CHALMERS

Stabilisering av autonom quadrotor

Examensarbete inom Högskoleingenjörsprogrammet i data- och elektroteknik

PATRIK LARSSON
VICTOR NEUGEBAUER

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Stabilisering av autonom quadrotor

Patrik Larsson, Victor Neugebauer

Institutionen för data- och informationsteknik
Chalmers tekniska högskola
412 96 Göteborg
Tel: 031-772 1000
Fax: 031-772 3663

Bilden på framsidan illustrerar quadrotorn som denna rapport baseras på.

Institutionen för data- och informationsteknik
Göteborg, 2014

Stabilisering av autonom quadrotor

P. LARSSON
V. NEUGEBAUER

Institutionen för Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Förord

Denna rapport är ett resultat av det examensarbete som utförts på Chalmers Tekniska Högskola i Göteborg under vårterminen 2014. Arbetet utfördes på plats i Chalmers lokaler.

Vi vill tacka vår handledare Sakib Sistek och Bertil Thomas för det stöd vi fått under arbetets gång. Samt övriga studenter och personal som intresserat sig för detta projekt.

Abstract

The different physical properties of the quadrotor have been observed through various step-response analysis. Four PID-controllers have been designed using Ziegler-Nichol's method. These controllers have been programmed into an Arduino Mega which is connected to multiple different sensors with a low sampling frequency of approximately 11 Hz. The stability of the finished product acheived the specifications set at the beginning of the project. There is also an option for an end-user to communicate with the quadrotor through a Wi-Fi connection with a basic user interface. A GPS circtuit has been mounted on the quadrotor, enabling navigation between different waypoints specified by GPS coordinates.

Sammanfattning

Genom olika stegsvarsanalyser har olika fysiska egenskaper hos quadrotorn observerats. Fyra regulatorer har utöver detta dimensionerats med hjälp av Ziegler-Nichols självsvängningsmetod. Dessa regulatorer har programmerats på en Arduino Mega som är kopplad till en rad olika givare med en låg samplingsfrekvens på ungefär 11 Hz. Stabiliteten hos quadrotorn uppfyllde de kraven som ställdes innan projektet drog igång. Utöver detta har man möjlighet att kommunicera med quadrotorn externt via Wi-Fi igenom ett enkelt användargränssnitt. En GPS-krets har även monterats på quadrotorn som möjliggör navigation mellan olika koordinater.

Ordlista

Quadrotor: En helikopter med fyra rotorer.

Roll: Rotation kring X-axeln.

Pitch: Rotation kring Y-axeln.

Yaw: Rotation kring Z-axeln.

Throttle: Förflyttning längs Z-axeln.

PID-regulator: Proportionell-, Integrerande- och Deriverande reglerteknisk metod för styrning av signaler.

Process *, Reglerteknisk*: Ett fysiskt system som skall styras.

Överföringsfunktion: Ett matematiskt samband som beskriver hur insignalen påverkar utsignalen i ett system.

Bodediagram Ett diagram som illustrerar fas- och förstärkningsegenskaper hos ett system med avseende på frekvens.

Gyroskop: Vinkelgivare.

Ultraljud: Ljud i högre frekvenser som människor inte kan höra.

GPS: Global Positioning System, positionering med hjälp av satelliter.

I^2C: Kommunikationsprotokoll utvecklad av Philips för inbyggda system.

Wi-Fi: Trådlöst kommunikationssystem.

PVC: Polyvinylklorid, en platssort.

Matlab: Datorprogram för matematiska beräkningar och simuleringar.

Telnet: Okrypterat, textbaserat kommunikationsprotokoll.

CLI: Command Line Interface. Icke-grafisk, textbaserad interaktion med program eller operativsystem.

Master: Den enhet på den seriella I^2C-bussen som styr klockfrekvensen.

Slave: Den, eller de, enheter som på den seriella I^2C-bussen som tar emot förfrågningar från Master.

Innehåll

Figurer

Tabeller

1 Inledning

1.1 Bakgrund

Projektet började redan för drygt ett år sedan då en projektgrupp i kursen DAT065 vid Chalmers tekniska högskola påbörjade arbetet att från grunden skapa en helt autonom quadrotor-enhet. Mycket av detta arbete resulterade i en väl bekantskap med hårdvaran, men aldrig i en helt autonom enhet. Däremot lyckades de stabilisera höjden och även pitch och roll något.

1.2 Syfte

Syftet är att förbättra stabiliseringen hos quadrotorn samt integrera ett GPS-system som ska kunna användas för att navigera till givna koordinater.

1.3 Mål

Målet med detta arbete är att undersöka hur man stabiliserar och navigerar en quadrotor samt hur man effektiviserar stabiliseringsberäkningarna av ett 3-axlat system med en mikroprocessor.

1.4 Avgränsningar

I projektet kommer endast färdiga komponenter att användas. Inga kretskort tillverkas utan redan färdiga utvecklingskort kommer att användas.
För att uppnå målet bör en rimlig avvägning mellan teoretiska modeller samt praktiska försök användas. Hypoteser verifieras lämpligen genom falsifikation, men kan även verifieras genom upprepade lyckade försök.

2 Metod

Arbetets gång kommer att ha ett sekventiellt utförande. Innan arbetet drog igång formulerades en uppsättning delmål, med syftet att strukturera upp arbetet. Delmålen nedan är dimensionerade så att de ska bygga på varandra.

2.1 Delmål

Delmål 1 Quadrotorn ska kunna starta och lyfta från marken till en bestämd höjd. Den ska även vara stabil i roll, pitch och yaw. Över Wi-Fi ska man kunna starta och stänga av maskinen.

Delmål 2 Quadrotorn ska kunna starta och lyfta från marken till en bestämd höjd. Den ska även vara stabil i roll och pitch, fast nu ska man även kunna ge den en ny höjd. Den nya höjden skickas till quadrotorn via Wi-Fi.

Delmål 3 Quadrotorn ska kunna starta, lyfta från marken samt manövrera från en position till en annan på ett stabilt sätt. Via Wi-Fi ska man kunna skicka en höjd som den ska befinna sig på och även enklare manövreringar, det vill säga, förflyttning fram, bak och i sidled.

Delmål 4 Quadrotorn ska kunna starta, lyfta och kunna förflytta sig på ett stabilt sätt genom steginstruktioner som ges via Wi-Fi. Tillägget är att den ska kunna rotera i 90 graders steg medurs eller moturs, om roteringsinstruktion skickas till den.

Delmål 5 Quadrotorn ska kunna manövrera stabilt i roll, pitch och yaw. Koordinater, eller instruktioner, ska kunna ges så att quadrotorn kan förflytta sig från punkt A till punkt B obehindrat. Detta ska ske med hjälp av ett GPS-system.

2.2 Mikroprocessor

Valet av mikroprocessor föll på Arduinos lösning med en färdig plattform inklusive processor som är framtagen för att vara enkel att använda för många olika typer av utvecklingsprojekt. Arduino-korten har många ut- och ingångar samt en snabb processor vilket kommer att lämpa sig för ett reglertekniskt system.

2.3 Stabilitet

Stabilitet kommer att uppnås genom blandade teoretiska beräkningar parallellt med Ziegler-Nichols tumregelmetoder. Den teoretiska metoden kommer att baseras på stegsvarsanalys för att kunna ställa upp en överföringsfunktion som kan användas för att dimensionera en rimlig PID-regulator. Dessa resultat kommer att jämföras med de resultaten som erhölls av Ziegler-Nichols metoder. Stabiliteten är grunden för ett lyckat projekt, därför kommer stor del av tiden att läggas på stabilisering samt att optimera stabiliseringen.

2.4 Kommunikation

Kommunikation till och från helikoptern kommer att ske via en Wi-Fi-anslutning. Användargränssnittet blev klart i en tidigare projektkurs, men behöver modifieras för att hantera de nya funktioner som tillkommer under projektets gång.

2.5 GPS

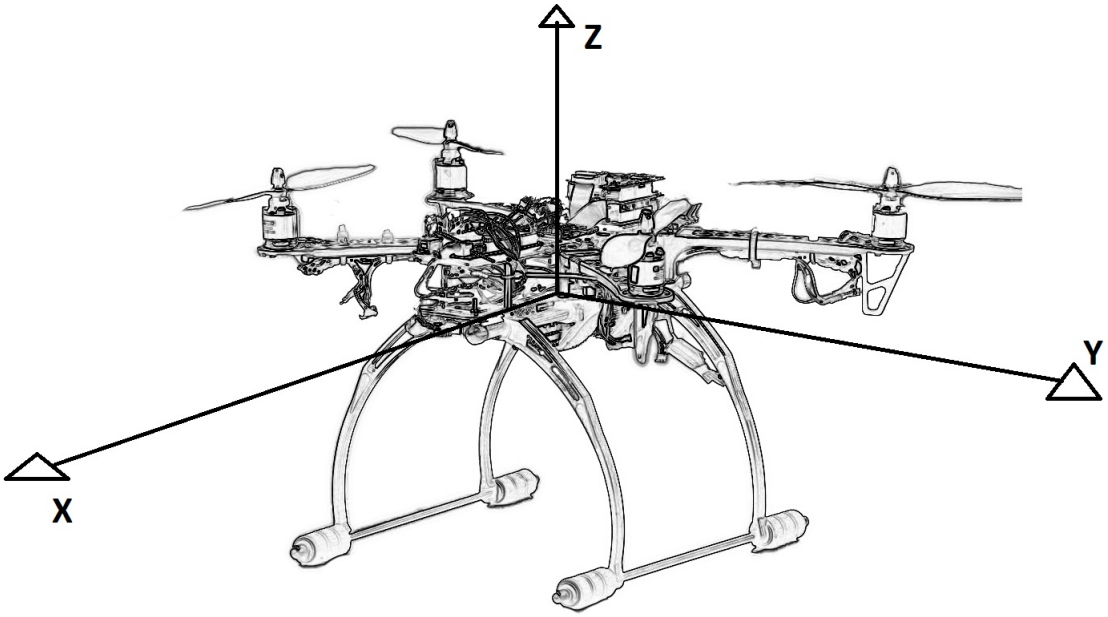
GPS-systemet kommer att undersökas samtidigt som vi försöker integrera detta i vårt projekt. För att uppnå detta kommer relevant litteratur att studeras för att få en tillräcklig teknisk förståelse.

3 Teknisk bakgrund

3.1 Grundläggande quadrotorteori

3.1.1 Grundläggande koncept

Figur 3.1 illustrerar hur quadrotorn är positionerad relativt X-, Y- och Z-axlarna i denna rapport. Rotationer kring X-axeln benämns roll, rotationer kring Y-axeln benämns pitch och rotationer kring Z-axeln benämns yaw. Förflyttningar längs X-axeln kommer vara fram och bak, förflyttningar längs Y-axeln kommer vara höger eller vänster och förflyttningar längs Z-axeln kommer vara ändringar i höjd.



Figur 3.1: *Illustration av quadrotorn samt de tre axlarnas positionering*

3.1.2 Design

En quadrotor, är som namnet antyder, en helikopter med fyra stycken rotorblad. Vid quadrotordesign är det till stor fördel om den är symmetrisk. En krysskonfiguration är vanligt förekommande i dessa sammanhang. De fyra rotorbladen är monterade på motorer som finns på varje vingspets. Figur 3.1 illustrerar quadrotordesignen som användes under detta arbete. Observera att quadrotorn i Figur 3.1 inte är fysiskt symmetrisk, det medför att det sätter extra press utvecklingen av effektiva stabiliseringsmetoder.

3.1.3 Aerodynamik

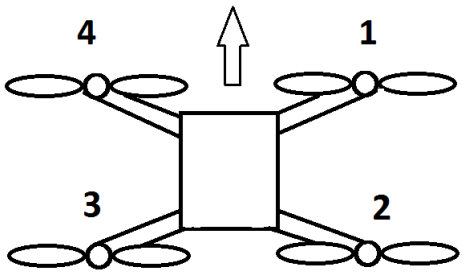
Rotationsriktningen hos motorerna är viktig. Framför allt måste motorerna delas upp i par, där de två paren har olika rotationsriktningar. Skälet till denna konfiguration är att de motsatta rotationerna tar ut vridmomentet som uppstår om alla motorer skulle ha samma rotationsriktning. Detta sätt förhindrar oplanerade rörelser i yaw, alltså rotation kring Z-axeln.

