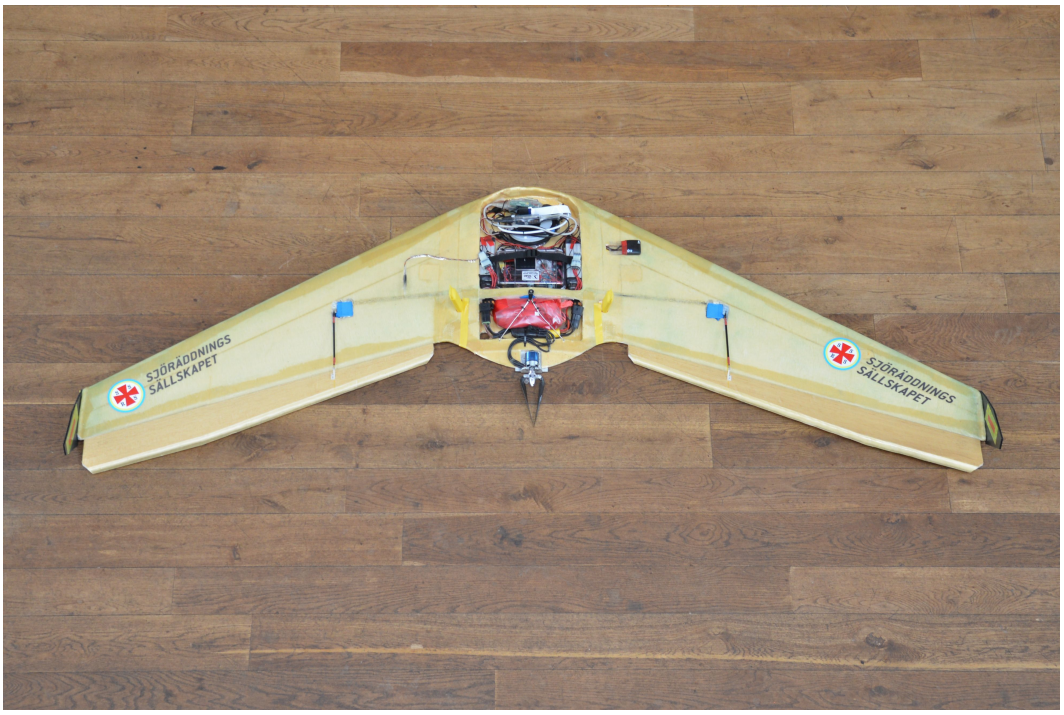
CHALMERS




# Providence – UAV system to support search and rescue

*Master's Thesis*

Anton Lidbom

Efstratios Kiniklis

Department of Signals & Systems
Chalmers University of Technology
Gothenburg, Sweden 2015
Report no. EX050/2015

# Providence – UAV support for search and rescue

Anton Lidbom
Efstratios Kiniklis

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2015

**Providence – UAV support for search and rescue**
**Anton Lidbom**
**Efstratios Kiniklis**



Department of Signals and Systems
Chalmers University of Technology
SE-41296 Gothenburg
Sweden
Telephone +46(0)31-772 1000

Providence − UAV support for search and rescue
Anton Lidbom
Efstratios Kiniklis
Department of Signals and Systems
Chalmers University of Technology

**Abstract**

This thesis is focused on designing and developing a small Unmanned Aerial Vehicle (UAV) to aid in search and rescue missions for the Swedish Sea Rescue Society (SSRS). The aircraft is intended to be launched within minutes of receiving a distress call and autonomously travel to a specified location. During travel, a live video stream is to be captured and sent from the aircraft via the mobile network to a ground station. The obtained information is to be used for aiding in critical decisions for the rescue mission such as; what boat to use, the size of the rescue crew, what tools to bring etc.

The thesis presents the components involved in constructing the UAV, such as the autopilot, the video system, the communication link and the airframe. The hardware and software choices are combined into a system and implemented onto a fixed wing airplane. The UAV uses a flying wing model design that is constructed from Expanded Poly-Olefin Resin (EPOR) foam. The Flight Management Unit (FMU) uses a PI-controller to autonomously maneuver the UAV which is tuned in a simulation environment. The autonomous mission is tested in simulation and also in real flight. The plane operates successfully in Hardware-In-the-Loop (HIL) simulation, but during real test flights the UAV displays poor flight behavior which is analyzed and discussed. Likely reasons are related to the weight and balance of the plane and also inaccuracies in the simulation model for controller tuning.

## Acknowledgements

The work conducted in this thesis has been developed in cooperation with the Swedish Sea Rescue Society (SSRS). We would like to thank SSRS for their hospitality, providing us with a workplace and supplying us with any equipment needed for the project.

We would also like to give a special thanks to our supervisor at SSRS, **Fredrik Falkman**, who have been very helpful and dedicated towards this project. Providing us with ideas, expertise and design.

Finally we want to thank our examiner at Chalmers university, **Petter Falkman**, for his feedback regarding content and structure of the planning report and thesis.

Anton Lidbom & Efstratios Kiniklis
Gothenburg, Sweden

# Contents

# List of Figures

# 1

# Introduction

Unmanned Aerial Vehicles (UAVs) are and have been prominent within both military and government operations during the most recent years. With the rapidly advancing technology, these drones are becoming increasingly effective and significantly less costly. The development has piqued the interest of many organizations with intentions to explore and adapt UAVs as one of their modern solutions. Mapping, surveying and search/rescue missions are some of the applications where the attributes of an UAV are appealing. These are functions not only valuable in modern warfare, but also for nonmilitary purposes. There is a lot to gain by replacing the pilot with an automated system; it is cheaper, always available and can be customized. The most important aspect is that it makes the airspace an accessible medium where airborne vehicles can be used to aid in numerous operations.

Using UAVs for search and rescue has advantages in both response time and need for manpower compared to piloted aircrafts. These are qualifications that attract organizations such as the Swedish Sea Rescue Society (SSRS), hence, they have taken an interest in the idea. The SSRS is responsible for 70 % of all sea rescues in Sweden and receives no government funding [1]. The Society is financed by membership fees, donations and voluntary work. Despite this, or possibly as a result of this, the Society has doubled the number of sea rescue stations in recent years. Furthermore, they have tripled the number of rescue volunteers available and built 70 modern rescue vessels. This expansion has enabled SSRS to meet their goal of departing within 15 minutes or less from the time an alarm is received.

Obtaining information about an accident before departure would ease decisions such as; what boat to choose, equipment to bring and how many crew members that are needed. By the use of UAVs, an aircraft could be deployed as soon as a distress signal is received. The UAV would then autonomously travel to the specified coordinates and be able to

send a live video feed of the situation. This information would aid in crucial decisions and also allow the rescue mission to be supervised. By having UAVs on standby at several locations around the Swedish seaboard, SSRS would have the ability to get visual aid at every coordinate close to the coast. By the use of such a system, a live video feed of the accident could be provided within minutes of an emergency call. Moreover, if the location of the accident is not completely established, the UAV could autonomously scout the area and locate the person in distress.

## 1.1 Problem description and goal

This thesis aims to initiate the idea by designing an UAV that autonomously travels to a location given some specified coordinates. By forming the system for one UAV that covers an area within its radius, the system can be implemented in additional UAVs in order to together cover larger areas. The UAV is intended to, given some coordinates, autonomously launch from a specialized housing and travel to the specified location. During travel, a video camera is used to capture and send live footage from the flight. When the aircraft reaches its destination it should circulate the accident whilst constantly streaming a live video feed of the scenario back to a ground station.

At any point of the flight, the UAV should be able to receive new directives from the ground control station. Instead of using Radio Control (RC) which is the most commonly used communication medium, this thesis aims to explore the possibility of exploiting the mobile network (3G/4G). By successful implementation, the UAV would not be restricted to operate within a certain radius, only the wireless network coverage would constrain the communication. This would also allow the UAV to be controlled from all over the globe under the assumption that the user has access to the internet.

## 1.2 Related Work

There are companies nowadays that are producing ready-made UAV systems with sophisticated ground station software for commercial use. A company with such an orientation is The UAS Europe [2] that provides portable Unmanned Aerial Systems (UAS) for security, defense and research purposes. UAS supplies their customers with their own autopilot system, portable ground station unit and an impressive software for planning and communicating with the UAV. Similar solutions are easy to setup and user friendly as it concerns the planning of a mission and controlling of the plane. The drawback of using a solution like this is the cost. Normally, large funds are required in order to acquire a complete system like this, hence, it is not viable for a majority of companies.

However, with the recent drastic development within the field of drones, there are now cheap devices available on the market. Drone Deploy [3] offers a user friendly software intended to easily plan missions and directly retrieve data from drones. The software automatically handles the flight planning, manual download and post-processing. Hence,

removing the need for expertise and a large budget. In addition, another company that launched recently with a similar product is the Botlink [4]. Botlink is an experienced team of military and professional pilots, software developers, electrical engineers, and communicators on a mission to create the world's safest and most secure drone operations platform. Their cloud-based platform features fully-automated drone control with manual flying from any smart phone or tablet. Airspace within the application features 100% regulation compliance and includes temporary flight restrictions, active military operation areas and restricted airspace. Similar platforms can be found offering a variety of solutions with various features.

Furthermore, the camera systems that compliments the UAV systems are under rapid development. Fully autonomous video systems are offering live streaming capabilities over wireless communication. Sky Drone FPV [5] is one application that is focused around a sophisticated camera that utilizes the cellular network to transfer a live video stream, all in one package. Such a camera in combination with a gimbal would form a complete visual system for a drone. The combination of such an application can ensure the stability in the incoming video streaming together with high increases precision and quality features.

Another area that has been thoroughly explored refers to the controlling methods used for an unmanned aerial vehicle. Vast research has been conducted around UAVs and a major part of it is found in recent studies. The primary focus lies within the theoretical background that is required in order to setup and fly an unmanned aerial vehicle. The control methods and the path planning techniques are the two most common subjects of interest. There are several available control methods that allows for precise maneuvering and stabilization. Amongst the more famous methods are Linear Quadratic Control (LQR) [6], Model Predictive Control (MPC) [7] and adaptive control methods [8]. Despite the superior performance of these methods, not many are implemented in real applications. The reason for this, according to Kada and Ghazzawi [9], lies within their complexity, nonlinear nature and computational cost. Hence, the commonly used and most implemented method is the traditional linear Proportional-Integral-Derivative (PID) controller. This is mostly due to its low complexity and adequate performance.

Related projects have been initiated with autonomous missions using PID control methods on smaller UAVs. In the paper "Autopilot Design and Path Planning for a UAV" by Henrik Granvist [10] search algorithms were implemented on a virtual flying wing. Although real flight tests were never conducted, the autonomous missions showed promise in simulations. Furthermore, the paper "Autopilot Design for Unmanned Aerial Vehicles by Ingrid Hagen Johansen [11] presents a detailed investigation of the autopilot application. This paper investigates the performance of a PID controller and analyzes robustness with different payloads.

## 1.3 Constraints and Assumptions

The design of a sophisticated landing is not considered for the plane. Instead a parachute is installed on board to minimize the ground impact when a mission is complete. It is assumed that the plane is operating in reasonable calm weather conditions. Finally, due to cost and safety reasons, simulation software will be used for testing and evaluating the model of the UAV before physical implementation. The project is intended as a proof of concept and is not assumed to be robust or safe enough for commercial use.

## 1.4 Swedish Regulations

According to The Swedish Transport Agency's Statute book [12] there are laws and regulations that must be fulfilled in order to design and fly an UAV in the Swedish aerial space. For this specific project the plane that is to be built belongs to the category 1B that specifies: An Unmanned aircraft with maximum take-off weight of more than 1.5 kg but less than or equal to 7 kg, which develops a maximum kinetic energy of 1000 J and is flown only within the visual line of sight of the pilot. Moreover there are rules that must be followed in order to proceed to actual flight of the unmanned aerial system of the category 1B.

# 2

# System Overview

Creating any aerial device requires interaction between several applications and hardware. The complete aerial system, for this project, can be divided into five main categories. Each one of them with its distinct purpose and separate solution. Thus, these categories are investigated independently and merged to build a finished product. The categories are referred to as:

- The Autopilot
- The Airframe
- The Communication Link
- The Ground Control Station
- The Video system

Figure 2.1 shows the parts included in the system. The parts themselves can be sorted into two independent systems which are the aircraft and the video system. Although they both are included in the plane, they are essentially not connected to each other and even have their own communication link.

To be able to travel to a destination the necessary components are an airframe (the structure of the model), autopilot and ground control station. The airframe used for this project are obtained from a Zephyr XL [13]. The wings are combined with a fuselage designed to hold the required electronics. The complete plane is shown in figure 2.2. Every airframe has a set of attributes such as stability, maneuverability, payload capacity etc. These qualities vary with different airframes and hence the choice should be guided by the application requirements. The choice of airframe and autopilot are presented in section 7 and 3 respectively, as the essential parts of an aircraft. The autopilot is the device responsible for controlling the airplane's control surfaces based on sensory values.

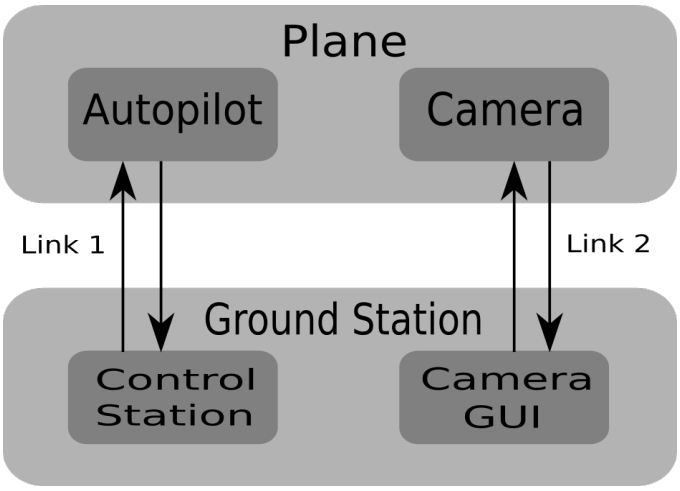



**Figure 2.1:** Illustration of the interaction between the elements involved in the system. The autopilot and camera is on board the plane, communicating with the ground control station through the mobile internet.

It keeps the airframe steady by regulating the control surfaces and does also maneuver the plane to follow a reference trajectory. The autopilot contains the control method and any filter used for signal processing. The trajectory, or flight path, is generated by the Ground Control Station (GCS), described in section 4, and is uploaded to the autopilot when a mission is established. Commonly, the path generated by the GCS is transmitted to the autopilot via radio. However, since radio communication is limited by distance, another approach is using the wireless network for transmitting data. The communication method is described in section 5.

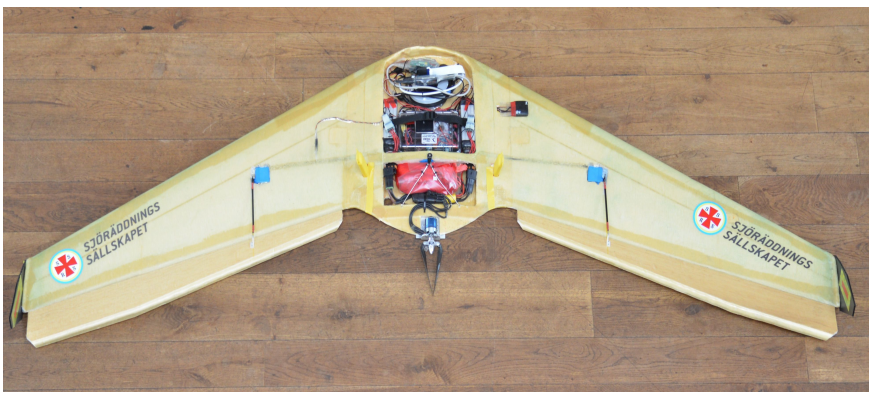



**Figure 2.2:** The top-side of the complete aircraft.

Successful function and integration of the described parts would allow the aircraft to travel to a destination autonomously. However, in order to obtain a live video feed a visual system needs to be implemented. The requirements for such a system is to be

able to transmit the footage immediately to the ground control station and further be able to control the focus of the camera. Combining these attributes grants the user the ability to scout an area beneath the aircraft manually. There are a lot of aspects that affects the performance of these systems. A high resolution footage is appealing but may result in longer delays. Hence, for this system there is a trade-off between quality and speed which is depicted in section 6.

The evaluation of the aircraft is handled through simulations, in prior to actual testing. In simulations the validity of the model is verified and initial controller tuning is performed. The virtual flying is however only an approximation of the physical behavior. The simulation software and evaluation are presented in section 7 followed by the results from the physical implementation in section 8.2.

