

Quadcopter Control using Android-based Sensing

Hannes Bergkvist

August Bjälemark



LUND
UNIVERSITY

Department of Automatic Control

MSc Thesis
ISRN LUTFD2/TFRT--5936--SE
ISSN 0280-5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

© 2013 by Hannes Bergkvist & August Bjälemark. All rights reserved.
Printed in Sweden by Media-Tryck
Lund 2013

Abstract

This thesis investigates the concept of a quadcopter implemented using the sensors and computational power of a smartphone. The main goal is to give an example of the possible applications of external hardware combined with commercial off the shelf (COTS) electronics. Practical problems that required attention were implementation of sensor fusion using the smartphone's sensors and the controller as a smartphone app. Attention was also paid to the real-time aspects of the Android operating system.

Acknowledgments

Thanks to Jerker Nordh and Anders Robertsson for good supervision and availability when help was needed.

Contents

1. Introduction	5
1.1 Background.....	5
1.2 Problem formulation	7
2. Quadcopter	8
2.1 Overview	8
2.2 Principle of Quadcopter.....	11
2.3 Model	13
3. Android.....	17
3.1 Overview	17
3.2 Aspects of Android and Java	19
4. Mechatronics.....	20
4.1 Circuit boards.....	21
4.2 The custom made circuit board	22
4.3 Motor controllers	25
4.4 Power supply	26
5. Communication	28
5.1 Networking.....	28
5.2 Implementation of Android-Computer Communication	34
5.3 PWM.....	35
5.3 Android and Circuit board	39
6. Sensors	43
6.1 Sensors used in the quadcopter	43
6.2 Sensor fusion	46
6.3 Implementation of filter.....	54
6.4 Calibration.....	58
6.5 Linear movement	60
7. Control.....	62
7.1 Theory.....	62
7.2 Discretized PID controller.....	70
8. Interface and software.....	72
8.1 Interface.....	72

9. Results	76
10. Conclusion	83
11. Future work	85

1. Introduction

External hardware based on commercial of the shelf (COTS) electronics is a concept that enables development of simple and low cost devices that get their functionality and intelligence when connected to the COTS-product.

A smartphone is an example of COTS electronics with multiple features such as:

- Computational power
- Sensors
- Wireless connectivity
- Touch screen
- Camera
- Easy development and distribution of third party applications

Worldwide sales of smartphones exceeded those of ordinary mobile phones in 2013 and there will be 1.4 billion smartphones in use by the end of the year.^{1 2} As the number of smartphones increase it is of interest to investigate the concept of combining a smartphone with external hardware.

A quadcopter is a good platform for this investigation as it:

- Need sensors (Accelerometer, Magnetometer and Gyroscope)
- Need computational power (Processor)
- Need algorithms for control and functionality (Easy development of applications)
- Benefit from wireless connectivity (WiFi, Bluetooth, 3G/4G)
- Benefit from other sensors available (pressure sensor)
- Benefits from GPS
- Benefit from camera
- **Is a challenge as it has real time requirements**

In this project it will be shown that it is possible to control a quadcopter using only the sensors on a smartphone and simple circuitry that enables the smartphone to regulate the rotors.

Since Android is the most used mobile operating system when this project started, an Android based smartphone is suitable for this investigation.

1.1 Background

Quadcopter

A quadcopter is an aerial vehicle with four rotors, see figure 2.1. The first designs made for manned flight was introduced in the beginning of the 20th century.³ Today quadcopters are almost exclusively unmanned, small, electrical and used as hobby toys or within research. Other more practical applications of quadcopters are beginning to appear, mainly within aerial photography and surveillance.

The quadcopter needs a control system to be able to fly steadily. This is usually implemented with some kind of microprocessor and accelerometer and gyroscope sensors measuring the state of the quadcopter.

Smartphone

The term smartphone appeared for the first time in 1997, when the company Ericsson described its mobile phone named Penelope as a “smart phone”.⁴ Today it is usually used for a mobile phone with more advanced features such as touch screen, camera, sensors and a sophisticated operating system. The most common operating systems are Android, iOS and Windows phone.⁵ The computing power of smartphones have increased during the last years,⁶ the fastest smartphone on the market in July 2013 features a 1.9 Ghz quad-core processor. Most smartphones have the possibility of easy development, distribution and use of third party applications. These applications can further expand the functionality of the smartphone.

State of the art

In 2013 there are several commercial and research products that are based on smartphones. One of them is Romo,⁷ which is a commercial robot based on an iPhone. It can drive around by itself, be controlled from another iPhone or a web browser and perform image processing such as facial recognition. An example of a research application is NASA’s satellites based on Android phones (PhoneSat).⁸ The PhoneSat project strives to decrease the cost of satellites while not sacrificing performance. An example of a commercial available quadcopter is the DJI Phantom.⁹ It is remotely controlled with a radio transmitter, has a built in high definition camera and GPS for navigation.

1.2 Problem formulation

The main goal of this project is to build and implement a quadcopter using the sensors and computational power of a smartphone. No external sensors are to be used and all computations must be made by the smartphone. External mechatronics for generation of pulse width modulated signals and three phase current will be required.

2. Quadcopter

2.1 Overview

The quadcopter is made up of many parts that have their own unique tasks. Every task needs to be synchronized with the others and therefore need a way to communicate with each other. Figure 2.1 shows the quadcopter built in this project. Figure 2.2 illustrates all the parts, how they are connected and how they communicate with each other.

Android phone

The smartphone can via sensors and signal processing algorithms estimate the quadcopter's angles and angular velocities around its different axes. This data is used to determine control signals which are sent to the circuit board. All of this information can also be sent to the computer.

Computer

The computer is the link between the operator and the smartphone. It runs a Java program with an implemented Joystick for steering and a Matlab script with a GUI, in which the operator can set different parameters (control parameters, filter parameters etc.). The computer also runs Kst2, which is a real time plotting program, to plot all signals from the Android phone.

Circuit board

The circuit board's only purpose is to translate data from the smartphone to signals for the motor controllers.

Motor controller

The motor controllers read the signal from the circuit board and generates three phase current for the brushless motors.

Rotors

The rotors are the actuators on the quadcopter. When rotating they give the quadcopter its lifting force, also denoted as thrust.

Power supply

The smartphone has a battery that provides power to the phone and half of the circuit board. The other half of the circuit board, the motor controllers and the motors can be supplied by either a LiPo (Lithium polymere) battery or a power supply unit (PSU), depending on if the quadcopter is going to fly or if an experiment is going to be performed at surface level.

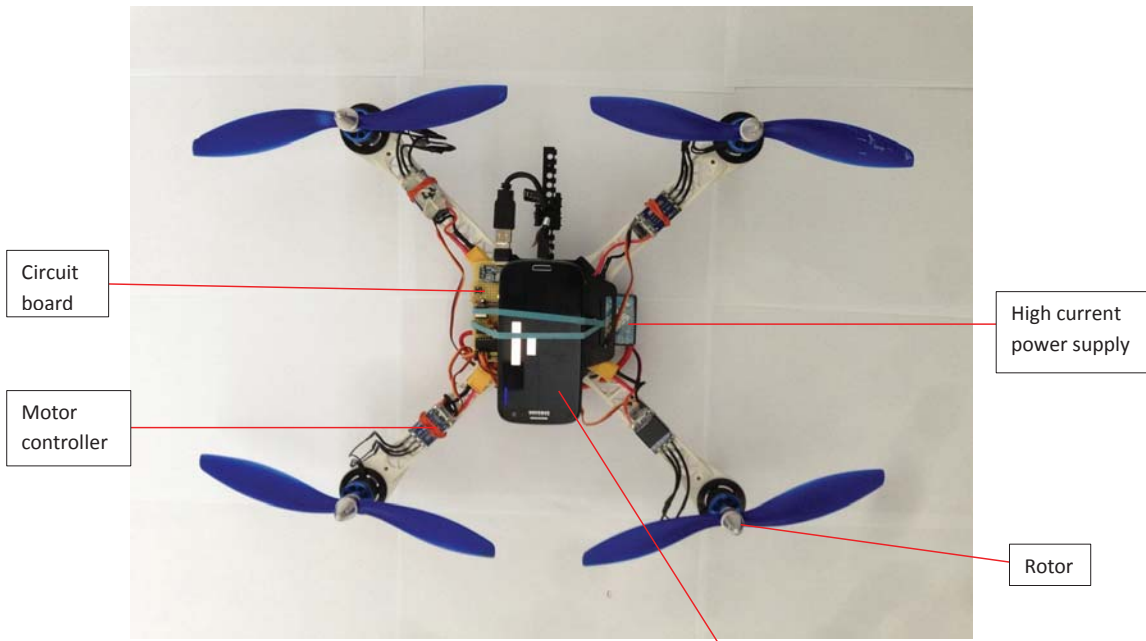


Figure 2.1: Picture of the Quadcopter built in this project

Android phone

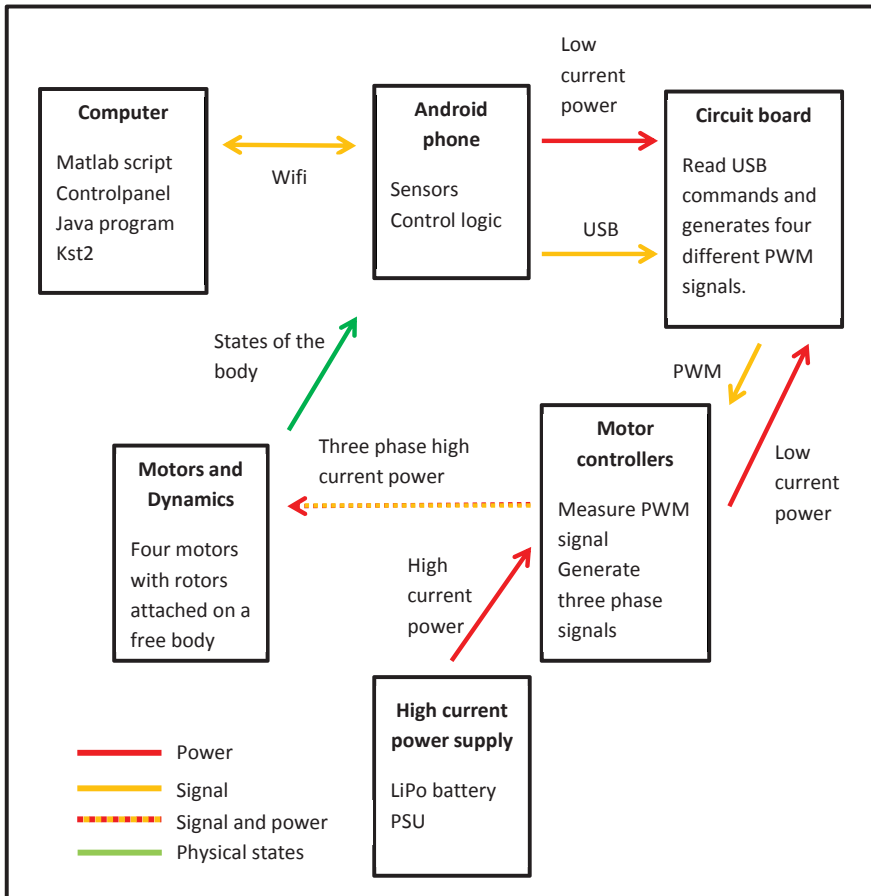


Figure 2.2: Overview of the Project. PWM (pulse width modulated) signals is described in the Communication section.

2.2 Principle of Quadcopter

Four rotors

The design of a quadcopter enables it to move without any other moving parts except the rotors. The speed of the four motors determines all movements of the quadcopter. It needs a control system and angular sensor data for continuous stabilization during flight.

A quadcopter is usually oriented in a plus- or X-configuration, the difference is the definition of forward movement see figure 2.3. This project use X-configuration because of practical reasons regarding the fastening of the smart phone device.

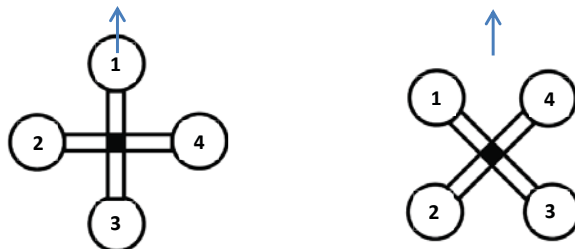


Figure 2.3: Quadcopter, plus and X-configuration with the rotors numbered 1-4. The arrow represents the direction of forward movement.

Dynamics

The coordinate system of the quadcopter used in this project is illustrated in figure 2.4.

Each motor will add an upward thrust and a torque around its own axis. The motors next to each other have opposite rotational directions, to cancel the motor's rotational torque against each other. In order for the quadcopter to hover, it is a necessary but not sufficient condition that all the motors thrusts cancel the gravitational pull.

When the two motors with the same rotational direction is increased relative to the two others the quadcopter is rotating around the Z-axis. This is called yaw (figure 2.5.a). When all the motors increase their thrust the quadcopter will move upwards (figure 2.5.b).

When two motors next to each other increase their thrust relative to the two others, the quadcopter will rotate around the X- or Y- axis. This is called pitch and roll respectively (figure 2.5.c).

A rotation around the Y-axis will result in a movement along the X-axis and a rotation around the X-axis will result in a movement along the Y-axis (figure 2.6).

In this way the quadcopter can move in all three directions and rotate around all three axes.

The rate at which the quadcopter rotates around its axes (angular acceleration) depends primarily on the moment of inertia around the axis and the torque the rotors exert around that axis. The quadcopter linear acceleration depends primarily on the orientation of the quadcopter, the force the motors exert on it and its mass. The relationship between these quantities and a more detailed explanation is provided in section 2.3, Model.

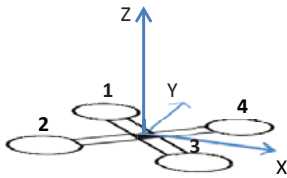


Figure 2.4. The coordinate system of the quadcopter used in this project. The forward movement is in the Y direction.

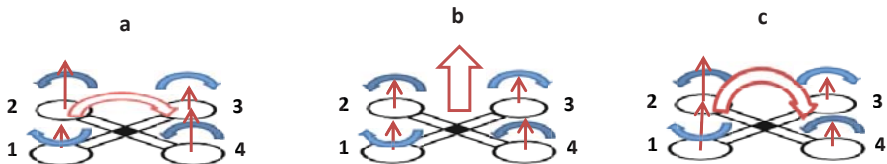


Figure 2.5: Movements of Quadcopter: **a**: rotation around the Z-axis (yaw), **b**: movement in the Z-direction, **c**: rotation around the Y-axis (roll). Rotation around the X-axis is called pitch.

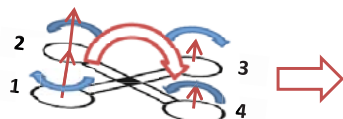


Figure 2.6: Rotation around Y-axis results in movement along the X-axis.

2.3 Model

Motors

The thrust of a motor is determined by a pulse width modulated (PWM) signal to a motor controller which controls the three phase current to the motor. The PWM signal is a pulse train, where the pulse lengths determine the thrust of a motor (see section 5.4). A measurement of the thrust of the motor with a varying length of the PWM pulse shows a linear relationship between one and three newtons of thrust, see figure 2.8. The model used for the motors is therefore chosen as (2.1) where T is the thrust of the motors and P the pulse length. a and b are parameters that describe the linear relationship.

$$T = a * P + b \quad (2.1)$$

The measurement of the thrust / PWM relationship was performed, as seen in figure 2.7, with the quadcopter mounted on a balljoint. The motors were measured individually with a dynamometer fastened under the rotor. The dynamometer registered the thrust generated for each PWM signal.

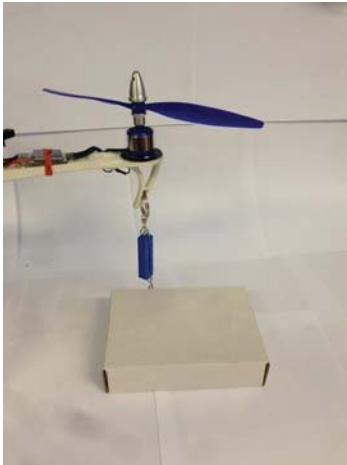


Figure 2.7: Measurement arrangement

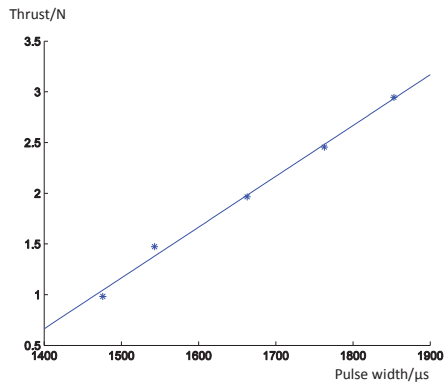


Figure 2.8: Thrust / PWM pulse length relationship.