Utöver detta finns även ytterligare två aerodynamiska egenskaper som bör nämnas, nämligen markeffekten och takeffekten. Markeffekten är det fenomen som uppstår när en helikopter, eller flygplan, flyger nära marken. Om quadrotorn befinner sig nära marken tillåter inte rotorbladen luften att expandera på samma sätt under varje rotorblad som det gör högre upp från marken, vilket resulterar i en minskad motverkande luftström under rotorbladet. Rotorbladen får alltså mer lyftkraft närmare marken. Takeffekten fungerar nästan på samma sätt. Det uppstår alltså ökad lyftkraft hos rotorbladen när dessa är närmare ett tak, dock något lägre än den som uppstår av markeffekten, och kan potentiellt resultera i en krasch. [4]

Detta är ett väldigt djupt ämne som den nyfikna läsaren kan hitta mer information om i referenserna. [5, p.2]

3.1.4 Manövrering

Motorernas varvtal styr hur quadrotorn beter sig i luften. Genom att öka samtliga motorer med samma varvtal så kommer den att stiga i höjden. Genom att öka varvtalet på, till exempel, motorerna 1 och 2 samtidigt som man sänker varvtalet på motorerna 3 och 4 så börjar den en rörelse i roll åt vänster, motsatsen resulterar i roll åt höger. Genom att öka motorerna 1 och 3, eller motorerna 2 och 4 kan man styra yaw-rörelsen. Detta illustreras i Figur 3.2.



Figur 3.2: *Quadrotorns motornummer sett från ovan.*

3.2 Hårdvara

3.2.1 Arduinokort

Reglering av helikoptern kommer att ske genom att programmera utvecklingskort från Arduino. Dessa valdes på grund av användarvänligheten då arbetet ska fokusera på stabilisering och implementering av diverse periferienheter. För fullständiga specifikationer av dessa enheter, se Appendix C.

Arduino Mega Baserad på Atmel 2560-processorn med en 16 MHz kristalloscillator. Mega-versionen har 54 stycken digitala in- och utportar samt 16 stycken analoga inportar. [6]

Arduino Uno Baserad på Atmel 328-processor med en 16 MHz keramikresonator. Uno-versionen har 14 stycken digital in- och utportar samt sex stycken analoga inportar. [7]

Arduino Wi-Fi Shield Wi-Fi-enheten monteras på Uno-kortet för att möjligöra trådlös kommunikation via Wi-Fi mellan quadrotorn och en dator.

Adafruit Ultimate GPS Logger Shield GPS-enheten uppdateras kontinuerligt med en uppdateringsfrekvens på 10 Hz. Den här enheten, som monteras ovanpå Wi-Fi-kortet, används för att ge quadrotorn förmågan att veta var den befinner sig. [8]

3.2.2 Givare

MPU-9150 För att mäta variationer i lutning, acceleration och rotation används en nioaxlad givare, MPU-9150. Givaren skickar seriell data till Arduino Mega-kortet över I^2C-bussen där dessa värden används för att reglera och stabilisera quadrotorn. Givaren använder sig av det seriella kommunikationsprotokollet I^2C. Om ingen enhet på den seriella bussen är deklarerad som en Master tar MPU-9150 rollen som Master automatiskt. Om det redan finns en Master på bussen känner den av detta och blir per automatik en Slave. Adressen för MPU-9150 bestäms av spänningen på ingången AD0. Om ingen spänning existerar på AD0 sätts adressen till 0x68, finns där en spänning blir adressen 0x69. Det kan på detta vis existera maximalt två stycken MPU-9150 på en seriell buss.[9][10]

PING))) För att mäta variationer i höjdled används en ultraljudssensor från Parallax. Givaren består av en sändare som transmitterar ultraljudvågor och en mottagare som tar emot de utsända ljudvågorna. Ljudvågorna studsar mot närmsta

föremål och fångas upp av mottagaren. Tiden från sändning till mottagning mäts och avståndet beräknas i Arduino Mega-kortet. Denna givare har dock ett begränsat användningsområde då den maximala höjden den kan mäta ligger på cirka 3.5 meter.

FGPMMOPA6H Detta är GPS-enheten som är monterad på Adafruit Ultimate GPS Logger-modulen. Denna enhet har en noggrannhet på ner till 2.5 meter.

3.2.3 Motorer

Lyftkraft till konstruktionen fås av fyra stycken borstlösa trefasmotorer, T-Motor 900KV, där 900KV kan översättas till 900 varv per minut och volt. För att förhindra att quadrotorn roterar runt sin egna axel kopplas motorerna in så att de korsvis roterar åt olika håll. Motorerna får spänning från varsitt fartreglage som är den enhet som bestämmer spänningsmatningen.

3.2.4 Fartreglage

Fartreglagen får i sin tur PWM-signaler från Arduino Mega-kortet som översätts till en spänning för att driva motorerna. I detta projekt har Hobbywing flyfun 30A använts.

3.2.5 Stabilisator

FY-901, från FeiYu-Tech, är en flygstabilisator som är speciellt anpassad för stabilisering av quadrotorer. Den har ett inbyggt gyroskop som används för att bestämma signalen till fartreglagen. FY-901 får information, om hur den ska stabilisera quadrotorn, i form av PWM-signaler från Arduino Mega-kortet. Det intervallet av PWM-signaler som stabilisatorn accepterar finns angivna i Tabell 4.3.

3.2.6 Kraftkälla

Hela systemet drivs av Gravity 3300mAh, ett trecells, 12 volts litium-polymerbatteri. Batteriet har kapacitet nog att driva alla komponenter inklusive de fyra trefasmotorerna.

3.2.7 Konstruktion

Alla komponenter monteras på en kryssformad plastram, Reptile 500-V3, med tillhörande stödben. Motorerna fästs längst ut på ramens fyra armar och Arduinokorten samt gyroskop placeras i mitten. Undertill fästs batteri samt ultraljudsgivaren.

3.3 Kommunikation

3.3.1 Intern kommunikation

I2C I^2C är ett kommunikationsprotokoll utvecklat av Philips Semiconductors som definierar hur kommunikationen sker mellan enheter mjukvarumässigt och hårdvarumässigt. Grundprincipen är att alla enheter ansluter sig till en gemensam databuss, Serial Data (SDA), där seriell data skickas och tas emot. Utöver databussen ska även alla enheter vara kopplade mot en gemensam seriell klocka, Serial Clock (SCL). Denna buss är således ett tvåvägsgränssnitt (TWI) som möjliggör kommunikation mellan en mängd olika enheter. För att undvika kollisioner på bussen tilldelas en enhet Master-behörighet och övriga enheter Slave-behörighet. Vid uppstart är det "Master"-enheten som bestämmer klockfrekvensen på den seriella klockan och Slave-enheterna på bussen synkroniserar deras klockor efter denna frekvens. Master-enheten kan begära data från, eller skicka data till, en specifik enhet på bussen. Detta är möjligt eftersom alla enheterna har en unik 8-bitars adress.

3.3.2 Extern kommunikation

Extern kommunikation behövs för att kunna ge eller ta ut data från det slutna, interna system som sitter på helikoptern.

Radiovågor För att kommunicera trådlöst används elektromagnetiska vågor i olika frekvenser för att bära informationen. För att skicka data med vågen varierar man vågens egenskaper, t.ex amplitud, frekvens eller fas. När en av egenskaperna ändras kan mottagaren tolka detta och omvandla ändringen till en analog signal.

Wi-Fi Wi-Fi, eller "Wireless Fidelity", är en samling standarder för att med hjälp av radiovågor kunna kommunicera trådlöst med kompatibla enheter. Wi-Fi har hög överföringshastighet, över 54 Mbps med de nyaste standarderna, och lämpar sig därmed för kommunikation mellan persondatorer, mobiltelefoner och surfplattor. Wi-Fi är lämpat för lokala nätverk då räckvidden sällan sträcker sig över 30 meter inomhus.[11][12]

GPS Global Position System, GPS, utvecklades i början av 1970-talet av USAs försvarsdepartement för att bistå militären med ett satellitbaserat navigeringssystem. En GPS-mottagare kommunicerar med GPS-satelliter som cirkulerar runt jorden i en omloppsbana. GPS-satelliter skickar hela tiden ut en exakt synkroniserad tid som GPS-mottagare kan ta emot och jämföra med sin egna synkroniserade tid och differensen där emellan används för att beräkna avståndet till satelliten.

Tre satelliter avståndsbedöms och en fjärde används för att synkronisera tiden hos GPS-mottagaren.[13]

3.4 Reglerteknik

3.4.1 Grundläggande principer

I reglerteknik talar man ofta om *processer*, *system* och *återkoppling*. En *process* är något som kan påverkas, enkelt uttryckt. Tänker man närmare efter så är det mesta runt omkring oss processer. En process har oftast en insignal, i denna rapport kan detta vara till exempel motorerna, och en utsignal. Om vi håller oss till motorerna så har dessa en elektrisk insignal och en mekanisk utsignal i form av ett varvtal som är beroende av insignalen. Ett *system* beskriver hur flera processer samverkar. Dessa kan ha en, eller flera, insignaler och även en, eller flera, utsignaler. Ett system beskrivs oftast med blockscheman som följer insignalerna och beskriver hur dessa omvandlas för att bilda utsignaler. En *återkoppling* innebär att man tar utsignalen från en process och adderar eller subtraherar denna till insignalen.

3.4.2 Regulatorer

Regulatorer används inom reglertekniken för att erhålla en önskad utsignal, genom att reglera insignalen. Hur man väljer att dimensionera dessa regulatorer beror helt på processens egenskaper, samt krav på utsignalen. Genom att reglera signaler kan man skapa system som tolererar störningar av olika slag och ändå producerar satisfierande utsignaler. Till exempel används flera regulatorer i detta arbete för att erhålla stabila flygningar. De regulatorerna som kommer att användas i detta arbete kommer att vara av typen Proportionell, Integrerande och Deriverande regulator, även kallad PID-regulator. Matematisk kan denna regulator beskrivas enligt (3.1).

$$u_{PID}(t) = K[e(t) + \frac{1}{T_I}\int e(t)\ \mathrm{d}t + T_D e'(t)] \tag{3.1}$$

Där $e(t)$ är insignalen, $u(t)$ är utsignalen, K är förstärkningen, T_I är integrationstiden och T_D är deriveringstiden. Observera att denna matematiska modell beskriver en ideal PID-regulator.

3.4.3 Processidentifikation

För att kunna dimensionera regulatorer på ett korrekt och effektivt sätt krävs viss kännedom om själva processen. En effektiv metod att bestämma processens egen-

skaper är med hjälp av en stegformad insignal, som definieras enligt (3.2).

$$\sigma(t) = \begin{cases} 0 & : t < t_0 \\ 1 & : t \geq t_0 \end{cases} \tag{3.2}$$

Genom att ge en bestämd insignal till processen vid tidpunkten t_0 kan man studera processens egenskaper och bestämma en överföringsfunktion, det vill säga förhållandet mellan insignalen och utsignalen. Genom att bestämma processens överföringsfunktion kan man utgå från denna modell vid dimensioneringen av regulatorn. Det är viktigt att påpeka att oberoende av hur noggranna mätningar som erhålls så är dessa trots allt approximationer. Det gäller att försöka förutse allt som kan påverka modellen, som till exempel friktionen i testriggar som kan göra att modellen blir inkorrekt vid testningar. Att identifiera alla dessa motverkande faktorer är tidskrävande. En lämplig avvägning bör göras mellan fysiska tester och teoretiska modeller. En relativt exakt modell kan återskapas och användas som utgångspunkt för att sedan effektiviseras efter fysiska tester.

3.4.4 Regulatordimensionering

Regulatordimensionering är en konst. Det finns teoretiska metoder, det finns tumregelmetoder och det går även att dimensionera regulatorn approximativt med upprepade försök. Dock är den sistnämnda metoden lättare att genomföra om man kan basera försöken från ett utgångsläge.

Ziegler-Nichols svängningsmetod Denna metod baseras på att man hittar självsvängningsfrekvensen ω_0 med ren P-reglering. Man väljer en lämplig förstärkning, K_0 som sätter systemet i självsvängning genom att prova sig fram. Därefter använder man sig av sambandet (3.3).

$$T_0 = \frac{2\pi}{\omega_0} \tag{3.3}$$

Har man en bra matematisk modell över systemet kan även (3.4) med fördel användas för att bestämma självsvängningsförstärkningen.