# 3

# The Autopilot

The autopilot is a platform used to control the stability and trajectory of an aerial vehicle. The autopilot focuses on assisting or taking full control of a vehicle in real-time. Autopilots have evolved significantly over time, early autopilots merely held the attitude control compared to modern autopilots capable of performing fully automated missions. There exists a variety of different autopilots, most of them custom made and tailored to certain airplanes. For smaller foam planes the list of available autopilots is significantly reduced, even more so for the open-source autopilots that allow modifications to the software. These specific boards are very small computers that incorporates a Flight Management Unit (FMU) and an Input/Output board into the same device. The FMU is the autopilot itself, that holds the logic, whilst the I/O board is implemented to enable servo and motor control. In this section these simpler autopilots are introduced and the used platform is described in detail.

## 3.1 Flight Management Unit

The FMU is a specialized computer system that automates a variety of in-flight tasks. The primary function being achieving stability and in-flight management of the reference path. An important tool for the FMU is the Inertial Measurement Unit (IMU) used for determining the plane's current attitude, position and speed. The IMU includes a combination of sensors such as gyroscope, accelerometer, magnetometer and barometer. In addition to the IMU, devices such as a Global Positioning System (GPS) and an airspeed sensor aid in establishing the aircraft's current position.

The retrieved information from these sensors are used to guide the aircraft along a predefined flight path. The aircraft's ability to follow this path and maintain stability is decided by the implemented control algorithm. As mentioned in section 1.2, the most

common control algorithm in UAVs is the PID controller, which is similar to the control method used in this project. However, tuning of the PID gains is needed to achieve a good flight behavior. The control method and tuning of the autopilot used in this project is further explained in section 3.2 and 7.

### 3.1.1 Choice of hardware and software

The implemented autopilot is the *Pixhawk* board with the PX4 flight stack software [14]. This board is popular amongst hobby applications and is a rather new platform compared to its competitors. The alternative autopilot with similar functions is the *APM* board. The main advantage of the Pixhawk is that it runs a 32-bit software architecture compared to the limited 8-bit used by the APM. Additionally, the Pixhawk is running a faster CPU with more memory. With regard to processing power and speed, the Pixhawk is the superior alternative. However, for a flying wing airframe the speed of the autopilot is not as crucial as for a copter. The APM board is capable of managing an aircraft, but to be able to add functionality and tasks to the aircraft, a faster device is needed.
There are two flight management software that are open-source and generally used with the Pixhawk and APM. These are the PX4 and the ArduPilot flight stack. The APM board is only compatible with the ArduPilot flight stack whilst the Pixhawk works as a platform for either one. Hence, choosing the Pixhawk board allows an additional choice regarding the software. Since the ArduPilot is tailored for a 8-bit platform, it will not use the full capacity of the Pixhawk board. Because of this, the PX4 flight stack is the chosen flight management software. The downside of choosing the PX4 flight stack is that it is a relatively new software under constant development. It is not as tested as the ArduPilot and can therefore be missing functions. Furthermore, the PX4 software is likely to be more error prone and even difficult to troubleshoot due to the lack of available information.

## 3.2 Control Method and Tuning

The PX4 autopilot, as many other autopilots, uses a Proportional-Integral-Derivative (PID) control method. The PID controller ensures stable behavior with low complexity, together with a sufficient time response and robustness. However, for fixed wing control, only the Proportional (P) and Integral (I) parts are used in the control loop. The Derivative (D) gain is only involved for regulating quadrocopters. This controller is complemented with an Extended Kalman Filter (EKF) [15], in order to obtain precise sensor values. The control method is coded in the firmware and requires extensive knowledge about the structure to alter. Hence, only the tuning of the controller gains are covered in this report.

For a flying wing airframe, the PX4 controller is only using the Porportional-Integral (PI) part for the angular rate error together with a Feed-Forward (FF) term from the

plain angular error to regulate the pitch and roll. The control loop is not published by the developers of the PX4 flight stack, but since it is an open-source software it is possible to study the code and obtain the structure of the attitude controller. Figure 3.1 shows a control loop diagram for the roll, the pitch control uses the same logic.

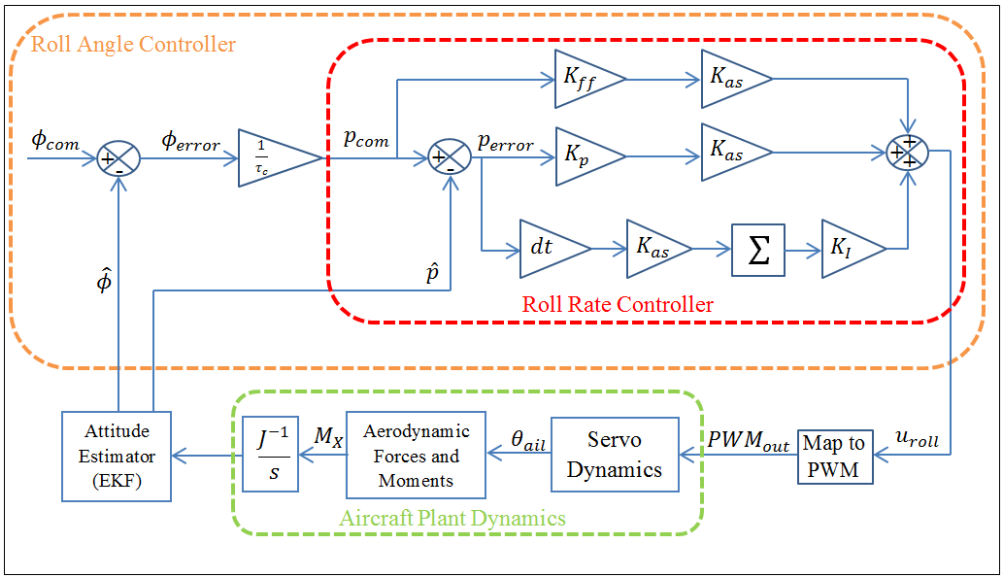



**Figure 3.1:** Control loop for the roll angle. The tuning parameters are the P-, I- and FF gains denoted $K_p, K_I$ and $K_{ff}$ respectively. These are used in series with an airspeed scalar, $K_{as}$ to adapt the gains for different airspeeds. The roll angle is denoted with $\Phi$, the angular roll rate by $p$ and the normalized controller output by $u_{roll}$. The index *com* stands for commanded value by the controller and $\hat{\cdot}$ describes an EKF estimate.

As seen in figure 3.1, the controller for the angle error is basically only a P-controller. The roll angle error translates to an error rate that is multiplied with the feedforward gain, $K_ff$. However, the angular rate controller implements an integral part as well, with anti-windup logic. The sum of the feed forward term and the angular roll rate controller is normalized to give the resulting controller output, $u_roll$.

The control of the pitch and roll is handled by two separate controllers that are tuned independently. Hence, the main parameters for an appropriately tuned controller are the P-,I- and FF terms. These are established using the Ziegler-Nichols heuristic tuning method [16] in a simulation environment. The tuning process if further explained in section 7.

Beside these main parameter gains, there are additional functions that allows higher controller accuracy and personal configuration of the flight behavior. The most significant feature is the *L1 logic* implemented for improved position control, it is a non-linear guidance logic for trajectory tracking [17]. The complete list and explanation of the functions are given by the PX4 developers [18] and the values for this controller is pre-

sented in the appendix A.1. The ones mainly related to controller response and accuracy are further explained in the simulation section 7.

## 3.3 Parachute solution

A controlled landing option is not covered in this project. Hence, a parachute is implemented for reducing the ground impact when terminating a mission. The parachute deployment mechanism is seen figure 3.2, located at the tail of the plane. The parachute is wedged between a spring and a latch that holds it in placed when the autopilot is armed. By commanding the servo holding the latch to move from its mid position, the latch is released and the spring shoots the parachute out from the hatchet. The mechanical solution is simple, however, controlling a servo manually using the PX4 is not.

The PX4 firmware has no independent servo control implemented in its software. In other words, the Pixhawk does not support execution of manual servo commands from the ground control station. To bypass this issue the parachute deployment, and camera retraction described in section 6.3, is implemented as a fail-safe action instead. Fail-safe is a flight mode that executes a safety action for various situations where the plane is considered to be out of control. Using this feature, servo commands can be carried out by the Pixhawk whenever it enters fail-safe mode. These servo commands does need to be predefined before the flight as a part of the start-up script for the Pixhawk and can not be altered during flight.

**Figure 3.2:** Close up picture of the parachute hatchet located at the tail of the plane.

### 3.3.1 Fail-safe action

The fail-safe mode sets the servos to a specific Pulse Width Modulation (PWM) value. The following lines of code are written in the Pixhawk's SD-card as a start-up script:

**Listing 3.1:** Text file in Pixhawk's SD card

```
pwm failsafe -c 1 -p 1500    (Right wing)
```

```
pwm failsafe -c 2 -p 1500    (Left wing)
pwm failsafe -c 3 -p 900     (DC motor)
pwm failsafe -c 4 -p 900     (Camera servo)
pwm failsafe -c 5 -p 900     (Parachute servo)
pwm terminatefail on
```

Where,

- The PWM values range from 1000 (lowest value) to 2000 (highest value), which corresponds to the full movement range of the servo. Any value below 1000 or over 2000 is set to the min/max PWM number.
- *pwm failsafe* is defining the PWM value when a fail-safe/flight termination is triggered.
- *-c <channels>* are the servo channels.
- *-p <pwm value>* is the PWM value.
- The PWM value for armed servos are 1500 (mid position) if nothing is specified.
- *pwm terminatefail on* enables the flight termination.

The PWM values in the file showed in 3.1 are written to the corresponding servos when a fail-safe is induced. The elevons (channel 1 and 2) are set to 1500 (mid position), keeping the plane leveled for the parachute deployment. The throttle (channel 3) is set to the lowest value of 1000 to keep the motor from spinning. The camera servo and parachute servo is set to move away from their armed position (mid position), resulting in a retraction of the camera and deployment of the parachute.

### 3.3.2 Inducing fail-safe mode

To be able to manually deploy the parachute, a custom widget is used in the ground station that sends a MAVlink message that sets the autopilot into fail-safe mode. The widget implemented is not a part of the ground control software and needs to be added into the package. The widget is explained further in section 4. By combining the widget with the fail-safe settings, the parachute can be deployed manually at any time. However, since the autopilot enters fail-safe mode when it is triggered, the system needs to power cycle (restart) in order to function normally again. It is not possible to recover the plane when the fail-safe mode is triggered manually.

### 3.3.3 Additional fail-safe features

Besides using the fail-safe mode as a way to control the parachute deployment and camera retraction, it is also used what it is meant to be used for; as a fail-safe. When flying autonomously it is necessary to have safety precautions for unexpected behavior. There are available settings in the autopilot that specifies what action to take for potentially

dangerous scenarios. The action does need to be similar for every situation, it can be customized to act differently depending on the current mode and system status. The different behavior is set from the parameter list in the autopilot. The common cause for executing a safety action is when all communication is lost with the autopilot, the motor is malfunctioning or that the GPS signal is lost. During an autonomous mission, the following safety actions are applied:

- For RC signal lost and data link lost, i.e. no communication with the aircraft for a couple of seconds, the plane enters fail-safe mode and deploys the parachute.
- For motor malfunction, the plane enters fail-safe mode and deploys the parachute.
- For GPS signal loss, the plane loiters at its position for 2 minutes and tries to regain the signal. If unsuccessful, the plane enters fail-safe mode and deploys the parachute.

# 4

# Ground Station

Although the autopilot is responsible for controlling the plane, the ground station is the software that gives the commands. Every flight path is generated by the ground control station and transmitted as a reference trajectory for the autopilot to follow. It is the software that allow users to interact with the aircraft, specify the flight path and track its movement. A ground control station is a standard tool for controlling UAVs. However, they are often tailored for certain autopilots. For the Pixhawk autopilot with the PX4 FMU, the QGroundControl (QGC) [19] station and MAVproxy [20] are the compatible control stations. QGroundControl station is specialized for micro UAVs that uses the MAVlink protocol, see chapter 5, for communicating with the autopilot. Other popular ground stations used for the same application are Mission Planner, APM Planner and DroidPlanner. These uses the MAVlink protocol but are not suited for the PX4 flight stack.
The main features of QGroundControl are presented below:

- Open-source MAVLink Micro Air Vehicle Communication Protocol
- 2/3D aerial maps with drag-and-drop waypoints
- In-flight manipulation of waypoints and onboard parameters
- Real-time plotting of sensor and telemetry data
- Logging and plotting of sensor logs
- Support for UDP and serial (radio modem)

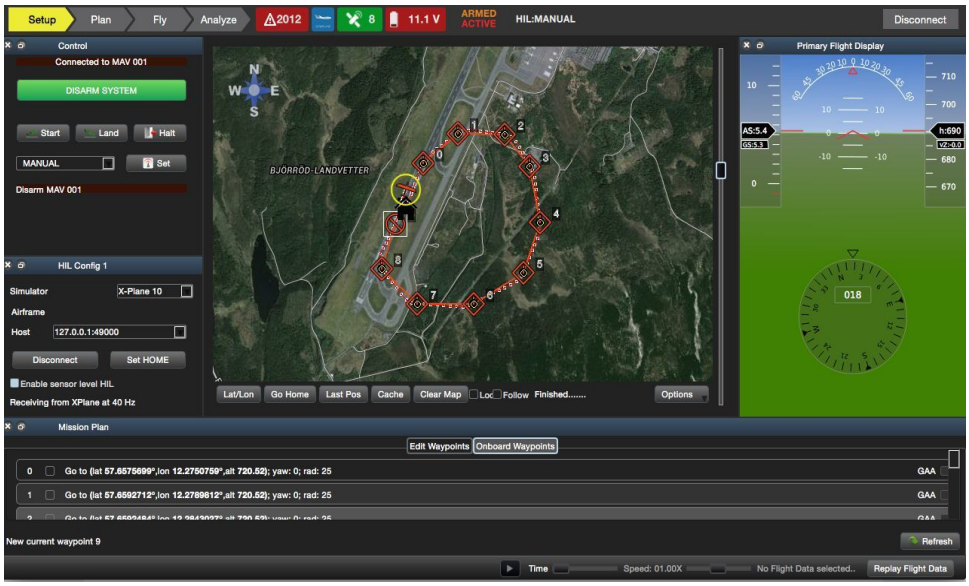



**Figure 4.1:** Graphical User Interface for QGroundControl

In figure 4.1 the Graphical User Interface (GUI) for QGroundControl is displayed. The ground station provides information about location, altitude, attitude, warnings and other relevant data about the plane. The main features of this program is the ability to generate flights paths by setting way-points. After a way-point is set, the shortest route is uploaded to the autopilot and an autonomous mission can be initiated. It is the tuning of the autopilot that determines the aircraft's ability to follow the reference trajectory. Besides being used as a tool for generating flight paths, the control station is also used for sensor calibration and controller tuning.

## 4.1 Customized Widget

The QGroundControl package provides widgets for additional functions and information. However, none of these support manual control of servos nor the possibility to induce fail-safe. Since this is needed for the parachute deployment and camera retraction to be executed, as explained in section 3.3, a widget is customized to induce fail-safe mode. The customized widget sends a MAVlink command, MAVlink is explained in section 5, that sets the autopilot into a flight termination state. Flight termination induces the same action as for fail-safe, successfully deploying the parachute. The widget is presented as a button in the ground control interface and the code is presented in the appendix A.5. On button click, the command is sent and the autopilot enters fail-safe mode.

# 5

# Communication

As stated in the introduction, in section 1.1, this thesis aims to utilize the mobile wireless network in order to establish communication between the autopilot and ground station. The UAV and the ground station should be able to send and receive information to successfully execute an autonomous mission. To do so the autopilot's default communication protocol, Mavlink, is used.

## 5.1 Mavlink Protocol

MAVLink is a light library protocol, specifically designed for small air vehicles. It uses C language structures and is a single line message protocol [21]. This structure makes the protocol easy to understand and to use for custom commands, as the widget described in section 4.1. Each message corresponds to a specific command, list of all commands can be found in the MAVlink documentation [21]. The flight termination widget uses the command:

$$controller.sendCommand(185, 50, 3, 1, 0, 0, 0, 0, 0, 0)$$

Where the parameter ID, 185, refers to flight termination and the fourth parameter, set to 1, enables the termination. This is a standard way to send commands to the autopilot and the MAVlink documentation shows the available functions. The MAVlink protocol supports sending manual PWM values to a any servo, making it trivial to manually control parachute deployment and camera retraction. The reason for not sending a PWM value directly to the autopilot, is that the use of this command is not implemented in the PX4 firmware.

