Design Specification
Autonomous Unmanned Aerial Vehicle

Version 1.2
Author: Henrik Ohlsson
Date: April 4, 2006

Status

<table>
<thead>
<tr>
<th>Reviewed</th>
<th>Kristina Jersenius, Mårten Hedberg</th>
<th>2006-03-03</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved</td>
<td>David Törnqvist</td>
<td>2006-03-07</td>
</tr>
</tbody>
</table>

---

Course name: Control Project
Project group: Duvan
Course code: TSRT71
Project: AUAV
E-mail: uav_liu@yahoogroups.com
Document responsible: Henrik Ohlsson
Author's E-mail: henoh509@student.liu.se
Document name: designspec_12.pdf
Project Identity

**Group E-mail:** uav.liu@yahoogroups.com
**Homepage:** http://www.edu.isy.liu.se/~henoh509

**Orderer:** David Törnqvist, Linköping University
**Phone:** +46 709-600447, **E-mail:** tornqvist@isy.liu.se

**Customer:** Rickard Karlsson, Linköping University
**Phone:** +46 13 281890, **E-mail:** rickard@isy.liu.se

**Course Responsible:** Anders Hansson, Linköping University
**Phone:** +46 13 281681, **E-mail:** hansson@isy.liu.se

**Project Manager:** The three managers of the sub-groups together
**Advisors:** Jeroen Hol, Linköping University
**Phone:** +46 13 282803, **E-mail:** hol@isy.liu.se
Johan Sjöberg, Linköping University
**Phone:** +46 13 282803, **E-mail:** johans@isy.liu.se

---

### Group Members

<table>
<thead>
<tr>
<th>Name</th>
<th>Responsibility</th>
<th>Phone</th>
<th>E-mail (@student.liu.se)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control Group</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sara Lundin</td>
<td>Project Manager</td>
<td>073-6297277</td>
<td>sarlu871</td>
</tr>
<tr>
<td>Henrik Ohlsson</td>
<td>Design Responsible</td>
<td>070-9200665</td>
<td>henoh509</td>
</tr>
<tr>
<td>Jonas Köhlström</td>
<td>Document Responsible</td>
<td>073-6307330</td>
<td>jonko288</td>
</tr>
<tr>
<td>Magnus Johansson</td>
<td>Test Responsible</td>
<td>070-6686134</td>
<td>magjo518</td>
</tr>
<tr>
<td>Mårten Hedborg</td>
<td>Quality Responsible</td>
<td>073-9638150</td>
<td>marhe866</td>
</tr>
<tr>
<td>Rikard Falkeborn</td>
<td>Customer Responsible</td>
<td>073-6456972</td>
<td>rikfa098</td>
</tr>
<tr>
<td><strong>Hardware Group</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Per Jonsson</td>
<td>Project Manager</td>
<td>070-5642016</td>
<td>perjo542</td>
</tr>
<tr>
<td>Dejen Meless</td>
<td>Design Responsible</td>
<td>070-8742417</td>
<td>dejme948</td>
</tr>
<tr>
<td>Robert Isaksson</td>
<td>Document Responsible</td>
<td>070-5864188</td>
<td>robis579</td>
</tr>
<tr>
<td>Giuseppe Franco</td>
<td>Test Responsible</td>
<td>070-1493535</td>
<td>giufr446</td>
</tr>
<tr>
<td><strong>Positioning Group</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carl Svärd</td>
<td>Project Manager</td>
<td>070-3925303</td>
<td>carsv595</td>
</tr>
<tr>
<td>Anders Jörnångard</td>
<td>Design Responsible</td>
<td>070-2665656</td>
<td>andjo454</td>
</tr>
<tr>
<td>Jan Betshammar</td>
<td>Document Responsible</td>
<td>073-9235603</td>
<td>janbo729</td>
</tr>
<tr>
<td>Tina Erlandsson</td>
<td>Document Responsible</td>
<td>070-6236579</td>
<td>timer501</td>
</tr>
<tr>
<td>Karl Andersson</td>
<td>Test Responsible</td>
<td>070-9130601</td>
<td>karan583</td>
</tr>
<tr>
<td>Kristina Jersénius</td>
<td>Quality Responsible</td>
<td>073-5690333</td>
<td>ingje224</td>
</tr>
<tr>
<td>Nina Wollinger</td>
<td>Customer Responsible</td>
<td>070-6579522</td>
<td>ninwo463</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Changes made</td>
<td>Sign</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>0.1</td>
<td>2006-02-23</td>
<td>First draft.</td>
<td>Henrik Ohlsson</td>
</tr>
<tr>
<td>0.2</td>
<td>2006-02-24</td>
<td>Second draft, Ctrl Chapter modified, <strong>rudder</strong> changed to <strong>delta_rudder</strong>.</td>
<td>Henrik Ohlsson</td>
</tr>
<tr>
<td>0.3</td>
<td>2006-02-26</td>
<td>Third draft, blockdiagram updated.</td>
<td>Henrik Ohlsson</td>
</tr>
<tr>
<td>0.4</td>
<td>2006-02-27</td>
<td>Fourth draft, HW part updated.</td>
<td>Henrik Ohlsson</td>
</tr>
<tr>
<td>0.5</td>
<td>2006-03-03</td>
<td>Fifth draft, a major update of basically all chapters.</td>
<td>Henrik Ohlsson</td>
</tr>
<tr>
<td>1.0</td>
<td>2006-03-12</td>
<td>Small adjustments in all chapters except the hardware chapter.</td>
<td>Henrik Ohlsson</td>
</tr>
<tr>
<td>1.1</td>
<td>2006-03-30</td>
<td>Some minor typo changes.</td>
<td>Rikard Falkeborn</td>
</tr>
<tr>
<td>1.2</td>
<td>2006-04-01</td>
<td>Some minor typo changes.</td>
<td>Sara Lundin</td>
</tr>
</tbody>
</table>

