Bachelor Thesis

Ar.Drone Autonomous Control and Position Determination

Jacco van der Spek 4002512

Mario Voorsluys 4048482

Delft University of Technology

June 2012

TU Delft Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science

Challenge the future

Bachelor Thesis V1.0 (Wednesday 27$^{\text{th}}$ June, 2012)

This Bachelor thesis is made using LaTeX in Computer Modern Roman 11pt.

Preface

For a not yet established Unmanned Aerial Vehicle (UAV) competition a prototype UAV is needed. This UAV has to be able to recognize a moving vehicle. When this vehicle is recognized the UAV should be able to keep tracking the vehicle. This prototype will be used to show that the challenges of the competition are feasible. It will also be used to make promotional materials like videos.

Two teams will be working on this prototype; a team of ten students of Aerospace Engineering and a team of six students of Electrical Engineering. The Aerospace Engineering students will design a fixed wing UAV. We, as Electrical Engineering students, will develop the electrical system. A design brief was drafted to state the requirements we want to meet (see Appendix A).

The development of the electrical system is split up in three sub-groups of two students. One team is responsible for the development of a data connection to a ground station. They also visualize the information that is exchanged between the UAV and the ground station. The second team develops an intelligent camera system that recognizes and tracks the vehicle. We, the authors of this thesis, will develop a sensor system and enable our test UAV to fly autonomously. A functional block diagram of the whole system can be found in Appendix B where the elements of our subsystem are highlighted. The three subsystems will form one system that, we think, will deliver a proof of concept for our clients: T. Durieux BSc and J. Melis BSc.

Jacco van der Spek & Mario Voorsluys
Delft, June 2012

Acknowledgements

We would like to thank the following people and organisations:

Our supervisor dr. ir. C.J.M. Verhoeven for his continued enthusiasm and (financial) support
T. Durieux B.Sc. and J. Melis B.Sc. for their support and the assignment
The Microelectronics Department for the provision of two quadrocopters and budget
The Raspberry Pi foundation for their support with a Raspberry Pi.
Our team members for the collaboration and great time together

Summary

For a prototype UAV an object recognition and tracking system had to be developed. This system has been developed by a team of six Electrical Engineering students. The system has been tested on a Parrot AR.Drone, which is a commercially available drone. The system has been divided in three subsystems and each subsystem has been designed by a team of two people. The first subsystem creates a data connection between the system and a ground station and displays this data in a Graphical User Interface. The second subsystem is tracking a moving object and calculating the position of this object. The third subsystem is a system that determines the position of the AR.Drone and controls the AR.Drone. The third subsystem has been designed by the two writers of this thesis.

The aim of this thesis was to select a suitable positioning method and design an autonomous control for the AR.Drone. Therefore two main questions were formulated:

1. What positioning method should be used for determining the position of a UAV?

2. How can the AR.Drone have autonomous control?

It is important to know the position (including the altitude) of the UAV for three reasons. The first reason is a safety reason: by knowing the position it is able to prevent crashes by adjusting the course of the UAV. The second reason is that for the calculation of the position of the object it is necessary to know the altitude. The third reason is that the data communication team wanted to display the location of the UAV on a map. In order to track the recognized object, an autonomous control for the AR.Drone was designed. This control enables the AR.Drone to follow the moving object.

To answer the two main questions, first research was done into appropriate positioning methods and AR.Drone control options. Then the results of this research were used to implement the desired positioning and control methods for the AR.Drone. Thereafter the implementation was tested.

To select an appropriate positioning method, five different methods were looked into: Global Positioning System (GPS), Inertial Navigation System (INS), a GPS/INS combination, a GPS/barometer combination and stereo vision. After investigating these methods, it was decided to use the combination of a GPS sensor and a barometer. The GPS sensor is used for positioning in the horizontal plane while the barometer is used to determine the altitude. This method was chosen because it gives an acceptable accuracy for a reasonable price.

To build an autonomous control for the AR.Drone first a control model was derived. The AR.Drone has got four control variables these are: *pitch*, *roll*, *yaw* and vertical speed (*gaz*). To control the x and y position, it is necessary to control the *pitch* and the *roll* respectively. The parameters of the control were determined empirically by sending commands to the drone and collecting the data of its sensors.

To run this control model an appropriate hardware solution had to be selected. There were three options for implementing the control. The first option is remote control which means the AR.Drone is completely controlled by a ground station. The second option is onboard processing without extra hardware. The third option is onboard processing with extra hardware. Because the video processing and communication handling should also be done onboard, it was decided to choose the option which uses extra hardware on the drone.

The combination of a GPS sensor and a barometer was implemented and tested. The position determination can be done with an accuracy of 3 meters in the horizontal plane and an accuracy of 1.2 meters in the vertical plane. This is respectively 50 % and 20 % above the design specifications. Therefore the positioning method should be improved to meet the design specifications. However it is recommended to use a position determination method that is compatible with and required by the autopilot system. When this autopilot system does not deliver the required data; an improved version of the chosen determination method, possibly supplemented by an INS should be used.

The autonomous control was designed by a three layered model. The lowest layer is already implemented in the AR.Drone, this layer controls the motors and has *yaw*, *pitch*, *roll* and *gaz* as input. The highest layer generates the flight commands based on the position of the object, this layer was implemented in combination with the object tracking subsystem. The middle layer translates the flight commands to the input of the lowest layer. This middle layer was designed by first modelling the AR.Drone's behaviour and designing control loops subsequently.

Due to a software bug, the control loops could not be tested in the given time frame. Therefore it is unknown if the designed control loops are able to control the AR.Drone.

Contents

List of Figures

List of Tables

List of Abbreviations

gaz	vertical speed
I^2C	Inter-Integrated Circuit
pitch	turning around y-axis
roll	turning around x-axis
yaw	turning around z-axis
COTS	Commercial Of The Shelf
CPU	Central Processing Unit
ftp	file transfer protocol
GLONASS	Globalnaya Navigatsionnaya Sputnikovaya Sistema or Global Navigation Satellite System
GNSS	Global Navigation Satellite System
GPL	General Public License
GPRMC	Global Position Recommended Minimum sentence version C
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
NMEA	National Marine Electronics Association
SDK	Software Development Kit
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol

USB	Universal Serial Bus
WGS84	World Geodetic System 1984

Chapter 1

Introduction

UAVs (Unmanned Air Vehicles) are nowadays used for a wide range of applications ranging from rescuing people [1] to acting on a battlefield [2]. For all these applications it is essential to know the position of the UAV.

There are various methods for determining the position of a UAV. It can be done by using a GNSS (Global Navigation Satellite System) receiver [3]. Another method used by [4] is INS (Inertial Navigation System). A third method combines these systems [4]. Hereby the INS is a complementary system of the GPS, the INS provides a high accurate position when the GPS accuracy is poor [5]. According to [5, 6, 7, 8, 9] the combination of GNSS/INS is a good method for determining the position of a UAV. [10] and [11] propose the use of stereo vision to determine the position of a UAV.

Not only the position of a UAV is important, the determination of the altitude is even more important. The aforementioned methods for determining the position are also suited for determining the altitude of a UAV [5], [6]. [12] uses a barometer to measure the pressure. A method used by [11] is stereo vision to determine the altitude. However GPS/INS and barometer sensors are less suited for landing due to a too low accuracy [13]. [13] proposes two methods for determining the altitude while landing: using a wireless sensor network or using ultrasonic sensors. Another method for measuring the altitude while landing is a circular mark placed on a runway which is recognized by a camera [14].

Applications and systems for UAVs should also be tested. This is preferably done as cheap as possible. In 2010 a low-cost commercial UAV appeared on the market, the AR.Drone [15]. This AR.Drone has proved to be useful for research [16]. The AR.Drone has been used as a research platform in [16, 17, 18, 19]. [16] mentions the possibility of embedding software in the drone.

The aim of this thesis is to select a suitable positioning method and design an autonomous control for an AR.Drone. The object recognition and tracking system that will be designed, will be tested on an AR.Drone. An AR.Drone however does not contain a sensor for position determination [20]. There are also no examples of AR.Drones that have autonomous control. Therefore this thesis aims on two main questions:

1. What positioning method should be used for determining the position of a UAV?

2. How can the AR.Drone have autonomous control?

To answer these two questions, in chapter 2 a study will be done to select an appropriate positioning method. In chapter 3 a study will be conducted for a suitable control method for the AR.Drone. In chapter 4 the implementation of both methods will be described. The test results will be presented in chapter 5. Next the test results will be discussed in chapter 6. Finally chapter 7 will give the conclusion of this thesis and give some recommendations for future research.

Chapter 2

Choice for a position determination method

The aim of this chapter is to select an appropriate method for the determination of the position of the AR.Drone. First several methods for determining a location will be described in section 2.1. Next an appropriate method, according to several criteria, will be chosen in section 2.2.

2.1 Possible position determination methods

In this section four different methods to determine a position will be described: GPS(subsection 2.1.1), INS(subsection 2.1.2), GPS with additional sensors(subsection 2.1.3) and stereo vision(subsection 2.1.4).

2.1.1 Global Positioning System

GPS provides positioning, navigation and timing services. Two GPS positioning services are provided: the Precise Positioning Service which is available to the military of the U.S. and its allies and the Standard Positioning Service which is less accurate and provided to civil users. The GPS Standard Positioning Service is defined as follows: "The SPS is a positioning and timing service provided by way of ranging signals broadcast at the GPS L1 frequency. The L1 frequency, transmitted by all satellites, contains a coarse/acquisition (C/A) code ranging signal, with a navigation data message, that is available for peaceful civil, commercial, and scientific use" [21].

GPS is an elaboration of the Global Navigation Satellite System (GNSS) architecture. Besides GPS there are a few more elaborations of the GNSS architecture. Examples are GLONASS , the Russian version of GPS [22], and Galileo, a civil European initiative [23]. A schematic overview of the GNSS architecture can be found in Figure 2.1.

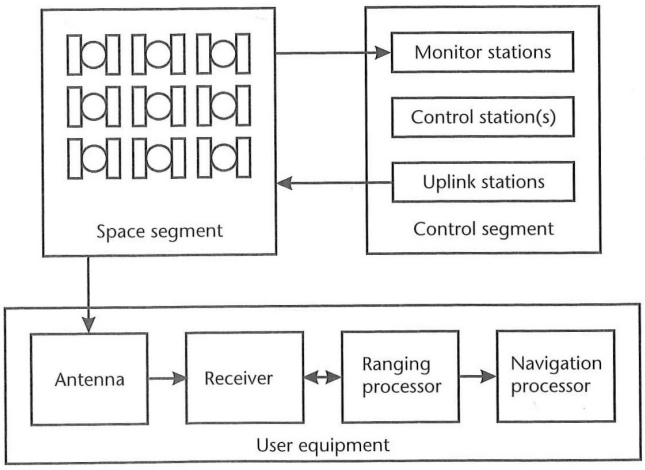


Figure 2.1: GNSS system architecture [24, p. 162]

The GPS system consists of 24 slots where each slot contains at least one operational satellite. A satellite broadcasts binary sequences modulated onto two L band carriers. These two L bands carriers are: L1(1575.42 MHz) and L2 (1222.6 MHz). The L1 band carrier is available for civil use [21]. These broadcasted signals contain ranging codes and navigation data messages. The ranging codes are to determine at what time the received signals were transmitted, while the data message consists of timing parameters and information about the satellite orbits. Next the receiver equipment converts the signals from the satellite into a position, velocity and time [24, p. 162,163].

A GPS receiver needs at least signals from four different satellites to determine its position [24, p. 165]. This position of the GPS receiver is the position relative to earth. The accuracy of GPS depends on the quality of the receiver. In [25] it is shown that the accuracy errors in the horizontal plane are less than five meters 90% of the time which is near the desired accuracy. The vertical height however is less accurate, accuracy is about 12 meters [26] which is not accurate enough for the UAV.

The price of GPS receivers ranges from tens of Euros to thousands of Euros. A weak point of GPS receivers is the loss of signal in areas surrounded by tall buildings [24, p. 166].

2.1.2 Inertial Navigation System

An Inertial Navigation system (INS) is a three-dimensional dead-reckoning system. A dead-reckoning system is a system that measures the change in position, velocity or acceleration. This measurement is added to the previous position to obtain the current position.

The main component of an INS is an inertial measurement unit (IMU) which consists of three orthogonal accelerometers and three gyroscopes (see Figure 2.2). The gyroscopes measure the angular rate which is used to compute the attitude. This attitude is used to correctly compute the acceleration. The accelerometers measure all forces. Integrating the acceleration gives the velocity and integrating the velocity provides the distance travelled. All this information combined with the previous position gives the current position [24, p. 7].

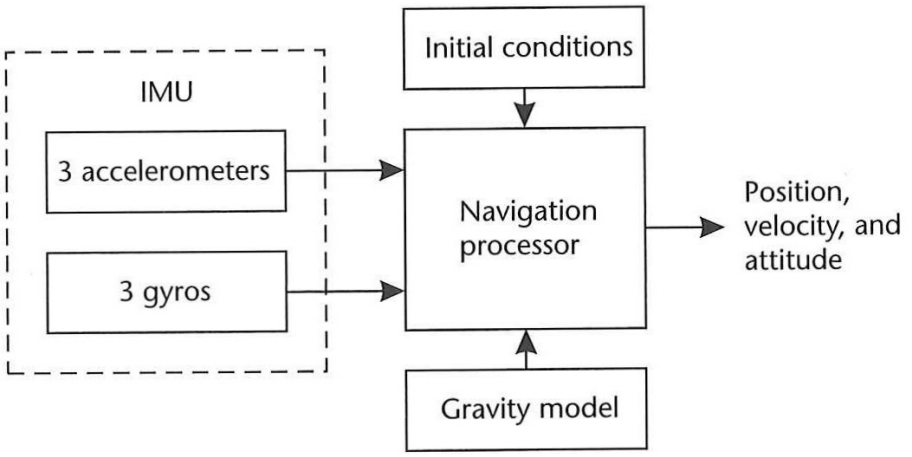


Figure 2.2: Basic schematic of an inertial navigation system [24, p. 7]

In the past most of the IMUs were too expensive or too heavy for the use in UAVs [27]. However, with the development of Micro-Electro-Mechanical Sensor (MEMS) technology the sensor has improved which makes it applicable in UAVs [5].

An INS has two advantages. The first advantage of an INS is that the system does not depend on an external signal. The second advantage is a short-time high accuracy level [7]. A disadvantage is that an INS only determines its location relative to the starting point. The coordinates of the starting point have to be known to determine the absolute position. Another disadvantage is that all errors are summed over time, due to the integration, so the system becomes less accurate over time [24, p. 8].

2.1.3 Global Positioning System with additional sensors

A GPS receiver can be combined with several other sensors to obtain more accurate results. In this subsection two methods will be described. First the aforementioned methods will combined: INS and GPS. Next the combination of a GPS sensor and a barometer will be described.

GPS/INS

GPS/INS combines the long-time high accuracy level of GPS with the short-time high accuracy level of the INS [7]. The combination of GPS and INS is implemented with the use of a Kalman filter. A Kalman filter is an estimation algorithm that uses a stream of measurements to estimate the desired parameters [24, p. 55]. In [4] three methods for coupling a GPS and an INS are described. It appears that the combination of GNS and INS is suited for the use in UAVs according to [6, 27, 28].