## 5.2 Communication link between aircraft and QGroundControl

The QGroundControl software [19] is utilized as the ground station unit for controlling the UAV. There are several methods and applications that can be used in order to achieve a two way communication between the GCS and the autopilot. A communication bridge between these two components must ensure long range capabilities as well as a stable and continuous link. For a data link to be established between the autopilot and the QCS, both require a internet connection at all times. By running the GCS on a laptop or tablet, an internet connection is obtained using either Wi-Fi, modem or another device to create a hotspot (wireless access point).

On the other side of the link, the autopilot requires access to the internet in order to be able to receive or send information. Since the autopilot can not directly manage information from a modem, an external device is required to forward the MAVlink messages to the autopilot. A suitable forwarding tool for the MAVlink protocol is a simple ground control station called MAVproxy [20]. MAVproxy can be used to distribute MAVlink messages between devices, such as a modem and the Pixhawk. Implementing MAVproxy as the communication bridge between the autopilot and a modem, requires hardware capable of running MAVproxy. A small on board computer connected to the autopilot is used as a distributor between modem and autopilot. A reasonable and sufficient board is the Odroid XU3 lite platform together with a high-speed modem, Huawei 4G E3276 5.2. The operating system (OS) installed on the Odroid is a light version of Ubuntu, called Lubuntu v14.04. The Lubuntu is compatible with the Huawei modem and fully supports the MAVproxy software.

A potential alternative besides the MAVProxy approach is using a SIM900 GSM module, substituting the MAVproxy and Odroid board. The SIM900 is a compact and reliable wireless module capable of User Data Protocol (UDP). Each of these two solutions are capable of successfully establishing a link, but even though they are using the same logic they differ as it concerns the implementation techniques. The connection between the autopilot and ground control is illustrated in figure 5.1

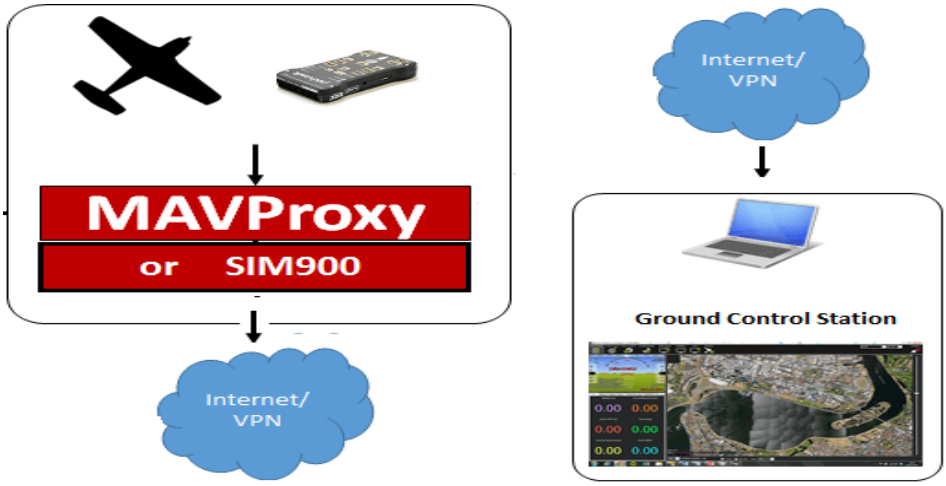



**Figure 5.1:** MAVproxy under Linux

(a) Odroid XU3 Lite (b) Huawei E3276

**Figure 5.2:** Communication Bridge Components

## 5.3 Communication setup and link description

The Odroid and the Pixhawk is connected physically by a telemetry link, needed to ensure that the connection is reliable [22]. Furthermore, the Odroid is supplied with a modem to be able to connect to the mobile internet. At this point the Pixhawk can communicate with the Odroid and the Odroid can access the internet. In order to create a tunnel between the modem and autopilot in the Odroid, an additional ground control station is used; MAVproxy [20]. MAVproxy is a light-weight fully-functioning control station especially designed for MAVlink messages. It doubles as a ground control station extension and is capable of acting as a router for MAVlink messages. Hence, it

is used as the communication bridge between the autopilot and the modem in the Odroid.

To extend the communication from the modem to QGroundControl, MAVproxy needs to send the data from the autopilot to the correct Internet Protocol (IP) address on which the GCS is found. The Odroid is set to send data to the same static IP address at all times. Consequently, in order to connect to the GCS, the device running the GCS needs a specific fixed IP address.

For the autopilot to connect to a specific IP at all times at any device, a Virtual Private Network (VPN) tunnel is used on the ground station side. By connecting the ground station to a certain VPN server the data sent to that server's IP is re-routed to the GCS. This step requires that the VPN is set to forward data to selected ports. Hence, the Odroid sends the data to the same VPN server at all times and that VPN server forwards to the device currently connected to the server.

The full connection between the Pixhawk and QGroundControl is initially established when QGroundControl receives a heartbeat message indicating that the Pixhawk is alive. This is a User Datagram Protocol (UDP) message that is sent continuously from the autopilot to show that it is responsive. The heartbeat is forwarded by the Odroid to the static IP of the VPN server. The VPN port forwards the data to the connected device that is running the GCS. The heartbeat message also specifies what channel the package is sent over, telling the GCS where to send data in order to establish a connection. Finally, with the communication between the Pixhawk and MAVproxy successfully established, the ground station is now able to communicate with the UAV through the UDP connection.
The MAVproxy commands below, further explained in A, are executed in the Odroid's terminal at start-up and initiates the forwarding of messages between the autopilot and the VPN:

**Listing 5.1:** Script that initiates forwarding between autopilot and ground control station.

```
mavproxy.py --master=/dev/ttyACM0 --baudrate 57600
 --out IPADDRESS:14550 --console --dialect pixhawk
```

The command forward information between the port at *master=/dev/ttyACM0* and *out IPADDRESS:14550*, where the *IPADDRESS* refers to the public IP address of the VPN server that the GCS device is connected to.

## 5.4 Odroid's Start-up Scripts

In order to initiate the data link, some commands are executed at the start-up of the Odroid. To avoid manually enter these at boot, scripts are used and run automatically when the Odroid is powered. The scripts do nothing else than executing the MAVproxy command shown above in 5.1.

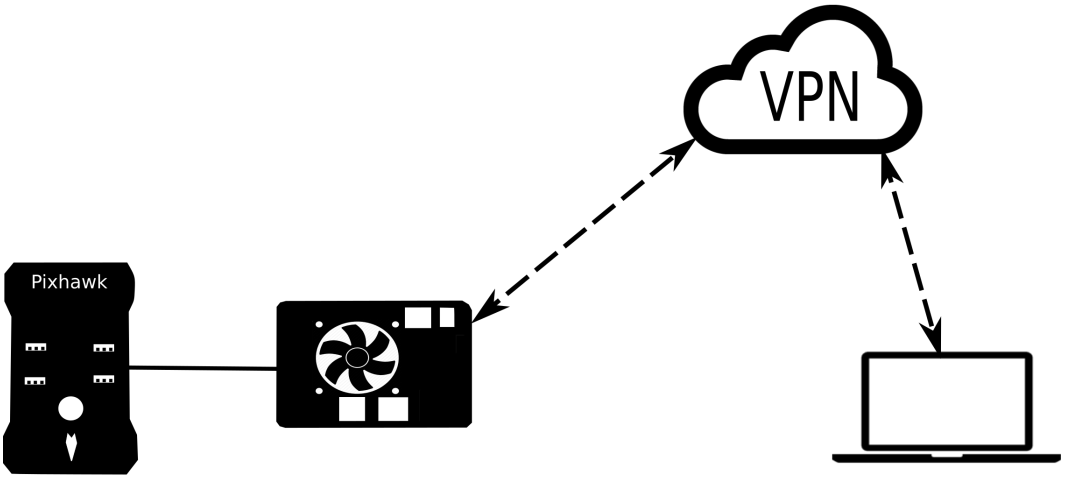



**Figure 5.3:** Telemetry Communication Setup. On the left side the autopilot is shown connected with the Odroid that communicates with the ground station through a VPN tunnel.

**Listing 5.2:** Script that initiates forwarding after a 25 second delay.

```
#!/bin/bash
sleep 25
mavproxy.py --master=/dev/ttyUSB0 --out=udp:62.102.148.187:14550
 --baudrate=921600 --console --dialect=pixhawk
```

Where **sleep 25** delays the execution of the command, giving the modem time to establish an internet connection. For the Lubuntu system to execute this specific command automatically, the user needs root privileges. This is bypassed by creating another script that sets the root privileges before the execution of the MAVProxy script.

**Listing 5.3:** Script to enable root privileges for MAVproxy.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    setuid( 0 );
    system( "/home/odroid/Mavproxy.sh" );

    return 0;
}
```

Moreover, the application needed that launches the MAVproxy software when the system is powered up must be created. The Lubuntu software "Unity Launcher And Desktop Files" is utilized for the start-up script. These files are stored in the Odroid with the extension **.desktop** and they are used for launching specific applications. The desktop file in this case is set to start on launch and is presented below:

**Listing 5.4:** Setting to automatically run scripts at boot

```
[Desktop Entry]
Version=1.0
Name=Mavproxy
Comment=Enable Mavproxy
Exec=lxterminal -e /home/odroid/Runscript
Icon=/home/odroid/Pictures/connect.png
Terminal=false
Type=Application
Categories=Utility;Application;
X-KeepTerminal=true
```

To clarify, this is the script execution order when booting the Odroid:

- System boots and uses the embedded start-up software to execute all ".desktop" files.
- The script 5.3 runs and sets the root privileges.
- The MAVproxy script 5.2 is executed which after a delay starts the data forwarding of UDP messages, at which time it is possible to connect with the ground station.

# 6

# Real-Time video

An essential part for the system is the ability to quickly obtain visual information from a situation. Moreover, it is useful to be able to continuously supervise its development. For a fixed wing aircraft to loiter at a location it needs to be in constant motion, often circulating the scene. Because of this, the visual system must be able to compensate for its motion. The camera system used for this application is fully independent, i.e., it does not interact with the autopilot in any way. It consists of a camera, router, modem and a GUI for control.

## 6.1 On-board Camera

The on-board camera is a Canon VB-S31D seen in figure 6.1 with specifications shown in appendix A.


**Figure 6.1:** Canon VB-S31D

The key feature of this camera is the possibility to pan and tilt the viewing angle within a small dome. The aerodynamics of a fixed wing aircraft are important for good stability

and all external additions do affect the flight performance. Hence, the part of the camera on the exterior of the fuselage is preferably small and smooth. The camera can be manually controlled from a GUI which allows pan, tilt and zoom. However, the camera does not have gimbal properties. A gimbal is the ideal tool for eliminating the moving orientation of the aircraft and keeping the focus fixed. These are popular on copters, for their image stabilizing features. However, these gimbals are not suitable for fixed wing aircrafts due to the increased aerodynamic drag.

The Canon VB-S31D does not have any target locking features and needs to be manually controlled to compensate for orientation changes of the aircraft. The control of the pan/tilt/zoom features are accessible through a web browser. The mobile network is used to support the communication between the camera and its GUI.

## 6.2 Video transmission through the mobile network

The Canon VB-S31D is a network camera, or Internet Protocol (IP) camera. By combining this camera with a router and a modem, a completely wireless surveillance system is obtained as illustrated in figure 6.2.

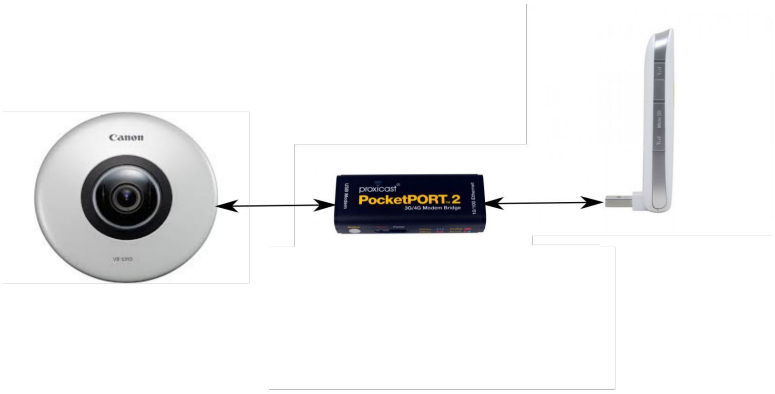



**Figure 6.2:** Complete video system; Canon VB-S31D, PocketPORT 2 and Huawei modem.

The aircraft is already equipped with a modem used for the communication between the ground control station and the autopilot. However, instead of tapping in to this modem for visual control, the camera is assigned its own separate modem in order to have access to its full bandwidth. A major concern when streaming live footage is the data transfer rate. Slow connections results in poor video quality and significant delays.

The PocketPort is a 3G/4G cellular modem to ethernet bridge and allows easy communication between the IP camera and the modem. Every PocketPORT is running a Domain Name System (DNS) update service, making the PocketPORT accessible at the same IP address at all times. Hence, it is reachable through a web browser from anywhere, assuming that an Internet connection is available. By accessing the PocketPORT the

camera's GUI is displayed and the camera can be controlled.

## 6.3 Protection by retraction

The camera is a sensitive device not built to handle repeated impacts. In order to avoid damage to the hardware or scratches to the dome; the camera is given the ability to retract into the fuselage of the aircraft. This feature is intended to mitigate damage at ground impact, hence, should only be used before landing. When a mission is terminated, the parachute deploys and the camera retracts as a part of the fail-safe, explained in section 3.3. The structure that allows the camera to retract is designed in the software SolidWorks and printed using a 3D-printer. Figure 6.4 shows the virtual model of the camera together with the 3D-printed mount in its neutral position. The servo controlling the position of the structure is mounted at the axis. When the system is armed and in flight, the camera is resting at the bottom of the fuselage with the dome going out from a hole in the belly of the plane. When fail-safe mode is active the servo pulls the camera up into the fuselage as in figure 6.3 to protect it during ground impact. Damping plugs are also attached to the structure for reducing vibrations.

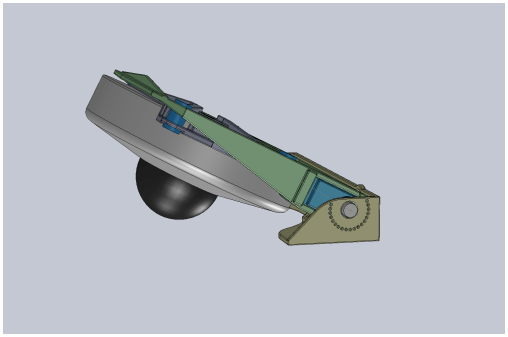


**Figure 6.3:** Camera mount in retracted position.

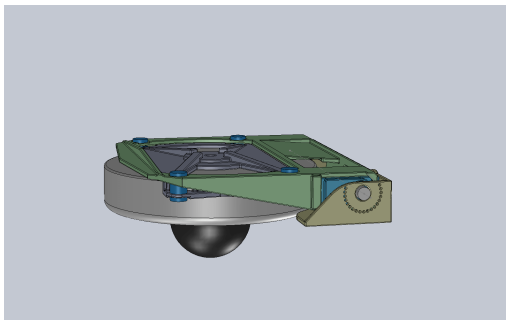


**Figure 6.4:** Camera mount in neutral position.

# 7

# Simulation

The main purpose of the simulation is to be able to tune the PI controller, described in section 3.2, within the autopilot before the initial flight tests. Without a reasonably tuned controller there will be difficulties stabilizing the aircraft which can result in a crash. The autopilot is initially tuned using Hardware-In-The-Loop-Simulation (HILS). In a HILS setup the autopilot is connected to a computer running a simulation program. The autopilot feeds servo commands to the simulator which responds with sensory values from a simulated plane. The simulation environment is in a sense overriding the sensory inputs of the IMU. Achieving flight stability in the simulation environment indicates that the controller is operating normally and is approximately tuned. The residual tuning error is dependent on the accuracy of the virtual model as well as the physics engine of the simulation environment. In this chapter, the flight simulator, the virtual model and tuning process is presented.

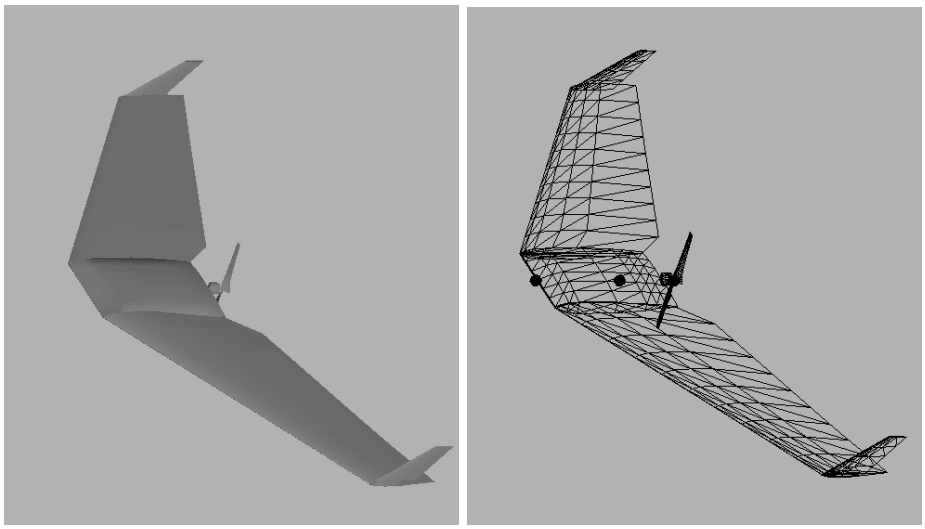
## 7.1 Modeling in X-Plane 10

*X-Plane 10* is a realistic flight simulator capable of HILS and supports custom made aircrafts. With the X-plane package there is additional software such as *The Plane Maker* and *The Airfoil Maker*. The purpose of these programs are to be able to custom design an aircraft and validate the architecture using the flight simulator. Using these programs it is possible to construct a virtual model of the modified Zephyr XL aircraft based on its form and dimensions.

### 7.1.1 Airframe

The airframe is describing the exterior of the aircraft and its control surfaces. The airframe used in the simulation is designed in *The Plane Maker* and is shown in figure 7.1(a) and 7.1(b). This is an approximation of the modified Zephyr XL and is based on the dimensions of the physical aircraft. The complete airframe consists of the wings,

fuselage and vertical stabilizers. These are designed from several segments, as seen in figure 7.1(b). The most significant inaccuracies in the model is the shape of the fuselage and the thrust capabilities of the Direct Current (DC) motor. Although the dimensions are correct, the additional hardware attached to the fuselage have not been accounted for in the model. This is mainly due to limitations within the software.



(a) Virtual Model (b) Virtual Model with segments visible

**Figure 7.1:** Simulation model in X-Plane

### 7.1.2 The Airfoil

Even though the airframe of a model is accurate in the simulator, it does not completely characterize the aircraft's aerodynamics. For the virtual model to mimic the physical plane, the airfoils need to be similar. An airfoil, exemplified in figure 7.2, is the shape of a wing or blade (of a propeller, rotor, or turbine) seen in cross-section. An airfoil-shaped body moved through a fluid or air produces an aerodynamic force. The component of this force perpendicular to the direction of motion is called lift. The component parallel to the direction of motion is called drag. The airfoils characteristics determines the lift and drag attributes of the wing. Hence, they affect the aerodynamics of the aircraft and can therefore have impact on the controller tuning. To mimic the flight characteristics of the physical aircraft, the airfoils needs to be similar to the Zephyr XL.

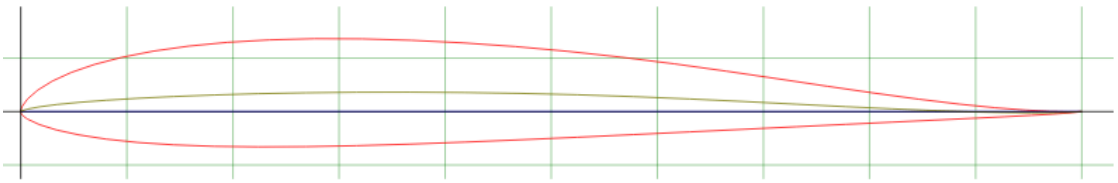


**Figure 7.2:** The shape of the MH 60 airfoil used in simulations.

An extended list of airfoils is available for all kinds of aerial vehicles. However, the list of airfoils for smaller flying wings, such as the Zephyr XL, is not as extensive. The data for the airfoil that the plane uses is not known completely and requires extensive knowledge and testing to calculate. The used airfoil in the simulation environment is an approximate airfoil common amongst smaller flying wings, known as MH 60. The choice of MH 60 is based on approximate airfoil thickness and shape. The shape of the airfoil is shown in figure 7.2 and the necessary data is given by Dr Martin Hepperle [23]. The data for this airfoil is analyzed in JavaFoil, which is needed to adapt the airfoil into the plane maker program. JavaFoil uses the information about the airfoil to calculate attributes such as lift and drag at different angles of attack. After the analysis in JavaFoil the airfoil is imported to the plane maker program and used by X-Plane 10's physics engine during flight simulation.

### 7.1.3 The Control Surfaces

The airframe sets the specifications for the control surfaces of the aircraft as well. Elevons and the motor propellers are the only control surfaces in this design and are illustrated in figure 7.3. The elevons are a combination between the traditional elevator (used for pitch control) and the aileron (used for roll control) , shown in figure 7.4. Elevons are suitable for tailless aircrafts such as flying wings, where an elevator or rudder is not an option. The location of the elevons are on each side of the aircraft at the trailing edge of the wing.

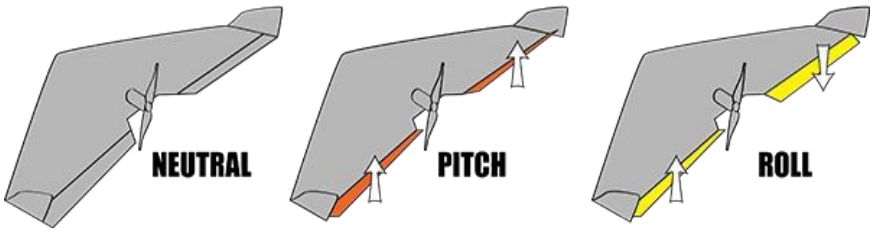



**Figure 7.3:** Elevons

When the elevons move at the same direction, the aerial vehicle changes its pitch angle. When moved in opposite direction, the aircraft alters its roll angle. These movements

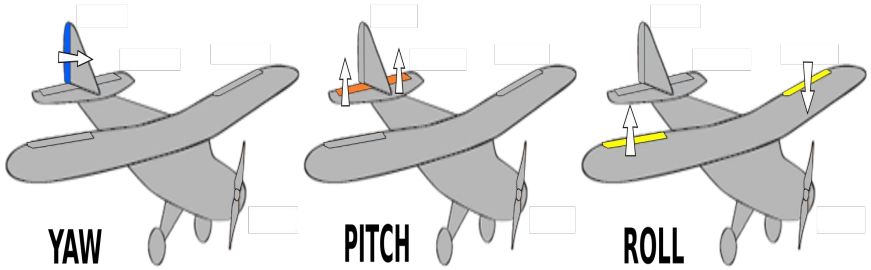



**Figure 7.4:** Control Surfaces of a standard airplane

can also be combined, enabling full control of the aircraft. However, the use of elevons require mixing of the control signals to the servos. This is necessary since pitch and roll are controlled by the same servos and need to interact to operate successfully. The inputs of the control signals are mixed mechanically and electronically in order to achieve the correct movement for each elevon.

### 7.1.4 Channel Mixing Configuration

Adjusting the mixing of the channels for creating an elevon control surface requires modification of the default mixer files used by the flying wing airframe. This file defines mixers suitable for controlling a flying wing aircraft using the PX4 FMU. The configuration assumes that the elevon servos are connected to servo output channel 1 and 2 and the motor speed control to output channel 3. Output 4 is assumed to be for camera control and output 5 assigned for the parachute deployment, these are not involved in the mixing. The file contains the following start-up script:
*Flying Wing Elevon mixers*