Course name: Control Project  
E-mail: uav_liu@yahoogroups.com  
Project group: Duvan  
Document responsible: Henrik Ohlsson  
Course code: TSRT71  
Author’s E-mail: henoh509@student.liu.se  
Project: AUAV  
Document name: designspec_12.pdf
# Contents

1 Introduction ............................................. 1  
   1.1 Involved Parties .................................... 1  
   1.2 Goals .................................................. 1  
   1.3 Definitions ......................................... 1  

2 System Overview ........................................... 2  

3 Hardware System ........................................... 3  
   3.1 General Description of the Hardware ................. 3  
   3.2 Interface Towards Other Subsystems .................. 3  
   3.3 Gatekeeper ............................................. 4  
      3.3.1 The Microcontroller .............................. 4  
   3.4 GPS ..................................................... 5  
   3.5 IMU ..................................................... 5  
   3.6 Linux Computer ....................................... 5  
      3.6.1 Software Environment ............................ 5  
      3.6.2 Software Communication .......................... 6  

4 Positioning System ....................................... 7  
   4.1 General Description of the Positioning System ....... 7  
   4.2 Interface Towards Other Subsystems ................. 7  
      4.2.1 Interaction With the Hardware System .......... 7  
      4.2.2 Interaction With the Control System .......... 7  
   4.3 Design of the Positioning System ................. 8  
      4.3.1 Block Diagram for the System ................. 8  
      4.3.2 Measurement Instruments ........................ 9  
      4.3.3 State-Space Model ................................ 9  
   4.4 Hardware Implementation ............................... 10  

5 Control System ........................................... 11  
   5.1 General Description of the Control System ........... 11  
   5.2 Interface Towards Other Subsystems ................. 11  
   5.3 Design of the Controller ............................. 11  
      5.3.1 $F_{xyz}$ Module .................................. 12  
      5.3.2 $F_{\alpha}$ Controller ........................... 12  
      5.3.3 $F_{\omega}$ Controller ............................ 12  
      5.3.4 $F_{rudders}$ Controller ......................... 12  
   5.4 Hardware Implementation ............................... 13  

A Coordinate Systems ...................................... 14  
B Quarternions ............................................. 14  
C Interface .................................................. 15  
D Data Logging ............................................. 16  
E Aeroplane Model .......................................... 17
E.1 Introduction to Aeroplane Model ............................................. 17
E.2 Interface ................................................................................. 17
E.3 Input Signal ........................................................................... 17
E.4 States ..................................................................................... 17
E.5 The Structure of the Aeroplane Model ..................................... 18
E.5.1 Aerodynamics ...................................................................... 18
E.5.2 Engine Block ....................................................................... 19
E.5.3 Gravitation ......................................................................... 19
E.5.4 Rigid Body Dynamics .......................................................... 20
F Linux Computer ........................................................................ 22
F.1 Special Considerations Regarding Software Environment ....... 22
1 Introduction

The purpose of the project is to develop a positioning and control system for an autonomous aeroplane. The positioning system will be based on a GPS, a 3-axial accelerometer, a 3-axial gyro, an electronic compass and if necessary additional sensors. Information from the positioning system and the control system will be processed by an embedded computer running on Linux. The computer will control two of the three rudders. For safety reasons the aeroplane engine will be controlled manually. The user should be able to steer the aeroplane manually or choose the autonomous mode once the aeroplane is in mid-air.

This document outlines the system and the implementation in hardware and software. It also defines the modules of the system and their purpose.

1.1 Involved Parties

The customer is Rickard Karlsson and the orderer is David Törnqvist from the department of Automatic Control at Linköping University. The project will be carried out by a group of 17 students in their fourth year of studies attending the course Automatic Control Project Course, TSRT71.

1.2 Goals

The goal of the project is to construct an autonomous unmanned aerial vehicle (AUAV) which should to be able to follow a pre-defined trajectory, given as a number of predefined points, using GPS and inertial navigation.

1.3 Definitions

AUAV Autonomous Unmanned Aerial Vehicle.

IMU Inertial Measurement Unit, a unit for determining the AUAV’s attitude and motion.

GPS Global Positioning System, a unit for determination of the AUAV’s position.
2 System Overview

The following chapter gives an overview of the system and its components.
A model aeroplane will be modified to autonomously follow a given trajectory while in mid-air. It will be possible to switch from manual mode to the autonomous when we are on flight. The model aeroplane will be equipped with a Linux computer, a GPS, a gyro, a compass and an accelerometer.

The system can be divided into the three subsystems:

1. Positioning system
2. Control system
3. Hardware system

The different subsystems and their dependencies are visualized in Figure 1. The positioning and control boxes symbolize packages of code running on the Linux computer aboard the plane and the hardware system is all the hardware including the model aeroplane, sensors and the Linux computer. The positioning system will get sensor data from the hardware system and provide the control system with state data. The control system will produce control signals that are used by the hardware system to steer the rudders.

