \frac{2mh}{g}v_{0y}^2} \tag{4.22}$$

$$= \sqrt{\frac{2mh}{g}}\sqrt{v_{0x}^2 + v_{0y}^2} = \sqrt{\frac{2mh}{g}}|V_0| \tag{4.23}$$

The decomposed velocity vectors are shown in Figure 4.3.

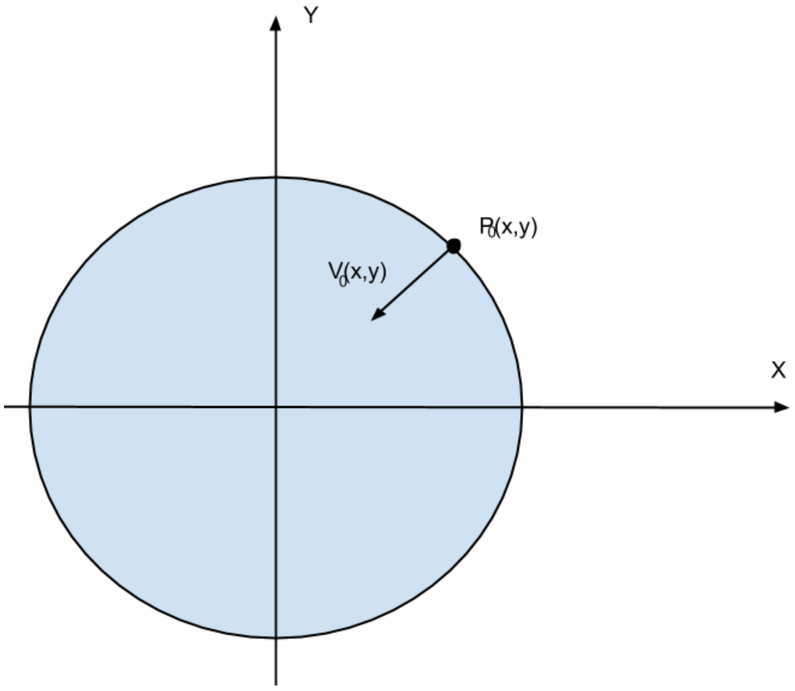


Figure 4.2: The release point and the velocity vector at the point of release, where the point of impact is the origin.

The conclusion is that to hit a point $P_i(x, y, z)$ with a particle with mass m from the height h, the particle can be released with a velocity V_0 pointing towards the point of release, when the distance in the radial direction is given in 4.24.

$$r = \sqrt{\frac{2mh}{g}}|V_0|. \tag{4.24}$$

In Cartesian coordinates the point of release is:

$$P_0(x, y, z) = (-\sqrt{\frac{2mh}{g}}v_{0x}, -\sqrt{\frac{2mh}{g}}v_{0y}, h) \tag{4.25}$$

4.2 A Throw with Air Resistance and Wind

Given a particle with mass m and initial velocity v_0. The particle is released at the height h, as illustrated in Figure 4.1. Assuming air resistance and wind influencing the particle in addition to the gravity. Then Equation 4.2 will be adjusted to Equation 4.26:

$$\sum \vec{F} = \vec{g} + \vec{F_w} + \vec{F_a} = m \times \vec{a} \tag{4.26}$$

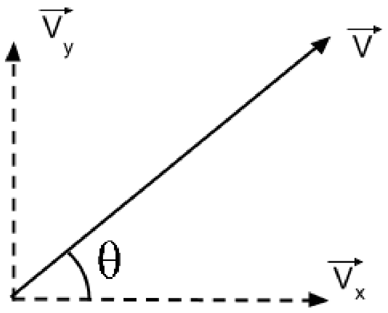


Figure 4.3: The decomposition of a velocity vector in x- and y direction.

Where $\vec{F}_w$ is the wind force and $\vec{F}_a$ is the force of the air resistance. The wind force will either reduce or extend the distance from the point of release to the point of impact. Approaching the point of release against or with the wind will be the two simplest cases. As Figure 4.4 shows that if the particle is released against the wind, then it will have a lot lower velocity than if it is released with the wind. The distance from the point of release to the point of impact will therefore be greater than the solution in Equation 4.25. The air resistance will reduce the distance from the point of release to the point of impact.

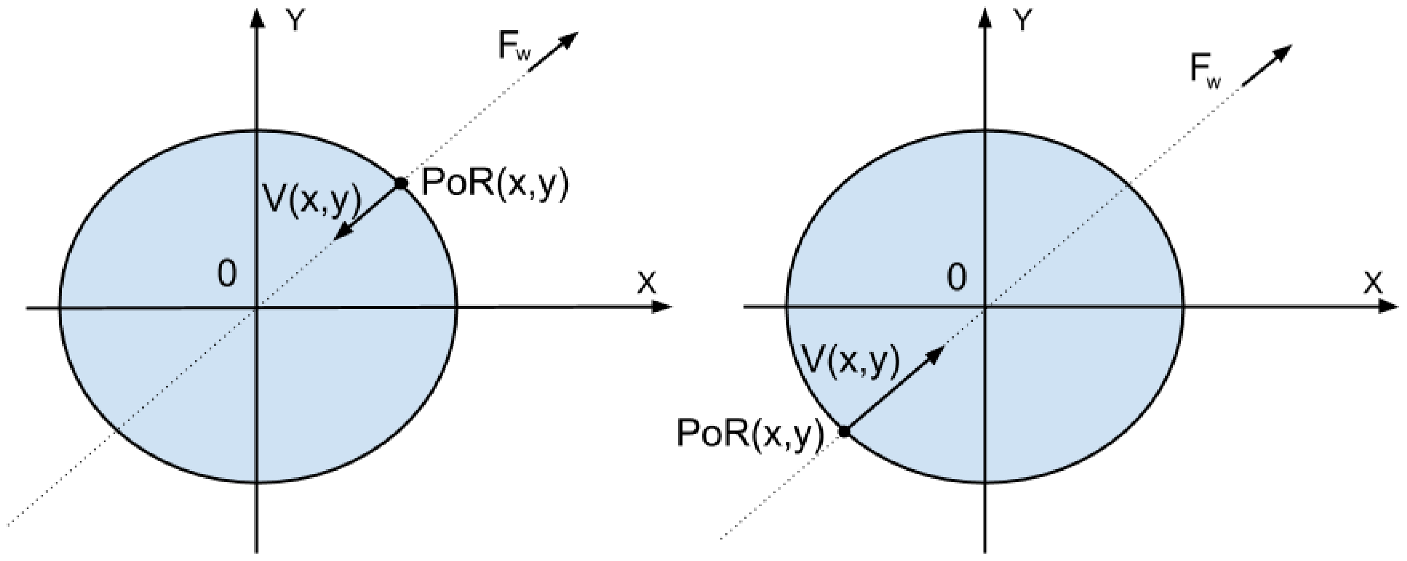


Figure 4.4: The particle's velocity after the release if it is released against or with the wind.

4.3 Dubins Path

In this Master Thesis, a directed point means a position given with a coordinate, where the approaching angle to the point is given. Therefore, in three dimensions, a position P is given with the position vector $\vec{P} = [x, y, z]$ as well as with the velocity vector $\vec{V} = [u, v, w]$, or corresponding using other coordinates. In 1957, Lester Eli Dubins showed that the time-optimal curve between two directed points in a two dimensional plane consists of only two circles of and a straight line that is a tangent to both lines (Dubins, 1957; Beard and McLain, 2012). The problem that this solution solves is showed in Figure 4.5. His solution to this problem is called the Dubins Path.

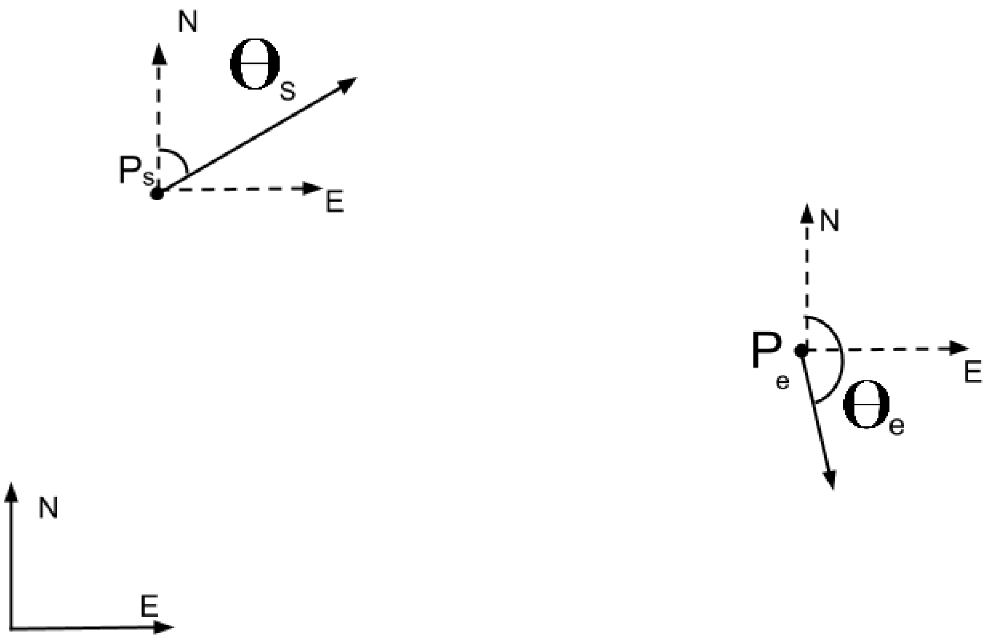


Figure 4.5: The problem that is solved with Dubins Path: Create an optimal trajectory from point P_s, which has the heading θ_s, to point P_e, which has the heading θ_e.

NED-frame

The word NED-frame is derived from the abbreviations of the words north, east and down. It is a local geodetic coordinate system that is similar to the Cartesian coordinate system, where the origin is a point on the earth sphere. Within a given and not too large distance of the origin, the xy-plane of the NED-frame is approximately equal to the surface of the earth, making a tangent plane to the earth on the point of origin of the NED-frame. The x-axis of the NED-framepoints towards north, the y-axis points towards east and the z-axis points towards the center of the earth (Wikipedia, 2014b).

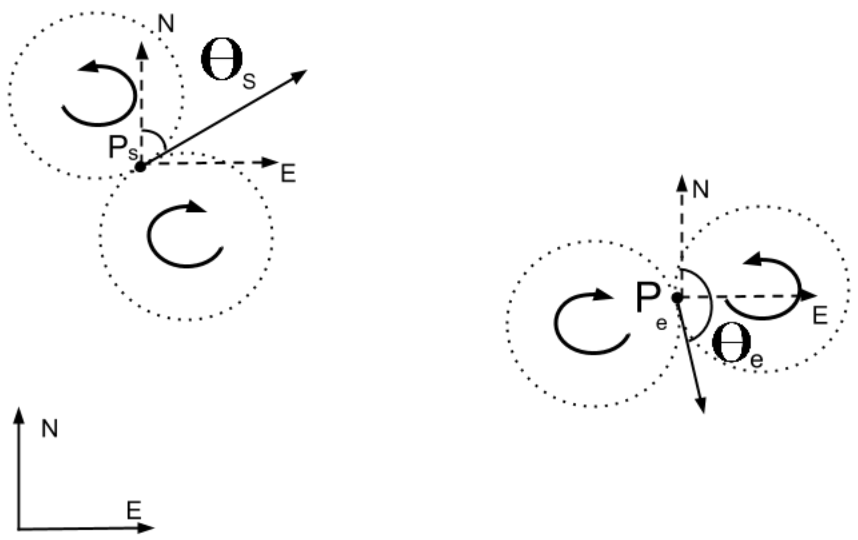


Figure 4.6: The problem that Dubins Path solves: The shortest curve from one directed point to another directed point.

Assuming a plane on a NED-frame of a given height. From a starting position P_s with a velocity with the heading θ_s relative to the north axis of the NED-frame, a UAV should follow a path that ends up in P_e with velocity heading θ_e relative to the north axis of the NED-frame, see Figure 4.5. The UAV has a minimum turning radius R, which is the radius of the smallest circle the UAV can make. The principle of Dubins Path is that the UAV starts by flying in a circle until it reaches the straight line that touches both that circle and the circle formed by the ending point. The ending point forms a circle because a particle with position P and velocity V is part of two circles, one on each side.

As Figure 4.6 shows, a particle that has the initial starting point P_s and velocity θ_s can be part of two turning circles, one clockwise to the right of the starting position and one counter-clockwise to the left of the starting position. The end point with the end velocity heading can also be part of two circles: One clockwise on the right hand and one counter-clockwise on the left hand. This means that there are four circles, one right-handed and one left-handed for both start and end position. They have centres definer by:

$$c_r = \mathbf{p} + R(cos(\theta + \frac{\pi}{2}, sin(\theta + \frac{\pi}{2}), 0)^\top \tag{4.27}$$

$$c_l = \mathbf{p} + R(cos(\theta - \frac{\pi}{2}, sin(\theta - \frac{\pi}{2}), 0)^\top \tag{4.28}$$

That means that there are four combinations that fulfil the circle-straight-line-circle conditions, all of which are shown in Figure 4.7. Dubins path is the shortest path out of the four possible ones (Beard and McLain, 2012). According to (Beard and McLain, 2012), the total path length is therefore:

Case 1: Right-hand circle, straight line, righ-hand-circle

ϑ is the angle formed between the north axis and the straight line between the center of the chosen circles.

$$RSR : L_1 = \|c_{rs} - c_{re}\| + R\left\langle 2\pi + \left\langle \vartheta - \frac{\pi}{2}\right\rangle - \left\langle \theta_s - \frac{\pi}{2}\right\rangle\right\rangle + R\left\langle 2\pi + \left\langle \theta_e - \frac{\pi}{2}\right\rangle - \left\langle \vartheta - \frac{\pi}{2}\right\rangle\right\rangle \tag{4.29}$$

Case 2: Right-hand circle, straight line, left-hand-circle

ϑ is the angle formed between the north axis and the straight line between the center of the chosen circles, $\ell = \|c_{le} - c_{rs}\|$ and

$$\vartheta_2 = \vartheta - \frac{\pi}{2} + sin^{-1}\left(\frac{2R}{\ell}\right) \tag{4.30}$$

$$RSL : L_2 = \sqrt{\ell^2 - 4R^2} + R\left\langle 2\pi + \langle \vartheta_2 \rangle - \left\langle \theta_s - \frac{\pi}{2}\right\rangle\right\rangle + R\left\langle 2\pi + \langle \vartheta_2 + \pi\rangle - \left\langle \theta_s + \frac{\pi}{2}\right\rangle\right\rangle \tag{4.31}$$

Case 3: Left-hand circle, straight line, right-hand-circle

ϑ is the angle formed between the north axis and the straight line between the center of the chosen circles, $\ell = \|c_{re} - c_{ls}\|$ and

$$\vartheta_2 = cos^{-1}\frac{2R}{\ell} \tag{4.32}$$

$$LSR : L_3 = \sqrt{\ell^2 - 4R^2} + R\left\langle 2\pi + \left\langle \theta_s + \frac{\pi}{2}\right\rangle - \langle \vartheta + \vartheta_2\rangle\right\rangle + R\left\langle 2\pi + \left\langle \theta_s - \frac{\pi}{2}\right\rangle - \langle \vartheta + \vartheta_2 - \pi\rangle\right\rangle \tag{4.33}$$

Case 4: Left-hand circle, straight line, right-hand-circle

ϑ is the angle formed between the north axis and the straight line between the center of the chosen circles.

$$LSL : L_4 = \|c_{ls} - c_{le}\| + R\left\langle 2\pi + \left\langle \vartheta + \frac{\pi}{2}\right\rangle - \left\langle \theta_s + \frac{\pi}{2}\right\rangle\right\rangle + R\left\langle 2\pi + \left\langle \theta_e + \frac{\pi}{2}\right\rangle - \left\langle \vartheta + \frac{\pi}{2}\right\rangle\right\rangle \tag{4.34}$$

The case with the shortest total path length is Dubins Path and therefore the time-optimal curve from one directed point in the plane to another directed point in that same plane (Beard and McLain, 2012).

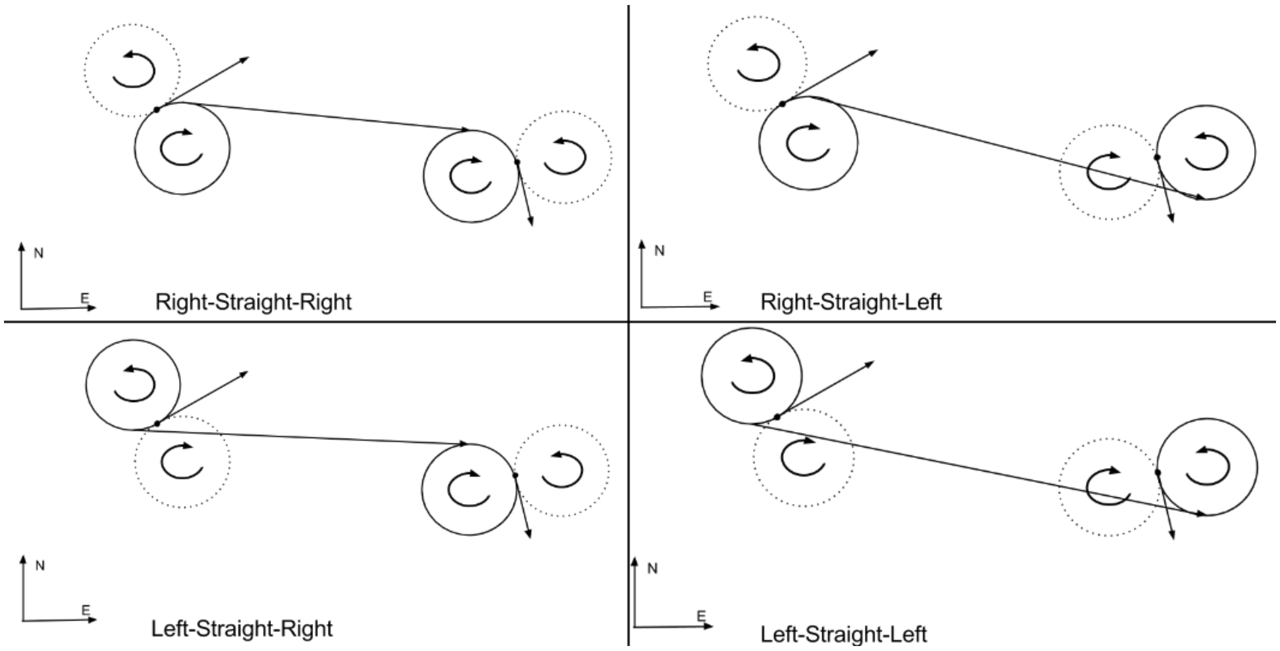


Figure 4.7: The four possible suggestions for a Dubins Paths with the initial P_s,θ_s and P_e,θ_e. The two lower paths indicate that the UAV has to fly almost a whole circle before starting on the straight line.

.

4.4 Half Planes used to Accept Waypoints

A waypoint is a position given with a geographic coordinate system, meaning that the point consists of a latitude, a longitude and an altitude. To accept a waypoint as entered, a sphere around the waypoint can create an acceptance area. When the UAV's GPS position says that the UAV is within this sphere, the waypoint is accepted. The radius forming the acceptance sphere is called an acceptance radius.

As an alternative to an acceptance radius around the tracked waypoint, half planes can be used. When the line that defines the change of half planes is crossed, the waypoint is accepted. The theory names the waypoint that should be accepted for w_i, the previous waypoint is w_{i-1} and the next waypoint is w_{i+1}. These three waypoints form a plane, and two half planes are formed if that plane is divided by a line. The point of using half planes to accept a waypoint is to say that as long as the UAV is on the first half plane, it has not yet reached the waypoint. As soon as it crosses the line to the next half plane, the waypoint is accepted.

Given a point $\mathbf{r} \in \Re^3$ and a normal vector $\mathbf{n} \in \Re^3$, a half plane is defined as in Equation 4.35.

$$H(\mathbf{r},\mathbf{n}) \triangleq \{\mathbf{p} \in \Re^3 : (\mathbf{p}-\mathbf{r})^\top \mathbf{n} \geq 0\} \tag{4.35}$$

Waypoints i is written as w_i, and the unit vector from w_i to w_{i+1} is written as Equation 4.36.

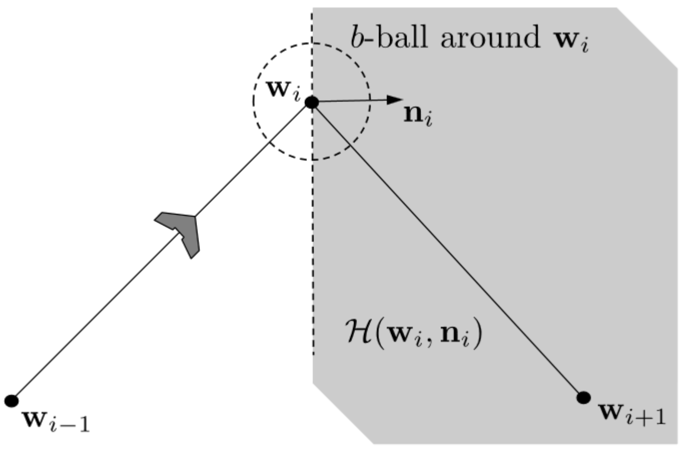

b-ball around $\mathbf{w}_i$
$\mathbf{w}_i$
$\mathbf{n}_i$
$\mathcal{H}(\mathbf{w}_i, \mathbf{n}_i)$
$\mathbf{w}_{i-1}$
$\mathbf{w}_{i+1}$

Figure 4.8: Illustration showing the half plane that indicates when the UAV has reached waypoint w_i and can continue to track the next. Source: (Beard and McLain, 2012)
.

$$\mathbf{q}_i \triangleq \frac{\mathbf{w}_[i+1] - \mathbf{w}_i}{\|\mathbf{w}_{i+1} - \mathbf{w}_i\|} \tag{4.36}$$

The unit normal vector of the halfplane in 3 dimensions is given by Equation 4.37:

$$\mathbf{n}_i \triangleq \frac{\mathbf{q}_[i+1] + \mathbf{q}_i}{\|\mathbf{q}_{i+1} + \mathbf{q}_i\|} \tag{4.37}$$

The source for this whole Section is (Beard and McLain, 2012) and further information can be found here.

Part II

System Description, Planning and Implementation

Chapter 5

System Description

Summed up, the assignment of this Master Thesis is to drop a payload from a UAV precisely on a given target from a predetermined height. This is a large and complicated task that must be described thoroughly to get a proper overview of the situation. This chapter aims to describe the system and create a connection between the tasks that are described later on in the Master Thesis.

5.1 The Work Flow of the System

The line of action in this system is illustrated in Figure 5.1. The given start condition is that the UAV loiters around the point of impact, but on a given height, to measure the wind. Until this point, the UAV pilot has controlled the UAV. This loitering has a radius that is at least as big as the minimum turn radius of the UAV, but probably bigger. After the wind is correctly measured, the point of release and the angle, from which the UAV must approach the point of release (PoR), is calculated. This calculation takes the direction and the strength of the wind into consideration. The next task is to calculate a trajectory that takes the UAV from the current position to the PoR. Here, it is important that the UAV reaches the PoR from the correct angle, that was calculated in the previous step. When the trajectory is calculated, it can be tracked by sending instructions to the autopilot. When the correct PoR is reached, the sensor must be released. Afterwards, the UAV can return to base or continue to loiter around the target position, ready to be controlled again by the UAV pilot.

5.2 Overview of the System Modules

The two superset modules of the system are the UAV and the ground station, communicating with each other using wireless communication (Figure 5.2). The UAV is a Penguin B, more closely described in Section 3.3, using a Piccolo autopilot to control the flight, see section 3.4. A payload mount forms the bottom of the UAV body, on which a Pandaboard (see 3.2), equipment for a drop mechanism, some electronics and a radio are placed. The radio in the Penguin B is a Ubiquiti Rocket MS from Ubiquiti Networks, see section 3.5. The drop mechanism is described

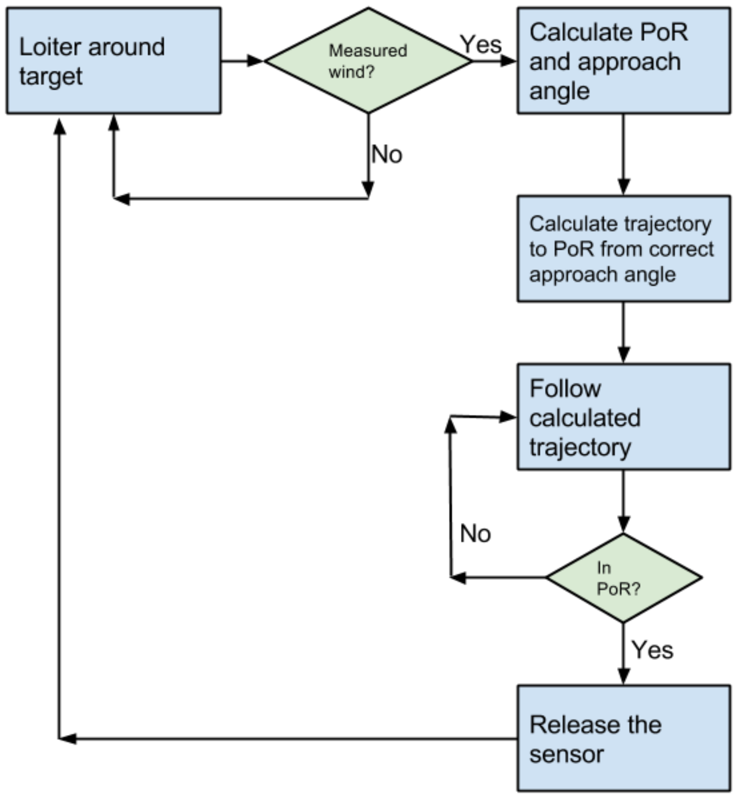

Loiter around target
Measured wind?
Yes
No
Calculate PoR and approach angle
Calculate trajectory to PoR from correct approach angle
Follow calculated trajectory
No
In PoR?
Yes
Release the sensor

Figure 5.1: The flow of the system used to release a sensor from a UAV onto a target. The loitering around the target happens on the same height as all other flight activity. Before the loitering, the UAV pilot is responsible for controlling the path of the UAV.

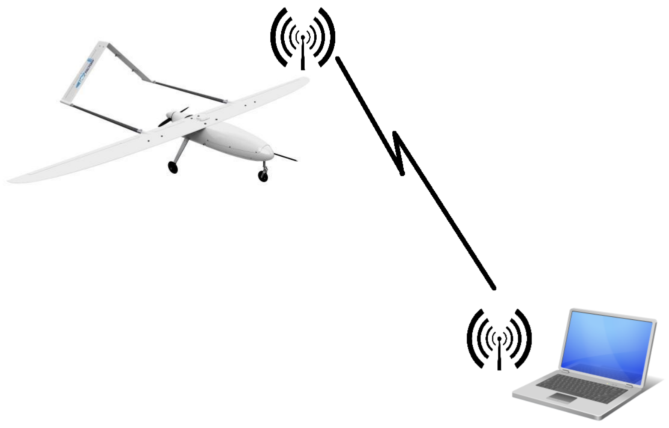


Figure 5.2: Superior System Overview. The UAV communicates with the ground station with radio communication.(sources: UAV Factory (2014), www.gettyicons.com)

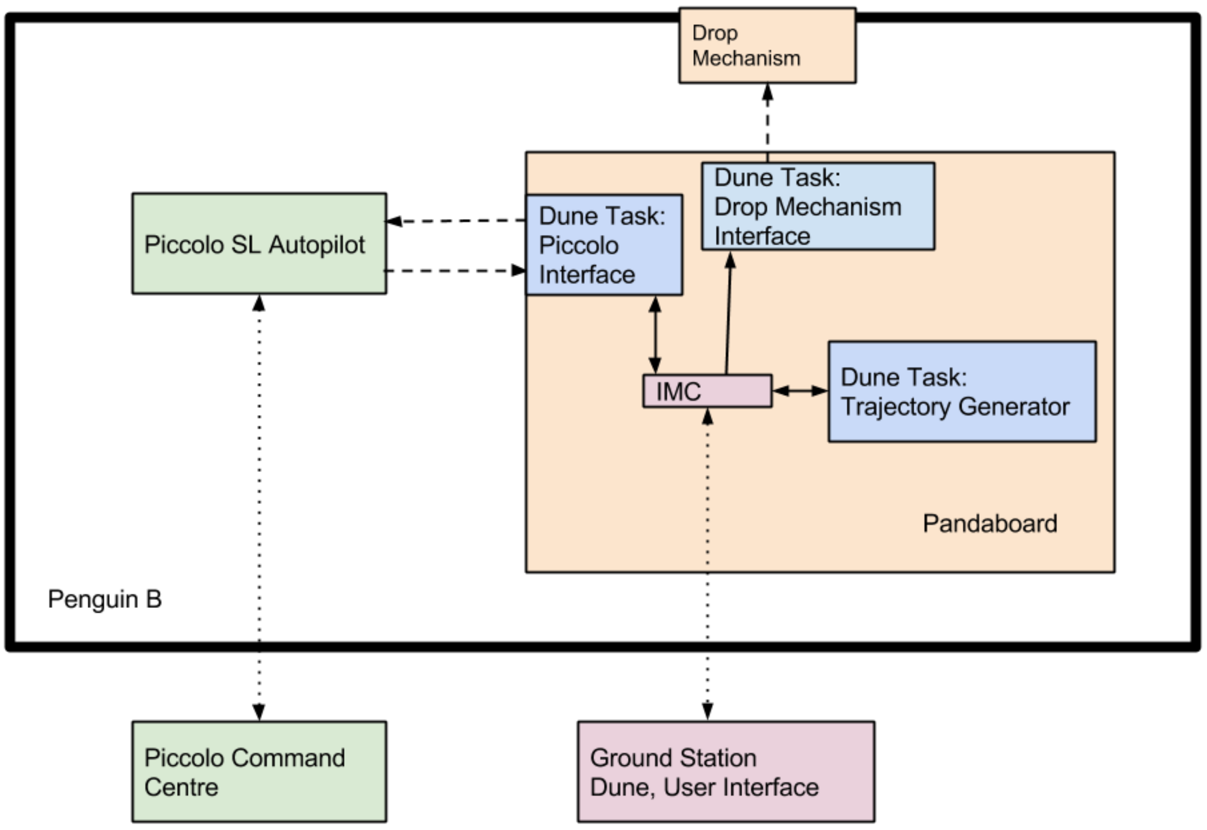

Drop Mechanism
Piccolo SL Autopilot
Dune Task: Drop Mechanism Interface
Dune Task: Piccolo Interface
IMC
Dune Task: Trajectory Generator
Pandaboard
Penguin B
Piccolo Command Centre
Ground Station Dune, User Interface

Figure 5.3: The system architecture of the payload. The green boxes are controlled by the UAV pilot, but the autopilot may receive directions via DUNE

in detail in Chapter 7.

The UAV communicates with the ground station over wireless communication, as anything else would be impossible when it is in the air. Piccolo has a radio link connecting it with Piccolo Command Centre on the ground, independent of the communication between the payload and the ground. This is controlled by a UAV pilot. The user interface is also on ground and consists of software written with DUNE that communicates with the payload using IMC messages wirelessly.

Onboard the UAV, the Pandaboard is placed on the payload mount. The software on the Pandaboard communicates with the ground station using the UDP protocol over radios. The software on the Pandaboard is also responsible for calculations of the point of release, calculations of an optimal path from the current position of the UAV to the point of release, control of the drop mechanism and interface to the Piccolo autopilot. This is shown in Figure 5.3.

Chapter 6

Trajectory and Drop Planning

A precision air drop implies that the payload to be deployed should end up as close to the target position as possible. Although there are uncertainties involved in this, it is possible to calculate a point of release by the use of mechanics equations. This will be done in Simen Fuglaas' Master Thesis and will therefore not be described here, but some of the theory involved is found in Sections 4.1 and 4.2.

As described in Chapter 5, the UAV is controlled by the UAV pilot until it reaches the point of impact, projected on the same height as the UAV is supposed to fly. This is probably close to the point of release, though not the same point. After measuring the wind, the Pandaboard calculates the optimal point of release (PoR) and the trajectory that leads it to this point from the correct angle. The minimum turning radius of the UAV must also be taken into consideration. From the UAV leaves this circle until it reaches the point of release (PoR), it must follow a trajectory. The shortest way between two points is a straight line, but as the UAV is restricted by technicalities such as minimum turning radius and initial heading, a straight line is not always an option. One method to solve this problem is elaborated in Section 4.3.