```
%Channel 1 - Right Servo
M: 2
O:      10000  10000      0 -10000  10000
S: 0 0  -7500  -7500      0 -10000  10000
S: 0 1  -8000  -8000      0 -10000  10000
%Channel 2 - Left Servo
M: 2
O:      10000  10000      0 -10000  10000
S: 0 0   7500   7500      0 -10000  10000
S: 0 1  -8000  -8000      0 -10000  10000
%Channel 3 - DC motor (default)
M: 1
O:      10000  10000      0 -10000  10000
S: 0 3      0  20000 -10000 -10000  10000
```

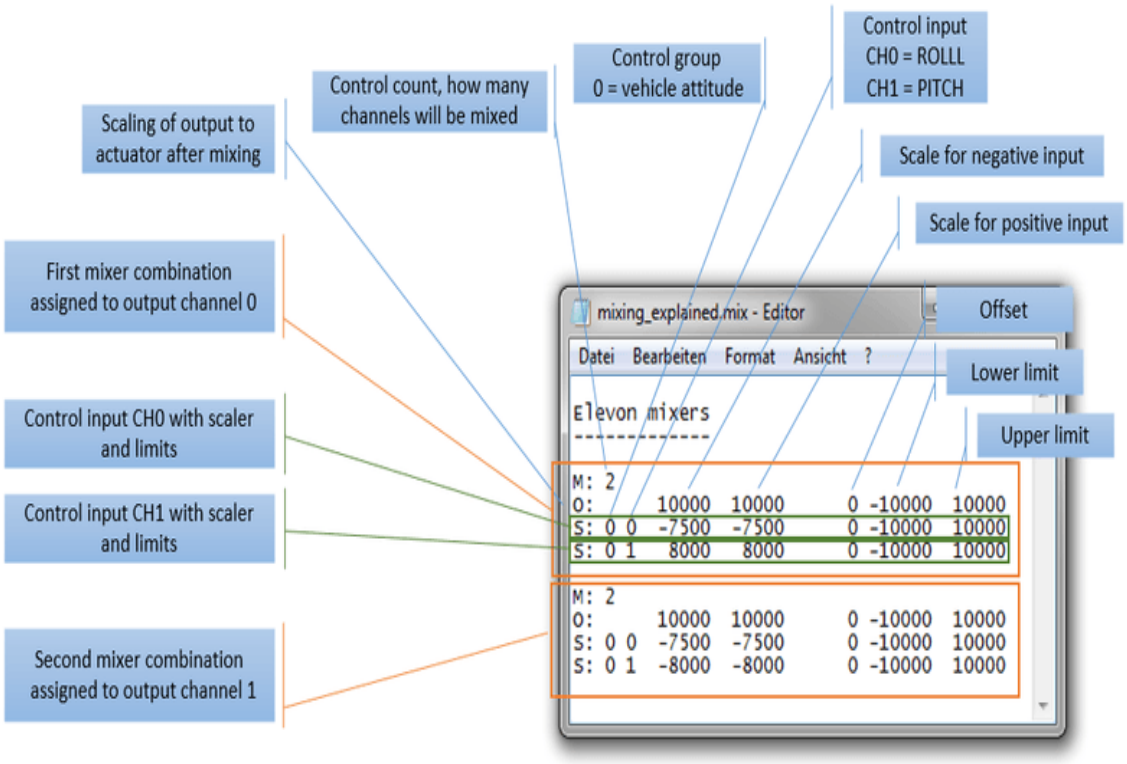



**Figure 7.5:** Mixing code layout

Figure 7.5 shows how the mixing file is read by the Pixhawk. The first mixer combination (orange rectangle) corresponds to the mixing for servo controlling the right elevon, the second mixer combination (bottom rectangle) to the servo controlling the left elevon etc. The file basically maps control commands so that the servos move as intended. This is necessary for both manual control and when performing an autonomous mission. Note that the mixing is also dependent on servo placement on the airframe. A plane will not fly with faulty mixing, even slight errors can make the plane stall easily because of too large elevons movements.

### 7.1.5 Center of Gravity

Since the flying wing is a tailless aircraft, the Center of Gravity (CG) is a vital factor. Even small deviations from the appropriate CG can result in stabilization issues. The appropriate CG is also dependent on the aerodynamics of the aircraft. For example; different lift attributes at the tip of the plane affects the suitable position for the center of gravity. According to the creators of the Zephyr XL, the CG should be located at 28 centimeters from the tip [24]. All additional payload that is added should be placed in a manor that keeps the position of the CG. Figure 7.6 illustrates where the CG is located in relation to the plane.

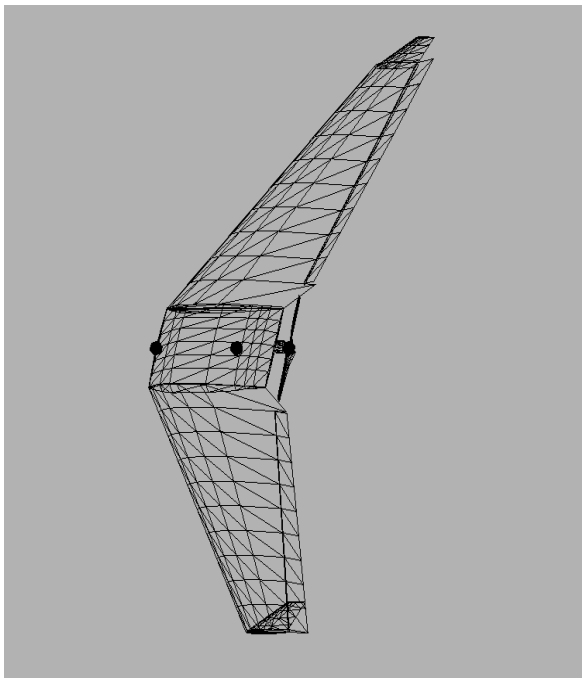


**Figure 7.6:** The Center of Gravity illustrated in the virtual model. The black dot in center indicates the CG point on the aircraft.

## 7.2 Simulate a flight

The performance of the autopilot is analyzed using the HILS setup with the Pixhawk autopilot and X-Plane 10. In figure 8.1 the used GUI from QGroundControl is visible together with a map over Landvetter airport in Sweden, used as the simulation location. A mission is initiated by placing way-points at the desired destination and uploading these to the autopilot. The way-points can be altered and uploaded at any time during the flight, the aircraft adjusts its heading accordingly. After the mission is finished the plane loiters at the last way-point if not instructed to land. For this project, the loiter feature is suitable since it is intended to circulate the scene when reaching its destination.

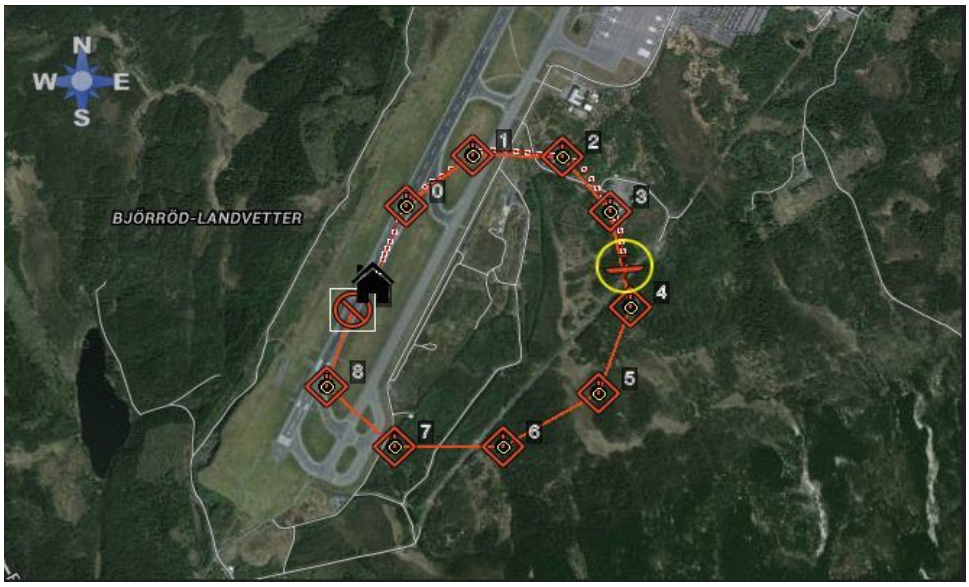



**Figure 7.7:** Shows the execution of a simple autonomous mission in the simulation environment. The map is displayed in the QGroundControl interface.

Figure 8.1 shows the ground control station during a simulated flight. At the same instance, the animation of the simulation is visualized in X-Plane shown in figure 7.8. The original scenery is replaced by water in X-Plane. The animation feature is very useful for analyzing the behavior of the plane, especially during the controller tuning process.

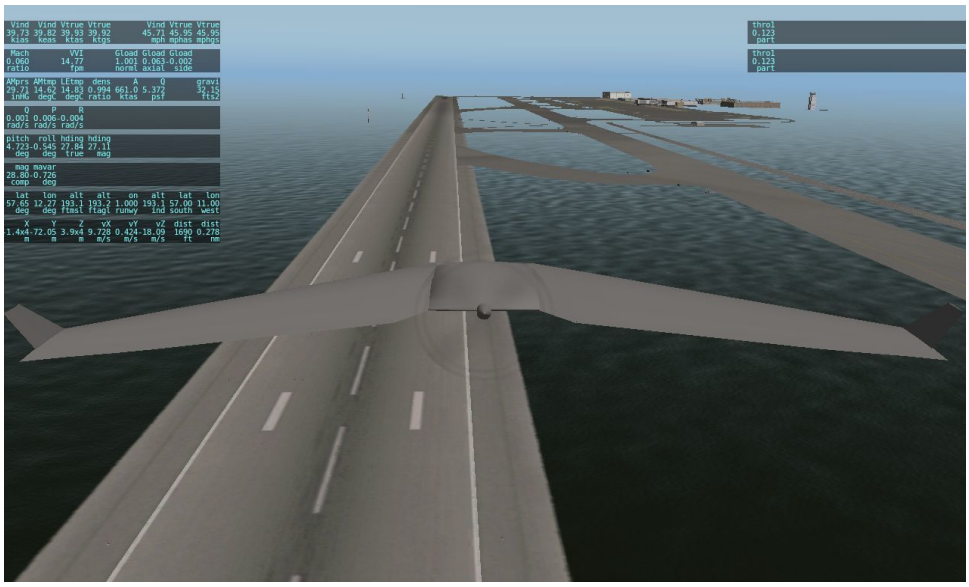


**Figure 7.8:** Shows an illustration taken while the aircraft was executing a mission in the simulation environment.

## 7.3 Tuning in simulation

The Ziegler-Nichols tuning method [25] is a heuristic approach where the control response needs to be observed for each change. By the use of a simulation environment such as X-Plane, the feedback is available without endangering the physical plane. The goal is to achieve acceptable stability and response time. In order words, be able to travel to a destination and loiter around a waypoint.

The control loop parameters are used for altering the behavior of the aircraft. Correctly modifying and configuring its properties results in a stable flying vehicle. As mentioned in section 3.2, the P-,I- and FF gains are essential but not the only parameters involved. Other properties are also configured as the roll-to-thrust compensation, launch settings, fail-safe operations etc. A full list of all the parameters can be found at [18].

### 7.3.1 Tuning Procedure

To setup the tuning procedure the plane is set on a autonomous mission that consists of a number of waypoints. This route is repeated continuously, i.e., when the mission is complete the plane travels to the first waypoint and repeats. During this indefinite flight it is possible to change the gains and study the roll and pitch control response. Both the commanded pitch/roll and the actual pitch/roll is showed as the plane is maneuvering along the path. The proportional and feedforward gains are tuned independently for both the pitch gains and the roll gains. Each parameters is increased until an oscillation is displayed in the control deflections and then decreased according to the Ziegler-Nichols

rules. The integral gain is increased based on residual errors in the attitude or altitude.

The tuning procedure is repeated iteratively for both the roll and pitch controller, until the route is completed smoothly. Besides the P-, I- and FF gains, other parameters are altered by trial and error. These are adjusted after the tuning of pitch and roll to improve the behavior of the autopilot.

### 7.3.2 Relevant Parameter Changes

A large number of parameters is available for the specific category but not all of them are needed for deriving a stable aerial vehicle. The parameters that are used for this specific airframe are presented and defined below.

- **Maximum Airspeed (m/s) : FW_AIRSPD_MAX = 22**
  Maximum airspeed allowed, if the airspeed reaches values higher than this the controller will try to decrease the speed.
- **Minimum Airspeed (m/s) : FW_AIRSPD_MIN = 12**
  Minimum airspeed allowed,if the airspeed reaches values higher than this the controller will try to increase the speed.
- **Trim Airspeed (m/s) : FW_AIRSPD_TRIM = 22**
  The controller will try to fly at this airspeed.
- **Pitch rate feed forward : FW_PR_FF = 0.1**
  Direct feed forward from rate setpoint to control surface output
- **Pitch rate integrator gain : FW_PR_I = 0.02**
  This gain defines how much control response will result out of a steady state error. It trims any constant error.
- **Pitch rate proportional gain : FW_PR_P = 0.04**
  This defines how much the elevator input will be commanded depending on the current body angular rate error.
- **Pitch Setpoint Offset (deg) : FW_PSP_OFF = 15**
  An airframe specific offset of the pitch setpoint in degrees, the value is added to the pitch setpoint and should correspond to the typical cruise speed of the airframe.
- **Positive pitch limit (deg) : FW_P_LIM_MAX = 40**
  The maximum positive pitch the controller will output.
- **Negative pitch limit (deg):FW_P_LIM_MIN =-40**
  The minimum negative pitch the controller will output.
- **Roll to Pitch feedforward gain : FW_P_ROLLFF =2**
  This compensates during turns and ensures the nose stays leveled.

- **Roll rate feed forward : FW_RR_FF =0.2**
  Direct feed forward from rate setpoint to control surface output. Use this to obtain a tighter response of the controller without introducing noise amplification.
- **Roll rate integrator Gain : FW_RR_I = 0**
  This gain defines how much control response will result out of a steady state error. It trims any constant error.
- **Roll rate proportional Gain : FW_RR_P=0.045**
  This defines how much the aileron input will be commanded depending on the current body angular rate error.
- **Pitch damping factor : FW_T_PTCH_DAMP =0.1**
  This is the damping gain for the pitch demand loop. Increase to add damping to correct for oscillations in height.
- **Roll to Thrust feedforward : FW_T_RLL2THR = 45**
  The amount of throttle that will be used to compensate for the additional drag created by turning.

The remaining tuning parameters are presented in the appendix A.1

## 7.4 Loitering

The usage of a fixed wing aircraft causes some additional difficulties when monitoring a scenario. The purpose of the system is to be able to survey a fixed point on the ground. This goal puts constraints on the loitering radius and altitude. Having a large loiter radius whilst keeping a low altitude might put the point of interest out of Line Of Sight (LOS) for the camera. This event is illustrated in figure 7.9 and mainly caused by the roll angle of the aircraft while turning. To avoid the phenomenon, the altitude needs to be increased or the loitering radius decreased. However, decreasing the loiter radius also increases the roll angle of the aircraft; effectively directing the camera away from the scene. This is exemplified in figure 7.11.

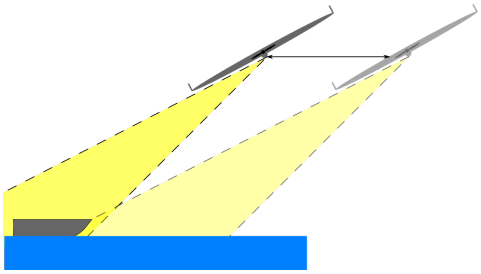


**Figure 7.9:** Loiter in relation to the longitude of the UAV

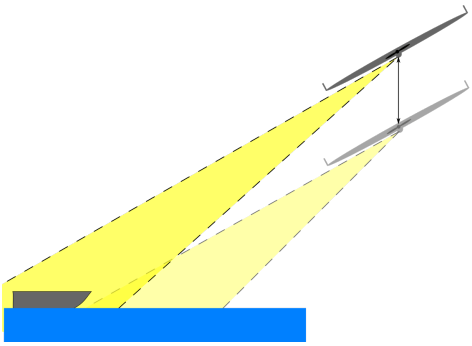


**Figure 7.10:** Loiter in relation to the altitude of the UAV

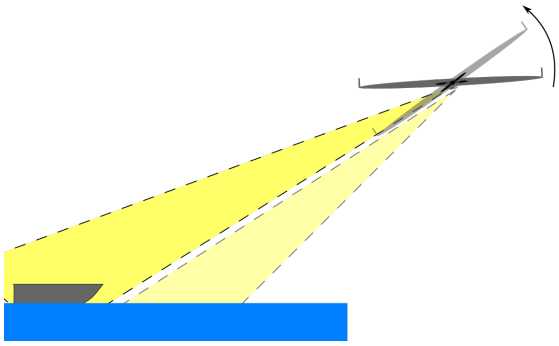


**Figure 7.11:** Loitering in relation to the rotation of the UAV

Increasing the altitude in which the plane operates is the easiest way to ensure a clear LOS at all times, as in figure 7.10. Although, the altitude is not allowed to go higher than 120 meters according to the Swedish regulations for drones [12]. Moreover, as the distance to the target increases the resolution of the video footage declines. It is possible to calculate the minimum operating altitude based on information about the loiter radius and roll angle. From figure 7.12 some relations can be established that ensures a visible LOS for the camera.

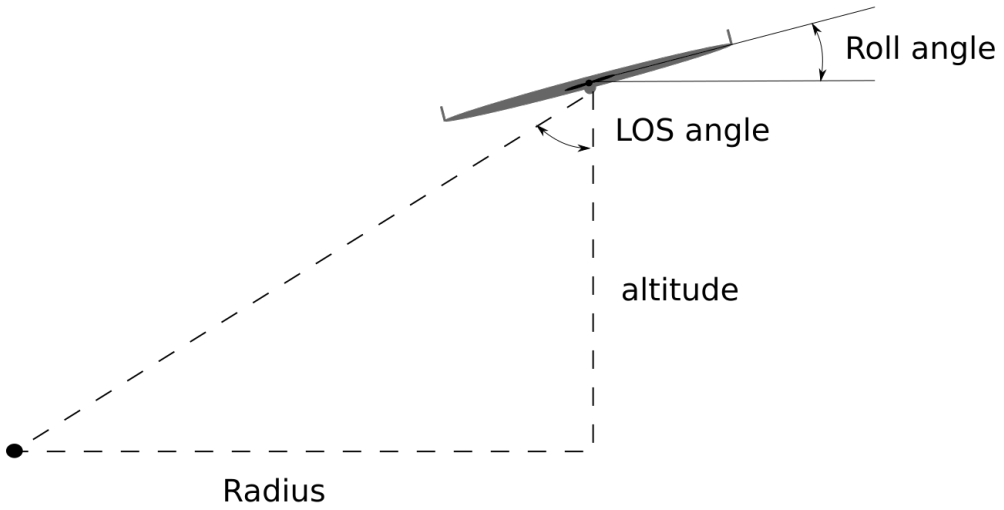



**Figure 7.12:** Illustration of the relations between the angle , radius and and altitude of the UAV. The Roll angle and the LOS angle is denoted $\Phi$ and $\alpha$ respectively in the calculations.

For a camera with a 90° tilt angle:

$$\phi + \alpha < 90 \tag{7.1}$$

Where $\phi$ is the roll angle of the plane and $\alpha$ is the LOS angle. The LOS angle $\alpha$ is described by:

$$\alpha = tan^{-1}\left(\frac{altitude}{radius}\right) \tag{7.2}$$

By combining the equations 7.1 and 7.2 the expression for minimum altitude is obtained:

$$altitude > radius \cdot tan(90 - \phi) \tag{7.3}$$

Equation 7.3 should be kept in consideration for setting the desired altitude in a autonomous mission.

# 8

# Evaluation

In this chapter the performance of the aircraft is evaluated, both in simulation and physical implementation. Additionally, the function of the video system is analyzed, the communication link and the construction of the aircraft. The main goal of the system is to be able to travel to a destination, loiter and send footage to the ground station.