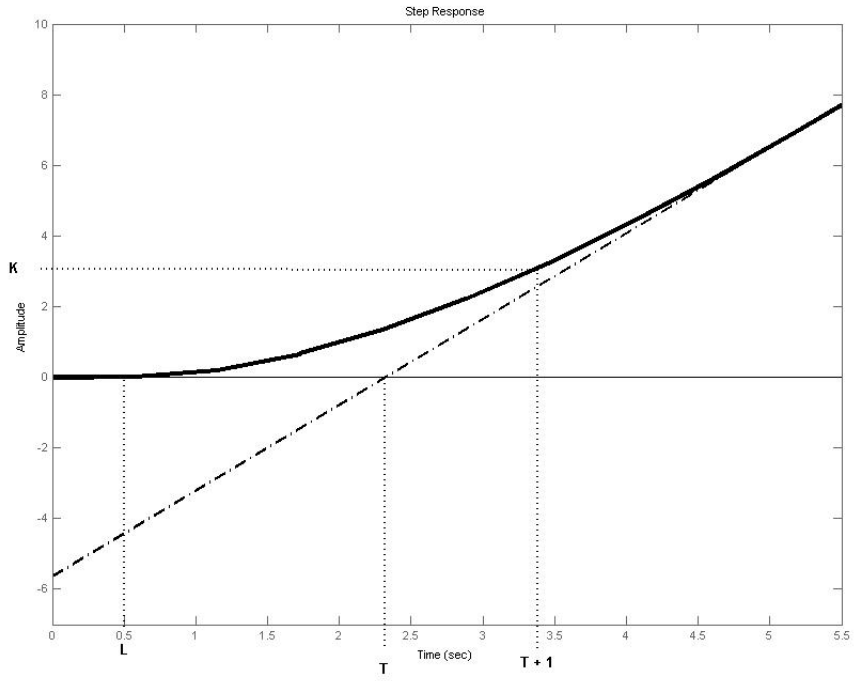
$$K_0 = \frac{1}{|G_p(\omega_\pi)|} \tag{3.4}$$

När dessa två parametrar har bestämts använder man den standardiserade tabellen 3.1 för att få regulatorns parametrar.[1][14]

Regulatortyp:	K	T_I	T_D
P-regulator	$0.5K_0$	-	-
PD-regulator	$0.45K_0$	$0.85T_0$	-
PID-regulator	$0.6K_0$	$0.5T_0$	$0.125T_0$

Tabell 3.1: *Tabell för dimensionering av regulatorer med Ziegler-Nichols svängningsmetod* [1, p.191]

Ziegler-Nichols stegsvarsmetod Ytterliggare en tumregelmetod av Ziegler-Nichols är stegsvarsmetoden. Denna baseras på parametrar som erhålls från stegsvaret. En förutsättning för denna metod är att stegsvaret görs på ett öppet system där det går att erhålla parametrarna K, L och T enligt Figur 3.3.



Figur 3.3: *Exempel på ett stegsvar där man kan bestämma K-, L- och T-parametrarna.*

När dessa parametrar har bestämts ställer man in regulatorns parametrar enligt tabell (3.2).[3]

Regulator Typ:	K	T_I	T_D
P-Regulator	$\frac{1}{K_P L}$	-	-
PD-Regulator	$\frac{0.9}{K_P L}$	$3L$	-
PID-Regulator	$\frac{1.2}{K_P L}$	$2L$	$\frac{L}{2}$

Tabell 3.2: *Tabell för dimensionering av regulatorer med Ziegler-Nichols stegsvarsmetod* [2][3]

Teoretisk metod En förutsättning för att kunna simulera systemet är att man har matematiska modeller som stämmer så bra som möjligt överens med verkligheten. Dessa kommer givetvis att inte vara optimala i verkligheten, men det ger ett stabilt utgångsläge för att sedan finjustera parametrarna baserat på observationer och upprepade försök.

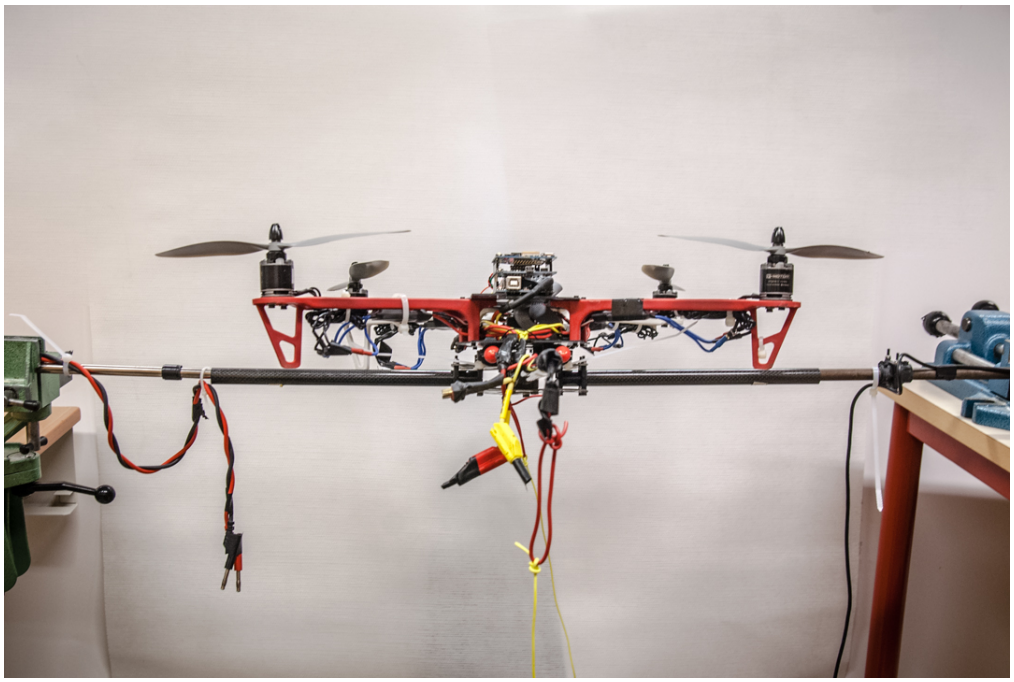
4 Genomförande

4.1 Testriggar

Inför stabiliseringen och dimensioneringen av de olika regulatorerna så uppstod det ett behov av pålitliga och robusta testriggar. Dessa riggar har som syfte att begränsa quadrotorns rörlighet till endast en axel i taget.

4.1.1 Pitch och roll

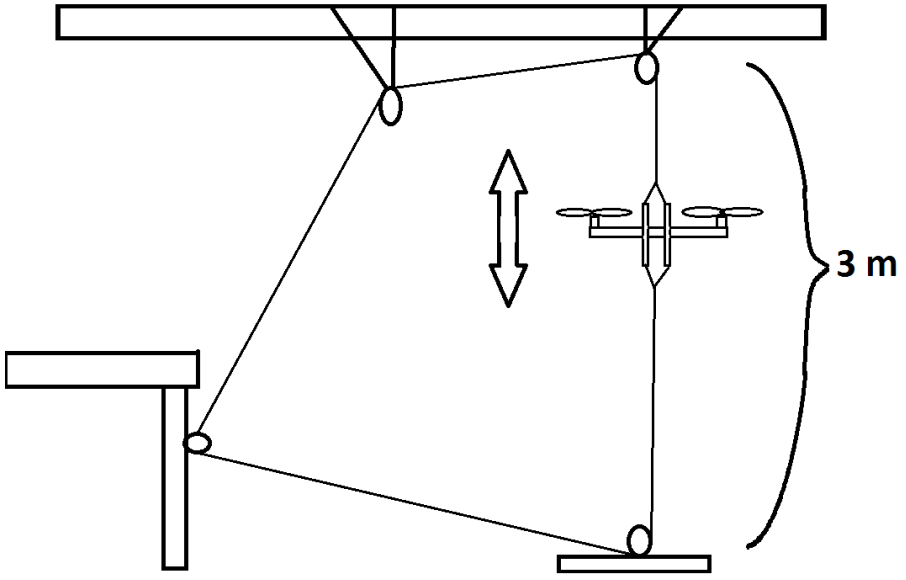
Pitch- och roll-regulatorn innebär dimensionering av två väldigt lika regulatorer. Samma sorts rörelse ska stabiliseras. Detta medförde att samma rigg kunde användas för både pitch- och roll-regulatorn. Denna testrigg benämns Testrigg A, se bild 4.1. En kolfiberram monteras på undersidan av quadrotorn. Denna har fyra stycken kolfiberrör i en krysskonfiguration. Stålstavar monterades på två bord och fördes in i kolfiberrören. Detta begränsade quadrotorns rörlighet till en axel. Utöver detta begränsades även rotationen kring roll-axeln till $\pm 70°$. Genom att rotera quadrotorn 90° runt yaw-axeln i denna rigg kan man ändra mellan roll- och pitch-stabilisering. Dessa riggar är lika funktionsmässigt, det som skiljer sig är hur quadrotorn monteras. Testrigg A genomgick flera versioner innan den slutgiltiga versionen användes.



Figur 4.1: *Testrigg A i pitch-konfiguration*

4.1.2 Höjd

Det utvecklades två stycken testriggar för stabilisering i höjdled. De ser funktionsmässigt likadana ut och benämns Testrigg B Den första bestod utav fyra stycken linor som gick från taket till marken. På quadrotorn monterades fyra stycken PVC-rör, cirka 30 cm långa. Linorna löpte igenom dessa rör och längden på rören förhindrade att quadrotorn från att luta och på så sätt skära av dessa linorna med rotorbladen. På denna rigg genomfördes majoriteten av experimenten. Det upptäcktes dock senare att friktionen mellan linorna och rören påverkade mätresultaten då översvängar blev kraftigt förstärkta vid tester utan testriggar. Detta resulterade i ytterligare en rigg. Den nya riggen bestod utav fyra stycken seglingsblock som fästs på olika ytor. Dessa valdes då kullagret i dessa erbjöd väldigt låg friktion. I dessa block löpte ett polyesterrep som fästs på ovansidan och undersidan av quadrotorn. Denna rigg erbjöd mindre friktion än den tidigare riggen och tillät även mindre fördlyttningar i sidled. Detta gjorde att denna rigg även kunde användas för tester av hela systemet, det vill säga, hur de kombinerade regulatorerna för pitch, roll och höjd fungerade tillsammans. Denna rigg finns illustrerad i Figur 4.2



Figur 4.2: *Testrigg B*

4.2 Stegsvarsanalys

Inför dimensioneringen av varje regulator gjordes utförliga stegsvarsanalyser. Dessa stegsvarsanalyser kommer att presenteras i ordningen som de genomfördes.

4.2.1 Roll

Testrigg A användes för dessa stegsvarsförsök. De första försöken strävade efter att finna ett samband mellan vinkelförändringshastigheten och styrsignalen. Ett Matlab-program konstruerades som läste in data från gyroskopet och illustrerar, i realtid, den aktuella vinkeln på samtliga axlar. Genom att spara vinkeldatan kunde detta sedan användas för att studera stegsvaret mer noggrant. Stegsvaret i sig utfördes genom att skicka en bestämd styrsignal till ett motorpar i taget. Genom att filma försöken i 240 bilder per sekund kunde videon analyseras för varje försök. Syftet med videoriggen var att kunna observera tiden och uppstarten av motorerna mer noggrant. De två motorparen utsattes för 10 försök i varje fall. I varje fall var styrsignalerna 75, 80, 85, 90 och 95 % av maximalt värde. Det upptäcktes redan efter några tidigare försök att det handlar om en ostabil process, men detta bekräftades under de noggrannare försöken. Den uppmätta mätdatan kan beskrivas med ekvationen (4.2). Processen har generellt sett en exponentiell tillväxt för varje givet stegsvar $A\sigma(t)$, där A är en konstant. Eftersom testriggen var begränsad till $\pm 70^\circ$ så var det relevant att endast studera utseendet hos stegsvaren i det intervallet. Det antogs att stegsvaret skulle ha liknande karakteristik även vid längre tider, så utifrån detta kan man tolka det att den exponentiella utvecklingen avtar och man kan således ställa upp en modell i S-planet enligt Figur 4.1. [1]

$$G_{ROLL}(s) = \frac{Ke^{-Ls}}{s(1+Ts)} \tag{4.1}$$

Efter att parametrarna hade bestämts erhölls ett stegsvar som var likt den ursprungliga mätdatan, dock var avvikelsen relativt stor. Det tolkades som att denna approximerade modellen inte var tillräckligt noggrann i detta fall, och en djupare metod påbörjades. Genom att använda Matlabs "Curve-fitting Toolbox" kunde mätdatan ställas upp som en funktion av tiden enligt (4.2).

$$f_{ROLL}(t) = 0.4194e^{4.422t} + 0.004476e^{7.686t} \tag{4.2}$$

Genom att sedan laplacetransformera uttrycket (4.2) kunde denna funktion representeras i S-planet enligt (4.3).