Figure 1: Overview of the system
3 Hardware System

3.1 General Description of the Hardware

This chapter describes the hardware, see Figure 2 and 3. The hardware system consists of

- a Gatekeeper.
- a GPS unit.
- an IMU.
- a Linux computer.

![Figure 2: An overview of the hardware system.](image)

![Figure 3: A view of the hardware within the dashed part in previous figure.](image)

3.2 Interface Towards Other Subsystems

See Appendix E.2.
3.3 Gatekeeper

To increase reliability of the aeroplane a module called Gatekeeper will be implemented. The function of the Gatekeeper is to control the servos, see Figure 4. If the mode is set to *manual* the Gatekeeper will provide the servos signals from the radio receiver, and if the mode is set to *autonomous* the servos will be controlled by the Linux computer. If the Linux computer crashes the Gatekeeper should still be functional and the user should be able to maintain manual control of the aeroplane. There will also be more resources for other processes on the Linux computer when we have a Gatekeeper designated to control the servos.

The onboard computer will keep track of the aeroplane and register input signal, time and position. Data will most likely be recorded as a text file and once the aeroplane is landed it can be used to visualize the flown trajectory and to validate the model.

The Gatekeeper will consist of a microcontroller and necessary switches. A separate IC (Integrated Circuit) board will be used for the Gatekeeper.

3.3.1 The Microcontroller

The microcontroller that has been chosen is an ATmega16 from Atmel [1]. ATmega16 is an 8-bit RISC device. The microcontroller will run on a +5V power source and a clock frequency of 16Mhz.

**Communication with the Linux computer** will be performed through SPI (Serial Peripheral Interface) of the computer. The SPI interface on the Linux computer is based on +3.3V therefore an IC that transforms the level from +3.3V to +5V and vice versa will be used. This IC will probably be MAXIM’s MAX3390E.

**Communication with the servos** will be performed using PWM (Pulse Width Modulation). When the plane is in manual mode the servos will communicate with the radio receiver. The Gatekeeper will only track the behavior of the servos and send that information to the computer.

---

![Flowchart of Gatekeeper function.](image-url)
3.4 GPS

The GPS Smart Antenna Module, SAM-LS [7], will be providing information of the global position of the aeroplane, for further information see chapter 4 about the different subsystems. The GPS will be connected to the computer through a RS-232 serial port. A standard kernel driver will be used for the serial port.

As the GPS supports different protocols the project group decided to use the NMEA 0183 which is an international standard. The NMEA 0183 Interface Standard defines electrical signal requirements, data transmission protocol, time protocol and specific sentence formats for a 4800-baud serial data bus.

The position coordinates, latitude, longitude and height in meters above the mean sea level will be extracted from the NMEA message.

3.5 IMU

The IMU will also be connected to the computer through a RS-232 serial port. The purpose of the IMU is to provide information of the inertial motion of the aeroplane, for further information see chapter 4 about the different subsystems. Communication with the IMU is simply performed through its own binary communication protocol, for detailed information regarding the protocol please refer to the MTi/MTx Low-level communication protocol found in the user manual [8].

3.6 Linux Computer

Figure 5 shows the Linux computer that will be onboard the aeroplane. Both the positioning system and control system will be implemented on this computer.

![Image of onboard computer, Embedded Arm TS-7200.](image-url)

Figure 5: The onboard computer, Embedded Arm TS-7200.

The system will run a Linux kernel most likely with real-time patches, provided by the supplier of the computer, to make it possible to take advantage of the possibilities a real-time kernel provides for implementing periodic tasks.

3.6.1 Software Environment

Most, if not all, real-time tasks will run in user-space, using the LXRT (Linux RealTime) module available with the RTAI (RealTime Application Interface) API. The reason for
running in user-space is to reduce the vulnerability of the system to errors in the software. Kernel-space code is much more prone to introduce errors that affect the whole system. User-space has a much smaller likelihood of creating errors that causes the computer system to crash, see Appendix F.1 for further discussion regarding this matter.

3.6.2 Software Communication

All code running on the computer that communicates with external hardware will be subdivided into different parts. One part will run in kernel-space and do the low-level operations of communicating with the hardware. The other part will run in user-space and take care of communications with the different modules (GPS and IMU) using their respective protocols to provide the positioning and control system with necessary data. The data that is going to be provided is listed in Appendix C.

This division is done to ensure that as little code as possible runs with higher privileges than necessary to ensure stability of the system.

For communications with the Gatekeeper module, and thus the servo hardware, the SPI (Serial Peripheral Interface) of the computer will be used. There are no drivers for the SPI capabilities of the computer so a driver needs to be implemented.
4 Positioning System

4.1 General Description of the Positioning System

The purpose of the positioning system is to provide the control system with the data that is necessary for controlling the aeroplane so that the predefined trajectory can be followed. The positioning will be performed using GPS and inertial navigation. The measurements from the inertial sensor are very accurate, but have errors that have a tendency to drift in time. The GPS on the other hand gives noisy but unbiased measurements. The system will combine the different types of measurement information with a model describing the relation between the quantities, as well as the dynamics of the aeroplane. By using suitable signal processing estimations of the position, velocity, attitude, acceleration and angular velocity will be obtained.

The measurements units, i.e. the IMU and the GPS, will provide data in two different frames. The GPS coordinates will be given in Earth Frame (E) and the data from the IMU in IMU Frame (I), see Appendix A.

4.2 Interface Towards Other Subsystems

Figure 6 shows how the Positioning System interacts with the other two subsystems.