## 8.1 Simulation results

The main results gathered from the simulations are the controller tuning and model verification. There are two criteria that need to be fulfilled in the simulation in order to achieve the goals of the project. Primarily, the plane needs to be able to travel to a destination. Secondly, it needs to be able to loiter, preferably at an altitude high enough for the camera to keep a fixed focus on a point on the ground.

### 8.1.1 Test - Travel to a destination

To test that the plane operates satisfactory, two different routes are tested in the simulation environment. One simple route is used to verify basic functions such as launching and turning. Furthermore, another complex route is generated that tests a search behavior and the turning capabilities of the plane.

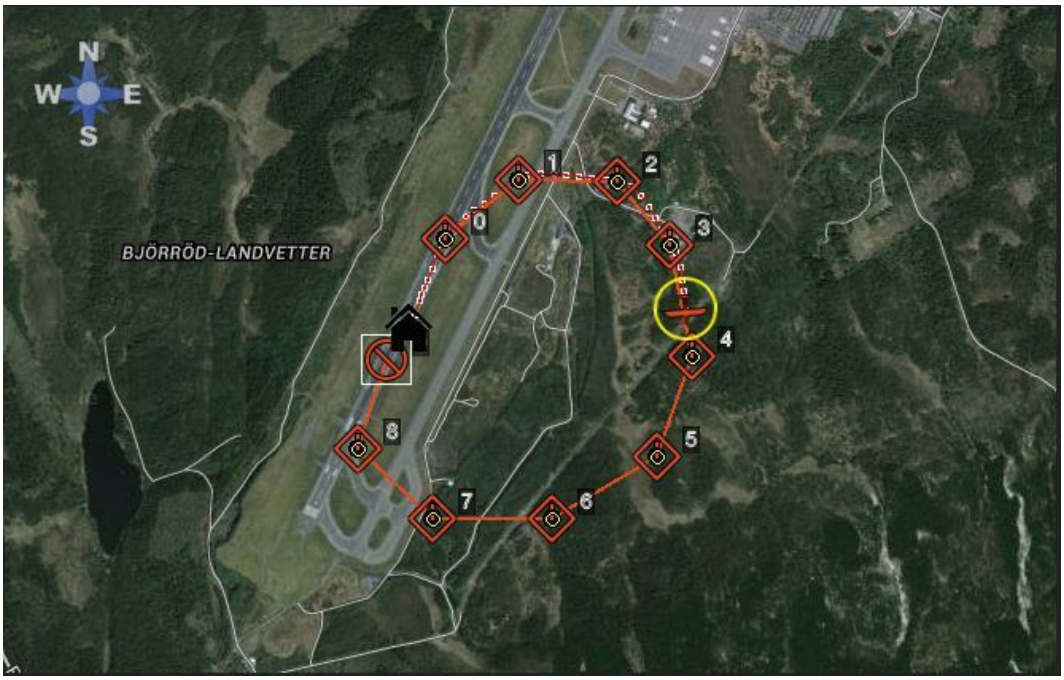



**Figure 8.1:** Simple route and flight path at the middle of an autonomous mission.

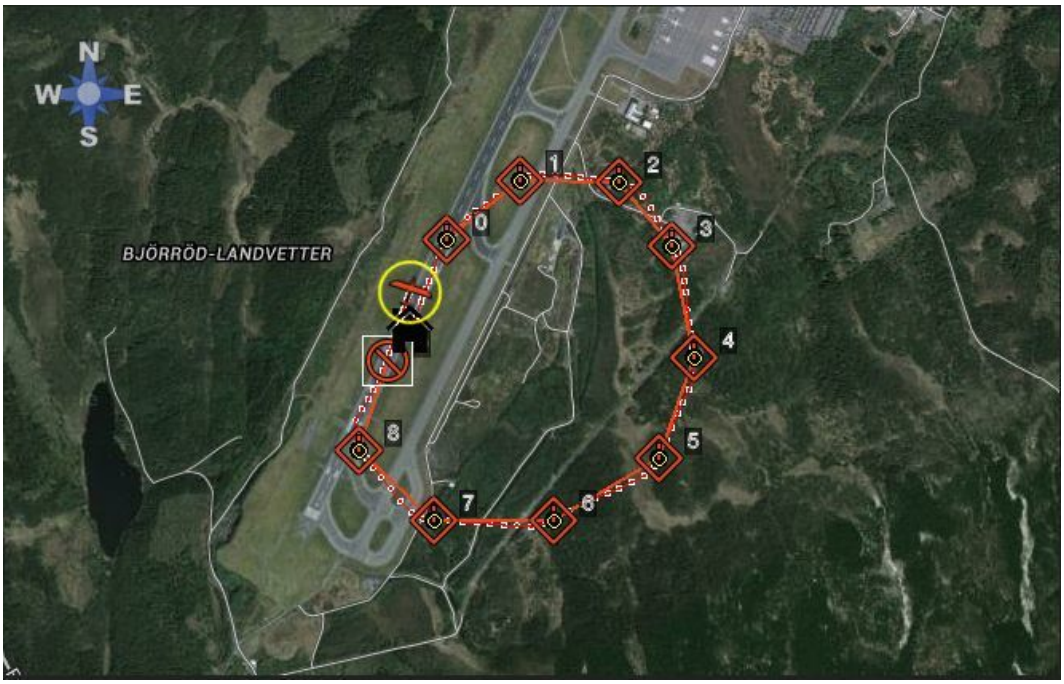



**Figure 8.2:** Simple route and path at the end of the autonomous mission.

Figure 8.1 and 8.2 presents the flight path of the aircraft whilst performing the simple route. As seen, the plane follows the reference trajectory and completes a full mission from launch to landing. Although, the landing is not necessary in the real system since the plane is intended to land by parachute deployment. This test verifies that the model is valid and that the controller is able to handle smooth reference trajectories.

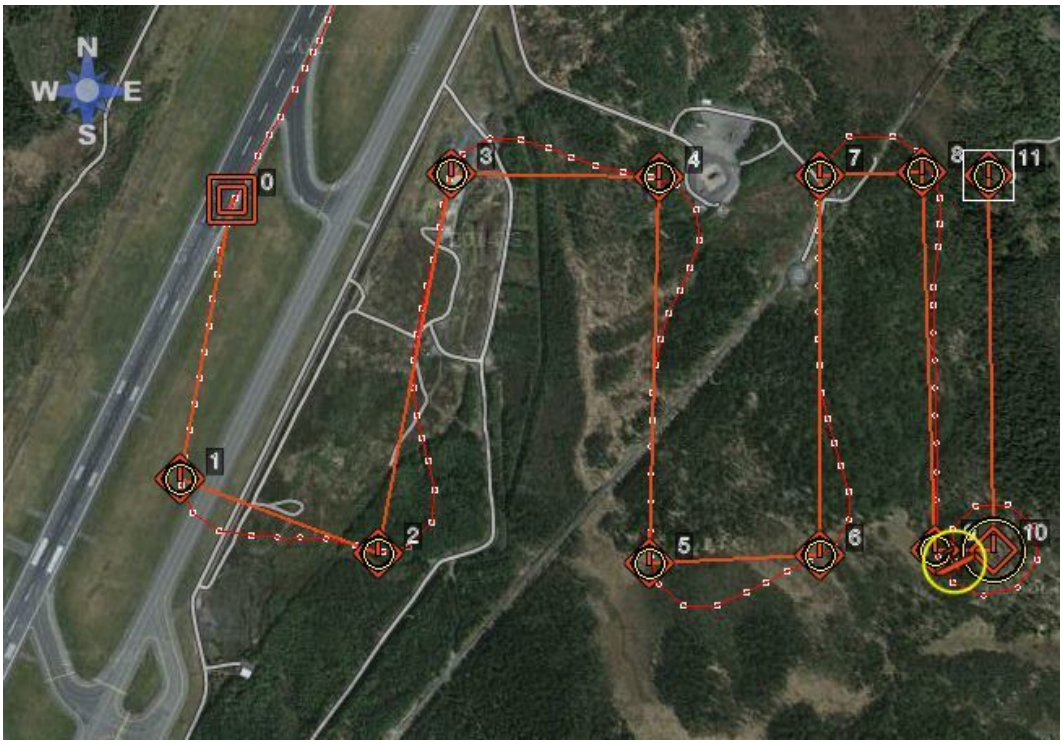



**Figure 8.3:** The slightly more complex route and the flight as shown in the simulation

A slightly more complex route is shown in Figure 8.3 together with the flight path for the aircraft. The purpose of this trajectory is to test a search pattern for scanning an area. The turns become sharper as the plane flies in order to investigate for which distance it fails to reach a way-point. In the figure, way-point 10 is never reached. The small distance between way-point 9 and 10 is difficult for the plane to handle, consequently causes the plane to circulate way-point 10 but never reaching it. The gap between way-point 9 and 10 is here approximately 85 meters. The turning radius of the plane is less than 85 meters, but the Radius of Acceptance (RoA) induces an earlier turn which causes the aircraft to miss the way-point. The RoA is the region around a waypoint where the plane is considered to have reached the waypoint successfully.

### 8.1.2 Test - Check Loitering altitude

This test is performed to check the plane's ability to loiter and at what altitude it needs to have for the camera to be able to focus at a scenario. The altitude calculations are

based on the plane's minimum loiter radius according to the equation in section 7.4. Figure 8.4 shows the loitering path of the aircraft when striving to follow a circular path with a 50 meter radius.

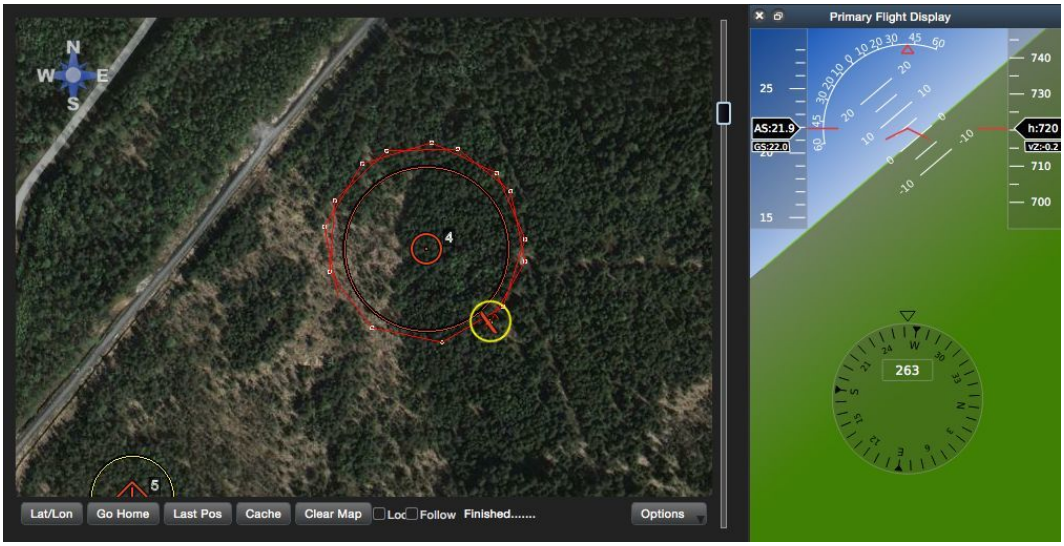



**Figure 8.4:** Loitering around a way-point with radius of 50m

As seen in figure 8.4, the plane almost manages to follow the circular path. It is estimated to deviate 10 meters from the reference path, where it keeps a constant roll angle of 40 degrees. For a circular reference with smaller radius, the roll angle still stays at approximately 40 degrees, shown in figure 8.5. Hence, the plane is not capable of turning more aggressively with the current controller tuning. Using a circular reference with 60 meter radius gives a flight path as displayed in figure 8.6, which shows an accurate path following behavior.

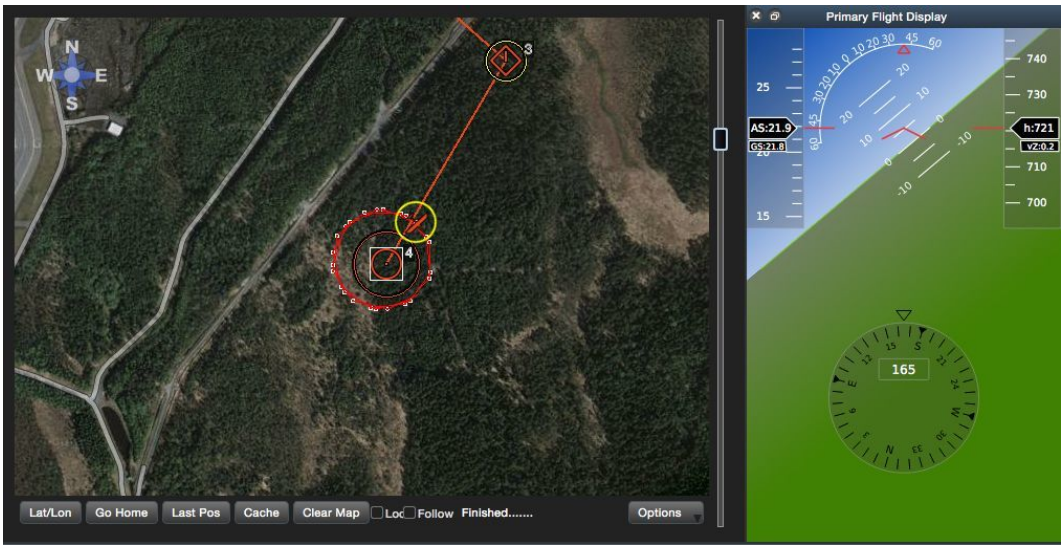



**Figure 8.5:** Loitering with radius of 40m


**Figure 8.6:** Loitering with radius of 60m

From chapter 7.4 it is depicted how the appropriate altitude is calculated based on the loiter radius and roll angle. It is beneficial to have a lower altitude since the quality of the footage increases at closer range. Using equation 7.3, repeated below, gives an expression for the minimum altitude to ensure a visible LOS.

$$altitude > radius \cdot tan(90 - \phi) \tag{7.3}$$

Inserting the minimum loiter radius of 60 meters with a roll angle ($\phi$) of 40° gives a minimum altitude of approximately 72 meters. Hence, the reference altitude for the real implementation is set to 75 meters to have some margin.

## 8.2 Real time Results

After successfully completing the flights in the simulation environment, the physical UAV is tested. The tests involve investigating the functionality of the controller, the camera system, the communication link and airframe design.

### 8.2.1 Aircraft Model

The constructed aircraft have the same dimensions and weight as the virtual plane. The CG is located 30 cm from the nose which with the modified fuselage coincides with the 28 cm suggestion from the Zephyr XL manufacturers. The wingspan of the complete aircraft is 2.2 meters and the total weight is 4.7 kg. The styrofoam airframe is covered in thin kevlar to enhance the robustness and mitigate damage in potential impacts. Furthermore, all the electrical devices and connections are waterproofed to protect the system during water landings. The two main computers of the aircraft, the autopilot and the Odroid computer is placed in a waterproof box in the center of the fuselage. Moreover, the camera mount for controlling the retraction of the camera, seen in figure 6.4, is realized using a 3D printer. The parachute is located at the back of the plane and is deployed using a spring when the servo releases the lock. A closer look at the fuselage is shown in figure 8.7.

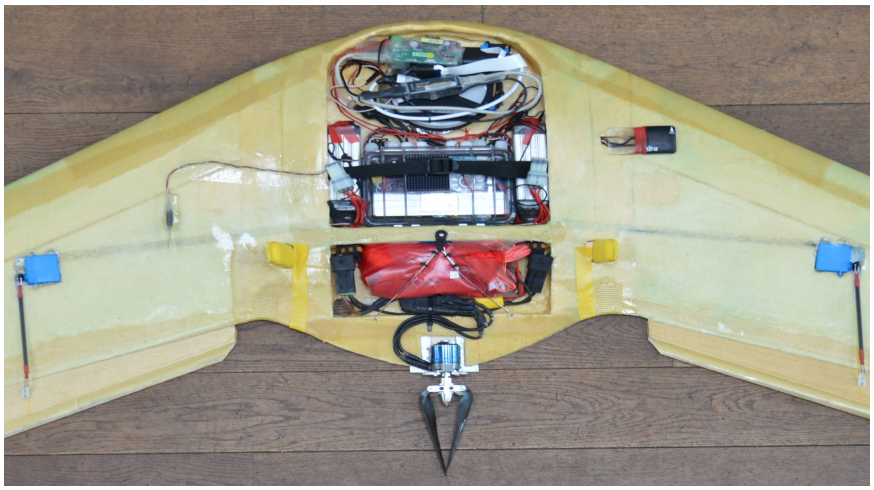


**Figure 8.7:** The fuselage of the UAV with the camera, the parachute and the waterproof box with the Odroid XU3 and Pixhawk inside

### 8.2.2 Flight tests

The flight tests are conducted on a large field located close to the sea in Långedrag, Gothenburg. The weather conditions are mostly sunny with wind speeds ranging from 5 m/s up to 8 m/s.

### Test - Hand Launch

For the initial testing of an easy autonomous mission the autopilot did not manage to stabilize the plane at launch. By attempting to launch the plane by hand the aircraft instantly crashed, not being able to gain enough speed or altitude. With a *hand launch* the plane is set to a autonomous mission and instantly thrown when the system initiates the launch. Several unsuccessful tests proved that the hand launching option is not applicable for this specific design. As a countermeasure to this problem, a construction of a catapult (see appendix A.4) was initiated. The purpose of the catapult is to ensure that the plane remains leveled during deployment. Moreover, the aircraft gains a higher initial speed using a catapult. Higher speed is useful for gaining altitude and control quickly. The launching device is shown in figure 8.8 together with the plane.

**Figure 8.8:** The UAV as it seats on the catapult in order the launching procedure to be initiated

### Test - Catapult Launch

Another simple autonomous mission was initiated as for the hand launch case, however, by the use of the designed catapult. The aircraft is placed on the catapult as in Figure 8.8. The autopilot is set to launch the autonomous mission when it senses the acceleration created by the catapult. Using this approach, several tests gave a similar behavior from the system. The aircraft does a stable launch from the catapult shown in Figure8.10, but shortly after turns violently which causes it to crash seen in Figure 8.9. Figure 8.11 shows the flight log from one of the tests that resulted in a crash. The plane does a left turn seen in Figure 8.11 from the negative roll value (green line), despite the controllers attempt to compensate (black line). The controller is not responsive enough towards small roll deviations, which suggests that the proportional gain of the controller is insufficient to counteract the initial roll. Furthermore, higher speeds are advantageous for rapid control, which makes the take-off a demanding stage for the controller.

**Figure 8.9:** The aircraft just after a balanced launch.

**Figure 8.10:** The aircraft turning to the left before crashing.

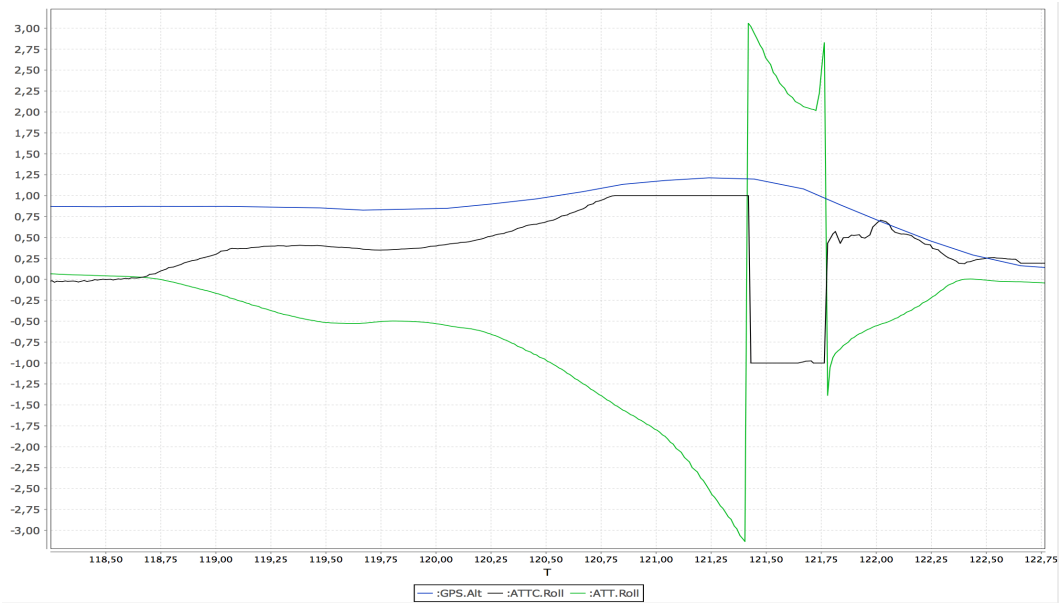



**Figure 8.11:** Shows the altitude, attitude roll and controller roll values from the plane from launch to crash. The plane is launched at T=118 seconds and meets the ground at T=122,75 seconds. The green line is the roll in radians, the black line is the control value for the roll (value between -1 and 1) and the blue line corresponds to the altitude in meters. At approximately T=122,4 s the plane flips on its back mid-air, reversing the roll values.

Although the controller reacts mildly to roll deviations, it does behave correctly. The pitch is controlled separately by its own PID controller and is shown in figure 8.12. The plot does not reveal any controller errors and the pitch is kept relatively stable at approximately 0.4 rad ($\approx 23°$) at T=119,50-120,5. After this interval the large roll angle sets the pitch in reverse and the plane crashes.

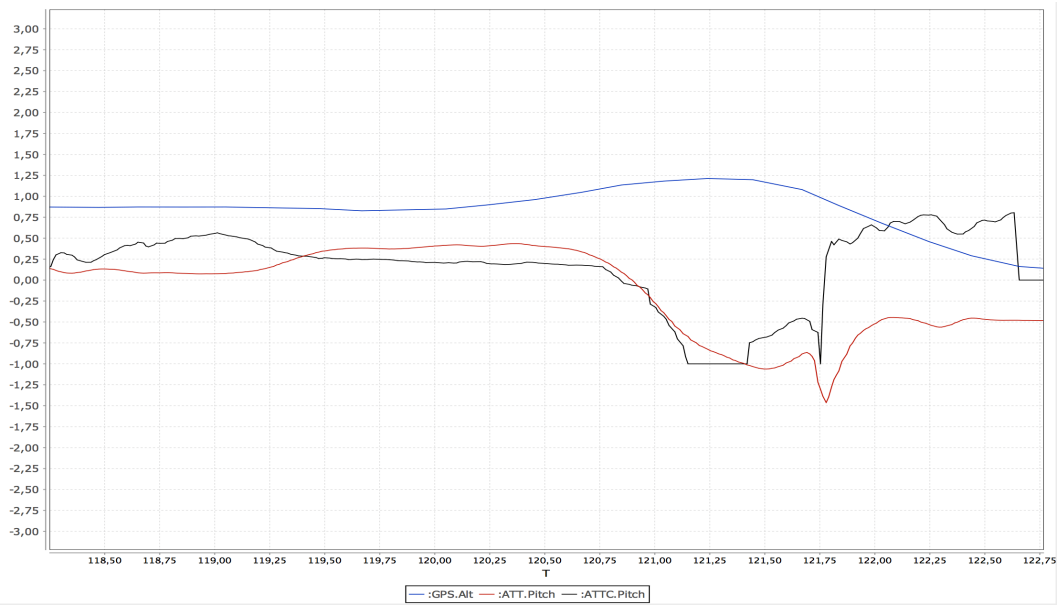



**Figure 8.12:** Shows the altitude, attitude pitch and controller pitch values from the plane from launch to crash. The plane is launched at T=118 seconds and meets the ground at T=122,75 seconds. The red line is the pitch in radians, the black line is the control value for the pitch (value between -1 and 1) and the blue line corresponds to the altitude in meters. At approximately T=122,4 s the plane flips on its back mid-air, reversing the pitch values.

### Test - Manual Flying

To troubleshoot what is preventing the plane from take-off, the plane is controlled manually using the radio controller. Pronounced instabilities in the airframe can make it difficult for the autopilot to balance the plane. By controlling the aircraft manually it is possible to check if the plane is capable of flight. Plot 8.13 shows the flight log from a 10 second manual flight, which ended in a crash. Although difficult to control as seen from the unsteady attitude in 8.13, the plane is able to fly. The test was conducted by an inexperienced pilot, hence the spiky roll and pitch values. In figure 8.14 the first three seconds of the roll values in manual flight is shown to be compared with the autonomous in figure 8.11. As seen, the commanded roll values when flying manually is much larger and faster than for the commanded roll values when the autopilot is in control. The manual control kept the aircraft airborne for a significantly longer time, which could suggest that the controller needs to have higher gains.

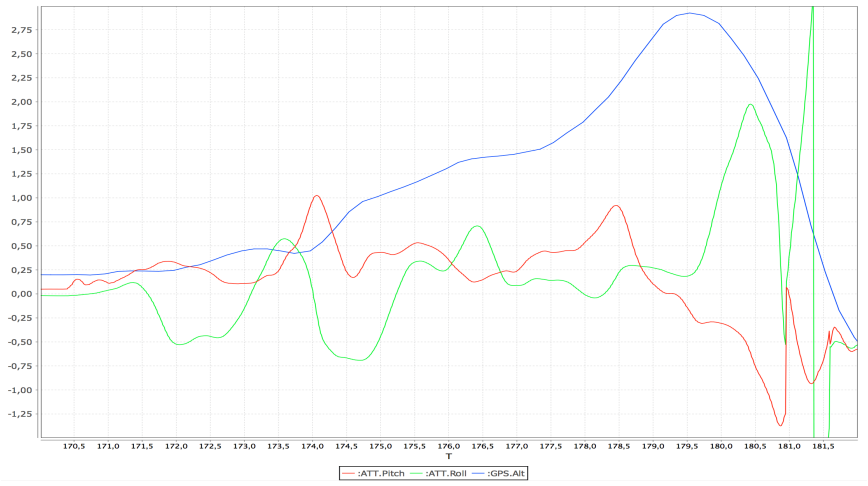



**Figure 8.13:** Plot showing the roll, pitch and altitude of the plane during a 10 second manual flight. The plane lifts off at T=170,5 seconds and meets the ground at T=181,5 seconds. The blue line corresponds to the altitude in meters scaled by 0.1. The red and green line is the pitch and roll respectively in radians.

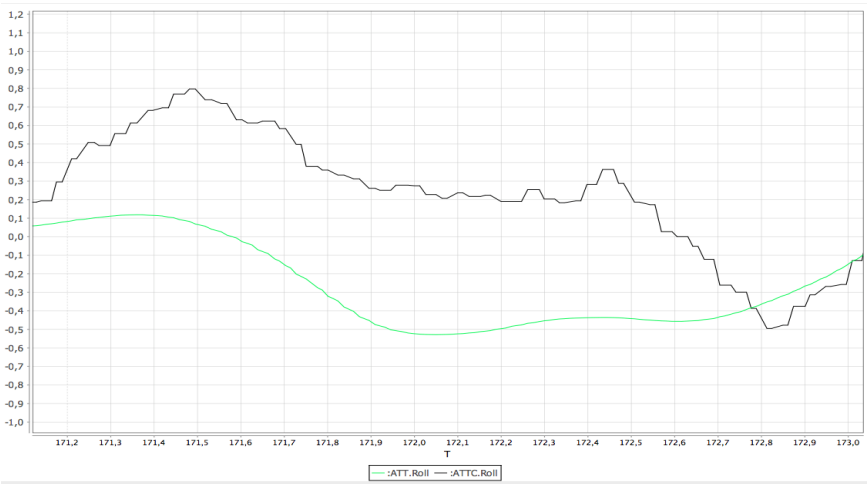