The coupling of GPS/INS will provide much better accuracy than GPS or INS alone [6, 9]. The coupling can also provide a backup when the GPS signal is lost. Another advantage is that it is possible to obtain good results with low-cost sensors compared to normal sensors [5, 7, 9, 27, 28]. A disadvantage is that for every coupling a Kalman filter has to be designed. The use of this Kalman filter will also require computing power.

Barometer

A barometer can be used to determine the altitude. Such a barometric altimeter measures the air pressure p_b. The height can then be calculated with (2.1).

$$h_b = \frac{T_s}{k_T}\left[\left(\frac{p_b}{p_s}\right)^{-\frac{Rk_T}{g_0}} - 1\right] + h_s \tag{2.1}$$

Where p_s and T_s are the surface pressure and temperature, h_s is the height at which they are measured. The gas constant $R = 287.1 Jkg^{-1}K^{-1}$, $k_T = 6.5x10^{-3}Km^{-1}$ is the atmospheric temperature gradient and $g_0 = 9.80665ms^{-2}$ is the average surface acceleration due to gravity. (2.1) only applies at heights up to 10.796 km, above this another air temperature is assumed [24, p. 328]. This altitude measurement can be combined with the altitude of the GPS receiver or used instead of the altitude measurement of the GPS receiver. [29] describes the use of an barometric altimeter for determining the altitude of an unmanned helicopter and proposes some improvements for the formula to determine the height.

The advantage of a barometer is a good accuracy, depending on the sensor, as low as one meter [24, p. 328]. Barometers are also affordable which is desirable when there is a small budget. A disadvantage is that there can be errors because of the difference in true and modelled atmospheric temperature and pressure [24, p. 328]. Another disadvantage is the ground effect in rotor driven applications described in [29]. It is expected that the ground effect will be negligible on the position determination system because the sensor will be placed in the body of the UAV.

2.1.4 Vision based system

A stereo vision system can be used to determine the position, attitude and altitude of a UAV [30]. In this system two cameras are, by a special mirror technique, focussed on the same ground plane (see Figure 2.3 for a schematic illustration). With algorithms the position of a point can be determined. By computing the image disparity between two stereo pairs a visible environment can be reconstructed. With this information the attitude and altitude of the UAV can be determined [11]. The position of the UAV is then determined relatively to the starting point. In [10] two cameras are used to recognize landmarks with known coordinates which makes it possible to calculate the absolute position. In [31] is explained that it is possible for a UAV to manoeuvre autonomous using video data for navigation.

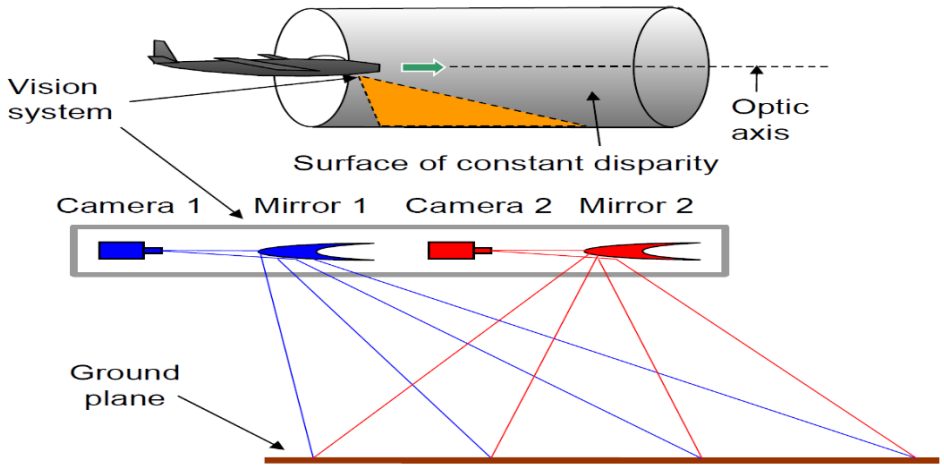


Figure 2.3: Schematic illustration of a stereo vision system, surface of constant disparity, and collision-free cylinder. [11]

Depending on the used cameras a good accuracy can be achieved [11]. An advantage of a vision based system is that the cameras can also be used for object avoidance [11]. A disadvantage is the need for two cameras, which can be expensive. Another disadvantage is the need to process all images and the need for algorithms to compute the relative position which may require considerable computing power. A third disadvantage is that weather conditions may influence or disable the system.

2.2 Consideration of alternatives

2.2.1 Criteria

There are nine criteria which the positioning method should meet. These criteria will be explained in this subsection, most of them will be based on the design brief (Appendix A).

Commercial of the shelf

First of all the components should be commercial of the shelf (COTS) available. Preferably the components should work right out of the box except for soldering wires. This is because there is no need and time to design a new positioning system.

Complementary

The positioning method should be complementary to the sensors that are already on the AR.Drone [20]. A positioning method is installed because the AR.Drone lacks sensors for determining the data that is needed by the tracking system and the ground station.

Accuracy

Based on the design brief an accuracy of 2 meters in horizontal direction and 1 m in vertical direction should be achieved. The AR.Drone has got an ultrasound altimeter [20] with a range of 6 meters. Because the system should be able to operate on heights over 6 meters another positioning method for the altitude is needed.

Absolute positioning

The position of the UAV should be known relative to Earth. This is important for displaying the position of the UAV on a map. It is also important because it should be able to return to base, which is easier with absolute positioning.

Weight and size

The whole electrical system should be less than 250 grams according to the Aerospace Engineering group. The positioning system should not weigh more than 15 grams because that is feasible and then there is a larger margin on the weight of other components. Because there is little space in UAVs a small positioning system is preferred.

Update rate

The final UAV will be flying at a speed of about 30 ms^{-1}. To give a good status of the position an update rate of at least 5 Hz is required. When flying in a straight line this corresponds to a distance of 6 meters covered before a new update is available.

Power

The whole system should be able to operate one hour on a battery that can be fitted in a UAV. Therefore it is required that the positioning system should consume not more than 200mW. The higher the power consumption of all components together is, the bigger battery is needed.

Vibrations

The positioning system should not be affected in its operations by vibrations. This means that solder does not release and that connectors do not disconnect due to vibrations of the aircraft's propulsion.

Price

All components that are ordered, are ordered on the budget of the team and the TU Delft. Therefore the expenses should be kept reasonable. For that reason the positioning system should cost no more than €80,00.

2.2.2 Consideration

In this paragraph an overview will be given of the methods mentioned in section 2.1 and the criteria in subsection 2.2.1 are met. This will be done with the help of Table 2.1. The information in Table 2.1 is based on the articles of the previously mentioned sources. Based on Table 2.1, Table 2.2 is created which summarizes the technical data in pluses and minuses. The scores will range from $--$ (meaning very undesirable) to $++$ (meaning very desirable).

Table 2.1: Overview of positioning methods and criteria

Criteria	GPS	INS	GPS/INS	GPS/Baro	Vision based
COTS	yes	yes	not available as a module	yes	not available as a module
Complementary	yes	already onboard	partly, INS is onboard	yes	yes
Accuracy	h[a]: < 5m, v[b]: 12m	decreases with time	h: < 1m, v: < 2m	h: < 5m, v: < 3m	h: <2m, v: <2m
Absolute positioning	yes	no	yes	yes	no
Weight	<12 grams including antenna	0 grams	< 15 grams	<15 grams	depends on the used components
Update rate	up to 10 Hz	50 Hz	GPS up to 10 Hz, INS up to 50 Hz	GPS up to 10 Hz, Baro up to 128 Hz	25 Hz
Power	100 mW	1 mW	100 mW	100 mW	depends on the used cameras
Vibrations	no influence	needs filtering	no influence	no influence	might affect the images
Price	<€ 60,00	<€ 35,00	<€ 95,00	<€ 80,00	<€ 40,00 + the price of a suitable microprocessor

[a] horizontal
[b] vertical

Table 2.2: Positioning methods rating

Criteria	GPS	INS	GPS/INS	GPS/Baro	Vision based
COTS	++	++	−−	++	−−
Complementary	++	0	+	++	++
Accuracy	−−	−	++	+	++
Absolute positioning	++	−−	++	++	−−
Weight	+	++	+	+	−
Update rate	+	++	+	+	++
Power	0	++	0	0	−−
Vibrations	++	0	++	++	−
Price	+	++	−	0	0

2.2.3 Final choice: Global Positioning System/barometer

After considering all alternatives it was decided to purchase the combination of the GPS and the barometer. This might not be the best choice in terms of accuracy but it is in terms of implementation and needed additional resources. According to Table 2.2 GPS/Baro is overall the best solution.

The vision based method would be very hard to implement and would require extra processing power because all images have to be processed. The combination of GPS/INS would be the best solution for a reasonable price if there are ready to use modules. These ready to use modules are however hard to find and really expensive and therefore not an option. Implementing a GPS and INS combination ourselves is also not an option because it would require too much time. Because the very rapid degradation of the accuracy of an INS sensor this method is not very suitable for positioning use. Furthermore an INS sensor would deliver the relative position which is not ideal. A GPS sensor is very inaccurate in the altitude which is important data for a UAV.

Therefore we decided to choose the GPS sensor combined with a barometer. This gives acceptable accuracy in horizontal and vertical direction at a reasonable price.

Chapter 3

AR.Drone control

The aim of this chapter is to select an appropriate implementation for the control of the AR.Drone. Therefore in section 3.1 a control model of the AR.Drone will be explained. Subsequently in section 3.2 a study is conducted on the different options for the hardware to control the AR.Drone. Finally in section 3.3 these options are compared and the best option is chosen.

3.1 Control loops

In this section a control model for the AR.Drone will be studied. Therefore first some background information on autonomous control of UAVs will be given. Next the control model of the AR.Drone will be derived.

3.1.1 Layered model

In [32] a layered model to control autonomous UAVs is described. The first layer is the kernel control, responsible for asymptotic stability, the second layer is the command generator, responsible for generating flight commands for the kernel control, and the highest level is the flight scheduling. The function of the flight scheduler is to generate a flight plan, flight tasks and references.

The AR.Drone has the first layer implemented. The drone receives flight commands over a Wi-Fi connection and performs them. The angle stabilization and vertical speed are controlled by the software running on the drone. However the AR.Drone does not have the second and the third layer implemented. If this layer structure is to be used to make the AR.Drone autonomous, the second and third layer have to be implemented. In subsection 3.1.2 the design of the second layer will be described. The third layer, in this project, is partially human control and partially commands generated by the target-tracking system.

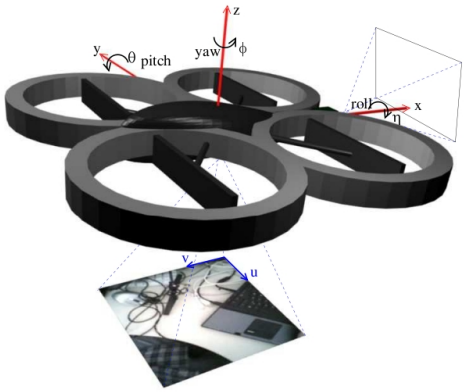


Figure 3.1: The directions and turning angles of the AR.Drone. Figure from [16]. The orientation of the drone is defined by its camera view. At the bottom of the picture the view of the bottom camera is displayed while at the right the view of the front camera is displayed.

3.1.2 AR.Drone control model

As indicated in subsection 3.1.1 the second and third layer of the layered model are not implemented in the AR.Drone. The third layer is designed in combination with [33]. The design of the second layer will be explained in this subsection.

The second layer, also called the command generator, receives input from a higher layer (flight planner), and send commands to the lower layer (kernel control) to reach the destination given by the flight planner [32]. The kernel control of the AR.Drone is different from the kernel control developed in [32].

As the command generator has to generate commands that should be executed by the lower layer, the second layer will also be different. In this layer structure, the controllers explained in [16] perform the function desired for the command generator.

To design the controllers of the command generator, it is necessary to know more about the system dynamics and the control variables. [16] describes how to control the drone with four different control loops. The first is for position x, the second for position y, the third for z and the last one is for the yaw. Figure 3.1 demonstrates this coordinate system relative to the UAV.

The control variables for the AR.Drone are $roll$, $pitch$, yaw and vertical speed (gaz) . To control the x and y position, it is necessary to control the $pitch$ and the $roll$ respectively [16], disabling the use of yaw at the same time.

Before the drone's controller can be designed, a dynamic model has to be made, and the parameters of the model have to be determined. For the simplicity of the model, the *yaw* will not be used together with *pitch* and *roll*. This is because the *yaw* influences the forward and sidewards movements. This means the drone will first reach its desired x, y, z coordinates and then *yaw* is used to turn the drone to the desired orientation.

This way there are 3 completely separated control loops. One for the altitude, one for the forward/backward direction and one for the left/right direction. The resulting model, adapted from the model [16, Fig. 4], can be seen in Figure 3.2.

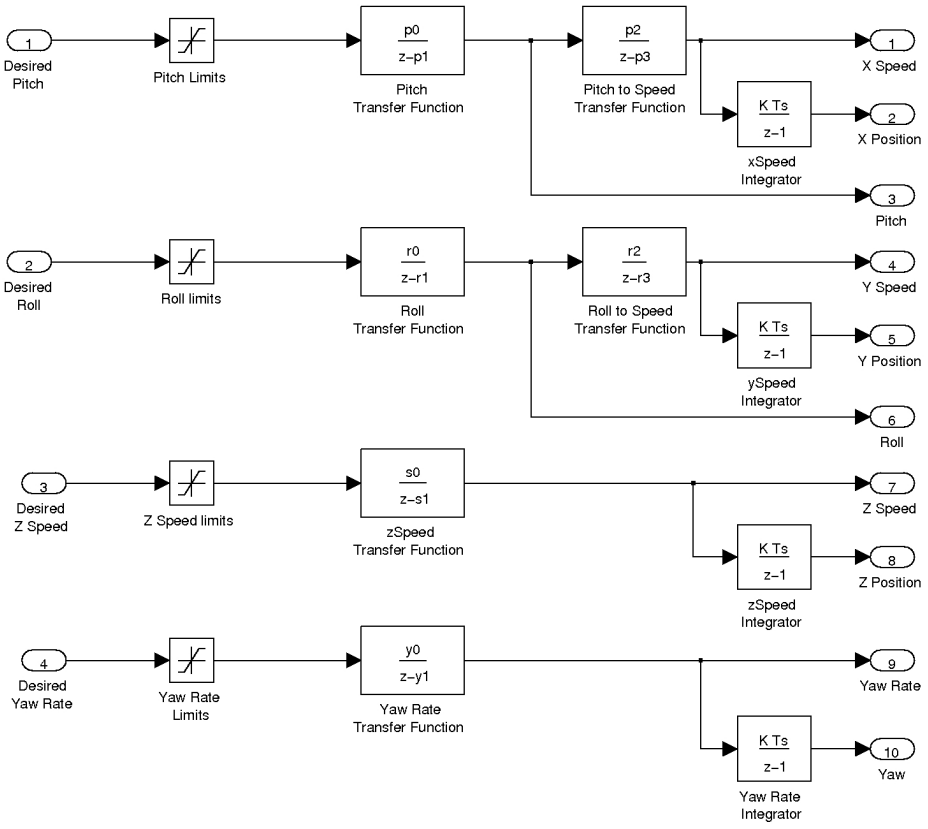


Figure 3.2: Model of the Parrot AR.Drone with input variables and the resulting outputs. Adapted from [16, Fig. 4].