Quadcopter

This section provides a model description of the quadcopter's rotation around its different axes as well as a description of the position in z-direction. The model is derived *Modelling, Identification and control of a quadrotor Helicopter*,¹⁰ with the important results summarized in (2.1) and (2.2).

$$\begin{aligned}
 \ddot{X} &= (\sin(\psi) \sin(\phi) + \cos(\psi) \sin(\theta) \cos(\phi)) \frac{U_1}{m} \\
 \ddot{Y} &= (-\cos(\psi) \sin(\phi) + \sin(\psi) \sin(\theta) \cos(\phi)) \frac{U_1}{m} \\
 \ddot{Z} &= -g + \cos \theta \cos \phi \frac{U_1}{m} \\
 \ddot{\theta} &= \frac{I_{yy} - I_{zz}}{I_{xx}} \dot{\phi} \dot{\psi} - \frac{J_{TP}}{I_{xx}} \dot{\phi} \Omega + \frac{U_2}{I_{xx}} \\
 \ddot{\phi} &= \frac{I_{zz} - I_{xx}}{I_{yy}} \dot{\theta} \dot{\psi} - \frac{J_{TP}}{I_{yy}} \dot{\theta} \Omega + \frac{U_3}{I_{yy}} \\
 \ddot{\psi} &= \frac{I_{xx} - I_{yy}}{I_{zz}} \dot{\theta} \dot{\phi} + \frac{U_4}{I_{zz}}
 \end{aligned} \tag{2.1}$$

$$\Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \tag{2.2}$$

where Ω_i is the angular velocity of each propeller, X, Y, Z the position of the quadcopter, θ , Φ , ψ the angle around the X, Y and Z axis respectively, J_{TP} the total rotational moment of inertia around the propeller axis and U_{1-4} are the inputs to process described in (2.3).

$$\begin{aligned}
 U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\
 U_2 &= lb(-\Omega_2^2 + \Omega_4^2) \\
 U_3 &= lb(-\Omega_1^2 + \Omega_3^2) \\
 U_4 &= d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)
 \end{aligned} \tag{2.3}$$

where l is the distance between the centre of a propeller and the centre of mass of the quadcopter, d the drag factor (aerodynamic property) and b the thrust factor (aerodynamic property).

The model derived is for a plus configuration, while the quadcopter in this project is in X-configuration. Furthermore in this project the thrust / input signal relationship of each motor is modelled, instead of the angular velocity / input signal relationship. The input signals U_{1-4} are therefore changed to (2.4).

$$\begin{aligned}
 U_1 &= T_1 + T_2 + T_3 + T_4 \\
 U_2 &= l(T_4 + T_1 - T_2 - T_3) \\
 U_3 &= l(T_1 + T_2 - T_3 - T_4) \\
 U_4 &= R_2 + R_4 - R_1 - R_3
 \end{aligned} \tag{2.4}$$

where T_{1-4} are the thrusts from the different motors and R_{1-4} the torque they provide around the Z-axis.

This project aims to stabilize the orientation of the quadcopter and not the position. The interesting model is therefore summarized as (2.5) and (2.6).

$$\begin{aligned}
 \ddot{\theta} &= \frac{I_{yy} - I_{zz}}{I_{xx}} \dot{\phi} \dot{\psi} - \frac{J_{TP}}{I_{xx}} \dot{\phi} \Omega + \frac{U_2}{I_{xx}} \\
 \ddot{\phi} &= \frac{I_{zz} - I_{xx}}{I_{yy}} \dot{\theta} \dot{\psi} - \frac{J_{TP}}{I_{yy}} \dot{\theta} \Omega + \frac{U_3}{I_{yy}} \\
 \ddot{\psi} &= \frac{I_{xx} - I_{yy}}{I_{zz}} \dot{\theta} \dot{\phi} + \frac{U_4}{I_{zz}} \\
 \ddot{z} &= -g + \cos \theta \cos \phi \frac{U_1}{m}
 \end{aligned} \tag{2.5}$$

$$\begin{aligned}
 U_1 &= T_1 + T_2 + T_3 + T_4 \\
 U_2 &= l(T_4 + T_1 - T_2 - T_3) \\
 U_3 &= l(T_1 + T_2 - T_3 - T_4) \\
 U_4 &= R_2 + R_4 - R_1 - R_3
 \end{aligned} \tag{2.6}$$

Simplification of model

As the model (2.6) and (2.7) derived is nonlinear and too complicated for the purpose of this project, some simplifications are made. The controller will be derived under the assumption that the quadcopter will mostly hover with all its angles and angular velocities equal to zero.

The equations that describe the angle in (2.5) consist of three terms. The first term can be neglected due to the product of the angular velocities, which are assumed to be close to zero. Ω is considered to be small, as all the propellers rotate at about the same speed when hovering. The second term can therefore be neglected due to the product of Ω in equation (2.2) and the angular velocity.

The simplified model is therefore

$$\begin{aligned}\ddot{\theta} &= \frac{U_2}{I_{xx}}, \ddot{\Phi} = \frac{U_3}{I_{yy}}, \ddot{\psi} = \frac{U_4}{I_{zz}} \\ \ddot{Z} &= -g + \cos \theta \cos \Phi \frac{U_1}{m}\end{aligned}\tag{2.7}$$

with

$$\begin{aligned}U_1 &= T_1 + T_2 + T_3 + T_4 \\ U_2 &= l(T_4 + T_1 - T_2 - T_3) \\ U_3 &= l(T_1 + T_2 - T_3 - T_4) \\ U_4 &= R_2 + R_4 - R_1 - R_3\end{aligned}\tag{2.8}$$

3. Android

3.1 Overview

Android is an operating system made primary for touch screen mobile devices. The operating system was first developed in the company Android, which was founded in 2003 and acquired by Google in 2005. It is based on Linux, and released under the Apache license.¹¹ The first phone with Android as operating system was the HTC Dream which was released in 2008.¹² In 2013 Android.com stated that there are more than 900 000 000 devices activated with Android as operating system.¹³ Every major update of the Android software is named after a dessert, the latest release is 4.3 Jelly Bean. There are numerous third party applications available for Android devices. Google Play is the biggest market place for Android Apps with over 975 000 applications.

Samsung Galaxy S3

The device used in this project is the Samsung Galaxy S3. It was released in May 2012. The specifications of the model are presented in table 3.1.

Samsung Galaxy S3	
Dimensions	136.6x70.6x8.6 mm
Weight	133g
Display	4.8 inch, 1280x720
CPU	1.4 GHz quad-core ARM Cortex-A9
Network/Bearer and Wierless connectivity	2.5G (GSM/ GPRS/ EDGE): 850/ 900/ 1800/ 1900 MHz 3G (HSPA+ 21Mbps):850/ 900/ 1900/ 2100 MHz 4G (Dependent on market)
Memory	32GB + microSD slot (up to 64 GB)
OS	Android 4.1 (Jelly Bean)
Battery	2.100 mAh
Connectivity	WiFi a/b/g/n, WiFi HT40 GPS/GLONASS NFC Bluetooth 4.0(LE)
Audio and Video	Audio Codec: MP3, AMR-NB/WB, AAC/AAC+/eAAC+, WMA, OGG, FLAC, AC-3, apt-X Video Codec: MPEG4, H.264, H.263, DivX, DivX3.11, VC-1, VP8, WMV7/8, Sorenson Spark Recording & Playback: Full HD (1080p)
Camera	Main(Rear): 8 Mega pixel Auto Focus camera with Flash &Zero Shutter Lag, BSI Sub(Front): 1.9 Mega pixel camera, HD recording @30fps with Zero Shutter Lag, BSI
Sensors	Accelerometer, RGB light, Digital compass, Proximity, Gyro, Barometer

Table 3.1. Specifications for Samsung Galaxy S3 ¹⁴

3.2 Aspects of Android and Java

The Android phone is a complex system that is nondeterministic and therefore usually not suited for hard real time applications. The nondeterministic issues arise from different areas including:

1. It is based on Java and Java has its own problems regarding the real time aspect.
2. It is hard to know, and take into consideration, all the details about how the system is implemented. It includes system services that take up processing power and settings that developers are unaware of.

Java aspects

The main disadvantage of using Java in real time applications is its garbage collector. Its task is to locate allocated memory on the heap (a large pool of memory) that is never going to be used again and deallocate it. This process takes time and it is hard to predict when it occurs and therefore is not suited for hard real time applications.

Android has however support for C/C++ development, which are more deterministic alternatives, using NDK (native development kit). This involves writing C/C++ functions that are later going to be called from the Java program. Using native calls (calls to C/C++ functions) will however affect the performance.¹⁵ It is therefore not always better to use native functions over regular Java functions.

There are some other means of minimizing the effect of the garbage collector. One of them is to avoid allocating memory on the heap during the execution of the program (don't create objects using the new keyword).

Android uncertainties

The smartphone was not developed to be used as a real time device. Because of this, there are services in the smartphone that can affect performance unexpectedly. One such service that was noticed in this project was that the performance of the smartphone increased when the USB cable was plugged into the computer.

It can be seen in figure 3.1 how different unexpected behaviors explained above affect the time it takes to complete a control cycle.

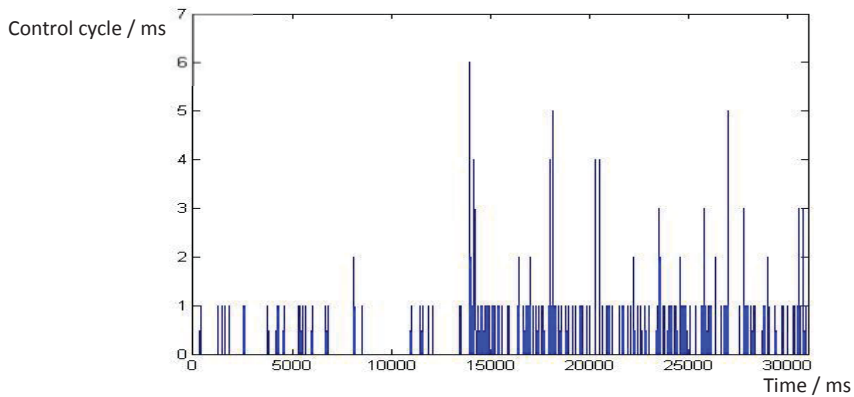


Figure 3.1: The time it takes to perform all calculations during a control cycle at different moments of time. All measured values are truncated. After 14 seconds some additional code is running in the control loop, with 10 objects being allocated every cycle. As seen in the graph the time to complete a cycle is now more often 1ms or above than before the 14 seconds mark. There are also several spikes. These could be a result of the garbage collector.

4. Mechatronics

The mechatronics part of the project involves the circuit board, the motor controllers, the motors and the power supply.

4.1 Circuit boards

The circuit board is the smartphone's connection to the quadcopter. By using USB communication, the smartphone is able to control the thrust of the four motors. The circuit board receives commands from the smartphone and generates four different PWM signals to the motor controllers.

Different circuit boards have been tested in this project, to investigate what was commercially available (table 4.1): the IOIO, an Arduino board and a custom made board specifically made for this project. There are Advantages and drawbacks with every board, and they use different methods to communicate with the Android phone.

IOIO

The IOIO (pronounced yoyo) is a couple of years old circuit board that is made for hardware development with Android phones.¹⁶ In order to use the IOIO you also need to use the IOIO library that is free to download [13]. This library is used with the preloaded software in the IOIO. In this way it is quite easy to quickly develop a project, as you don't have to program the IOIO directly. If you for example want to turn on a pin on the IOIO (set its voltage to 5V), you just call the appropriate function from the Android code. The rest is handled by the software running on the IOIO and the Java library on the smartphone. The IOIO is powered via two of its pins (Vin and gnd, see Appendix A.2). It acts as the master in the USB communication with the smartphone and supplies the smartphone with power (for more information regarding the USB communication, refer to the *USB cable* section).

Arduino

The Arduino tested in this project is based on an Atmega2560 microprocessor, a USB to serial converter and a chip that makes the Arduino master in the USB communication.¹⁷ The schematics of the board can be seen in Appendix A.3.

Unlike IOIO, you have to program the circuit board yourself and establish your own protocol of communication with the Android. This makes it more difficult to develop, as you have to program two devices. However, there are a lot of free open source libraries to download to the Arduino that simplifies a lot of the programming.

In the same way as for the IOIO, the Arduino functions as the master in the USB communication with the Android phone. That means power has to be supplied externally and the Arduino, which will in turn supply power to the smartphone.

4.2 The custom made circuit board

This project aims to use as little external circuitry as possible. Therefore a simpler custom made circuit board was constructed. The board is based on the ATmega88 microprocessor and the schematic of the circuit board can be seen in Appendix A.1.¹⁸ It is divided into two parts using two optocouplers, the smartphone part and the ATmega88 part. Unlike the Arduino and IOIO, part of the power to the circuit board comes from the Android phone, as the smartphone in this case acts as master in the USB communication. The first part, bottom half in the figure, is powered by the smartphone and the second part is powered via the motor controllers by the LiPo battery or the power supply unit.

USB to serial converter

The first component of the custom made circuit board is a chip that converts USB to RS232, which is a serial protocol. It is needed because the microprocessor on the circuit board (ATmega88) communicates using the RS232 protocol (see section 5.3).

Status diodes

The UART (Universal Asynchronous Receiver/Transmitter), is a hardware used in this project that sends serial data to another UART (see section 5.3). It has a RTS (request to send) pin, which has positive voltage if there is a valid connection between the devices and negative voltage if the connection is not valid. This can be used to determine if something is wrong in the communication, by placing two LEDs in parallel, but with opposite polarity. In this way one of the LEDs will be on if there is a positive voltage (able to send information) and the other will be on if there is a negative voltage (unable to send information).

Optocouplers

The two chips in the middle of the circuit board are called optocouplers and connect the two parts of the circuit board without galvanic connection. An optocoupler is composed of an infrared LED and a photodiode, a diode that conducts only when exposed to infrared light. If there is a positive input voltage on the optocoupler, the led will light up and the photo diode will conduct. This way one can send information without the need of conduction, which works as a safety measure. If something generates a high voltage spike on one side of the circuit board, the spike will not be transferred to the other side of the board, and not enter the Android phone.

USB cable

In a USB communication there is a master and a slave. The master supplies power to the USB cable and is responsible for all the data transfer over the bus, while the slave can only signal that it requires attention.¹⁹ Therefore the master requires more software and circuitry than the slave.

The Arduino and IOIO each acts by default as master, which means that the phone is charging when plugged into them. The custom made circuit board was not implemented to be able to act as a master and therefore the smartphone had to be the master.

In order to signal to the smartphone that it should be the master, a special USB cable was used in the communication between the smartphone and the circuit board. This type of cable is called a USB on-the-go (or USB-OTG).

The architecture of the cable is similar to a regular USB cable, with the only difference being that on the smartphone end, the fourth connector is connected to ground, see figure 4.1. In a regular cable, the fourth pin is not connected to anything.

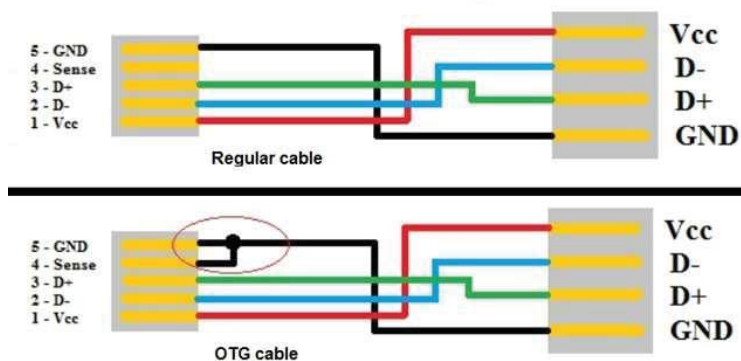


Figure 4.1: comparison between a regular USB to USB micro cable and a USB OTG cable. In the USB OTG cable, the fourth (sense) connector is connected to ground to tell the smartphone that it should act as master.²⁰

	IOIO	Arduino	Custom made board
Communication speed	~2 400 000 bps ²¹	-	115200 bps, could theoretically be implemented with higher speed.
Communication latency	4 ms-12 ms, depending on how much activity the board has to do.	About 1 ms.	5-7 ms.
Power requirements	About 0.04 ampere	About 0.1 ampere.	About 0.04 ampere. Half the board is powered by the Android.
Weight/size	13g / 77x32 mm	53g / 110 x 55 mm	70g
Difficulty	No custom code for the IOIO needed. No custom protocol needed.	Custom code for the Arduino needed, but with help with from the Arduino libraries. Custom protocol needed.	Custom code and hardware implementation needed. Custom protocol needed.
Cost	About 350 SEK ²²	About 700 SEK ²³	The components for the board designed in this project cost about 100 SEK.

Table 4.1: Comparison of circuit boards.

4.3 Motor controllers

The LiPo battery and the power supply unit are only able to provide DC. The motors are brushless three phase AC motors. A motor controller transforms the DC into three phase AC.

The motor controllers internal logic measures the length of a pulse received from the circuit board, draws the required power from the power source and creates three phase current with the appropriate frequency.

Another feature of the motor controllers is that they have an on board 5V voltage regulator. This feature is used by the motor controller to supply power to its logic. In this project the onboard voltage regulators were connected in parallel to supply power to the circuit board as well.

4.4 Power supply

The power supply is composed of the Android's Li-ion (Lithium-ion) battery and a LiPo battery or a power supply unit. Different parts of the system have different power requirements. The circuit board for instance requires much less power than the motors. However, the circuit board requires more stable power supply at 5V.

Li-ion battery

The battery normally supplies the smartphone with power. In this project, the battery is supplying power to both the phone and a part of the circuit board. The battery has a capacity of 2200 mAh, which is enough to keep the smartphone with the program on and supplying power to the circuit board, for about an hour.

LiPo battery

The LiPo (Lithium polymere) battery supplies power to the motors, motor controllers and the largest part of the circuit board. The battery is connected to the four motor controllers in parallel. The motor controllers then supply power to the motors and via the voltage regulator to the second part of the circuit board.

LiPo batteries are one of the newest designs of batteries that are used commercially in Radio controlled planes and cars. They have a quite good power-storage-to-weight ratio. They also have a very high C value, which is a measure of how much current they can supply at a given moment in time. The amount of current they can provide is equal to their capacity in Ah times the C value.

The battery that was used in this project has a capacity of 5 Ah, a C value of 30, a voltage supply of 7.4 V and weighted 250g. This means the battery can supply a maximum of 150A at 7.4 V.

The amount of time the quadcopter can fly with this battery depends on the current to the motors. At full power the motors draw 15A each and all the circuits that are supplied by this battery draw less than 0.5 Ampere (which is negligible under these circumstances). 5 Ah with 60 Ampere means that the batteries will be drained in 5/60 of an hours, or 5 minutes.

The motors usually never work at full effect for the whole range of 5 minutes. A more reasonable assumption is that they work at about a third to half of the maximum effect. In this way the quadcopter will fly for about 10 to 15 minutes.