**Figure 8.14:** Plot showing the attitude roll and commanded roll of the plane during the first three seconds of the manual flight in figure 8.13. The black line corresponds to the commanded roll by the pilot and the green line is the attitude roll.

### Test - Manual launch with autonomous flight

A different approach to test the function of the aircraft is to control the plane manually during launch and switch to the autopilot mid-air. This method bypasses the demanding launch for the autopilot and tests if it is able to stabilize the plane at cruise speed and higher altitude. The test showed an improved response from the autopilot and the plane

flew autonomously in approximately 50 seconds. During this time it approached its waypoint, but was not stable enough to reach it successfully. The aircraft went into a stall which eventually caused a fatal ground impact that destroyed the airframe. The reasons behind the stall are yet undefined since the flight log got damaged from the crash and the communication disrupted. The results are discussed in section 9.1.

### 8.2.3 Communication link

During off-board testing of the communication link the system behaved as intended. The communication between the autopilot and the ground station was fast and continuous. The operator had the complete control over the UAV and the initiation of the automatic missions. However, when the plane was tested in the field under real flight conditions the communication link was unreliable and disconnected randomly. Moreover, the connection often got disrupted at launch and the ground station did not regain the connection whilst in the air. These issues were not resolved until after the fatal crash, which made it impossible to fully test it in flight. The random connection losses emerged from bad shielding of the modem's USB cable. The customized cable caused the modem to be susceptible to interference, resulting in random disconnections to the mobile network. Furthermore, the connection losses at launch were a result of a loose fuse connected in series with the cable supplying the Odroid with power. The acceleration form the launch sometimes caused a disruption in the power supply, inducing a restart of the Odroid and thus loosing the connection to the internet. Correcting these hardware errors made the communication reliable and robust against impacts, but was never tested in a real flight.

### 8.2.4 Live footage Streaming

Already from the testing on the bench the camera system operated as expected. A stable communication over the mobile cellular network allowed the operator on the ground to receive live footage and interact with the camera over distance. However, the camera system suffers delays in both receiving footage and executing commands. From the beginning the whole camera together with the camera mount module was placed in the plane and in order to evaluate the whole operation of the complete system. However, the first crashes appeared to be risky for the safety of the camera and its mount module. The impact and the force generated from these crashes resulted in a broken camera mount. Thus, due to the fragile components the camera was removed from the plane to ensure the safety of the system. In the place of the camera a dummy weight was used in order to represent the actual impact that the camera has on the weight and center of gravity of the aircraft.

# 9

# Discussion

In this section the different choices, results and possible improvements are discussed. Furthermore, issues that have been met are presented and also how these have been resolved.

## 9.1 Flight analysis

For testing how the plane operates the only option is to launch it and observe the results. This approach is problematic since there are a lot of functions that are tested simultaneously, i.e. there are several factors that might not be working properly. Hence, a plane that is not able to fly is difficult to troubleshoot. Also, if it is able to fly but is unstable, there is a risk of damaging the plane and even people. The conducted tests have mainly been focused around the plane's ability to fly. The results show that the aircraft is unable to take-off in autonomous mode, but can stay airborne for a shorter time when launched manually and then switched to autonomous mode. The major factors that affects the flight performance are the airframe, the controller and the weather conditions. The inability to fly can be a result of any of these, or a combination of them.

### 9.1.1 Simulation and controller tuning

HILS is the only proof obtained that shows that the autopilot is working correctly, but might not be optimally tuned. In the simulation environment a virtual model is used that have the same or similar attributes to the physical plane. The size, weight, balance and center of gravity is the same or very close to the physical plane. However, there are approximations due to limitations in the simulation software. These are although expected and the achieved tuning from the simulation is not assumed to be optimal, but at least be able to keep the plane airborne when testing. Evidently, the controller could

not stabilize the physical plane. It did however operate the virtual plane with precision and speed in simulations. Some of the possible causes are presented below:

- The approximations of the virtual model and physics in the simulation environment were too large and not similar to the physical plane.
- The simulated weather conditions in the simulation were not as harsh as the weather conditions during physical testing, hence, the controller is not tuned to be robust against heavy disturbances.

One approximation in the virtual model that could have caused deviating flight characteristics are the airfoils. In order for the simulation to calculate the correct forces acting on the plane it needs to know the shape of the wings, i.e. the airfoils. Since the airfoils are determining the lift, drag and other aerodynamics they are essential for designing an exact virtual model, explained in section 7.1.2. However, the exact airfoils for the Zephyr XL is not known and could not be obtained. Instead an approximate airfoil was chosen based on being similar to the cross-section of the Zephyr XL and also being recommended as an appropriate airfoil for smaller flying wings. If this approximation have made a major difference in flight characteristics is not obvious since the subject is complex and requires extensive testing and knowledge to answer.

The wind speeds during testing were not similar to the weather conditions within the simulation. The simulated weather was calmer, with winds speeds only reaching up to 3-4 m/s compared to the 5-8 m/s in testing. During controller tuning, there is a trade-off between having smoother/slower control or harsh/rapid control. For the initial tuning, smoother control seemed appropriate in order to avoid rapid turns and oscillating behavior. Additionally, since the plane is not designed for complex routes, the smoother/slower tuning was chosen as a starting point for testing. However, this approach together with moderate wind speeds can possibly have made the controller too stagnant for harsher weather conditions. This theory is supported by analyzing the roll controller response in figure 8.11 and figure 8.14 in section 8.2.2. It is clear that the commanded roll from the autonomous mode in 8.11 is not as aggressive as for the commanded roll given by a pilot in 8.14. Since the plane was capable of flight when manually controlled, it is possible that the controller needed higher gains in order to stabilize the plane. A better approach could have been to tune the controller in rough simulated weather conditions to make sure that it could handle higher wind speeds. Although the goal was to operate in calm weather, the higher wind speeds when testing is inevitable, which should have been accounted for.

### 9.1.2 Size, Weight and Power

Another factor contributing to the poor flight performance is the relation between the size, weight and power on the aircraft. The initial idea was to build a large aircraft capable of heavy payloads and handle intermediate weather conditions. The original Zephyr XL is large but did not have the necessary space to carry all the components

needed, which is why significant modifications were made to the fuselage. Hence, the size of the aircraft was increased and the new airframe was created. In doing so, more power might have been needed to push the plane, i.e. a more potent DC motor with a larger propeller. The used motor was on the verge of being too weak to push the Zephyr XL, hence, the extra fuselage made the motor unsuitable for such a large wing. Additionally, in order to handle intermediate wind conditions, the motor needs to be powerful enough to push the plane through headwind as well.

The weight of the aircraft is also a factor when considering the choice of motor/propeller combination. The total weight of the aircraft was 4.7 kg, far too heavy for the available thrust. The desire to make the plane robust and waterproof resulted in a lot of extra weight. The waterproof connections, kevlar shielding, box and isolation added more weight than initially expected. Moreover, the attached camera system had a total weight of 590 grams, which is relatively heavy compared to other viable camera options. The plane could have been built a lot lighter by sacrificing some of the less important functions as waterproofing, kevlar, parts of the camera system and even the parachute. Another approach would be to attach a powerful motor capable of pushing such a heavy plane. Although, this was not regarded as an issue since the plane flew in simulation with the same weight, size and motor. That was however under calm weather conditions, perfect motor efficiency and a perfectly balanced airframe.

### 9.1.3 Center of Gravity

The balance is one of the most important aspects on the overall flight performance, especially for a flying wing. It is connected and influenced by the size, the weight and the placement of the components on the aircraft. The CG of the modified Zephyr XL was placed according to the measurements defined for the Zephyr XL. Although, slight modifications were made in order to adapt to the modified airframe. The poor flight performance shown in the result was likely influenced by a misplaced CG. The changes made to the fuselage could have caused a different lift and drag compared to the original model, shifting the in flight CG. Based on observations from the flights, the plane did display a behavior that suggests that it was tail-heavy. Tail-heavy flying wings are either highly unstable or not capable of flight, especially near ground [26]. They are likely to stall at launch and at slow speeds. This could be the reason that prevented autonomous take-off, but allowed unstable flight after it gained speed and altitude. To ensure a correct CG, extensive calculations and testing is needed. Alternatively, the original airframe should be kept and the recommended CG could be used.

## 9.2 Hardware/Software Selection

A major part of the project consisted of choosing and combining both hardware and software into a full system. Every choice is constrained by several factors such as size, weight, performance, price, compatibility, complexity etc. The result of some of the