The control parameters can be determined empirically by sending commands to the drone, and collecting the data from its sensors, or measuring the effects by other means. After that the model can be fitted, using least squares [16].

If the models are valid, it is possible to start designing the controllers.

The x controller is modified from the controller designed by [16]. The input of their controller is a relative x position. The calculation of the relative distance is included in the controller design. The resulting design can be found in Figure 3.3. The same controller can be used for the y direction, since the dynamics are similar. Of course, the parameters should be adjusted to the model for both controllers.

The *gaz* and *yaw* controllers can be used from [16, Fig. 6]. For these controllers the only necessary step is to adjust the model parameters to the drone. For clarity, the controllers are reproduced in Figure 3.4.

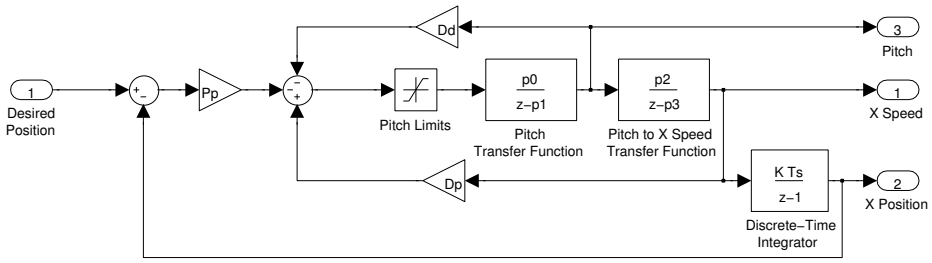


Figure 3.3: Model of the Parrot AR.Drone's *pitch* controller. Adapted from [16, Fig. 7].

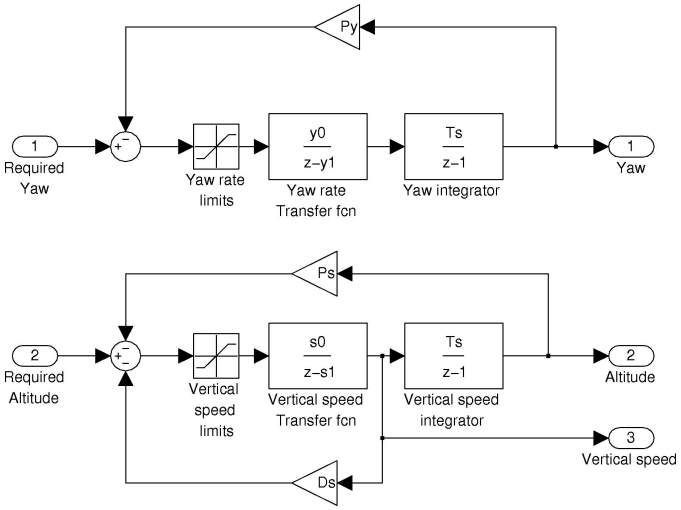


Figure 3.4: Model of the Parrot AR.Drone's *yaw* and *gaz* controller[16, Fig. 6].

3.2 Control options

In this section three hardware options to run the model of subsection 3.1.2 and the third layer controller are explained. In section 3.3 a choice is made for the most appropriate option.

3.2.1 Remote control

The most used method to control the AR.Drone in research projects is to use a ground station computer which connects to the drone. This computer receives navigation data (accelerometers and gyroscopes values) and camera images from the UAV, performs the necessary calculations and sends commands to the UAV to perform the desired action.

This option limits the autonomy of the UAV to a specific range around the ground station, due to the limited range of the wireless network. Especially when flying outdoors this is not desirable.

A big advantage of this method is that no extra power is used from the drone's battery. There are no extra boards connected to the drone consuming energy and there is no extra weight for the drone to carry. The battery performance would be the same as when the drone is used in a normal way.

Another advantage is the processing power. The computer functioning as ground station can have much more processing power than most embedded systems. And, if more processing power is necessary, it is easy to change the computer by one with more processing capabilities.

Developing the software of a remote control system can be done on a PC using Software Development Kits (SDK) like QtCreator or Eclipse. As the software runs on the computer, its libraries dependencies are easier to meet. Another advantage is that the software does not need to be uploaded to the AR.Drone every time, making it easier and faster to debug.

3.2.2 Onboard without extra hardware

One of the options described by [16] is to cross compile software to run on the AR.Drone. Their work does not mention if this was successfully achieved or not. In theory it should be possible, since the AR.Drone uses an ARM9 based processor and runs a BusyBox[1] based Linux distribution. The step of uploading software and running can be done through ftp (uploading) and telnet (starting it up).

One of the problems with this option, also mentioned by [16], is that the power processing of the embedded CPU is limited, and an extra process could interfere with the processing speed of the stabilization control loops, which could lead to crashes.

[1] www.busybox.net

Power consumption in this option is not a problem. As more CPU power is necessary the power consumption of the embedded computer would increase. Relative to the motors, this extra consumption is negligible. No extra weight is added to the drone so it will not need more power to fly.

The range is not a problem like in the previous option, except if the third control layer is still on the ground station, and the communication is done through the drone's wireless network. An option to increase the range is being developed by [34]. This option requires more power, but the power consumption of the communication system is outside the scope of this project.

The software that has to run on the AR.Drone has to be cross compiled and uploaded to the UAV using ftp. This makes debugging slower, and consequently slows down the development speed.

3.2.3 Onboard with extra hardware

The last option is to control the drone with extra embedded hardware. This can be done in two different ways, which differ from each other in how they communicate with the AR.Drone. In both ways, a small system such as a BeagleBoard [35] or a Raspberry-Pi [36] is placed in the drone and performs the calculations. The first communication option is to use a usb-wireless adapter on the development boards, so that they can connect to the wireless network of the drone to communicate. The second option is to use the UART or the USB options available at the drone board, and run a small program that converts the communication into network packets to control the drone.

The power consumption of this alternative is higher than the first two options. The main reason is that the extra hardware also means extra load to the UAV, and the extra hardware itself also consumes energy. An extra battery could be added to feed the external CPU, but this would mean more weight to carry, and more power is consumed by the drone itself.

For this option the range only depends on the drone's battery capacity. Because the drone has to carry more weight in this option, the range is lower than using only on board hardware. With more CPU power it would also be easier to implement the third layer controller. If communication is still necessary, another communication link could be used instead of the Wi-Fi network of the drone, so that the range is not limited to the Wi-Fi range.

In this option, there is at least as much work as the option to embed software in the drone itself if it is chosen to communicate with the AR.Drone through the UART interface. If a wireless adapter is used for communication it is not necessary to embed software in the AR.Drone.

3.3 Consideration of alternatives

3.3.1 Criteria

One of the options has to be chosen to be implemented. This is done by comparing them according to five criteria, listed in this section. The importance of each criterion is also explained.

Price

The choice for an AR.Drone as research platform is influenced by its relatively low price. A low cost controller option is desired as well. If extra hardware is necessary, it should not cost more than €200.

Power consumption

The AR.Drone already has a limited flying time due to the limitations of its battery. A higher power consumption will result in even less flying time for the drone, limiting its range. Therefore the power consumption of the control system itself should not be more than 5 Watt.

Reachable range

A larger range means the drone can fly farther from its take-off site. A minimum range of 1000 meters is desired, so that the drone could cover an larger area when performing autonomous identification, and tracking of a target.

Computational power

The computational power should be at least enough to handle the four control loops from the second (command generation) layer. Preferably the solution should be able to handle the third layer as well. This layer will include video processing.

Easy implementation

The solution should not be very complex and difficult to implement, as the time available for development is limited.

3.3.2 Consideration

The three alternatives are evaluated according to the previously mentioned criteria in Table 3.1. The information in the table is extracted from section 3.2. The scores will range from $--$ (meaning very undesirable) to $++$ (meaning very desirable).

Table 3.1: Overview of control hardware and criteria

Criteria	Remote Control	Without extra hardware	Extra onboard hardware
Price[a]	++	++	R.P. [b]: − B.B. [c]: −−
Extra Power Consumption	++	++	–
Range	−−	++	+
Computational Power	++	−	+
Easy Implementation	++	−	−

[a]Assuming laptops and other development tools are available
[b]Raspberry-Pi Made available after requesting it for educational reasons, costs €38,00
[c]Beagle Board: Available at Farnell, costs €160,00

3.3.3 Final choice: onboard with extra hardware

The choice is not only based on the criteria previously mentioned, but also on the necessities of other parts of this project, like the video processing and the communication link with a ground station.

To implement the second layer the choice is to embed software in the drone's hardware and communicate using the serial port (UART). The third layer also requires video processing and is being developed together with another group. This layer will be implemented on extra hardware on the AR.Drone. The chosen hardware platform is a BeagleBoard since it fits better in the requirements for video processing [33].

Chapter 4

Implementation

In chapter 2 and chapter 3 a positioning method and a control method were chosen. This chapter will describe which components were purchased and how the systems are implemented. In chapter 5 the test results of the implementation can be found. This chapter will first give some information on the used programming language. Next the implementation of the GPS sensor and the barometer are described. Finally it will describe the implementation of the control of the AR.Drone.

4.1 Software Development

4.1.1 Requirements

Developing software is necessary to make the control loops and read the sensors. In [37], software requirements for unmanned vehicles are stated. There a layered model is proposed, from hardware to remote user interface, with an execution and a service agent layer in between. The priority of the tasks that the system should perform are:

1. Sensor readings and motor control for stabilization
2. Position control loops
3. Flight tasks calculations

These tasks can be closely related to the control layers described in [32]. The first layer requires hard real-time programming, but is already implemented in the AR.Drone. If this layer is really implemented as a hard real-time system is unknown. The second layer has firm real-time requirements. That means that some of the tasks can be delayed without having consequences it should however not happen frequently. In the case of the control loops that position the AR.Drone, the task should be performed at least once each 30 ms. The results of not meeting this requirement is that

the drone will start hovering, but if this only happens once in 10 times, the effects are not really notable. And finally on the third layer a soft real-time processing is desirable. That means the quality of the service is affected by tasks that are not performed on time. For a moving target detection system this could cause the system to loose track of the target, which makes it necessary to start searching again.

Another requirement added to the software of this project, is how easy it is to develop the software. The main reason is the limited time available and people working on it.

4.1.2 Choice: Qt-framework

The chosen development platform was the Qt-framework, using C++ as programming language. The choice was based in the experience of one of the team members with the framework.

The Qt-framework is a cross-platform C++ framework currently maintained by Nokia, available under open source licenses like the GNU GPL version 3. One of the key advantage of the Qt-framework for this project is the availability of libraries which are easy to use on different platforms. So functions can be first debugged on the laptops of the team members. This reduces the debugging time necessary on the embedded system.

Another option was to use Java, but none of the team members had experience with Java and serial port interfaces. Besides that, the AR.Drone does not have a Java Virtual machine running, making it difficult to run Java programs on it.

4.2 Implementation of GPS/Barometer

4.2.1 GPS sensor

Purchase

The desired GPS sensor had a number of requirements. The first requirement was that it should have an update rate of at least 5 Hz as stated in subsection 2.2.1. The second requirement was that it should have an accuracy of 2 meters horizontally. The third requirement was that it should have a low weight (<12 grams). The fourth requirement was that it should be available at Farnell or RS Components because the TU Delft is able to order there and these companies deliver next business day. Otherwise it would have to be possible to order this component in the Netherlands because the delivery times from other countries would become too long. The last requirement was that the GPS sensor should cost less than €60,00 because we are ordering on the budget of the TU Delft.

With the above mentioned requirements a GPS sensor was selected at RS Components: the Fastrax UP501. The features of the GPS sensor are

summarized in Table 4.1 and an image can be found in Figure 4.1. The complete datasheet can be found in [38].

Table 4.1: Key features of Fastrax UP501

Feature	
Update rate	up to 10 Hz
Accuracy	1.8 m
Power consumption	75 mW typ.
Dimensions	22mm x 22mm x 8mm
Weight	9 g
Price	€ 43,00

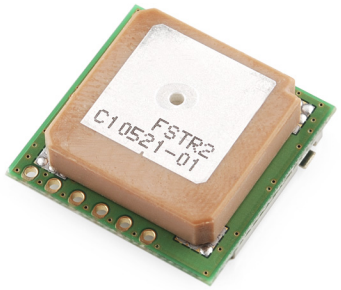


Figure 4.1: Fastrax UP501 GPS receiver

source: `http://www.sparkfun.com`

Software

The GPS has a UART interface to communicate. This UART interface is used to set up the GPS sensor and to receive the information from the GPS sensor. The GPS sensor uses the NMEA-183 standard for communication between the sensor and the UART interface. In [39] the NMEA manual of the GPS sensor can be found. With the use of this standard it is possible to set certain settings like baud rate, update rate and desired messages. It also describes the message formats the sensor outputs.

An appropriate output message had to be selected. In [39] the 19 available message formats are mentioned. The message format should at least output the GPS coordinates. This can be done with the NMEA_SEN_GLL which just gives you the latitude, longitude and time. The NMEA_SEN_RMC however is the recommended minimum specific GPS sentence . This format

will give as extra information the speed over ground, GPS signal availability and date. Especially the navigation validation is important to check, in that way it can be checked if the information is valid. The RMC message will be requested from the GPS sensor. The format of this message can be found in Appendix C.

The serial port to which the GPS sensor is connected, is read out using a Qt library named *qtextserial*. This library enables it to read the serial port. The read out information is stored in a buffer which can be read out with software. This buffer is read out character by character. All RMC messages start with a '$' and end with a '\n'. In this way a complete message can be easily detected. The detected message is sent to a *gpshandler* which checks if the message is of the RMC format. If the message is of the RMC format, the checksum of the message is calculated. The checksum is calculated by performing a XOR operation on all characters between the $ and the * . When the calculated checksum equals the transmitted checksum, the message is split and all wanted information is stored in a data object *GPSData*.

While the data is stored in *GPSData*, the latitude and longitude coordinates are converted from the NMEA format to the, by Google Maps used, WGS84 format. This is done to facilitate the use and transmission of GPS data to the ground-station.

The NMEA format uses the degree minute representation of coordinates (DDMM.mmmm) while WGS84 uses the decimal representation. The conversion is done by dividing the minutes of NMEA by 60 and adding it to the degrees. So WGS84 is $DD + MM.mmmm/60$. The *GPSData* object can be called to get certain data like latitude and longitude. While the data object is called the North/South (East/West respectively) is used to make the latitude (longitude) positive or negative. North (East) is converted to positive while South (West) is converted to negative.

4.2.2 Barometer

Purchase

For purchasing a barometer there were two important requirements. The first requirement was that it should have an accuracy of 1 meter vertically. The second requirement is that is should be available, in the Netherlands, as an out of the box working module. According to these requirements, the Sparkfun BMP085 Breakout was selected. Partly because this is one of the few available modules and because it meets the requirements. This module uses a Bosch Sensortec BMP085 barometer to measure the temperature and pressure. A short overview of the main features is provided in Table 4.2, more specifications can be found in [40]. An image of the barometer can be found in Figure 4.2.