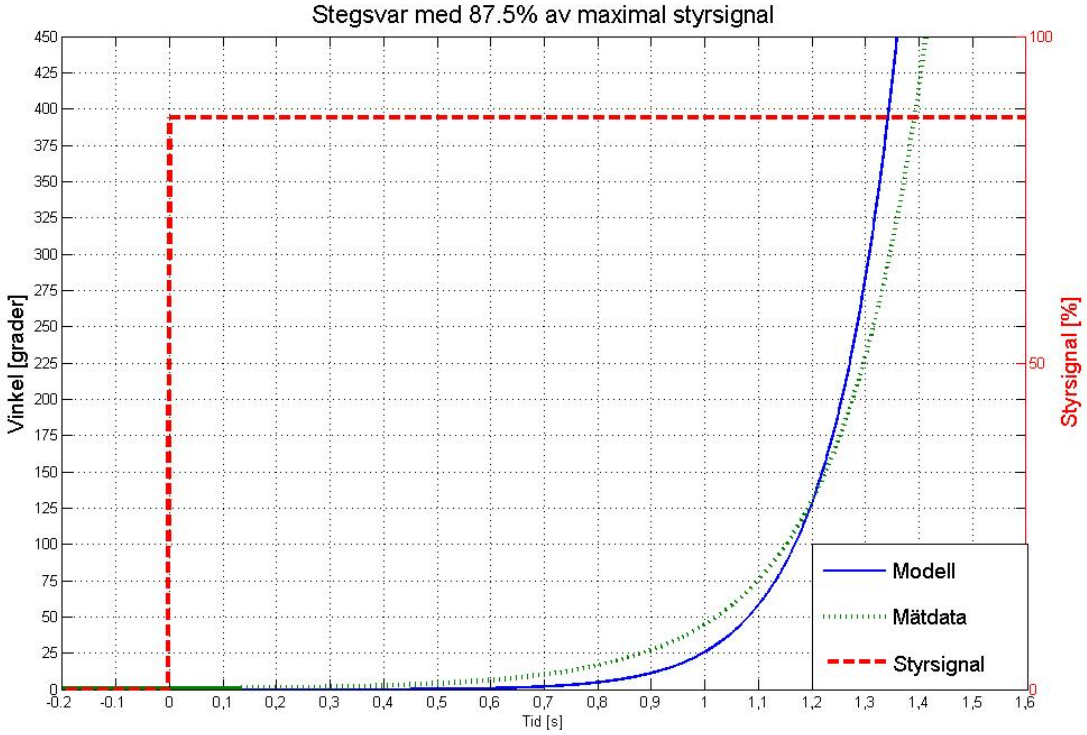
$$\mathfrak{L}\{f(t)\} \triangleq \int_0^\infty f(t)e^{-st}\mathrm{d}t \tag{4.3}$$

$$\mathfrak{L}\{f_{ROLL}(t)\} = \frac{211.9s + 1622}{s^2 - 12.11s + 33.99} \tag{4.4}$$

Där (4.3) är definitionen av Laplacetransformen, och (4.4) är Laplacetransformen av (4.2). När man till slut bryter ut stegfunktionen ur (4.4) erhålls en överföringsfunktion för processen, som satisfierade våra behov i avseende på pålitlighet, enligt (4.5).

$$G_{ROLL2}(s) = \frac{0.2119s^2 + 1.622s}{2.3s^2 - 27.855s + 78.17} e^{-0.45s} \tag{4.5}$$

Uttrycket (4.5) illustreras i Figur 4.3. Denna modell baserar sig på medelvärdet av alla medelvärden i denna stegsvarsanalys. Överföringsfunktionen som erhölls baserade sig då på medelvärdet, det vill säga, en styrsignal på 87.5 %.



Figur 4.3: *Stegsvarsjämförelse mellan stegsvaret, mätdatan och modellen.*

Det är även värt att notera att det upptäcktes att för stegsvarsfallet där styrsignalen 95 % användes kunde man observera att vinkelförändringshastigheten kraftigt avtog och hamnade någonstans mellan den som erhölls vid 85 och 90

%. Någon djupare analys av detta gjordes aldrig, men det konstaterades att begränsningar bör sättas på regulatorn som begränsar styrsignalen på grund av dessa effektförluster.

4.2.2 Pitch

Då regulatorn för roll-axeln dimensionerades kunde pitch-regulatorn dimensioneras med nästintill identiska parametrar 20 minuter efter roll-regulatorn dimensionerades. Någon stegsvarsanalys gjordes aldrig för denna axel.

4.2.3 Höjd

Stabilisering av höjden innebar två olika komponenter, lyftning och stabilisering. Stegsvarsanalyser genomfördes i både öppna och återkopplade system. Då pitch och roll var ostabila processer med exponentiella stegsvar, var det förväntat att även processen i höjdled skulle besitta samma egenskaper. Överföringsfunktioner ställdes upp med liknande karakteristik som pitch och roll enligt (4.1). Dock resulterade de approximerade överföringsfunktionerna och regulatorer som dimensionerades från dessa i bra simuleringar, men i praktiken fungerade dessa inte. En mer exakt överföringsfunktion eftersträvades. PID-parametrar för höjdledets regulator prövades slumpmässigt och till slut erhölls några som gav relativt bra resultat. Dock ser man, i Figur 4.5, att den första översvängen är väldigt kraftig, närmare bestämt 100.6 cm, och ett dämpat system skulle behövas för att återgärda detta. Denna kurva går att tolka som ett andra ordningens system, med en generell beskrivning enligt en andra ordningens överföringsfunktion i (4.6). Konstanterna ζ och ω_0 beskrivs i (4.7).

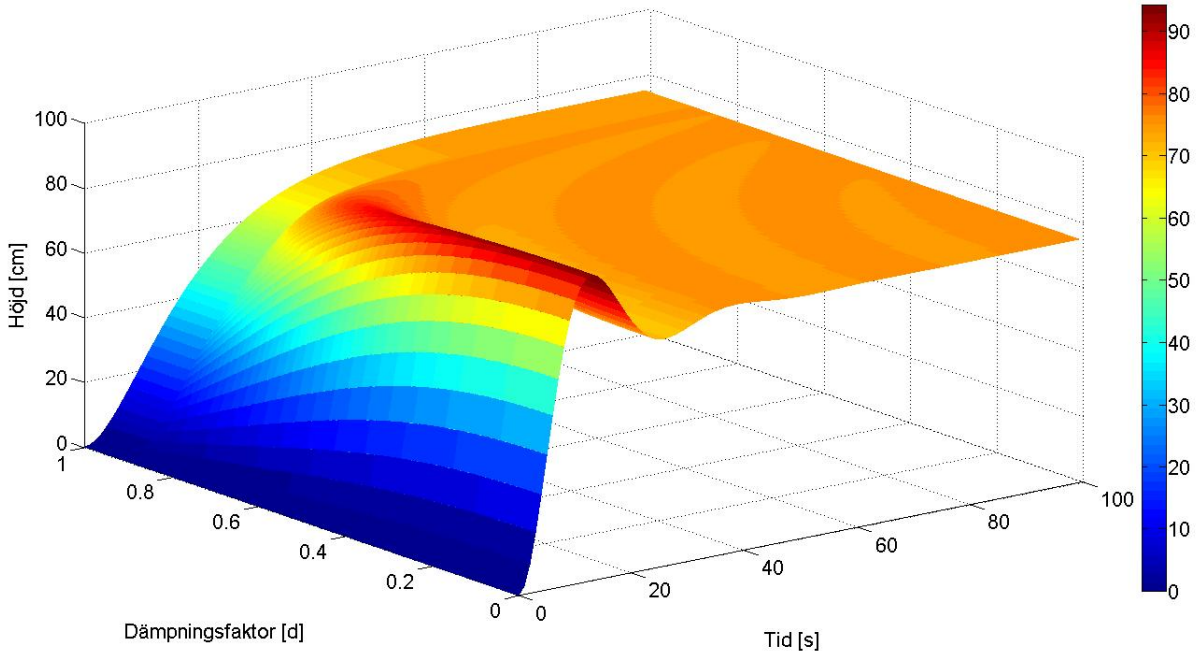
$$G_{tot1}(s) = \frac{\frac{K_s}{K_{SP}}\omega_0}{s^2 + 2\zeta\omega_0 s + \omega_0^2} \tag{4.6}$$

$$\begin{cases} \zeta = \frac{1}{\sqrt{(\frac{2\pi}{\log d})^2+1}} \\ \omega_0 = \frac{2\pi}{T_o\sqrt{1-\zeta^2}} \end{cases} \tag{4.7}$$

Där ω_o är vinkelfrekvensen av översvängarna, T_o är översvängarnas periodtid, K_s är slutnivån, K_{SP} är börvärdet och d är dämpningsfaktorn, förhållandet mellan den första och andra översvängen.Det misstänktes att ett mätfel hade uppstått på dämpningsfaktorn, därför beslutades det att skriva ett program i Matlab som utvärderar 1000 olika d-värden mellan noll till ett. Syftet med detta var att undersöka hur modellen beter sig när dämpningsfaktorn ökar, alltså när d-parametern går mot 0. Figur 4.4 visar resultatet av denna undersökning som resulterade i ett

gränsvärde för modellen enligt (4.8). Detta gränsvärde säger att med ett börvärde på 70 cm kan vi maximalt få en översväng på 30 cm.

$$\lim_{d\to 1} \frac{\frac{K_s}{K_{SP}}\omega_0}{s^2+2\zeta\omega_0 s+\omega_0^2} \leq 100 \tag{4.8}$$



Figur 4.4: *Undersökning av begränsningarna hos modellen för den andra ordningens system*

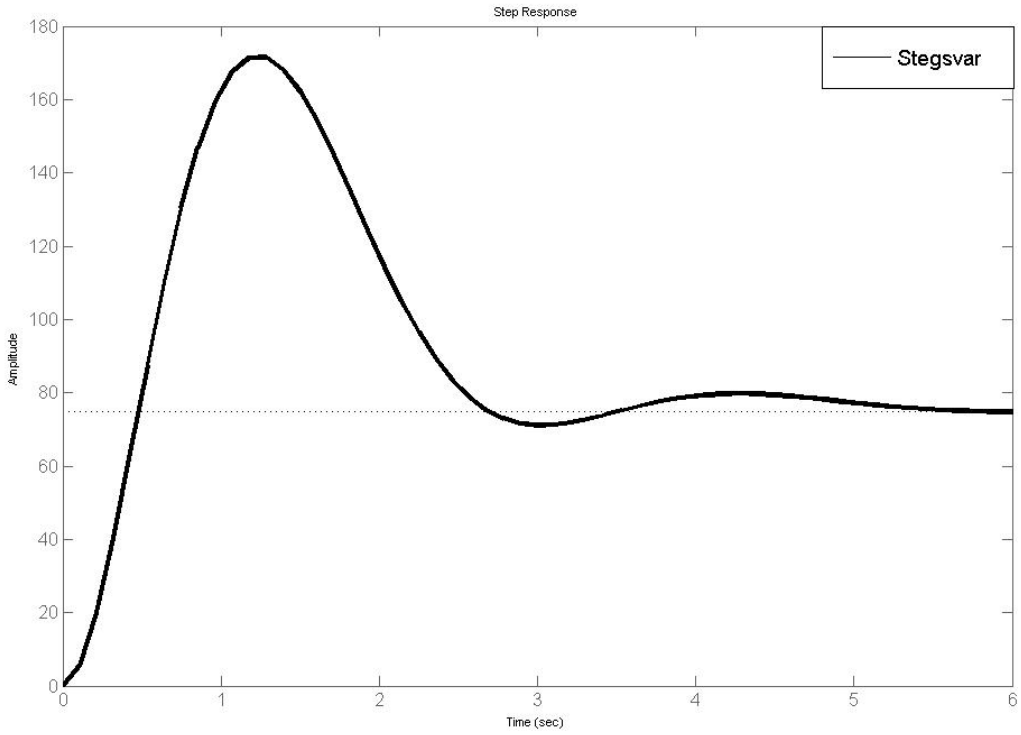
Genom att lägga till ett nollställe i modellen kan vi få en tredje ordningens överföringsfunktion som kan beskrivas med (4.9). Villkoret för att lägga till ett nollställe på detta sätt är att kvoten $\frac{a}{b} > 1$.

$$G_{tot2}(s) = \frac{\frac{K_s}{K_{SP}}\omega_0}{s^2+2\zeta\omega_0 s+\omega_0^2}\frac{as+1}{bs+1} \tag{4.9}$$

Efter numeriska försök kunde det konstateras att om a = 2.8 och b = 1 erhölls tillfredställande resultat. Den totala överföringsfunktionen kunde sedan bestämmas till (4.10).

$$G_{totThr}(s) = \frac{15.27s+5.657}{s^3+2.98s^2+7.259s+5.28} \tag{4.10}$$

Modellen (4.10) visade sig vara den stabilaste, mest pålitliga. Stegsvarssimuleringar genomfördes i Matlab med bra resultat. Figur 4.5 illustrerar simuleringen av det stegsvar som mätdatan kom ifrån. Värt att notera är att den första översvängen nu stämmer bättre överens med verkligheten, dock uppstod ett fel på cirka 20 cm på undersvängen. Detta var dock en uppoffring som kunde tolereras, då dämpningsfaktorn är såpass hög som den är.



Figur 4.5: *Stegsvar av (4.10) från 0 till 70 cm*

Genom att sedan använda blockschematransformation kunde en överföringsfunktion bestämmas för processen. Överföringsfunktionen för PID-regulatorn som användes var bekant, enligt (4.11) som beskriver överföringsfunktionen för en ideell PID-regulator med parametrarna P = 3, I = 2 och D = 1.