Power supply unit

When a new software or algorithm is to be tested, as a safety measure the first tests were performed on ground level, with the quadcopter suspended on a ball joint. In those cases it is more convenient to supply the quadcopter with a power supply unit instead of the battery, as the PSU does not need to be recharged.

Another advantage with using the power supply unit is that there is a switch on it. In the scenario when there is a bug in the code that makes the quadcopter behave violently, one can quickly turn off the power to the motors with the switch, before any damage is done. A switch implemented in software, for example a button in the Matlab GUI is more convenient, but not as safe as a physical switch, due to potential bugs in the switch code or problems with the wi-fi connection.

A drawback of using the PSU is that it has other characteristics than the battery. Furthermore, as the battery in this case is not attached to the quadcopter and contributing to its weight, center of mass, and moment of inertia, the characteristics of the quadcopters dynamics are changed. All the tests done with the PSU will therefore not be completely representative of the quadcopter flying with battery.

The PSU used in this project supply 12 V at a maximum of 45 A. The motors will not be able to be on full effect, as the PSU can only supply 45 A and the motors draw a total of 60 A when on full effect. This is not an issue, since the point of using the PSU is to test algorithms on the ball joint when the motors work at low power.

5. Communication

5.1 Networking

The computer and the Android communicate over a wireless network with the help of two sockets, which are endpoints in communication flow across a computer network. The socket types used in this project are TCP sockets and a UDP socket (see section *TCP* and *UDP*) which are the most common ones.

IP addresses and ports

The way internet communication works is very complex, but the most basic aspects of it is that every device connected to the internet has an IP (Internet Protocol) address. The two mostly used IP protocol versions today are IPv4 and IPv6. IPv4 is older and more common. The IP addresses corresponding to IPv4 are four byte numbers, usually written as four numbers ranging from 0 to 255. This makes it possible to have about 4 billion different addresses. As the number of devices and the need of unique addresses increase, the IPv6 was developed. It uses IP addresses that are 16 bytes long making it possible to have $3.4 \cdot 10^{38}$ different addresses.

There are differences in how you implement the IPv4 and IPv6 protocols. Because of previous experience with IPv4, this version was chosen.

The internet communication algorithm uses IP addresses to know where information is being sent to from a program. In this way only the device with the correct IP address will receive the message.

The IP address shows which device is connected to the network. There are usually multiple programs running on a device, each program using one or many ports. When the programs communicate across the network, there has to be an identifier telling which port the message is being sent to. This identifier is the port number which ranges from 1 to 65535.

NAT

A device's IP address doesn't have to be globally unique. NAT (Network Address Translation) makes all the networks, created by a router, to have a unique IP address and all devices inside the network have local IP addresses.

When a program wants to send a message to another program running on another device, the internet communication algorithm first looks in the local network for the correct IP address before going outside the local network, to the internet.

It is not easy for a program to send information to another device located behind another NAT, which means it's located in another network. The reason for this is that only the router that creates the network is seen globally, not the device, located behind the NAT.

There are a couple of solutions to this problem:

1. The receiving program is running on a server. A server is not behind a NAT and is accessed directly through an external IP address. The sending program's router knows and remembers that the program sends the message to that server. When information is later sent back from the server, the router knows which program it needs to redirect the message to, as it was this program that initiated the communication.
2. One can manually tell the router to redirect messages incoming on a specific port to a certain device.
3. The two programs communicating could have a common server, which has an external IP address. The individual programs communicate with the server like alternative 1, and the server redirects the messages sent from the programs.

Socket

A network socket is a software implementation of an endpoint in a communication between two processes across a computer network. Wi-Fi sockets are based on the internet protocol, and are therefore called internet sockets. A socket could be interpreted as an object in an object oriented language, meaning it has certain attributes associated with it:

- Local socket address, which is the local IP address and the port number associated with the communication.
- Transport protocol, which is the way the communication is occurring through this socket. The most common are TCP or UDP, which are described below.
- The IP address and port number of the remote host.

A socket is a way for a program to send information to another program running on another device. The receiving program then has its own socket to receive the information and possibly send something back. There are a lot of ways this information could be sent. It all depends on the communication protocol, where the most common is TCP followed by UDP.

Problems with network communication

In a network communication, a message is composed of small packets of information that are sent after each other. All of these packets have information regarding the IP address and the port they are being sent to.

There are many problems that can occur when a message, composed of these packets, is sent over a network. Among other are:

- Some packets do not arrive
- Some packets arrive in different order.
- There is too much information being sent, the router overflows and cannot operate correctly.

TCP

TCP (transmission control protocol) is the most common type of protocol.²⁴ The protocol is more advanced than UDP. The aim of this protocol is to solve all the problems mentioned above. The way it does this is complex, but a simple explanation is that in a TCP communication there are two sockets; one is called server and the other client.

In a TCP communication, the two sockets establish a connection between each other. After the connection is established, the communication can commence. The communication is established in the following matter:

1. The server is waiting for a connection from a client
2. A client, which has to know the IP address and port number of the server socket, calls the server and asks to establish a connection.
3. The server has the choice of either accepting the call or rejecting it.
4. If the server accepts the call, a connection is established and communication can begin. If the call is rejected, then the client has to either try again or abandon the attempt of establishing a connection.

Compare the procedure with a regular phone call, where the caller is the client and the one who receives the call is the server. In the same manner it is only the caller that has to know the number of the one it's calling. Then it is up to the receiver to either accept or reject the call. If the receiver accepts the call, then communication can begin in both ways.

The reason for the connection is the way TCP solves the problems mentioned above.

- If a package is sent, for instance from the server to the client, the server awaits an acknowledgement from the client that the package has arrived. If the server does not receive one, the package is assumed to be lost and the server automatically resends it.
- When all packets in a message arrive, the TCP protocol rearranges them (if necessary) so they are in the correct order.
- If there is a lot of traffic on the network, the TCP protocol decreases the speed of the messages sent with a control algorithm, so the router does not overflow.

The fact that TCP solves these problems makes the protocol very reliable and the default choice for most applications.

Nagle's algorithm

Nagle's algorithm is a means of improving the efficiency of a TCP connection by reducing the amount of packets being sent.²⁵ It accomplishes this by buffering up information until a predetermined package is full, before it is being sent. The package will also be sent if a certain amount of time has passed without the package being sent.

Without the algorithm there is a possibility that many packages would only contain one byte of information and the header, which is between 20 and 60 bytes. The downside of this feature is that it takes some more time to send the information. There is therefore the option of turning it off, which is desired when low latency is required.

In figure 5.1 the latency is measured with the Nagle's algorithm enabled and in figure 5.2 the latency is measured with the algorithm disabled. The latency is measured with a keyword which is being sent to the Android device and back. The time it takes for the keyword to return is then measured.

As can be seen in the graphs, the latency varies a lot. Figure 5.1 has some measurements with very low latency, which can be a result of that the package was filled immediately when the keyword was added. In both cases 146 bytes was sent from the smartphone every 10th millisecond and 13 bytes was sent from the computer every 10th millisecond.

Latency / ms

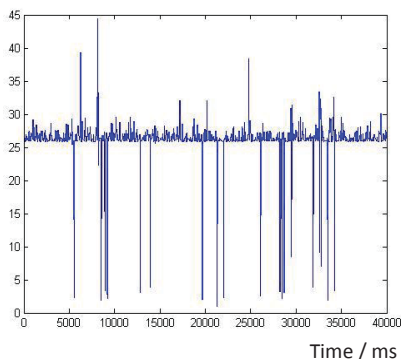


Figure 5.1: The time it takes to send messages with Nagle's algorithm enabled

Latency / ms

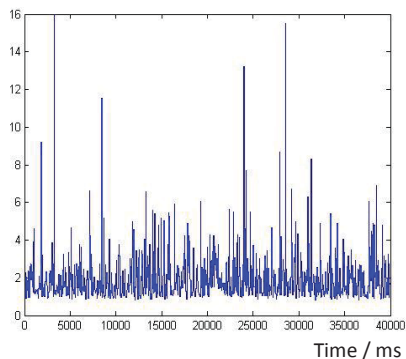


Figure 5.2: The time it takes to send messages with Nagle's algorithm disabled

UDP

A UDP (user datagram protocol) socket is theoretically much simpler than a TCP socket in the way that no connection needs to be established between a server and a client.²⁶ The user has to create the packets himself and then send them to an IP address and a port. There is no guarantee that the packet arrives or that it arrives in the right order. The advantages of UDP compared to TCP are that it's somewhat faster, as there is no resending or reordering of packages. It is better to use UDP when the problems mentioned above are not an issue.

3G

To make communication with and steering of the quadcopter possible over the 3G network, the problems with NAT must be solved. Furthermore, the latency cannot be too high. In Figure 5.3 the latency is measured over a 3G connection in the same way as Figure 5.1 and 5.2.

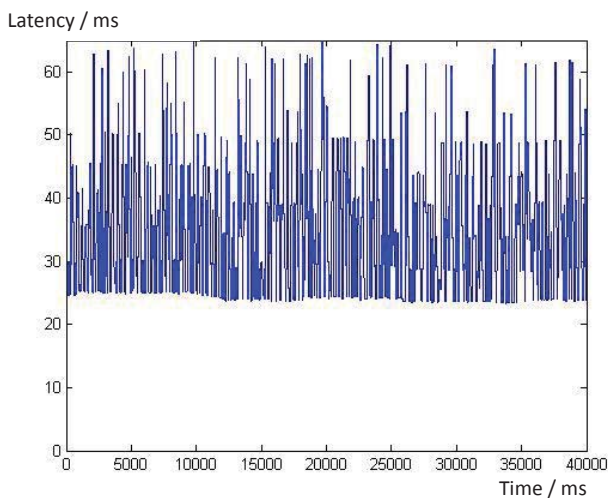


Figure 5.3: The time it takes to send messages over a 3G network, with Nagle's algorithm disabled. The latency is at most about 60 ms which is considered low enough for steering the quadcopter.

5.2 Implementation of Android-Computer Communication

In the communication between the Android and the PC there are three sockets on the smartphone, one in the Matlab script and two in the Java program (figure 2.1). The communication with the Matlab script is made using the TCP protocol, since it is important that all the commands sent from the Matlab script arrive.

The steering is implemented in the Java program. It communicates with a TCP connection, as the steering is event driven and every event is important to arrive. For convenience the same TCP connection is used to send log data to the computer.

The video streaming on the other hand is implemented through a UDP communication, for reasons described in section UDP. Mainly that it is ok if a frame is lost. It is more important that the new one arrives instead of the old one being resent. The sending of frames is done by first compressing the image, then splitting the compressed image into many packets. A header is added to every packet, that describes how big the packet is, what order that packet arrives in and how many packets there are building up the current frame. If some packets that build up a frame arrive in the wrong order, the whole frame is regarded as corrupt, and therefore not decompressed and displayed. If all the packets arrive and in the correct order, the compressed frame is built up, decompressed and displayed.

Safety

If the connection between the PC and the quadcopter is broken without user input (if it is broken while flying), the quadcopter must stop flying and land by itself to avoid damage. This problem is solved by letting the Java program send a “-“ sign ten times a second (heartbeats), which is usually referred to as “watchdog”. If the smartphone does not receive a message in 1.5 seconds, the connection is considered broken and the power to the motors is gradually decreased.

Protocol

The Matlab script and the Java program are implemented as servers and the smartphone as a client. When the Android application runs and is not connected to the PC, it continuously tries to connect two times a second. If the servers start to listen to their ports, the client connects and the Java server immediately starts to send “-“ signs. There are many commands the Matlab server can send to the Android. Each command has the following structure:

KEYWORD :12345.678

The keyword tells what the Android should do, for instance change a controller parameter. After the “:” sign it is a floating point number, saying for instance what value the new parameter should get. A list of all the commands can be seen in appendix A.2.

5.3 PWM

The circuit board and the motor controllers communicate via PWM (pulse width modulation). It is a form of signal that most microprocessor can generate. The signal is illustrated in figure 5.5.

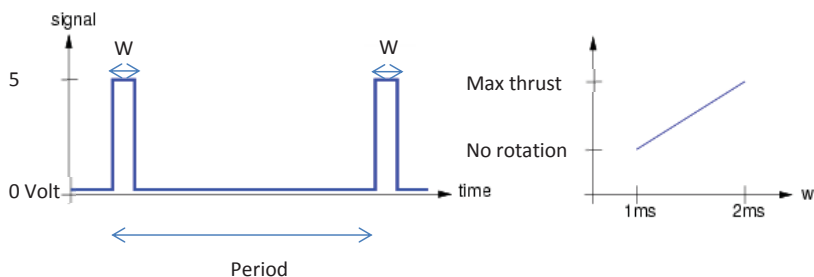


Figure 5.5: Description of PWM and relation between the PWM signal and the thrust of the motor. The thrust increases linearly with increasing width, W , of the pulses.

It is composed of rectangular pulses with varying length that appears periodically. It is the length of the pulses that decide how much thrust the motor will give. The motor controllers that were used in this project worked with pulses in the range from 1 ms to 2 ms. A pulse of 1 ms equals no speed and 2 ms equals full speed. The period of the PWM signal does not affect the thrust and should therefore be as short as possible. The shorter the period, the less delay there will be from the moment a command is sent from the circuit board to the reaction of the motor controller. For example, if the period is 100ms, then it is 100ms between the pulses and in the worst case scenario a delay of 100ms.

The period can't be arbitrarily short. This is because, the period can't be made shorter than 2ms. There is also a practical limitation of the motor controller. Some motor controllers can't accept a period less than a certain limit. For ordinary RC servos this limit is 20ms. But the limit in this project was set because the way the software was implemented on the processor that generates these PWM signals. An explanation of how the generation of PWM is implemented is described under *PWM generation*.

PWM generation

The PWM signal can be generated by the hardware of the microprocessor or implemented in the software. The ATmega88 can only generate two different hardware PWM signals at the required resolution. A software implemented PWM signal can be generated by a timer and two variables on the microprocessor. The timer is a variable that counts up every clock cycle. The microprocessor can use a prescaler, which is a setting that tells the timer to count up on a multiple clock cycle instead.

The two other variables are so called compare match variables, meaning that when the timer equals the value of one of the variables, an interrupt will be called.

The PWM is normally generated with the output pin set high (gives high voltage). When the timer reaches the value of the first interrupt variable (call it compare variable), the output pin is set to low. When the timer then reaches the second variable (call it top variable), the timer restarts from zero and the pin is again set to high. By setting the pulse length variable to the desired length of the PWM pulse and the period variable to the desired period of the PWM signal, an arbitrary PWM signal can be generated (with the constraint of the timer resolution and clock frequency). This procedure is illustrated in figure 5.6.

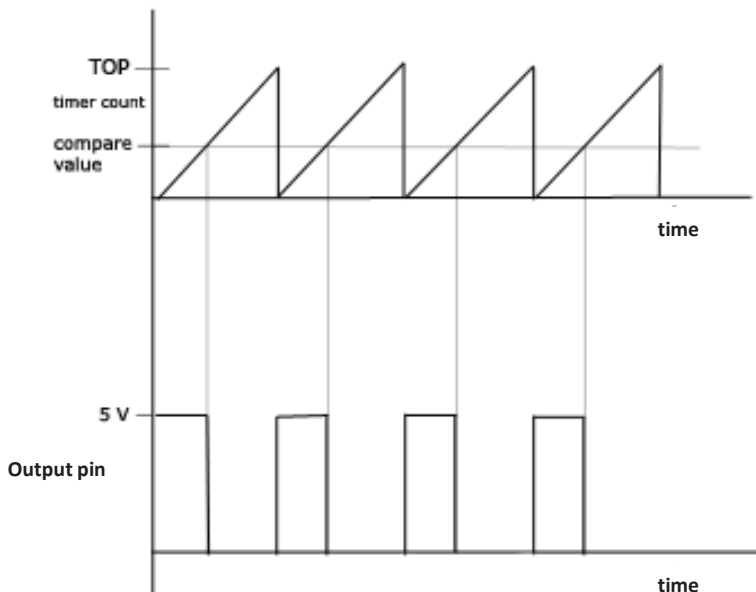


Figure 5.6: Generation of a PWM signal. The top graph represents the timer value. The lower graph represents the voltage on the output pin.

Multiplex PWM

If there were four different compare variables, four different PWM signals could be generated. Unfortunately the microprocessor used in this project (ATmega88) support only two compare variables, so an algorithm has to be implemented to generate more software PWM signals.

The way the four different PWM signals were generated in this project is illustrated in figure 5.7. There are four regular integer variables that have the value of the pulse length for the four different signals. When the timer starts the compare variable is set to the value of the first integer variable and the first pin is set high. When the first interrupt routine is called, the pulse length variable is set to the current time value added to the second integer variable, the first pin is set to low and the second pin is set to high. This continues on until the last pin is set low and the timer counts up to the top variable, from where the process repeats.

A limitation of this approach is that all the pulses have to fit into one period, meaning the period cannot be smaller than the sum of all the pulse lengths, which is at most 8 ms.