To release the payload from the UAV, there are several methods found in the literature for delivering a payload from an aerial vehicle. Many scientists have presented their experience with different solutions. Restrictions on the UAV as well as requirements for the delivery of the payload must be taken into consideration, and not all solutions are realistic in the time frame assigned to this master thesis.

This chapter is devoted to development and implementation of an algorithm for a trajectory from a starting point to the point of release, and to the choice of a method used to deploy the payload. Because the trajectory from a starting point to an end point depends on where the end point is, the calculation of a point of release is treated before the first. And since the payload deployment method is critical when it comes to calculating the point of release, that first step will be treated before the second.

6.1 Choice of Payload Deployment Method

The payload deployment methods found in the literature can be divided into two groups: Free falls and guided falls. The free fall group contains all methods that drops the payload directly from the UAV, and can either use parachutes or not. The guided falls use facilities to release the payload, for example the wire coil slide mentioned by Williams and Trivailo (2006) or the rotating wire coil, mentioned by Corke et al. (2004). These deployment methods are described in Section 2.2.

Free Fall without Parachute

This method is by far the simplest one. It requires nothing but a facility that disconnects the payload from the UAV, and the control of this facility. That means that there is less equipment that can be malfunctioning and that has to be maintained. Besides, it means that the calculations for a point of release are simple. The drawback is that there are no mechanisms implemented in the deployment method that prevent the sensor from being destroyed by the impact on the ground.

Free Fall with Parachute

This method has the simplicity of the free fall without parachute, but provides a mechanism that prevents the sensor from being destroyed by the fall, namely the parachute. The canopy decelerates the speed of the falling object by increasing the air resistance and thus decreases the shock when the object hits the ground. However, the increased air resistance means that the parachute is much more exposed to the wind, which can make the object drift far away from its point of impact.

Guided fall with a Spiralling Wire Slide

It is far more complex than both of the free falls, as it requires the wire to be released and a facility that releases the payload on the wire slide, in addition to anchoring of the wire to the ground or a heavy weight on the tip of the wire. The weight on the wire and the weight of the wire itself will moreover be restricted by the maximum payload weight on the Penguin B. Another challenge with this method is the anchoring to the ground for stability: It has to be done automatically as there are no human beings on the iceberg, ready to anchor a wire to get packets. An automatic wire anchoring mechanism then has to be developed, and the resulting amount of work would then probably exceed the workload available of this Master Thesis.

Guided Fall on a Wired Coil

Although this method is placed among the deployments that do not use free fall, the payload is dropped from the coil directly to the ground. Its main purpose is to enable several sensors to be dropped at given intervals from the same UAV, and is therefore not very relevant in this case.

Choice of Deployment Method

The spiralling wire slide is a time-consuming and complicated solution that requires a lot of weight, especially for a deployment from a great height, to be carried by the UAV. That makes it a unfit solution for this application. The free fall is a simple solution that is manageable in the course of a Master Thesis. However, there is no use in a sensor that is precisely dropped to the ground, if the impact breaks it. A parachute might solve that problem, but the effect it has on the wind forces is unforeseeable. Depending on the height where the payload is released, it can drift very far away. As precision is important, then parachutes are out of question. The only solution of the researched alternatives is the free fall without parachute, but with protection for the deployed sensor that prevents it from destruction.

This will be realised with a drop mechanism that releases the payload sensor from the body of the UAV, and this drop mechanism will be discussed in Chapter 7.

6.2 Development of an Algorithm for a Trajectory to the Point of Release

6.2.1 Determining the Initial Conditions

The requirement for the trajectory is that it should lead the UAV from the starting point to the point of release. The point of release depends on the wind and the approach direction of the UAV, which means that it is necessary to know the force and direction of the wind. This can be effectively measured by loitering around the point of impact. With its sensors, the Piccolo SL autopilot can measure the wind and communicate that to the control unit onboard the Penguin B. The wind will, as mentioned, influence the position of the point of release and the calculations for this point are easiest if the UAV comes up against the wind or flies with the wind. These two choices are approximately equivalent, but still, the first alternative is preferable: If the UAV flies against the wind, then the length between the point of release and the point of impact is confined, although the wind strength might change. If the UAV flies with the wind, then a change of the wind strength will blow the payload further away, helped by the initial velocity of the payload that is gained from the UAV. That means that the point of release is decided to be placed such that the UAV must fly against the wind, see Figure 6.1.

Now the trajectory requirements are as follows: A trajectory should be calculated from a start position, which is somewhere on the circle around the point of impact to measure the wind, to an end position, which is somewhere else. The end position will be along the opposite direction of the wind from the point of release, depending on the strength of the wind, the air resistance and the speed of the UAV. The UAV has a velocity when the trajectory is calculated, and the point of release should be approached from a correct angle. This must be considered when the trajectory is calculated. This is illustrated in Figure 6.2. Another requirement is that the minimum turn radius of the UAV should be respected - no turns should be planned with a smaller radius.

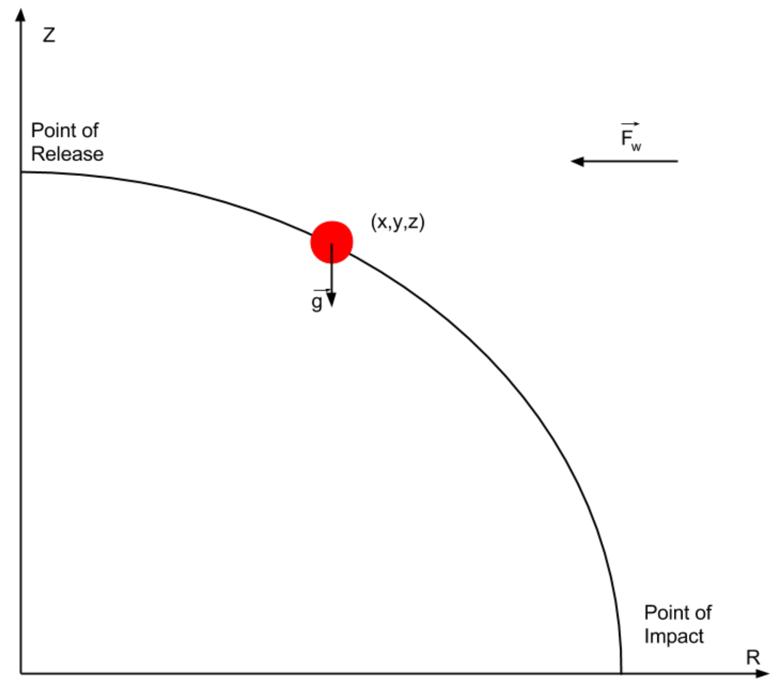


Figure 6.1: An illustration of the point of release, where $\vec{F_w} = -|F_w|\vec{R}$.

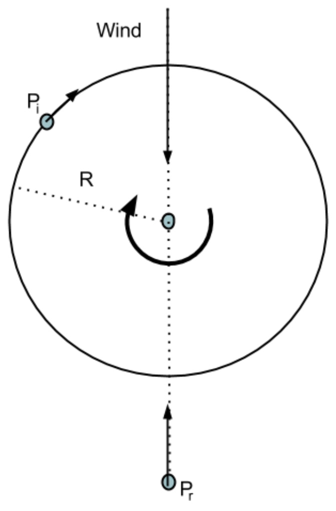


Figure 6.2: The problem to be addressed when creating a trajectory from an initial point to an end point. P_i is the initial position of the UAV, the arrow points its heading. P_r is the point of release, the arrow points against the wind. R is the minimum turn radius of the UAV.

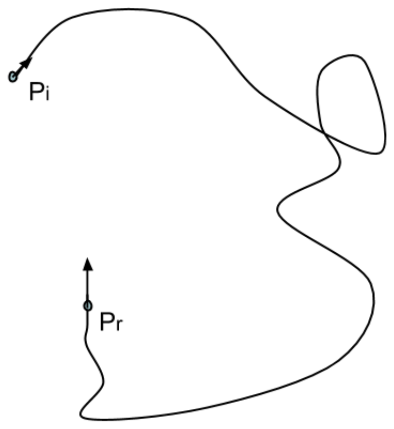


Figure 6.3: On out of many possible paths from one directed point to another.

A countless number of paths lead from the directed starting position to the directed ending position, one of which is shown in Figure 6.3. It is clearly an advantage to use a deterministic path. The natural choice is then to use the optimal path between the points, starting with a given direction and ending up with a given direction.

6.2.2 Choice of Navigation Method

The Penguin B uses a Piccolo SL autopilot to control its flights. The autopilot can accept flight directions in the form of waypoints and heading change. Waypoints consist of latitude, longitude and several variants of height, while heading means the angle the autopilot should fly with relative to the north axis of the NED frame. This was not experimented with before a method was chosen, as there was no appropriate test time: The testing happened in the last few weeks of the Master Thesis project, and there were not enough test times available. Besides, if the waypoint and heading experiments should be conducted before one was chosen, then the real testing would have happened much later. Therefore, the decision was taken based on experience other Master students had had with the system, and sensible discussion.

The preferred choice is to use heading, as that is more easily controlled with feedback. If a heading is given to the autopilot, then the measured heading received from the autopilot says how well the directions are followed. That is exact and easy to use control theory upon. With waypoints, that is different. If the autopilot receives a waypoint, it flies in that direction, but does not have to be exactly on the waypoint spot to accept it: If the UAV is within a given radius of the waypoint, it counts as if the UAV is in the waypoint. That creates uncertainty, if the waypoints are not tracked. It is possible to track the waypoint in a payload computer and be more exact than the autopilot.

Some other Master students, Espen Skjong and Stian Nundal, experienced that the Piccolo autopilot did not track a path well when it used heading instead of waypoints, as there are no good ways to measure the heading onboard the UAV, and therefore no possibilities for a feedback loop. That, in addition to the improved tracking of waypoints, led to the decision that waypoints should be used for navigation.

6.2.3 The Straight Line approach

This is a very simple approach, which does not give any optimal path. The UAV approaches the point of release (PoR) from a predetermined angle and starts by loitering around the point of impact on the given height above the ground.

A straight line is drawn through the PoR, parallel to and in the same direction as the wind. A point is chosen on the straight line, far away from the PoR in the direction of the wind. The path of the UAV goes from the starting point, through the point far away from the PoR on the straight line, and then through PoR against the wind. This is illustrated in Figure 6.4, where the trajectory is the dotted line.

However, no UAV is able to turn on the spot. A more realistic path is shown as a solid-drawn line in the figure, with an arch and lines drawn from the arch to the dotted line. The size of the arch on this line depends on the minimum turn radius of the UAV. To be sure to reach the PoR fram the correct angle, it is important that the *far away* point is, in fact, far away. If the distance were equal to two minimum turn radius of the UAV (R), then the worst case scenario is that the UAV would approach the PoR from the side, simply following the circle arc, with 90^o angle deviation from the ideal approach. For each extra turning radius added to the distance, the angle with which the UAV approaches the PoR, will be more similar to the ideal angle. This is however in a worst case scenario. In reality, the autopilot will aim to follow the dotted line and thus add a curvature to the straight lines shown in Figure 6.4. That makes approximately four times the minimum turning radius a reasonable choice for the distance, as that distance would allow the UAV to fly not one, but two full circles before reaching the PoR.

6.2.4 A Pseudocode Implementation of the Straight Line Approach

The Straight Line Approach will be implemented in the calculations tasks (see Section 8.2.1.2) in C++. The algorithm for following the Straight Line Approach is found in Table 11.2 in Chapter 11. The implementation of and calculations for the Straight Line Approach are conducted by Simen Fuglaas and this will be described in his Master Thesis. This solution is meant to be a temporary solution that solves the problem, but not in any optimal way, as the precision of the approaching angle on the point of release depends on the distance to the *far away*-point.

6.2.5 Using Dubins Path

The Straight Line approach gives a deterministic, though not very optimal solution to the trajectory generation problem. Dubins Path, described in Section 4.3, is an optimal path from one directed point to another. To find Dubins Path, four possible Dubins Paths must be evaluated and the shortest one chosen. As the UAV is already loitering when the path is calculated, this excludes the two paths that do not start with the same rotation as the loitering, but still, two

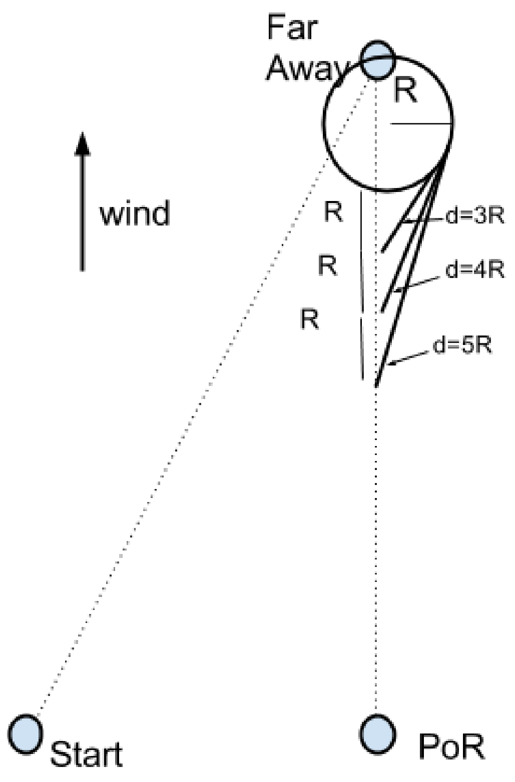


Figure 6.4: Straight Line Approach. The figure shows the starting point, the chosen point, PoR and an ideal path between the points drawn with dots.

paths must be calculated. To make the solution simpler, we do not demand a time optimal, but a deterministic trajectory from the directed starting position to the point of release. That means that one possible Dubins Path can be chosen from the beginning, without any heavy comparison calculations.

The point of release can be approached from two circles, see Figure 6.5. To simplify calculations, the same circle is chosen every time, namely the circle with the same rotation as the wind measurement circle is chosen. In Figure 6.5, this means the right circle. Then, the theory found in Section 4.3 can be used, creating a path built froms two circles and a straight line. This is shown in Figure 6.6.

Although the velocity vector of a particle following a circle is tangential to the circle, the acceleration is pointed towards the center of the circle, see Appendix B. This creates a centripetal force that influences the path of the dropped sensor and pushes it out of its course. In the calcu-

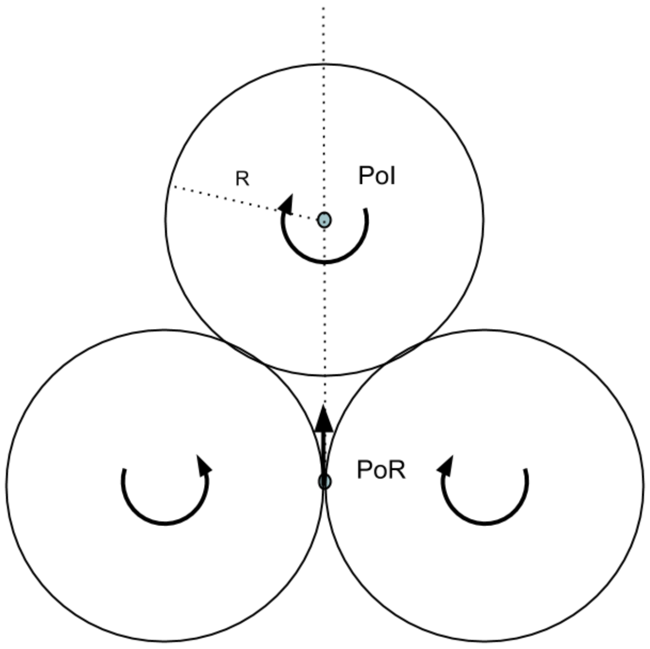


Figure 6.5: Choosing the correct circle for the Adapted Dubin's Path.

lations of the release point (see Section 4.2 a constant velocity has been assumed, which means no angular velocity either. The solution to this problem is to add a straight line to the trajectory before releasing the target. This can easily be arranged by moving the circle connected to the point of release, along a straight line from the point of impact to the point of release. This is shown in Figure 6.7

6.2.6 A Pseudocode Implementation of the Adapted Dubins Path

The Adapted Dubins Path will be implemented in the calculations tasks (see Section 8.2.1.2) in C++. The algorithm for following the Adapted Dubins Path is found in Table 11.3 in Chapter 11. The implementation of and calculations for the Adapted Dubins Path are conducted by Simen Fuglaas and this will be described in his Master Thesis.

6.3 Choice of Waypoint Acceptance Criteria

As described in Section 4.4, there are two possible ways to accept a waypoint: Either by using an acceptance radius and say that the waypoint has been entered by registering the position of the UAV within a given distance of the waypoint, or by the use of half planes. The efficiency of the two waypoint acceptance criteria is dependent on how good the communication between the Pandaboard and the autopilot is. If the autopilot sends the GPS position of the UAV to the

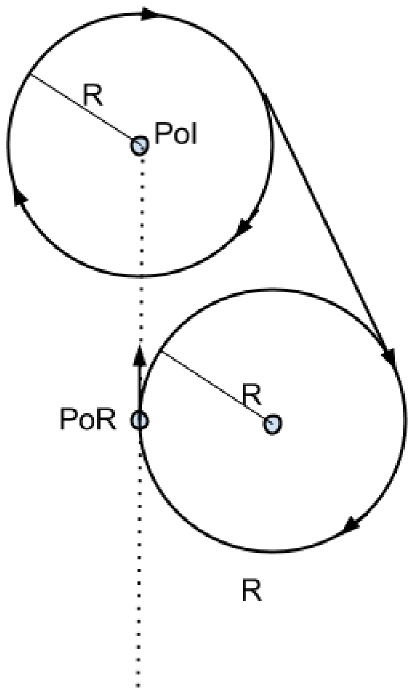


Figure 6.6: The chosen possible Dubins Path. PoI is the point of impact of the sensor, while PoR is the point of release. R represents the minimum turn radius of the UAV.

Pandaboard on a very high frequency, then the acceptance radius will accept waypoints as soon as it enters the acceptance circle. That way, the distance between the waypoint and the position of the UAV when it accepts the same waypoint, is approximately equal to the acceptance radius. That encourages a very small acceptance radius. The problem with that is if the Piccolo is somehow prevented from sending its position to the Pandaboard within that very small acceptance circle, then the waypoint will not be accepted at all, and the UAV has to fly back to try that waypoint again. If however a half plane were used, then the UAV would still be on the other side of that half plane when the Piccolo is again able to send its position to the Pandaboard, and the waypoint will always be accepted.

The trajectory tracking will be implemented with both acceptance radius functionality and with half plane functionality. That way, the two methods can be tested and it is possible to see which one should be used in future work on the system.

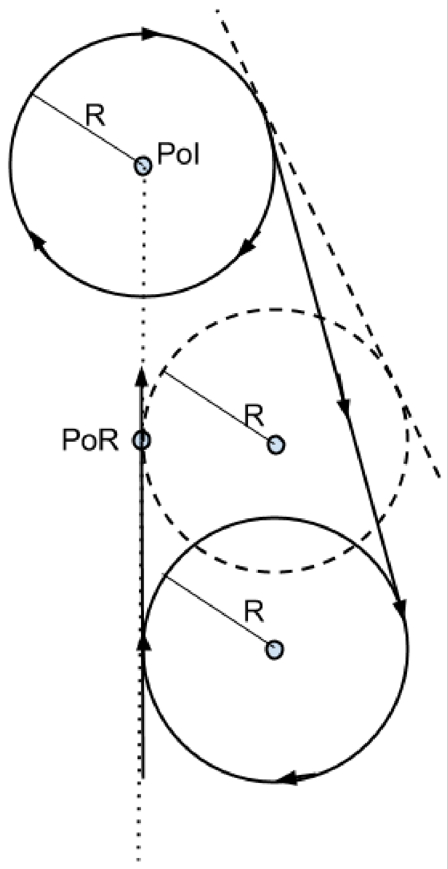


Figure 6.7: This Adapted Dubins Path gives constant velocity at PoR.

Chapter 7

Drop Mechanism

When the UAV has tracked the optimal trajectory correctly and reached the drop location, it must be able to release its payload. As discussed in Chapter 6, the sensor deployment method is to come against the wind to an optimal deployment point and disconnect the sensor from the UAV. As the UAV in use (see Section 3.3) does not come equipped with any drop mechanism, an external payload release mechanism will have to be attached to the air plane. The requirements for this payload release mechanism are:

1. It should drop the payload with a known and deterministic delay
2. The dropped payload should not end up it the propeller of the UAV (see figure 3.3).
3. It should not be too power consuming
4. It should not be too heavy
5. It should be reasonably easy to operate
6. It should be reusable

That the drop mechanism should be reusable means that if one part of the mechanism is connected to the sensor as it is released from the UAV, then this part should be easy to replace. This Chapter is devoted to the choice of a drop mechanism and to the implementation of that drop mechanism in the system.

7.1 Choice of Drop Mechanism

The drop mechanisms that were investigated are presented in Table 7.1. The EFLA405 Servoless Payload Release was purchased in the beginning of January 2014, the Quanum RTR Bomb system arrived in the middle of march 2014 and the Tinder Rocketry Peregrine was never ordered.

Name	**Weight**	**Voltage**	**Max Current**	**Signal**	**Length**	**Width**	**Height**
EFLA405 Servoless Payload Release	18 g	4.8-8.5 V	350 mA	PWM	79.0 mm	15.5 mm	27.0 mm
Quanum RTR Bomb System	103 g	4.8-8.5 V	350 mA	PWM	Bomb: 235 mm Base: 85 mm	Bomb: 81 mm Base: 64 mm	Bomb: 81 mm Base: 20 mm
Tinder Rocketry Peregrine	8-38 g	NA	NA	Ignition	NA	NA	NA

Table 7.1: The specifications of the drop mechanisms. The different producers present the data in different ways.

EFLA Servoless Payload Release

The EFLA405 Servoless Payload Release has a design that makes it easy to fasten the mechanism to other objects, see Figure 7.1. It consists of two parts, one that is twice as big as the other and contains the electronics and one that is supposed to be fastened of the sensor. Both parts have holes in their bodies, which makes them suitable to be fastened to the UAV system and to the sensor. The two parts are connected through a metal pin on the active half that goes through a hole on the passive part. When the drop mechanism receives a pulse-width-modulated (PWM) signal in the range between 0-1 V, it pulls back the pin and the passive part is disconnected from the active part. The interface to a UAV system are three cables, demanding 5 V as power, ground and the control signal. The EFLA405 Servoless Payload Release can carry up to 340 g as a payload (E-flite advancing electric flight, 2014).

When the ELFA405 Servoless Payload Release releases its payload, it also releases its passive part. That means that this passive part should be easy to replace. One solution would be to buy a lot of drop mechanisms and use only the passive parts, but a simpler one is to make them with a 3D printer. This will not be further elaborated in this Master Thesis.

Quanum RTR Bomb System

The Quanum RTR Bomb System is built similar to the EFLA405 servoless Payload Release. See Figure 7.2 for a picture. It, too, consists of two parts where the larger part is active and releases the smaller, passive part when it receives a control signal. In Table 7.1, the active part is named *Base* and the passive part is named *Bomb*, as this mechanism originally is a toy that releases bombs. It, too has an interface with one cable for 5 V as power, one for ground and one for the control signal. But the Quanum RTR Bomb System is not suited to be fastened on anything with screws, as the bodies of both the active and the passive halves have smooth surfaces without holes. The only options would then be to fasten the parts by drilling holes into the bodies, by gluing them or by taping them to the UAV and to the sensor. Besides, the passive part is connected to the active part by only two plastic hooks. They do not yield when being pulled apart

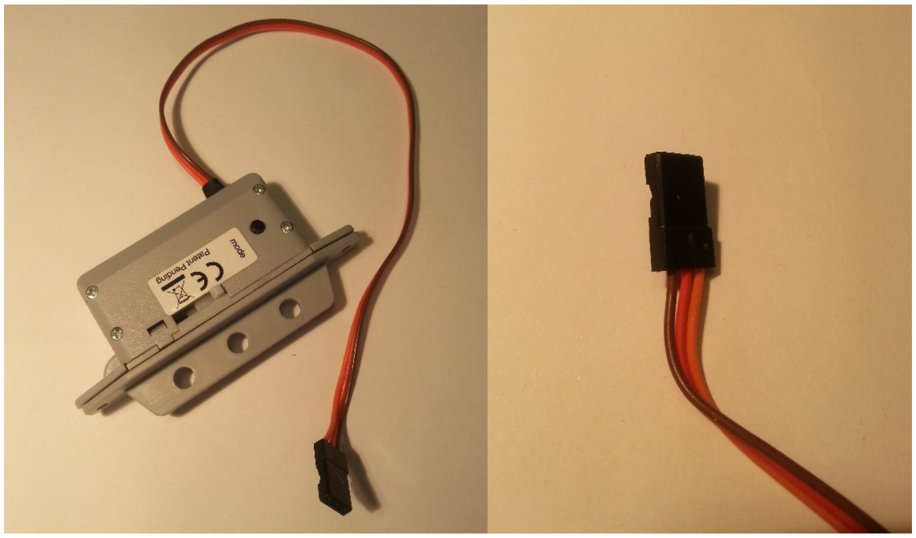


Figure 7.1: EFLA405 Servoless Payload Release. The brown cable is for ground, the red is for V_{cc} and the orange is for the signal

by moderate force, but seem less robust than the EFLA405 Servoless Payload Release.

Tinder Rocketry Peregrine

This might be used as a drop mechanism, although that may not be the original intention. An illustration of this mechanism is shown in Figure 7.3. The housing seems to be launched from the mount and cap by the force produced when compressed CO_2 is released, and the spring between the mount and the housing that pushes the housing away. A detailed instruction to how this launch should be prepared is shown in the manual by Tinder Rocketry (2014), but the manner of operation is not explicitly stated and might be different from the description above. It looks like the mount can be fastened to the UAV, having the drop mechanism standing perpendicular to the surface where the mount is fastened. Because of the uncertainties of this mechanism, and because it would demand an ignition to launch this mechanism, which might be unfortunate on a fuel driven UAV, it was never ordered and considered unfit for this project.

Selection of Drop Mechanism

The description of the three drop mechanisms that were evaluated for this project, indicates that one of the three is more suited than the others. The Tinder Rocketry Peregrine was excluded already in the investigation of the drop mechanisms, and was not ordered. The reasons for this was the ignition launching and the insufficient information about the system. The EFLA405 Servoless Payload Release has a specified payload weight limit, it seems more robust than the Quanum RTR Bomb System and is easier to fasten on the UAV system than the Quanum RTR Bomb System. The EFLA405 Servoless Payload Release is therefore selected as the drop mechanism for this project.

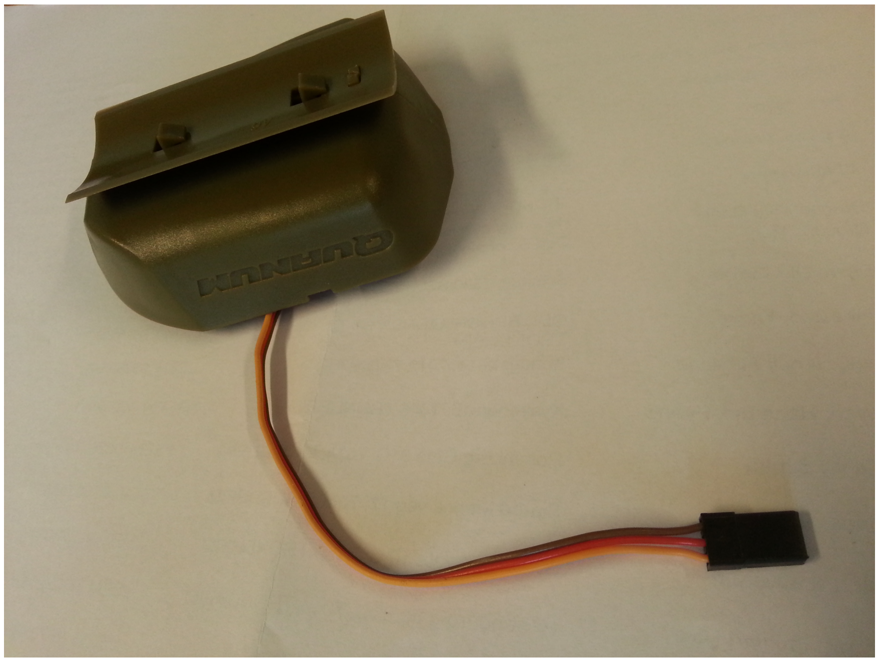


Figure 7.2: Quanum RTR Bomb System 1/6 scale plug-n-Drop. The brown cable is for ground, the red is for V_{cc} and the orange is for the signal

7.2 Signal Generator

The chosen drop mechanism, the EFLA405 Servoless Payload Release, is meant to be controlled with a radio channel, although none will be used in this Master Thesis for that purpose. The control signal is supposed to be a pulse width modulated (PWM) signal that controls the angle of the servo motor, which in turn pulls the pin in or pushes it out. The only requirements for this PWM signal is that a pulse should arrive at the drop mechanism's control cable periodically. This period is not given in the data sheet. As opposed to a standard PWM signal, this control signal is not determined by the duty cycle of the signal, but of the length of the pulse.

7.2.1 Choice of Signal Generator

The Pandaboard, which is supposed to signal to the drop mechanism when it should release its payload, does not have any dedicated PWM output and therefore has no easy way of generating the signal. That means that a PWM generator must be made. The debated alternatives are: To force through a PWM signal on the Pandaboard, to use an Arduino or to use an AVR directly.

The first alternative was researched for a while, but with meagre results. Although the OMAP4460 processor used in the Pandaboard has the ability to generate pulse-width-modulated signals, there are certain drawbacks with the alternative. First, the operating system on top of the Pandaboard makes it more difficult to use the instructions from the OMAP datasheet. Second, this solution would mean that if the Pandaboard was exchanged into another computer, then the

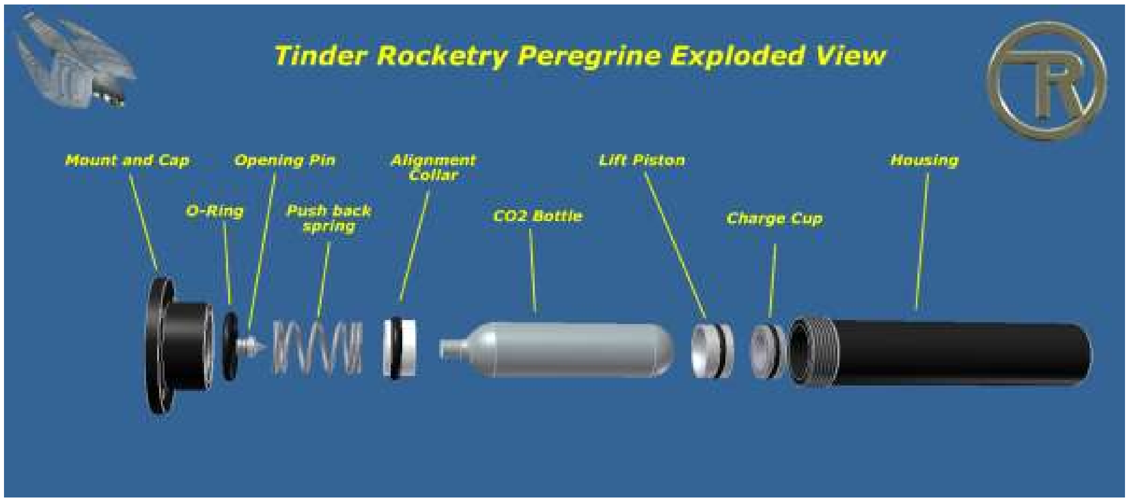


Figure 7.3: The Tinder Rocketry Peregrine. Exploded View (source: Tinder Rocketry (2014))

PWM would have to be implemented again, possibly with new difficulties. It would mean a poor modularity to be dependent on one part to make the rest work.

The second alternative, the use of an Arduino to produce a PWM signal, solves both of the problems that appeared with PWM generated on the Pandaboard. An Arduino is a platform that is particularly easy to use and only requires a programmer and a free and easily obtainable integrated development environment (IDE). The Arduino is build on top of an Atmel microcontrollers enabling most of what is possible on normal Atmel microcontrollers, but offering a simpler solution with pre-made functionality (Arduino, 2014). In our case, the Arduino Servo library offers a function called *write()*, which produces a PWM signal. The drawbacks of the Arduino alternative are first and foremost that it is too much. It is a good alternative, but even the tiniest Arduinos are bigger than the next alternative, the simple AVR microcontroller. Furthermore, inquiries done on peer students revealed that the Arduinos are not that robust. The final argument was that there were no Arduino available at the moment when a PWM generator was needed.