Table 4.2: Key features of Bosch Sensortec BMP085

Feature	
Pressure range	300-1100 hPa
Accuracy	0.2 hPa ˜1.7m
Power consumption	25 μ W
Dimensions	16.5mm x 16.5mm x 3mm
Weight	1 g
Price	€ 17,00

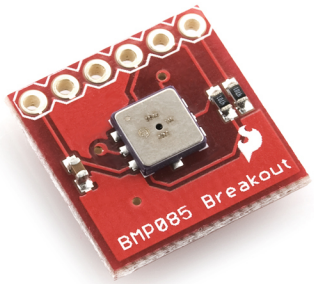


Figure 4.2: Sensortec BMP085 barometer breakout board

source: `http://www.sparkfun.com`

Software

The barometer uses an I^2C interface to send and receive. With this interface it is able to read and write the memory of the E^2PROM of the sensor. In this memory 11 calibration values are stored. The sensor is tested in the factory and then these calibration values are stored to be used to compensate offset, temperature dependence and other parameters. The E^2PROM also contains the uncompensated temperature and pressure measurement.

To be able to communicate using the I^2C, a FTDI HT4232 module was used. The Qt framework has no support for I^2C implemented. The communication is handled in a low level, using the FTDI library.

To calculate the compensated temperature and pressure, the procedure described in [40] is used. First all calibration values are read and stored. Next the uncompensated temperature and pressure are requested, measured and stored. Then these values are all combined to calculate the compensated temperature and pressure. Finally a function is built to calculate the altitude relative to the ground station. Therefore the temperature and pressure at the ground have to be stored at initialization. (2.1) is used to calculate the

altitude whereby T_s and p_s are the temperature and the pressure at the ground station respectively.

At intermediate tests it appeared that the barometer had big spikes in the measurement of the altitude (see Figure 4.3). Therefore a moving average filter was implemented to smooth the results. The last 10 altitude measurements are stored in a array. When the altitude is requested the last 10 measurements are averaged and this average is returned.

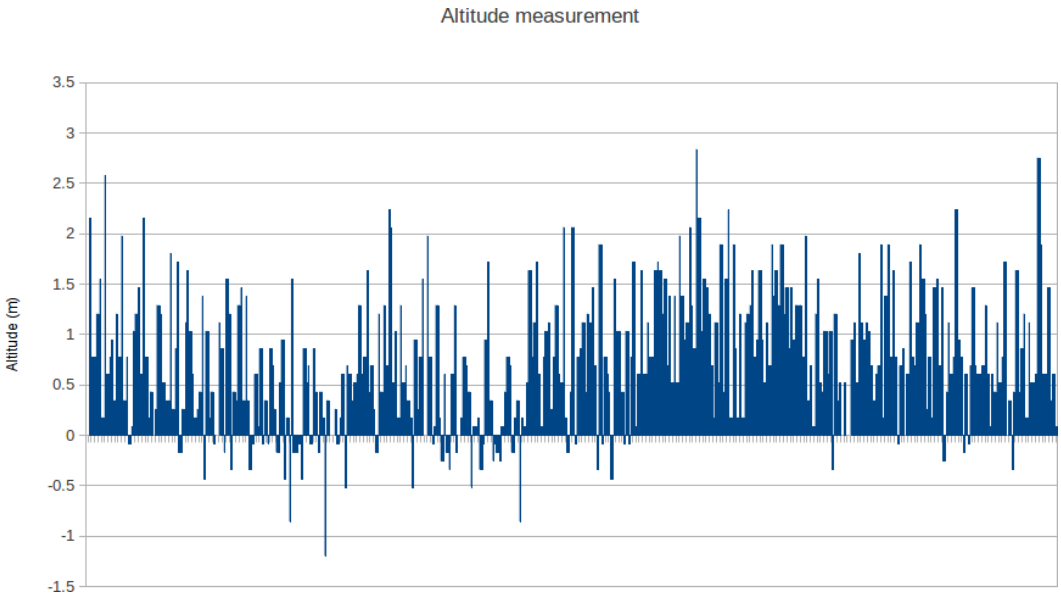


Figure 4.3: Altitude measured for 60 seconds without using a filter.

4.3 Implementation of AR.Drone control

As explained in section 3.3, the control of the drone is implemented partially on the drone itself, and partially on the extra hardware available. The communication between the two boards is done through the UART interface of the AR.Drone and the HT4232 module [41] connected to the Beagleboard. The applications will be explained in subsection 4.3.2 and subsection 4.3.3.

In the first subsection the implementation of the application developed to model the drone will be explained. This application is not directly a part of the drone control software, but it is a step to design the control loops.

4.3.1 AR.Drone Modelling

To model the drone, a computer with wireless networking is necessary to send flight commands and read the data that comes from the AR.Drone.

In [16] a lot of information about the communication with the AR.Drone can be found. All the communication is based on UDP packets. The Nav-Data packets sent by the drone contain flight information, while the AT Command packets contain the commands sent to the drone. In order to simultaneously send and receive data packets, the receive and send actions are run in separated threads.

Another thread handles the user input. The program is command-line, and the user has to type the commands. This is not very user-friendly, but enough for the few times it has to be used to model the drone.

When performing a model process, the data is stored in a file, and this file is then imported in Matlab, where the data can be fitted.

4.3.2 AR.Drone Software

Control loops

The most important part of the software are the controller loops. The controller should work in two modes: the GPS mode, or relative mode. The GPS mode can control the drone to fly to a specific GPS coordinate, while the relative mode sets a position relative to the current drone position. The first mode makes it possible to fly back to the base, or fly to a specific area where a task should be performed. The second mode can be used to follow a target given its relative position to the UAV. At the current implementation, the GPS mode has been disabled, and only the relative control is used. The choice to do that is based on the time available to implement the control loops.

To implement the control loops described in subsection 3.1.2, the gain parameters had to be derived. This is done by writing the transfer functions in a state space design. From this state space design the gain parameters could be calculated [42, ch 7]. The state space design of *pitch* and *roll* is the same except for the used parameters and can be found in (4.1).

$$\begin{bmatrix} v \\ \dot{v} \\ \dot{pitch} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & p_3 - 1 & p_2 \\ 0 & 0 & p_1 - 1 \end{bmatrix} \begin{bmatrix} x \\ v \\ pitch \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ p_0 \end{bmatrix} pitch_{desired} \tag{4.1}$$

The state space design of gaz can be found in (4.2) while the state space design of yaw can be found in (4.3). The values of p, r, g and y can be found in Table 5.1

$$\begin{bmatrix} gaz \\ \dot{gaz} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & g_0 - 1 \end{bmatrix} \begin{bmatrix} alt \\ gaz \end{bmatrix} + \begin{bmatrix} 0 \\ g_1 \end{bmatrix} yaw_{desired} \tag{4.2}$$

$$\begin{bmatrix} yaw_{rate} \\ \dot{yaw_{rate}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & y_0 - 1 \end{bmatrix} \begin{bmatrix} yaw \\ yaw_{rate} \end{bmatrix} + \begin{bmatrix} 0 \\ y_1 \end{bmatrix} yaw_{desired} \tag{4.3}$$

The gains for the closed loop controller can then be determined by pole placement [42, ch 7]. First it is necessary to determine the desired pole locations. Knowing the desired poles and knowing the system, it is possible to determine a matrix K such that $u = -Kx$.

The calculation of K can be done using Matlab's functions *place* or *acker*. As *place* does not allow poles with a higher multiplicity than one, *acker* is more suitable.

Communication

Inside the AR.Drone, the developed software is responsible for the control-loops that control the position of the UAV. As explained in subsection 3.1.2, the drone already has the control loops implemented that control the stabilization. The communication between the two control layers will be done using the loop-back network interface, using the same protocol as used by the AR.Drone through a network.

The software implemented to run on the drone is also responsible for handling communication between the drone and the BeagleBoard. This communication is wired, through a UART interface available at the drone. This interface is intended initially to debug software running on the AR.Drone, but changing some settings allows it to be used for the desired communication.

The communication is bidirectional. The software receives targets where to fly, or commands to land, take off or emergency stop, and sends navigation data (speed, angles, altitude, battery status). The read and write processes are asynchronous. This avoids blocking the software to read data, but could reduce the overall system performance. If the software still meets the requirements for soft real-time processing, this reduction in performance is not a problem.

4.3.3 Beagleboard Software

The Beagleboard contains the third-level controller, as described in subsection 3.1.1. In this software the image processing algorithm from [33] is implemented together with ground-station communication and flight planning.

To handle communication, video processing and flight-planning, more than one thread is used. Each of the processes runs on a different thread. The flight planner uses information coming from the GPS sensor, the barometer, the ground-station, the video processing algorithm and the AR.Drone itself to make the planning. The logic behind it is not very sophisticated, because of the limited time to make it work.

Chapter 5

Tests

This chapter describes the testing methods, results and discussion of the in chapter 4 implemented hardware and software. In section 5.1 the GPS Sensor was tested, next in section 5.2 the barometer was tested, in section 5.3 the AR.Drone's behaviour model was tested and finally in section 5.4 the control loops were tested.

5.1 GPS sensor

5.1.1 Test methods

The GPS sensor was tested outside because it is very hard to gain a signal fix inside and the final system is used outside. Four methods were used to test the GPS sensor.

The first method was to place the GPS sensor in one spot and log its coordinates for two minutes. This test was done to determine the accuracy of the GPS sensor. This test was repeated at four different spots.

The second method to test the GPS sensor was to measure the GPS coordinates at a point, walk around the block and return at exactly the same point. This was done while logging the GPS coordinates.

The third method to test the GPS sensor was by measuring 10 meters and walk this distance. At the origin and at the destination the coordinates were logged. This test was repeated by walking 10 meters in exactly the opposite direction and logging the coordinates again. This measurement was conducted to determine if the distance between two opposite spots is the same.

The fourth method to test the GPS sensor was by logging the GPS coordinates while driving through a town. This was done with an average speed of 35 km/h and a maximum speed of 60 km/h.

5.1.2 Test results

Method 1

The results of the GPS sensor placed in one spot were plotted in Matlab. A scatter plot was used to show the scatter in coordinates. The results of the measurement at spot 1 can be found in Figure 5.1. The results of the other spots can be found in Appendix D.

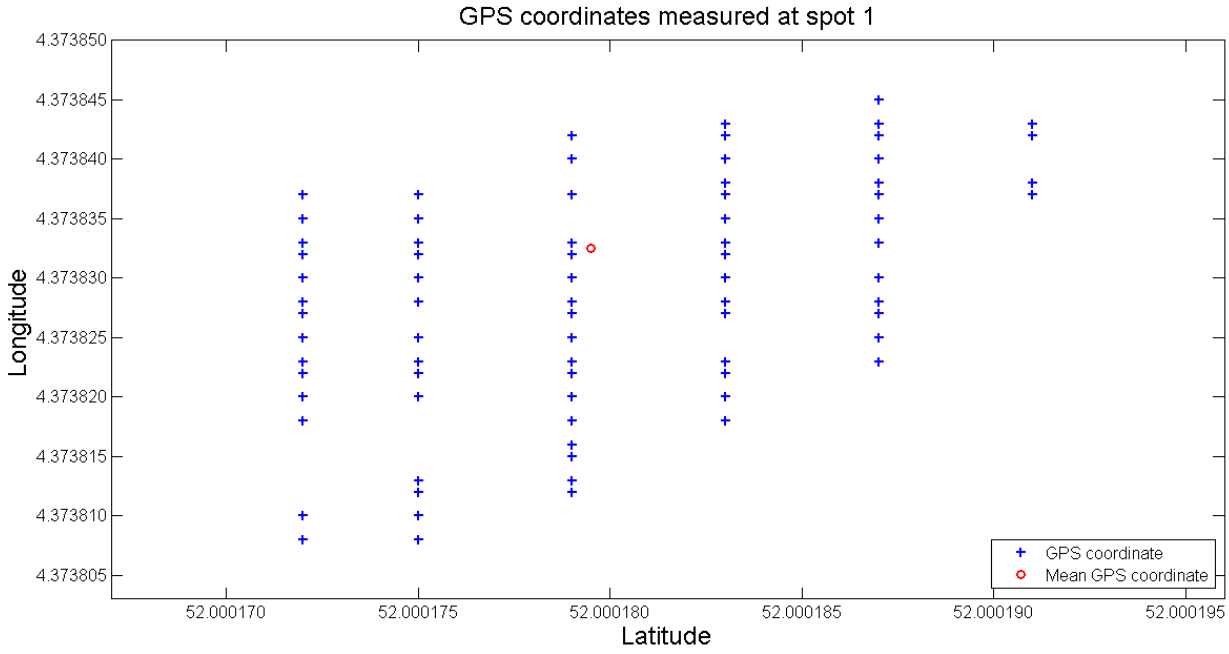


Figure 5.1: GPS coordinates measured at spot 1

A difference of $5.0x10^{-}6$ on the horizontal axis corresponds to a distance of 0.56 meters, a difference of $5.0x10^{-}6$ on the vertical axis corresponds to a distance of 0.34 meters. This is calculated with the use of [43].

Method 2

From this measurement the start coordinates were: 52.044898;4.570319 while the end coordinates were: 52.044868;4.570322. The difference between these points is 3.3 meters [43].

Method 3

The average coordinates of the middle point were: 51.999432;4.374242. The average coordinates of the left-hand and right-hand point were respectively

51.999506;4.374162 and 51.999366;4.374342. The distance between the middle point and the left-hand point is 9.9 meter while the distance between the middle point and the right-hand point is 10.0 meters [43].

Method 4

The result of test method 4 is shown in Figure 5.2. The starting point is indicated with a red arrow and the driving direction with arrows along the path.

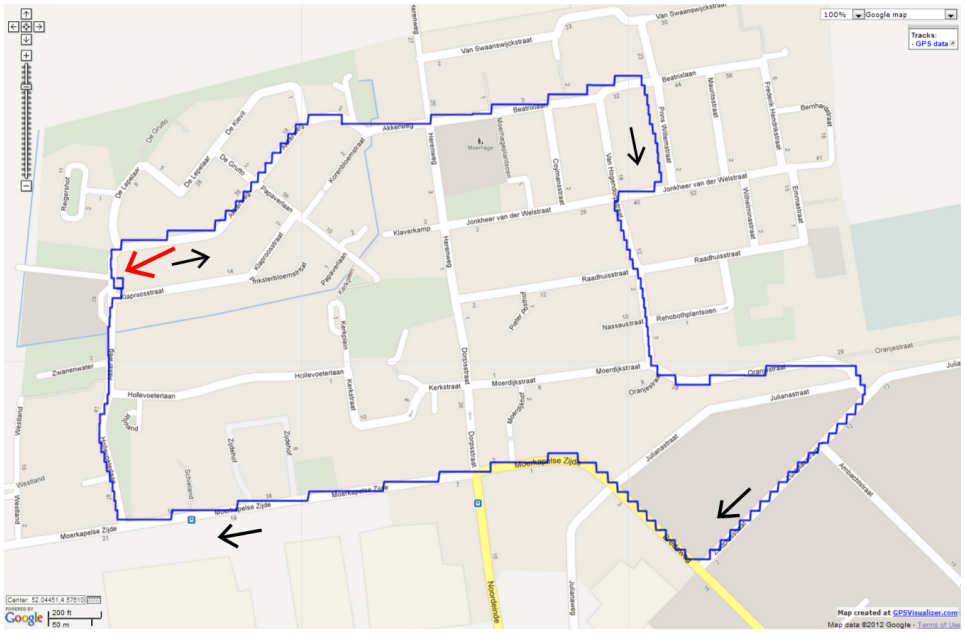


Figure 5.2: GPS coordinates measured while driving through Moerkapelle

5.1.3 Discussion

From section 5.1.2 and Appendix D it could be concluded that there is an average scatter of about 3 meters in both the longitude direction and the latitude direction.