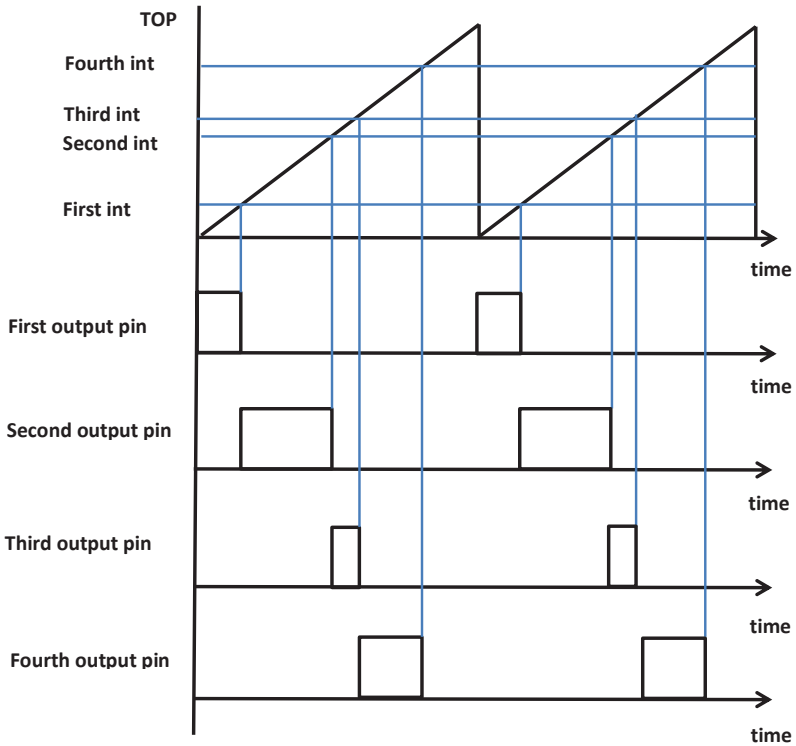


Figure 5.7: Generation of four different PWM signals. The top graph corresponds to the value of the timer. The other four graphs correspond to the voltage of the output pins. If for example the second output pin is high, then the compare variable is set to the timer value of when it should be set low and the third output pin should be set high. Notice that the period must be longer than the sum of all pulse lengths, so that they all “fit” into one period.

5.4 Android and Circuit board

The Android device can use several ways to send and receive information (Table 3.1). A microprocessor can receive and transmit data in various ways as well. (Table 4.1). This thesis uses the Universal Serial Bus (USB) on the Android to connect to the Universal Asynchronous Receiver/Transmitter (UART) on an Atmega88 (figure 5.4). An external clock is added to the circuit board to enable more precise PWM-generation. It increases the frequency of the ATmega88 from 8 MHz to 20 MHz. The UART implements the RS232 standard for serial communication, while the USB uses several layers of protocols. The translation between these interfaces is done with the PI2303 chip (Appendix A.2). Other solutions that were tested are the IOIO circuit board using Android debugging bridge (ADB) and Arduino using the Android Open Accessory.

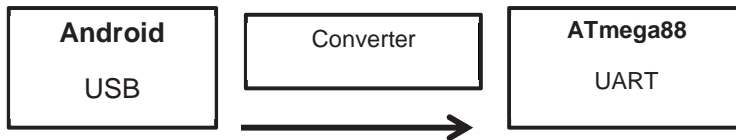


Figure 5.4: Android device and Atmega88 Communication.

Protocol

The data that has to be sent from the smartphone to the circuit board consists of information regarding motor identity and desired PWM pulse length. This communication is built around a protocol of five bytes, where the first byte decides what action and the following four bytes the values for that action if needed (see table 5.1). If the first byte is number one up to four, it represents the motor number and the following four bytes the PWM pulse length for that motor. If the first byte is a zero it calls for a loopback test and all the bytes are sent back. First byte as number five puts the microprocessor in control mode (start sending PWM signals to the motor controllers) and number six changes the PWM period (distance between the pulses) to the value given by the following four bytes.

First byte	
0	Loopback test All bytes are sent back
1	Set PWM for motor 1 Bytes 2 – 5 represents the PWM pulse length
2	Set PWM for motor 2 Bytes 2 – 5 represents the PWM pulse length
3	Set PWM for motor 3 Bytes 2 – 5 represents the PWM pulse length
4	Set PWM for motor 4 Bytes 2 – 5 represents the PWM pulse length
5	ON/OFF Control mode Byte 2 represents ON/OFF
6	Change PWM period Byte Bytes 2 – 5 represents the PWM pulse length

Table 5.1: Android - Microprocessor communication protocol

UART

The UART, Universal Asynchronous Receiver/Transmitter, transmits and receives serial information. It is integrated in the ATmega88.

The communication is Asynchronous, which means the smartphone and the circuit board use separate clocks and must instead agree on a transmission speed, the baud rate.

Baud rate, frequency and delay

Baud rate is the number of symbols sent per second. In this case the symbols consist of ones and zeros and the Baud rate is equal to the bit rate. Five bytes protocol, with 8 bits plus one start bit and one stop bit for each byte, gives a 50 bits length of a communication package. Four motors need four packages thus 200 bits. With a desired minimum of 100 updates per second the baud rate must be at least 20 000 bits/s. 20 000 bits/s and 200 bits of information give a reading time for the microprocessor of 10 ms which is too much for a 100 Hz communication. Delays from transfer, software, and hardware will add to the 10ms.

Custom made circuit board

The highest baud rate achieved in this project between the Galaxy S3 and the ATmega88 was 115200 bits/s. The resulting frequency is 575 Hz and the delay is 1.7 ms. The Android program limits the communication frequency to 100 Hz by adding a sleep in the control loop, which is done because the sensors samples at 100 Hz. A loop back test gives a one way delay of 4-6 ms which indicates the extra delay from hardware and software to be about 2.3-4.3 ms.

The baud rate error on the ATmega88 with 115200 bits/s UART communication and a 20 MHz frequency is 1.4 %. The error could be decreased with a clock more suitable for this purpose,²⁷ such as 18.4320MHz, which gives an error of 0% but was not available for this project.

IOIO

IOIO makes use of the Android phone's ADT (Android Debug Bridge) to communicate with the Android phone. The ADT protocol is used to program the Android phone and also to use the Logcat. The Logcat is a way to debug a program, by printing strings to the Eclipse console, if the Android device is connected to the computer. Since the protocol between the Android device and the IOIO is pre implemented in the supported library it is not possible to calculate the communication frequency the same way as with the custom made board. A loopback test gave a delay of 4-12 ms.

Arduino

The Arduino board used a USB protocol called open accessory, or ADK. It's a USB standard which was introduced to Android two years ago to simplify hardware development with Android. It was used with the same protocol as with the custom made board. A Loop back test gave a delay of about 1 ms.

Software Android

The Android application has a specific class to handle the communication with the circuit board. The information is sent and received to the USB by writing to a file that is automatically created when the circuit board is connected.

Software AVR

The microprocessor on the circuit board uses interrupts to handle the UART. When a signal is received on the Rx pin an interrupt occurs. The microprocessor reads the information from the register UDR0, checks the validity, and translates and executes according to the protocol (Table 5.1). The interrupt routine can be seen in code 5.1.

```
ISR(USART_RX_vect)
{
    nbrOfLoops=0;
    receivedByte=UDR0;
    if(receivedByte>9+48||receivedByte<0+48)
    {
        return;
    }
    message[currentByte]=(uint8_t)receivedByte;
    if(currentByte==4)
    {
        currentByte=0;
        motor = message[0]-48;
        if(motor>=1&&motor<=4)
        {
            motorSpeed[motor-1]=(message[1]-48)*1000+(message[2]-48)*100+(message[3]-48)*10+(message[4]-48);
        }
        if(motor==0)
        {
            for(int n=0;n<5;n++)
            {
                while ((UCSR0A & (1 << UDRE0)) == 0) {};
                UDR0=message[n];
            }
        }
        if(motor==5)
        {
            if(message[1]-48==1)
            {
                control0n=1;
            }else
            {
                control0n=0;
            }
        }
        if(motor==6)
        {
            OCR1A=((message[1]-48)*1000+(message[2]-48)*100+(message[3]-48)*10+(message[4]-48))*10;
        }
        connectionDecreaser=200;
        return;
    }
    currentByte++;
}
```

Code 5.1: Interrupt routine for UART and protocol reading.

6. Sensors

A smartphone uses a various number and types of sensors to acquire information. The Samsung Galaxy S3 used in this project features the sensors listed in table 3.1.

6.1 Sensors used in the quadcopter

This project makes use of the accelerometer and gyroscope data to estimate the rotation around the X- and Y-axis. Then these rotations are used to level the magnetometer. Finally the magnetometer and the gyroscope is used to calculate the rotation around the Z-axis (see figure 6.1).

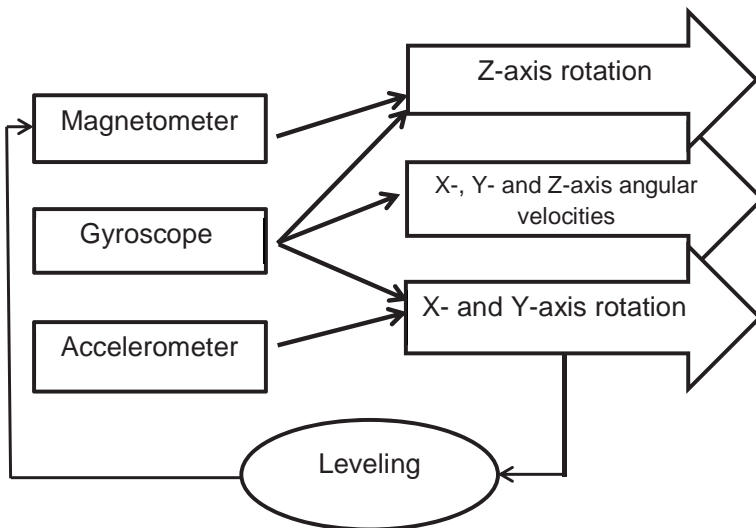


Figure 6.1: Usage of sensors.

Accelerometer

The accelerometer can be seen as a mass suspended on springs. It gives the acceleration by measuring force per unit mass of a weight within the sensor (Figure 7). When the device accelerates, the frame of the sensor moves and the weight within the sensor dislocates from equilibrium. The dislocation of the weight and the stiffness of the springs give the force on the weight (6.1). The force, and the mass of the weight gives the acceleration of the device (6.3).

$$F = -k * \Delta x \quad (6.1)$$

$$F = m * a \rightarrow a = F/m \quad (6.2)$$

$$a = (-k * \Delta x)/m \quad (6.3)$$

If the accelerometer is resting on a surface the accelerometer gives a constant value of -9.8 m/s^2 in Z-direction. This is the acceleration the surface gives on the device to counter the earth's g-force, $1g = 9.8 \text{ m/s}^2$. The accelerometer in a device during free fall will measure zero acceleration. With the constant value from the g-force, the accelerometer can be used to calculate the angle of the device with reference to the force (Figure 6.2). If the g-force acts in the default Z-direction, the rotation around the Y-axis can be calculated from the g-force's Z and X components on the sensor (6.4) and corresponding for the angle around X-axis (6.5).

$$Y_{angle} = \arctan(X/Z) \quad (6.4)$$

$$X_{angle} = \arctan(Z/Y) \quad (6.5)$$

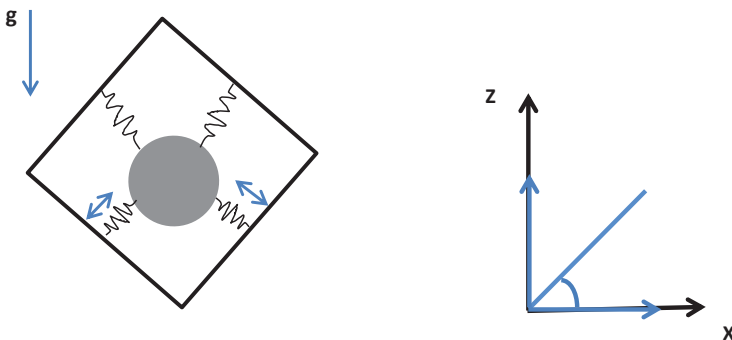


Figure 6.2. Accelerometer in 45 degrees rotation in the relation to the g-force.

A problem with using the accelerometer to estimate an angle, is that linear acceleration will affect the measurement. The assumption in this project is however that the quadcopter's tilt angles are close to zero at all times (see section 2.3 Modeling). The consequence of this is that it has no, or relatively small linear acceleration.

Gyroscope

A gyroscope measures the angular velocity and in reference to its own coordinate system and not to an external factor as with accelerometer (gravity) and magnetometer (magnetic fields). To obtain the angular orientation the signal has to be integrated.

Magnetometer

A magnetometer measures the strength and direction of the magnetic fields. The sensor can therefore use the earth's magnetic field, or other external magnetic fields, as a reference.

To be able to use the earth's magnetic field as reference for angular rotation around the Z –axis, the sensor has to be leveled, since the strength of the magnetic field will vary with rotation around Y- and X-axis as well. The values obtained from the accelerometer/gyroscope complementary filtered signal gives the level offset, and the rotational matrix to compensate for it (6.6). This enables calculation of the accurate magnetic field in the x/y – plane (6.7).

$$\begin{aligned} (m'_x, m'_y, m'_z)^T &= \\ &= \begin{bmatrix} \cos y & 0 & \sin y \\ 0 & 1 & 0 \\ -\sin y & 0 & \cos y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos x & -\sin x \\ 0 & \sin x & \cos x \end{bmatrix} (m_x, m_y, m_z)^T = \end{aligned} \quad (6.6)$$

$$= \begin{bmatrix} \cos y & 0 & \sin y \\ \cos x \sin y & \cos x & -\sin x \cos y \\ -\cos x \sin y & \sin x & \cos x \cos y \end{bmatrix} (m_x, m_y, m_z)^T \quad (6.7)$$

$$Z_{angle} = \arctan(m'_y/m'_x) \quad (6.8)$$

6.2 Sensor fusion

Sensors used for orientation of the device are accelerometer, gyroscope and magnetometer. These sensors have different characteristics, to get optimal orientation data for the device the outputs are combined with regard to their strengths and weaknesses, this is called sensor fusion. Sensor fusion can be implemented in various ways, in this project the complementary filter is used.

Complementary filter

The quadcopter's sensor fusion consists of two different complementary filters, one for the accelerometer/gyroscope fusion and one for the gyroscope/magnetometer fusion, see figure 6.3.

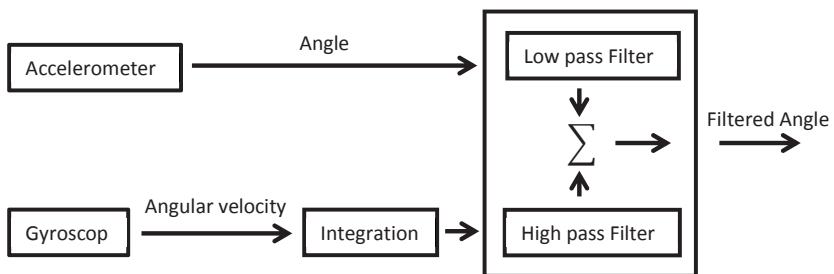


Figure 6.3: Principle of complementary filter

The low pass and high pass filters in a complementary filter must be each other's opposite, meaning that the sum of their frequency responses must at all frequencies be equal to one, see figure 6.4. If that wasn't the case, information at those frequencies would be distorted or lost.

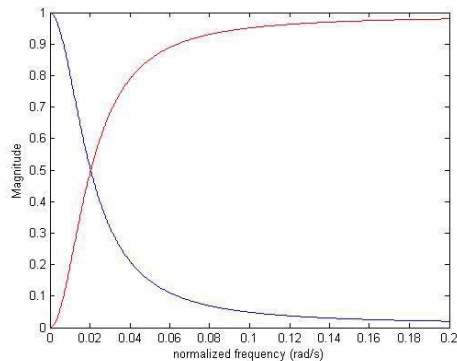


Figure 6.4: The sum of the lowpass and highpass filters frequency response in a complementary filter must equal to one. The red line represents the highpass filter and the blue line represents the lowpass filter.

The accelerometer's weakness is high frequency noise, which can be removed with a low pass filter. This removes most errors and makes the sensor accurate, but adds a weakness in making it unable to react to fast changes, see figure 6.5.

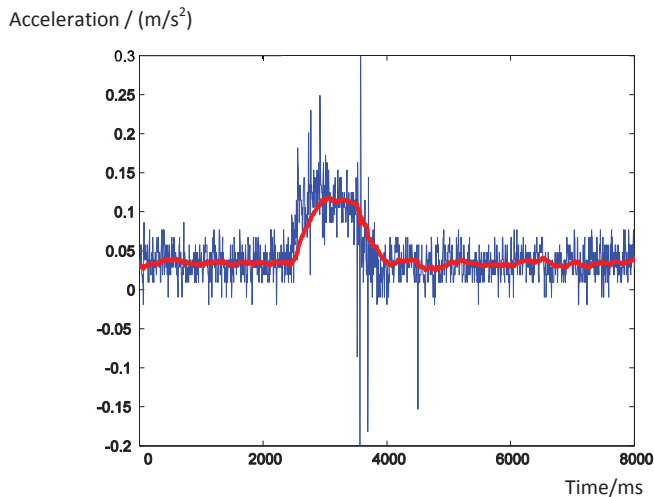


Figure 6.5: Accelerometer data. The blue line represents the unfiltered data with a variance of 3.3473×10^{-4} and the red line the lowpass filtered data with a variance of 9.6169×10^{-6} . After about 2.5 seconds the smartphone is tilted several degrees and then returned to its original orientation. It is possible to see a slight delay in the filtered data when the tilt is performed.

The Gyroscope on the other hand is fast and accurate but when integrated to get the angle, it has a drift error which makes the data increasingly inaccurate with time. This can be resolved with a high pass filter, see figure 6.6.

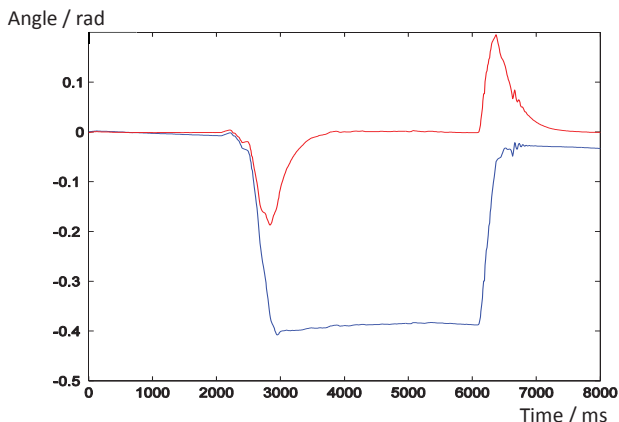


Figure 6.6. integrated gyro data is used to estimate the angle. Blue is unfiltered data and red is filtered data. After about 2.5 seconds the smartphone is tilted about 0.4 radians, held still for 3 seconds and then returned to its original orientation. Notice that the blue curve has value 0 at first, but it has declined to about -0.04 after 8 seconds.