However, there are plenty of AVR microcontrollers available. A quick search to find out which AVR was required, revealed that as long as only a PWM signal is needed, the tinies ATtiny is enough. *Omega Verksted*, who has a large supply of electronic parts, had an ATtiny85 through-hole microcontroller available. The ATtiny85 uses 5 V as power supply, and can give an output of maximum 5 V on its I/O pins.

7.2.2 Control Signal

The lock on the drop mechanism connects the active and the passive part. When the lock is closed, then the passive part is fastened to the active part. When the lock is opened, then the passive part is released. A PWM signal generator is programmed in C on the ATtiny85. Fast PWM

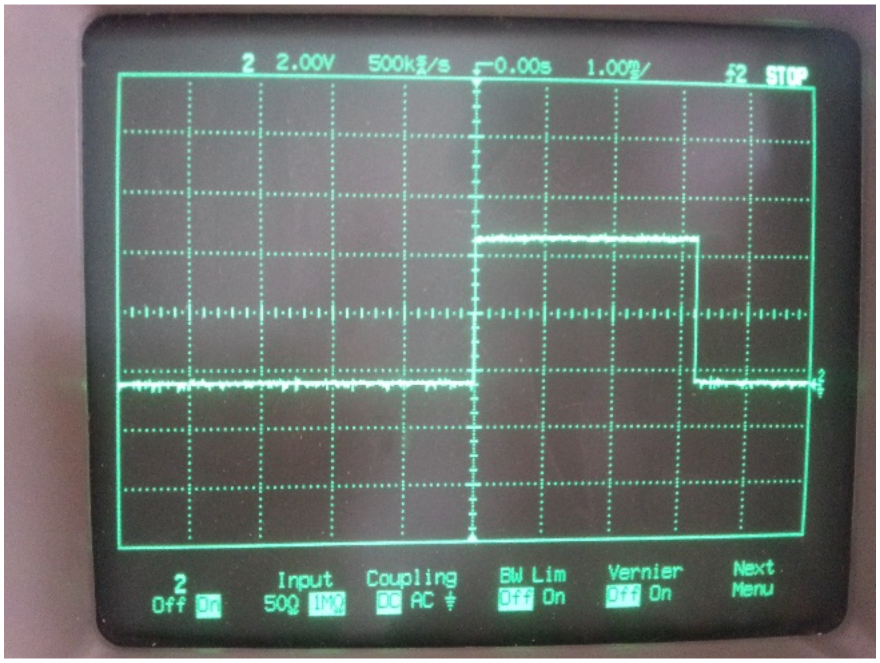


Figure 7.4: The PWM signal measured with a HP oscilloscope. This picture shows the maximum width of the signal. It repeats periodically every 16.5 ms

mode is used on the ATtiny85 with 255 as the highest value. A counter counts from 0 to the top value and is then reset to 0. When the counter is reset, the output value on the ATtiny85 pin that is assigned with the PWM output is set hight, that is, to 5 V. When the counter reaches a given value called *output compare*, the output pin is set low, that is, to 0 V. The *output compare* value counts from 0 to 50 and back to 0 continuously when the Pandaboard commands the ATtiny85 to generate the control signal for the drop mechanism. As the counter goes to 255 and the maximum value of the output compare register is 50, then the value of the PWM output pin, measured with a voltmeter, would vary between 0 and 1 V.

The output value of the ATtiny85's PWM output pin is a periodic signal with a period of 16.5 ms, with a width that varies between 60 μs and 3,3 ms. The drop mechanism opens its lock when it receives a narrow PWM signal and closes the lock when it receives no PWM signal or when the width of the PWM signal is too wide. This way, a short circuit will not result in an open lock. The code for the ATtiny85 PWM signal generator is found in Appendix B. The PWM signal is measured with an oscilloscope and a picture of the signal at its broadest is found in Figure 7.4.

As the datasheet of the EFLA405 Servoless Payload Release did not specify the control signal for the mechanism, a working control signal was found by trial and error. As the values for opening and closing the lock are not given, it is reasonable to try with a signal sweep. The one described in in this section may therefore be less than optimal, but it works as it should and consequently. Therefore, no further effort is put into optimizing the control signal.

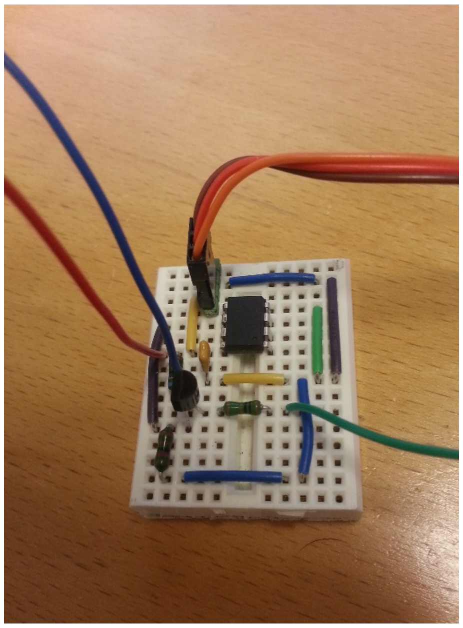


Figure 7.5: The temporary signal generator on a breadboard.

7.2.3 Interface between Signal Generator and Pandaboard

Only one challenge remained: As described in section 8.2.1.1, the payload drop is signalled by a GPIO pin on the Pandaboard. When the Pandaboard output pin is high, it measures 1.9 - 2 V. This is not enough to register as a 1 on the input pin of the ATtiny85, which has a threshold voltage at 2,5 V when the V_{cc} is 5 V. To achieve at least 3 V as input voltage on the ATtiny85, a voltage controlled voltage switch was built with a transistor and two resistors, see Figure A.5 in Appendix A. The transistor in use is an NPN transistor of the 2N4402 model, which inverts and amplifies the input signal. That means that the signal for "start the control signal" on the Pandaboard should be low, and the signal for "stop the control signal" should be high. Then the signal coming from the Pandaboard controls a 5 V signal, which is fed as input to the ATtiny85, causing a control signal for the drop mechanism.

When the Pandaboard activates the drop mechanism, the drop mechanism starts by pulling the lock pin in, continues by pushing the pin out and continues on with that until deactivated. An alternative solution for this would be to withdraw the pin once and push it out again, or to pull the pin in as long as the drop mechanism is activated and push it out upon deactivation. As there is only necessary to release the payload once, the second or third alternative would be the sensible ones. However, if the hardware got stuck or a mechanical error occurred, then the payload would not be released at all. Therefore, the drop mechanism opens and closes the pin continuously when activated, and does nothing when deactivated.

The temporarily built signal generator is shown in Figure 7.5. After testing this design, it was found suitable and soldered onto a veroboard, see Figure 7.6.

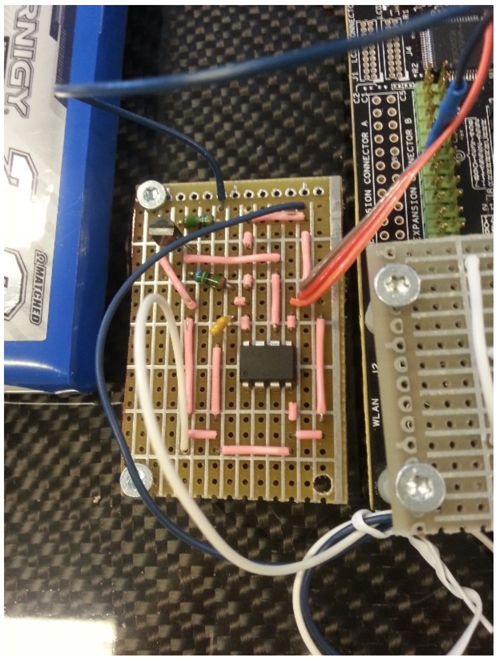


Figure 7.6: The soldered signal generator

7.3 Placement of the Drop Mechanism on the UAV

When the drop mechanism has released its payload, the payload should fall down to the earth. This requires the drop mechanism to be mounted outside the body of the UAV, or a mechanism for opening a gap or lid on the body of the UAV. The second alternative was abandoned without discussion, as the UAV should remain as whole as possible. Then the drop mechanism must be placed on the fuselage on the UAV, and a placement must be found.

As Figure 3.3 shows, the Penguin B has a propeller behind the body of the UAV. This must be taken into consideration, as any object that is caught in the propeller can damage the propeller or itself. Another consideration is that the drop mechanism will be exposed to varying vibration and air drag, depending on where it is placed on the UAV. The discussed alternatives to a placement of the drop mechanism were underneath its belly, on the wings, on the tailboom assembly or on the wheel assembly (see Figure A.3, Appendix A).

Placement of the Drop Mechanism under the Belly of the UAV

A sketch of this solution is shown in Figure 7.8. The main problem with the this alternative, to place the drop mechanism under the UAV body, is the risk of the payload hitting the propeller. The Penguin B moves in a horizontal direction and the payload is released vertically, so the risk of it hitting the propeller depends on the difference in speed between the UAV and the released payload caused by air resistance, the wind and drag forces and the size of the propeller.

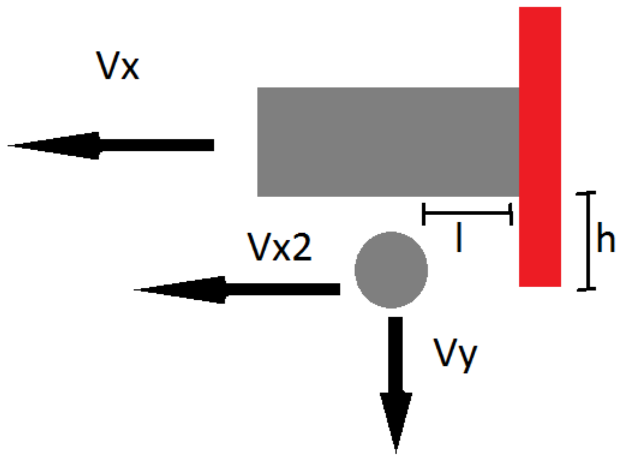


Figure 7.7: A sketch to illustrate the problem connected to dropping a payload from under the UAV. The circle illustrates the payload, the grey rectangle shows the UAV and the red rectangle is supposed to be the propeller.

In Figure 7.7, this solution's physics is illustrated: V_x is the assumed constant horizontal velocity of the UAV, while V_x2 is the constantly decreasing horizontal velocity of the payload after it is dropped: Initially it equals V_x, but it is decelerated as it meets air resistance. V_y is the increasing vertical velocity of the payload after being dropped, h is the length of the part of the propeller that reaches under the UAV body and l is the length between the placement of the payload on the UAV and the propeller. With the drop mechanism attached to the body of the UAV, without anything holding it lower than the propeller, it is hard to say what will happen without further calculations - whether or not the payload will hit the propeller or not. The air drag force that might lift the payload up after it has been released is difficult to predict and the air resistance slowing the payload down in the horizontal direction depends on the shape of the payload. It is also difficult to test, as one failure might damage the propeller severely. The conclusion is that this is placement is not an option.

If, however, something could hold the drop mechanism below the lowest point of the propeller, that problem would be avoided. That arm could either be in a permanent lowered position, or it could be lowered for the drop, and then raised afterwards. If the permanent low situation were chosen, this might cause problems during the launch or landing of the UAV. The launch is preferably done with a catapult and this arm might spoil the process by getting in the way. The landing of the Penguin B is a project under development but for the time being, the Penguin B lands like a normal air plane using its landing wheels, and any lower point than the wheels would spoil the landing. If the drop mechanism was lowered down from the UAV on a rigid arm that were later raised, the the sensor could be dropped from an arbitrary low point. However, the arm would have to be raised again and this process could contain errors. Besides, it is a complicated process to build such an arm. This could be a good solution if there were more time at disposal, but can not be used in this Master Thesis.

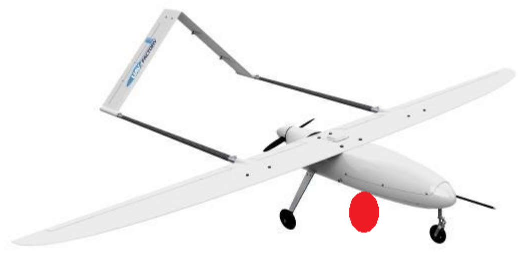


Figure 7.8: A sketch of the placement of the drop mechanism under the belly of the UAV (source for the figure of the UAV: UAV Factory (2014))

Placement of the Drop Mechanism on the Wing Tips

The next alternative discussed is to attach the drop mechanism on one (or both) of the wing tips. A sketch of this solution is shown in Figure 7.9. The sensible solution would be to have two drop mechanisms and two sensors to drop, as the UAV would then be better balanced. On the wing tips, the drop mechanisms are out of the way of the propeller and the mounting process would not offer many problems. The drawback is that way out on the tips of the wings, the drop mechanism (and the payload, which need not be small and neat) will inflict the aerodynamics of the UAV badly and this might change the center of gravity and at least give the UAV some unbalance. That makes this placement a possible, but maybe not optimal solution.

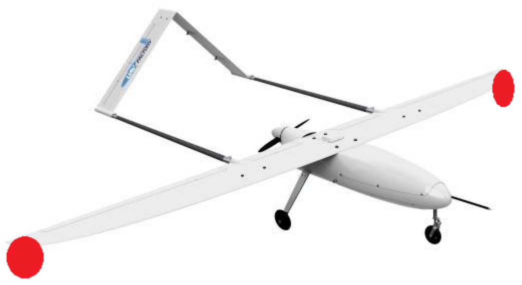


Figure 7.9: A sketch of the placement of the drop mechanism on the wing tips of the UAV (source for the figure of the UAV: UAV Factory (2014))

Placement of the Drop Mechanism on the Tailboom Assembly

To place the drop mechanism on the tailboom assembly is a good alternative without any of the mentioned drawbacks. A sketch of this solution is shown in Figure 7.10. Two drop mechanisms could be places on the two tailboom bars or one could be places in the middle of the tail joint. Being behind the propeller means that the drop mechanisms can not release any sensors that will fly into the propeller, and as the tailboom is almost in the middle of the two wing tips, the drop mechanisms will not disturb the center of gravity. The main drawbacks of this solution are the difficulties to control the drop path of the payload after it is caught by the backwater of the propeller and that the UAV gets heavier astern than before. Apart from that, this is a possible placement.

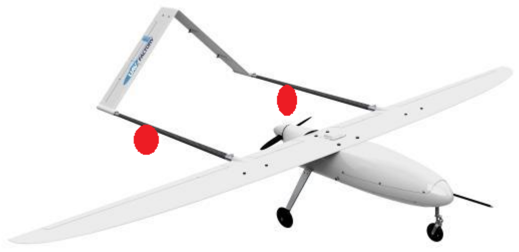


Figure 7.10: A sketch of the placement of the drop mechanism on the Tailboom Assembly of the UAV (source for the figure of the UAV: UAV Factory (2014))

Placement of the Drop Mechanism on the Wheel Assembly

The last placement alternative is to place the drop mechanism on one of the wheel assemblies. A sketch of this solution is shown in Figure 7.11. This alternative involves having two drop mechanisms, one on each of the bars where the wheels are placed. Here, the hight difference between the drop mechanisms and the propellers can be adjusted and the drop mechanisms will not affect the landing. However, depending of the design of the attachment mechanism, it might disturb the launch catapult. If that is not the case, this is a preferred placement of the drop mechanism: The released sensor will not be caught in the propeller, its drop to the ground is pretty deterministic and the distance from the payload mount containing the signal generator to the drop mechanisms is relatively small.

Selection of Drop Mechanism Placement

There are several possible placements for the drop mechanism. Although both the tailboom assembly, the wing tips and the wheel assembly are alternatives, the wheel assembly is chosen. There, the two metal bars between the fuselage of the UAV and the wheels can be used to fasten

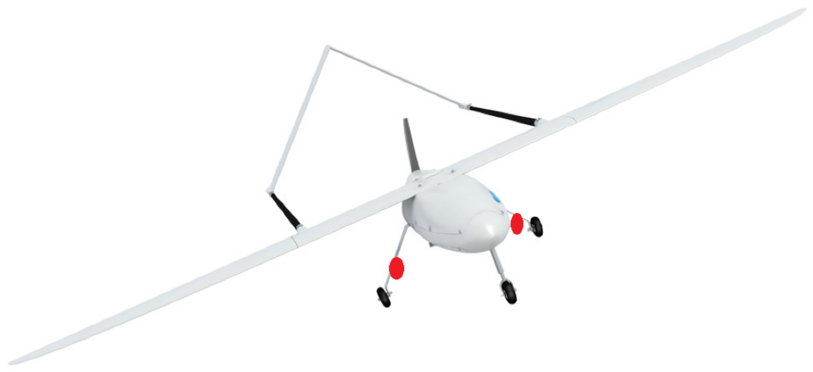


Figure 7.11: A sketch of the placement of the drop mechanism on the Wheel Assembly of the UAV (source for the figure of the UAV: UAV Factory (2014))

the drop mechanisms onto, which is not a possibility on the wing tips. Besides, the UAV does not get any heavier on its aft than it has to. The only drawback is the possibility of getting caught in the catapult, which must be tested. The tailboom assembly stands as a possible spare solution.

7.4 The Substitute for the Sensor

The sensor that should be dropped on the iceberg from the UAV is not yet designed, and a substitute will have to be used during the tests. This was made by the engineering workshop of the Department of Engineering Cybernetics out of polyvinyl chloride (PVC) and can be seen in Figure 7.12. It has the shape of a spherical sefment with a diameter of approximately 6 cm and a height of approximately 2 cm. On the middle of the flat side, a hole has been made. The passive part of the EFLA405 Servoless Payload Release is attached into the hole.

7.5 The Drop Mechanism Assembly

When the drop mechanism is chosen, the signal generator is made and the placement of the drop mechanism is chosen, the assembly is next. The EFLA405 Servoless Payload Release is fastened to a metal bar with the use of thin aluminium plates and hard plastic, see Figure 7.12. The metal bar is attached to the wheel assembly with two hard plastic joints. They are clenched onto the wheel bars with screws and can be moved up or down on the wheel assembly bars, see Figure 7.13. The horizontal bar carries two drop mechanisms and is attached to both wheel assembly bars. The design was tested on the catapult, and the test did not reveal any problems with the design.

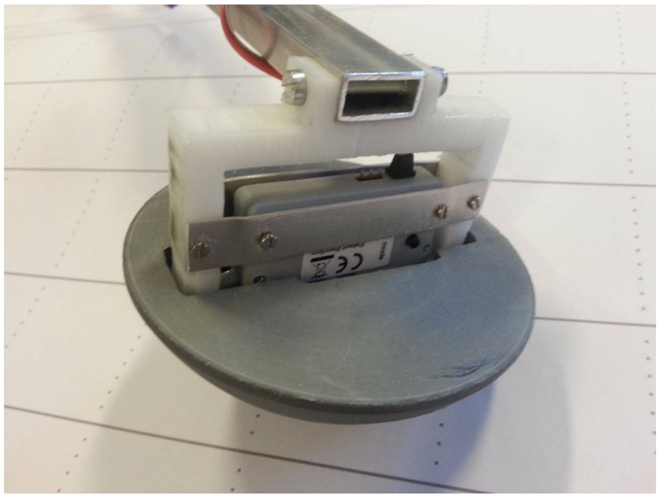


Figure 7.12: The EFLA405 Servoless Payload Release is attached to the metal bar with aluminium plates and a hard plastic frame.

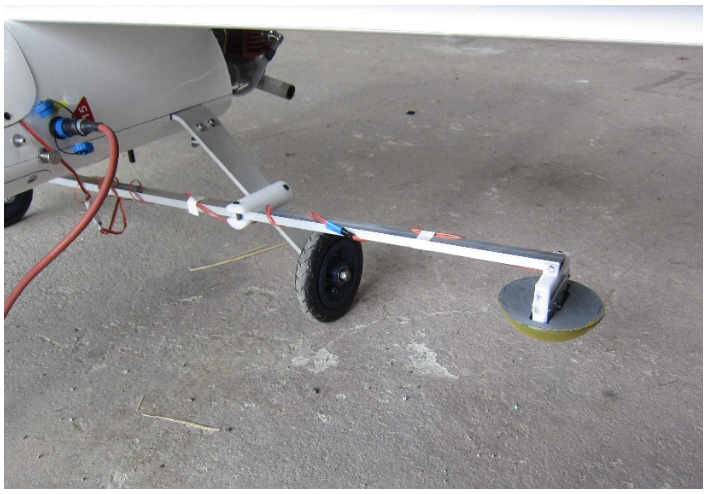


Figure 7.13: The metal bar with the drop mechanisms on it is attached to the UAV wheel assembly bars with two plastic joints.

Chapter 8

The Software Implementation

Although a small part of the software is placed in the AtTiny85 controlling a PWM generator, most of the software in this project runs on the Pandaboard or on the ground station computer. It uses an extended version of the DUNE software solution (mentioned in Section 3.1.1), developed especially for this project, and has an interface to the autopilot, to the drop mechanism through the Pandaboard hardware and to the radios. This chapter describes this software architecture and its interfaces.

8.1 Software Architecture on Pandaboard and User Interface

The main structure of the software architecture in this project is the DUNE software provided by LSTS, see Section 3.1.1. A separate branch of the open-source code was made using the subversion control system Git, for use in this specific project. The branch was called *precisionAirDrop*. In addition to the already existing tasks, some new tasks were added. The software was divided into two distinct modules: The ground station and the on-board code, which are two separate DUNE *.ini*-files. The Ground Station will be the user interface. It sends the geographic coordinates of the target point to the on-board side and in return, it gets the states of the UAV.

All communication follows the IMC protocol, and the same IMC message can be fetched in the ground station as in the on-board code. To communicate wirelessly between the ground station and the on-board code, the IMC messages use an underlaying communication protocol called UDP. This protocol is used to transfer IMC messages with the Ubiquity radio equipment. There is built-in functionality for this in DUNE by including a task called *Transports.UDP* in the *.ini* file. Then IMC messages can be dispatched from a task on the Pandaboard, and without further code it is consumed by a task on the ground station. That way, IMC messages can easily be transmitted to a specific IP address that uses DUNE, or simply be broadcasted.

The user interface will be made with hard coded input instead of input from the command prompt. This is made because that solution is the simpler one, and because small changes in the code on the user interface do not conflict the code on the Pandaboard, as long as the messages received on the user interface are the once sent from the Pandaboard, and vice versa.

The on-board code calculates an optimal point of release for the UAV based on the received point of impact from the ground unit. It then calculates an optimal path to the point of release for the UAV from any point nearby the point of impact (on the same level as the point of release, though: the same height as the UAV is always supposed to fly). This path is communicated to the Piccolo Autopilot via IMC messages, which in a special task called Piccolo are translated to the Piccolo protocol. The same task translates messages from the autopilot to the IMC protocol and makes them available on the IMC bus. When the UAV is in the point of release, it tells the drop mechanism to let its payload be released.

The finished DUNE code is cross-compiled for an ARM kernel, is packed and transferred to the Pandaboard. The Pandaboard has a Linux operating system, which lies on a Secure Digital (SD) memory card attached to the board. This operating system has a start-up script that runs the DUNE program with the correct *.ini*-file.

8.2 Class Relationship Overview

As DUNE operates with *.ini*- files that include different tasks as parameters when the program is running, the normal class diagram has to be adjusted slightly. A relationship overview is a more appropriate term, which tells about the relationship between the tasks, *.ini*-files and their messages.

8.2.1 Piccolo

The *.ini*-file that is used on the Pandaboard is called Piccolo. An *.ini*-file is described in Section 3.1.1. When an *.ini*-file includes a task, it means that it lists the tasks that should be used when the program is running. The Piccolo *.ini*-file includes all the tasks that are necessary to have on the Pandaboard, both tasks designed for this purpose and other, more general tasks. To be able to communicate with the autopilot over TCP, it needs to include the task Transports.SerialOverTcp. This is a general task which takes as arguments the serial port address, baud rate and the TCP port. The baud rate is 57600 and the TCP port is 2001, while the serial port /dev/ttyUSB0 is used to communicate with the autopilot.

To be able to communicate with the Ground Station, the task Transports.UDP is included. It takes as arguments its local port, the destination IP and port and a list of which IMC messages that should be transported over UDP. The local port is set to 1024, while the destination port is 1025. The destination IP address is the IP address of the Ground Station. The IMC message EstimatedState is sent over UDP from the Pandaboard to the Ground Station.

The third general task included in this *.ini*-file is the Piccolo task. This task communicates with the autopilot and translates the Piccolo protocol to IMC messages. The IMC-messages received from and sent to Piccolo are shown in Table 8.1. To communicate with the autopilot,

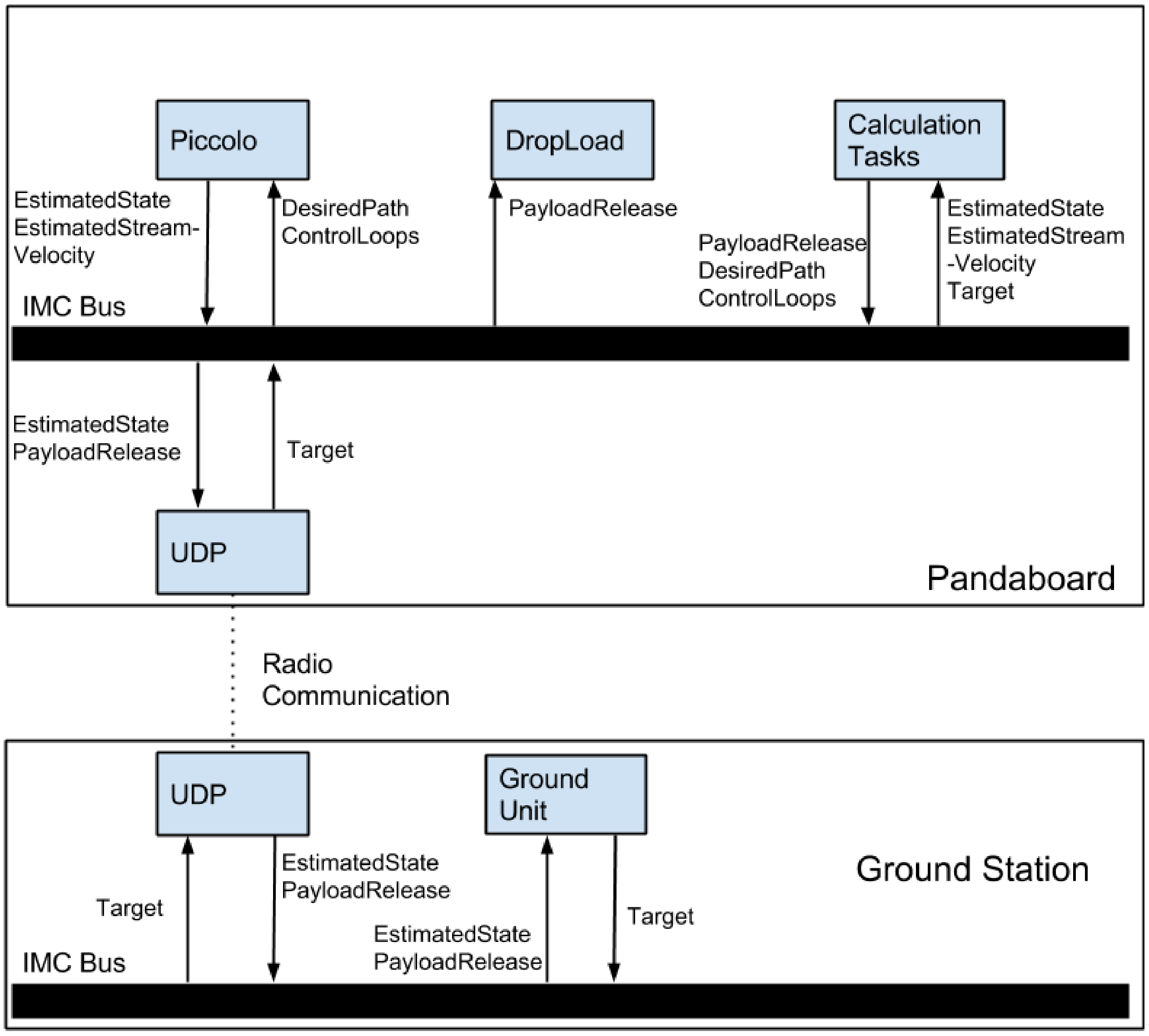


Figure 8.1: The software architecture on the Pandaboard and the ground station with user interface

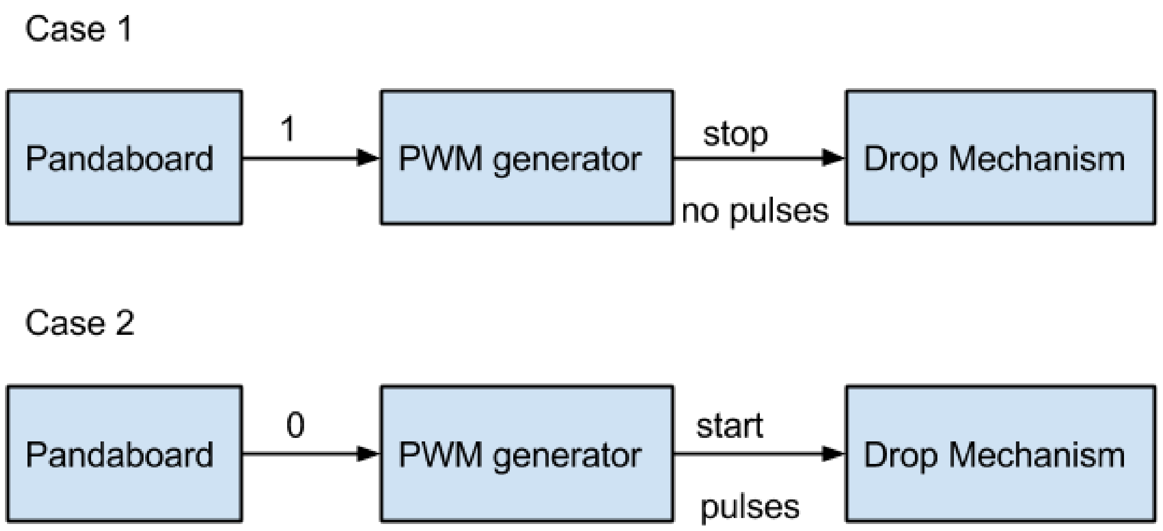


Figure 8.2: The signal flow from the Pandaboard to the drop mechanism.

the Pandaboard is connected with the autopilot using an RS-232 serial cable. All messages to and from the autopilot are sent on this cable, and all of them are translated in the Piccolo task.

Received from Piccolo	Sent to Piccolo
IMC::EstimatedState	IMC::ControlLoops
IMC::EstimatedStreamVelocity	IMC::DesiredPath

Table 8.1: The IMC messages received from and sent to Piccolo in this project.

The rest of the included tasks are designed specifically for this project, and are presented in the following subsections.

8.2.1.1 The DropLoad Task

The task called *PenguinDrop.DropLoad* binds the IMC message called PayloadRelease. It creates a connection to the directory of GPIO36 on the Pandaboard, which corresponds to pin 17,J6 on the Expansion Connector on the Pandaboard, see Figure 3.2, and sets its value to 1 initially. When it receives IMC::PayloadRelease, it writes the value of this GPIO to 0. Pin 17, J6 on the Expansion Connector is connected to the PWM generator and is active low. That means that when Pin 17 is low, then the PWM generator generates pulses to the drop mechanisms, which in turn let their payload go. This is illustrated in Figure 8.2.

8.2.1.2 The Calculation Tasks

This is a group of tasks used to calculate the point of release (PoR), calculate a path to this point from the current point and decide when the payload should be released. As several methods are tested during the Master Thesis, several versions of the calculation tasks exist.

Version 1

The first version of the calculation tasks was used for the first HIL and field tests for drop on target. In this version, the calculation tasks group consists of only one task called *PrecisionAirdrop.ReleaseObject*. This task communicates directly with the Ground Station and receives the target position as well as the acceptance radius for an acceptance sphere used to accept waypoints. This task does not calculate a point of release; it simply receives IMC::EstimatedState from the Piccolo task, checks if the position of the UAV is within the acceptance sphere and if it is, it tells the DropLoad task to release the sensor. Acceptance with half planes is also implemented for test purposes.