![Diagram of interaction between GPS, IMU, Positioning, and Control](image)

Figure 6: The interaction of the Positioning System

4.2.1 Interaction With the Hardware System

The Positioning System will receive a vector called \( \text{Imu} \) from the Hardware System. The vector will contain the three-dimensional acceleration, angular velocity and earth magnetic field measured by the IMU in the IMU frame, (I), with respect to the Locally level Frame (L). The Hardware System will provide the Positioning System with three GPS coordinates (longitude, latitude and altitude) in the Earth frame (E) in a vector called \( \text{Gps} \). A scalar called \( \text{Throttle} \) with information regarding the level of throttle is also provided by the Hardware System. See also Appendices A and C.

4.2.2 Interaction With the Control System

The Positioning System will deliver a vector called \( \text{ISV} \) containing position in an earth-fixed coordinate system (L) and velocity, acceleration, attitude and angular velocity in a body-fixed coordinate system (B) to the Control System. The Control System will provide a vector called \( \text{Rudders} \) containing information about rudder angles. See also Appendices A and C.
4.3 Design of the Positioning System

The measurements from the IMU will be used as input signals to a model and estimations of position, velocity, attitude, acceleration and angular velocity will be calculated. With aiding measurements of the position from the GPS and attitude and acceleration from the IMU, a Kalman Filter is used to estimate the errors of the IMU measurements. After that corrected estimations, of the above mentioned quantities, will be calculated. The main idea behind this procedure can be found in [2].

4.3.1 Block Diagram for the System

A block diagram for the Positioning System is shown in Figure 7.

In the block Coord. Transform. the longitude and latitude obtained from the GPS are transformed to the position of the aeroplane, expressed in metres in the Locally level Frame (L). The block Attitude uses the compass direction to calculate a part of the attitude of the aeroplane.

The positioning system uses an Extended Kalman Filter (EKF) to process measurement data, therefore linearization will be done in every point according to [4]. The time update occurs in the block Model and the measurement update in the block KF. The states of the state-space model and the EKF are position expressed in the Locally level Frame (L), velocity and acceleration in the Body Frame (B) and attitude. All quantities are three dimensional, except for the attitude, which is represented in quaternions and therefore is four dimensional, see further [3].

Input signals to the block Model are the acceleration and angular velocity measured by the IMU and rudder angles and throttle provided by the Control System. The output signal \( \hat{x}_{t|t-1} \) contains position, velocity, corrected acceleration, attitude and corrected angular velocity. \( h(\hat{x}_{t|t-1}) \) contains position and attitude. The errors in \( \hat{x}_{t|t-1} \) are estimated using signals from the GPS and the compass of the IMU in the block KF. Finally the estimated error is added to the output signal of the block Model.
4.3.2 Measurement Instruments

The measuring instruments involved are a GPS and an IMU. The IMU will provide the positioning system with the acceleration and angular velocity in three dimensions, and also a compass vector describing the attitude of the aeroplane. The measurements from the IMU are very accurate, but include bias and scale errors. The biased and scaled errors are considered as constants that are found by means of calibration. These errors cause the measurements to drift in time. The position measurements from the GPS are inaccurate, but the errors are quite stable in time. The IMU measurements are updated more often than the GPS measurements, which are updated only four times per second. By combining the different types of measurement information with a model, describing the relation between the quantities and the dynamics of the aeroplane, good estimations of the quantities can be calculated.

4.3.3 State-Space Model

As mentioned above, a model describing the dynamic relation between quantities and the dynamics of the aeroplane will be used. The model will be used for updating measurements (input signals) from the IMU and also, together with a Kalman Filter, for measurement update with measurements from the GPS. Part of the model will be same as the Aeroplane model described in Appendix [E]. The state-space model are internal and used by the positioning group only. The states are:

<table>
<thead>
<tr>
<th>States</th>
<th>Coordinate system</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Locally level Frame (L)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Velocity</td>
<td>Body Frame (B)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Body Frame (B)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Attitude</td>
<td>-</td>
<td>$\epsilon_0$, $\epsilon_1$, $\epsilon_2$, $\epsilon_3$</td>
</tr>
</tbody>
</table>

The scale and bias error for the measured acceleration and angular velocity will in the initial phase of the project be considered as constants. If the calculation capacity of the computer is enough they will become states in the state-space model.

The input signals to the model are the measurements from the IMU, they are provided by the Hardware System and delivered as some elements in the vector called $\text{Imu}$. The model also use the rudder angles provided by the Control System in a vector called $\text{Rudders}$ and a scalar called $\text{Throttle}$ with information regarding the level of throttle provided by the Hardware System. See also Appendix [C].