When using accelerometer and gyroscope measurements, their respective filtered signals complements each other and produces both fast and long-term accurate data, see figure 6.7.

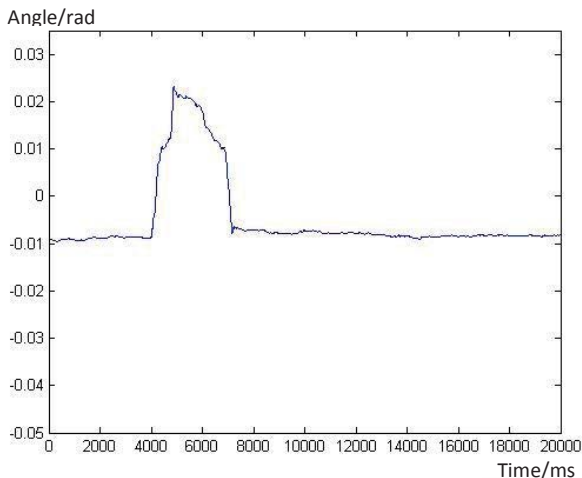


Figure 6.7: Accelerometer/Gyroscope Complementary filter. After about 4 seconds the smartphone is tilted several degrees and then returned to its original orientation.

The magnetometer has, similar to the accelerometer high frequency noise, which can be removed with a low pass filter, see figure 6.8.

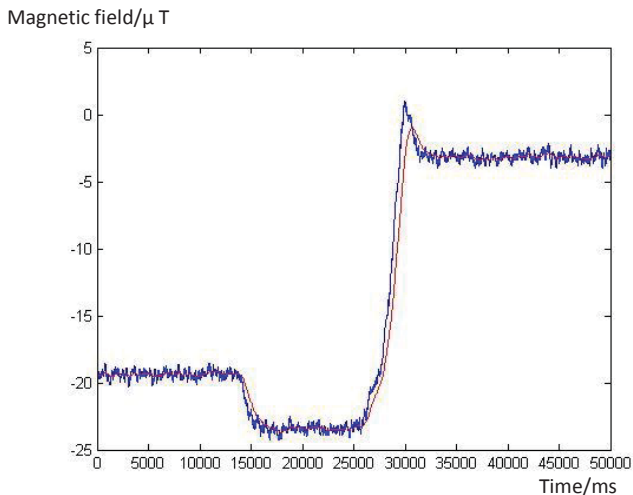


Figure 6.8: Magnetometer data. The blue line represents unfiltered data and the red line represents lowpass filtered data. After about 14 and 26 seconds, the smartphone is rotated about around the Z-axis.

When the magnetometer is combined with the gyroscope, it is filtered the same way as the accelerometer, see figure 6.9.

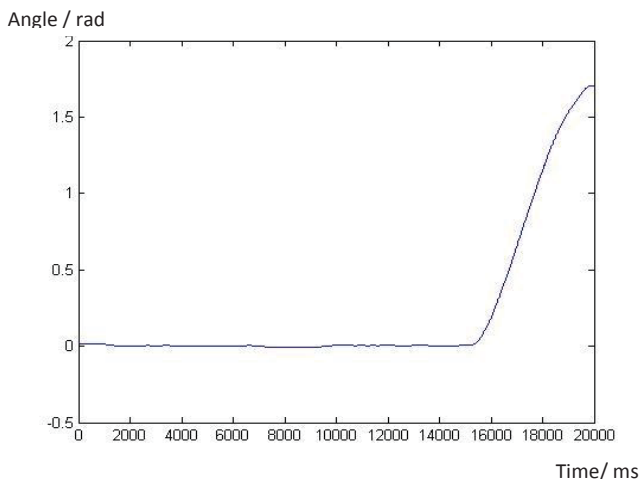


Figure 6.9: Magnetometer and Gyroscope complementary filter. After about 15.5 seconds the smartphone is slowly rotated 90 degrees around the Z-axis in the counter clockwise direction.

Derivation of complementary filter

A discrete filter is stable if all of its poles lie inside the unit circle.²⁸

A first order low pass filter has a pole on the real axis in the right half plane (between 0 and 1 in order to be stable). The closer it is to the unit circle, the higher the cutoff frequency will be.²⁹ The z-transform³⁰ of the simplest low pass filter will therefore be (6.9).

$$H_{LP}(z) = \frac{(1 - \alpha)}{1 - \alpha z^{-1}} \quad (6.9)$$

The corresponding high pass filter is

$$H_{HP}(z) = 1 - H_{LP}(z) = \frac{1 - \alpha z^{-1}}{1 - \alpha z^{-1}} - \frac{(1 - \alpha)}{1 - \alpha z^{-1}} = \frac{\alpha(1 - z^{-1})}{1 - \alpha z^{-1}} \quad (6.10)$$

The angle received from the accelerometer should be low pass filtered and the angle received from the gyro should be high pass filtered with (6.9) and (6.10) respective. The filtered signals should then be added together, as in (6.11).

$$\theta(z) = \frac{(1 - \alpha)}{1 - \alpha z^{-1}} A(z) + \frac{\alpha(1 - z^{-1})}{1 - \alpha z^{-1}} B(z) \quad (6.11)$$

where A is the angle calculated from the accelerometer data, and B is the angle calculated from the gyro data, after integration as in (6.12).

$$b(n) = h \sum_{k=-\infty}^n g(k) \quad (6.12)$$

Where g(k) is raw gyro data and h the sample period.

The z transform of B(n) is derived as

$$B(z) = h \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^n g(k)z^{-h} = \quad (6.13)$$

$$= h \sum_{n=-\infty}^{\infty} (g(-\infty) + \dots + g(n-1) + g(n))z^{-n} = \quad (6.14)$$

$$= h \sum_{n=-\infty}^{\infty} g(-\infty)z^{-n} + \dots + h \sum_{n=-\infty}^{\infty} g(n-1)z^{-n} + h \sum_{n=-\infty}^{\infty} g(n)z^{-n} \quad (6.15)$$

$$= hG(z) \sum_{n=0}^{\infty} z^{-n} = hG(z) \frac{1}{1-z^{-1}} \quad (6.16)$$

In (6.15) the z transform rule (6.17) was used and in (6.16) the expression for a geometric sum was used.

$$x(n-k) \xrightarrow{z} z^{-1}X(z) \quad (6.17)$$

Inserting (6.16) into (6.11) gives the complementary filter

$$\theta(z) = \frac{(1-\alpha)}{1-\alpha z^{-1}} A(z) + \frac{\alpha}{1-\alpha z^{-1}} hG(z) \quad (6.18)$$

By taking the inverse z transform of (6.18), the filter is expressed in the discrete time domain as (6.19).

$$\theta(n) = \alpha(\theta(n-1) + hg(n)) + (1-\alpha)a(n) \quad (6.19)$$

Time constant

The time constant of the filter is determined by choosing the parameter α and is defined as the time it takes for a filtered step to reach the value $1 - e^{-1}$.

The inverse z transform of (6.9) gives

$$\theta(z) = \frac{1 - \alpha}{1 - \alpha z^{-1}} A(z) \xrightarrow{z^{-1}} \theta(n) = \alpha \theta(n - 1) + (1 - \alpha) a(n) \quad (6.20)$$

The homogeneous solution of (6.20)

$$\theta_h: \quad \theta_h(n) = \alpha \theta_h(n - 1) \rightarrow \quad (6.21)$$

$$\theta_h(n) = \theta_0 \alpha^n \quad (6.22)$$

The particular solution of (6.20) is found by trying

$$\theta_p: \quad \theta_p = D \quad (6.23)$$

Inserting into (6.20) gives

$$D = \alpha D + (1 - \alpha) \rightarrow D - \alpha D = 1 - \alpha \rightarrow D(1 - \alpha) = 1 - \alpha \quad (6.24)$$

$$\theta_p = D = 1 \quad (6.25)$$

The total solution is the sum of the homogeneous and particular solutions.

$$\theta(n) = \theta^h + \theta^p = \theta_0 \alpha^n + 1 \quad (6.26)$$

The input signal is a step which leads to the initial condition of theta equals to 0.

$$\begin{aligned} \theta(0) = 0 &= \theta_0 \alpha^0 + 1 \rightarrow \theta_0 = -1 \\ \theta(n) &= 1 - \alpha^n \end{aligned} \quad \begin{array}{l} (6.27) \\ (6.28) \end{array}$$

The definition of the time constant

$$\theta(n) = 1 - e^{-1} = 1 - \alpha^n \rightarrow e^{-1} = \alpha^n \rightarrow n = \frac{-1}{\ln \alpha} \quad (6.29)$$

$$\tau = hn = \frac{-h}{\ln \alpha} \rightarrow \alpha = e^{-\frac{h}{\tau}} \quad (6.30)$$

Where τ is the time constant.

6.3 Implementation of filter

The choice of the parameters α (for the filters that estimates tilt angles) and β (for the filter that estimates the rotation around the Z-axis) is a tradeoff between how much of the angle estimation should come from the gyro and how much should come from the accelerometer, if a tilt angle is estimated, or magnetometer, if yaw is estimated. The higher the parameter, the more information comes from the gyro and the less variance the estimation will have. This comes at the cost of more biased estimation.

Accelerometer/Gyroscope Complementary filter

The accelerometer/gyroscope fusion filter is implemented in the Java code, see code 6.1.

```
filteredAngleY = (regParam.alfa)* (filteredAngleY + vals[1] *
(event.timestamp - timeFusion) *
0.000000001f) + (1-regParam.alfa)* (accAngleY);
```

Code 6.1: Accelerometer and Gyroscope Complementary filter for the Y-axis. The same principle applies for the X-axis.

Choice of α

Measurements of the angle around the Y-axis was performed with different α as seen in figures 6.10 to 6.15. The phone was first lying flat. After about four seconds the phone was tilted a small angle and then tilted back to the original position, whereupon the mean and variance is estimated. The result of the measurements can be seen in table 6.1.

Measurement	α	Variance	mean	Time constant
6.10	0.4	$2.7973 \cdot 10^6$	-0.0039616	0.0109
6.11	0.7	$1.5571 \cdot 10^6$	-0.0040342	0.0280
6.12	0.97	$1.9731 \cdot 10^6$	-0.0054117	0.3283
6.13	0.98	$1.5364 \cdot 10^7$	-0.0061776	0.4950
6.14	0.99	$1.3448 \cdot 10^7$	-0.0084724	0.9950
6.15	0.999	$2.2148 \cdot 10^5$	-0.0448112	9.9950

Table 6.1. Results from the measurements of the complementary filter for tilt estimation. The measurement indices correspond to the figures 6.10 – 6.15.

The result of the measurement implies that the desired α should lie between 0.97 and 0.99. Lower α results in to high variance of the angle estimation and to higher α results in to high time constant. The higher time constant results in poorer long term estimation of the angle, as seen in table 6.1, mean column (the smartphone was lying flat and the correct angle should be 0). For this project an α of 0.98 was chosen.

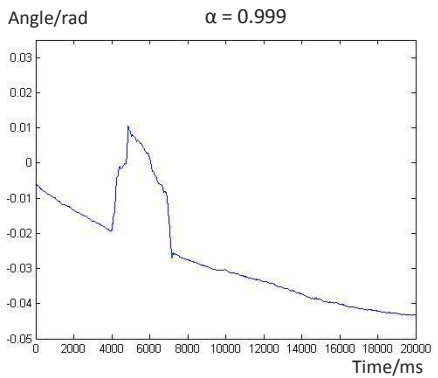
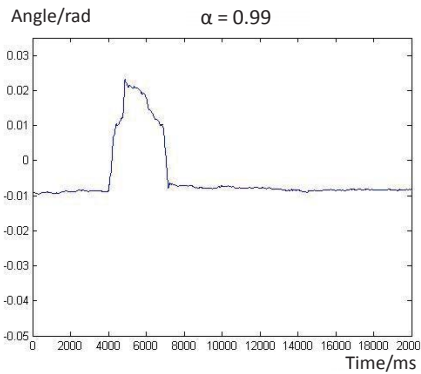
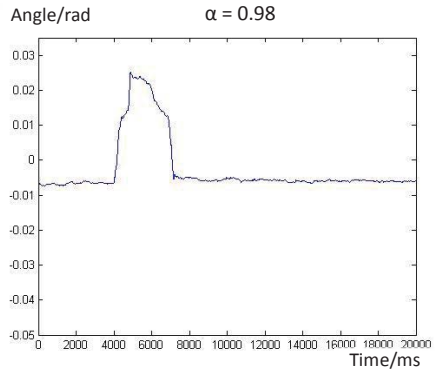
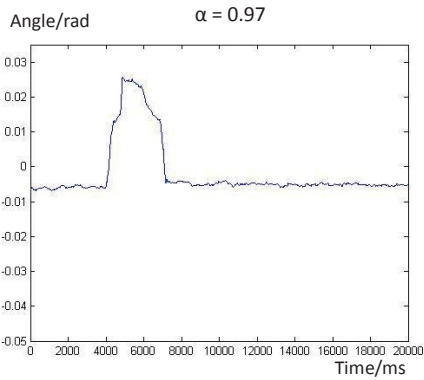
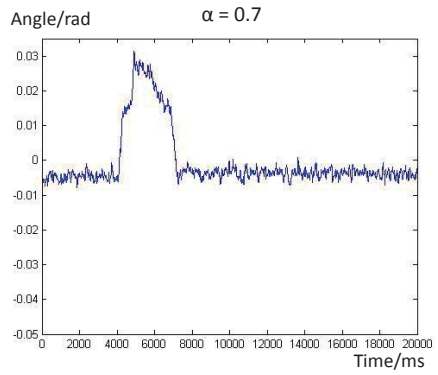
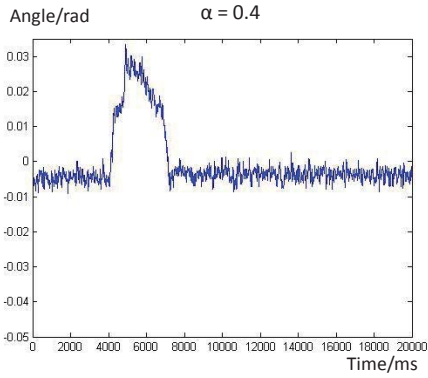


Figure 6.10 – 6.15: Accelerometer/Gyroscope Complementary filter with various α .

Magnetometer/Gyroscope Complementary filter

The magnetometer/gyroscope fusion filter is implemented in the Java code, see code 6.2.

```
filteredAngleZ = ( regParam.beta)* (filteredAngleZ + vals[2] *  
(event.timestamp - timeFusionZ)*0.00000001f) + (1-regParam.beta)*  
(magAngleZ);
```

Code 6.2: Magnetometer and Gyroscope Complementary filter for the Z-axis (Yaw).

Choice of β

Measurements of the angle around the Z-axis was performed with different β as seen in figures 6.16-6-21. The phone was first lying flat on a surface. Then the angle from the magnetometer was calibrated (see section 6.4, calibration). After about 15 seconds the phone was turned about 90 degrees around the Z-axis. The result of the measurements can be seen in table 6.2.

Measurement	β	Variance	mean	Time constant
6.16	0.4	$8.3483 \cdot 10^5$	0.0055945	0.0109
6.17	0.7	$2.8154 \cdot 10^5$	0.0042822	0.0280
6.18	0.97	$2.2518 \cdot 10^5$	0.0039458	0.3283
6.19	0.98	$1.6218 \cdot 10^5$	0.0033409	0.4950
6.20	0.99	$9.5418 \cdot 10^6$	0.0012268	0.9950
6.21	0.999	$5.0886 \cdot 10^5$	-0.0180366	9.9950

Table 6.2. Results from the measurements of the complementary filter for yaw estimation. The measurement indices correspond to the figures 6.16 – 6-21.