Version 2

In the second version of the calculation tasks group, the *.ini*-file only includes the task called *PrecisionAirdrop*. This version of the calculation tasks was used for the Straight Line Approach. *PrecsionAirdrop* includes three other classes called *ReleasePoint*, *ReleaseVelocity* and *WaypointGenerator*. *PrecisionAirdrop* communicates with the Ground Station and receives the position of the target and the radius of the acceptance sphere. It then calculates the point of release using the classes called ReleasePoint and ReleaseVelocity. The calculations are based on the target position received from the Ground Station and the wind measurements from the Piccolo Autopilot.

The next step is to generate a trajectory consisting of waypoints. This is done with the use of the class called WaypointGenerator. The trajectory only consists of three waypoint, where the starting waypoint is the *far away* point of the Straight Line Approach (see Section 6.2.3), the second waypoint is the point of release and the third waypoint is a point along the straight line, some distance after the point of release, making the UAV continue along the straight line after having released the sensor.

Version 3

In this version of the calculation tasks group, a modularization is done. It consists of one task called *PrecisionAirdrop.StraightLine* and one called *PrecisionAirdrop.AdaptedDubinsPath*, which communicate with the ground unit wirelessly and with the Piccolo task. Their relationship is illustrated in Figure 8.3. Both calculation tasks receive the IMC::Target message position from the ground station, and they then check for the Destination Entity of the message. The calculation task that is the correct receiver of the message then calculates a path from the UAV's current position to the point of release using their method. The waypoints are fed to the autopilot and the trajectory is tracked using IMC::EstimatedState from Piccolo. When the UAV-position sent from the autopilot shows that the UAV is within the acceptance sphere, it sends a message to the task

PenguinDrop.DropLoad, telling it to release the sensor.

All versions of the calculation tasks can use both half planes and acceptance radius to accept a waypoint, but the half plane is the standard acceptance criteria. The reason for that is that an acceptance sphere can be omitted from the trajectory of a UAV it the measurements come with too low frequence, while it is impossible to omit the crossing of a half plane that way.

8.2.2 GroundControl

The *.ini*-file GroundControl includes only two tasks: PenguinDrop.GroundUnit described in the following subsection, and Transports.UDP. The latter is the same tasks as in *Piccolo.ini*, but here, its local port is 1025 - the same as the destination port of Piccolo's UDP task - and the destination port is corresponding 1024. The destination address is the IP of the Pandaboard. This task sends the IMC messages Target and EstimatedState to the Pandaboard. The *GroundControl.ini*-file represents the user interface of the software.

8.2.2.1 The GroundUnit Task

The task called PenguinDrop.GroundUnit takes input from the user to the Pandaboard. This task can print desired output on a terminal and take input from the user, which it can send to the Pandaboard. The last version of the GroundUnit task sends the following parameters from the Ground Station to the Pandaboard: The Point of Impact for the sensor and the chosen calculation method.

8.3 Addresses and IDs

Because the system should be able to coexist with other equipment, a standardization of IP addresses and IDs is chosen. The subnet used to communicate on is 192.168.0.X. Here, the addresses above 100 are to be used by the different units in student projects. The ones used in this project are described in Table 8.2.

IP address	Unit
192.168.0.110	Pandaboard
192.168.0.111	Rocket M5
192.168.0.112	The Author's PC
192.168.0.113	Simen Fuglaas's PC
192.168.0.114	Nanostation M5

Table 8.2: The IP addresses used in this project.

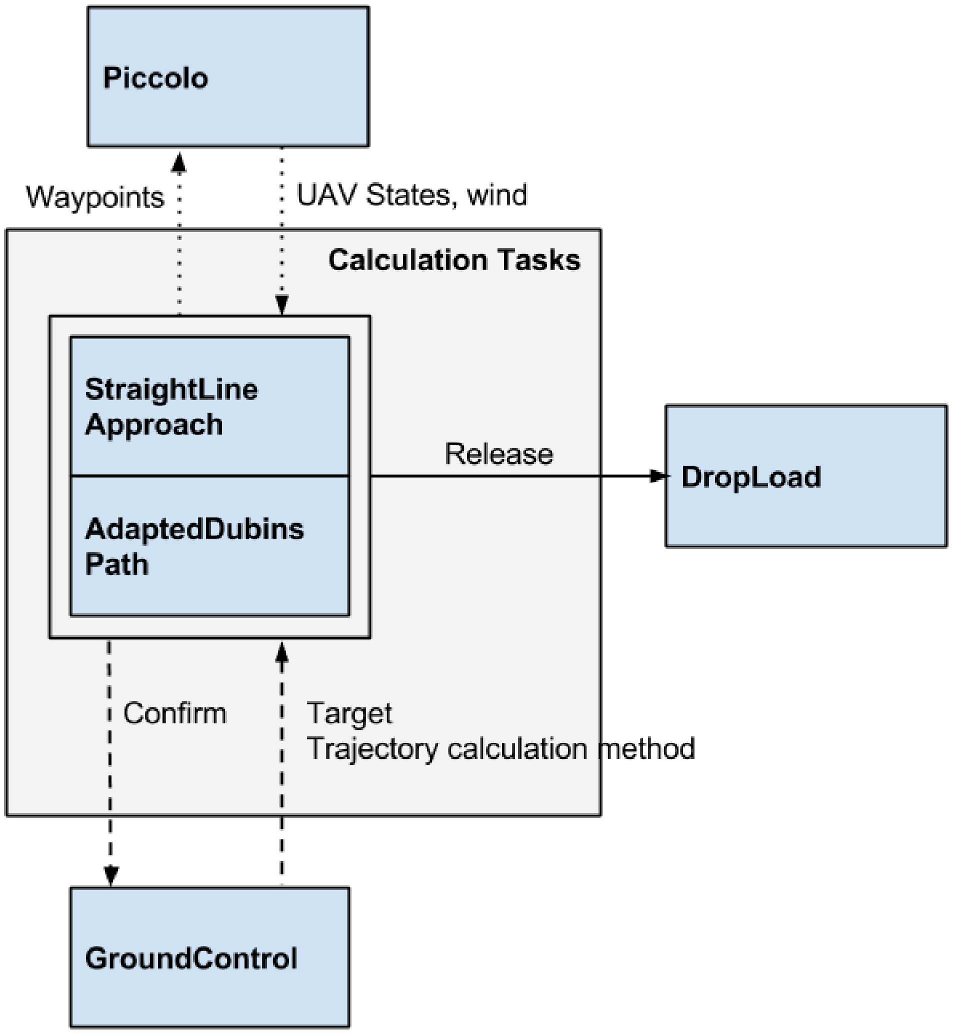


Figure 8.3: The ground station communicates with both calculation tasks at the same time, but only the one that is labelled as receiver of the target computes a trajectory and communicates with the autopilot.

Chapter 9

Implementation of Hardware on the Penguin B

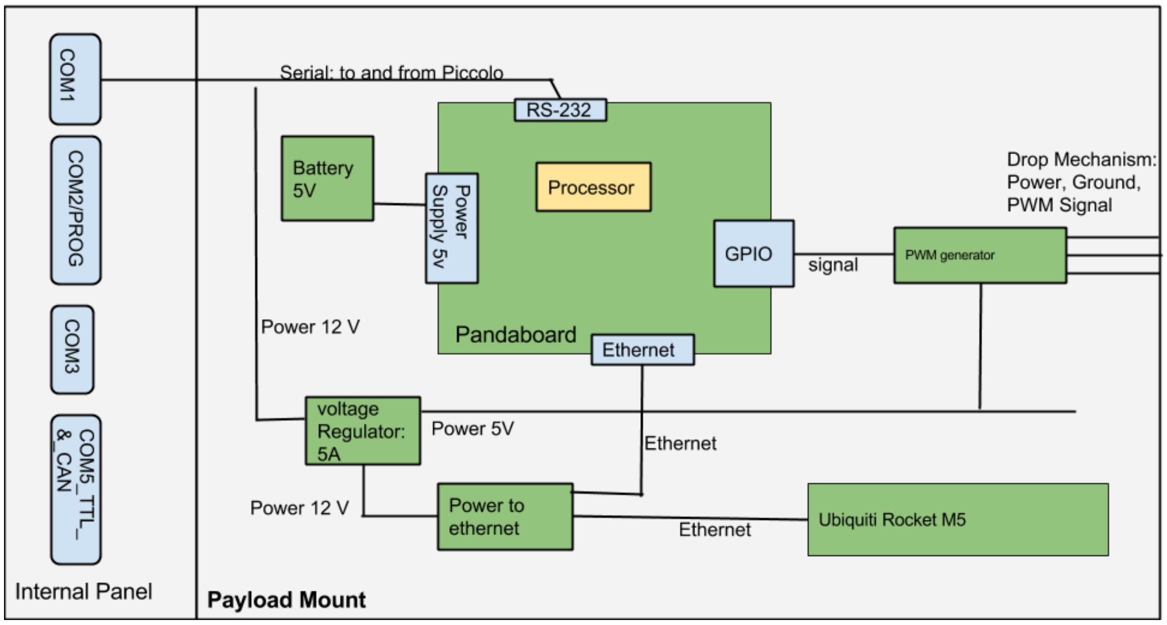


Figure 9.1: The hardware architecture of the payload mount

The first field test came faster than expected, which is why the hardware architecture has two editions. The first edition simply fastened the parts of the hardware design to the payload mount, while the second edition was more thought-through, with better attached cables and using the experiences gained in the first field test. Common for both editions are the hardware parts used, both the ones designed in the project and the commercials off-the-shelf. In this chapter, the hardware architecture, the construction and the placement of the hardware parts will be thoroughly described. See Figure 9.1 for the hardware architecture.

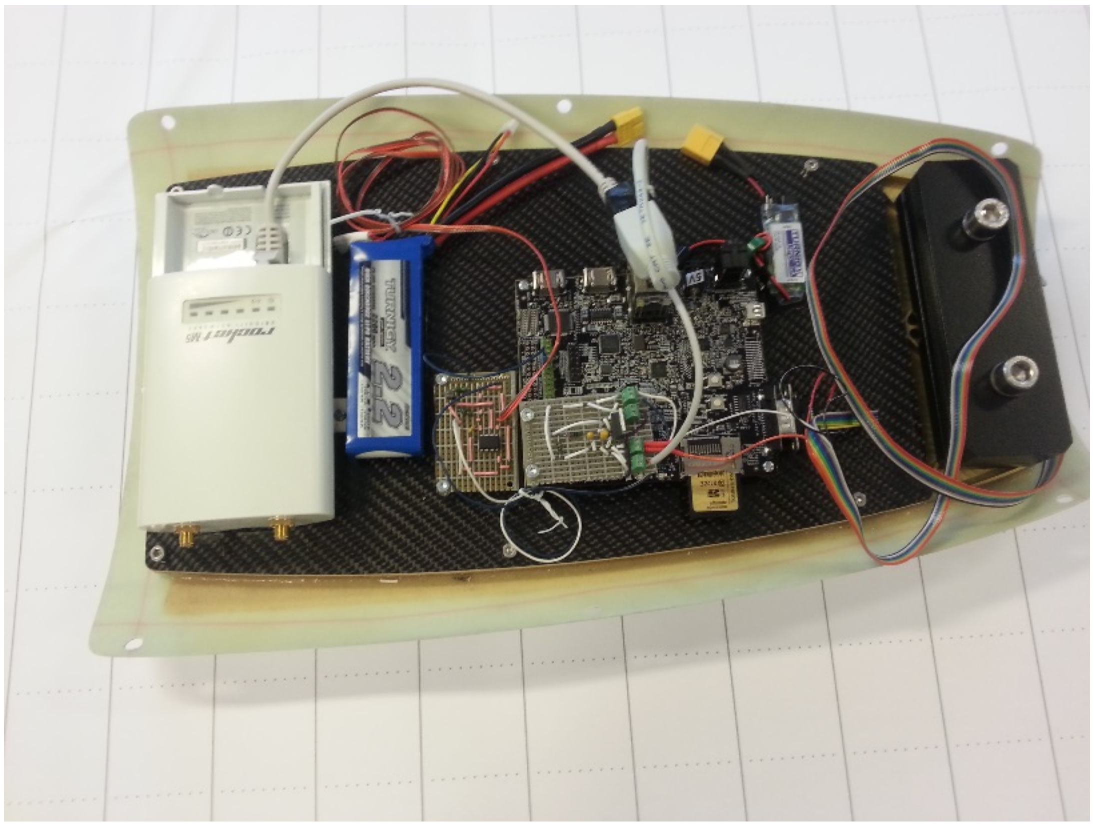


Figure 9.2: The first design of hardware, implemented on the payload mount.

9.1 Radio Communication

In order to have communication between the UAV and the ground, wireless communication is needed. For the Penguin B, it was decided by the project management that equipment with 5.8 GHz frequency from Ubiquiti Networks should be used, which is their airMAX series. More specifically, a Ubiquiti Rocket M5 should be used on the UAV, sending and receiving signals to and from the ground, and on the ground, a Ubiquiti Nanostation M5 should be used for testing. On field tests, another Rocket with a larger range is used. That rocket will not be described further in this Master Thesis. The Ubiquity equipment is described in Section 3.5.

Both radio nodes - Ubiquiti Rocket M5 and Ubiquiti Nanostation M5 - have an Ethernet interface. They are also powered over Ethernet (POE), meaning that they only have one input port, which is Ethernet carrying power to run the radio node. To supply the radio nodes with POE, power from the UAV will be used. The Rocket M5 requires 12 V DC.

The settings of the radio nodes are accessible from a web browser by typing the IP (internet protocol) address in the browser's address field. To make this work, the computer has to be on the same subnet as the radio nodes. This means that the computer's IP address must be changed from a dynamic IP set by a dynamic host configuration protocol (DHCP) to a static IP address on the 192.168.0.x subnet. A DHCP is a protocol used to distribute IP addresses to equipment that wants to communicate with the Internet Protocol (IP).

In the field tests, only the Rocket M5 connected to the Pandaboard will be used, as it communicates with a common transceiver with longer range than the Nanostation M5, but for lab testing, the Nanostation M5 is useful as it allows testing of the communication with the Rocket M5. The settings on the Wireless page of the Rocket M5 are shown in Figure 9.3. The SSID on the third line of the page in the figure is *UAVskywalker*, which is the SSID used for all Ubiquity equipment during the field tests. In addition to these settings, the *Network Mode* should be set to *Bridge*.

For the Nanostation M5, the test settings are equal to the ones of the Rocket M5, with the exception that the *Wireless Mode* should be set to *Access Point*. Besides, the IP address must obviously be the same, as there are two radio units communicating with each other. The IP address of the Nanostation M5 was changed from 192.168.1.20 to 192.168.0.114, and the IP address of the Rocket M5 was changed to 192.168.0.111. After this was changed, the radios provided the Pandaboard with internet.

9.2 Power Supply

The Penguin B has a power generator and delivers 12 V and 6 V. The 6 V power supply is reserved for servos used in another project, which leaves the 12 V power supply for other purposes. The Pandaboard requires 5V for power supply and is powered with a 2-cell Lipo battery from Turnigy. As this battery delivers 7.4 V, which means that the voltage from the battery must be transformed to 5 V before it reaches the Pandaboard. a Turnigy EBEC-5A voltage transformer is used for this purpose, see Figure 9.5.

The Rocket M5 that was described in Section 9.1 requires 12 V powering whereas the ATtiny of the PWM-generator described in Section 7.2 and the drop mechanism both require 5 V. To supply all devices with power from the Penguin B power outlet, a power supply transformation station was built, see Figure 9.4. The two lower green cable stations are the input side of the transformer. Here, 12 V from the Pegnuin B is put into one cable terminator and the POE cable for the Rocket M5 goes into the other. The two upper green boxes are the output of the transformer and provide 5 V. The PWM-generator gets power from one of these cable terminators with 5 V, and the drop mechanism collects its V_c from the PWM-generator card.

The voltage regulator in use is a LM1084 5A Low Dropout Positive Regulator from Texas Instruments (Texas Instruments, 2013), connected according to the datasheet, see Figure A.8, Appendix A.

9.3 Implementation of First Design

The payload mount started out entirely empty except for the ballast slug, see Figure A.1 in Appendix A. As the payload mount had not been tested on the Penguin B, and no directions on the hardware architecture had been received, it was assumed that the whole payload mount could be used. The Pandaboard and the rest of the hardware were mounted on the payload mount as quickly as possible, with an architecture that assumed vast space on the payload mount and little thought of future expansions. See Figure 9.2 for the implementation of the first design.

All parts were fastened with screws on the payload mount except for the battery and the voltage transformer, that were fastened with hook-and-loop fasteners. The voltage regulator circuit described in Section 9.2 was placed next to the Pandaboard while the PWM generator circuit described in Section 7.2 was placed on top of the Pandaboard with spacers between. The Rocket M5 is screwed on the payload mount. Pandaboard communicated with the Piccolo through a serial cable that was connected to the port *COM 1* on the UAV Piccolo interface, see Figure A.4 in Appendix A. Two of the wires on that port were however carrying power and ground for the power supply card, and were therefore taken out of the serial cable and guided to the power supply instead.

9.3.1 Drawbacks with the First Hardware Design

The first hardware design was tested on the first field test on Agdenes. The conclusion was that the design used too much space on the payload mount: Only the outer 25 cm were available. Apart from that, the hardware worked, but the cables would have to be more thoroughly fastened to the payload mount. These feedbacks resulted in the second hardware design.

9.4 Implementation of Second Design

To make room for all hardware parts on a length of 25 cm of the payload mount (the part furthest away from the ballast slug), and to design the hardware architecture with a view to future expansions, a payload box is made. The PWM generater remained as before, as well as the voltage regulator circuit. The drawings of this box and its lid are shown in Figures A.6 and A.7. The box was made on the engineering workshop of the Department of Engineering Cybernetics, out of aluminium sheets. Holes were drilled into them to fasten the box into the payload mount. The box and lid were made according to the drawings, but 1 cm shorter than the one in the drawing, to get a better margin. The finished box can be seen in Figure. 9.6.

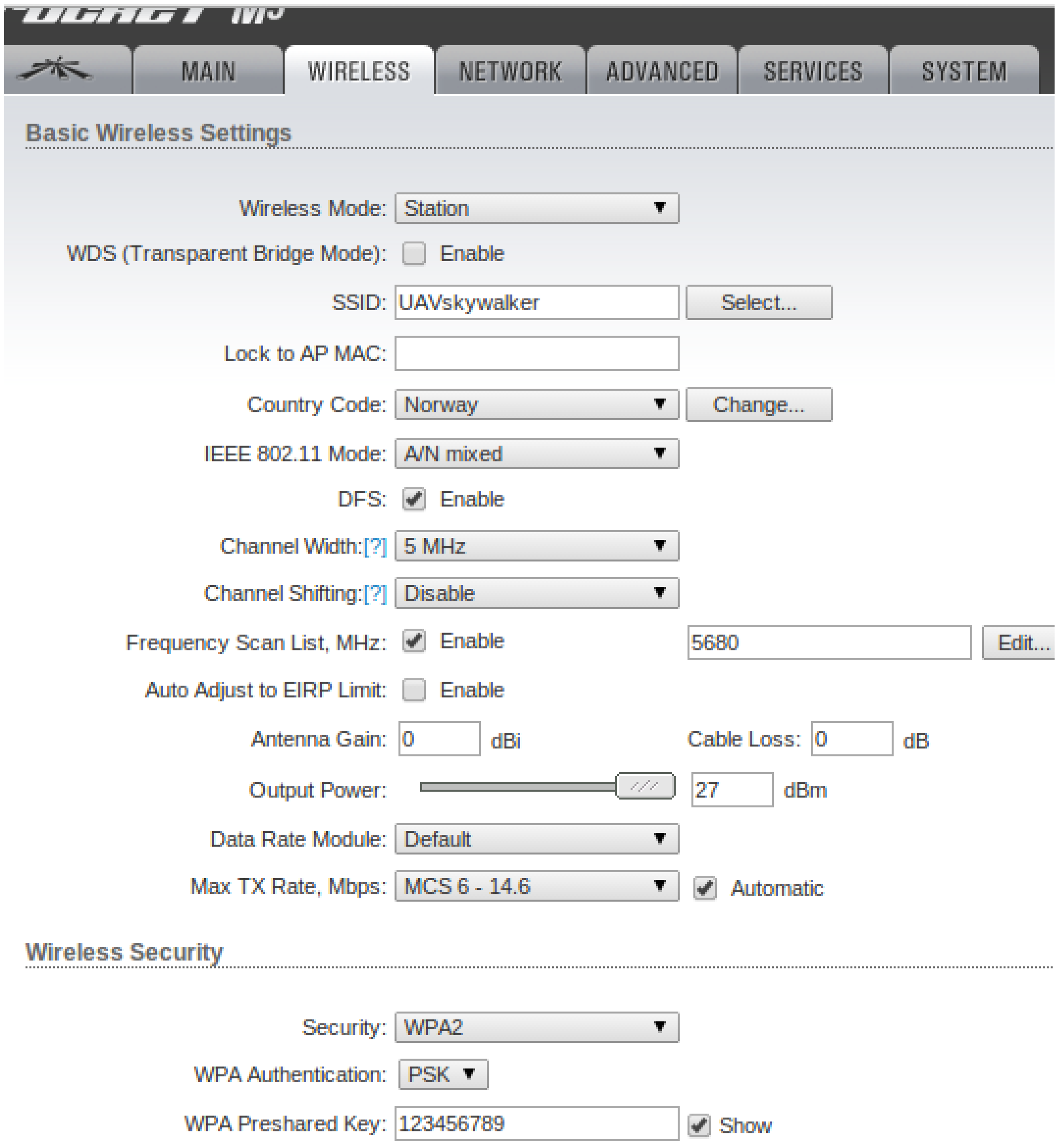
The Rocket M5 is placed on top of the lid. It is screwed onto the it and the POE cable is taken around the edge and in under the lid. The Pandaboard is screwed on to the lid inside the box, hanging upside down. The power supply remains fastened on top of the Pandaboard and the PWM generator is screwed to the right side of the payload box. The battery and voltage adapter are placed directly on the payload mount, still fastened with hook-and-loop fasteners.

The cables are fastened to the box to avoid too much strain on the soldered joints. The finished implementation of the second hardware design is shown in Figure 9.7

9.5 Hardware of the User Interface

The user interface consists of software on a computer that communicates with the Pandaboard. That means that the hardware needed is, in addition to any computer with the linux operating system on it, a means to communicate with the UAV.

For lab testing, the Nanostation M5 described in Section 9.1is used to communicate with the UAV. It has an ethernet interface, and only a switch connecting the Nanostation M5 and the computer with the user interface on it is needed. For field testing, an outdoor radio transceiver with long range is used, which is connected to another switch, providing both the UAV pilot and the user interface described in this Master Thesis with communication with the Pandaboard.



Basic Wireless Settings

Wireless Mode:	Station
WDS (Transparent Bridge Mode):	☐ Enable
SSID:	UAVskywalker
Lock to AP MAC:	
Country Code:	Norway
IEEE 802.11 Mode:	A/N mixed
DFS:	✓ Enable
Channel Width:[?]	5 MHz
Channel Shifting:[?]	Disable
Frequency Scan List, MHz:	✓ Enable 5680
Auto Adjust to EIRP Limit:	☐ Enable
Antenna Gain:	0 dBi Cable Loss: 0 dB
Output Power:	27 dBm
Data Rate Module:	Default
Max TX Rate, Mbps:	MCS 6 - 14.6 ✓ Automatic

Wireless Security

Security:	WPA2
WPA Authentication:	PSK
WPA Preshared Key:	123456789 ✓ Show

Figure 9.3: The properties in the *wireless*-page of user interface of the Rocket M5.

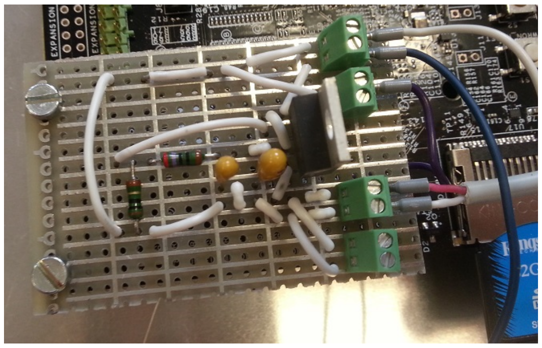


Figure 9.4: The power supply for the Rocket M5 and for the PWM-generator.

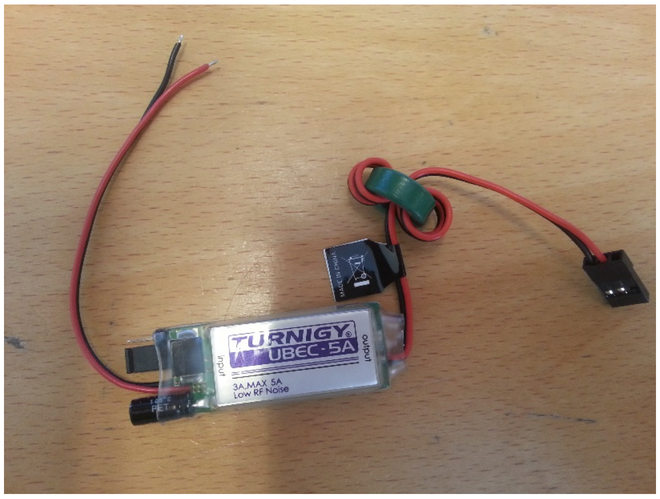


Figure 9.5: The voltage transformer used to supply the Pandaboard with 5 V

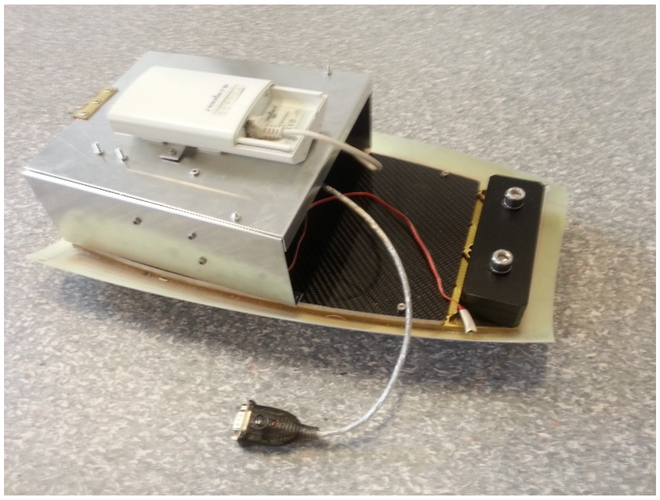


Figure 9.6: The hardware in the payload box with the second design.

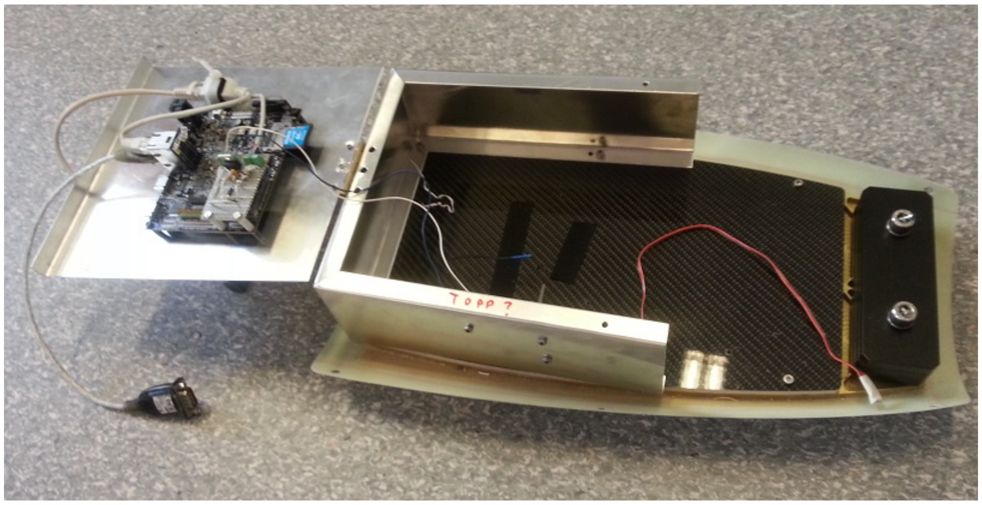


Figure 9.7: Overview of the finished payload box with the second design.

Part III

Experimental Procedure

Chapter 10

System Testing

To verify the system, it must be tested out with simulation and in real life. The verification of the system is done piecewise, including all hardware parts and different modules of the software. The software of the trajectory calculation is tested with hardware-in-the-loop (HIL) simulation and both the two different waypoint acceptance criteria as well as the two trajectory generation algorithms will be tested to evaluate which one is the more effective.

To test the system as a whole, it is possible to fly the Penguin B with the onboard payload mount on an old landing field on Agdenes, two hours by car away from NTNU. This chapter describes the hardware tests, the hardware-in-the-loop testing done on the UAV-laboratory as well as the field testing done on Agdenes. It includes tests of the two waypoint acceptance criteria and the trajectory generation algorithms, performed in the laboratory with HIL.

10.1 Hardware Tests

The hardware tests are conducted along with the completion of each hardware part, starting with the Pandaboard and ending with the communication with the Piccolo autopilot.

Test of the Pandaboard GPIO Pins

The Pandaboard has two 28 pin generic expansion connectors, see Figure 3.2, named J6 and J3. Some of these pins are reserved for other purposes, but some are general purpose input and output pins, like pin 17 on J6. This is the pin that controls the PWM signal generator for the drop mechanism, and the pin is active low.

The output of the Pandaboard is measured with a voltmeter on the GPIO pin 17 on J6. When the *DropLoad*-task tells the GPIO pin 17 on J6 to be set high or low, the output is measured with the voltmeter and the outcome is compared to the desired value. The registered value on the pin when the Pandaboard sets it to 1, is 1.9 V, while the registered value one the pin when the Pandaboard sets it to 0, is 0 V.

Test of the PWM-Generator and the Drop Mechanism

The PWM signal generator is described in Section 7.2. It is supposed to produce a pulse-width-modulated signal that makes the drop mechanism withdraw and push out the pin in the mechanism that connects the passive part to the active part. As further described in Section 7.2, the ATtiny85 requires an input signal that is at least 3 V, which is achieved with an inverting NPN transistor. The input of the transistor is the output signal from GPIO pin 17 on J6.

The voltage at the collector of the transistor is measured with voltmeter and found to be 5 V when the output of the Pandaboard is 0 and the transistor does not conduct current. When the Pandaboard output is 1, then the transistor conducts current to ground and the collector voltage is 0.

When the ATtiny85 receives a high signal on its input pin PB1 (see Figure A.5 in Appendix A), it produces a pulse-width signal on its output PWM pin, that goes gradually from 0 V to 1 V when measured with a voltmeter, and back to 0 V, repeatedly. When the input pin is low, the measured value on the PWM output pin is equal to 0 V. The oscilloscope testing is described in Section 7.2, and the period of the PWM signal was 16.5 ms, while the width varied between 60 μs and 3,3 ms.

When the drop mechanism is connected to the three output pins named *PWM Signal*, *Drop Mechanism Ground* and *Drop Mechanism Power*, see Figure A.5, of the PWM generator and the Pandaboard's pin 17 on J6 is used as input on the transistor for the PWM generator, then the Pandaboard is found to control the drop mechanism. When the ouput pin of the Pandaboard is set to 1, then the drop mechanism is calm. When the output pin of the Pandaboard is set to 0, then the drop mechanism is continually closing and opening its attachment pin.

Test Radio Communication

To test the radio communication used to communicate between the Pandaboard and user interface on the ground station, the Pandaboard is connected to the Rocket M5 with a twisted pair Ethernet cable and the ground station computer is connected to the LAN. The Nanostation M5 is also connected to the LAN. All settings are according to Section 9.1, and the ground station computer attempts to connect with the Pandaboard over SSH, using the IP of the Pandaboard. When the Pandaboard is accessible from the ground station computer without any serial connection, it shows that the Pandaboard is accessed over Ethernet. As the wireless LAN is turned off on the Pandaboard, only the radio communication remains.

Test the Power Supply Card

To supply the Rocket M5 with 12 V and the PWM-generator and drop mechanism with 5 V, a power supply card is built. It takes 12 V as input and has one output port with 12 V, to which the Rocket M5 is connected and which does not pass through any voltage transformer. The card also has one output port with 5 V that has passed through a direct voltage transformer, to which the PWM generator is connected. The power supply card is tested with a voltmeter, measuring the two outputs when an input voltage is applied to the card. With 12 V as input, 12 V are measured on the first port, and 5 V are measured on the second port.