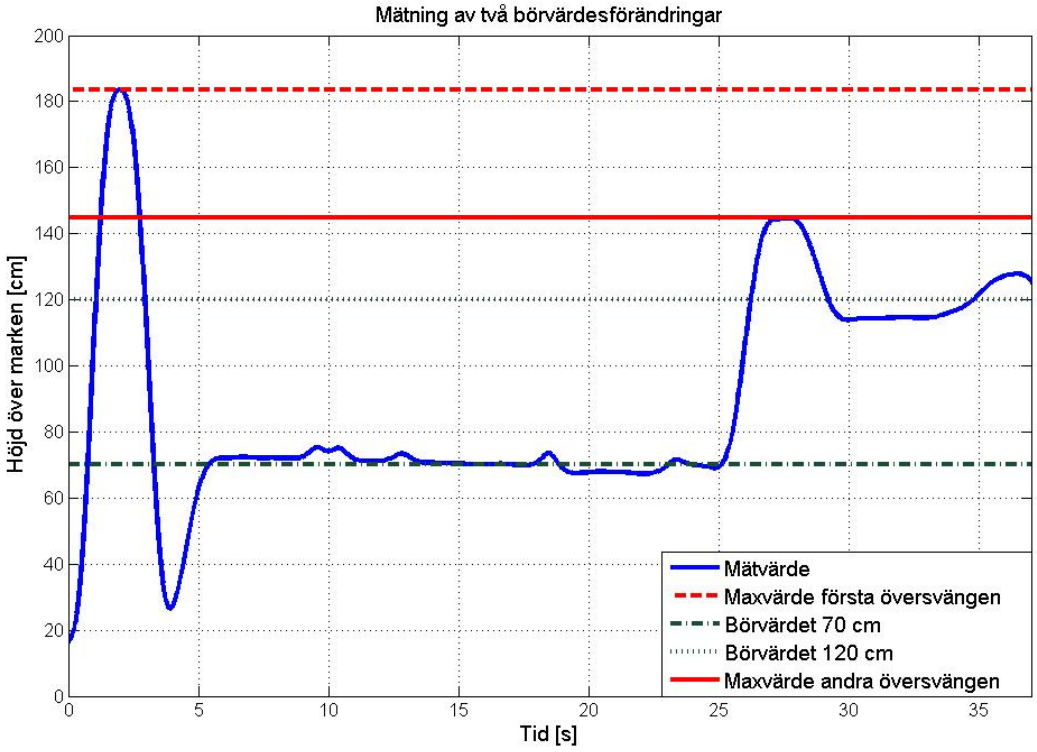
$$G_R(s) = 3(s^2 + s + \frac{1}{2}) \tag{4.11}$$

Genom att sedan använda blockschematransformation kunde processens överföringsfunktion G_P erhållas genom sambandet (4.12).

$$G_P(s) = \frac{G_{tot}}{G_R - G_{tot}G_R} \tag{4.12}$$

Processen fås om man använder (4.10) och (4.11) i (4.12). Genom denna metod får man en överföringsfunktion som beskriver processen enligt (4.13).

$$G_P(s) = \frac{15.13s^4 + 37.86s^3 + 98.74s^2 + 97.69s + 21.68}{6s^7 + 27.83s^6 + 196.5s^5 + 489.5s^4 + 1030s^3 + 1143s^2 + 488.4s + 65.04} \tag{4.13}$$



Figur 4.6: *Mäting av två börvärdesförändringar.*

Den uppmärksamma läsaren observerar att detta resultat avviker mycket från (4.1). Det var tidigare sagt att throttles öppna stegsvar kunde jämföras med pitch och rolls format. Däremot om man begränsar (4.13) till ett begränsat intervall i tiden $0 \leq t < 1.8$ s så är fortfarande båda dessa funktioner snarlika. Det är återigen värt att påpeka att dessa baserar sig på observationer och är trots allt approximationer av observerade fenomen. Det kunde dock observeras från samtliga stegsvarsförsök att amplituden hos den första översvängen var beroende av motorernas

varvtal. Högre varvtal på motorerna ger alltså lägre översväng. Detta kan tydligt illustreras med figur 4.6 som visar ett stegsvarsexperiment som genomfördes. Motorerna började med ett lågt varvtal som inte lyfte quadrotorn, därefter sattes börvärdet till 70 cm. Efter 25 sekunder ändrades börvärdet till 120 cm. Man ser att den första översvängen är mycket kraftigare än den andra översvängen.

4.3 Stabilisering

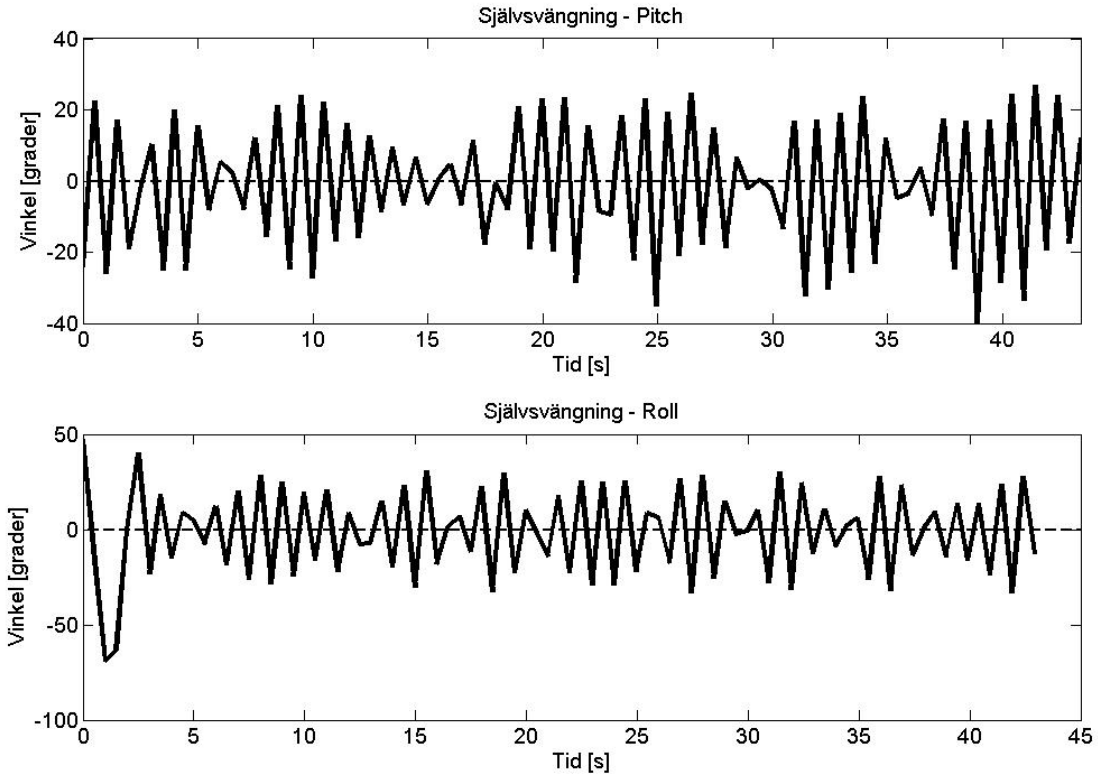
4.3.1 Ziegler-Nichols svängningsmetod

För att hitta P-, I- och D-parametrarna till quadrotorns regulatorer användes Ziegler-Nichols svängningssmetod som baserar sig på praktiska tester.

Pitch och roll Pitch- och roll-axlarna på en quadrotor är väldigt lika varandra och därför kan samma metod användas för att ta fram regulatorparametrarna. När väl parametrarna för den ena axeln arbetats fram kan parametrarna för den andra axeln snabbt tas fram med samma tillvägagångssätt, framför allt på grund av att testriggen för pitch även kunde användas för roll utan några större modifieringar. Det svåra med dessa praktiska tester är att hitta självsvängningsfrekvensen som metoden kräver för att kunna få ut periodtiden, T_0, för självsvängningarna. Då Testrigg A inte tillåter vinklar över 70° åt något håll kommer quadrotorn att nå max- och min-vinklar under tiden som mätningarna av självsvängningsfrekvensen sker. Detta löstes genom att använda en stabilisator, FY-901, för att sköta grovregleringen och sedan låta den egenutvecklade PID-regulatorn styra insignalen till stabilisatorn. På detta sätt kan vi begränsa de stora vinklarna samtidigt som vi mäter periodtiden för självsvängningen. Mätdata som insamlades med hjälp av gyroskopets X- och Y-axlar skickades till Arduino IDEs utskriftsfönster. Mätdata som skickades var tidpunkt för mätning i millisekunder och vinkeln från gyroskopet.

Dessa data behandlades sedan i Matlab där verktyget “Curve-Fitting Tool” andvändes för att beskriva mätdata som en summa av sinusvågor. Man kan se i Figur 4.7 att det handlar om två olika frekvenser i självsvängningen. Det ska dock påpekas att samplingsfrekvensen i båda dessa fall låg på cirka 2 Hz. De två frekvenser som uppstår beror på självsvängningen och att mätdatan inte filtrerades. Gyroskopet kunde ibland ge orimliga utslag, vilket motiverar behovet av filtrerad mätdata och resulterade i att systemet fick en extra svängningsfrekvens. Den högsta frekvensen användes vid dimensionering av pitch- och roll-regulatorerna. Den kritiska förstärkningen som gav upphov till självsvängningen för pitch var $K_{pitch0} = 70$ och för roll användes $K_{roll0} = 80$. Genom att sedan använda den standardiserade tabellen 3.1 erhölls PID-parametrarna som återges i Tabell 4.1. Dessa parametrar resulterade i ett stabilt system, därför utfördes aldrig försöket med

högre samplingsfrekvens. Detta kan göras i framtiden när högre krav på robusthet ställs på systemet.



Figur 4.7: *Självsvängningen för pitch och roll*

Efter att en regulator dimensionerades kunde en ny överföringsfunktion bestämmas baserad på observationer från datan. Det kunde konstateras att det nya systemet handlade om en överföringsfunktion med två tidskonstanter enligt (4.14). Dock så förklarar denna modell inte översvängen som uppmättes, alltså behöver vi lägga till en pol och ett nollsälle i vår modell. Detta realiseras genom att multiplicera (4.14) med (4.15)

$$G_1 = \frac{K}{(T_{1/3}s+1)(T_{2/3}s+1)} \tag{4.14}$$

$$G_2 = \frac{as+1}{bs+1} \tag{4.15}$$

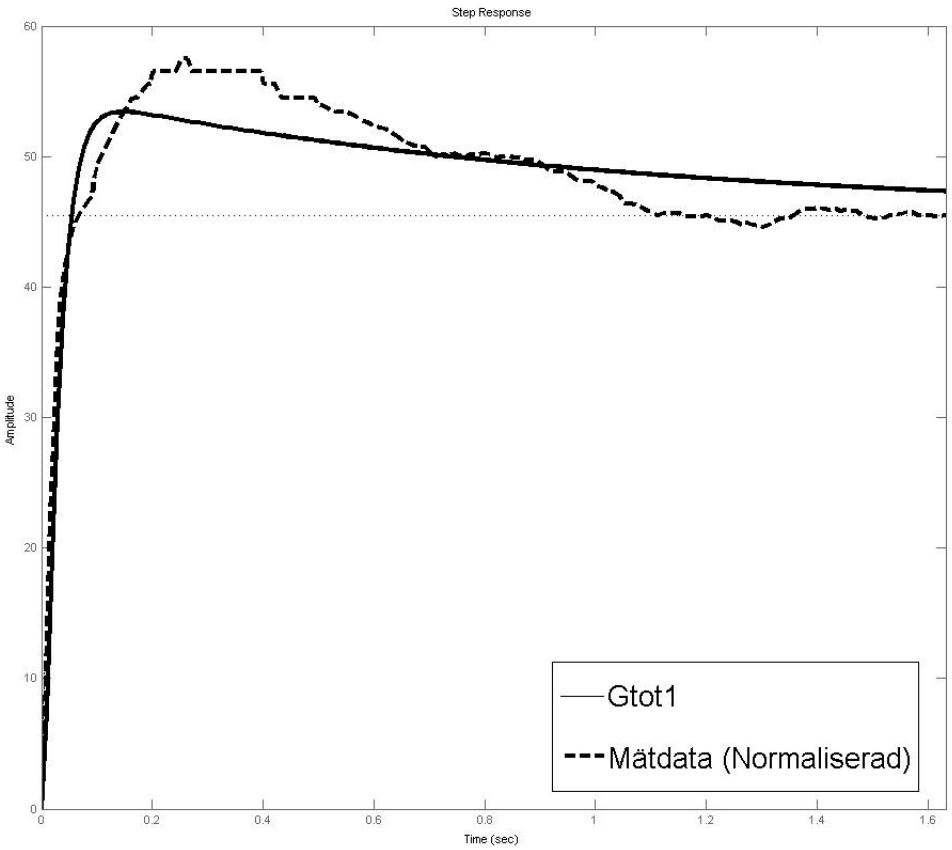
där a > b

Detta ger oss en modell enligt (4.16). Figur 4.8 visar en jämförelse mellan en simulering av (4.16) och mätdata. Mätdata och simuleringar normaliserades med avseende på att eliminera negativa värden i Y-axeln.

$$G_{TOTroll} = \frac{1.2s + 1}{0.000216s^3 + 0.03322s^2 + 1.033s + 1} \tag{4.16}$$

Slutligen, en omskrivning av (4.16) ger oss (4.17) på samma format som övriga modeller i denna rapport.

$$G_{totAil} = \frac{5555.6s + 4682.4}{s^3 + 153.8s^2 + 4782.4s + 4629.6} \tag{4.17}$$



Figur 4.8: *Jämförelse mellan mätdata och simulering utav roll-axelns regulator*

Höjdreglering Det antogs att Ziegler-Nichols svängningsmetod även skulle fungera för höjdregleringen. Det var dock andra faktorer som påverkade möjligheterna till en självsvängning. Att använda sig av proportionerlig reglering i detta sammanhang fungerade inte då gravitationen och begränsningar i testriggarna inte tillät detta. Det ansågs lämpligt att utgå från ett tidigare projekt på samma platform. Dessa parametrar gav bättre resultat, dock inte satisfierande. Olika parametrar prövades med utgångspunkt från dessa värden tills dess att en satisfierande uppsättning hittades. Denna uppsättning visas i Tabell 4.1.