In Figure D.2 a lot of scatter was seen, this could be explained by the fact that this spot was 1 meter from a building so the building partly might have blocked the GPS signal. The conclusion from this is that near buildings the inaccuracy increases.

From section 5.1.2 it could be derived that moving and then returning to the same point has no influence on the accuracy.

In section 5.1.2 the distances that were measured by the GPS sensor and by hand were, except for a small margin, the same. It should be noted that

the average coordinates are used which gives better results than comparing just three different coordinates. However the conclusion is that the GPS sensor is able to measure a difference of 10 meters very well.

In section 5.1.2 the GPS track roughly agreed with the driven track. There is some inaccuracy but this is within a margin of 5 meters. This could partly be explained by the fact that the output was rounded to four or five decimal places instead of six. The offsets could also be explained by this rounding error. The rounding error had been corrected in the other tests however there was no time to repeat test method 4.

The promised accuracy of 1.8 meters ([38]) was not achieved. The design specification of 2.0 meters accuracy (Appendix A) is also not achieved. However the specification in subsection 2.2.2 for GPS is achieved.

Part of the inaccuracy can be explained by the fact that all measurements were done in a not completely open area. The surrounding buildings may have been blocking the signal. As we saw in section 5.1.2, using an average coordinate gives better results. Therefore it is recommended to use an averaging filter before using the coordinates.

5.2 Barometer

5.2.1 Test methods

The barometer was tested both inside and outside. Two test methods were performed, both inside and outside.

The first method was to place the barometer at a flat surface and start logging the altitude for two minutes. This was done at four different spots outside and two different spots inside.

The second method is related to the first method, logging was started at ground level and then the barometer was moved to a higher level of which the height was measured. Inside the barometer was moved to a height of 1.40 meter while outside the barometer was moved to a height of 1.50 meter.

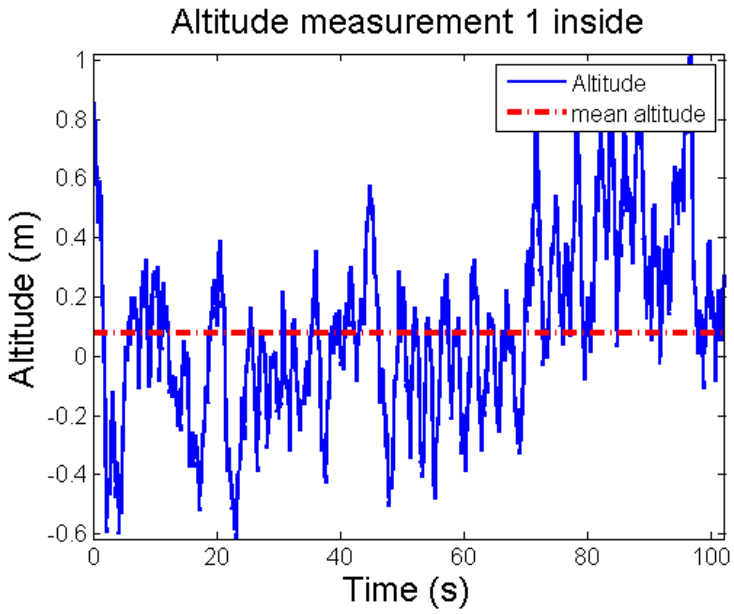
5.2.2 Test results

Method 1

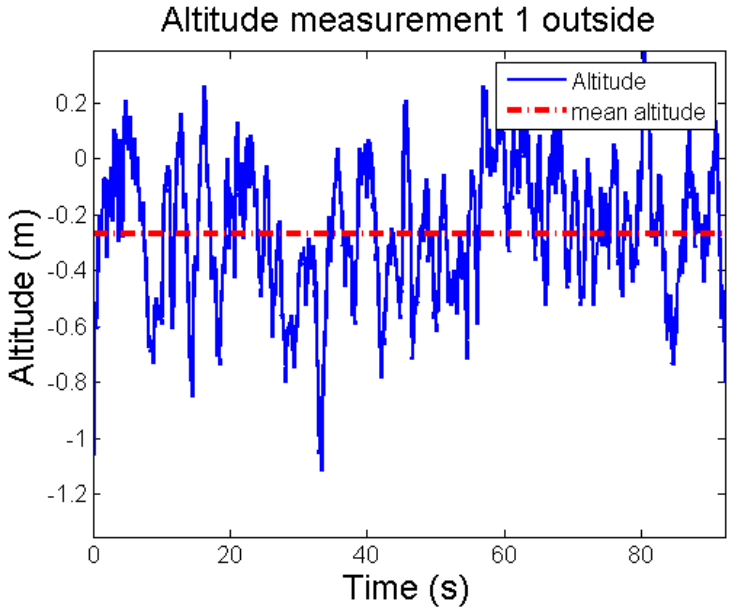
The results of the barometer placed at a flat surface are plotted in Matlab. The result of the measurement at the first spot inside and outside can be found in Figure 5.3. The results of the measurements at spots 2,3,4 outside and spot 2 inside can be found in Appendix E.

Method 2

The results of the measurement which places the sensor at a certain height can be found in Figure 5.4.

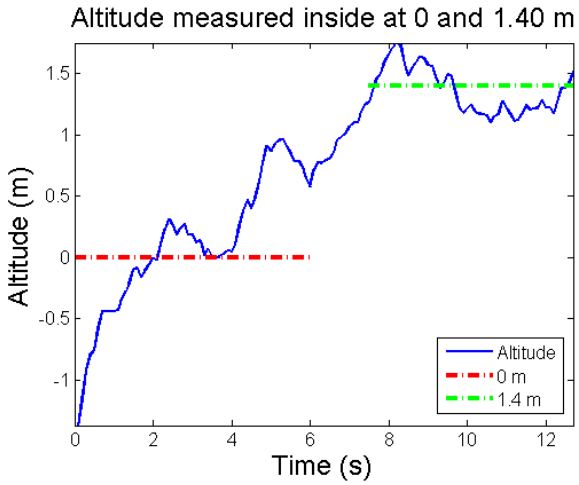


(a) Altitude measured at spot 1 inside

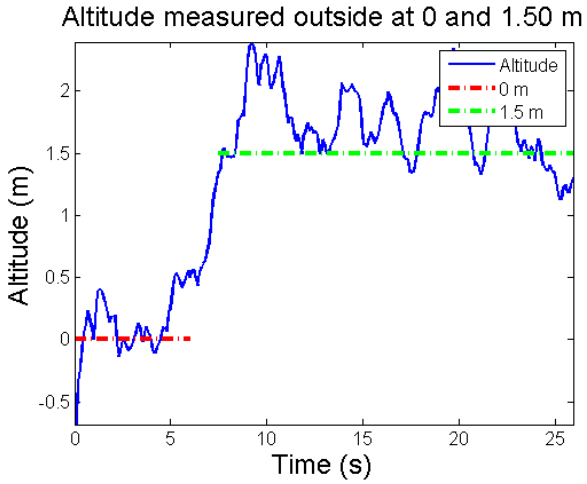


(b) Altitude measured at spot 1 outside

Figure 5.3: Altitude measured while the barometer is at a flat surface



(a) Altitude difference measured inside



(b) Altitude difference measured outside

Figure 5.4: Altitude measured while the barometer is placed at a height

5.2.3 Discussion

From Figure 5.3, Figure 5.4 and Appendix E it was concluded that there is no difference between inside and outside accuracy. The accuracy was ranging from 1.0 to 1.2 meters, the mean altitude was also off 1 to 7 decimetres. According to [40] the accuracy of the barometer is 0.2 hPa which is 1.7 meters. So the accuracy of the altitude measurement (with the use of a moving average filter) corresponded to the specifications. The design rule for the altitude was set at an accuracy of 1 meter. This design rule was not accomplished in the static measurements.

Part of the inaccuracy can be explained by the fact that when the barometer is initialised, the first 10 values determine the reference value for the altitude measurement. The altitude determination is based on this reference value so when this first 10 values do not give a good average value of the pressure on the ground, inaccuracy is introduced.

The accuracy of the altitude measurement can be improved by using an other filter or by using a sensor that is more accurate. Another option is the combination of a barometer and an INS to determine the altitude. The INS can be used to correct the barometer measurements. A third option is to use ultrasonic sensors when the UAV drops below a certain altitude, ultrasonic sensor can then be used for landing.

5.3 AR.Drone modelling

5.3.1 Test methods

To determine if the model of the AR.Drone's behaviour is properly derived, four models had to be tested. These control variables were tested by giving an input and measuring the output. The p, r, g and y parameters indicated in subsection 3.1.2 were derived and then the modelled output and actual output were compared.

The first output which was measured is the *pitch*. A desired *pitch* was given and the *pitch* measured by the AR.Drone sensors was read out. Exactly the same test method was used for *roll*, *gaz* and *yaw*.

5.3.2 Test results

The parameters that were derived can be found in Table 5.1.

Table 5.1: Parameters of the AR.Drone model

Parameter	Value	Parameter	Value
p_0	0.0014	r_0	0.0014
p_1	0.9994	r_1	0.9996
p_2	$-2.5028x10^{-4}$	r_2	$7.053x10^{-5}$
p_3	0.9994	r_3	0.9995
y_0	$1.3696x10^{-6}$	g_0	0.0220
y_1	0.9937	g_1	0.9970

With the parameters of Table 5.1 the modelled values and actual values could be compared. In Figure 5.5 the modelled pitch and actual pitch can be found. In Figure 5.6 the resulting modelled speed and actual speed is plotted. The modelled and real values of *roll*, *gaz* and *yaw* can be found in Appendix F

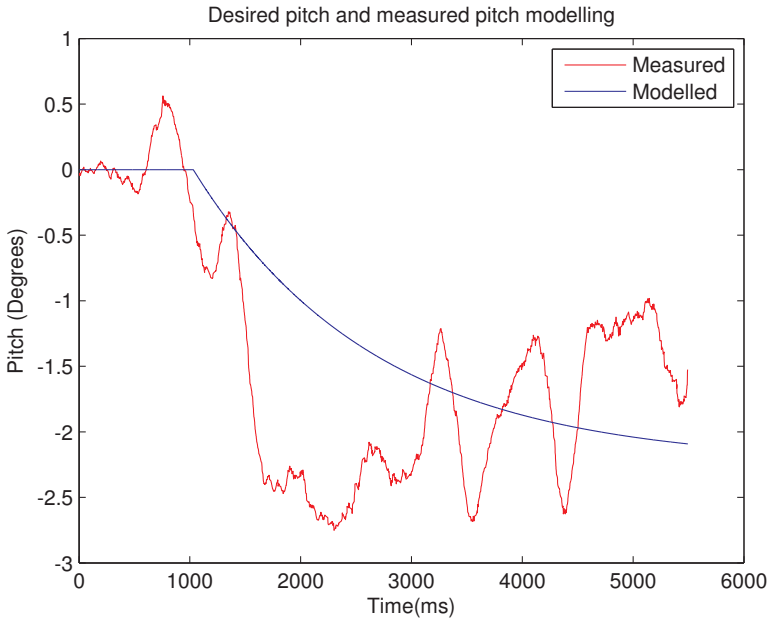


Figure 5.5: Modelled pitch compared to actual pitch

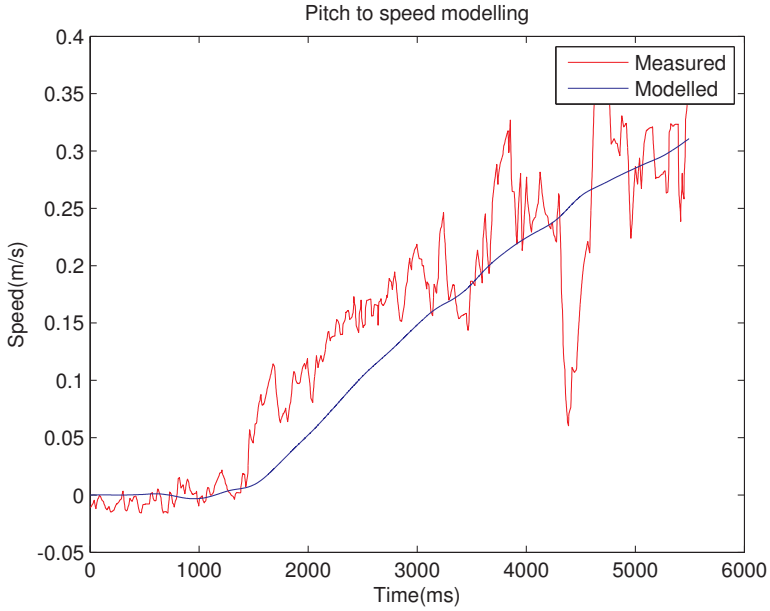


Figure 5.6: Modelled pitch to speed compared to actual speed

5.3.3 Discussion

From the images in subsection 5.3.2 and Appendix F it can be seen that *pitch* and *roll* are hard to model. The model has globally the same curve as the real values but it differs at some points 50 %. The modelled speed that is derived from the measured pitch, is following the curve well. However the spikes in the measurement are flattened in the model. The modelled *gaz* in Figure F.3 differs at least 50 % from the measured values. The modelled *yaw* however is differing less than 10 % most of the time.

The differences between the modelled values and measured values of *pitch*, *roll*, *gaz* and *yaw* can partly be explained. The spikes in the measured values can be caused by measurement errors in the sensors of the AR.Drone or by an instability of the AR.Drone. The measurement errors and instability can be caused by a small breeze or a damage to the AR.Drone.

The differences between the modelled speed and actual speed can partly be explained by the way in which the speed is measured by the AR.Drone. The AR.Drone measures it speed with accelerometers and the bottom camera [44]. Because the tests were performed on a reflecting floor, this might have influenced the speed detection. Due to the reflecting floor the AR.Drone was even drifting slightly when it was in hovering mode.

Another factor that might have influenced the measurements is that the hull of the AR.Drone was modified. Because the first layer controller of the AR.Drone is designed for the unmodified version of the AR.Drone, the controller is acting different.

5.4 AR.Drone control loops

5.4.1 Test methods

While in section 5.3 the model was tested, in this section the software that is controlling the AR.Drone has been tested.

The software was tested by letting the AR.Drone fly to destinations. This was the best method to make sure the software was doing what it should do. When the AR.Drone had reached its destination the relative position to its departure position was measured.

5.4.2 Test results

Due to bugs in the controller loop implementation the velocity information from the AR.Drone was not received correctly. This resulted in unstable controllers. Therefore the control loops could not be tested in the given time frame.

Chapter 6

Conclusion

This chapter will start with summarizing the goal of this thesis (section 6.1). Then some conclusions will be drawn from the research performed in chapter 2 and chapter 3 combined with the test results of chapter 5 (section 6.2). Next some recommendations, based on the conclusion, will be given (section 6.3). Finally suggestions for future work will be given (section 6.4). The future work section indicates what should be done, in relation to the subjects of this thesis, for the final system and what can be done to improve the current system.

6.1 Goal

For an object detection and tracking system for a UAV, a position determination system should be developed. This position data is used by the detection and tracking system to determine the drone's position, calculate the position of the object and control the UAV. Furthermore the detection and tracking system should be tested on a Parrot AR.Drone. This AR.Drone should therefore be enabled to fly autonomous. To design a position determination system and an autonomous control, two main questions were formulated:

1. What positioning method should be used for determining an position of a UAV?

2. How can the AR.Drone have autonomous control?

6.2 Conclusion

Position determination

In this subsection an answer is given to the question: what positioning method should be used for determining the position of a UAV?