Test the Communication between the Pandaboard and the Piccolo

To be able to guide the UAV on a trajectory, the Pandaboard needs to be able to send its generated waypoints to the autopilot. To know the position of the UAV at any time, the autopilot Pandaboard needs to be able to receive the state of the UAV from the autopilot. The physical interface between the Pandaboard and Piccolo autopilot is an RS232 cable. The Pandaboard uses DUNE to send Piccolo messages packed as Transmission Control Protocol (TCP) messages over this cable, by using the DUNE task called *Transports.SerialOverTCP*. This way, DUNE sends Piccolo messages between the two devices, and there is a DUNE Task on the Pandaboard that translates these messages to IMC messages and translates other IMC messages to Piccolo messages.

This communication is tested by connecting the Pandaboard to the Piccolo's COM 1 port (see Figure A.4 in Appendix A) on the HIL simulator and on the field test UAV respectively. The Pandaboard is programmed to print received IMC::EstimatedState messages on a terminal, when the board is connected to the ground station with an SSH connection. The IMC messages are printed with correct values. On the HIL simulator, IMC messages are also sent back to the autopilot with waypoints for it to follow. The simulated UAV then flies according to the waypoints.

10.2 Hardware In the Loop Testing

The HIL-testing is carried out by connecting the Pandaboard to a Piccolo Autopilot through a serial RS-232 link. The Piccolo Autopilot is connected to a computer that runs a Hardware In the Loop simulation software from Cloud Cap Technologies to simulate the flight. The simulator simulates the Penguin B UAV, providing the autopilot with simulated values for the state of the UAV. This includes GPS position and wind strength and direction. That computer is also connected to a Piccolo ground Station, which in turn communicates with the Piccolo Autopilot. See Figure 10.1 for an illustration of the HIL setup.

The Pandaboard communicates with the Piccolo Autopilot over a TCP protocol on a serial link. The serial link is plugged into the Pandaboard's USB port through a serial-to-USB-adapter.

10.2.1 Test of Drop on Target

To test the communication between the Pandaboard and the Piccolo Autopilot and the hardware of the drop mechanism, a small program is made and loaded on the Pandaboard. It is supposed to communicate with a user interface implemented in DUNE and receive a target position on which it should drop its payload. This target position will result in a drop position, calculated by the module described in Section 8.2.1.2. The program is supposed to receive signals from, but

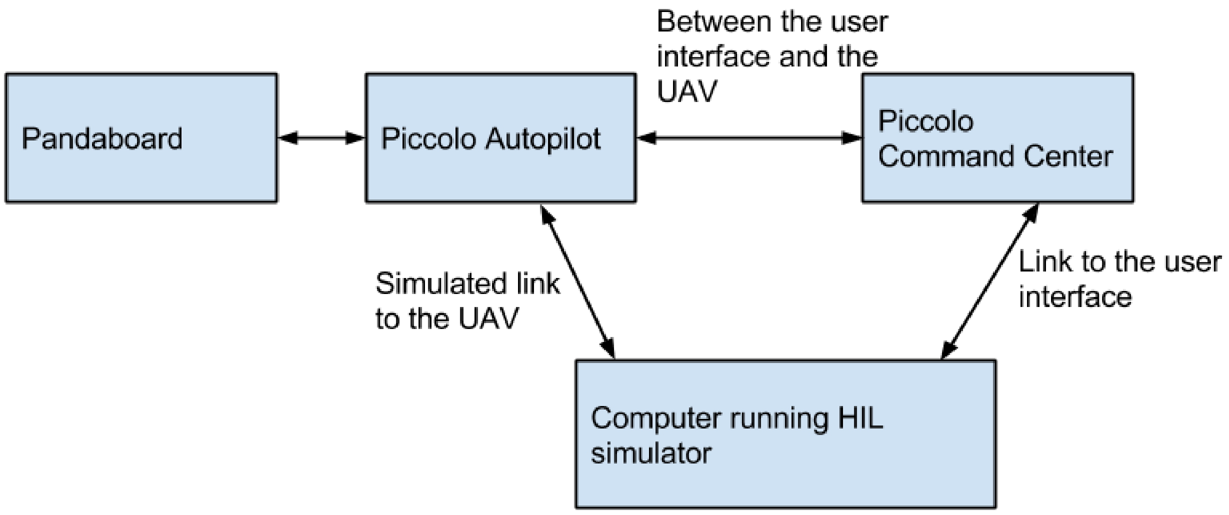

Between the user interface and the UAV
Pandaboard
Piccolo Autopilot
Piccolo Command Center
Simulated link to the UAV
Link to the user interface
Computer running HIL simulator

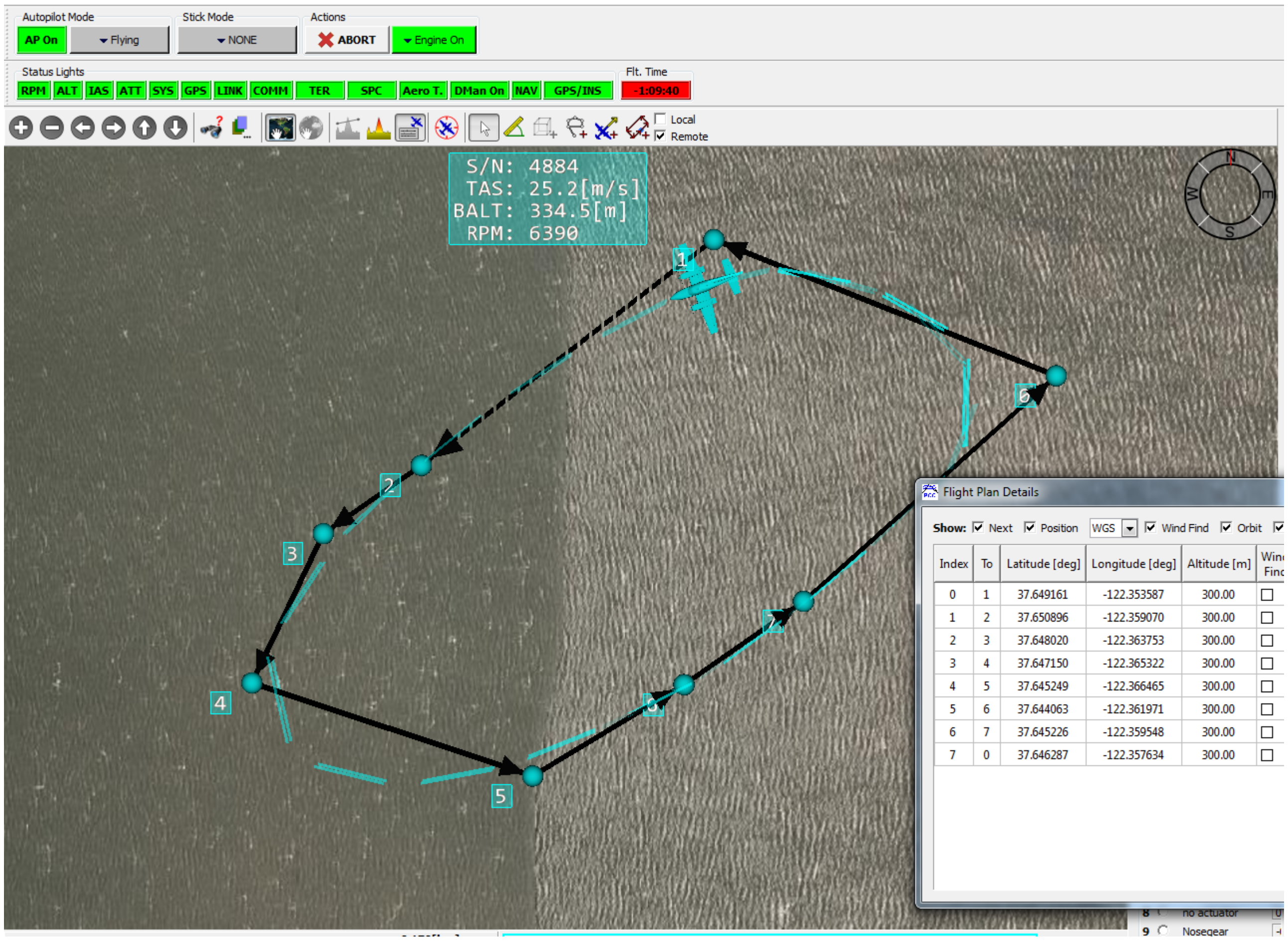
Figure 10.1: The HIL setup: This chart illustrates which units talks to which.

not return any messages to, the Piccolo Autopilot, reading the position of the simulated UAV. If the UAV is within a given radius of the target, it should signal to its drop mechanism to release the payload. Upon the exit of this target radius, the drop mechanism should signal not to try to release its payload any more.

To test the program, the simulated UAV fly a semi rectangular path where one of its way-points is the target where it should drop its payload, see Figure 10.2. The wind of the simulator is turned off.

10.2.1.1 First Test

The states of the simulated UAV are set to be measured once per second, and the radius of the acceptance sphere is set to 20 meters. The indicated air speed (IAS) is set to be 28 m/s. It is registered that although the system worked correctly most of the time, the UAV does not release its payload every time it passes the point of release. The DUNE Ground Station is then told to fetch the IMC messages that holds the GPS position of the simulated UAV, and display the position on a terminal. The distance between the UAV and the point of release is also displayed. This shows that sometimes the simulated UAV is as close as 21 or 22 meters away from the point of release, but the next measurement shows that it has passed the point of release and is 27 meters away. We conclude that as the frequency of the messages holding the state of the piccolo



S/N: 4884
TAS: 25.2[m/s]
BALT: 334.5[m]
RPM: 6390

Flight Plan Details

Show: ✓ Next ✓ Position WGS ✓ Wind Find ✓ Orbit ✓

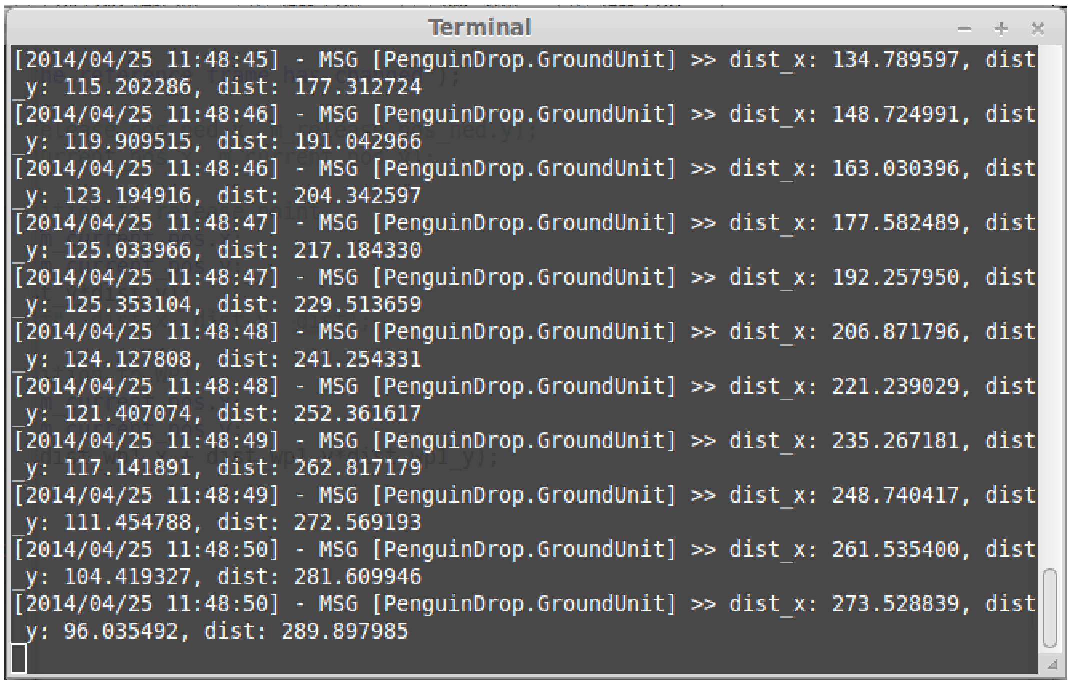
Index	To	Latitude [deg]	Longitude [deg]	Altitude [m]	Wind Find
0	1	37.649161	-122.353587	300.00	☐
1	2	37.650896	-122.359070	300.00	☐
2	3	37.648020	-122.363753	300.00	☐
3	4	37.647150	-122.365322	300.00	☐
4	5	37.645249	-122.366465	300.00	☐
5	6	37.644063	-122.361971	300.00	☐
6	7	37.645226	-122.359548	300.00	☐
7	0	37.646287	-122.357634	300.00	☐

Figure 10.2: The HIL simulator's test flight. Point 2 marks the point of release.

is 1 Hz, and the UAV fly with a speed of 28 meters per second, then a radius of 20 meters around the point of release will not be big enough. Sometimes the Pandaboard did not know that it was directly over the point of release, and therefore did not release its payload. The solution is to increase the frequency of the measurements: Using the setting called "Comms no flow (fast)". In addition to this, the radius is set to 30 meters, to make sure that the payload will be released every time the UAV passed the point of release.

10.2.1.2 Second Test

With 3 measurements of the UAV state per second and a radius of the acceptance sphere of 30 meters, in addition to an indicated air speed (IAS) that still is 28 m/s, the system is HIL-tested again. This time, the position of the UAV and the distance to the point of release is displayed in the terminal window running the ground control, every time a state estimation is received from the Piccolo and the UAV is within acceptance sphere (see Figure 10.3). The result is that approximately 4 measurements are taken every time the UAV passes the point of release, and that the payload is released successfully every time the UAV passes the area of interest.



```
[2014/04/25 11:48:45] - MSG [PenguinDrop.GroundUnit] >> dist_x: 134.789597, dist
_y: 115.202286, dist: 177.312724
[2014/04/25 11:48:46] - MSG [PenguinDrop.GroundUnit] >> dist_x: 148.724991, dist
_y: 119.909515, dist: 191.042966
[2014/04/25 11:48:46] - MSG [PenguinDrop.GroundUnit] >> dist_x: 163.030396, dist
_y: 123.194916, dist: 204.342597
[2014/04/25 11:48:47] - MSG [PenguinDrop.GroundUnit] >> dist_x: 177.582489, dist
_y: 125.033966, dist: 217.184330
[2014/04/25 11:48:47] - MSG [PenguinDrop.GroundUnit] >> dist_x: 192.257950, dist
_y: 125.353104, dist: 229.513659
[2014/04/25 11:48:48] - MSG [PenguinDrop.GroundUnit] >> dist_x: 206.871796, dist
_y: 124.127808, dist: 241.254331
[2014/04/25 11:48:48] - MSG [PenguinDrop.GroundUnit] >> dist_x: 221.239029, dist
_y: 121.407074, dist: 252.361617
[2014/04/25 11:48:49] - MSG [PenguinDrop.GroundUnit] >> dist_x: 235.267181, dist
_y: 117.141891, dist: 262.817179
[2014/04/25 11:48:49] - MSG [PenguinDrop.GroundUnit] >> dist_x: 248.740417, dist
_y: 111.454788, dist: 272.569193
[2014/04/25 11:48:50] - MSG [PenguinDrop.GroundUnit] >> dist_x: 261.535400, dist
_y: 104.419327, dist: 281.609946
[2014/04/25 11:48:50] - MSG [PenguinDrop.GroundUnit] >> dist_x: 273.528839, dist
_y: 96.035492, dist: 289.897985
```

Figure 10.3: The output on the DUNE Ground Station

The data containing the distance between the UAV and the target are plotted, showing a graph displayed in Figure 10.4. This graph shows that the UAV measures its position once while entering the circle around the point of release, twice close to the actual point of release and once while exiting the circle, see Figure 10.5. The registered positions are approximately 12 meters apart. The plot of all of the results shows that although the two closest measurements strays somewhat from the center of the circle, the closest point on every turn is always within 7 meters of the center of the circle.

To have some margin, the radius for the next test is set to 10 meters. To simplify matters, the program on the Pandaboard is modified to receive the acceptance radius from the DUNE Ground Station.

$$radius < \frac{IAS}{f} \tag{10.1}$$

10.2.2 Test of the Correctness of the Different Acceptance Criteria

To evaluate the accuracy of the two different acceptance criteria discussed in Section 6.3, a test is conducted. The simulated UAV will fly in a predetermined pattern several times and then log the first measured GPS positions that is within the acceptance circle, and the first one that is over the acceptance half plane. The measured size is then the distance between the waypoint itself and the position where the waypoint is registered.

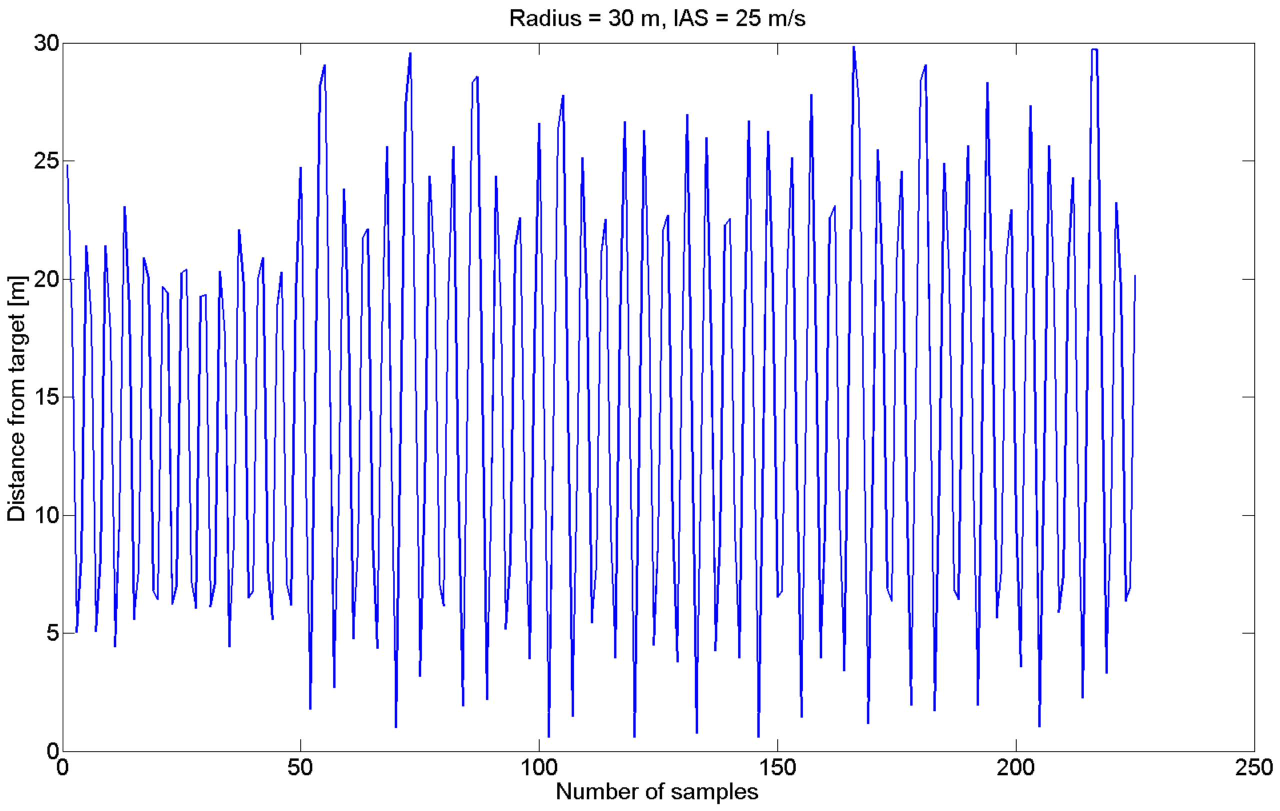


Figure 10.4: The distance between the UAV and the target when the UAV released its payload, repeated several times.

The program is started and the simulated UAV in the Piccolo Command Center is set to track a trajectory consisting of 6 waypoints shaped as a rectangle, see Figure 10.2. One of the waypoints of this flight plan will be tested for acceptance, and this waypoint will be referred to as the target position. An acceptance radius of 10 meters is used to accept the target position, as well as a half plane constructed as described in Section 4.4. Every time the simulated UAV passes the target position, the distance between the target position and the first GPS positions registered by the autopilot within the acceptance sphere is written to a text file. The distance between the target position and the first GPS position registered after the halfplane is crossed is recorded in the same file. The results are shown in Chapter 11.

10.2.3 Test of Straight Line Approach

To test the Straight Line Approach, the calculation tasks on the Pandaboard were adjusted, see *Version 2*, Section 8.2.1.2. The settings remained as they had been in the previous testing of the drop on target: The IAS was set to be 28 m/s and the measurements sending frequency was set to

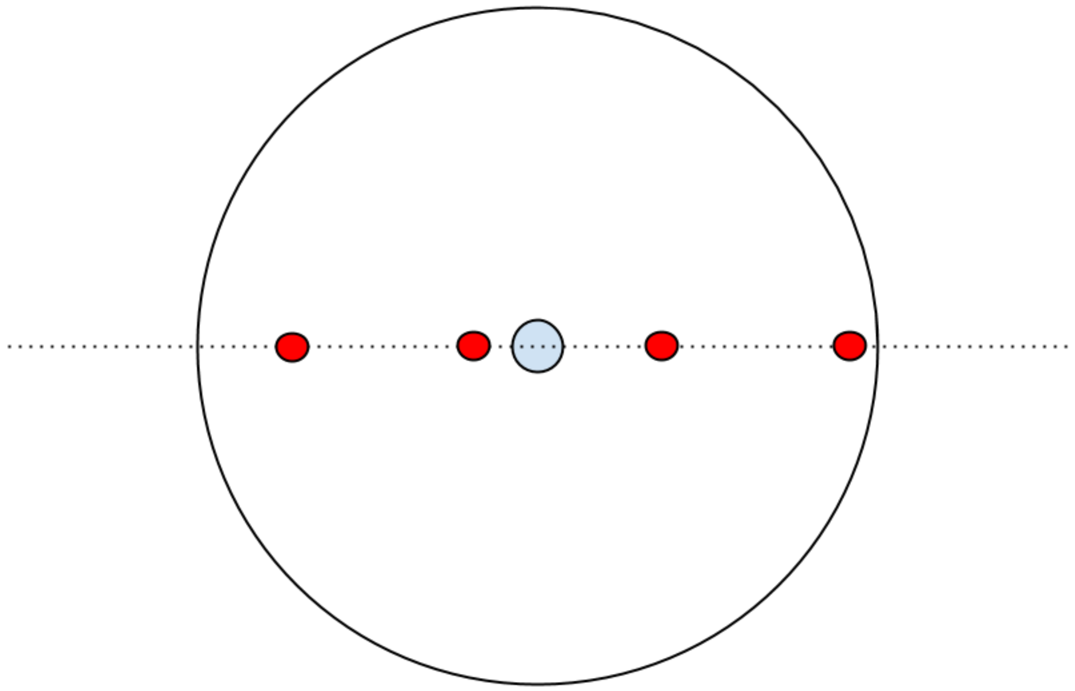


Figure 10.5: The red circles mark the registered GPS position of the UAV and the big circle is the acceptance sphere in two dimensions. The red circles appear approximately every 12 meters

"Comms no flow (fast)". The programs on the Pandaboard and the ground station were started, and the target position and tolerance radius were sent to the Pandaboard from the ground station.

The wind on the simulator was turned off. The target position was set close to the starting position of the simulated UAV, which means that the UAV did not have to travel far to reach its destination. In fact, the simulated UAV loitered around the target position as the program was started. As soon as the Pandaboard received the target position and tolerance radius, three points were shown on the Cloud Cap simulator screen, see points 130, 131 and 132 in Figure 10.6. Point 130 is the *far away*-point of the Straight Line Approach, while point 131 is the PoR, the point of release. Point 132 is automatically set 400 meters away from point 131 along the same line. This is fortunate because it forces the trajectory straight through the PoR.

The simulated UAV follows the waypoints smoothly, as the blue line in Figure 10.6 shows. In

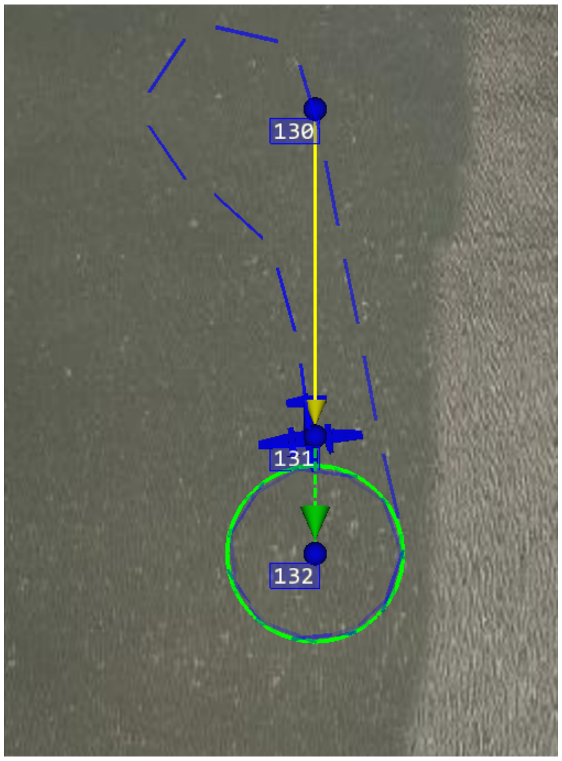


Figure 10.6: HIL simulation of the Straight Line Approach.

Section 6.2.3, it is said that the *far away*-point is placed as far away as necessary, to give the UAV space to turn around after point 132 and still meet point 131 from the correct angle. Figure 10.6 shows that the approach angle on point 131 is not exactly equal to the desired approach angle, which is indicated by the line from point 130 to point 132. The accept criteria used in this test was an acceptance radius set to 30 meters.

10.2.4 Test of the Accuracy of the Straight Line Approach

To find out how accurate the Straight Line Approach is, a test it conducted where the state of the simulated UAV is measured at the time when it releases its payload. That means the position when the point of release is accepted by the calculation tasks and the drop mechanism is activated. Both the NED frame position of the UAV and the heading of the UAV are measured. This measurement is repeated each time the UAV releases its payload. Every time the UAV has finished the Straight Line Approach algorithm, it sends an IMC message to the ground station. The ground station then resends the target, and the Straight Line Approach directs the simulated UAV to the point of release once more. The test is running without wind and with an indicated air speed (IAS) of 28 m/s. This simulation is continuously running overnight and all results are presented in Chapter 11.

10.2.5 Test of Adapted Dubins Path

To test the Adapted Dubins Path, the calculation tasks on the Pandaboard are once more adjusted. The tasks are set to follow Version 3 described in Section 8.2.1.2: Two tasks, called *StraightLineApproach* and *AdaptedDubinsPath*, both receive the IMC::Target message from the ground control, and the label called *DestinationEntity* decides which task is the receiver of the message. That task begins its calculations and communicates with the *Piccolo*-task, while the other task remains passive.

The IAS is still set to be 28 m/s and the measurements sending frequency is still set to "Comms no flow (fast)". The simulated UAV starts by loitering somewhere near the PoI, before the target is sent from the ground unit. The distance between the first circle of the algorithm and the second circle is set to 5 * IAS.

The wind is not turned on. The test starts with the simulated UAV loitering around a point nearby the PoR. This loiter circle is the one at the bottom, indicated in blue, in Figure 10.7. As soon as the Pandaboard receives the IMC::Target message from the ground station, it calculates a circle and sends the loiter command to the autopilot. This is the blue circle in Figure 10.7 that contains the point 131. The simulated UAV follows this circle until it crosses a half plane indicating the start of the straight line. The straight line waypoints are sent to the Piccolo and the simulated UAV starts tracking this line. When the crossing of a half plane indicates that the simulated UAV has reached the second circle, this circle is calculated and directions are given to the autopilot. The UAV follows this circle until it is headed directly towards the PoR, then follows a straight line to the point of release. After PoR, a last waypoint is sent to the autopilot, causing the UAV to approach point 131 with approximately correct angle.

This test is repeated, and every time with success. The solution seems very successful.

10.2.6 Test of the Accuracy of the Adapted Dubins Path

To find out how accurate the Adapted Dubins Path is, a test is conducted where the state of the UAV is measured at the time when it releases its payload. Both the NED frame position of the UAV and the heading of the UAV are measured. This measurement is repeated each time the UAV releases its payload. Every time the UAV has finished the Adapted Dubins Path algorithm, it sends an IMC message to the ground station. The ground station then resends the target, and the Adapted Dubins Path directs the simulated UAV to the point of release once more. The test is running without wind and with an indicated air speed (IAS) of 28 m/s. The distance between the first circle of the algorithm and the second circle is set to 10*IAS. This simulation is continuously running overnight and all results are presented in Chapter 11.

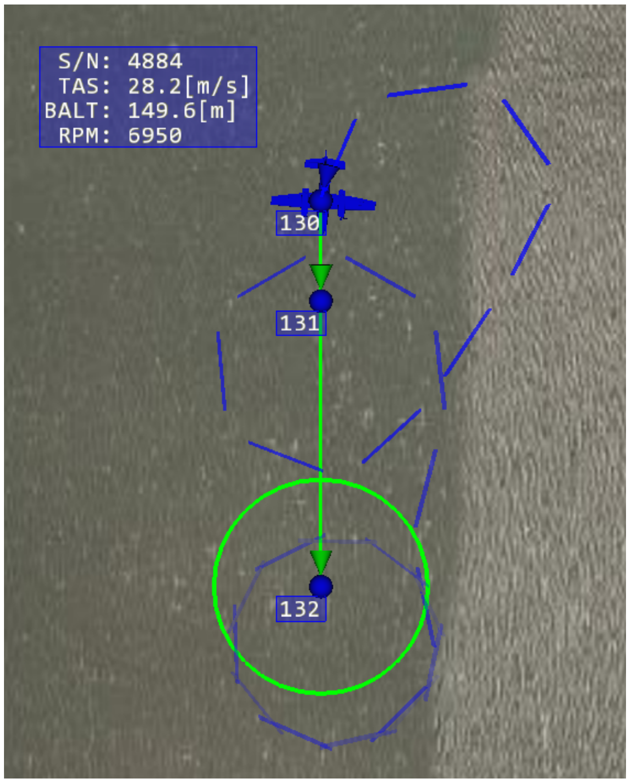


Figure 10.7: HIL simulation of the Adapted Dubins Path.

10.3 Field Testing

The field testing on Agdenes is restricted by the weather, causing all flying to cease if the wind is too strong. Thus, the test dates are unpredictable and testing may have to be conducted on short notice with scarce planning. Besides, as the landing field is very far away, impulsive testing is not possible. As there are several other groups also waiting to test their systems, the result is that there are not many test dates available for each group and it is not preferable to miss an opportunity to test. This means that as much as possible of the system should be ready to be tested each time, but also that it is not realistic to have the opportunity to test everything we want. Between the tests, the UAVs to be tested are kept in a small hangar

10.3.1 Test of Drop on Target from the Ground

The first test on Agdenes is conducted on the 9. of April 2014. It is raining very lightly and it is almost no wind. As some of the equipment on the Penguin B, belonging to the Piccolo Autopilot, is not fastened correctly, and because the hardware design of our payload is not according to the latest restrictions at this point, the test can not be carried out in the air. Instead, the Punguin B is manually controlled by the UAV pilot, while driving on the landing field. A test target point is found to be (63.628806 , 9.727369), approximately 20 meters from the start spot. The radius around the point of release is set to be 5 meters. The UAV is driven slowly, so it drops its payload correctly, approximately 5 meters before it reaches the target position, just as it enters the acceptance circle.