Från stegsvarsanalyserna och även från de praktiska försöken kunde det konstateras att översvängen vid börvärdesförändringar var starkt beroende utav motorernas initiala varvtal. Den största översvängen fås då quadrotorn först lyfter från marken. Detta gjorde att det fanns ett behov för en startfunktion. Denna funktion skulle se till att motorerna hade ett tillräckligt högt varvtal som dämpar den första översvängen samt inte lyfter den från marken. Efter två sekunder, när motorerna är uppe i varv, kommer styrsignalen till throttle att ökas med 70 efter varje programcykel. Detta testades genom praktiska försök i inomhusmiljö. Det ger en snabb men kontrollerad lyftning från marken. Efter att quadrotorn passerat 40 cm från marken kommer startfunktionen att stängas av och PID-regulatorn att ta över.

Axel	K_P	T_I	T_D
Pitch	42	0.411	0.103
Roll	48	0.467	0.117
Throttle	3	2	1

Tabell 4.1: *Tabell som visar de tre regulatorernas PID-parametrar.*

4.3.2 Jämförelse

Då Ziegler-Nichols svängningsmetod gav snabbare resultat än stegsvarsmetoden valdes denna metod för pitch- och roll-regulatorn. Throttle-regulatorn visade sig vara svårare att dimensionera med denna metod, men med start-funktionen erhölls satisfierande resultat.

4.4 Programvaruutveckling

Utvecklingsmiljön ”Arduino IDE” användes för att programmera Arduino-korten och dess programmeringsspråk baseras på C/C++. Mjukvaran på en autonom quadrotor behöver vara väldigt tidseffektiv då den hanterar stabiliteten hos farkosten och en onödig fördröjning i programkoden kan medföra en ojämn och ryckig

reglering av motorerna. Därför har mycket tid lagts på att effektivisera och minimera koden så mycket som möjligt. Varje Arduino-kort ombord på farkosten har varsitt program med tydligt skilda uppgifter. Arduino Uno-kortet hanterar all extern kommunikation och där har fokus lagts på att minimera risken att information försvinner på vägen genom att på ett smidigt sätt definiera tillåtna kommandon och hantera dessa på olika sätt beroende på vad kommandona skall göra. Arduino Mega-kortet har hand om inläsning av alla sensorer, förutom GPS-enheten, och den tar även hand om all reglering baserat på de sensorvärden som den läst. Här har mycket tid lagts på att effektivisera alla funktioner. Detta har gjorts med noggrann mätning av varje funktion i programmet och minimera dess tidsåtgång.

4.4.1 Arduino Uno

Arduino Uno-kortets huvuduppgift var att hantera extern kommunikation och förmedla denna information vidare till Arduino Mega-kortet. Extern kommunikation i detta fall innebar Wi-Fi-kommunikation med ansluten dator och GPS-kommunikation med satelliter. Signaler från quadrotorns fjärrkontroll för manuell styrning hanterades dock av Arduino Mega-kortet då denna kommunikation är tidskritisk och bör ligga så nära motorstyrningen som möjligt. Arduino Uno-kortet upprättar en anslutning till Wi-Fi-nätet och startar sedan en Telnet-server som används för att med hjälp av t.ex en dator ansluta till Telnet-servern och sedan skicka kommandon från datorn som Arduino Uno-kortet läser in, byte för byte, och packar in detta i en sträng. Strängen skickas sedan till Arduino Mega-kortet som tolkar strängen som ett kommando. Arduino Uno-kortet gör alltså ingen egen kontroll av de tecken som matas in från Telnet-klienten då det skulle behövas en uppdelning av strängen i formatet ”<*kommando*> <*integer*>” för att sedan packa tillbaka det i en enda sträng och skicka. Å ena sidan sparar denna metod tid för processorn, men å andra sidan kan den seriella bussen mellan de båda Arduino-korten bli högre belastad. Arduino Uno-kortet skickar mottagen data över den seriella bussen med hjälp av protokollet I^2C som hanteras med hjälp av Arduinos standardbibliotek ”Wire”. Då Arduino Uno-kortet är Slave på den seriella bussen, och Arduino Mega-kortet är Master, kommer Arduino Uno att få förfrågningar från Arduino Mega med jämna mellanrum. När Arduino Uno får en förfrågan anropas funktionen *requestEvent()* som kontrollerar om ett nytt kommando har tagits emot från Telnet-klienten. Om flaggan *cmd_sent* är *false* kommer strängen *cmd*, som innehåller kommandot, att skickas till Arduino Mega. Om flaggan är satt till *true* kommer Arduino Uno istället att skicka GPS-information.

4.4.2 Arduino Mega

Huvuduppgiften för Arduino Mega-kortet var beräkningar. På detta kort programmerades alla de regulatorer som behövs för stabilisering. Processorns hastighet är 16 MHz och hela programcykeln, i autonomt läge, tar 75 ms. Detta begränsar kraftigt samplingsmöjligheterna, samplingsfrekvensen f_s har på grund av denna begränsning blivit 11 Hz. För att få så hög samplingsfrekvens som möjligt krävs en tidsoptimerad kod. För att tidsoptimera koden användes en Tabell 4.2 där tider för alla funktioner i programmet ställdes upp. Tiderna uppmättes genom att använda processorns klocka. En tidpunkt, i millisekunder, togs när funktionen anropades och en annan tidpunkt togs när funktionen var klar. Den andra tidpunkten subtraherades med den första och då erhålls tiden som funktionen använde. När en funktion ändras, tas bort eller läggs till uppdateras tabellen med de nya värdena och det är då enkelt att se när en funktion tar för mycket tid.

Funktion	**Tidsåtgång(ms)**
Läsning av fjärrkontroll	9
Kolla efter nya kommandon	1
Läsning av höjdsensor	74
Läsning av gyroskop	7
Roll-regulator	0
Pitch-regulator	0
Rudder-regulator	0
Throttle-regulator	0
Take-off-rutin	0
Uppdatera PWM-utgångar	1
Total tidsåtgång	92

Tabell 4.2: *Tidsåtgång för funktioner i Arduino Mega*

Tidsåtgången i Tabell 4.2 visar hur lång tid varje funktion tar under en normal flygning. Det som tar upp den största delen av tiden är höjdsensorn. Det beror bland annat på att ljudets hastighet i luft är begränsad, att vi gör tre mätningar seriellt varje gång för att säkerställa bra mätvärde, samt att det finns en fördröjning mellan varje mätning. Att en funktion tar 0 millisekunder betyder att tidsåtgången för den funktionen är försumbar.

Programflödet följer flödesschemat i Appendix A. Funktionerna är uppdelade för att varje funktion ska ha en specifik uppgift.

Initiering Vid initieringen deklareras alla globala variabler, regulatorer och Ping)))-sensor. Här görs även en del inställningar som ska gälla vid start:

- Frekvensen på utgångar till stabilisatorn sätts till 50 Hz
- Ut- och inportar definieras
- Startvärden skrivs till PWM-utgångarna för att stabilisatorn ska initieras.
- PID-regulatorernas utsignal begränsas för att orimliga förändringar att PWM-värden till stabilisatorn ska undvikas
- Gyroskopet initieras och börjar mäta de vinklar som används
- Kalibrerade börvärden sätts för att quadrotorn ska få en stabil start

Eftersom gyroskopet inte sitter helt parallellt med quadrotorn behövdes nya kalibrerade börvärden. De kalibrerade börvärdena togs fram genom praktiska försök.

Läsning av fjärrkontrollen Läsning av fjärrkontrollens olika värden görs genom att från fjärrkontrollens mottagare, som sitter monterad på quadrotorn, läsa av längden på de digitala signaler som mottagaren genererar. Längden på dessa signaler motsvarar vilket läge fjärrkontrollens olika spakar och knappar är i. Först kontrolleras en knapp på fjärrkontrollen som är till för att ändra mellan automatisk och manuell kontroll. Är knappen inställd på manuell kontroll läses värden från fjärrkontrollens spakar in och quadrotorn sätts i manuellt läge. Om inte, så sätts quadrotorn i automatiskt läge och fjärrkontrollens spakar blir oanvändbara.

Läsning av nytt kommando från Uno Arduino Uno-kortet kan ta emot textsträngar över en Wi-Fi-anslutning till användaren. Eftersom Mega-kortet är ”master” på I^2C-bussen kommer den att fråga Uno-kortet om det finns en ny textsträng från användaren, om det finns så skickas den till Mega-kortet. Om Mega-kortet fått en ny textsträng kommer den att delas upp till formatet *”<kommando> <heltal>”*. Beroende på vad som står i strängen *<kommando>* kommer olika inställningar att göras på quadrotorn. Exempel på kommandon kan vara ”start”, ”reset”, ”set_height” som ändrar börvärden för höjdregleringen till det värde som erhölls i *<heltal>*.

Läsning av höjdsensorn Denna funktion körs endast på följande villkor:

- Quadrotorn är i automatiskt läge
- Användaren har skickat ett startkommando

Om alla villkoren är uppfyllda kommer avståndet till marken att mätas tre gånger och medianvärdet blir de värde som regulatorn för höjdreglering kommer att använda som ärvärde.

Läsning av Gyroskop Denna funktion körs endast på följande villkor:

- Quadrotorn är i automatiskt läge
- Användaren har skickat ett startkommando
- Höjden från höjdsensorn överstiger 20 cm

Om alla villkoren är uppfyllda kommer Mega-kortet att läsa de vinklar som gyroskopet har uppmätt. Då höjdsensorn registrerar ett värde under 20 cm betyder det att quadrotorn står på marken. Vi vill då inte göra någon avläsning från gyroskopet eftersom all reglering av roll och pitch innan quadrotorn har lyft kan få den att tippa omkull när den väl lyfter. Detta beror på att när regulatorn försöker kompensera för den lutning som finns på marken, som quadrotorn står på, händer inget och regulatorn blir då mer aggressiv och ökar varvtalen på motsvarande motorer tills det att den tippar omkull.

Reglering av pitch, roll och yaw Denna funktion körs endast på följande villkor:

- Quadrotorn är i automatiskt läge
- Användaren har skickat ett startkommando
- Höjden från höjdsensorn överstiger 20 cm

Om alla villkoren är uppfyllda beräknas de utsignaler för regulatorerna som sköter stabiliteten för roll, pitch och yaw.

”Take off”-rutinen Denna funktion körs endast på följande villkor:

- Quadrotorn är i automatiskt läge
- Användaren har skickat ett startkommando
- ´´Take off”-flaggan är satt

Då den första översvängen som uppstår vid automatisk höjdreglering är starkt beroende av motorernas varvtal vid en börvärdesförändring valdes det att programmera en rutin som håller noggrannare koll på motorerna under lyftsekvensen. Flödesschema för denna rutin finns bifogat i Appendix B. Denna sekvens kan grovt delas upp i tre faser:

1. Vid initieringen av ´´take off”-rutinen sätts motorerna till ett högt varvtal, som dock inte lyfter quadrotorn, i två sekunder. Detta skapar bra förutsättningar för att minska den första översvängen när quadrotorn lyfter från marken.