The aiding measurement signals are used to estimate the errors in the inertial measurements and the estimated states. They are provided by the Hardware subsystem and delivered as a vector called $\text{Gps}$, and some of the elements of the vector called $\text{Imu}$, see also Appendix [C]. They consists of:

- Measured angular velocity
- Rudder angles (Elevator, Aileron)
- Throttle

<table>
<thead>
<tr>
<th>Input signals</th>
<th>Coordinate system</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measured angular velocity</td>
<td>IMU Frame (I)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Rudder angles (Elevator, Aileron)</td>
<td>-</td>
<td>$\delta_e$, $\delta_a$</td>
</tr>
<tr>
<td>Throttle</td>
<td>-</td>
<td>scalar</td>
</tr>
</tbody>
</table>

The aiding measurement signals are used to estimate the errors in the inertial measurements and the estimated states. They are provided by the Hardware subsystem and delivered as a vector called $\text{Gps}$, and some of the elements of the vector called $\text{Imu}$, see also Appendix [C]. They consists of:

Output signals from the model are the corrected estimates of the state vector. They are given to the Control subsystem and delivered as a vector called $\text{ISV}$, see also Appendix [C]. The output signals are:
<table>
<thead>
<tr>
<th>Measurement signals</th>
<th>Coordinate system</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position from GPS</td>
<td>GPS Frame (G)</td>
<td>long, lat, z</td>
</tr>
<tr>
<td>Compass direction from IMU</td>
<td>IMU Frame (I)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Acceleration from IMU</td>
<td>IMU Frame (I)</td>
<td>x, y, z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output signals</th>
<th>Coordinate system</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>Locally level Frame (L)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Velocity</td>
<td>Body Frame (B)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Body Frame (B)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Angular velocity</td>
<td>Body Frame (B)</td>
<td>x, y, z</td>
</tr>
<tr>
<td>Attitude</td>
<td>-</td>
<td>e₀, e₁, e₂, e₃</td>
</tr>
</tbody>
</table>

### 4.4 Hardware Implementation

The Positioning System will be developed in MATLAB but implemented in C on the Linux computer, either using C-code generated in MATLAB or by immediate C-coding.
5 Control System

5.1 General Description of the Control System

The purpose of the control system is to control the AUAV in such a way that it follows the predefined position points. A model of the AUAV will be used to design the controller. The controller and the model will be developed in Matlab/Simulink, but implemented in C. As a help in the development of the control system a tool for analysis and visualization will be created.

5.2 Interface Towards Other Subsystems

The input signals for the control system are the predefined position points, the throttle value, and the estimated states of the AUAV. With the help of these signals an output will be created as angles on the different rudders.

Input signals/vector:

- **ISV** set by the Positioning system.
- **Throttle** set by the Hardware system.

Output signals/vector:

- **Rudders** used by the Hardware system.

Predefined position points are also given to the control system. These positioning points will be decided before takeoff and stored in the memory of the computer.

For more information about the input and output vector see Appendix C

5.3 Design of the Controller

The control system is made up by four different modules, see Figure 8. \( F_{xyz} \) decides what position point to aim for and calculates an angle \( \beta \) and a distance \( l \) while the other modules are controllers in a more normal sense.

Inputs to the controllers are \( \alpha_{est}, \omega_{est} \) and \( xyz_{est} \). These are all vectors taken from the control systems input vector ISV. \( \alpha \) is the angle between the velocity vector for the aeroplane and the direct path to the aimed positioning point while \( \omega \) is the change in \( \alpha \) i.e. angle velocity. \( xyz \) is a position.

There is a chance that other elements of ISV will be included as inputs to the controllers if it is needed for calculating a reference value. The described control system is only an overview of how to solve the control problem.

Figure 8: An overview of the control system
5.3.1 $F_{xyz}$ Module

When the aeroplane is switched into autonomous mode the AUAV is supposed to follow certain position points. Given these predefined reference position points and the current estimated position of the aeroplane the $F_{xyz}$ module decides what position point to aim for. The module creates a straight trajectory between the new and just passed position points. This line is called the optimal trajectory. An angle $\beta$ is calculated as the angle between the optimal trajectory and the shortest trajectory of the aeroplane to the new position point, see Figure 9. $l$ is also calculated as the orthogonal distance to the optimal trajectory.

The AUAV has passed a position point when it has flown through a two dimensional plane orthogonal to the optimal trajectory.

![Figure 9: How $\beta$ is calculated](image)

5.3.2 $F_\alpha$ Controller

The $F_\alpha$ controller will try to minimize the difference between the position of the next position point and the estimated position for the aeroplane, $xyz_{ref}$ - $xyz_{est}$, as well as minimizing the $\beta$ angle or/and the distance $l$. The output of the controller will be a reference value for the angles of the aeroplane.

5.3.3 $F_\omega$ Controller

The $F_\omega$ controller will create a reference value for angle velocities using reference and estimated angles.

5.3.4 $F_{rudders}$ Controller

Using the angle velocity reference and estimation the $F_{rudders}$ controller sets the vector Rudders.
5.4 Hardware Implementation

The controller will be implemented in C, either using C-code generated in Matlab or by immediate C-coding.

Visualization and Analysis

The trajectory and the flown path will be presented partially on a detailed map from Google Earth. MyGPS is the program chosen to calibrate the map and display the above mentioned data. This program is very user-friendly and position data can easily be imported as well as exported via text-files. To get a measure of how well the trajectory was followed by the airplane, a Matlab program for analysis will be written. This program will also complement MyGPS with additional plots and functionality. If there is enough time additional useful data apart from the position will also be presented. In Figure 10 a possible graphical visualization interface is presented.

![Possible graphical visualization](image.png)

Figure 10: Possible graphical visualization
APPENDIX

A Coordinate Systems

The different quantities will be expressed in five coordinate systems [6]:

**Earth Frame (E)**  The GPS gives the position in longitude, latitude and altitude, height above the Mean sea level, in the Earth Frame (E). The \( Z_E \)-axis is pointing through the north pole and the \( X_E \)-axis is directed along the Greenwich meridian.