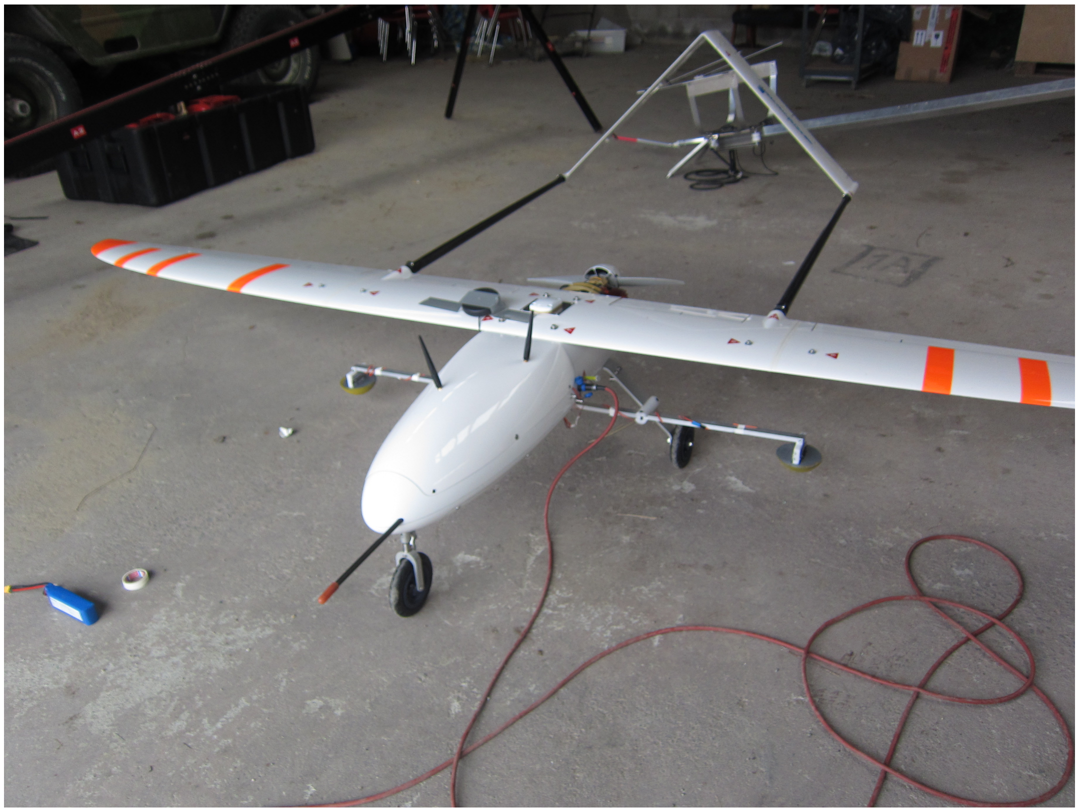


Figure 10.8: The Penguin B with the drop mechanisms mounted under its wings and the antennas on top of the fuselage

After refastening the drop mechanisms on the UAV, they are stress tested. On this first test, the distance between the bottom of the payload to be dropped and the ground is approximately 5 centimetres. As the ground is very bumpy, the drop mechanism can get caught on a tuft of grass and potentially crash the UAV during a landing. However, the stress test shows no problems with the design. To be on the safe side, though, it is decided that the drop mechanism should be mounted higher up one the legs of the UAV. But although the landing would not be troubled by the drop mechanisms, the take-off might. To get the Penguin B into the air, it is possible to take off with enough speed on a landing strip, but the preferred way is to shoot the Penguin B out with a catapult, which used compressed air to shoot the UAV. The Penguin B in the catapult is shown in Figure 10.10. The bar carrying the drop mechanisms is attached to the landing gear, and as shown in the picture, it is between the front and rear pair of arms on the catapult. This was noted as a possible problem and testing would have to be done on the construction to validate it.

However, as the drop mechanism bar was tested on the catapult with a testing rack, it turned

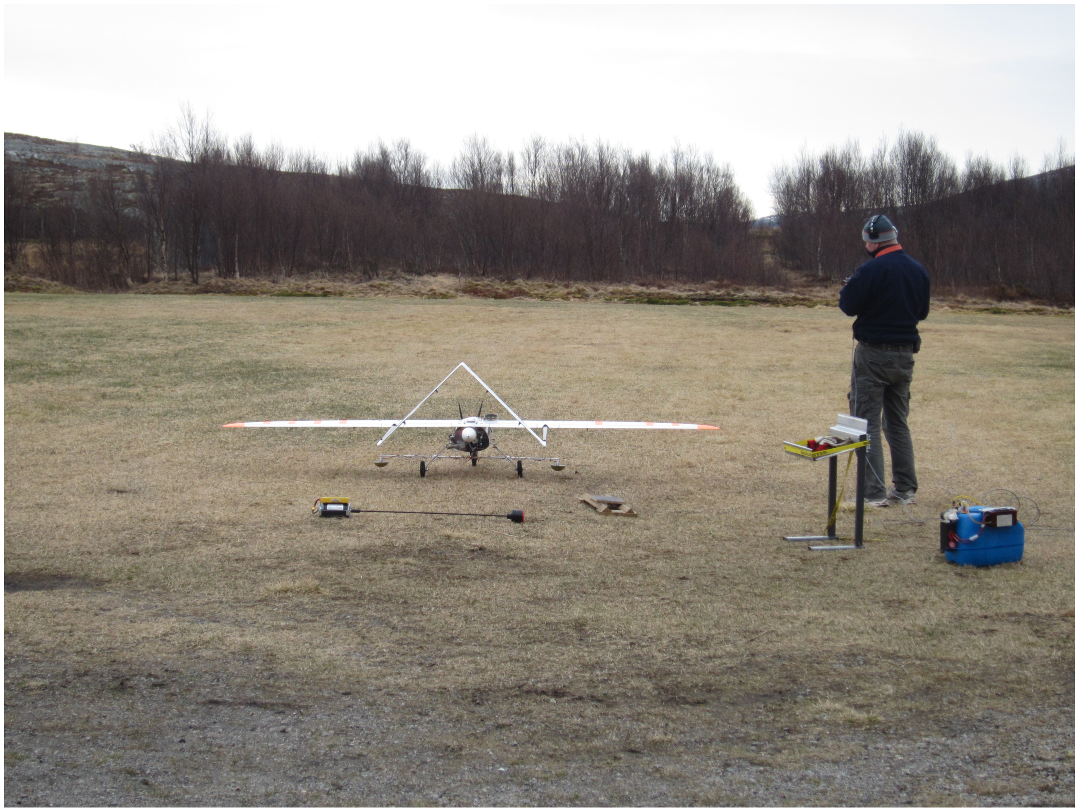


Figure 10.9: The Penguin B driven manually by its pilot

out that the placement was good enough. All in all, this was a successful test that showed that both the hardware and most of the software except for the calculation tasks, worked well.

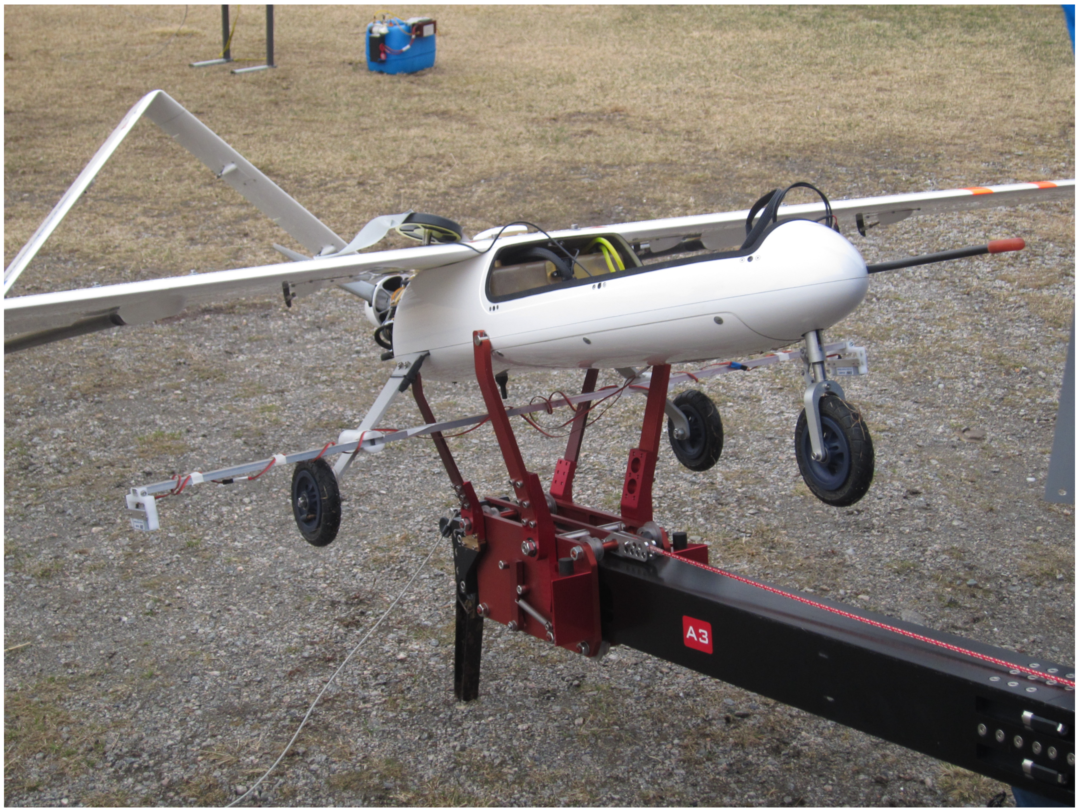


Figure 10.10: The Penguin B in its shoot out catapult

Part IV

Results and Conclusions

Chapter 11

Results

After the software has been coded and the hardware has been built, the system was tested. This is described in Chapter 10. The results of the testing are the results of this Master Thesis, and include both qualitative results and some quantitative results. These are described in this chapter.

11.1 Results of the Hardware Tests

Although the description of the hardware that is used and build in this thesis is placed in different chapters in Part *System Description, Planning and Implementation*, the tests of the hardware are conducted in Chapter 10. The tests are mostly simple tests that verifies that the system is working as it should when asked to, but they do not exclude potential errors in the system. The results of the hardware tests are presented in Table 11.1:

Test	**Result**
Test the Pandaboard's GPIO pin	Working Correctly
Test the PWM generator and the drop mechanism	Working Correctly
Test the radio communication	Working Correctly
Test the power supply card	Working Correctly
Test the communication between the Pandaboard and the Piccolo autopilot	Working Correctly

Table 11.1: The Results of the Hardware Tests

11.2 Results of the Test of the Correctness of the Different Acceptance Criteria

In Section 10.2.2, the two acceptance criteria for a waypoint are tested. The first GPS positions that is within the sphere created by the acceptance radius and the first GPS position after cross-

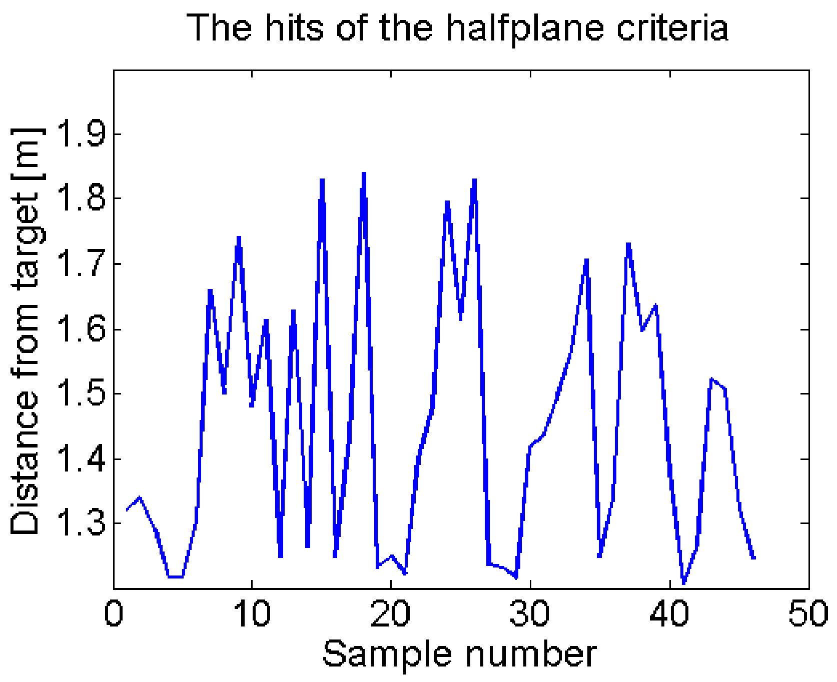


Figure 11.1: The distance from the target to the first GPS position after the half plane is crossed.

ing the half plane of the target, are registered. The distance from these positions to the target position are calculated, and the distances are shown as plots in Figure 11.1 and Figure 11.2.

11.3 Resulting Trajectory Generation Algorithms and their Accuracy

Resulting Trajectory Generation Algorithms

One task in this Master Thesis was to develop an algorithm that is used to control the UAV trajectory to the optimal deployment point. This was mainly Simen Fuglaas' responsibility, but algorithms were developed jointly by him and the author. The algorithm for the Straight Line Approach is given in Table 11.2 and the algorithm for the Adapted Dubins Path is given in Table 11.3.

Accuracy Tests of the Trajectory Generation Algorithms

The accuracy tests of the trajectory generation algorithms produced the deviation between the calculated point of release, and the state of the simulated UAV when the calculation tasks accepted the point of release. Both the distance from the measured point of release to the real point of release, and the difference in heading between the simulated UAV and the ideal approach angle, were calculated. The results were plotted and are shown in Figure 11.3 and Figure

```
Algorithm for following the Straight Line Approach
Input: GPS and wind measurements from the autopilot
Require:
1: measure wind
2: while no wind measurement
3:         keep measuring
4: end while
5: calculate optimal PoR
6: calculate Far Away point
7: while calculations not done
8:         keep calculating
9: end while
10: send waypoints to the autopilot
11: track Far Away point
11: while not entered Far Away point
12:         track Far Away point
13: end while
14: track PoR
15: while not entered PoR
16:         track PoR
17: end while
18: release the payload
19: start loitering around PoI
```

Table 11.2: Algorithm for following the Straight Line Approach

```
Algorithm for following the Adapter Dubins Path
Input: GPS and wind measurements from the autopilot
Require: UAV loiters around the point of impact
1: measure wind
2: while no wind measurement
3:     keep loitering
4: end while
5: calculate optimal PoR
6: calculate Adapted Dubins Path
7: while calculations not done
8:     keep calculating
9: end while
10: start on the first circle
11: while not reached the straight line between the circles
12:     keep loitering
13: end while
14: start on the straight line between the circles
15: while not reached second circle
16:     if current waypoint tracked
17:         send next waypoint
18:     end if
19: end while
20: start on the second circle
21: while not reached the straight line from second circle to PoR
22:     keep loitering
23: end while
24: start on the straight line from second circle to PoR
25: while not reached PoR
26:     if current waypoint tracked
27:         send next waypoint
28:     end if
29: end while
30: release the payload
31: start loitering around the PoI
```

Table 11.3: Algorithm for following the Adapted Dubins Path

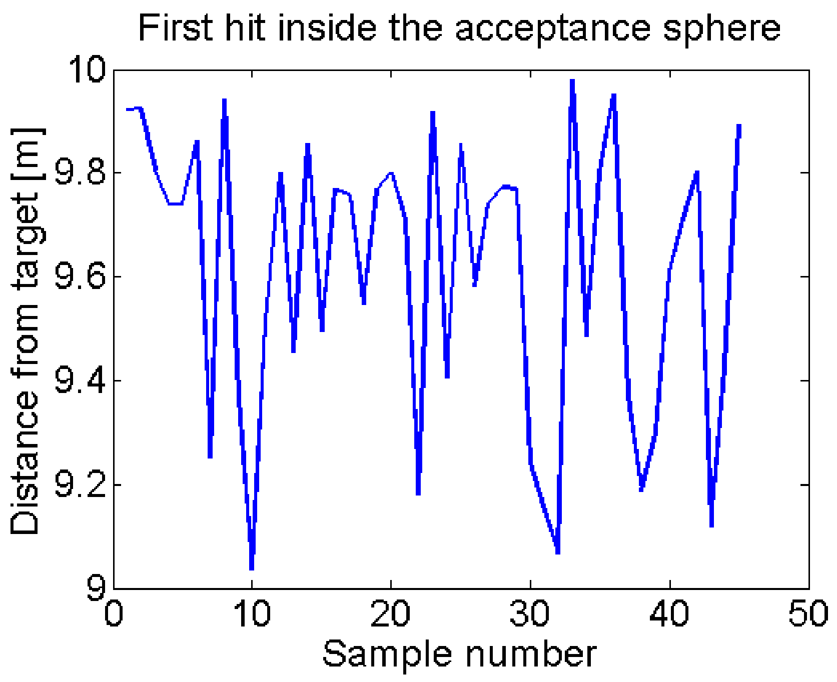


Figure 11.2: The first hit inside the acceptance sphere

11.4.

For the Straight Line Approach, 615 samples were measured. The distance between the calculated PoR and the accepted PoR varied between 12.92 meters and 13.9558 meters, with a mean value of 13.3401 meters. The difference between the calculated approaching angle at the PoR and the actual approaching angle at the PoR varied between 6.6091 degrees and 6.9008 degrees, with a mean of 6.7326 degrees.

For the Adapted Dubins Path, 493 samples were measured. The distance between the calculated PoR and the accepted PoR varied between 0.1701 meters and 4.9853 meters, with a mean value of 1.2911 meters. The difference between the calculated approaching angle at the PoR and the actual approaching angle at the PoR varied between -0.6910 degrees and 0 degrees, with a mean of -0.3612 degrees.

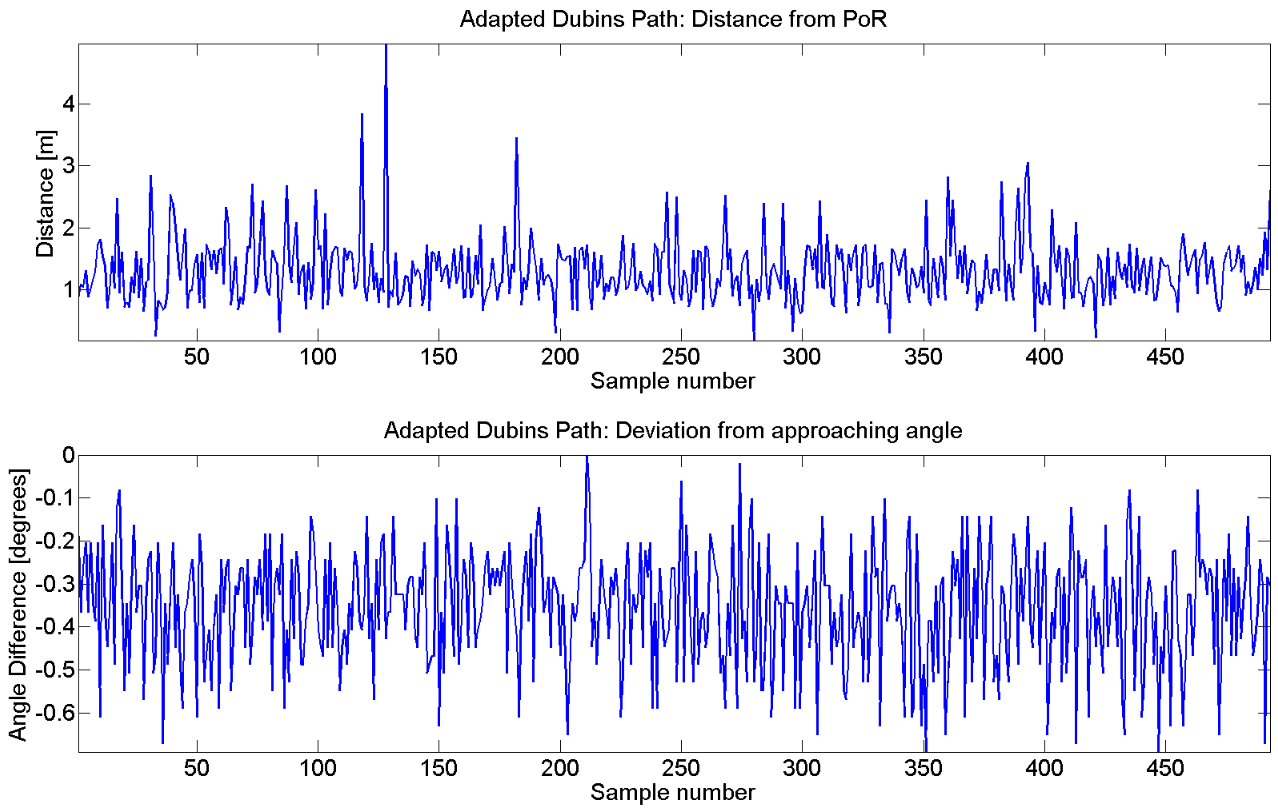


Figure 11.3: The difference between the state of the simulated UAV and the calculated PoR and approach angle of the PoR for the Adapted Dubins Path.

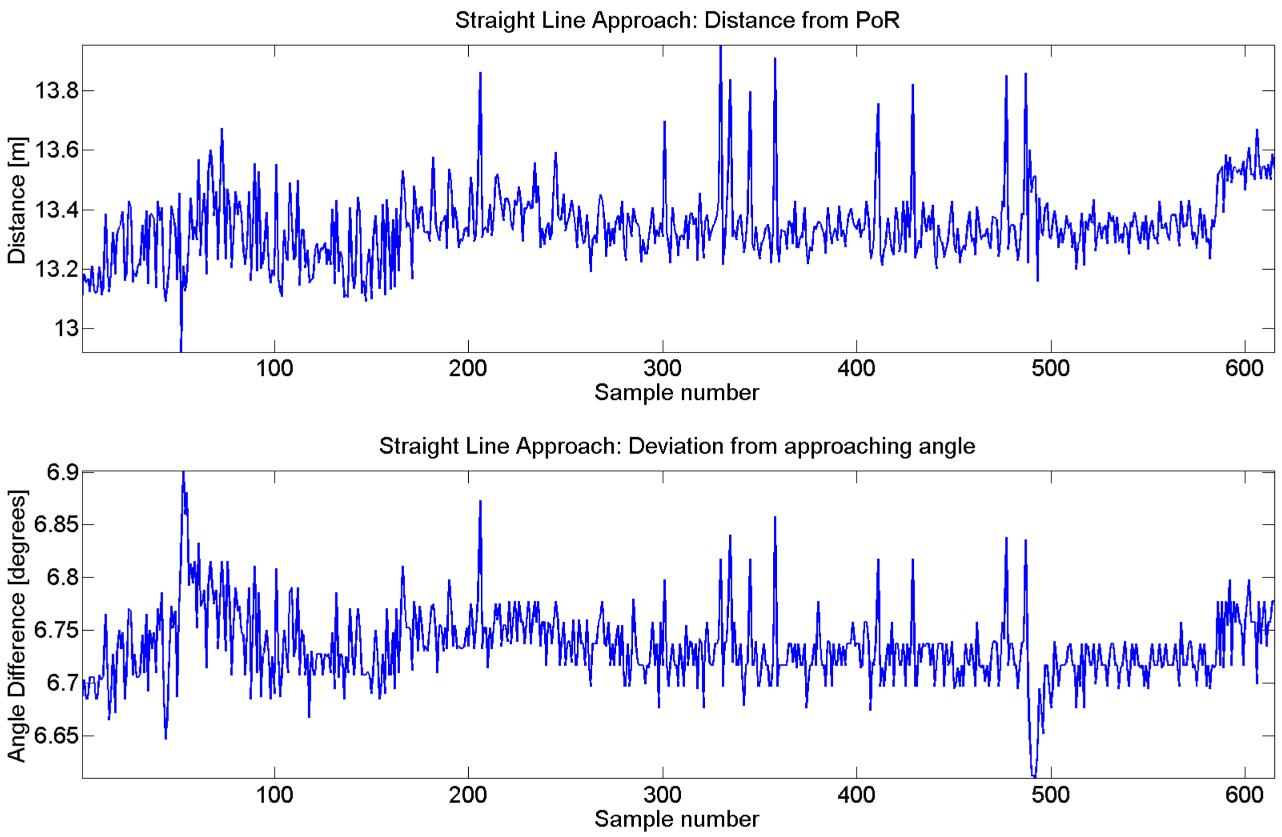


Figure 11.4: The difference between the state of the simulated UAV and the calculated PoR and approach angle of the PoR for the Straight Line Approach.

Chapter 12

Discussion

Work described in a Master Thesis does not always follow the wisest approach and sometimes the premisses for a decision change after more information is found. The results of such a work will also have to be commented, as results are only numbers while discussed results are information. This chapter contains a discussion of the decisions that have been influencing the project described in this Master Thesis. It also discusses the results described in Chapter 11 from the tests described in Chapter 10, the remarkably few field tests and the significance of the study.

12.1 Discussion of Choices Made

Some early choices in this Master Thesis have laid the foundation for the later work. Every time two alternatives are possible and one is chosen, an ocean of possibilities are no longer available. These choices are taken all the time, and all of them cannot be debated in this Master Thesis. However, some will be discussed in this section. The important decisions taken in the course of this project will be discussed in this section. There are also choices made prior to this project, which have yet more influence on the work. The most important ones will also be discussed in this section.

The Choice of Payload Deployment Method

The chosen payload deployment method was a free fall without any parachute. The choice of payload deployment method was thoroughly debated in Section 6.1, but that was prior to the testing. The field testing described in Section 10.3 did not show any problems with the free fall from the UAV, but the fall height was not very high either: As the UAV was only taxing on the landing strip, the sensor fell down approximately 5 centimetres.

As there has not been any deployment of the sensor from a flying UAV, it is difficult to say whether that choice was correct or not, but for the time being, the choice seemed like the most reasonable one.

The Choice of Approaching Angle on the Point of Release

The approaching angle on the point of release was chosen to be against the wind. That means that the UAV would fly straight north towards the point of release if the wind pointed straight south. The other possible solutions were the same direction as the wind, namely straight south if the wind pointed straight south, or arbitrary according to the wind. The first two choices were the simpler ones, and were therefore the two candidates, but the first alternative was eventually chosen. There is no apparent reason for choosing that alternative over the other candidate, as the wind force would be taken into account in the calculations of a drop point anyway. The test results did not draw any conclusion on whether that choice was correct or not, but then, no drop tests from the air were conducted. To test both alternatives would be a good approach for deciding which of the two approaching angles would be the better one. But for the time being, the choice seems like one out of two reasonable ones.

The Choice of Drop Mechanism

The chosen drop mechanism was the EFLA305 Servoless Payload Release. It was tested during the hardware tests, and stress tested during the field test. It proved to be robust enough on the field and to perform as it should on the laboratory drop tests. One drawback is that the passive part of the drop mechanism is connected to the sensor and will therefore be dropped together with the sensor. However, that is easily solved by making new ones with a 3D printer, that can make the new parts in plastic within short time. The other two candidates were not tested, which means that they could have been fantastic as well. On the other hand, the arguments of the debate in Section 7.1 were well-founded and just as existing now. Thus the choice seems like a very good choice, both based on the qualities of that drop mechanism and compared to the alternatives.

The Choice of Navigation Method

Early in the project, it was decided that the trajectory generation algorithm should navigate the UAV with waypoints instead of with a change of the heading of the UAV. This choice was based on the poor heading measurement availabilities on the Piccolo, and was therefore the only possible choice. However, with a different autopilot, or simply a better heading sensor for the UAV, navigation with heading change would be possible. That way, the autopilot could send the measured heading relative to the north axis of the NED frame. The Pandaboard could use this measured heading in a feedback loop and control the path with a control algorithm. With only waypoints, that is more difficult. With waypoints, each waypoint must be tracked and a deviation, between the intended waypoint and the actual waypoint that the UAV reaches, is inevitable.

At the time when the decision was made, it seemed like the use of waypoints was the only choice available for the Piccolo autopilot. As this was based on a statement from fellow student, it should have been tested before the choice was made. But as the discussion in Section 6.2.2 stated, it was not possible to test out the heading navigation on the autopilot in the field before developing a the trajectory tracking. However, the testing of the heading navigation with hardware in the loop was not considered. That could have been a sensible and time effective way of finding out whether or not the heading measurements were accurate enough. That way, a more informed choice could have been taken on the matter of navigation methods. And if the test actually showed that the heading measurements were not good, then the acquisition of a better sensor would have been a good action.

The Choice of Placement of the Drop Mechanism on the UAV

The drop mechanism was placed on the wheel assembly of the Penguin B, see Figure 10.8. This choice proved to be working well on the only field test, without causing problems when the UAV was taxing on the landing field. This placement could have caused problems when the UAV was thrown out from the catapult to take off. It could have gotten caught in the arms of the catapult, as they were stretched up during the start of the take off, and thrown down when the UAV was launched. However, tests done with a dummy bar showed that it would not be the case, see Section 10.3. Another problem could have been that the drop mechanism assembly bar could crash into the ground when the UAV landed. The bar did not touch the ground when the UAV was taxing, but it never landed with the bar fastened on it. That way, it was not tested and only assumptions could be made. The assumptions is that the bar will cause no problems, but that remains to be seen. All in all, there is no reason to doubt that this placement is worse than any other, although it remains to be tested out with a full field experiment.

The Choice of UAV

This choice was not made in this Master Thesis. The choice of UAV was the Penguin B from UAV Factory. It is a heavy weight UAV that has a long possible flight time due to large payload capacity: It can carry a lot of fuel. The payload capacity means that larger sensors can be dropped from the Penguin B, which makes it suited for the project described in this Master Thesis. The main drawback of this UAV is that it is very expensive. It means that on days when cheaper UAVs could defy the extreme weather, it cannot be risked with the Penguin B. A consequence of this is less flight time with the Penguin B than with other UAVs.

The only other possible fixed-wing UAV is one called X8, which is much lighter and cannot carry as much weight or fly as long as the Penguin B. On the other side, it is cheap enough to defy more wind than the Penguin B, and although it cannot carry as much weight as the Penguin B, tests could have been carried out on the X8. When most of the errors have been removed from the system, it could have been carried out on the Penguin B. That could have increased the amount of field tests, although it is difficult to foresee which difficulties would occur. All in all, the sensible choice of UAV was not the Penguin B, as it is too expensive.

The Choice of Autopilot

This choice was not made in this Master Thesis. The choice of autopilot is closely connected to the choice of UAV. The Penguin B is controlled with the Piccolo autopilot and the X8 UAV, which is mentioned in the previous paragraph, is controlled with an autopilot called Ardupilot. If it was, after all, possible to combine the two, a choice would have involved a thorough study of the properties of the two autopilots. Although this study has not been conducted by the author, the experiences with the autopilot were good and the only difficulties with the autopilot were moderate, basically caused by the interface between DUNE and Piccolo.

The Choice of Test Field

This choice was not made in this Master Thesis. The field tests were conducted on Agdenes, a small place two hours by car away from Trondheim. It is the test field of not only this project, but of all AMOS projects that involve fixed wing UAVs. This is a windy and hilly place, but it is close to the sea, which is important for some of the other AMOS projects. For the project described in this Master Thesis, however, the test field was not a good choice. The wind caused many test days to be cancelled, and this project could only conduct one field test. However, closeness to the sea is necessary for other projects, and that is most likely what causes the wind. That way, the test field could not have been changed, although it was not that well suited for this project.

The Choice of using DUNE

This choice was not made in this Master Thesis. DUNE was chosen to be an easy interface to the autopilot and to easily communicate between modules on the Pandaboard and between the Pandaboard and the user interface on the ground station. It worked perfectly and as intended, and the choice of using DUNE was a very good one for this project.

Of course, some time has had to be invested in learning to use DUNE, but that time has been returned by the advantages DUNE offers: The communication and the modularization were well suited for programs on an embedded computer on a UAV. DUNE also makes it very easy to change the different equipment of the system, like vehicle or autopilot. If there had been no such framework, then the exchange of UAVs between the Penguin B and the X8, that is mentioned earlier in this discussion, would have been much more difficult and demanded much more work. Just like DUNE has an interface to the autopilot currently in use, Piccolo, it has a similar interface to the other autopilot, Ardupilot, that is mentioned earlier in this discussion. This possible change is also facilitated by DUNE and its modular construction.

The only exception on the good experiences with DUNE was the Piccolo interface, that sometimes had small bugs that had to be dealt with. However, they were minor and manageable. All in all, the choice of using DUNE has been a good one.

12.2 Discussion of Results

The results presented in Chapter 11 are of different quality: Some are results of a development, some are results of tests that are supposed to verify the hardware, while some are results meant to find the better solution out of a selection of solutions. The results will be discussed in the same order as they were presented in Chapter 11.

12.2.1 Discussion of the Hardware Test Results

The test results of the hardware tests show that the hardware works as is should when asked to. Some of the equipment, like the power supply card and the PWM generator, are simple constructions and not likely to behave in any other way than supposed to. The GPIO pins on the Pandaboard are also highly logic mechanisms and behave according to the programs onboard the Pandaboard. The drop mechanism, however, did not come with any data sheet, and was only found to react upon a PWM signal between 0-1 V by trial and error. So although the system works perfectly in the tests, there may be hidden behaviour that has not been discovered.