2. Efter två sekunder påbörjas nästa fas. En räknare initieras och inkrementerar med 70 efter varje programcykel för att sedan adderas till den styrsignalen som skickas till stabilisatorn. Detta värde ansågs tillräckligt bra i avseende på stigtid och den resulterande översvängen. Utöver detta så ändras även börvärdet för pitch till +25°. Orsaken till detta är för att begränsa att quadrotorn driver för mycket under startsekvensen. Detta värde provades fram och tillåter vertikala lyftningar från marken med minimal drivning.

3. När quadrotorn är högre än 40 cm i luften kommer den sista fasen att inledas. Denna fas nollställer throttle-regulatorn för i ytterligare ett förebyggande syfte mot den första översvängen. “Take off”-flaggan nollställs och börvärdet för pitch återställs även.

Höjdreglering Denna funktion körs endast på följande villkor:

- Quadrotorn är i automatiskt läge
- Användaren har skickat ett startkommando
- ´´Take off”-flaggan är nollställd

Om alla villkoren är uppfyllda beräknas utsignalen för den regulator som sköter höjdreglering baserat på den insignal som ges av PING))-sensorn.

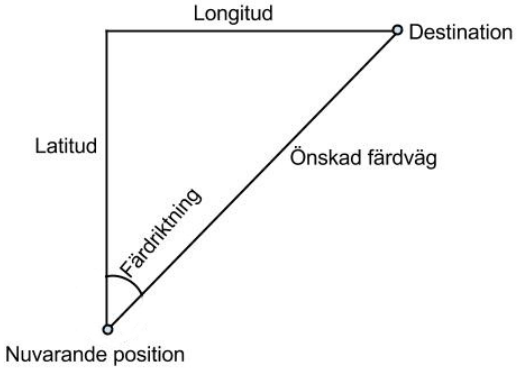
Generering av PWM signaler I Arduino finns färdiga bibliotek för att generera PWM-signaler på specifika digitala utportar. Detta bibliotek genererar PWM-signaler baserat på ett värde mellan 0-255. Då vår reglering är väldigt känslig och även små variationer i PWM-signalerna bidrar till stor påverkan på flygstabiliteten beslutades det att använda ett mer avancerat bibliotek utvecklat av Brett Beauregard.[15]

Det nya biblioteket utnyttjade Arduinos timer-funktion för att ändra hastigheten med vilken de digitala utportarna kan generera de pulser som behövs i en PWM-signal. Med det nya biblioteket erhölls ett intervall mellan 0-65536 där 0 motsvarar 0 % PWM-signal och 65536 motsvarar 100 % PWM-signal.

4.5 GPS

För att uppnå helt perfekt stabilitet behöver quadrotorn veta var den befinner sig i rummet. Det gör att den kan hitta en fast position som den alltid kan återgå till om det till exempel kommer en vindpust som får den att driva åt sidan.

Informationen från GPS-enheten kan användas på många olika sätt. Bland annat kan man få ut det väderstreck quadrotorn åker åt, höjden i meter över havet, hastighet och de koordinater, i longitud och latitud, man befinner sig på. Med hjälp av koordinaterna från GPS-enheten, koordinater för en destination och gyroskopets inbyggda kompass kan man med enkel trigonometri beräkna i vilken riktning och hur långt den ska färdas. Det beräknas genom att låta skillnaderna mellan nuvarande och destinationens longitud respektive latitud vara varsin katet och den önskade färdlängden vara längden av hypotenusan. Vinkeln mellan katet och hypotenusa är quadrotorn önskade färdriktning. Figur 4.9 visar ett exempel på hur en sådan situation kan se ut.

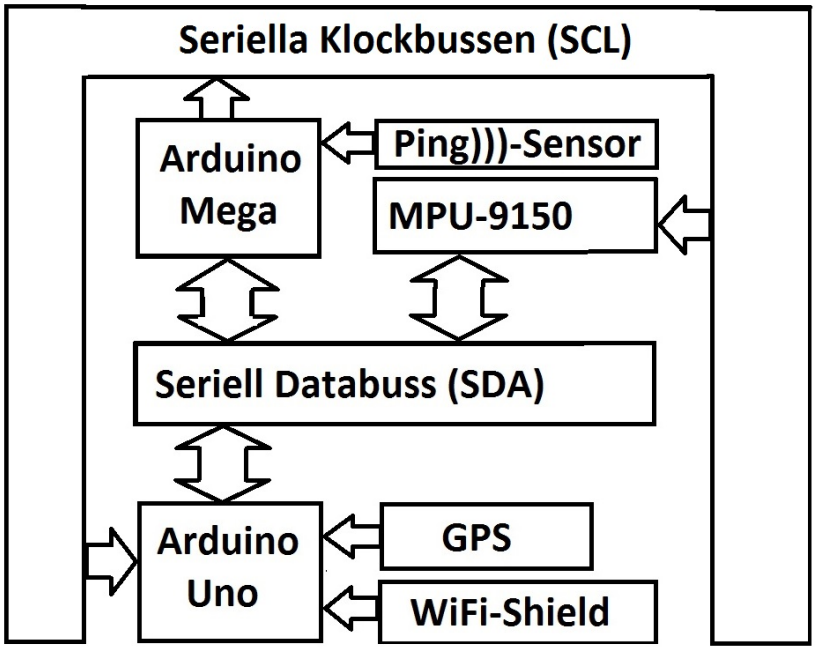


Figur 4.9: *Beräkning av önskad färdväg*

4.6 Kommunikation

4.6.1 Intern kommunikation

Den interna kommunikationen genomfördes genom implementering av en I^2C-buss. Arduino Mega utsågs till master på den seriella bussen medan övriga enheter blev Slaves. Arduino Mega anropar gyroskopet och Arduino Uno när den behöver data från någon av dessa enheter. PING)))-sensorn kommunicerar direkt med Arduino Mega utan en databuss, GPS och Wi-Fi Shield kommunicerar direkt med Arduino Uno genom Unos digitala ut- och inportar. Genom att koppla dessa två enheter på I^2C-bussen kan Arduino Mega direkt begära information från dessa enheter.



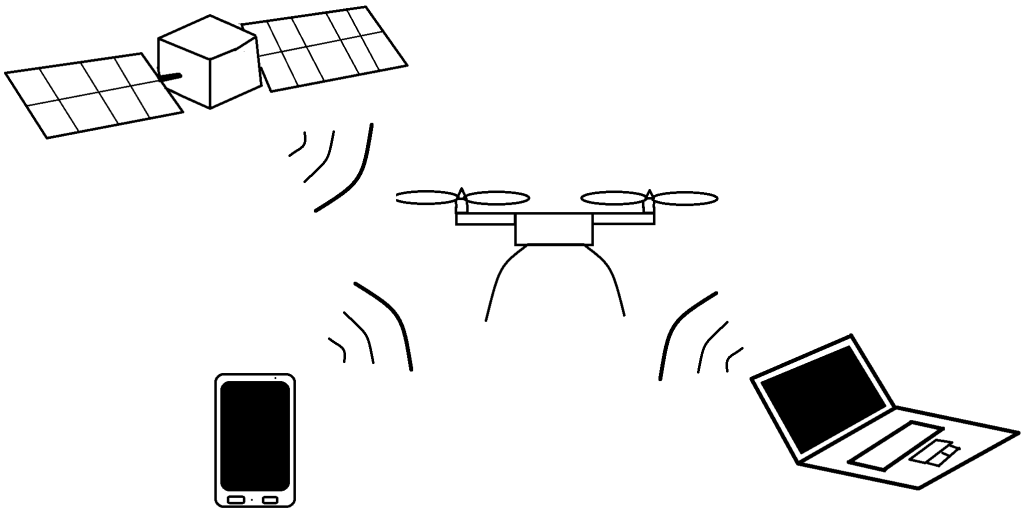
Figur 4.10: *Den interna kommunikationstopologin.*

4.6.2 Extern kommunikation

Genom att montera en Wi-Fi-shield på quadrotorn uppstod möjligheten att kommunicera med processorn från en extern dator. Detta visade sig vara väldigt användbart i flera tester när det önskades till exempel en nödstoppsfunktion, eller vid sensorkalibrering. Det programmerades även in funktioner som möjliggör styrning av quadrotorn genom Wi-Fi från en dator eller mobil. GPS-mottagaren på quadrotorn möjliggör även att quadrotorn kan få positionsrelaterad information från satelliter. En grundläggande Telnetserver har implementerats för kommunikationen från användaren till quadrotorn. Enkla instruktioner kan skickas till quadrotorn om en Telnetsession har initierats av användaren.

4.6.3 Radiomottagare

Radiomottagaren genererar PWM-signaler utifrån spakarnas och knapparnas lägen på fjärrkontrollen, en signal för varje spak och knapp. PWM-signalerna som genererades kunde med oscilloskop analyseras och översättas till det intervall, 0-65536, som Arduino hanterar. Av analyserna visade det sig att intervallet som radiomottagaren arbetar inom är 5 % - 10 % av maximum. Det kan av Arduino tolkas som att radiomottagarens arbetsområde ligger mellan 3277-6553. Arduino programmerades för att mäta signalerna från radiomottagaren och omvandla den



Figur 4.11: *Den externa kommunikationstopologin.*

till det korrekta intervallet.

4.6.4 Stabilisator

Det konstaterades, genom praktiska tester, att stabilisatorn och radiomottagaren är kompatibla med varandra. Det gjorde att intervallet som erhölls från radiomottagaren direkt kan skickas som PWM-signaler till stabilisatorn. Tabell 4.3 beskriver hur signalerna till stabilisatorn tolkas. Genom praktiska tester upptäcktes det att värdet 4915, som anges som startvärde för pitch och roll, ligger i mitten i intervallet och det betyder att stabilisatorn tolkar detta som att den ska räta upp quadrotorn och vara så plan som möjligt. En minskning eller ökning av det värdet gör att den stabiliserar quadrotorn åt respektive håll. För att stabilisatorn ska initiera och tillåta flygning kräver den att PWM-signalen som den tar emot ligger i den undre tiondelen av det accepterade intervallet. Detta för att quadrotorns motorer inte ska få någon spänning om man av misstag skickar fel PWM-signal.

Typ	**Min.**	**Startvärde**	**Max.**
Pitch	3277	4519	6553
Roll	3277	4519	6553
Yaw	3277	4519	6553
Gas	3277	3277	6553

Tabell 4.3: *Tabell för de PWM-signaler som stabilisatorn accepterar*

5 Resultat

5.1 Testriggar

Två stycken testriggar konstruerades som begränsade rörligheten till en axel i taget. När testriggarna konstruerades låg fokus på att minimera den friktion som testriggarna har. Detta löstes med kullagerutrustade block som minskade friktionen i den mest friktionsutsatta testriggen, nämligen den för höjdregleringen.

5.2 Stegsvarsanalys

De viktigaste resultaten som erhölls från stegsvarsanalysen var främst information relaterad till hur insvängningsförloppen går till. Vid pitch och roll erhölls inga resultat som användes vid regulatordimensioneringen, dock erhölls matematiska modeller som ser ut att förhålla sig bra till mätdatan. Stegsvaranalysen för throttle var dock mer givande. Förutom matematiska modeller så erhölls även kritisk information om förhållandet av den första översvängen och motorernas varvtal. Denna upptäckt resulterade i att en "take off"-rutin programmerades som styrde motorernas varvtal från marken till en höjd på 40 cm innan PID-regulatorn för throttle tog över. Bode-diagram för dessa två system finns bifogade i appendix D

$$G_{totThr}(s) = \frac{15.27s + 5.657}{s^3 + 2.98s^2 + 7.259s + 5.28} \tag{5.1}$$

$$G_{totRoll} = \frac{5555.6s + 4682.4}{s^3 + 153.8s^2 + 4782.4s + 4629.6} \tag{5.2}$$

5.3 Stabilisering

Genom Ziegler-Nichols svängningsmetod lyckades PID-regulatorerna för pitch och roll dimensioneras. Dessa regulatorer klarar även stabila flygningar utomhus med vindhastighet på fem meter per sekund.

Axel	K_P	T_I	T_D
Pitch	42	0.411	0.103
Roll	48	0.467	0.117
Throttle	3	2	1

Tabell 5.1: *Tabell som visar de tre regulatorernas PID-parametrar.*

5.4 Programutveckling

En stabil programbas krävdes för ett lyckat projekt och detta har realiserats genom många tester och omskrivningar av koden. Till slut uppnåddes kraven och quadrotorns många kringkomponenter fick den snabbhet och robusthet som de kräver.