The results indicate that β should lie between 0.97 and 0.99. In this project β was chosen to 0.99 for similar reasons as of why α was chosen to 0.98 in the previous section

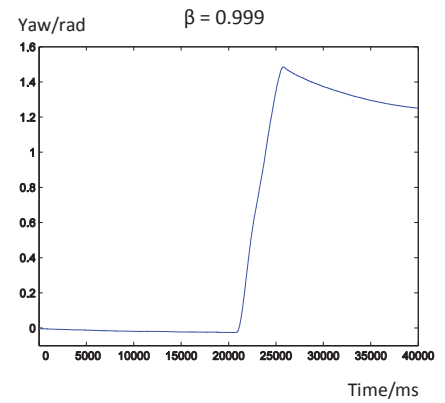
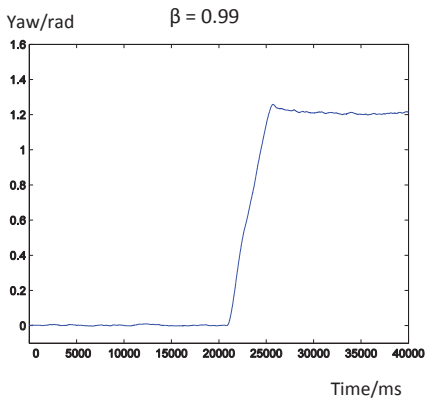
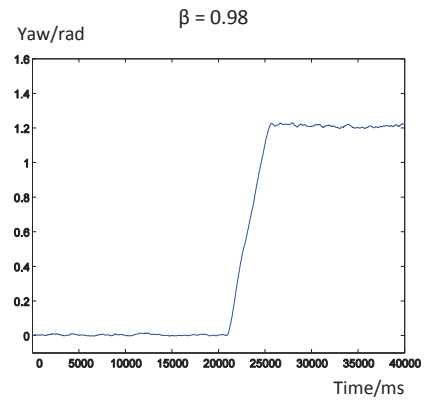
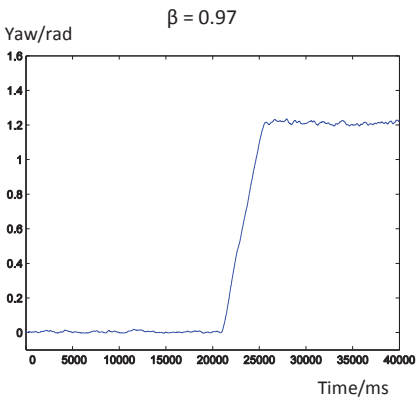
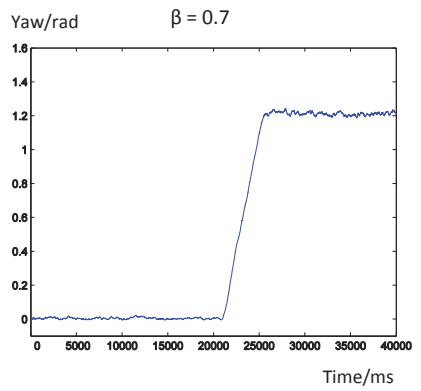
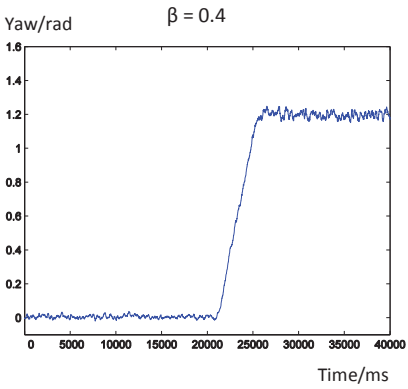


Figure 6.16 – 6.21: Magnetometer/Gyroscope Complementary filter with various β .

6.4 Calibration

X- and Y-axis

The hovering of a quadcopter depends on its ability to stay totally leveled and to reject external disturbances. To achieve this with as good control as possible the sensor data for angular position needs to be calibrated. This is done before flight, by placing the quadcopter on a flat surface and then subtract any occurring errors from the sensor data. The calibration is done on the data from the accelerometer/gyro complementary filter.

Magnetometer

In order to control and orient itself around the global Z-axis, the quadcopter needs a reference angle regarding rotation around its own Z-axis. The default Zero angle is when the quadcopter's Y-axis is pointing towards north. To set the Zero-angle to a certain direction, the quadcopter is positioned in that direction, the rotational angle from the magnetometer is estimated and then a rotation matrix is created which is used to transform the angle from the magnetometer. This is called calibration of the magnetometer in this report.

Motor compensation

The four motors on the quadcopter are of the same model. Usage and wear can result in damage and reduced performance on a motor, see figure 6.22. If there is a variation in the thrust/PWM ratio between the motors, it will be seen as a load disturbance in the control loop, the result will be a static error in the control of the quadcopter. This could be resolved with an integrator in the controller, but for large errors it is a better solution to manually compensate for the variation. The variation can be measured as described in section 2.3. with the relationship 2.1, stated here again for convenience.

$$T = k * P + m \tag{2.1}$$

A function is added in the android software, that makes it possible to compensate for reduced performance by manually adjusting the thrust to PWM pulse relationship of that motor. This is done by multiplying k and m with factors in software, see figure 6.23.

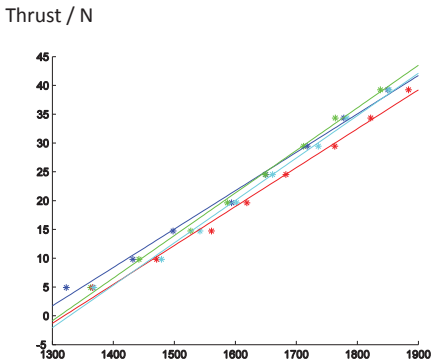


Figure 6.22. Motors thrust / PWM pulselength relationship before compensation

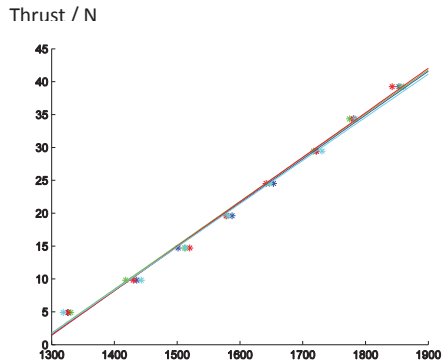


Figure 6.23. Motors thrust / PWM pulselength relationship after compensation.

6.5 Linear movement

The acceleration data, with compensation for gravity, can be integrated to get the velocity of the device (6.31), and then integrated again to get the position (6.32). In that way the accelerometer can be used to get data for linear movement and position. Since gravity is constant in linear directions, the accelerometer is unable to use it as a reference value and is dependent on consistent accuracy of the acceleration measurements. When integrating the acceleration data, the Bias will create a drift on the resulting velocity, see figure 6.24. The second integration will increase this errors resulting in a fast decline in accuracy of the position data, figure 6.24. In addition to this, it is stated in ³¹ that most of the position estimation error is due to error in the angle estimation, which is needed to know the direction of the linear movement.

$$v = \int a dt \quad (6.31)$$

$$x = \int v dt \quad (6.32)$$

Position / m

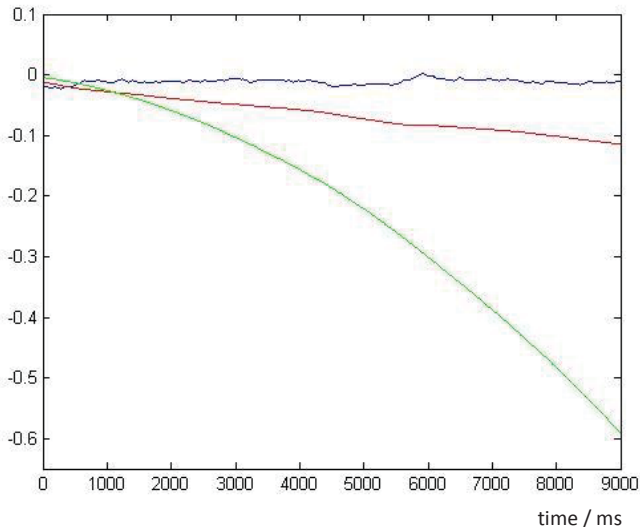


Figure 6.24: Acceleration(blue), velocity(red) and position(green)

Bias

Bias is the output of a sensor when the measured property is zero. It is not possible to completely eliminate the bias by estimating it and then remove that estimation from the sensor value. This is because the bias is not constant, but depends on factors as temperature.³² By letting the Android device lie still and collect accelerometer and gyro data, the bias can be estimated as in figure 6.25 and 6.26.

Acceleration / (m/s²)

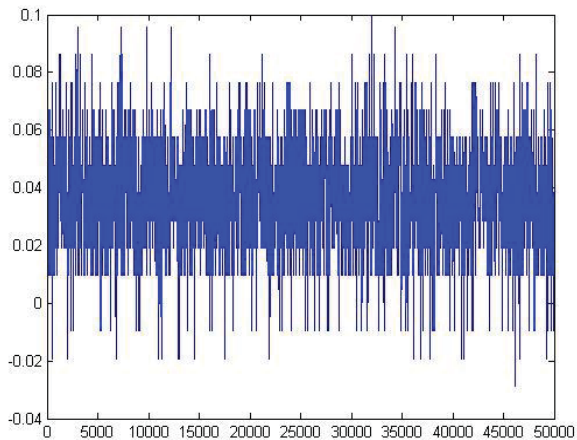


Figure 6.25: Accelerometer bias. The bias is estimated to be 0.0364 m/s^2 .

Angular velocity/ (rad/s)

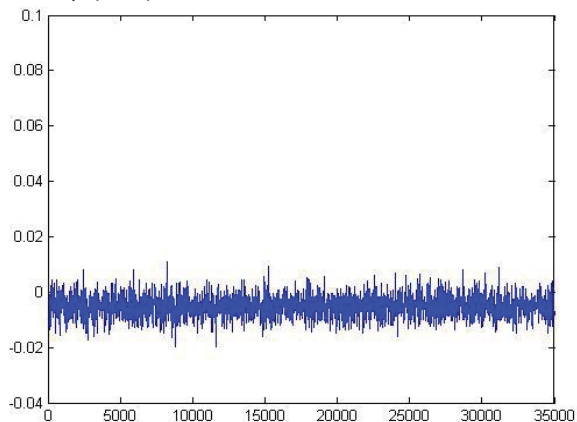


Figure 6.26: Gyroscope bias. The bias is estimated to be -0.0052 rad/s

7. Control

7.1 Theory

As the quadcopter is unstable, a controller is required to stabilize it. In fact, none of the degrees of freedom (absolute position and orientation) are asymptotically stable. Pitch and roll are unstable, while the x, y and z positions and the yaw angle are stable, but not asymptotically stable.

The most critical controllers are therefore the ones that control pitch and roll. Yaw is also important to control as a small disturbance will create a constant angular deviation, which makes it harder to steer.

Control design

The control design is performed in such a way that a model is first assumed, then a controller of two degrees of freedom is designed.

The model for the quadcopter is presented in section 2.3 Modeling. The simplified model for the pitch, roll and yaw is the following:

$$\ddot{\theta} = \frac{U_2}{I_{xx}} \ddot{\phi} = \frac{U_3}{I_{yy}} \ddot{\psi} = \frac{U_4}{I_{zz}} \quad (7.1)$$

where U_2 , U_3 and U_4 are torques generated by the actuators and serves as input signals to the process. As the states (θ , ϕ , ψ and their derivatives) and the input signals are decoupled according to this model, the design procedure can focus on a controller that controls one angle. The same controller can then be used to control the other angles, with some modification of parameters.

Dynamics of the orientation around one axis can be described as the transfer function

$$P(s) = \frac{1}{s^2 I} \quad (7.2)$$

Where s is the laplace variable and I the moment of inertia around an axis. The two degrees of freedom controller chosen, has a feed forward and a feedback part as seen in (7.3).

$$u = rR(s) - yT(s) \quad (7.3)$$

Where $R(s)$ and $T(s)$ are the feed forward and feedback transfer functions respectively. r and y are the reference signal and measurement signal, respectively.

The whole system can be seen in figure 7.1.

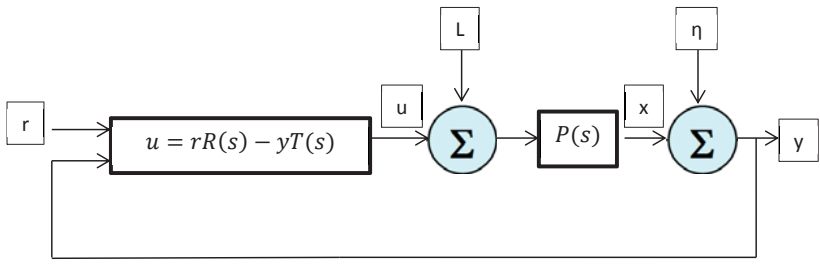


Figure 7.1: Overview of the system. r is the reference signal, u the control signal, L load disturbances, x outputs of the process, η measurement noise and y measured output signal. $P(s)$ represent the transfer function of the process.

The transfer functions from the different input signals to the states of the process are

$$G(s)_{r \rightarrow x} = \frac{R(s)P(s)}{1 + T(s)P(s)} = \frac{R(s)}{Is^2 + T(s)} \quad (7.4)$$

$$G(s)_{L \rightarrow x} = \frac{P(s)}{1 + T(s)P(s)} = \frac{1}{Is^2 + T(s)} \quad (7.5)$$

$$G(s)_{\eta \rightarrow x} = -\frac{R(s)P(s)}{1 + T(s)P(s)} = -\frac{R(s)}{Is^2 + T(s)} \quad (7.6)$$

$$G(s)_{\eta \rightarrow y} = \frac{1}{1 + T(s)P(s)} = \frac{1}{Is^2 + T(s)} \quad (7.7)$$

By observing the characteristic polynomial of the system, it can be seen that, if choosing T to a constant, the poles of the system would be placed on the imaginary axis. This would in practice make the system unstable. By choosing T to a first degree polynomial, the poles can be placed arbitrarily and make the system stable. Therefore T should be at least as (7.8) and the characteristic polynomial will become then become as (7.9)

$$T(s) = K_D s + K_p \quad (7.8)$$

$$Is^2 + K_D s + K_p \quad (7.9)$$

The roots can be chosen arbitrarily by changing the parameters K_p and K_D .

The static gain is defined as the amplification of a constant input signal after a long time, which is the same as the amplification of the zero frequency, $|G(0)|$. It is of interest to make the output signal follow the reference and make sure that load disturbances and measurement noise gets attenuated when the system is stationary. The following goals are therefore of interests:

- Make $G_{r \rightarrow x}(0) = 1$. The steady state error ($r-y$) will then be 0.
- Make $G_{r \rightarrow L}(0) = 0$. Constant load disturbances will be then attenuated completely.
- $|G_{\eta \rightarrow x}(s)|$ to be attenuated for high frequencies, as measurement noise usually consists of a lot of high frequency components

Inserting (7.8) in the transfer function from r to x, (7.4), provides (7.10) and inserting (7.8) in the transfer function from L to x, (7.5), provides (7.11) with their respective static gain (7.12) and (7.13).

$$G(s)_{r \rightarrow x} = \frac{R(s)}{Is^2 + K_Ds + K_p} \quad (7.10)$$

$$G(s)_{L \rightarrow x} = \frac{1}{Is^2 + K_Ds + K_p} \quad (7.11)$$

$$G_{r \rightarrow x}(0) = \frac{R(0)}{K_p} \quad (7.12)$$

$$G_{L \rightarrow x}(0) = \frac{1}{K_p} \quad (7.13)$$

It is evident that a static gain from r to x equals to 1 is achievable, by choosing R(s) as the constant K_p . But it is impossible to achieve a static gain from L to x equal to 0, if K_p is finite. Therefore T(s) must be chosen on the form

$$T(s) = K_Ds + K_p + K_I \frac{1}{s} \quad (7.14)$$

The two transfer functions now become

$$G(s)_{r-x} = \frac{R(s)s}{Is^3 + K_Ds^2 + K_p s + K_I} \quad (7.15)$$

$$G(s)_{L-x} = \frac{s}{Is^3 + K_Ds^2 + K_p s + K_I} \quad (7.16)$$

The static gain from L to x now becomes 0.

In order to make sure that the static gain from r to x becomes 1, R(s) is chosen to (7.17), which makes the transfer function from r to x equal (7.18)

$$R(s) = bK_p + \frac{1}{s}K_I \quad (7.17)$$

$$G(s)_{r-x} = \frac{bK_p s + K_I}{Is^3 + K_Ds^2 + K_p s + K_I} \quad (7.18)$$

This transfer function has a static gain of 1.

$R(s)$ could have been chosen to only

$$R(s) = \frac{K_I}{s} \tag{7.19}$$

and one still would achieve a static gain of 1, but with a drawback. If the reference changes, the feedforward term of the controller, $rR(s)$, would not be able to respond quickly enough, as it would take time for the integrator ($1/s$) to reach a control signal. But with a constant term bK_p , a control signal would immediately be created from the reference via the feedforward

The reason that the constant has a factor b , is because it opens up the possibility to place the zero in (7.18) arbitrarily and independently of the poles.

The transfer function from measurement noise to output is the same as from the reference to the output besides a minus sign. As the measurement noise is considered to be random noise with high frequency, it's important that the transfer function attenuates these high frequencies. This is the case of the system, as can be seen in figure 7.2.

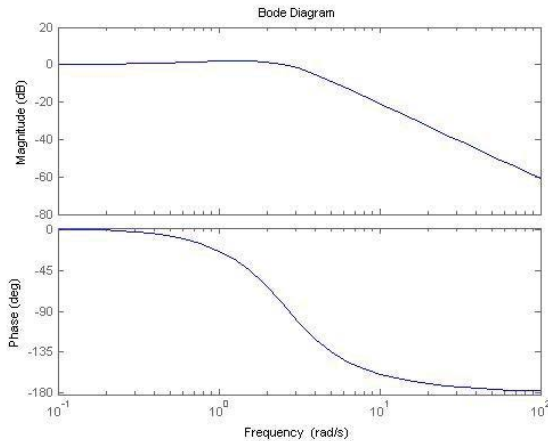


Figure 7.2: The bode diagram of the transfer function 7.18. The parameters used are $K_p=180$, $b=1$, $K_i=80$, $l=20$ and $K_D=80$;

The derived controller is shown in (7.20), (7.21).

$$u = rR(s) - yT(s) = r * \left(bK_p + \frac{K_I}{s} \right) - y * \left(K_p + sK_D + \frac{K_I}{s} \right) \quad (7.20)$$

$$u = (rb - y)K_p + (r - y)\frac{K_I}{s} - yK_Ds \quad (7.21)$$

Which is very similar in structure to a PID controller. Therefore theory of PID control tuning was used to determine the parameters K_p , K_I , K_D and b .

PID controller

The derived controller in the previous section (7.20), is similar to a classical PID controller,³³ as shown in (7.22)

$$u(s) = \left(K_p + \frac{K_I}{s} + sK_D \right) e(s) \quad (7.22)$$

The only differences are the existence of the b parameter and that the D part only acts on the measured signal. This structure however presents three major drawbacks:

1. It is improper, meaning that the numerator has a higher degree than the denominator, and therefore not physically realizable. The problem arises from the D part that magnifies arbitrary high frequencies.
2. The controller calculates the derivative from the error. If the reference changes rapidly, for example if it is a step, the derivative would become an impulse and the control signal would immediately saturate the actuators.
3. The saturation of the actuators combined with the I part can produce a nonlinear behavior. If some actuators are saturated, the I part will continue to integrate and the control signal will grow. When the error changes sign, the integrated value of the I part could be so large, that it takes a lot of time for the system to return to its ordinary behavior. This is called integrator wind-up.