**Locally level Frame (L)**  The origin of the Locally level Frame (L) is placed at a fixed point on the ground. The \( X_L \)-axis is pointing towards the north, the \( Y_L \)-axis towards the east and the \( Z_L \)-axis towards the center of earth.

**Body Frame (B)**  The Body Frame (B) is attached in the center of mass of the aeroplane and rotates with the aeroplane. The \( Z_B \)-axis is pointing downwards according to the aeroplane, the \( X_B \)-axis is pointing through the nose and the \( Y_B \)-axis is directed along the right wing. The B-frame is related to the L-frame with an attitude and a distance.

**GPS Frame (G)**  The GPS Frame (G) is attached to the GPS. Like (B) the GPS frame rotates with the aeroplane. Observe that the measurements from the GPS will not be given in this frame but more so in the Earth frame.

**IMU Frame (I)**  The IMU Frame (I) is attached to the IMU. Like (B) and (G) the IMU frame rotates with the aeroplane. Measured data from the IMU will be given in this frame.

B Quarternions

This appendix will describe how a vector in cartesian coordinates is represented in quaternions [3, 6]. It will also explain how calculations like translation and rotation from a coordinate system a to b is performed.

Basic Definitions

A quaternion \( e \in \mathbb{Q} = \{ \mathbb{R}^4 : ee^T = 1 \} \), is a 4-tuple of real numbers is denoted by \( e = (e_0, e_1, e_2, e_3) \). Alternatively it is denoted by \( e = (e_0, \mathbf{e}) \), where \( e_0 \) is called the scalar part and \( \mathbf{e} \) the vector part of a quaternion. For quaternions the following operators will be used:

- addition \( p + e = (p_0 + e_0, \mathbf{p} + \mathbf{e}) \)
- multiplication \( p \odot e = (p_0e_0 - \mathbf{p} \cdot \mathbf{e}, p_0\mathbf{e} + e_0\mathbf{p} + \mathbf{p} \times \mathbf{e}) \)
- conjugation \( e^* = (e_0, -\mathbf{e}) \)

A cartesian vector \( \mathbf{x} \in \mathbb{R}^3 \) is represented in quaternions as \((0, \mathbf{x}) \in \mathbb{R}^4\).
Translation

The translation from the \( b \) to the \( a \) coordinate system of \( \mathbf{e}^a \) is defined by

\[
\mathbf{e}^a = \mathbf{e}^b + \mathbf{v}^a,
\]

where \( \mathbf{v}^a = (0, \mathbf{v}) \) and \( \mathbf{v} \) is the vector from the origin of coordinate system \( a \) to the origin of coordinate system \( b \), expressed in coordinate system \( a \).

Rotation

A rotation matrix from the \( b \) to the \( a \) coordinate system, \( R^{ab} \), in Cartesian coordinates is defined by

\[
R^{ab} = \begin{pmatrix}
  x_x & x_y & x_z \\
  y_x & y_y & y_z \\
  z_x & z_y & z_z
\end{pmatrix}.
\]

This rotation matrix can be represented in a quaternion vector, \( \mathbf{e}^{ab} \), with the following elements:

\[
e_0 = \frac{\sqrt{x_x + y_y + z_z + 1}}{2}
\]

where \( q_1 \) is negative if \( z_y - y_z \) is negative and otherwise positive.

\[
e_1 = \frac{\sqrt{x_x - y_y - z_z + 1}}{2},
\]

where \( q_2 \) is negative if \( x_z - z_x \) is negative and otherwise positive.

\[
e_2 = \frac{\sqrt{-x_x - y_y + z_z + 1}}{2},
\]

where \( q_3 \) is negative if \( y_x - x_y \) is negative and otherwise positive.

The rotation from the \( b \) to the \( a \) coordinate system, \( R^{ab} \), of a position, \( x \), is defined using quaternion algebra by

\[
x^a = R^{ab}(x^b) \equiv \mathbf{e}^{ab} \odot x^b \odot (\mathbf{e}^{ab})^c.
\]

C Interface

Set by Hardware System, Used by Positioning System

Vector name: \( \text{Gps} \)

- GPS-position (long,lat,z)

Vector name: \( \text{Imu} \)

- Acceleration (\( x, y, z \))
- Compass (\( x, y, z \))
- Angular velocity (\( x, y, z \))

All data are unprocessed, "raw", and therefore given in the coordinate frame given by the different measurement units. The \( \text{Imu} \)-vector will be in the IMU frame (I) and the \( \text{Gps} \)-vector in an Earth frame (E), see also Appendix [A].
Set by Hardware System, Used by Positioning System, Control System

Vector name: 

- Throttle ($\gamma$)

Scalar symbolizing the level of the throttle.

Set by Positioning System, Used by Control System

Vector name: ISV

- Position ($x^e, y^e, z^e$)
- Velocity ($x^b, y^b, z^b$)
- Acceleration ($x^b, y^b, z^b$)
- Angular velocity ($x^b, y^b, z^b$)
- Attitude ($e_0, e_1, e_2, e_3$)

The positioning is given in a local level coordinate system (L) while all the others are given in a body-fixed system (B). ISV refers to Interface State Vector.

Set by Control System, Used by Hardware System, Positioning System

Vector name: Rudders

- Elevator ($\delta_e$)
- Aileron ($\delta_a$)

Scalars symbolizing the angles of the two rudders that can be controlled.

D Data Logging

Set by Hardware System

Vector saved: ISV

In lack of memory the positioning data in ISV has highest priory.
E Aeroplane Model

This appendix describes the aeroplane model.