In subsection 2.2.3 it was concluded that, with the stated criteria, the best position determination method to be used on the AR.Drone is the combination of a GPS sensor and a barometer. The GPS sensor is used for the position in the horizontal plane while the barometer is used for the position in the vertical plane (altitude).

In section 5.1 it appeared that the design specification of 2.0 meters accuracy in the horizontal plane was not achieved by the GPS sensor. Instead an accuracy of 3 meters is derived which is 50 % higher than the design specification. In section 5.2 an accuracy of 1 to 1.2 meters is derived for the altitude measurement of the barometer. This is maximal 0.2 meter above the design specification of 1 meter.

The conclusion is that the combination of a GPS sensor and a barometer is able to determine a position. However because the design specifications are not met, this position determination system is not accurate enough for the desired application. In section 6.3 some recommendations will be given how the design specifications can be met.

Autonomous control

In this subsection an answer is given to the question: How can the AR.Drone have autonomous control?

First a three layer model is derived (subsection 3.1.1). The first layer is the kernel control, the second layer is the command generator and the third layer is the flight scheduling. The first layer is already implemented in the AR.Drone by the control software of the AR.Drone. The second layer should be designed to control the AR.Drone. The third layer should be designed to make the control autonomous. The third layer generates desired destinations and the second layer converts these destinations to commands that can be used as input for the first layer.

The second layer of the AR.Drone can be designed by first modelling the AR.Drone's behaviour. Then this model is used to design control loops that can be used to control the AR.Drone's position.

From the tests conducted in section 5.3 it was derived that modelling the AR.Drone can be done but the model is not very accurate with deviations up to 50 %.

As described in section 5.4 the autonomous control could not be tested. Therefore it is unknown if the autonomous control is working as desired. However it is expected that the AR.Drone can be controlled by the designed controllers.

The conclusion is that an AR.Drone can have autonomous control by designing a command generator layer and a flight scheduling layer. The command generator can be designed by first modelling the AR.Drone's behaviour and designing appropriate control loops subsequently.

6.3 Recommendations

Position determination

To improve the accuracy of the used position determination method, there are a few recommendations.

To improve the accuracy of the GPS sensor a Kalman filter can be used. Due to limited time this filter could not be implemented a quick literature survey however showed that a Kalman filter would be applicable.

To improve the accuracy of the barometer, an other filter than the current moving average filter can be used. The determination of the reference values should be improved to improve the determination of the altitude. This can be done by taking a longer period to determine the reference coordinates.

However when an of-the-shelf autopilot is used for the UAV it is recommended to use the position sensors that are required by the autopilot.

To gain the best position determination result, a combination of an INS, a GPS sensor and a barometer is recommended. The INS can be used to correct the values of the GPS sensor and barometer. The INS can also provide position data when there is no GPS signal available.

Autonomous control

To improve the autonomous control, there are a few recommendations.

To improve the accuracy of the modelled behaviour an unmodified AR.Drone should be used or the first layer of the AR.Drone should be adjusted. Maybe the accuracy of the model can also be improved by doing the tests on a non-reflective, textured floor. This will improve the speed measurements of the AR.Drone.

As the controller gains are calculated based on the modelled behaviour of the AR.Drone, their values could be improved if a better model is available.

6.4 Future work

Because the limited time to implement, test and improve all proposed methods, there are some points whereon further work should be done. These points are listed below in order of priority. Some of these points are focussed on the object detection and tracking system while others are focussed on the AR.Drone as a testing platform.

I **Debugging the AR.Drone control software**
As indicated earlier, there was no time left to debug the software and therefore the control loops could not be tested.

II **Testing of the position determination method on a UAV**
Due to time constraints, it was not possible to test the position determination method on a UAV. To fully test the position determination method, it should be tested on a UAV.

III **Improvement of the position determination**
As indicated in section 6.3 the position determination can be improved in several ways. It is suggested to use an INS to improve the accuracy. The inaccuracy should be as low as possible because the position is crucial for the determination of the object and for safety reasons.

IV **Improvement of the AR.Drone model and controller**
For the use of the AR.Drone as a testing platform the model of the AR.Drone should be improved. Also other methods could be used to determine the control loops.

V **Implementation of an autopilot**
For the final UAV an autopilot should be designed or an existing autopilot should be adjusted to control the UAV. Because in the current system the autopilot of the AR.Drone is used.

VI **Implementation of absolute coordinates**
The AR.Drone control model can be adjusted to take the current position and the desired position and calculate the relative coordinates. When this is implemented, it is possible to give the desired GPS coordinates as input and the AR.Drone will fly to these coordinates. To implement this, it is recommended to implement a compass because the bearing of the AR.Drone should be known to calculate the relative coordinates.

Bibliography

[1] Y. Naidoo, R. Stopforth, and G. Bright, "Development of an UAV for search & rescue applications," in *AFRICON, 2011*, sept. 2011, pp. 1 –6.

[2] J. Sullivan, "Revolution or evolution? The rise of the UAVs," in *Technology and Society, 2005. Weapons and Wires: Prevention and Safety in a Time of Fear. ISTAS 2005. Proceedings. 2005 International Symposium on*, june 2005, pp. 94 – 101.

[3] J. Leyssens, "GNSS positioning for UAV Applications," *http-wwwgnssus*, 2009. [Online]. Available: http://www.gnss.us/uploads/GNSS_positioning_for_UAV_Applications_Leyssens.pdf

[4] G. A. Ch. Kreye, B. Eissfeller, "Architectures of GNSS/INS Integrations -Theoretical Approach and Practical Tests," 2004.

[5] C.-J. Sun, C.-W. Hung, W.-C. Huang, and C. Lin, "Implementation of GPS/INS navigation system using low-cost MEMS sensors," in *Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on*, june 2010, pp. 1588 –1592.

[6] X. Lei, J. Liang, S. Wang, and Tianmiao, "An integrated navigation system for a small UAV using low-cost sensors," in *Information and Automation, 2008. ICIA 2008. International Conference on*, june 2008, pp. 765 –769.

[7] Y. Jiong, Z. Lei, D. Jiangping, S. Rong, and W. Jianyu, "GPS/SINS/BARO Integrated Navigation System for UAV," in *Information Technology and Applications (IFITA), 2010 International Forum on*, vol. 3, july 2010, pp. 19 –25.

[8] B. Mohr and D. Fitzpatrick, "Micro air vehicle navigation system," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 23, no. 4, pp. 19 –24, april 2008.

[9] Z. Tao and W. Lei, "SINS and GPS Integrated Navigation System of a Small Unmanned Aerial Vehicle," in *Future BioMedical Information*

Engineering, 2008. FBIE '08. International Seminar on, dec. 2008, pp. 465 –468.

[10] L. Wang, S.-C. Hsieh, E.-W. Hsueh, F.-B. Hsaio, and K.-Y. Huang, "Complete pose determination for low altitude unmanned aerial vehicle using stereo vision," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, aug. 2005, pp. 108 – 113.

[11] R. Moore, S. Thurrowgood, D. Bland, D. Soccol, and M. Srinivasan, "UAV altitude and attitude stabilisation using a coaxial stereo vision system," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, may 2010, pp. 29 –34.

[12] Z. Zhu, S. Xiong, and Z. Zhou, "A micro barometric altimeter with applications in altitude-holding flight control for MAVs," in *Instrumentation and Measurement Technology Conference, 2004. IMTC 04. Proceedings of the 21st IEEE*, vol. 2, may 2004, pp. 1039 – 1041 Vol.2.

[13] T. Puls and A. Hein, "Outdoor position estimation and autonomous landing algorithm for quadrocopters using a wireless sensor network," in *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*, july 2010, pp. 285 –290.

[14] H.-C. Lee, "Simple landing distance measurement with circular mark between aircraft and runway," in *Digital Avionics Systems Conference, 2009. DASC '09. IEEE/AIAA 28th*, oct. 2009, pp. 5.A.5–1 –5.A.5–8.

[15] Parrot SA, "AR.Drone Parrot," [Online; available at: http://ardrone.parrot.com/parrot-ar-drone/usa], accessed on May 11, 2012.

[16] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "AR-Drone as a Platform for Robotic Research and Education," in *Research and Education in Robotics - EUROBOT 2011*, ser. Communications in Computer and Information Science, D. Obdržálek and A. Gottscheber, Eds. Springer Berlin Heidelberg, 2011, vol. 161, pp. 172–186, 10.1007/978-3-642-21975-7_16. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21975-7_16

[17] K. Higuchi, T. Shimada, and J. Rekimoto, "Flying sports assistant: external visual imagery representation for sports training," in *Proceedings of the 2nd Augmented Human International Conference*, ser. AH '11. New York, NY, USA: ACM, 2011, pp. 7:1–7:4. [Online]. Available: http://doi.acm.org/10.1145/1959826.1959833

[18] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *Robotics and Au-*

tomation (ICRA), 2011 IEEE International Conference on, may 2011, pp. 5776 –5783.

[19] W. S. Sharlin, Ehud Ng, "Collocated Interaction with Flying Robots," Tech. Rep., 2011.

[20] Parrot SA, "A technological first," [Online; available at: http://ardrone.parrot.com/parrot-ar-drone/en/technologies], accessed on May 17, 2012.

[21] U. S. DoD, "Global positioning system standard positioning service performance standard." *Distribution*, no. September, p. 160, 2008. [Online]. Available: http://www.navcen.uscg.gov/gps/geninfo/2001SPSPerformanceStandardFINAL.pdf

[22] NIS GLONASS, "GLONASS Technical Description," [Online; available at: http://www.nis-glonass.ru/en/glonass/technical_descript/], accessed on May 18, 2012.

[23] European Space Agency, "What is Galileo," [Online; available at: http://www.esa.int/esaNA/galileo.html], accessed on May 18, 2012.

[24] P. Groves, *Principles of GNSS, inertial, and multi-sensor integrated navigation systems*, ser. GNSS technology and applications series. Artech House, 2008.

[25] I. Mohamad, M. Ali, and M. Ismail, "Availability, reliability and accuracy of gps signal in bandar baru bangi for the determination of vehicle position and speed," in *Space Science and Communication, 2009. IconSpace 2009. International Conference on*, oct. 2009, pp. 224 –229.

[26] Y. Liu and S. Tian, "Vertical positioning technologies and its application of pseudolites augmentation," in *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on*, oct. 2008, pp. 1 –3.

[27] G. Elkaim, M. Lizarraga, and L. Pederseny, "Comparison of low-cost gps/ins sensors for autonomous vehicle applications," in *Position, Location and Navigation Symposium, 2008 IEEE/ION*, may 2008, pp. 1133 –1144.

[28] L. Sahawneh and M. Jarrah, "Development and calibration of low cost MEMS IMU for UAV applications," in *Mechatronics and Its Applications, 2008. ISMA 2008. 5th International Symposium on*, may 2008, pp. 1 –9.

[29] H. Nakanishi, S. Kanata, and T. Sawaragi, "Measurement model of barometer in ground effect of unmanned helicopter and its application

to estimate terrain clearance," in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, nov. 2011, pp. 232 –237.

[30] R. Moore, S. Thurrowgood, D. Bland, D. Soccol, and M. Srinivasan, "A stereo vision system for uav guidance," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, oct. 2009, pp. 3386 –3391.

[31] Y. chi Liu and Q. hai Dai, "Vision aided unmanned aerial vehicle autonomy: An overview," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, vol. 1, oct. 2010, pp. 417 –421.

[32] K. Peng, G. Cai, B. M. Chen, M. Dong, K. Y. Lum, and T. H. Lee, "Design and implementation of an autonomous flight control law for a uav helicopter," *Automatica*, vol. 45, no. 10, pp. 2333 – 2338, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109809003082

[33] J. T. Zimmerling and H. A. R. Homulle, "Intelligent uav camera system object searching and tracking," Bachelor Thesis, Delft University of Technology, June 2012.

[34] S. L. van Berkel and X. F. van Kooten, "Ground station user interface and data link for unmanned aerial vehicles," Bachelor Thesis, Delft University of Technology, June 2012.

[35] Beagleboard.org, "Home," [Online; available at: http://www.beagleboard.org/], accessed on May 31, 2012.

[36] Raspberry Pi Foundation, "Raspberry Pi," [Online; available at: http://www.raspberrypi.org/], accessed on May 31, 2012.

[37] W. E. Hong, J.-S. Lee, L. Rai, and S. J. Kang, "Embedded real-time software architecture for unmanned autonomous helicopters," *Journal of Semiconductor Technology and Science*, 2005.

[38] "Fastrax UP501 Data Sheet," Fastrax Ltd, Valimotie 7, FIN-01510 Vantaa, Finland.

[39] "NMEA Manual for Fastrax IT500Series GPS receivers," Fastrax Ltd., Valimotie 7, FIN-01510 Vantaa, Finland, June 2009.

[40] "BMP085 Data Sheet," Bosch Sensortec GmbH, Gerhard-Kindler-Strasse 8,72770 Reutlingen, Germany.

[41] "USB Hi-Speed FT4232H Evaluation Module," Future Technology Devices International Ltd (FTDI), Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow, G41 1HH, United Kingdom, June 2011.

[42] G. Franklin, J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Pearson, 2009.

[43] Moveable Type Scripts, "Calculate distance, bearing and more between Latitude/Longitude points," [Online; available at: http://www.movable-type.co.uk/scripts/latlong.html], accessed on June 4, 2012.

[44] P.-J. Bristeau, F. Callou, D. Vissière, and N. Petit, "The navigation and control technology inside the ar.drone micro uav," in *18th IFAC World Congress*, Milano, Italy, 2011, pp. 1477–1484.

Appendix A

Design Brief

On the following pages the design brief of the final product can be found. The final product is a completely working object detection and tracking system.

1. Introduction

For this Bachelor Thesis Project within the curriculum of Electrical Engineering at the department of EEMCS of Delft University of Technology, a system will be designed for implementation on an Unmanned Aerial Vehicle (UAV). This system will be a combination of hardware and software. It will focus on the professional market and, with minor modifications, is to be used in the UAV Competition organized by Thijs Durieux and Johan Melis. This document will specify project requirements and design constraints.

2. Requirements derived from the intended use

2.1. The system must flexible to the extent that it can be implemented on both so-called 'quadro-copters'[1] and fixed-wing UAVs. A detailed definition of 'flexible':

2.1.1. The system must be able to function properly on a fixed-wing UAV with a velocity of 100 km/h.

2.1.2. The battery power, the number of available I/O ports and the available processing power must be sufficient to allow for an extension of the system with a second camera and additional sensors. This requirement is related to the future possibility of tracking other airborne vehicles.

2.2. The system must be capable of detecting a ground-based moving vehicle from the air.

2.3. The system must be capable of tracking (i.e., following) the ground-based vehicle at a distance of no more than 50 m.

2.4. The velocity of the detected vehicle may not exceed 55 km/h.

2.5. The system must be capable of taking a photograph of the detected ground-based vehicle.

2.6. The system must be capable of transmitting the photograph to a ground station.

2.7. The system must be able to determine its position (x and y) with a resolution of 2 m.

2.8. The system must be able to determine its height (i.e., z) with a resolution of 1 m.

2.9. The system need *not* autonomously control a UAV.

2.10. The system need *not* withstand rain or extreme humidity.

2.11. The mass of the system – including camera and battery – may not exceed 500 g.

2.12. The system must function at least 1 hour with its own battery.

2.13. The system must be delivered and documented in such a way that work can be continued by people not regularly affiliated to the project and our project team.