The solutions for these problems in this project are the following:

1. A gyroscope is used to measure the angular velocity directly. Therefore no differentiation is needed.
2. The D part only acts on the output signal y.
3. Implement an anti-windup to the controller that makes sure that the integral term stops growing when an actuator is saturated. This can be accomplished in many ways, but in this project saturation was added to the integral, which limits its value to a predetermined limitation.

The PID controller used in this project can be visualized in figure 7.3.

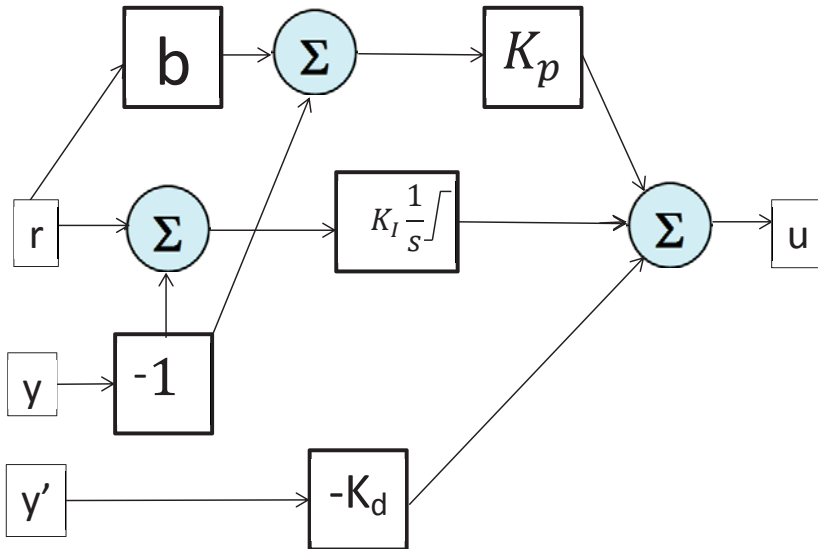


Figure 7.3: Visualization of the PID controller used in this project. For implementation of the integrator block with saturation, refer to code 7.2

The controller used does not support changing of parameters when controlling the quadcopter as no bump-less transfer is implemented. Instead the quadcopter must first land, before changes can be made.

PID tuning

It is possible to estimate parameters on the quadcopter such as moment of inertia and mass. The knowledge of these parameters makes it possible to derive a model based PID controller that is in some sense optimal for the quadcopter used in this project.

A drawback with the model based approach is that changes on the structure of the quadcopter (moving the battery, adjusting the position of the smartphone, wear of the motors etc.) will result in changes of the model. Because of these reasons, no parameter was estimated and instead the control parameters were tuned until desirable results was achieved.

7.2 Discretized PID controller

Because the PID controller in this project is implemented on digital hardware, it has to be discretized. No special attention to this issue has been made in this project besides that the angles and angular velocities are sampled, to create discrete control signals. On the Android, sampling of the sensors are driven by an interrupt. In the interrupt routine, filtering and sensor fusion is made to produce an approximation of the angular velocities and angles. These values are then used by a control thread every 10th ms, to create control signals that are immediately sent to the microcontroller.

The important parts of the control thread are the following:

Calculation of control signals:

```
controlY = (int) (regParam.TK * (regParam.refAngleY *
regParam.TPropb - sensors.getAngleY()) -
sensors.getAngleVelY() *
regParam.TD + IangleY * regParam.TI);
```

code 7.1. The control signal for the roll angle

Integration and saturation:

```
IangleY += (regParam.refAngleY - sensors.getAngleY());
if (IangleY*regParam.TI > regParam.TWindup) {
    IangleY = regParam.TWindup/(regParam.TI);
}
if (IangleY*regParam.TI < -regParam.TWindup) {
    IangleY = -regParam.TWindup/(regParam.TI);
}
```

Code 7.2. Implementation of windup

The messages being sent to the AVR:

```
comm.sendMessage(1+""+((int)((regParam.Thrust +
controlY+controlX+controlZ-2500)*regParam.M1extra+2500)
+regParam.M1m));
```

Code 7.3. The control signals are sent to the circuit board

8. Interface and software

8.1 Interface

The programs used in this project are

- The Matlab script, which is used to set parameters of the controller, filter, PWM period etc.
- The computer Java program, which is used to send reference signals to the Android device (to steer the quadcopter), as well as to receive logdata from the Android device and write it to a textfile.
- Kst2, which is a program that reads from the textfile the Java program is writing to and plotting the data in realtime.³⁴
- The Android program.

Matlab

The Matlab script is GUI based. The GUI can be seen in figure 8.1. It can be divided into five parts:

1. Set parameters of different functions of the Android program. The dropdown boxes selects the parameter, the sliders sets the parameters and the text displays the current value of the parameters.
2. Steering. By moving the mouse in the green area, the thrust of all the motors can be changed. By pressing the “f”, “g”, “h” and “t” keys, the reference of the two tilt angles can be changed. And by pressing the “i” key, the integrators of the controllers can be turned on and off.
3. Start and stop streaming as well as changing the camera settings.
4. A command can be sent by typing it into the box, see appendix A.2
5. Connect to the Android, enable the controller, start logging data, set a reference step, save and load parameter settings.

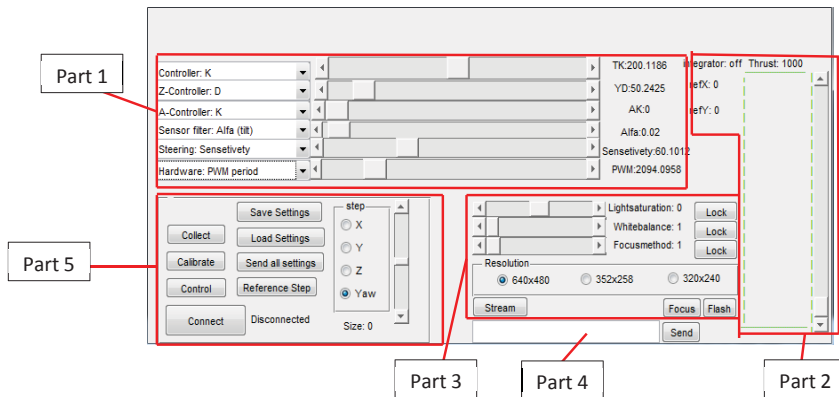


Figure 8.1: The Matlab GUI.

Java program

The Java program is started automatically when the Matlab script is started. The visual interface is just a display where live stream images can be shown, as seen in figure 8.2. It also listens for joystick inputs and writes logdata to a textfile.



Figure 8.2: The Java program running on the computer. Visually it's only a display that can show live images.

Kst2

A screenshot of Kst2 can be seen in figure 8.3. It can plot arbitrarily many datasets in real time by reading from a textfile.

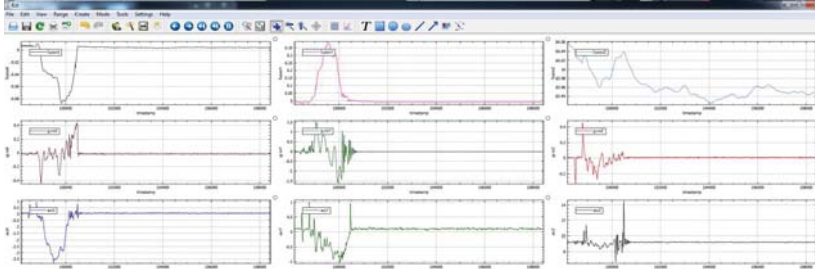


Figure 8.3: Kst2 screenshot, plotting 9 different datasets.

Android Program

A screen shot of the Android program can be seen in figure 8.4. An IP address can be inserted in the textbox. When one pushes the “set IP address” button, the program starts trying to connect to the Matlab script and the Java program running on that IP address.



Figure 8.4: The Android program

9. Results

Model

The simplified model of the quadcopter used in this project is presented in (2.7) and (2.8) restated below for convenience

$$\ddot{\theta} = \frac{U_2}{I_{xx}}, \ddot{\phi} = \frac{U_3}{I_{yy}}, \ddot{\psi} = \frac{U_4}{I_{zz}} \quad (2.7)$$

$$\ddot{Z} = -g + \cos \theta \cos \phi \frac{U_1}{m}$$

$$\begin{aligned} U_1 &= T_1 + T_2 + T_3 + T_4 \\ U_2 &= l(T_4 + T_1 - T_2 - T_3) \\ U_3 &= l(T_1 + T_2 - T_3 - T_4) \\ U_4 &= R_2 + R_4 - R_1 - R_3 \end{aligned} \quad (2.8)$$

The motors were not identical, which meant that compensation was needed. Figure 9.1 and 9.2 shows the motors thrust / PWM pulse length relationship before and after compensation respectively.

Thrust / N

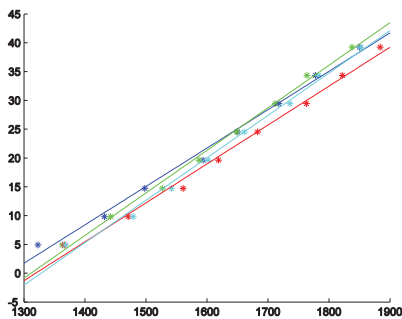


Figure 9.1. Motors thrust / PWM pulse length relationship before compensation

Thrust / N

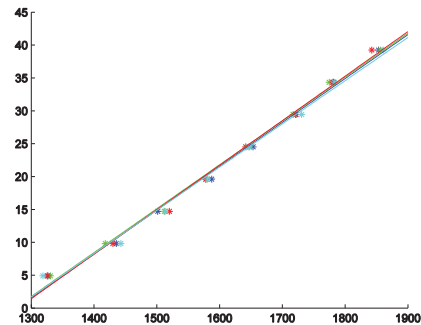


Figure 9.2. Motors thrust / PWM pulse length relationship after compensation.

There are some aspect that needs to be taken into consideration when using an Android phone for hard realtime tasks. One of them is java's garbage collector that makes the system nondeterministic. This is shown in figure 9.3, where it is clearly seen that the time to do a control cycle is nondeterministic and changes behaviour when objects are being allocated.

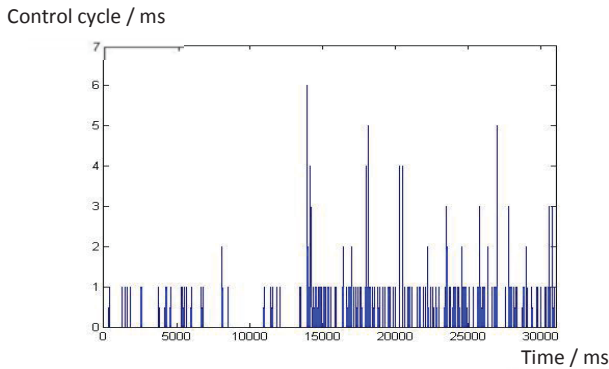


Figure 9.3: The time it takes to perform all calculations during a control cycle at different moments of time. All measured values are truncated. After 14 seconds some additional code is running in the control loop, with 10 objects being allocated every cycle. As seen in the graph the time to complete a cycle is now more often 1ms or above then before the 14 seconds mark. There are also several spikes. These could be a result of the garbage

Mechatronics

Three different circuit boards were tested. It was shown that the most simple of them, the custom made board, was enough to achieve satisfactory results.

communication

The communication between the PC and the smartphone was achieved via a wireless connection. The steering was done conclusively over a Wi-Fi connection, but comparison between a 3G connection shows promise in the possibility to steer over 3G, see figure 9.4 and 9.5. Even though the jitter and latency is larger over the 3G network, it is considered to be low enough to allow steering in real time.

Latency / ms

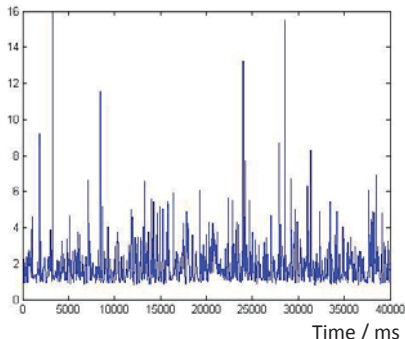


Figure 9.4: The time it takes to send a message with Nagle's algorithm disabled

Latency / ms

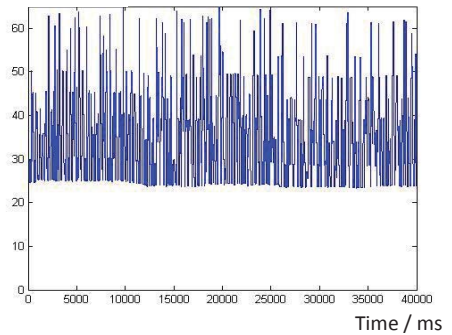


Figure 9.5: The time it takes to send a message over a 3G network, with Nagle's algorithm disabled

Sensors

It was shown in this project that the sensors used to control the quadcopter have different weaknesses. The accelerometer and magnetometer are noisy, while the gyro has a bias, which by integrating becomes a drift. The problem was solved by lowpass filtering the accelerometer (figure 9.6) and magnetometer and highpass filter the gyro (figure 9.7). The filtered signals were then added together to create a better estimation of the angle (figure 9.9). That is the principle of a complementary filter.

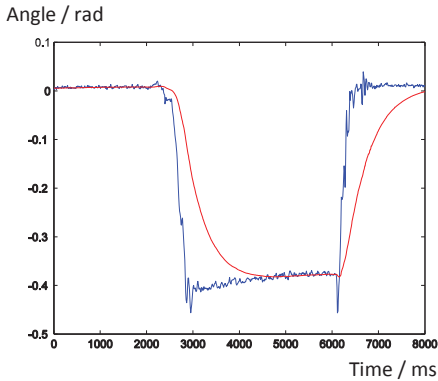


Figure 9.6. Accelerometer is used to estimate the angle. Blue is unfiltered data and red is filtered data.

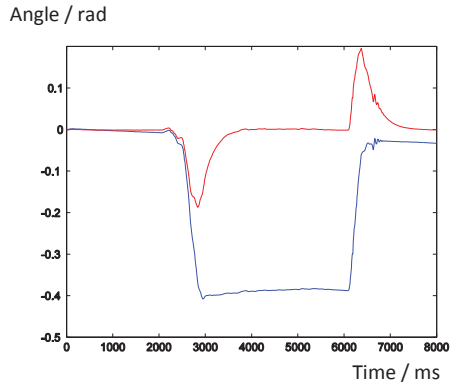
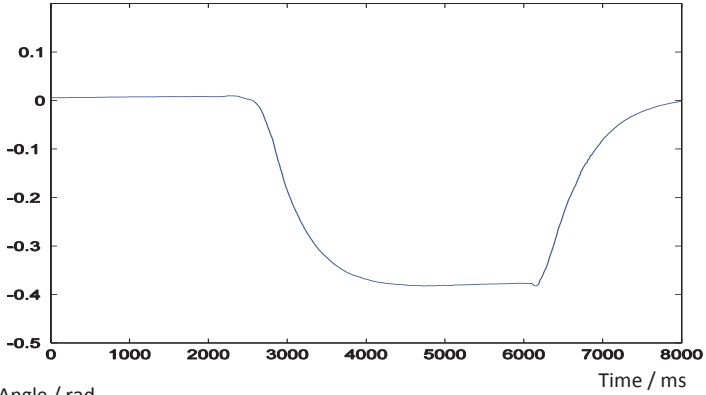


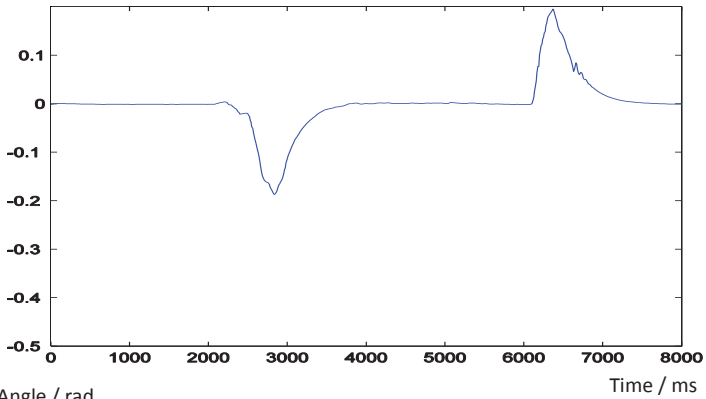
Figure 9.7. integrated gyro data is used to estimate the angle. Blue is unfiltered data and red is filtered data.

Angle / rad



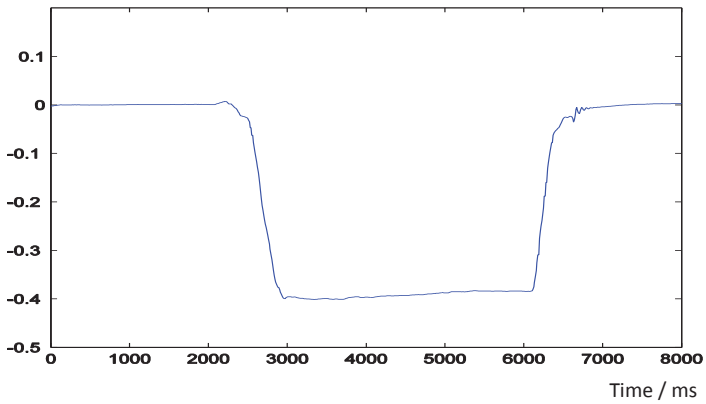
a

Angle / rad



b

Angle / rad



c

Figure 9.8. **a** is lowpass filtered accelerometer data that is used to estimate the angle. **b** is highpass filtered integrated gyro data. **c** is the sum of **a** and **b** and is the complementary filter.

Flight

After implementing each feature, the quadcopter was first tested on the balljoint. The result of one test can be seen in figure 9.9.