The method chosen for deploying the target with the drop mechanism, to continuously pull in and push out the pin connecting the passive to the active part, is also discussable. This is described in Section 7.2.3. It does not give any immediate drawbacks except for the disturbance while testing the system, but it seems a bit pointless. The argument for choosing that method was that it made sure the sensor was actually deployed from the UAV: If the first attempt to release the payload was unsuccessful, then the next ones could add redundancy to the system. A more sensible solution, though, would be to signal to the drop mechanism to continuously open its lock when the sensor should be released. So instead of commanding the drop mechanism to open, then close, then open, then close, it is only commanded to open, then open, then open. After all, there is no use in closing the lock in the period when the sensor should be released. Then, after a given distance, the drop mechanism could be signalled to close the lock. In the unlikely scenario that the sensor was not dropped when it should, the closing of the lock prevents it from falling down somewhere else.

The radio communication test is also a simple test to see if he system works works in reality as it should according to the theory. The results show that the equipment works as it should, and there is no reason to believe anything different. The communication between the Pandaboard and the Piccolo autopilot is also found to be working correctly on the basis of simple tests. It would be very difficult to discover errors on the communication as long as the results are as expected for a working communication. However, it is not tested to send IMC messages from the Pandaboard to the autopilot on the field test UAV. That is because there were not enough field tests, see Section 12.3 for the full discussion on this subject. It would be reasonable to assume that if there is correct input from the autopilot to the Pandaboard, and as long as the Pandaboard is able to send messages to the autopilot in HIL simulations, then the Pandaboard is also able to send messages to the autopilot on a field UAV. But as long as that is not fully tested, these answers are only guesses and indications, and the result is only partly correct.

12.2.2 Discussion of the Correctness of the Different Acceptance Criteria

The correctness of the different acceptance criteria was tested, and the results were the distances between the registered GPS positions that satisfied the acceptance criteria, and the point of release. Figure 11.2 shows the distances between the target and the first registered GPS positions within the acceptance sphere, for all tests. Figure 11.1 shows the distances between the target and the first registered GPS positions after the UAV has crossed the half plane of that waypoint, for all tests.

It is obvious from the figures that the half plane gives better results than the acceptance sphere, as the distances when the half plane is used are far smaller than the ones using the acceptance sphere. The mean of the half plane acceptance positions is 1.4415 and the mean of the acceptance ball acceptance positions is 9.9785. However, the acceptance radius is set to 10 meters, so with high frequent measuring, then the first registered GPS position within the acceptance sphere should be near the perimeter. If the acceptance radius were smaller, then the distance from the target position to the first accepted GPS point inside the acceptance sphere, would also be much smaller. The minimum distance from the target to the first hit inside the acceptance sphere is 9.0362 meters. If the acceptance radius were smaller, say 9.0362 meters smaller, then the mean would be approximately 0.5725. That is smaller than the mean of the distances when the half plane criteria is used.

When the half planes are used, then all accepted GPS position are placed after the half plane is crossed, while in an acceptance sphere, the UAV is first approaching and then diverging from the target. If it was certain that the UAV would fly straightly across the target, it would be easy to shrink the acceptance radius. But then, if a random gust of wind should take the UAV out of course with some meters, then the acceptance sphere would never be entered. If, however, the half planes were used, then the waypoint would always be accepted, although the distance between the GPS position where the waypoint is accepted, and the waypoint itself, could be arbitrarily large, depending on the wind.

The GPS position is measured by the autopilot at regular intervals, and the diameter of the acceptance radius can never be larger than the distance between two measured points. However, if this size in meters were called M, and the UAV would fly directly through the target position, then the hits within an acceptance sphere would be evenly distributed around the target position within a maximum distance of $\frac{M}{2}$ from the target position. Using a half plane, all first accepted GPS positions would be after the UAV has passed the target position, evenly distributed with a maximum distance of M from the target.

This means that with an acceptance sphere, it is not absolutely sure that the UAV will register the waypoint, but in a best case scenario, the registered waypoints will be closer to the target position. With a half plane, all waypoints are eventually accepted, but on the other hand, they may be accepted although the UAV is far away from the waypoint.

On the basis of this discussion, it can be concluded that the acceptance sphere can give accepted GPS positions that probably are closer to the target position than if the half planes were used. However, that is in an ideal situation. In reality, disturbances can happen. The acceptance sphere has the insecurity that the UAV might miss it altogether, and the waypoint is not accepted at all. That is a worse outcome than a waypoint that is accepted a bit further away from the target position. The conclusion is therefore that half planes will be used, as they secure that the waypoint is accepted.

12.2.3 Discussion of the Trajectory Generation Algorithms

The algorithms for a trajectory from one directed point to another in a two dimensional plane are described in Section 11.3. They have been implemented by Simen Fuglaas and tested with hardware in the loop simulation, and found to work just as they should. However, they have not been field tested, as there have been no opportunity to do so. Field testing could have given information about how the algorithms react when they meet nature conditions like changing wind and precipitation. But as far as this Master Thesis is concerned, the algorithms have done what they have been asked to.

The Straight Line Approach was only ever intended to be a temporary solution. It does not seem like a sensible solution, as the accuracy of the UAV's position and heading in the point of release is proportional to the distance from the point of release on the *far away* waypoint. The Adapted Dubins Path, on the other hand, is derived from an optimal solution, namely the Dubins Path, according to the inventor of that algorithm (Beard and McLain, 2012). Using that algorithm, the hardware in the loop-tested UAV showed a high degree of accuracy in the point of release. The runtime on that algorithm was not measured, but evidently higher than the one with the Straight Line Approach, but then the accuracy was far better.

The Adapted Dubins Path is just an adjusted Dubins Path, and can therefore not be called a new development. On the other side, the Straight Line Approach has probably also been used before, and is therefore no better alternative for a new development. But as the system demands an optimal solution and that has already been invented, then the needs of the system are more important than the request for something that no-one has ever seen before.

Accuracy of the Trajectory Generation Algorithm

The results from the accuracy tests of the trajectory generation algorithms are showed in Figure 11.3 and Figure 11.4. Both the plots and the calculated mean values show that the Straight Line Approach has a less accurate actual PoR than the Adapted Dubins Path. The mean value of the distance from the calculated PoR to the actual PoR is 13.3401 meters when the Straight Line Approach is used, while it is only 1.2911 meters when the Adapted Dubins Path is used. The mean value of the difference between the calculated approach angle at PoR and the actual approach angle t PoR is 6.7326 degrees when the Straight Line Approach is used, while it is only -0.3612 degrees when the Adapted Dubins Path is used.

It can be observed that the angle difference is strictly positive when the Straight Line Approach is used, while for the Adapted Dubins Path, it is negative. The reason for this may be the direction of the loop that the simulated UAV follows. The Straight Line Approach always flew counter-clockwise, as did the Adapted Dubins Path. The reason for that is that the IMC message with the point of impact was sent from the ground station to the Pandaboard as the simulated UAV had just left the point of release. After passing through the point of release, both algorithms are directing the UAV to start on a counter-clockwise loiter around the projected point of impact on the flight height. If they had followed a clockwise path, then the sign before the angles would very likely have been different.

Both algorithms were running constantly for one night, approximately the same amount of time. However, the Adapted Dubins Path only produced 493 samples, while the Straight Line Approach produced 615 samples. That is because the Adapted Dubins Path includes longer flight time than the Straight Line Approach. That is a drawback, as the Dubins Path is supposed top be time-optimal, see Section 4.3. But although the Straight Line Approach offers a solution that takes shorter time, it does not lead the UAV to the directed point where it should end up. The mean distance from the target at the drop of the sensor is 13.3401 meters, which is not negotiable, and the mean angle deviation from the ideal heading of the UAV is 6.7326 degrees. That means that if time is more essential than precision, then the Straight Line Approach is more effective than the Adapted Dubins Path. However, if the precision of the drop is essential, then the Adapted Dubins Path is the best choice. The mean distance from target at the drop of the sensor, and the mean angle difference of the heading, are 1.2911 meters and -0.3612 degrees, which is not equal to 0, but still fairly precise. The approach has demonstrated that it is a path from one directed point to another.

12.3 Discussion of the Limitation of the Field Tests

As Section 10.3 shows, there were few field tests with the Penguin B on Agdenes. In the work description, step 5 is to test the system with field experiments, but only one field experiment was conducted and the UAV never left the ground during the test. However, that test was successful as all functionality that was tested was working as it should.

There are several reasons why only one field test was conducted. First, the weather conditions on Agdenes are harsh: Several test days were cancelled because of too much wind. The Penguin B is an expensive UAV and the project could risk to damage it if it got caught by the wind and crashed. Besides, it requires a pressured air drived catapult to be launched on Agdenes, as the landing strip has too rugged ground for the UAV to take off on. These two reasons lead to a slow start for the Penguin B, and the first flight on the spring of 2014 was as late as the 10. of May. That flight went well and the UAV was not at all damaged, but the pilots lost radio communication with the UAV at some points. More delays occurred along with more problems, as the autopilot suddenly did not work any more and had to be changed.

On top of all that, Agdenes is situated close to a military airport on Ørlandet, and the week from the 19. of May to the 23. of May, the airspace was only available in small amounts of time. As time limitations demanded the field tests to be done before the 23. of may, the possibilities of testing were scarce. Therefore, on 13. may, the following alternatives were proposed by Tor Arne Johansen:

- Hope that all goes well with the Penguin B and then perform field tests on the available airspace test time.

- Move the field tests to Eggemoen (470 kilometres away from Trondheim) and avoid problems with the catapult and availability of the air space. Eggemoen has a much better landing strip than Agdenes and no military disturbance.

- Move the payload to another UAV, the X8, and fly on Agdenes on the available airspace test time.

- Postpone the tests to June or after the summer.

The options were thoroughly debated: The fourth alternative was not a real option, as the Master Thesis was due the 2. of June for the author and two weeks later for Simen Fuglaas. The second alternative would work well for this project, but would be a great disadvantage for all other project groups who field tested on Agdenes as the equipment would have to be moved. That left the first and the third alternative, and none of the debaters found it realistic to carry through any tests on the Penguin B before the 23. of May. The third alternative would involve a new autopilot called Ardupilot, with which neither the author nor Simen Fuglaas had much experience, as well as the new UAV. New hardware would have to be made, and new simulations.

It is a realistic possibility to change between the Penguin B and the X8, and between the Piccolo autopilot and the Ardupilot autopilot: DUNE has interfaces to both of the autopilot, and the control the UAV without any further problems. The automatic modularity that DUNE involves makes these changes possible. However, it would be time consuming.

One drawback with the Ardupilot on the X8 UAV is that there is no good HIL simulator for that system, which means that most tests are conducted as field tests, and all errors are discovered when the UAV is in the air.

To change to the X8 with an Ardupilot autopilot would be to jeopardise the project with only 10 working days available. But if all went well, important results would be achieved: Field tests would be conducted and the trajectory algorithms could be field tested. However, a failure would mean that all the time invested in the transfer would have been used in vain. Then there would be no field test nor any further HIL tests, as the time that could have been used for HIL test were used on a fruitless UAV transfer.

Although it seemed very wise to change from the expensive and complicated heavy-weight Penguin B to the cheap, simpler X8 on the long view, there would not be enough time to guarantee success. Besides, the available time would mean that even with success on the first attempt, the transfer would demand most of the remaining time of the Master Thesis. The decision was therefore to go with the first alternative, in reality not field test any more and concentrate on HIL tests. The choice led to no more field tests, but to HIL testing on waypoint acceptance criteria and the different trajectory generation algorithms.

12.4 Discussion of the Significance of the Study

This study is motivated by the problem that drifting icebergs constitutes for the shipping industry. It would be very practical to know the position of drifting icebergs at any given time, but still, that is not the real significance of this study. The ability to deploy an object from a UAV with high position is the real advantage this study causes, and that technique is useful for several areas, from GPS tracking of icebergs to deployment of supplies in disaster areas.

Research is done on the topic, as the literature study shows, but no article describes the same as this Master Thesis. Even the UAV described in McGill et al. (2011), which placed sensors on an iceberg, was controlled by a pilot. The study presented in this Master Thesis aims to do the same thing, only by an autonomous UAV, that does not need to be controlled by a pilot. The work described in this Master Thesis has not yet come that far, and a pilot is still required to launch the UAV and fly it to the area where the sensor should be dropped. Still, the project can be continued and the goal would be as relevant.

The study has also contributed to the AMOS research project. It provides an almost finished solution to one of the basic UAV operations that the AMOS project *Autonomous unmanned vehicle systems* wanted to master: Dropping a payload. Since the project uses DUNE as basis for the control system, the system is yet more useful for AMOS: It means that not only is there a system available for precision drops with Penguin B and the Piccolo autopilot, but minor adjustments permits the use of other UAVs like the X8 and other autopilots like the Ardupilot. As long as there is a DUNE interface for the equipment, it can be easily included in the system described in this Master Thesis, or it could replace equipment currently in use in the system.

No parts of this Master Thesis could have been omitted, as they only describe what has been done in the project and the debates that has led to the decisions taken. Especially Chapter 5 is valuable as it gives the thesis consistence and makes it easier to understand the later chapters. The tests described in this Master Thesis are also necessary, as they either verify that the equipment and code work as they should, or test the correctness of two different methods and decide which one will be used later. The results of these tests have been debated earlier in this chapter.

However, other tests could have been useful, for instance a test that find out which significance the fall height has on the insecurity of the point of impact for a released payload. Another useful test would be to see what strong or moderate wind does to the UAV when it tracks the path generated by the trajectory generation algorithms that are developed in this thesis. Simen Fuglaas has occupied himself with these tests and other aspects of this project. For supplementary reading on the topic of this project, his Master Thesis should be read.

Chapter 13

Future Work

Although two master students have worked on this project for one semester, a lot more could have been done: improvements found while testing the system and improvements found while not testing the system, planned work that there were no time for and the planned continuation of this Master Thesis. This project was carried out in the beginning of the AMOS project and some tests or implementations are easier to carry out when it has been done before, for instance the field tests. This Chapter contains suggestions to future work on this project.

Change of UAV and Autopilot

The most important part of the future work would be to change the UAV used to deploy the sensor from a Penguin B to the X8. This has been discussed in Section 12.1, and the conclusion was that the X8 would be a better UAV for testing, as it is not as expensive as the Penguin B. That would make it possible to carry out more field tests, which would certainly be useful. A change of UAVs would involve a change of autopilots. It is not possible to foresee all problems this change will cause, both caused of the actual transfer and caused by the possible different behaviour of that other autopilot. However the extra workload would not be too great, as DUNE has an interface to the other autopilot that resembles the interface to Piccolo a lot.

Conduct more Field Tests

One objective of this Master Thesis was to test the system with field experiments. It was only possible to conduct one field test on Agdenes because of reasons described in Section 12.3. That one test was successful and showed that the system worked as far as the test was able to show, but the system has never been tested on a flying UAV.

The first field test that should be conducted is to test the release on target from the air: The UAV pilot controls the UAV and guides it to the pre-calculated point of release, then the drop mechanism is signalled to drop its payload by the Pandaboard.

The second field test that should be conducted is to test the Straight Line Approach. This is not strictly necessary, as the HIL tests show that the precision of the position and heading of the UAV in what it believes is the point of release, does not largely agree with the actual position and heading for the point of release. However, it is good to have a demonstration of the trajectory from a field experiment. Besides, it is important to see if the Adapted Dubins Path is more precise than the Straight Line Approach in a field experiment, too.

The third field test that should be conducted is to test the Adapted Dubins Path. As this is the algorithm that is considered to produce the best trajectory for the UAV, it is of great importance to investigate its properties on a real UAV. A simulator can give good results where a field experiment discovers problems. The field tests will also give good feedback on the accuracy of the algorithm, and some adjustments of the parameters should be expected.

Test the Sensor Protection

When the trajectory generation algorithms have been field tested, the next step would be to test the system with the deployment of a real sensor. Firstly, it is important to find the correct model for this sensor to be able to calculate its fall path correctly. Secondly, it is necessary to see if it will survive the impact. Before this is experimented with, a dummy sensor should be used with the same protection that will be used for the real sensor. When that is perfected, a real sensor could be dropped.

Change Navigation Method from Waypoints to Heading

To get a controllable trajectory and simplify the tracking of the trajectory, heading should be used. This requires a good direction angle sensor on the UAV, either using the autopilot or with an external sensor. The change of navigation method will probably result in an optimal point of release for the UAV.

Implement a Graphical User Interface

The tool from LSTS called Neptus is described in Section 3.1 but has not been used in this project. It would be useful to have a graphical user interface for DUNE, and Neptus gives a ready-to-use solution.

Chapter 14

Conclusion

This chapter describes in short what has been performed and achieved in this project, and what can be concluded.

14.1 What has been Performed

The work started with a thorough literature study, which has given relevant information to this Master Thesis. Based on researched literature and theory, decisions have been taken throughout the project. The hardware for a payload mount, meant to be carried by a Penguin B UAV, has been built, tested and found to work. The tests have been conducted both on a laboratory and in the field, using a UAV that taxed on a landing strip. Two algorithms for guiding a UAV from one directed point to another in a two dimensional plane have been developed, tested and their accuracy have been compared.

Software has been developed, both for the Pandaboard and for a user interface, on a ground station. They were both implemented with the use of the DUNE framework, which facilitates the communication with the Piccolo autopilot that controls the Penguin B UAV. The whole system has been tested on a hardware in the loop simulator and on field tests, to find out if it behaved as planned. The two trajectory generation algorithms have also been tested to find out which algorithm lets the UAV release its sensor from a position closest to the calculated point of release. Last, but not least, two waypoint acceptance criteria have been tested to find out which gives the closest results. All of the results have been discussed and debated, to find out their relevance and value.

14.2 What has been Achieved

In this Master Thesis, a ready-to-use modular precision deployment system meant to be mounted on a Penguin B, has been made. The necessary calculations and decisions have been made, the hardware has been built and the system has been tested. The system is modularly built and easily modifiable, since it utilizes the DUNE software solution. That makes the system easy to use not only with the Penguin B and the Piccolo autopilot, but with a large range of UAVs and autopilots, for instance the X8 UAV and the Ardupilot autopilot mentioned in the discussion.

Tests have been conducted to find out which trajectory generation algorithm that produces the best path to the point of release for the UAV. Tests have also been conducted to find out which waypoint acceptance criteria is the more effective.

The work done in this Master Thesis has contributed a whole lot on the subject of high precision sensor deployment from unmanned aerial vehicles, although further testing should be conducted.

14.3 Conclusions

Conclusions drawn from this Master Thesis are as follows:

- The Adapted Dubins Path is found to be the trajectory generation algorithm that produces the best trajectory, measured by deviation between the calculated point of release and the point of release accepted in the algorithm.
- It is more accurate to use half planes as waypoint acceptance criteria than an acceptance sphere, but it will be more precise if they are combined.
- The system described in this Master Thesis can be used for high precision deployment of wireless sensors from a UAV, but should be further tested.
- The system described in this Master Thesis could easily be implemented on another UAV using another autopilot, as long as there is a DUNE interface for that equipment.

14.4 Recommendation for Future Work

By testing and discussion, it is demonstrated that this system works well, but that it can be further developed. The most important steps of the future work on this project are:

- To change the UAV and autopilot to an X8 and the Ardupilot
- To conduct more field tests
- To find a well suited protection for the deployed sensor
- To change the navigation method from waypoints to heading changes
- To implement a graphical user interface for the system

Bibliography

Arduino (2014). Arduino Introduction. http://www.arduino.cc/en/Guide/Introduction#.UxiRqoV213s.

Beard, R. W. and T. W. McLain (2012). *Small Unmanned Aircraft.* Princeton University Press.

Bousson, K. and P. Machado (2013). 4d trajectory generation and tracking for waypoint-based aerial navigation. *WSEAS Transactions on Systems and Control 8*(3), 105–119. cited By (since 1996)0.

Cloud Cap Technology (2014a). Ground stations. https://www.cloudcaptech.com/piccolo_groundstation.shtm.

Cloud Cap Technology (2014b). Piccolo autopilots - the standard in uas systems. http://www.cloudcaptech.com/piccolo_system.shtm.

Cloud Cap Technology (2014c). Piccolo command center - powerful flight management support. https://www.cloudcaptech.com/piccolo_command_center.shtm.

Corke, P., S. Hrabar, R. Peterson, D. c. Rus, S. Saripalli, and G. Sukhatme (2004). Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. Volume 2004, New Orleans, LA, pp. 3602–3608.

Dong, M. b., B. b. Chen, G. b. Cai, and K. c. Peng (2007). Development of a real-time onboard and ground station software system for a uav helicopter. *Journal of Aerospace Computing, Information and Communication 4*(8), 933–955. cited By (since 1996)27.

Dubins, L. E. (1957, Jul). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics 79*(3), 497–516.

Ducote, R. J. and R. J. Speelman (1966, July-August). U. s. air force concepts for accurate delivery of equipment and supplies. In *Aerodynamic Deceleration Systems Conference*, Volume 4.

E-flite advancing electric flight (2014). Servoless payload release - instruction manual. http://www.horizonhobby.com/pdf/EFLA405-Manual.pdf.

HistoryOrb (2014). Historical events for year 1849. http://www.historyorb.com/events/date/1849.

Laboratório de Sistemas e Tecnologias Subaquáticas (2013). Lsts. http://lsts.fe.up.pt/about.

Lai, C.-K., M. Lone, P. Thomas, J. Whidborne, and A. Cooke (2011). On-board trajectory generation for collision avoidance in unmanned aerial vehicles. Big Sky, MT.

Maza, I., K. Kondak, M. Bernard, and A. c. Ollero (2010). Multi-uav cooperation and control for load transportation and deployment. *Journal of Intelligent and Robotic Systems: Theory and Applications 57*(1-4), 417–449. cited By (since 1996)30.

McGill, P., K. Reisenbichler, S. Etchemendy, T. Dawe, and B. Hobson (2011). Aerial surveys and tagging of free-drifting icebergs using an unmanned aerial vehicle (uav). *Deep Sea Research Part II: Topical Studies in Oceanography 58*(11â€“12), 1318 – 1326. Free-Drifting Icebergs in the Southern Ocean.

Oliveira, T., G. Cruz, E. R. B. margues, and P. Encarnacão (2011). A test bed for rapid flight testest of uav control algorithms.

OMAPpedia (2012). Pandaboard faq. http://www.omappedia.com/wiki/PandaBoard_FAQ.

Omari, S., M.-D. Hua, G. Ducard, and T. Hamel (2013). Hardware and software architecture for nonlinear control of multirotor helicopters. *IEEE/ASME Transactions on Mechatronics*. cited By (since 1996)0; Article in Press.

Pandaboard.org (2014). Board references. http://pandaboard.org/content/resources/references.

Pinto, J., P. Calado, J. Braga, P. Dias, R. Martins, E. Marques, and J. Sousa (2012). Implementation of a control architecture for networked vehicle systems. Volume 3, Porto, pp. 100–105.

Pinto, J., J. Sousa, F. Py, and K. Rajan (2012). Experiments with deliberative planning on autonomous underwater vehicles. In *IROS Workshop on Robotics for Environmental Modeling*, Algarve, Portugal.

Santamaria, E., P. Royo, J. Lopez, C. Barrado, E. Pastor, and X. Prats (2007). Increasing uav capabilities through autopilot and flight plan abstraction. Dallas, TX, pp. 5B51–5B510.

SiriusXM (2014). Iceberg canadian adult alternative music. https://www.siriusxm.ca/Channels/Iceberg.aspx.

Tang, Y.-R. and Y. Li (2011). The software architecture of a reconfigurable real-time onboard control system for a small uav helicopter. Incheon, pp. 228–233.

Texas Instruments (2013, March). Lm1084 low dropout positive regulators. http://www.ti.com/lit/ds/symlink/lm1084.pdf.

Tinder Rocketry (2014). Peregrine exhaustless co2 ejection system. http://fruitychutes.com/files/peregrine/Tinder%20Rocketry%20Reloading%20Guide-Final-v2.pdf.

Tuna, G., T. Mumcu, K. Gulez, V. Gungor, and H. Erturk (2012). Unmanned aerial vehicle-aided wireless sensor network deployment system for post-disaster monitoring. *Communications in Computer and Information Science 304 CCIS*, 298–305.

UAV Factory (2014). Homepage. www.uavfactory.com.

Ubiquity Networks (2014). airmax. http://www.ubnt.com/airmax.

Wikipedia (2014a, April). Climate of the arctic. http://en.wikipedia.org/wiki/Climate_of_the_Arctic.

Wikipedia (2014b, May). Geodetic datum. http://en.wikipedia.org/wiki/Geodetic_system.

Wikipedia (2014c). Git (software). http://en.wikipedia.org/wiki/Git_%28software%29.

Williams, P. b. c. and P. b. Trivailo (2006). Cable-supported sliding payload deployment from a circling fixed-wing aircraft. *Journal of Aircraft 43*(5), 1567–1570. cited By (since 1996)2.

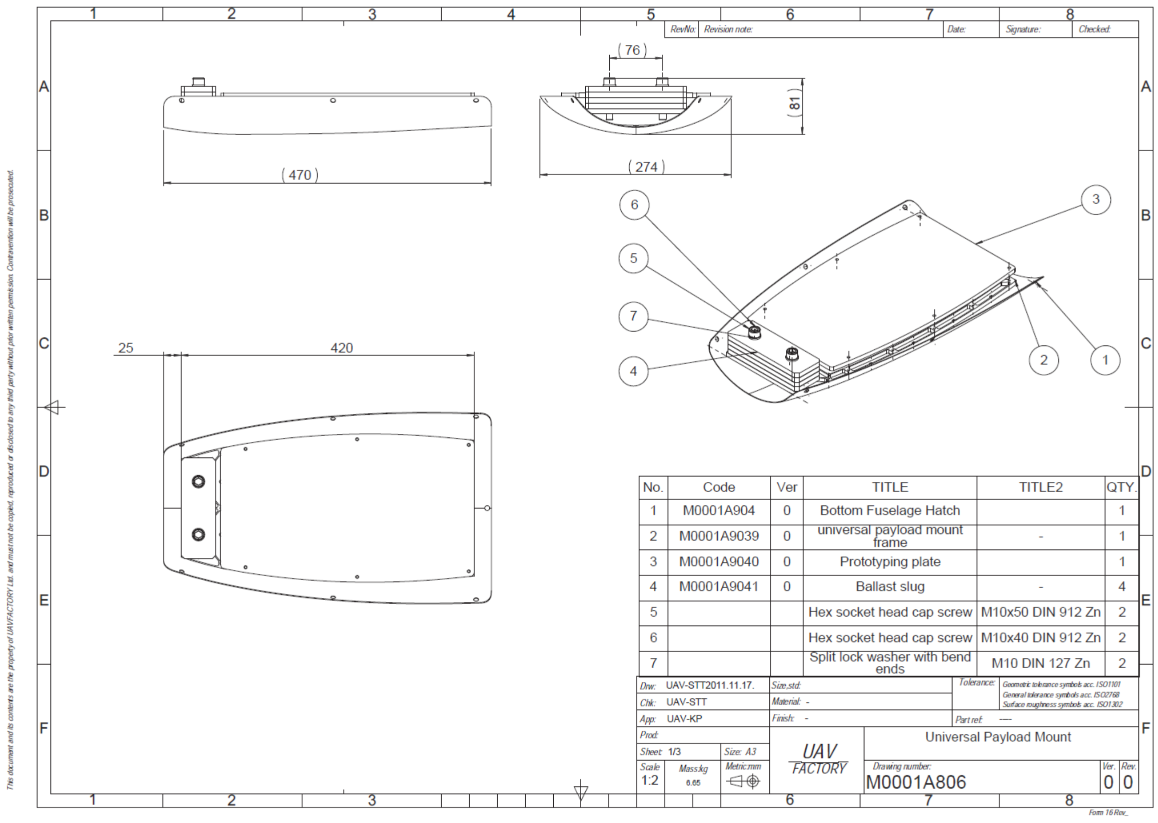
Wuest, M. R. and R. J. Benney (2005, December). Precsion airdrop (largage de precision). *Flight Test Techniques Series 24*.

Part V

Appendices

Appendix A

Drawings and Schematics



No.	Code	Ver	TITLE	TITLE2	QTY.
1	M0001A904	0	Bottom Fuselage Hatch		1
2	M0001A9039	0	universal payload mount frame	-	1
3	M0001A9040	0	Prototyping plate		1
4	M0001A9041	0	Ballast slug	-	4
5			Hex socket head cap screw	M10x50 DIN 912 Zn	2
6			Hex socket head cap screw	M10x40 DIN 912 Zn	2
7			Split lock washer with bend ends	M10 DIN 127 Zn	2

Figure A.1: The drawing of the payload mount for a Penguin B (source: UAV Factory (2014)).

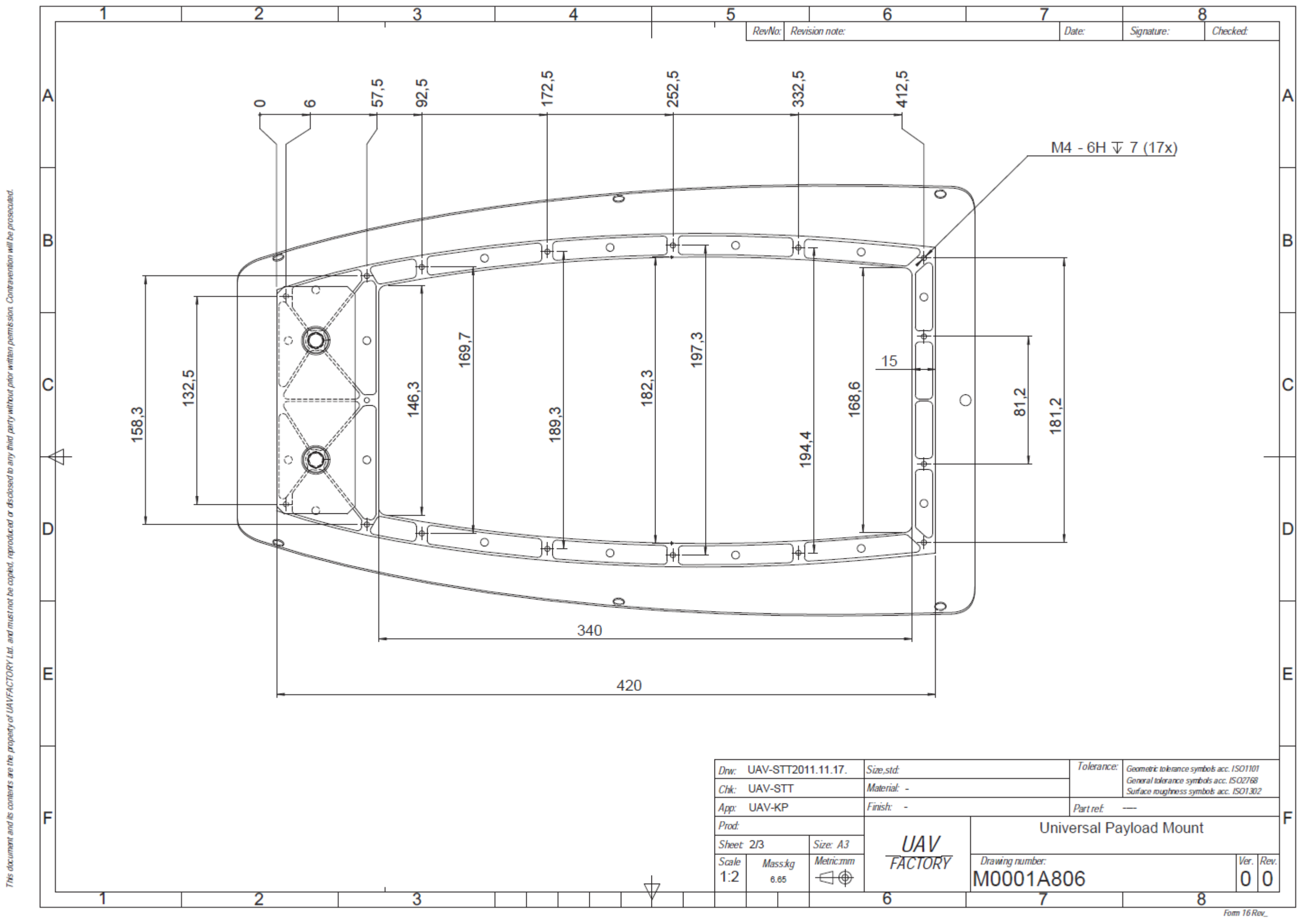

0
6
57,5
92,5
172,5
252,5
332,5
412,5
M4 - 6H ↧ 7 (17x)
158,3
132,5
146,3
169,7
189,3
182,3
197,3
194,4
168,6
15
81,2
181,2
340
420
RevNo: Revision note: Date: Signature: Checked:
Drw: UAV-STT2011.11.17.
Chk: UAV-STT
App: UAV-KP
Prod:
Sheet 2/3
Size: A3
Scale 1:2
Mass:kg 6.65
Metric:mm
Size,std:
Material: -
Finish: -
Tolerance:
Geometric tolerance symbols acc. ISO1101
General tolerance symbols acc. ISO2768
Surface roughness symbols acc. ISO1302
Part ref: —
UAV FACTORY
Universal Payload Mount
Drawing number: M0001A806
Ver. 0 Rev. 0
Form 16 Rev_
This document and its contents are the property of UAVFACTORY Ltd. and must not be copied, reproduced or disclosed to any third party without prior written permission. Contravention will be prosecuted.