choices are presented and discussed in this section.

### 9.2.1 Autopilot

As described in section 3, two autopilots were considered for this project. The software comparison was mainly focused between the PX4 and Ardupilot, being the more popular autopilots. Moreover, the viable hardware alternatives was the APM board and Pixhawk. The different softwares are optimized or only compatible with certain hardware, which created restrictions in combining hardware and software. The PX4 flight stack can only run on a PX4 middleware, found in the Pixhawk board. However, Ardupilot is implementable on both the APM board and the Pixhawk, but does not have all the functions when running on a Pixhawk. The Pixhawk/PX4 combination was chosen mainly because it is a newer and faster autopilot. The Ardupilot is older and runs on a 8-bit architecture, i.e. it does not have much room for improvement regarding speed and additional functions. The Pixhawk runs a 32-bit architecture and is therefore significantly faster, but is constantly being developed and is not as thoroughly tested as the Ardupilot. It was preferable to choose an autopilot that could be improved, but as it is still in development, the Pixhawk has been behaving erratically since the beginning. Missing features, bugs and incompatibility issues are encountered frequently.

The APM board was tested as well, as an alternative to compare the performance and robustness. The APM board displayed hardware malfunction at early stages, where the sensors had bad health which prevented the autopilot from operating correctly. The fragile hardware components, or product defect, was sufficient to discard the APM board as an alternative autopilot. However, the Pixhawk's I/O board failed mid-way into the project. It was switched for a clone, Fixhawk, that uses the same software but supposedly have better hardware quality. This autopilot has been used for the majority of the project and have been fully operational throughout.

An issue with the Pixhawk/PX4 and APM/Ardupilot is the lack of reliability and robustness. When developing a cheaper UAV, this is a problem that is difficult to bypass. Considering the safety aspects of this, UAVs that implement these autopilots are not suited for commercial use.

### 9.2.2 Camera - Live footage Streaming

The main discussion for the camera system revolves around what type that would give stable footage, a wide viewing angle and that would be small enough to be integrated into the UAV. The considered alternatives were a small camera with a gimbal mount, a dome camera or just a static camera with a wide angle of view. The static cameras soon was disregarded because of the inability to keep focus during turns. Even though their small size makes for an attractive solution for UAVs, the limitation on the angle of view is an vital attribute that could not be neglected. Hence, the dome camera and the gimbal solution were further investigated. A fixed camera with a gimbal mount is

preferable for achieving high quality footage. Some of the available cameras for this application are small and capable of High Definition (HD) live streaming. Together with the gimbal mount, the system is able to minimize disturbances, fully rotate and fix the camera's focus. The issue with the approach is the lack of space needed for such an application. The required space for the gimbal to be able to use its full movement is simply not available in the fuselage of the plane. There exists ball gimbals with the same features that are compact, perfect for flying wing applications. However, these are out of the price range for this of project. The alternative were to place a gimbal mount on the outside of the airframe, attached to the belly of the plane. The approach could be possible and effective at lower speeds, but the plane's aerodynamics would decline. Furthermore, a gimbal mount attached to the belly of the aircraft would be exposed and most likely take damage when landing.

The specific choice of the dome camera VB-S31D is a solution that combine both the advantages of a gimbal rotation and a static camera. It has pan tilt and zoom features without the need for additional space during rotation. Moreover, the control over the Internet is a fitting feature that set the camera as the better option amongst the alternatives. A wide range of view and the possibility of creating a system that could retract the camera in case of crashing/landing added value to the specific solution. The camera does however lack the stabilization and fixed focus capabilities of an actual gimbal mount. If the plane was to be used commercially, these gimbal features would be needed. Also, software that allowed target lock would ease the camera control significantly. The VB-S31D is although sufficient for the first build, as a proof of concept.

The camera system had an easy implementation and did not require a lot of work until a connection over internet was established. However, there were significant delays in the communication between the camera and the GUI. With full signal strength on the wireless connection, the stream had a delay at approximately 3-4 seconds. The delay was evident in both receiving footage and sending control commands to the camera. Hence, receiving footage and sending control commands based on the information would be problematic. The visual feedback would not be obtained fast enough to guide the plane or the camera, making it difficult to control the focus.

### 9.2.3 Communication over the mobile network

Communicating with the autopilot through the mobile network has the advantage of removing the distance constraint, that is present when using radio as transmission medium. However, communication over internet is not as reliable, safe or fast as using radio. Also, additional components and work is needed to establish a link between a ground station and the autopilot. The approach for this project is, as described in chapter 5, by using a companion computer (Odroid XU3). Since the computer is essentially only used as a router, its function could probably be replaced by something smaller as an SIM900 mentioned in 5.2. As the SIM900 consists of a simple circuit board it would be beneficial in regard to size, weight and power consumption. However, initially the Odroid was

intended to be used for more than just forwarding messages to the autopilot. It was implemented to be the only link to the mobile network, connecting both the camera and the autopilot to the internet. Additionally, it could serve as a platform for programming additional functions in the aircraft, such as gimbal control software and target locking. If the UAV was to operate successfully, it could be subject to further projects by adding functions and improvements into the Odroid. But by choosing a camera with its own control software the need for gimbal control was removed. Also, by striving for minimizing the delays between in the communication, the autopilot and camera system were given separate modems.

With a strong internet signal (4G), the internet communication displayed similar response time and reliability as when using radio for transmitting data. As mentioned in the results, chapter 8.2.3, the communication link was never fully tested in the air due to defect circuitry. When the issues were found and resolved, the airframe was already destroyed.

## 9.3 Moral and Ethics

While unmanned aerial vehicles are being deployed and explored at an increasing pace, the moral dilemma and debate of their use has grown. There are a variety of situations where UAVs are useful. As previously stated; their application within military, civilian and industrial areas are being explored and is often regarded as an effective solution. Consequently, as with most new technology, the ethics and morals in their establishment becomes a relevant topic of discussion. The use of UAVs Beyond-Visual-Line-of-Sight (BLOS) are currently illegal in several countries, Sweden being one of these. The reason behind these laws will be highlighted and also the benefits of removing them.

### 9.3.1 Safety

The obvious downside with regard to safety for UAVs is that it introduces a risk of falling out of the sky and hurting people, animals or property. Almost any malfunction or other unanticipated situation will result in a crash for a UAV. If a drone stop functioning correctly it will, unlike cars or other vehicles, gain speed instead of slowing down. It is also difficult to see and have time to react to an object falling from above, since it is never expected or within a humans visual line of sight. Moreover, since an UAV does not have an pilot onboard or no pilot at all, it can be difficult to execute an evasive action if an unexpected object presents itself.

Besides the safety of people on the ground, there is also the decreased safety for pilots in manned aircrafts sharing the same airspace as drones. A collision risk is introduced between manned and unmanned vehicles, which can pose significant danger to the pilots and other passengers. A drone is often small and can be difficult for pilots to spot. Even though UAVs would incorporate transponders to be easily detected and have a sophis-

ticated avoidance system, other pilots would not be able to communicate with these if necessary.

However, for the pilots controlling these drones or have set an autonomous mission the safety is drastically increased and even ensured. Using an UAV poses no threat at all to the one controlling it. It can be used for otherwise dangerous reconnaissance missions in a hazardous or unfriendly environment without risk for the pilot. Also, their ability to be deployed instantly have numerous uses for search and rescue missions. Information about a scene can be obtained quickly and be used for critical decisions.

### 9.3.2 Security and Privacy

Since drones are small and airborne they do not have the same constraints as a car. Other vehicles can usually be kept out of restricted areas by placing a fence around a property, this will have no effect on drones. Private areas or property are highly accessible and footage of possibly sensitive information can more easily be obtained. Moreover, since there is no pilot present, misconduct using an UAV does not involve the same risk for the one controlling it. Entering restricted areas is easier and does not involve the same risk as other methods.

On the positive side, the surveillance capabilities of drones can also significantly increase security. By using drones to monitor a premise or an event, the awareness in that area would be heightened. Drones could even be used in large numbers as an surveillance fleet, communicating and together having visual coverage over vast areas. Besides acting as moving security cameras, they also have the advantage of being airborne which gives a clear view without obstacles.