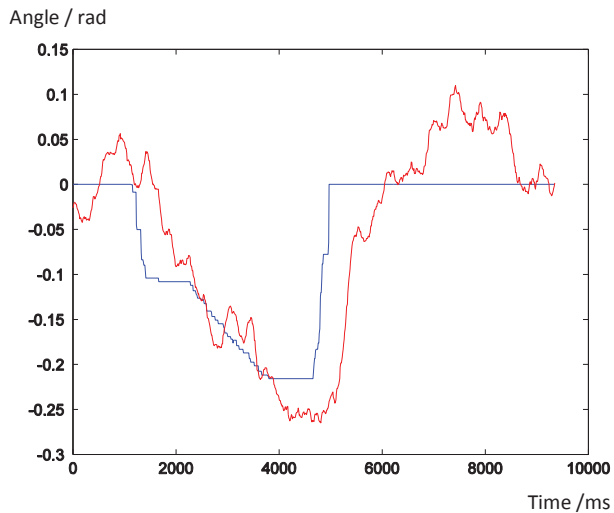


Figure 9.9. Quadcopter angle around the Y-axis. Blue: reference. Red: angle estimated with a complementary filter

When the behavior on the balljoint was satisfactory, a test flight was performed. The result is presented in figure 9.10, 9.11.

Angle /rad

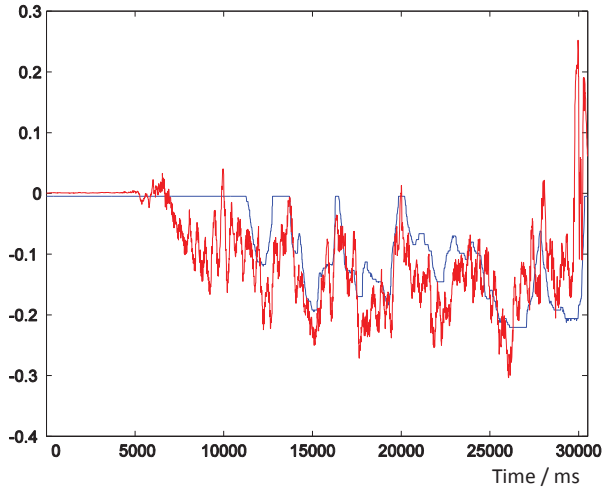


Figure 9.10. A testflight. The blue line represents the reference angle around the X-axis and the Red line represents the angle around the X-axis estimated with a complementary filter.

Angle /rad

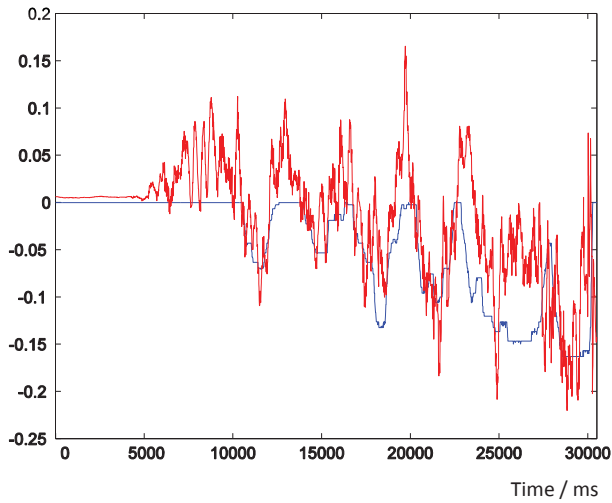


Figure 9.11. A testflight. The blue line represents the reference angle around the Y-axis and the Red line represents the angle around the Y-axis estimated with a complementary filter.

Theoretically it was possible to fly for as long as there was any power left in the battery, but there were some drift and the angle did not follow the reference very well. There were also disturbances of the magnetic field from the motors, which caused the magnetometer to provide incorrect estimates. These issues made the steering of the quadcopter harder, because the operator had to compensate for these issues instead of the controller.

The behavior on the balljoint was much better than in an actual flight and it is easier to see the relationship between the reference and the angle.

10. Conclusion

It is possible to implement a quadcopter using the sensors and computational power of a smartphone. The smartphone used was a Samsung Galaxy S3 with Android 4.1 Jelly Bean as operating system.

The quadcopter utilized the accelerometer, gyroscope and magnetometer of the smartphone. The sensor data was fused and filtered with two complementary filters. The sensors proved to be accurate and fast enough for the application. No external sensors were used.

Control of the quadcopter was implemented with a PID controller on the smartphone. The angular velocity for the D part was taken straight from the gyroscope data with no differentiation needed.

It was sufficient with a simple circuit board, based on an ATmega88, for PWM generation and motor controllers as external circuitry.

A computer running a Matlab script and a Java program was used to send commands to and receive data from the smartphone. The received data was plotted in real time with Kst2. The computer and the smartphone communicated using Wi-Fi which worked well as long as the network latency was low enough. It is easy to see that the angle follows the reference when the quadcopter is mounted on the balljoint. But it is harder to see the same when it is flying, even if there is some relationship. A reason for this behavior can be the hardware (motors and the frame) that creates large vibrations that disturb the sensors.

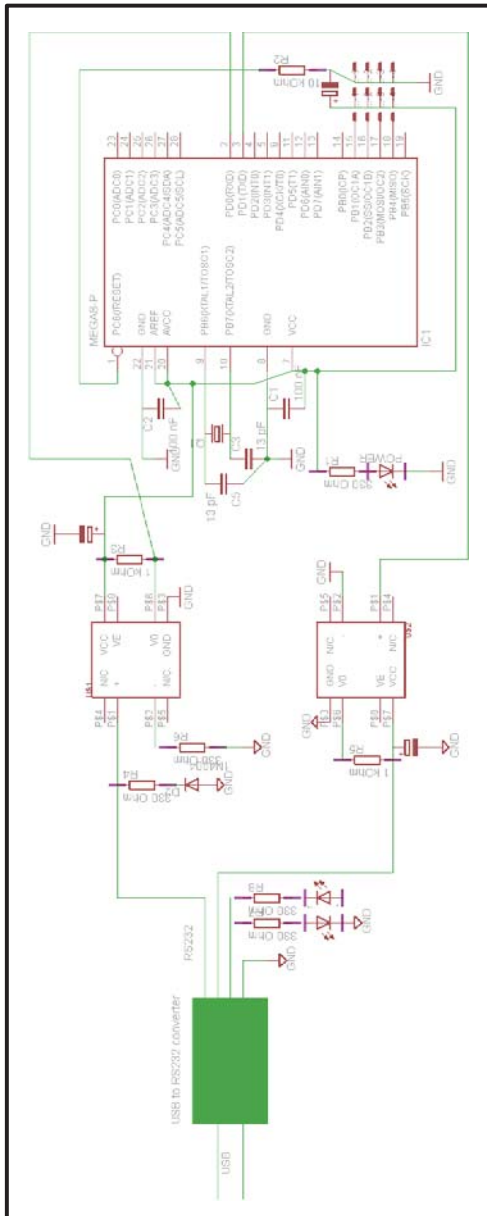
11. Future work

There are many possible improvements that can be implemented on the quadcopter. Among others they are:

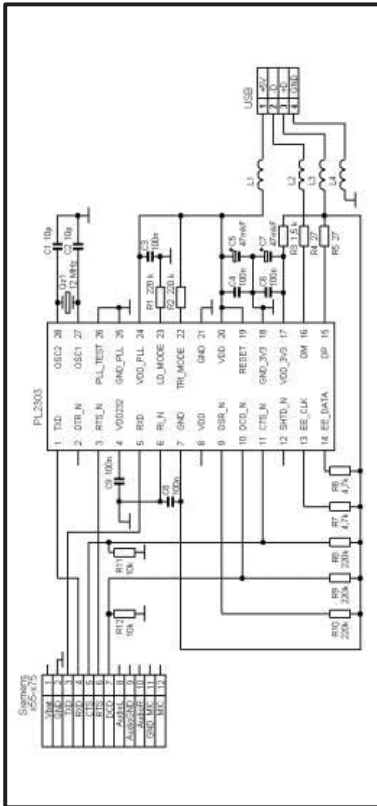
- Better control. The results should improve if the dynamics along the Z-axis is taken into consideration. This way the quadcopter could automatically maintain the same altitude, even if a tilt angle is changed
- Image processing. The camera in the smartphone used in this project can take about 16 frames per second at a resolution of 640x480. This is considered enough to do some minor image processing.
- Steer and stream images over a 3G network. It was shown in the report that the 3G network could be fast enough to provide the means to steer the quadcopter when it is far away.
- Better hardware. If the frame vibrates a lot from the motors, it will disturb the sensor readings.
- Compensations for the magnetic field generated by the motors. Every motor create a magnetic field that is dependent on the angular velocity it has. This phenomenon disturbs the magnetometer estimation of the angle. However, as the relationship is deterministic, it is possible to compensate for it.

Appendix

A.1 Schematics of Circuit boards



Custom made circuit board



PL2303 Chip

A.2 Commands

Network command protocol

Description	Keyword	number
Turn on / off the integrators of the tilt controllers	BID:	1 or 0
Sets the sensitivity of the steering.	Sensitivity:	0 to 200
Sets the PWM period	PWM:	1500-5000 (5000 represents a 20ms period and 1500 represents a 6 ms period.
Sets the static reference angle around the X axis.	AX:	(-30) – 30 representing the angle in degrees
Sets the static reference angle around the Y axis.	AY:	(-30) – 30 representing the angle in degrees
Sets the static reference angle around the Z axis.	AZ:	(-180) – 180 representing the angle in degrees
Start or stop streaming live images	SendImage:	1 or 0
Sets the tilt controller K value	TK:	0 to 500
Sets the tilt controller I value	TI:	0 to 500
Sets the tilt controller D value	TD:	0 to 500
Sets the tilt controller b value	TPorpb:	0 to 1
Sets the tilt controller saturation value	TWindup:	10 to 500
Sets the yaw controller K value	YK:	0 to 500
Sets the yaw controller I value	YI:	0 to 500
Sets the yaw controller D value	YD:	0 to 500
Sets the yaw controller b value	YPorpb:	0 to 1
Sets the yaw controller saturation value	YWindup:	10 to 500
Sets the parameter of the filter that estimates the tilt angles	Alfa:	0 to 1
Sets the parameter of the filter that estimates the yaw angles	Beta:	0 to 1
Sets the overall thrust of all the motors	Th:	2000 to 5000, (where 5000 corresponds to max thrust)
Sets the motor 1 k value	M1:	0 to 5

Sets the motor 2 k value	M2:	0 to 5
Sets the motor 3 k value	M3:	0 to 5
Sets the motor 4 k value	M4:	0 to 5
Sets the motor 1 m value	M1m:	100 to 400
Sets the motor 2 m value	M2m:	100 to 400
Sets the motor 3 m value	M3m:	100 to 400
Sets the motor 4 m value	M4m:	100 to 400
Turns on or of the LED on the phone	Flash:	1 or 0
Sets the light exposure compensation mode of the camera	LightExp:	1 to 5 depending on which mode
Focus the camera	Focus:	Can send any number
Locks the automatic light exposure compensator	LightExpLock:	1 or 0 (1 to lock and 0 to unlock)
Sets the whitebalace mode of the camera	Whitebalance:	1 to 5 depending on which mode
Locks the automatic whitebalance compensation of the camera	WhitebalanceLock:	1 or 0 (1 to lock and 0 to unlock)
Sets the focus method off the camera	Focusmethod:	1 to 5 depending on which method
Locks the automatic focus of the camera	FocusLock:	1 or 0 (1 to lock and 0 to unlock)
Sets the resolution of the images the camera is taking	NewRes:	1 to 3 (1 corresponds to a resolution of 640x480, 2 to 352x288 and 3 to 320x240)
The control loop starts to send commands to the circuit board.	Control:	1 or 0 (1 enables the sending and 0 disables it)
Starts to log and send data	Collect:	1 or 0
Calibrates different sensors	Calibrate:	gyro - calibrates the gyro fusionX – calibrates the angle around the x axis fusion – calibrates the angle around the z axis. acc – calibrates the accelerometer mag – calibrates the magnetometer all – calibrates all the sensors

-
- ¹ Gartner, Newsroom, 2013. <http://www.gartner.com/newsroom/id/2623415> (visited 2013-08-05)
- ² Business insider, Tech, 2013. <http://www.businessinsider.com/15-billion-smartphones-in-the-world-22013-2> (visited 2013-08-06)
- ³ Leishman, J.G. Principles of Helicopter Aerodynamics. New York, NY: Cambridge university press, 2000.
- ⁴ Stockholms smartphone, History. 2010. <http://www.stockholmsmartphone.org/history/> (visited 2013-08-12)
- ⁵ Garter, newsroom. 2013. <http://www.gartner.com/newsroom/id/2623415> (visited 2013-08-14)
- ⁶ Wikipedia, Comparison_of_smartphones , 2013. http://en.wikipedia.org/wiki/Comparison_of_smartphones (visited 2013-08-20)
- ⁷ Romo, meet romo, <http://romotive.com/meet-romo> (visited 2013-08-15)
- ⁸ Phonesat, 2013. <http://www.phonesat.org/> (visited 2013-08-15)
- ⁹ DJI, phantom, 2013. <http://www.dji.com/product/phantom/> (visited 2013-08-08)
- ¹⁰ Bresciani, Tommaso. *Modelling, Identification and control of a quadrotor Helicopter*, Master Thesis, Lund Institute of Technology, 2008.
- ¹¹ Open handset alliance, Android overview, http://www.openhandsetalliance.com/android_overview.html (visited 2013-08-10)
- ¹² Phandroid, Htc-dream, 2013, <http://phandroid.com/htc-dream/> (visited 2013-08-20)
- ¹³ Android, <http://www.android.com/> (visited 2013-08-21)
- ¹⁴ Samsung, galaxy S3, 2012. <http://www.samsung.com/global/galaxys3/specifications.html> (visited 2013-08-21)
- ¹⁵ Android, Developer resources, <http://developer.android.com/training/articles/perf-tips.html> (visited 2013-08-21)
- ¹⁶ Github, IOIO, 2013. <https://github.com/ytai/ioio/wiki> (visited 2013-08-12)
- ¹⁷ Arduino, 2013. <http://www.arduino.cc/> (visited 2013-08-12)
- ¹⁸ Atmel, atmega88, 2011. <http://www.atmel.com/devices/atmega88.aspx> (visited 2013-08-12)
- ¹⁹ USB, developers, 2011. <http://www.usb.org/developers/onthegeo> (visited 2013-08-12)
- ²⁰ Tech2.in, how-to, 2012. <http://tech2.in.com/how-to/accessories/how-to-make-your-own-usb-otg-cable-for-an-android-smartphone/319982> (visited 2013-08-10)
- ²¹ Github, IOIO, 2011. <https://github.com/ytai/ioio/wiki/IOIO-Over-OpenAccessory> (visited 2013-08-12)
- ²² Seeedstudio, depot, <http://www.seeedstudio.com/depot/ioio-for-android-p-1023.html> (visited 2013-08-12)
- ²³ Electrokit, Modules, <http://www.electrokit.com/en/arduino-mega-adk-rev-3.49226> (visited 2013-08-12)
- ²⁴ Microsoft, Technet, <http://technet.microsoft.com/en-us/library/cc940037.aspx> (visited 2013-07-25)
- ²⁵ Microsoft, Developer Network, 2006. <http://msdn.microsoft.com/en-us/library/ms883043.aspx> (visited 2013-07-25)
- ²⁶ Microsoft, Technet, [http://technet.microsoft.com/en-us/library/cc785220\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/cc785220(v=ws.10).aspx) (visited 2013-07-25)
- ²⁷ Atmel, atmega88, 2011. <http://www.atmel.com/devices/atmega88.aspx?tab=documents> (visited 2013-08-12)
- ²⁸ Proakis, John G. and Manolakis, Dimitris G. *Digital Signal Processing*, New Jersey: Pearson Prentice Hall, 2007, 168-192
- ²⁹ Ibid, 303
- ³⁰ Ibid, 141-200
- ³¹ Mannesson, Anders, *Joint Pose and Radio Channel Estimation*, Licentiate Thesis, Lund Institute of Technology, 2013.
- ³² Graves, Joseph Douglas, *Design and control of a vehicle for neutral buoyancy simulation*, Master Thesis, University of Meryland, 1997.
- ³³ Hägglund, Tore. *Reglerteknik AK Föreläsningar*, 2009.
- ³⁴ Kst, Kst-plot, <http://kst-plot.kde.org/> (visited 2013-08-10)

Lund University Department of Automatic Control Box 118 SE-221 00 Lund Sweden		<i>Document name</i> MASTER 'S THESIS	
		<i>Date of issue</i> September/December 2013	
		<i>Document Number</i> ISRN LUTFD2/TFRT--5936--SE	
<i>Author(s)</i> Hannes Bergkvist August Bjälemark		<i>Supervisor</i> Jerker Nordh, Dept. of Automatic Control, Lund University, Sweden Anders Robertsson, Dept. of Automatic Control, Lund University, Sweden (examiner)	
		<i>Sponsoring organization</i>	
<i>Title and subtitle</i> Quadcopter Control using Android-based Sensing			
<i>Abstract</i> <p>This thesis investigates the concept of a quadcopter implemented using the sensors and computational power of a smartphone. The main goal is to give an example of the possible applications of external hardware combined with commercial off the shelf (COTS) electronics. Practical problems that required attention were implementation of sensor fusion using the smartphone's sensors and the controller as a smartphone app. Attention was also paid to the real-time aspects of the Android operating system.</p>			
<i>Keywords</i>			
<i>Classification system and/or index terms (if any)</i>			
<i>Supplementary bibliographical information</i>			
<i>ISSN and key title</i> 0280-5316			<i>ISBN</i>
<i>Language</i> English	<i>Number of pages</i> 1-90	<i>Recipient's notes</i>	
<i>Security classification</i>			