2.14. A (scaled) demonstrator must be delivered, consisting of a *Parrot AR.Drone* that is capable of tracking an RC model car and transmitting an aerial photograph of this car to a base station. The control and position signals transmitted by the UAV must be visualized in a graphical user interface (GUI) on the ground station.

3. Requirements derived from the (ecological) situation of the system in its surroundings

3.1. The system must comply with all legislation concerning radio wave transmission.

3.2. The altitude of the system may not exceed 70 m.

3.3. The system must be able to communicate with its ground station in all possible geometrical positions.

3.4. The system must be able to communicate with its ground station over a range of 1000 m.

3.5. The system is to be built from standardized components and shall therefore comply with environmental standards such as RoHS.

3.5.1. The lifetime of the system will be determined by the lifetime of the components.

3.5.2. Requirements with respect to recycling and disposal are determined by the components of the system.

[1] Helicopter-like aircraft with four horizontal rotors, often arranged in a square layout.

4. Requirements concerning the system as such

4.1. The system must generate and transmit a signal if the battery power drops below a predetermined threshold value.

4.2. The system must be able to cope with flight vibrations to the extent that its functionality is not influenced during flight.

4.3. The system must send a signal to the autopilot if the connection between the system and its ground station is lost for more than t seconds. This threshold value t must be set to the desired value digitally.

4.4. The system must send signals to the autopilot protocol or bus (to be determined) concerning its position and the (estimated) position of the tracked vehicle.

4.5. The data link between the system and the ground station must be suitable for transmitting autopilot signals with a maximum bandwidth of 2 MHz.

MoSCoW model draft

Must have

- All points mentioned in the design brief.

Should have

- Possibility of detecting an airborne vehicle (e.g., another UAV).
- Possibility of tracking the detected airborne vehicle.

Could have

- Fully functional built-in autopilot.
- Possibility of defining geographical operating boundaries of the UAV.

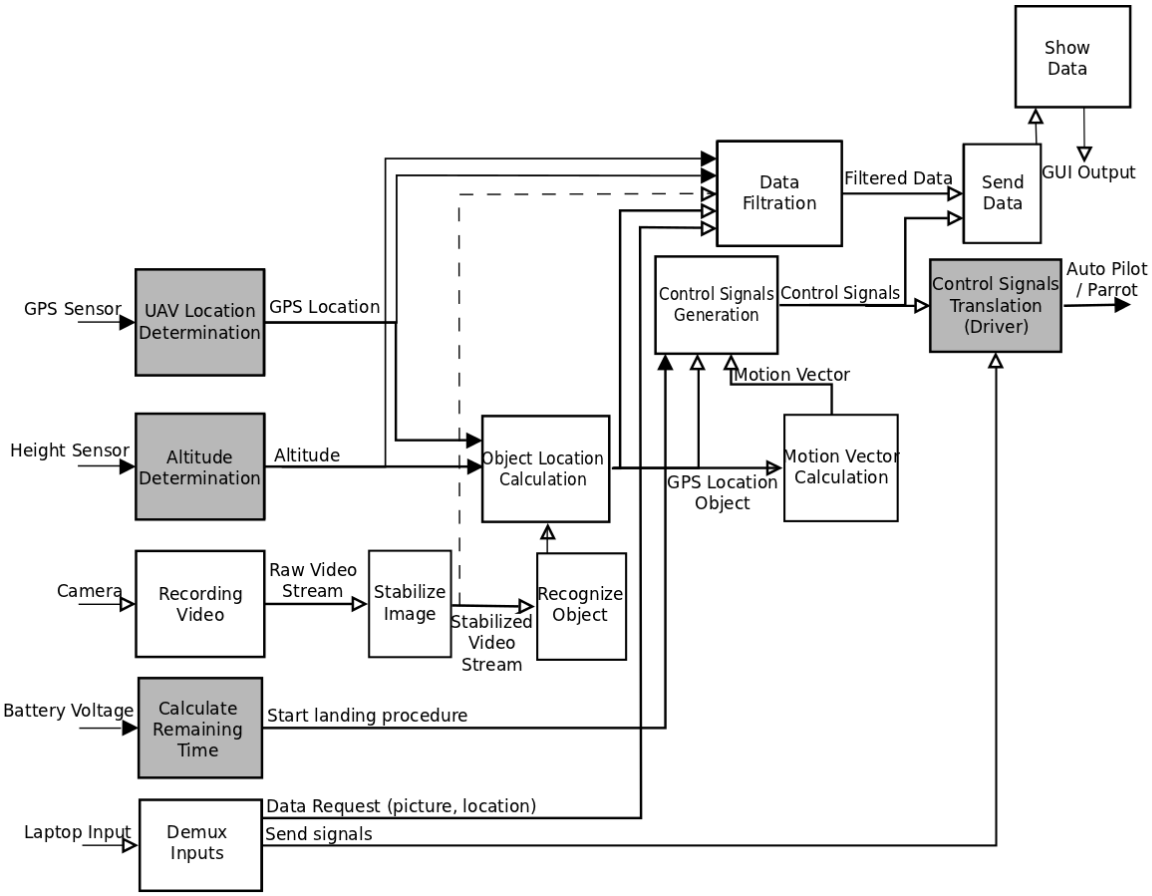
Would have (would be)

- A fully autonomous fixed-wing UAV that meets the demands of Thijs Durieux and Johan Melis.

Appendix B

Functional block diagram

In the figure below a functional block diagram of the whole system can be found. At the left the inputs of the system can be found while at the right the outputs can be found. In the middle the processing of the input to the output is showed.



Appendix C

RMC message format

$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,b,s.s,d.d,ddmmyy,x.x,c*hh

hhmmss.ss	=	UTC of position fix
A	=	Data status
llll.ll	=	latitude of fix
a	=	North or South
yyyyy.yy	=	longitude of fix
b	=	East or West
s.s	=	speed over ground in knots
d.d	=	track made good in degrees true
ddmmyy	=	UT date
x.x	=	magnetic variation in degrees
c	=	East or West
hh	=	checksum