Figure A.2: The dimensions of the payload mount for a Penguin B (source: UAV Factory (2014)).

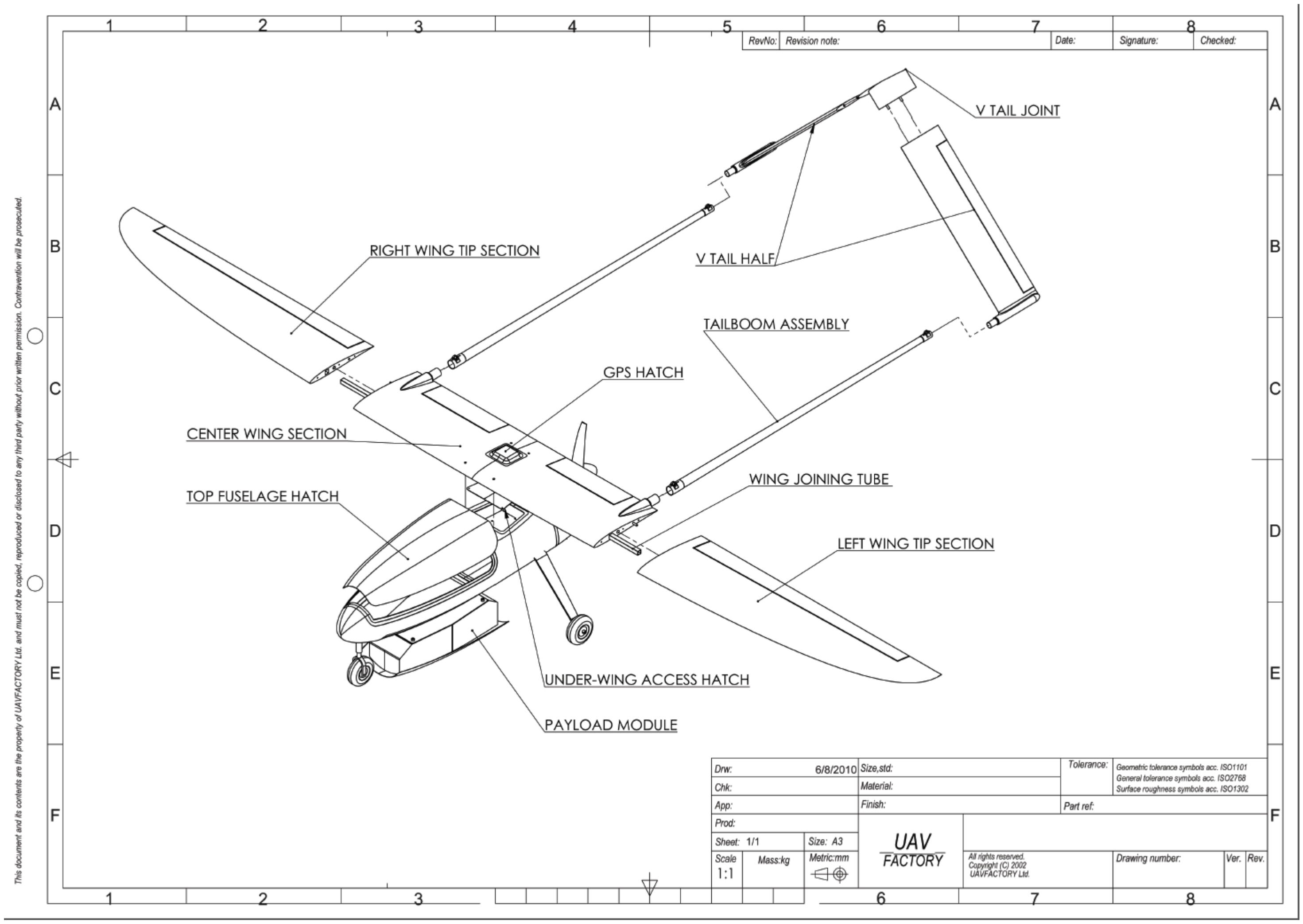

V TAIL JOINT
V TAIL HALF
RIGHT WING TIP SECTION
TAILBOOM ASSEMBLY
GPS HATCH
CENTER WING SECTION
WING JOINING TUBE
TOP FUSELAGE HATCH
LEFT WING TIP SECTION
UNDER-WING ACCESS HATCH
PAYLOAD MODULE
RevNo: Revision note: Date: Signature: Checked:
Drw: 6/8/2010
Size,std:
Material:
Finish:
Tolerance:
Geometric tolerance symbols acc. ISO1101
General tolerance symbols acc. ISO2768
Surface roughness symbols acc. ISO1302
Part ref:
Chk:
App:
Prod:
Sheet: 1/1
Size: A3
Scale 1:1
Mass:kg
Metric:mm
UAV FACTORY
All rights reserved. Copyright (C) 2002 UAVFACTORY Ltd.
Drawing number:
Ver.
Rev.
This document and its contents are the property of UAVFACTORY Ltd. and must not be copied, reproduced or disclosed to any third party without prior written permission. Contravention will be prosecuted.

Figure A.3: The drawing of a Penguin B (source: UAV Factory (2014)).

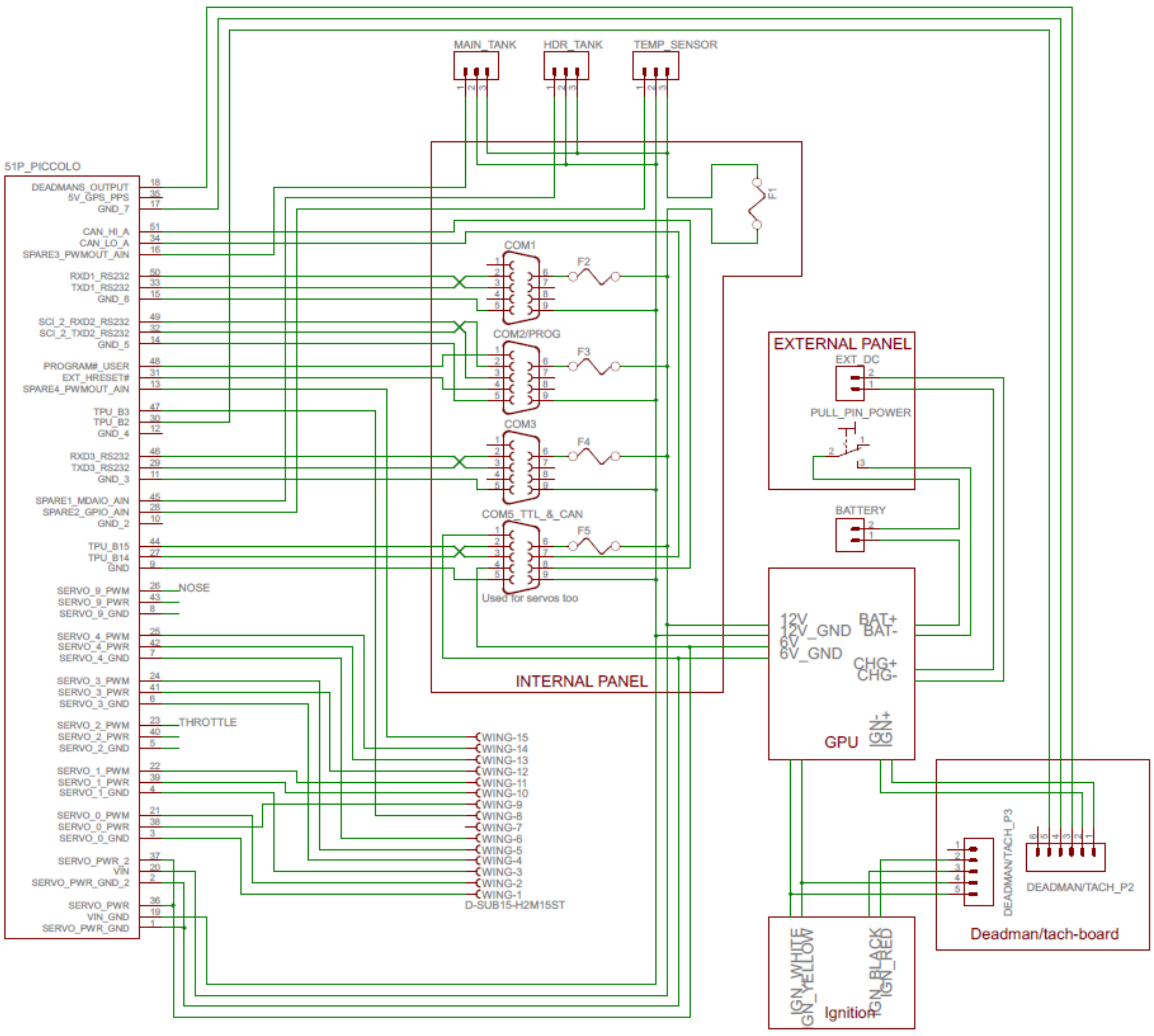


Figure A.4: The schematics of the Penguin B. The COM-ports are interfaces to the Piccolo and the power (Source: Jon Petter Skagmo).

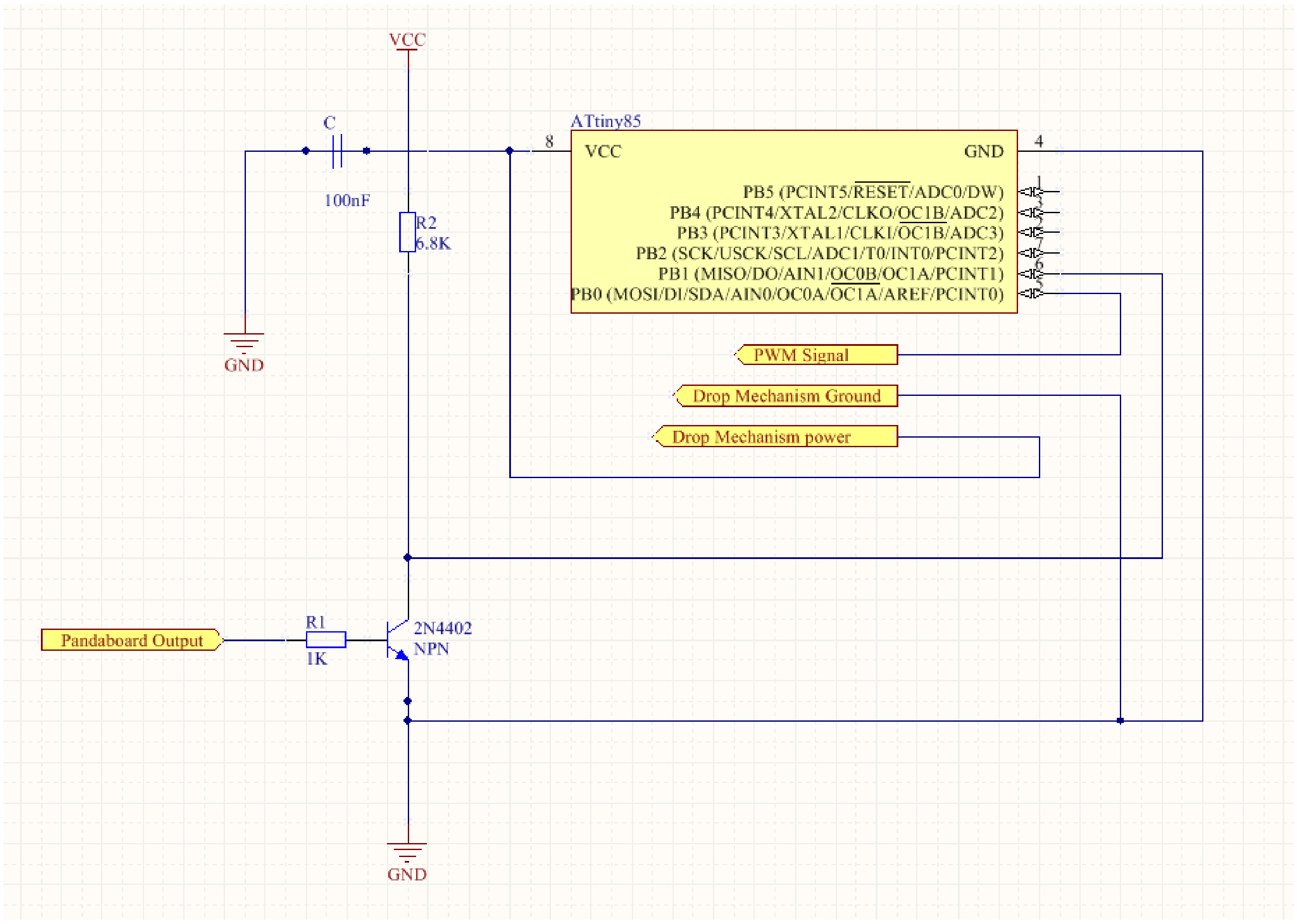

VCC
C
100nF
ATtiny85
8
VCC
GND
4
PB5 (PCINT5/RESET/ADC0/DW)
PB4 (PCINT4/XTAL2/CLKO/OC1B/ADC2)
PB3 (PCINT3/XTAL1/CLKI/OC1B/ADC3)
PB2 (SCK/USCK/SCL/ADC1/T0/INT0/PCINT2)
PB1 (MISO/DO/AIN1/OC0B/OC1A/PCINT1)
PB0 (MOSI/DI/SDA/AIN0/OC0A/OC1A/AREF/PCINT0)
R2
6.8K
GND
PWM Signal
Drop Mechanism Ground
Drop Mechanism power
Pandaboard Output
R1
1K
2N4402
NPN
GND

Figure A.5: Schematics of the PWM generator system

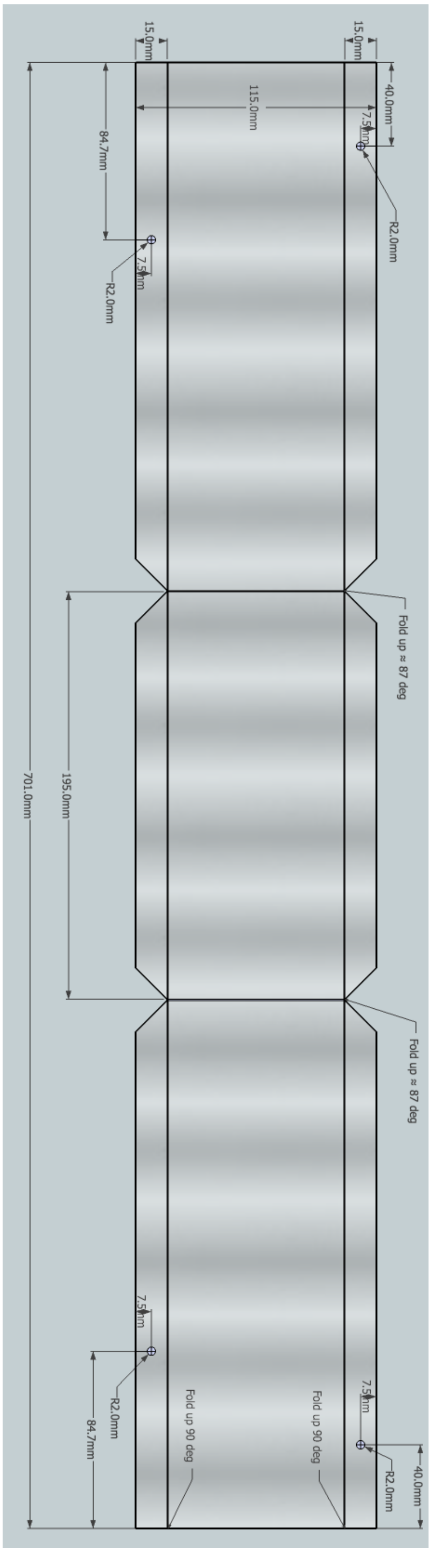


Figure A.6: Drawings of the payload box (source: uavlab.itk.ntnu.no)

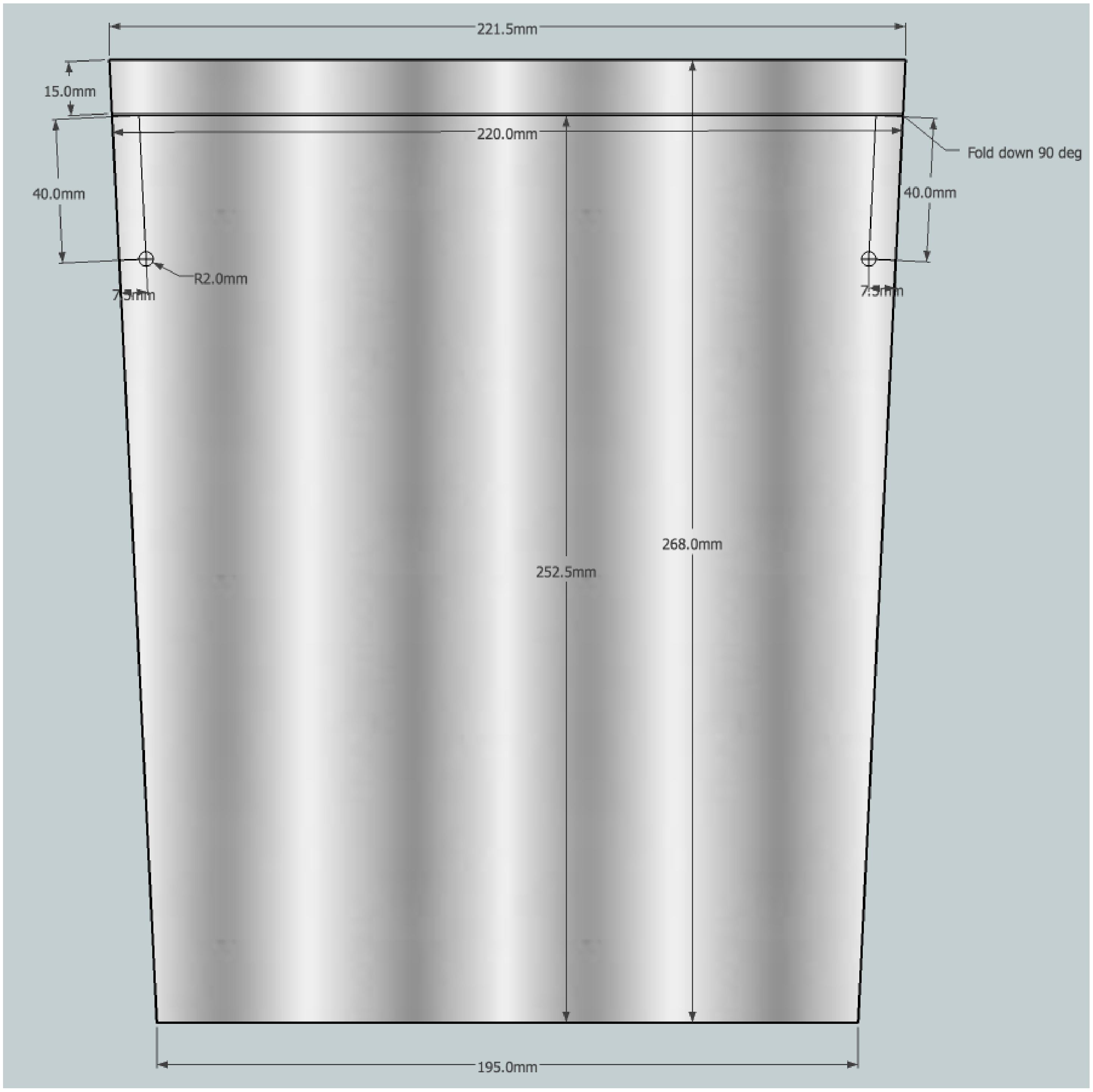


Figure A.7: Drawings of lid for the payload box (source: uavlab.itk.ntnu.no)

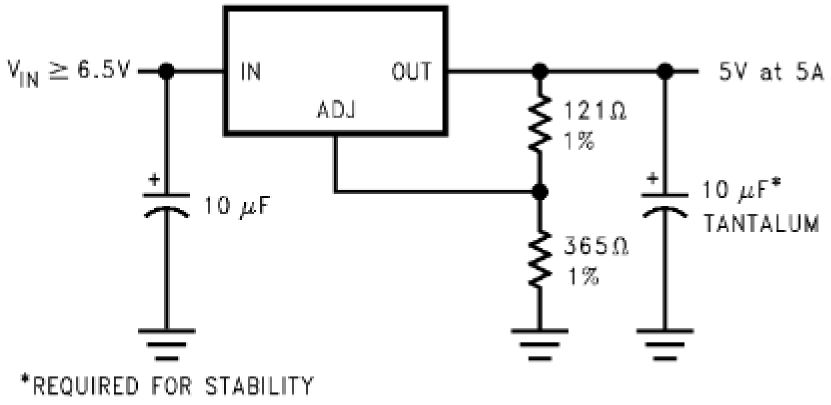


Figure 24. Adjustable @ 5V

Figure A.8: The schematics used to connect the voltage regulator circuit (source Texas Instruments (2013))

Appendix B

Definitions, Code Implementations, Equations and their Derivations

The Position, Velocity and Acceleration of a Particle following a Circular Path

$$\mathbf{r}(t) = cost\mathbf{i} + sint\mathbf{j} \tag{B.1}$$

$$\mathbf{v} = \frac{d\mathbf{r}}{dt} = -sint\mathbf{i} + cost\mathbf{j} \tag{B.2}$$

$$\mathbf{a}(t) = \frac{d\mathbf{v}}{dt} = -cost\mathbf{i} - sint\mathbf{j} \tag{B.3}$$

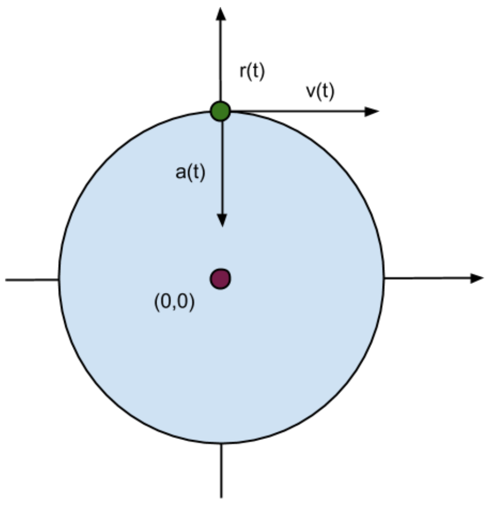


Figure B.1: The position vector $\vec{r}(\frac{3\pi}{2}) = (0,1)$, the velocity vector $\vec{v}(\frac{2\pi}{2}) = (1,0)$, the acceleration vector $\vec{a}(\frac{3\pi}{2}) = (0,-1)$.

The Definition of a Unit Tangent Vector

$$\mathbf{T} = \frac{d\mathbf{r}}{ds} = \frac{d\mathbf{r}/dt}{ds/dt} = \frac{\mathbf{v}}{|\mathbf{v}|} \tag{B.4}$$

Code for the PWM Generator

```
/*
 * PWMforAtTiny.c
 *
 * Created: 24.02.2014 10:46:25
 *  Author: siriholt
 */


#include <avr/io.h>

#define PWMPIN PB0
#define INPUTPIN PB1

void PWM_init(){
        //Enable input on the inputpin
        DDRB = 0x00;
        //Set the inputpin low
        PORTB |= (0 << INPUTPIN);
        //Enable output on the outputpin
        DDRB |= (1 << PWMPIN);

        //Clear OC0A on Compare Match, using fast PWM mode
        TCCR0A |= (1 << COM0A1) | (1 << WGM01) | (1 << WGM00);
        //Fast PWM mode with 0xFF = TOP, Clock selecy 64 prescaler
        TCCR0B |=  (0 << WGM02) | (1 << CS01) | (1 << CS00);

        //Init the PWM signal to be compared with the counter value
        OCR0A = 0;
}

void PWM_run(){
        uint64_t value;
        uint64_t sleeper;

        for(value = 0; value < 50; value++){
                OCR0A = value;
                sleeper = 0;
                while(sleeper < 300){
                        sleeper++;
                }
        }
        for(value = 50; value > 0; value--){
                OCR0A = value;
                sleeper = 0;
                while(sleeper < 300){
                        sleeper++;
                }
        }
}


int main(void)
{
        PWM_init();

    while(1)
    {
                if(PINB & (1 << INPUTPIN)){
                        PWM_run();
                }else{
                        OCR0A = 0;
                }
    }
}
```

Appendix C

Annotated Bibliography

Autonomous Deployment and Repair of a Sensor Network using an Unmanned Aerial Vehicle

In this article, Corke et al. (2004) present a method to plan and deploy a sensor node network using unmanned helicopters. The article does not go into detail about how the deployment is to take place, only that once the order to deploy a sensor reaches the controller, the helicopter is told to drop the sensor in that position. The deployment mechanism is also described. It consists of a spiral of wired coil, on which the sensors are attached. To deploy one sensor, the spiral is turned enough times. The sensor reaches the end of the coil and falls down.

The article is useful because of its alternative to a deployment mechanism. To rotate a wire coil seems like a sensible choice to drop several sensors from the same UAV. However, if it were only one sensor, this would be unnecessary complex. Corke et al. (2004) conclude that the use of UAVs will increase the reach of sensor networks, which concurs with the intention of this master thesis.

Multi-UAV Cooperation and Control for Load Transportation and Deployment

This article, written by Maza et al. (2010), addresses the cooperation of multiple UAVS when it comes to transportation of heavy loads and deployment of sensor networks. Although section 4 is concerned with an experiment where an UAV deploys a sensor node, monitored by two other UAVs, little useful information was given.

Unmanned Aerial Vehicle-Aided Wireless Sensor Network Deployment System for Post-disaster Monitoring

This article by Tuna et al. (2012) describes how UAVs can deploy sensor nodes for use in post-disaster environments. It assumes an optimal location for the sensor drop and describes how the navigation and trajectory tracking is done. The article suggests a combination of Global Positioning System (GPS) and Inertial Navigation system (INS) to combine the advantages of the navigation systems, and concludes that it is possible for the UAV to follow a trajectory to an optimal drop point.

This article is relevant as it describes how the navigation of an UAV can be done if it is important that the trajectory is followed as accurately as possible.

Precision Airdrop (Largage de précision)

This article by Wuest and Benney (2005) explains theory about and different practices to precise payload deployment from air crafts. It contains equations necessary for the computations, different drop methods, sensor techniques and deployment systems in use or under development.

The different computations presented in the article are useful for this master thesis and their considerations are sensible. That the methods presented in the article are currently in use, makes the theory more relevant.

U. S. Air Force Concepts for Accurate Delivery of Equipment and Supplies

This is an article from 1967 written by Ducote and Speelman (1966) that explains ballistic theory behind parachute drops from air planes, and what methods to use in different altitudes. Although the article is not concerned with UAVs, the theory is relevant and elaborates the characteristics of air drops with parachutes.

Cable-Supported Sliding Payload Deployment from a Circling Fixed-Wing Aircraft

This article by Williams and Trivailo (2006) explains a technique used for sliding payloads to the ground from a fixed-wing air craft using a cable. The proposed solution is to circle the air craft with a fixed diameter around the drop spot, and anchor the cable tip to the ground to increase the stability. The authors conclude with a recommendation to install a breaking mechanism to prevent the payload from sliding to quickly to the ground.

This article is very useful for this project, as it proposes a realistic and tested technique for precision payload drop. The method seems accurate, but might be too complicated as it requires a breaking mechanism as well as the circling move and the cable with an anchor. On the other hand, an air drop does not have any break either, so that may not be necessary.

Hardware and Software Architecture for Nonlinear Control of Multirotor Helicopters

In this article, Omari et al. (2013) model a system architecture for multirotor helicopters. The article includes the necessary mathematics for the model, and explains how the practical implementation can be done. This model applies to all UAVs that land and take off vertically, and will thus not apply to our project. However, the overview of the system architecture can be useful, as it will apply to any UAV with ground control.

The Software Architecture of a Reconfigurable Real-time Onboard Control System for a Small UAV Helicopter

Tang and Li (2011) discuss the software architecture and implementation in a UAV Helicopter. It includes figures of the architecture, which can be useful inspiration.

Development of a Real-Time Onboard and Ground Station Software System for a UAV Helicopter

Dong et al. (2007) describe the interaction between the onboard system of the UAV and the ground control station, and presents the software system. It explains how the synchronization of different onboard tasks can be done and how the different layers of the ground station are organized. This is a useful article and the theory can be applied in our project.

On-Board Trajectory Generation for Collision Avoidance in Unmanned Aerial Vehicles

This paper by Lai et al. (2011) is concerned with the field of collision avoidance in UAVs. It treats trajectory generation in a block diagram and mathematically. This seems like a useful article if one is working on collision avoidance.

4D Trajectory Generation and Tracking for Waypoint-Based Aerial Navigation

This article is written by Bousson and Machado (2013) and is concerned with trajectory planning where the arrival time to the waypoint is a constraint. That is not to be treated in our project, thus the article will be too complicated.

Increasing UAV Capabilities through Autopilot and Flight Plan Abstraction

In this article, Santamaria et al. (2007) present two subsystems, which are intended to create an abstraction layer for the autopilot and a flight plan manager. There will be one program onboard the UAV that makes the connection between payloads and the autopilot go smoother, and that communicates with the flight plan manager. The authors describe the systems, but do not seem to have tested it.

These two subsystems are approximately equivalent to the software tool chain that will be used in this project, and the article does therefore seem very relevant. However, as it is not tested, it only describes an alternative to the LSTS tool chain, and will probably not bring the project any further. Then again, someone might have used this article to work further on it, and that might contribute to our project.

Implementation of a Control Architecture for Networked Vehicle Systems

This article presents the LSTS tool chain, consisting of the modules: DUNE, Neptus and the IMC protocol. Pinto et al. (2012) explain the relationship between the entities, their different layers and their main properties. After describing how the tools work, some tests are mentioned where they were used and in the conclusion, a recommendation is given to use these tools in unmanned vehicles.

As these tools will be used in our project, this article is useful as it gives insight in and an introduction to them.

Experiments with Deliberative Planning on Autonomous Underwater Vehicles

In this article, Pinto et al. (2012) describes how the LSTS tool chain is used in combination with T-REX, an extended control architecture, to improve the onboard autonomy of an LAUV. This article says little more than the one by Pinto et al. (2012), but extends it and can extend the knowledge on the LSTS tool chain.

Aerial surveys and tagging of free-drifting icebergs using an unmanned aerial vehicle (UAV)

The research team presenting this article ((McGill et al., 2011)) has used UAVs to drop GPS sensors on drifting icebergs, to be able to keep them under surveillance. The UAV used is not the same as in our project and instead of having an autonomous UAV, they used a radio-controlled one, butapart from that, this project resembles our project a lot. In the article, they suggest the use of a toy ball to encapsulate the GPS sensor. The toy balls have four sticks pointing out to prevent it from rolling off the ice berg and fall freely from the UAV without any parachute. The drop mechanism used in this project is not that useful, as the UAV used does not have any propeller behind it, which it must be careful not to hit with the sensor.

A test bed for rapid flight testing of UAV control algorithms

This article is written by Oliveira et al. (2011) at the *Academia de Força Aérea Portuguesa, Faculdade de Engenharia da Universidade do Porto* and *Faculdade de Engenharia da Universidade Católica Portuguesa* in Portugal. It describes work with the Piccolo Autopilot on UAVs and how it cooperates with the LSTS toolchain. This is very important information for this Master Thesis and will make the familiarization with the LSTS tool chain in combined with the Cloud Cap Technology (see Section 3.4).