E.1 Introduction to Aeroplane Model

The purpose of the model is to simulate an aeroplane with aerodynamic forces, engine forces, gravitation force and rigid body dynamics. The model will be used in the development of the control- and positioning system.

E.2 Interface

Figure 11 shows the input and output signals of the aeroplane model. A closer description of the signals can be found in the following sections.

![Aeroplane model](image)

Figure 11: Aeroplane model

E.3 Input Signal

Inputs to the aeroplane model are the two vectors *Rudders* and *Throttle*.

*Rudders* consist of the elements:

- **Elevator** describes the angle of the elevator, \((\delta_e)\), in radians.
- **Aileron** describes the angle of the aileron, \((\delta_a)\), in radians.

*Throttle* consist of:

- **Throttle** describes the level of the throttle, \((\gamma)\), between 0 and 1.

E.4 States

The output from the model consists of a vector, *ISV* with 16 elements.

- **Position** Three elements describing the position of the aeroplane in an inertial system (L).
- **Velocity** Three elements describing the velocity of the aeroplane in a body-fixed system (B).
- **Acceleration** Three elements describing the acceleration of the aeroplane in a body-fixed system (B).
Angular velocity Three elements describing the angular velocities of the aeroplane in a body-fixed system (B).

Attitude Four elements describing the orientation of the aeroplane expressed in quaternions.

E.5 The Structure of the Aeroplane Model

The aeroplane model is divided into four separate blocks, aerodynamics, gravitation, engine and rigid body dynamics. A block diagram can be seen in figure 12.

Figure 12: Block diagram of the aeroplane model

The following section will describe these blocks in detail.

E.5.1 Aerodynamics

This block calculates the aerodynamic forces and moment that affect the aeroplane. To be able to perform these calculations the block uses the control signals, velocities and angular velocities of the aeroplane.

Inputs:

Rudder A vector containing two elements, describing the angles of the elevator and the aileron.

Angular velocity A vector containing three elements, (p q r), describing the angular velocity of the aeroplane expressed in a body-fixed system.

Velocity A vector containing three elements, (u v w), describing the velocity of the aeroplane expressed in a body-fixed system.

det

Outputs:
\( \mathbf{F}_{\text{aero}} \) A vector containing the resulting aerodynamic forces, \((X, Y, Z)\), expressed in a body-fixed system \((B)\). \(X\), \(Y\), and \(Z\) are scalar and \(X\) points in \(x^b\) direction, \(Y\) in \(y^b\) and \(Z\) in \(z^b\).

\( \mathbf{M}_{\text{aero}} \) A vector containing the resulting aerodynamic moment, \((L, M, N)\), expressed in a body-fixed system \((B)\). \(L\), \(M\) and \(N\) are scalar and \(L\) are moment around \(x^b\), \(M\) around \(y^b\) and \(N\) around \(z^b\).

The input signals **Angular velocity** and **Velocity** are feedback signals from the rigid body dynamics block. The vector **Rudder** is one of the two input signals to the aeroplane model. Small deviations about a steady flight condition are assumed. Then the equations are:\(\text{[5] page 107}\)

\[
X/m = X_u (u - u_0) + X_w (w - w_0) + X_\delta_e (\delta_e - \delta_e_0) + X_\delta_t (\delta_t - \delta_t_0)
\]

\[
X_u = \frac{\partial X}{\partial u}, X_w = \frac{\partial X}{\partial w}, X_\delta_e = \frac{\partial X}{\partial \delta_e}
\]

\[
Y/m = Y_v (v - v_0) + Y_p (p - p_0) + Y_r (r - r_0)
\]

\[
Y_v = \frac{\partial Y}{\partial v}, Y_p = \frac{\partial Y}{\partial p}, Y_r = \frac{\partial Y}{\partial r}
\]

\[
Z/m = Z_u (u - u_0) + Z_w (w - w_0) + Z_q (q - q_0) + Z_\delta_e (\delta_e - \delta_e_0)
\]

\[
Z_u = \frac{\partial Z}{\partial u}, Z_w = \frac{\partial Z}{\partial w}, Z_q = \frac{\partial Z}{\partial q}, Z_\delta_e = \frac{\partial Z}{\partial \delta_e}
\]

\[
L/I_x = L_v (v - v_0) + L_p (p - p_0) + L_r (r - r_0) + L_\delta_a (\delta_a - \delta_a_0)
\]

\[
L_v = \frac{\partial L}{\partial v}, L_p = \frac{\partial L}{\partial p}, L_r = \frac{\partial L}{\partial r}
\]

\[
M/I_y = M_u (u - u_0) + M_w (w - w_0) + M_q (q - q_0) + M_\delta_e (\delta_e - \delta_e_0)
\]

\[
M_u = \frac{\partial M}{\partial u}, M_w = \frac{\partial M}{\partial w}, M_q = \frac{\partial M}{\partial q}, M_\delta_e = \frac{\partial M}{\partial \delta_e}
\]

\[
N/I_z = N_v (v - v_0) + N_p (p - p_0) + N_r (r - r_0) + N_\delta_a (\delta_a - \delta_a_0)
\]

\[
N_v = \frac{\partial N}{\partial v}, N_p = \frac{\partial N}{\partial p}, N_r = \frac{\partial N}{\partial r}, N_\delta_a = \frac{\partial N}{\partial \delta_a}
\]

To simplify the model we will use constants that we have estimated from the shape of the aeroplane instead of the different derivatives. \(I_x, I_y\) and \(I_z\) are mass moments of inertia.