Appendix D

GPS fixed spot measurements

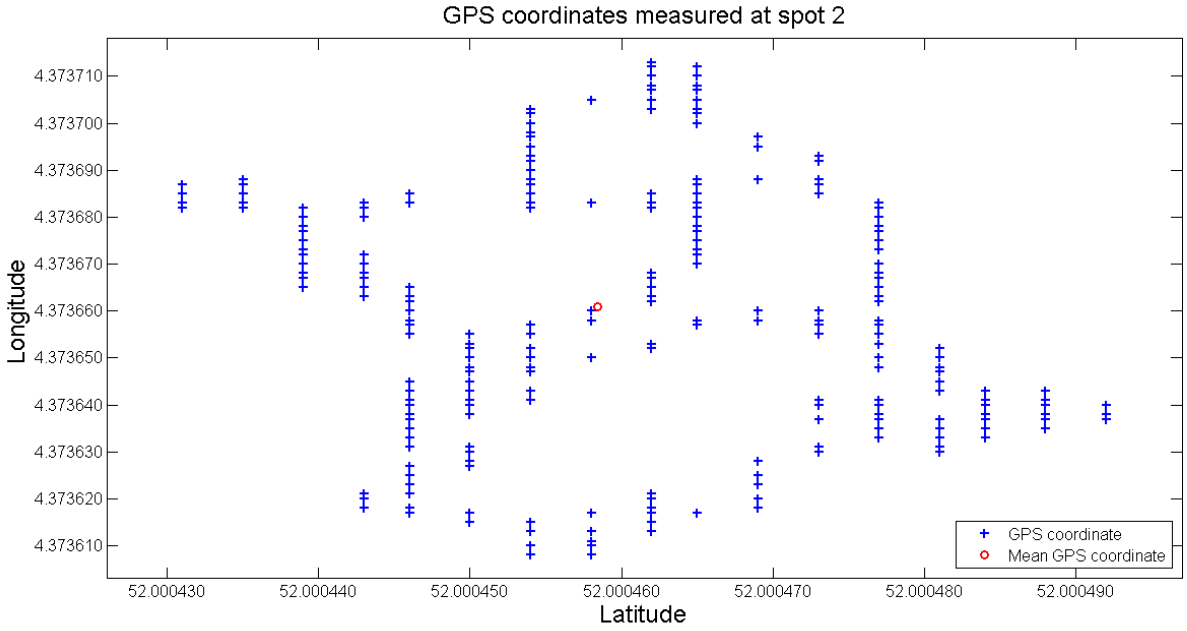


Figure D.1: GPS coordinates measured at spot 2

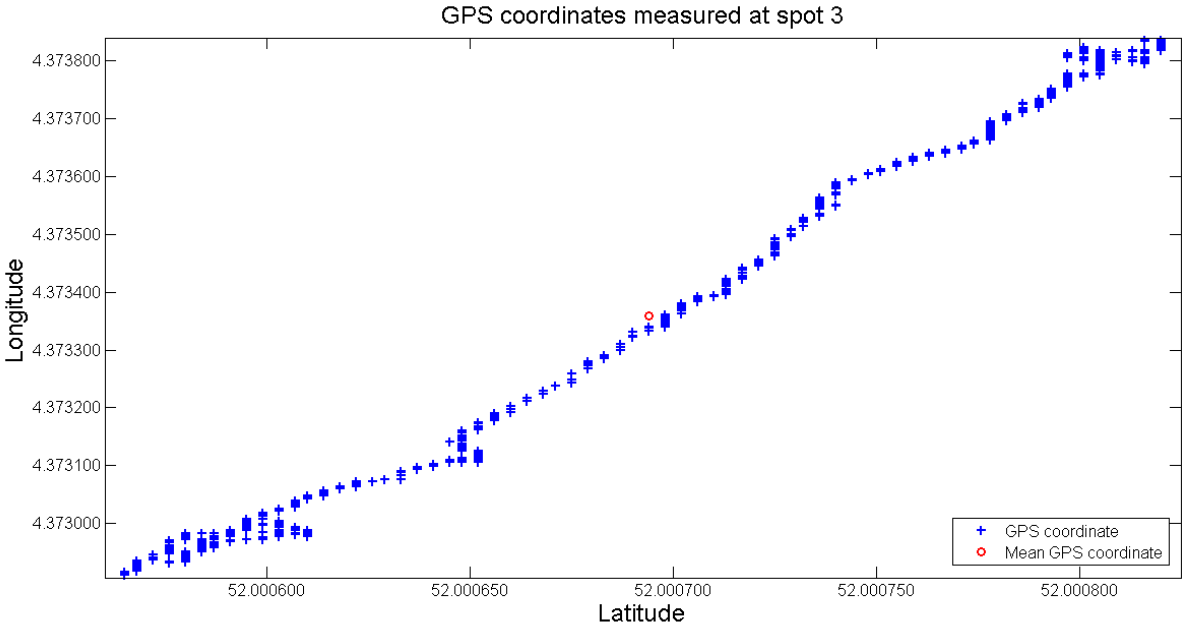


Figure D.2: GPS coordinates measured at spot 3

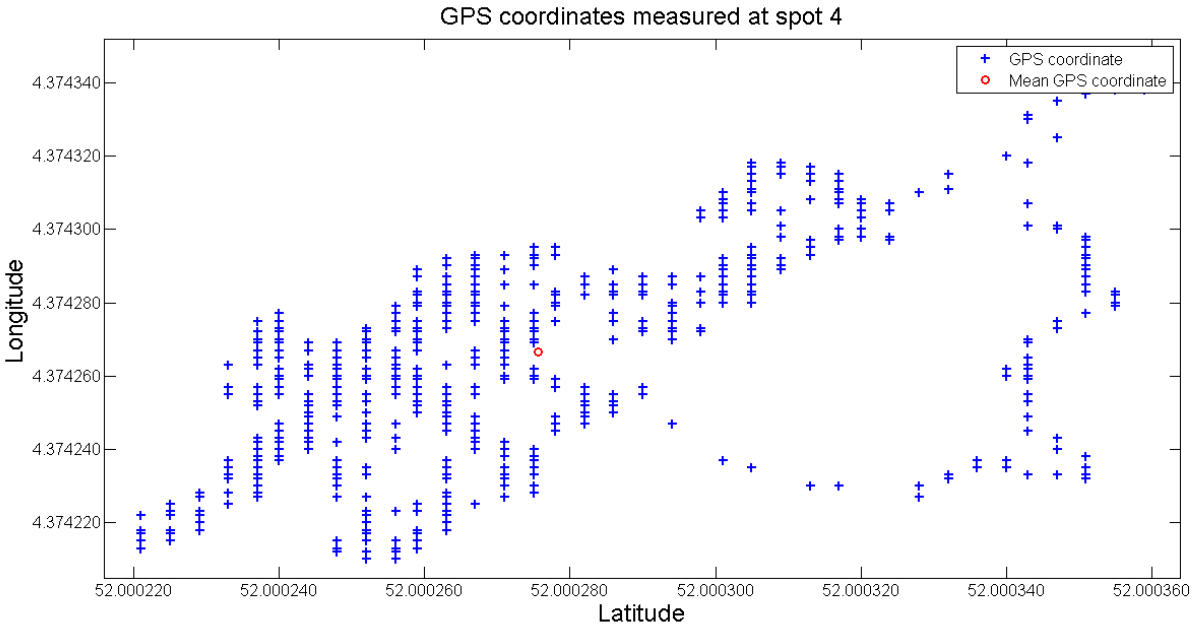


Figure D.3: GPS coordinates measured at spot 4

Appendix E

Barometer flat surface measurements

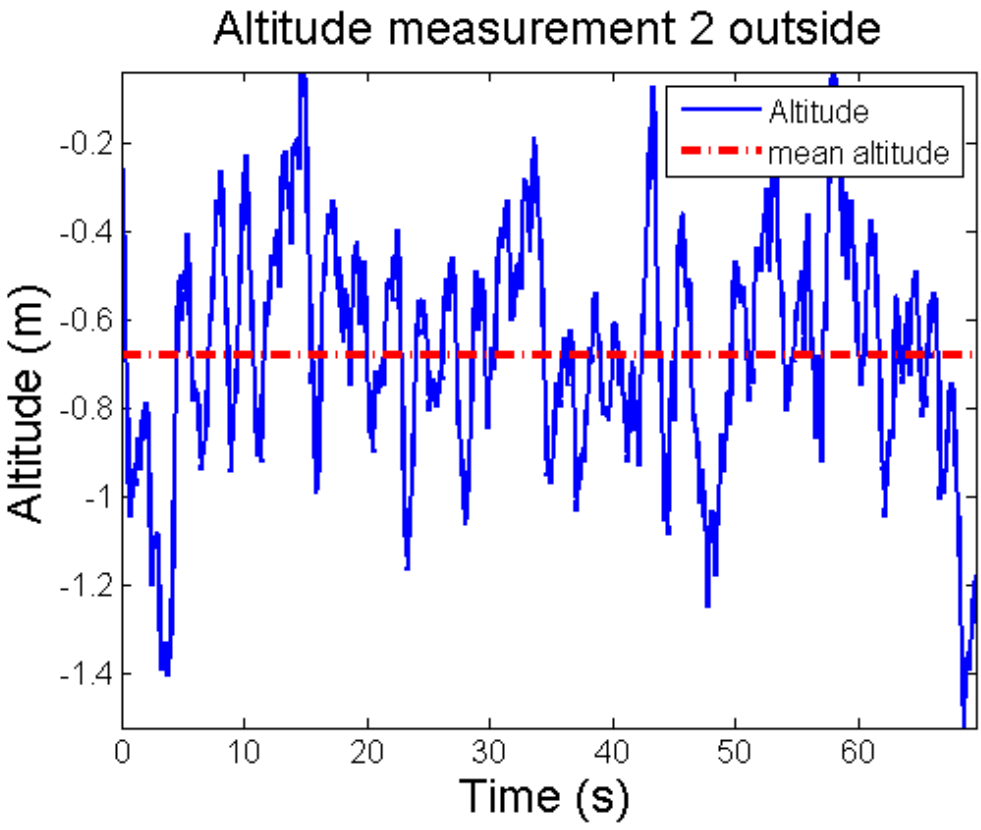


Figure E.1: Altitude measured at spot 2 outside

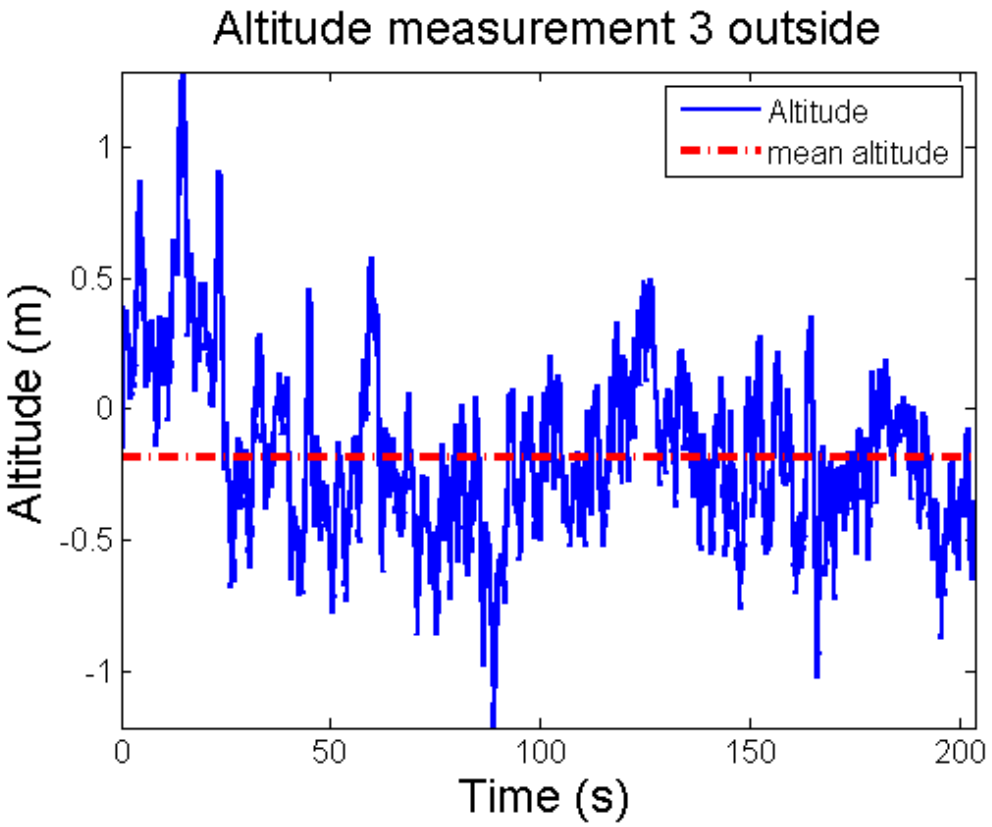


Figure E.2: Altitude measured at spot 3 outside

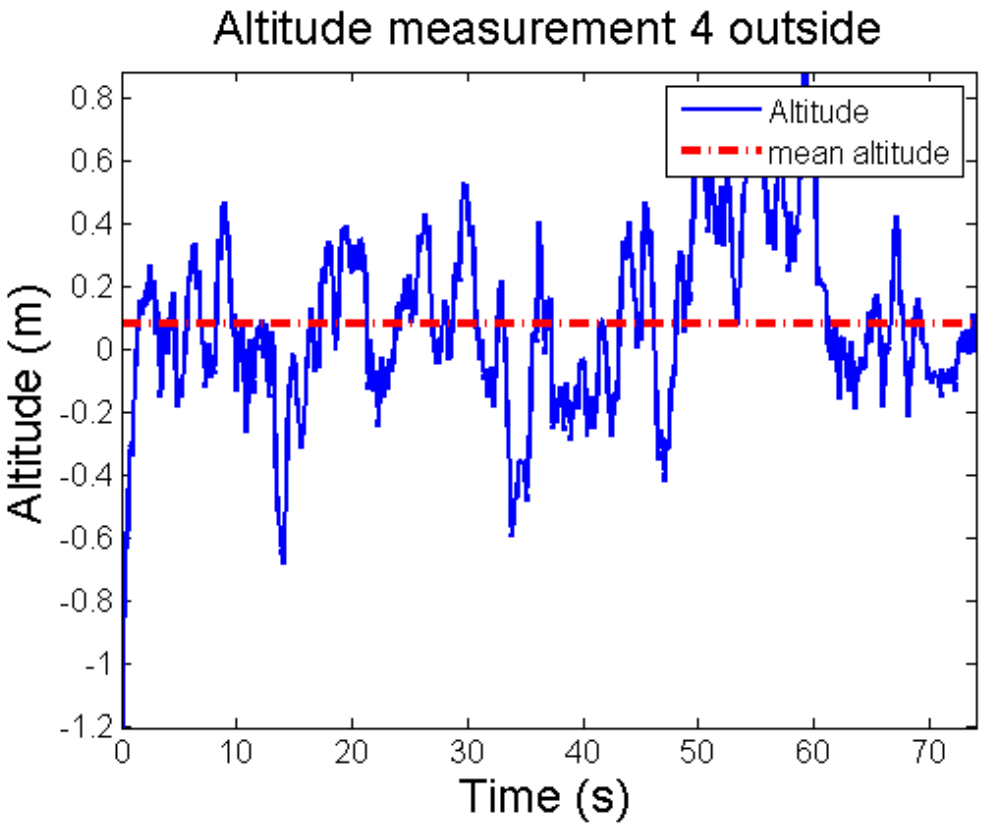


Figure E.3: Altitude measured at spot 4 outside

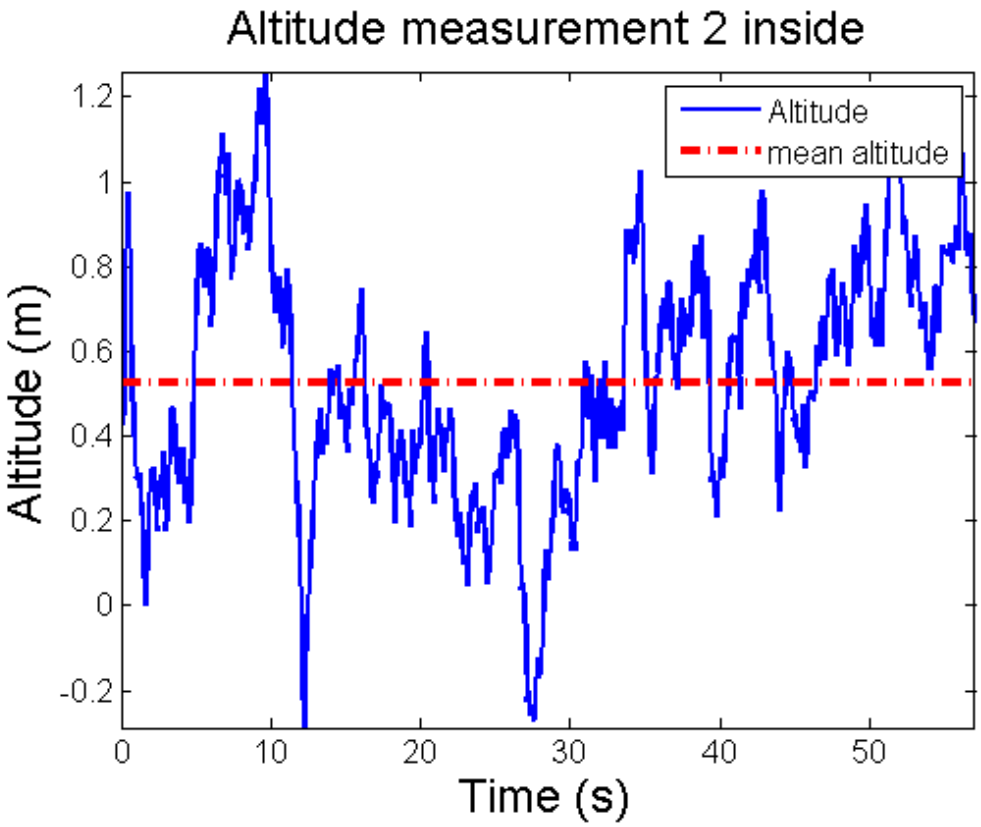


Figure E.4: Altitude measured at spot 2 inside

Appendix F

Behaviour model measurements

In this appendix the modelled and measured parameters are plotted.

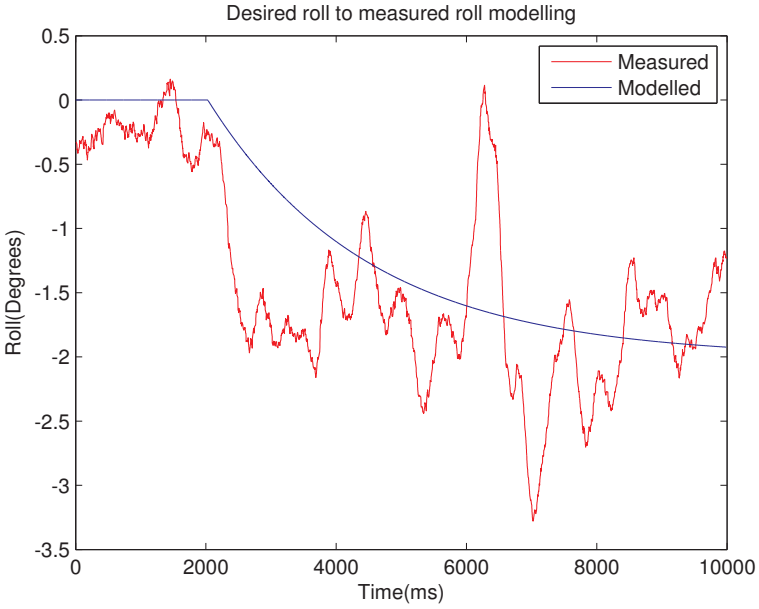


Figure F.1: Modelled roll compared to actual roll

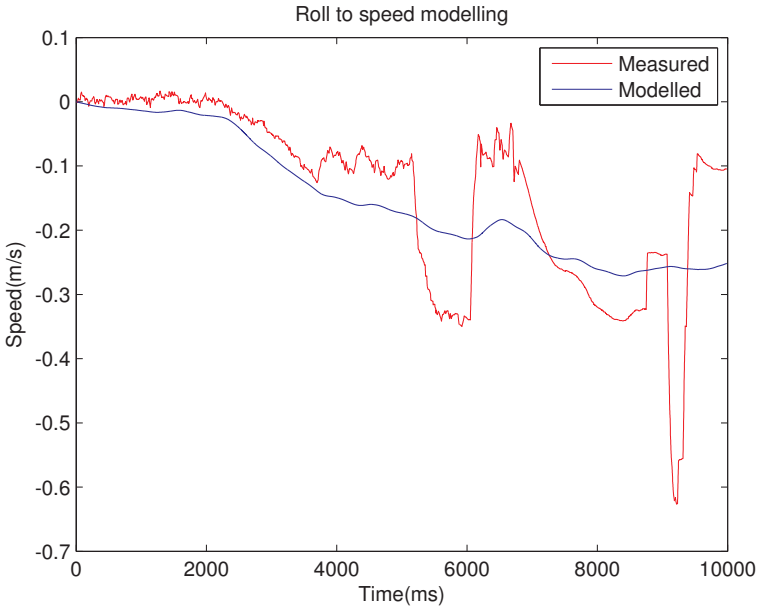


Figure F.2: Modelled roll to speed compared to actual speed

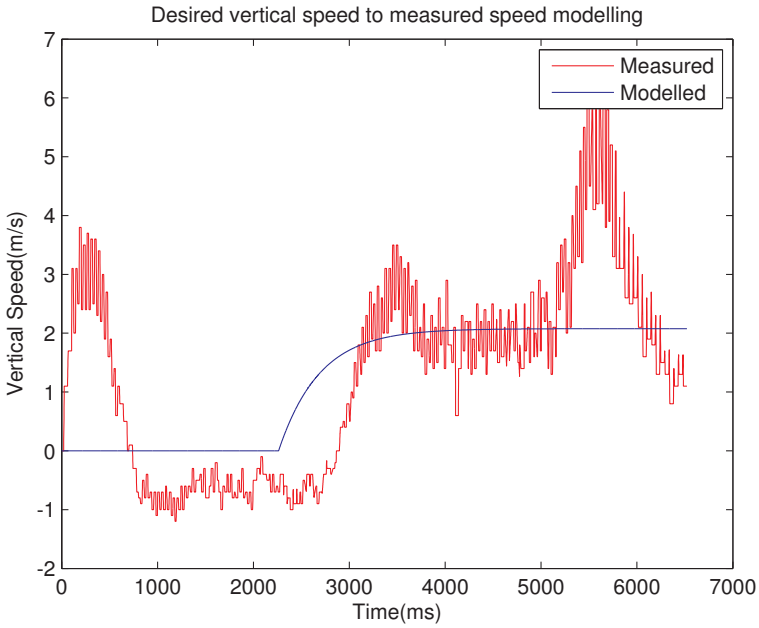


Figure F.3: Modelled gaz compared to actual gaz

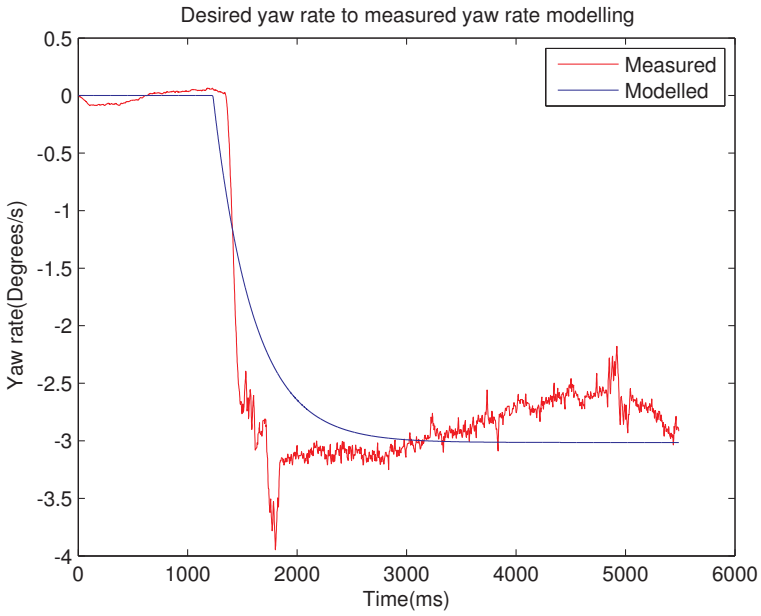


Figure F.4: Modelled yaw compared to actual yaw