With these survey capabilities, drones can also be regarded as intrusive and as an violation of privacy. If drones would become a common occurrence even amongst private persons, they could be regarded as surveillance cameras constantly roaming the sky. Also, people could be monitored relatively easy from high altitudes without alerting them.

### 9.3.3 Environmental aspects

There are tasks where UAVs can replace full-sized manned aerial vehicles. They are most prominent in surveying, but can be used for delivery and agriculture. UAVs that are using electric motors are significantly superior with regard to emissions and noise. Tasks that need airborne support that does not involve heavy payloads, could be carried out by employing drones. Moreover, with their low-cost and low-impact, drones could be deployed more often and in larger numbers. Consequently being able to do large-scale environmental monitoring and mapping.

# Bibliography


[1] (2015) Vi raddar liv till sjoss. The Swedish Sea Rescue Society. [Online]. Available: http://www.sjoraddning.se/detta-gor-vi/

[2] UAS. (2015) The uas europe. [Online]. Available: http://www.uas-europe.se/

[3] DroneDeploy. (2015) The dronedeploy. [Online]. Available: https://www.dronedeploy.com/

[4] (2015) The botlink. Botlink. [Online]. Available: http://botlink.io/

[5] SkyDrone. (2015) The skydrone. [Online]. Available: http://www.skydrone.aero/

[6] K. J. Aström and R. M. Murray, "Feedback systems an introduction for scientists and engineers," vol. 2.10b, pp. 190–220, February 2009.

[7] Y. Kang and J. Hedrick, "Linear tracking for a fixed-wing uav using nonlinear model predictive control," *Control Systems Technology, IEEE Transactions on*, vol. 17, no. 5, pp. 1202–1210, Sept 2009.

[8] G. Chowdhary, T. Wut, M. Cutler, N. Kemal, and J. P. How, "Experimental results of concurrent learning adaptive," *American Institute of Aeronautics and Astronautics*, 2012.

[9] B. Kada and Y. Ghazzawi, "Robust pid controller design for an uav flight control system," in *Proceedings of the World Congress on Engineering and Computer Science 2011 Vol II*, Oct 2011, pp. 945–950.

[10] H. Grankvist, "Autopilot design and path planning for an uav," Dec 2006.

[11] I. H. Johansen, "Autopilot design for unmanned aerial vehicles," in *Master Thesis at Norwegian University of Science and Technology ,Department of Engineering Cybernetics*, June 2012.

[12] S. T. Agency, "The swedish transport agencys regulations on unmanned aircraft systems (uas)," October 2009.

[13] RiteWing. (2015) The zephyr xl. [Online]. Available: http://www.ritewingrc.com/products.html

[14] P. D. Team. (2015) Px4 autopilot. [Online]. Available: http://www.pixhawk.org/

[15] N. Lwin and H. M. Tun, "Implementation of flight control system based on kalman and pid controller for uav," *Journal of Scientific and Technology Research*, vol. 3, April 2014.

[16] J. B. et al. (2015) Pid tuning classical. [Online]. Available: https://controls.engin.umich.edu/wiki/index.php/PIDTuningClassical#Ziegler-Nichols_Method

[17] J. D. Sanghyuk Park and J. P. How, "A new nonlinear guidance logic for trajectory tracking," *American Institute of Aeronautics and Astronautics*, june 2004.

[18] P. D. Team. (2015) Px4 autopilot. [Online]. Available: https://pixhawk.org/firmware/parameters

[19] L. M. et al. (2015) Qgroundcontrol. [Online]. Available: http://www.qgroundcontrol.org/

[20] A. Tridgell. (2015) Mavproxy. [Online]. Available: https://tridge.github.io/MAVProxy/

[21] L. Meier. (2015) Mavlink. [Online]. Available: http://qgroundcontrol.org/mavlink/start

[22] L. M. et al. (2015) Px4 autopilot. [Online]. Available: https://pixhawk.org/dev/companion_link

[23] M. Hepperle. (2008) Aerodynamics of a model aircraft. [Online]. Available: http://www.mh-aerotools.de/airfoils/

[24] RiteWing. (2015) The center of gravity specification. [Online]. Available: http://www.readymaderc.com/store/index.php?main_page=product_info&cPath=11_22&products_id=1235

[25] K. J. Aström and R. M. Murray, "Feedback systems an introduction for scientists and engineers," vol. 2.10b, pp. 302–313, February 2009.

[26] K. Nickel. (1997, July) On the importance of the correct c.g. location in flying wings. [Online]. Available: http://www.nurflugel.com/Nurflugel/Papers/Dr_Nickel_Paper/body_dr_nickel_paper.html

# A

# Coding/Configuration

## A.1 Pixhawk Tuning

The rest of the parameters completing the full list kept with their default values and they refer to the following :

| | |
|---|---|
| FW_ATT_TC | 0.5 |
| FW_FLARE_PMAX | 15 |
| FW_FLARE_PMIN | 2.5 |
| FW_L1_DAMPING | 0.75 |
| FW_L1_PERIOD | 25 |
| FW_LND_HHDIST | 15 |
| FW_LND_HVIRT | 10 |
| FW_MAN_P_MAX | 45 |
| FW_MAN_R_MAX | 45 |
| FW_PR_IMAX | 0.2 |
| FW_P_RMAX_NEG | 0 |
| FW_P_RMAX_POS | 0 |
| FW_RR_IMAX | 0.2 |
| FW_RSP_OFF | 0 |
| FW_R_LIM | 45 |
| FW_R_RMAX | 0 |
| FW_THR_CRUISE | 0.7 |
| FW_THR_LND_MAX | 1 |
| FW_THR_MAX | 1 |
| FW_THR_MIN | 0 |
| FW_THR_SLEW_MAX | 0 |
| FW_CLMBOUT_DIFF | 25 |
| FW_T_HGT_OMEGA | 3 |

| | |
|---|---|
| FW_T_HRATE_FF | 0 |
| FW_T_INTEG_GAIN | 0.1 |
| FW_T_PTCH_DAMP | 0.1 |
| FW_T_SINK_MAX | 2 |
| FW_T_SINK_MIN | 2 |
| FW_T_SPDWEIGHT | 1 |
| FW_T_SPD_OMEGA | 2 |
| FW_T_SRATE_P | 0.05 |
| FW_T_THRO_CONST | 8 |
| FW_T_THR_DAMP | 0.5 |
| FW_T_TIME_CONST | 5 |
| FW_T_VERT_ACC | 7 |
| FW_YCO_METHOD | 0 |
| FW_YCO_VMIN | 1000 |
| FW_YR_FF | 0.3 |
| FW_YR_I | 0 |
| FW_YR_IMAX | 0.2 |
| FW_YR_P | 0.05 |
| FW_Y_RMAX | 0 |
| FW_LND_ANG | 10 |
| FW_LND_FLALT | 10 |
| FW_LND_TLALT | 5 |
| FW_LND_USETER | 1 |
| FW_T_HRATE_P | 0.1 |

## A.2 Fail-safe Configuration

- **Loiter time for radio control loss : NAV_RCL_LT = -1**
  The amount of time that the plane should loiter in the specific position before triggering flight termination. If set to -1 the autopilot skips the Return to Launch process and enters immediately in flight termination.

- **Data link loss mode enabled : COM_DL_LOSS_EN = 0**
  Set to 1 or 0 in order to enable or disable action on Data link loss.

- **OBC mode for rc loss : NAV_RCL_OBC = 1**
  If set to 1 the behaviour on data link loss is set to a mode according to the OBC rules.

- **Circuit breaker for engine failure detection : CBRK_ENGINEFAIL = 0**
  Setting this parameter to 284953 will disable the engine failure detection.

- **Circuit breaker for flight termination : CBRK_FLIGHTTERM = 0**
  Setting this parameter to 121212 will disable the flight termination action.The IO driver will not do flight termination if requested by the FMU
- **Circuit breaker for GPS failure detection : CBRK_GPSFAIL = 0**
  Setting this parameter to 240024 will disable the GPS failure detection.
- **Circuit breaker for IO safety : CBRK_IO_SAFETY = 0**
  Setting this parameter to 894281 will disable input-output (IO) safety.

## A.3 MAVproxy Commands

The MAVproxy set of commands is used for establishing the link between the ground station and the aerial vehicle.The only mandatory command necessary to initiate MAVproxy is the −−master .The list of commands that are used can be seen below.

- −−master
  Defines the port that will be used for communicating witht he UAV.IT can also be used for defining the IP address if the communication is over internet.
  - mavproxy.py –master=/dev/ttyUSB0 ⇒ connected to a USB port
  - mavproxy.py –master="com14" ⇒ connected to a COM port
  - mavproxy.py –master=192.168.1.1:14550 ⇒ connected to a IP address port
- −−baudrate
  sets the baudrate of –master and –out ports.
- −−out
  Forwards the MAVlink to a remote device.this device can be a USB port, a serial or network port. If a network is to be used the default port that is used form the MAVlink protocol is the 14550. The type of the connection can also be defined, i.e. udp or tcp. Examples of the usage of the *out* option are,
  - mavproxy.py –master=/dev/ttyUSB0 –out=udp:192.168.1.1:14550 ⇒ USB connection to UDP port over internet
  - mavproxy.py –master=/dev/ttyACM0,115200 –out=/dev/ttyUSB0,57600 ⇒ ACM port to USB port
  - mavproxy.py –master=/dev/ttyACM0,115200 –out=COM17,57600 ⇒ ACM to COM port
- −−aircraft
  Sets a name for the logfile. If used, logfiles will be stored in /Logs/AircraftName/-Date/flightNumber/flight.tlog.

- −−load-module
  Load the specified module on startup. Can be used multiple times, or with a comma separated list.

- −−dialect
  MAVLink dialect. Uses the APM dialect by default.

- −−console
  Load the GUI console module.

A full list of commands available for MAVproxy can be found at `http://tridge.github.io/MAVProxy/starting.html`.

## A.4 Catapult Design

The catapult consists of a traveling holder guided by a rail. The UAV is placed on the holder that is launched by bungee cords on the destination of the rail. The holder is pushed back to the extreme position (launching) and is being held by a safety pin until the aircraft is ready to be deployed. A stop mechanism is also included for keeping the holder from crashing to the other end of the rail. There are 2 sets of three bungee wires (in total 6) in each side of the guidance rail.The force generated by the cords is able to launch the plane 3 meters away from the ending point of the catapult.
The aircraft is placed on the catapult but with the a launch detection attribute has been activated. The launch detection allows the controller to detect the acceleration generated by the catapult in order to initiate the auto mission. While the plane is standing still the throttle is set at 30 percent. The moment that the autopilot detects the acceleration it will issue the initiation of the auto mission after a delay that has been set by the operator.
All the parameters needed for the catapult are presented below:

- **Launch detection : LAUN_ALL_ON = 1**
  Enables the launch detection for the catapult.

- **Catapult Acceleration threshold : LAUN_CAT_A = 10**
  The acceleration threshold detection for the catapult.

- **Catapult time threshold : LAUN_CAT_T = 0.05**
  The time threshold detection for the catapult.

- **Motor delay : LAUN_CAT_MDEL = 0.1** A delay on the motor to start after the initiation of the launch.

- **Throttle before detecting launch : LAUN_THR_PRE = 0.03** Sets a value for the throttle while the catapult is ready for launch.

## A.5 Flight Termination Widget

Qml file that is loaded in QGroundControl to enable fail-safe mode on button click. The button sends MAVlink command number 185, that corresponds to flight termination. The full MAVlink command list is found in the MAVlink documentation [21]

**Listing A.1:** Customized ground control widget that sends flight termination command on click.

```
import QtQuick 2.2

import QGroundControl.Controls 1.0
import QGroundControl.FactSystem 1.0
import QGroundControl.FactControls 1.0
import QGroundControl.Controllers 1.0

FactPanel {
    id: panel

    CustomCommandWidgetController { id: controller; factPanel: panel }

    Column {

        QGCButton {
            text: Flight Termination
            onClicked: controller.sendCommand(185, 50, 3, 1, 0, 0, 0, 0, 0, 0
        }

        FactTextField {
            fact: controller.getParameterFact(-1, "MAV_SYS_ID")
        }
    }

}
```