### E.5.2 Engine Block

This block consists of a simple model translating the input signal, **Throttle**, to drag force \( (\mathbf{F}_{\text{drag}}) \). Any moment caused by the engine is neglected.

### E.5.3 Gravitation

This block computes the gravitation forces in a body-fixed system \((B)\). The gravitation force is assumed to be constant. The input signals are the orientation of the aeroplane.
expressed in quaternions, the mass of the aeroplane and the gravitation constant. The output signal is a vector containing the three forces, $F_{gx}$, $F_{gy}$ and $F_{gz}$.

$$
\begin{pmatrix}
F_{bx}^{b_g} \\
F_{by}^{b_g} \\
F_{bz}^{b_g}
\end{pmatrix} =
\begin{pmatrix}
(e_0^2 + e_1^2 - e_2^2 - e_3^2) & 2(e_1 e_2 + e_0 e_3) & 2(e_1 e_3 - e_0 e_2) \\
2(e_1 e_2 - e_0 e_3) & (e_0^2 - e_1^2 + e_2^2 - e_3^2) & 2(e_2 e_3 + e_0 e_1) \\
2(e_1 e_3 + e_0 e_2) & 2(e_2 e_3 - e_0 e_1) & (e_0^2 - e_1^2 - e_2^2 + e_3^2)
\end{pmatrix}
\begin{pmatrix}
0 \\
0 \\
mg
\end{pmatrix}
\begin{pmatrix}
R
\end{pmatrix}
$$

where $R$ is the rotation-matrix using quaternions. The quaternions are $\vec{e} = (e_0, e_1, e_2, e_3)$.

**E.5.4 Rigid Body Dynamics**

This block uses the output from the three previous blocks. These signals together with the equations of motion are used to calculate the velocity, acceleration and position of the aeroplane.

![Block diagram of the rigid body dynamics model](image)

The rigid body dynamics consists of three subsystems; total acceleration, total moment and equation of motion. Figure 13 shows the signals and subsystems of the block.

**Total Acceleration**

Inputs:

- $F_{aero}$
- $F_g$
- $F_{drag}$

Output:

- Acceleration - A vector containing three elements, $(a_x, a_y, a_z)$, describing the acceleration of the aeroplane in a body-fixed system (B)
Total Moment

Inputs:
- $M_{aero}$
- $F_{aero}$
- $C_g$ - Center of gravity

Output:
- Moment - The total moment of the aeroplane relative to $C_g$

Equation of Motion

Inputs:
- Acceleration
- Moment

Outputs:
- Position
- Velocity
- Acceleration
- Angular velocity
- Attitude

To calculate the outputs we use a set of non-linear equations. ([5] page 105)

\[
\dot{u} = -qw + rv + \frac{X}{m} + \frac{F_{gx}}{m}
\]
\[
\dot{v} = -ru + pw + \frac{Y}{m} + \frac{F_{gy}}{m}
\]
\[
\dot{w} = -pv + qu + \frac{Z}{m} + \frac{F_{gz}}{m}
\]
\[
\dot{p} = \left(\frac{I_z - I_x}{I_x}\right)q + \frac{L}{I_x}
\]
\[
\dot{q} = \left(\frac{I_x - I_z}{I_z}\right)p + \frac{M}{I_y}
\]
\[
\dot{r} = \left(\frac{I_y - I_z}{I_z}\right)p + \frac{N}{I_z}
\]
\[
\begin{bmatrix}
\dot{e}_0 \\
\dot{e}_1 \\
\dot{e}_2 \\
\dot{e}_3
\end{bmatrix}
= \frac{1}{2}
\begin{bmatrix}
0 & -p & -q & -r \\
p & 0 & r & -q \\
q & -r & 0 & p \\
r & q & -p & 0
\end{bmatrix}
\begin{bmatrix}
e_0 \\
e_1 \\
e_2 \\
e_3
\end{bmatrix}
+ \lambda
\begin{bmatrix}
e_0 \\
e_1 \\
e_2 \\
e_3
\end{bmatrix}
\]
\[
\lambda = 1 - (e_0^2 + e_1^2 + e_2^2 + e_3^2)
\]

\[
\begin{bmatrix}
x^f \\
y^f \\
z^f
\end{bmatrix}
= R^T
\begin{bmatrix}
u \\
v \\
w
\end{bmatrix}
\]
\[
R = \begin{bmatrix}
e_0^2 + e_1^2 - e_2^2 - e_3^2 & 2(e_1e_2 + e_0e_3) & 2(e_1e_3 - e_0e_2) \\
2(e_1e_2 - e_0e_3) & e_0^2 - e_1^2 + e_2^2 - e_3^2 & 2(e_3e_2 + e_0e_1) \\
2(e_1e_3 + e_0e_2) & 2(e_3e_2 - e_0e_1) & e_0^2 - e_1^2 - e_2^2 + e_3^2
\end{bmatrix}
\]

F Linux Computer

F.1 Special Considerations Regarding Software Environment

It is quite important to be careful when running code on an embedded system. During initial research of the capabilities of the system, it was found that it is crucial to use a correctly configured toolchain when building code for the system.

The toolchain, in this case gcc and glibc, must be built with the knowledge that the computer lacks a floating point operations unit in the processor. If an improperly configured toolchain is used, the performance of code using floating point operations will be extremely bad.
References