5.4.1 Arduino Mega

Programmet är uppdelat i funktioner baserat på de uppgifter som tilldelats. Det gör att man enkelt kan byta ut och flytta runt funktionsanrop, och snabbt och smidigt kalla på en funktion på flera olika ställen om så skulle behövas. Detta är inget som sker i det nuvarande programmet då allting körs sekventiellt enligt flödesschemat i appendix A där funktionerna istället körs på de villkor som finns uppställda. Villkor, som benämns i kapitel 4.4.2, är bland annat om fjärrkontrollens lägesbrytare är inställd på automatisk flygning eller ej, om quadrotorn har lättat från marken eller om användaren gett ett startkommando från en ansluten Wi-Fi-enhet. Vid normal automatisk flygning kommer dock de flesta av programmets funktioner att köras. Samtliga PID-regulatorer arbetar oberoende av varandra och stabiliserar systemet med en låg samplingsfrekvens på $f_s \approx 11$ Hz.

5.4.2 Arduino Uno

En pålitlig kommunikationsplattform har skapats på ett Arduino Uno-kort som tillåter extern kommunikation med quadrotorn via Wi-Fi och intern kommunikation med Arduino Mega-kortet via I^2C-bussen. Både den interna kommunikationen och den externa kommunikationen med användaren fungerar med låga paketförluster.

5.5 GPS-kommunikation

GPS-positionering har sina begränsningar på grund av bestämmelser i USA. Trots detta har GPS-positioneringen en noggrannhet på $\pm$5m. En elektronisk kompass implementerades aldrig, då den inbyggda magnetometern i MPU-9150 enheten utsattes för stora störningar. Detta gjorde att MPU-9150 ansågs ha begränsade möjligheter att användas som riktningsgivare. GPS-enheten har möjlighet att beräkna rörelseriktningen på quadrotorn, förutsatt att den är i rörelse. Den stora begränsningen här är att GPS-enheten ej vet vad som är fram och bak, utan det är rörelseriktningen som beräknas, oberoende av quadrotorns orientering.

5.6 Användargränssnitt

Användargränssnittet som erhölls var baserad på CLI-kommandon över telnet. Arduino Uno tar emot ett kommando som en textsträng som vidarebefordras till Arduino Mega. En rad instruktioner definierades in i Arduino Mega som sedan exekverade dessa. En lista av de instruktioner som quadrotorn kan exekvera finns i Tabell 5.2

Instruktion	**Beskrivning**
reset	Stänger av & återställer quadrotorn till startläge
set_height <*heltal*>	Sätter börvärde för höjden till <*heltal*> centimeter
set_ail <*heltal*>	Sätter börvärde för roll till <*heltal*> grader
set_ele <*heltal*>	Sätter börvärde för pitch till <*heltal*> grader
thr_p <*heltal*>	Sätter P-parametern till <*heltal*> för höjdregulatorn
thr_i <*heltal*>	Sätter I-parametern till <*heltal*> för höjdregulatorn
thr_d <*heltal*>	Sätter D-parametern till <*heltal*> för höjdregulatorn
ail_p <*heltal*>	Sätter P-parametern till <*heltal*> för rollregulatorn
ail_i <*heltal*>	Sätter I-parametern till <*heltal*> för rollregulatorn
ail_d <*heltal*>	Sätter D-parametern till <*heltal*> för rollregulatorn
ele_p <*heltal*>	Sätter P-parametern till <*heltal*> för pitchregulatorn
ele_i <*heltal*>	Sätter I-parametern till <*heltal*> för pitchregulatorn
ele_d <*heltal*>	Sätter D-parametern till <*heltal*> för pitchregulatorn
start	Startar quadrotorn. Om den är i automatiskt läge kommer den att lyfta till satt börvärde

Tabell 5.2: *Instruktionstabell för Wi-Fi-kommandon*

6 Slutsats

6.1 Resumé

Automatiserade drönare av olika slag blir vanligare i olika sammanhang, särskilt små, obemannade och luftburna farkoster. På dessa kan man montera kameror som kan användas i olika sammanhang. Exempelvis kan olika spaningsuppdrag genomföras, ansiktsigenkänning kan då programmeras i mjukvaran som kopplas mot kameran. Dessa farkoster kan erbjuda tjänster från säkerhetsuppdrag till olika estetiska verksamheter.

Denna rapport fokuserar huvudsakligen på stabiliseringen av en quadrotor, men går även in på hur en användare kan kommunicera samt även skicka nya instruktioner till farkosten. Rapporten beskriver stegsvarsanalysen samt systemidentifikationen hos quadrotorn för att kunna skapa matematiska modeller av processen. Den beskriver även hur stabilitet uppnåddes, programstrukturen som utvecklades, kommunikationsutvecklingen och vilka testsystem som användes.

6.2 Kritisk diskussion

Resultaten som erhölls uppfyllde förväntningarna. Delmålen som sattes upp och Gantt-schemat, som finns bifogat i Appendix E, följdes och utfördes systematiskt. Dock var vissa moment mer tidskrävande än förväntat, så Gantt-schemat följdes, men modifierades något under arbetets gång.

6.2.1 Stegsvarsanalysen

Stegsvarsanalysen gav de nödvändiga resultaten för att kunna konstruera modeller som kunde simulera olika förhållanden. Det ska dock påpekas att dessa modeller, då de kan fungera bra i simuleringar, är mindre bra för att förutse hur systemet beter sig i verkligheten. Detta beror troligtvis på begränsningar hos testriggarna. Olika faktorer så som friktion och olika begränsningar kan påverka mätdatan och systemet negativt. Trots detta så håller de olika modellerna bra. Insvängningsförloppet går dock snabbare i verkligheten än vad de flesta modellerna ger. Däremot så erhölls mycket annan viktig information från stegsvarsanalysen så som dödtid och förhållandet mellan varvtal och översväng.

6.2.2 Stabilisering

Enligt en hypotes från vår planeringsrapport påstår vi att MPU-9150 skulle räcka för att stabilisera quadrotorn. Detta provades inte, då grovstabiliseringsenheten FY-901 användes vid dimensioneringen av PID-parametrarna. Det ansågs att

FY-901 enheten var vital för att inom vår tidsram uppnå stabila autonoma flygningar. Med hjälp av Ziegler-Nichols svängningsmetod erhölls resultaten. Precis som med stegsvarsanalysen finns det en viss risk att olika begränsningar i våra testriggar gjorde att de PID-parametrar som erhölls ej var optimala. Dock var de tillräckligt bra för att stabilisera quadrotorn. Detta var särskilt uppenbart när höjden skulle stabiliseras. Friktion i riggen gjorde att översvängarna hos de tidigare försöken dämpades. Detta ledde till en falsk trygghet. Flygningar med dessa parametrar gjorde att enheten sköt uppåt tills vi hann reagera och stänga av farkosten. Problemet löste sig då en den slutgiltiga testriggen konstruerades. Denna dämpade förvisso översvängen också, dock inte lika mycket som tidigare revisioner av Testrigg B. Med resultaten från stegsvarsanalysen angående förhållandet mellan varvtal och översväng blev det uppenbart att en "take-off" rutin behövde implementeras. Detta löste problemet med dramatiska översvängar och resultatet blev tillfredsställande.

Överlag så fungerade stabiliseringsmetoden bra, även med den låga samplingsfrekvensen på ungefär 11 Hz.

6.2.3 Programvaruutveckling

De program som styr quadrotorn har en enormt viktig uppgift. De ansvarar för all stabilisering, all kommunikation och all inläsning av sensorer. Ett program som har så pass mycket ansvar kan alltid förbättras och optimeras. Det anses dock att de program som utvecklats under projektet erhåller tillräckligt hög kvalitet för att få en stabil och säker flygning. Funktioner optimerades hela tiden för att samplingsfrekvensen på mätsignaler skulle bli så hög som möjligt. En snabbare processor skulle kunna erbjuda fler kontroller av mätvärden och högre samplingsfrekvens, men då tiden för konvertering till en ny plattform ansågs vara för stor slopades denna idé.

6.2.4 Kommunikation

Då kommunikation med quadrotorn är kritisk ur säkerhetssynpunkt lades stor vikt vid att få denna att fungera felfritt. Vid flygningar utanför våra testriggar visade det sig att vi prioriterat kommunikationen rätt. Vid testflygningar kunde quadrotorn bete sig på ett sätt som vi inte hade räknat med och då har vi flera sätt och kommunikationsvägar för att snabbt och smidigt ta manuell kontroll över flygningen eller helt enkel stoppa alla motorer omedelbart. Dels har vi telnetanslutningen där vi kan skicka ett kommando till quadrotorn att antingen landa eller stänga av motorerna direkt. Vid en nödsituation kan vi även använda en brytare på fjärrkontrollen som direkt ger användaren manuell kontroll så att denne kan landa quadrotorn på ett säkert sätt.

6.2.5 Navigering

Då det i dagsläget inte går att orientera quadrotorn åt något väderstreck blir all form av navigering nära på omöjlig eftersom quadrotorn måste ha en referenspunkt att gå efter. Utan kompass blir det som när man försöker ta sig runt i ett kolsvart rum, man har ingen referenspunkt att gå efter och måste därför prova sig fram. Samma sak gäller quadrotor, man kan prova åka en bit framåt för att se vad GPS-enheten registrerar för väderstreck men det är troligtvis ingen praktisk lösning.

6.3 Frågeställningar

Autonoma drönare av olika slag utvecklas runt om i världen. När denna teknik har mognat är det inte helt orimligt att föreställa sig autonoma luftburna farkoster som en given del utav samhället. Polismyndigheter och säkerhetstjänster skulle dra stor nytta av att ha lätta, små spaningsdrönare som har möjligheten att leta efter specifika individer, eller hålla koll på större folksamlingar. “Search and rescue”-uppdrag skulle troligtvis tjäna på autonoma drönare också. Möjligheten att täcka stora områden på kort tid är väl värt en satsning på denna teknik.

Dock kan även nya problem uppstå. Det finns även stor potential för missbruk av denna teknik, integritetsfrågor kan komma bli aktuella, då vem som helst potentiellt kan ha möjligheten att börja lagra information om individer och deras vanor, genom att ha en drönare följa efter en individ.

Utöver detta är säkerhet ett stort bekymmer också. Individer kan potentiellt hacka sig in i drönarens system och ändra dess instruktioner, om säkerheten är låg. Om en drönare är helt autonom så existerar även risker att tekniken används i ett skadligt syfte. Om ingen styr enheten blir det väldigt svårt att spåra den tillbaka till dess ursprung. Detta öppnar nya möjligheter för terrorism, då det inte krävs en stor budget att skapa en liten autonom drönare.

Farkosten drivs även på elektricitet, vilket gör att energin som driver den kan väljas fritt. För autonoma drönare som kan tänkas vara luftburen långa perioder, kan det konstrueras dockningsstationer som markeras med GPS koordinater. Dessa dockningsstationer kan ha solceller på väggar och tak för att på så sätt tillåta enheten att vara självförsörjande från ett energiperspektiv.

6.4 Vidareutveckling

Denna teknik ger stora utrymmen för vidareutveckling. För specifikt quadrotorn som denna rapport täcker är denna listan lite kortare. Stabiliteten är som sagt bra. Men den kan bli bättre. Genom att effektivisera testriggar i avseende på friktionselimination så kan bättre regulatorer dimensioneras som potentiellt utesluter FY-

901 stabilisatorn helt och hållet. När enheten är stabil så skulle det med fördel kunna läggas till flera sensorer som håller koll på dess omgivning. Genom en snabbare processor skulle en virtuell värld kunna målas upp och en effektiv navigationsalgoritm konstrueras som tillåter enheten att förflytta sig från punkt A till punkt B och även navigera förbi potentiella hinder, så som träd, buskar och senare kanske även människor. Detta skulle kunna ske genom att montera PING)))-sensorer på vingarna för att på så sätt skapa sig en uppfattning om dess omgivning. Dock så tillåter detta inte förflyttning i högre hastigheter, då räckvidden på dessa sensorer är begränsade. Detta skulle även resultera i att quadrotorn endast har information om föremål på samma höjd som den själv. Avståndsmätare som använder sig av laser istället för ljud hade troligtvis varit mer effektivt. Dock så begränsas den i detta fall av föremål som är vinkelräta mot sensorn annars kan lasern reflekteras och inte registreras. Författarna av denna rapport tror starkt på att video borde vara den effektivaste lösningen på omgivningsidentifikation. Genom att montera kameror i par kan en stereoskopisk video användas. Genom att tolka pixlarna går det att bestämma avstånd till föremål med denna metod. Kamerapar, utrustade med vidvinkeloptik skulle med fördel monteras på varje vinge så att en virtuell bild av quadrotorns omgivning kan skapas i processorn. Processorn i detta fall skulle behöva vara väldigt kraftig, då videosignaler är väldigt prestandakrävande att behandla.

6.5 Miljö

Idag står all utveckling av teknik inför en stor utmaning med tanke på rådande klimatproblem. För att få bukt med problemet behövs avancerade tekniska lösningar som minskar utsläpp av gaser och andra ämnen som inte kan hanteras av jordens ekosystem. En elektrisk quadrotor är en sådan lösning. Vid, till exempel, filminspelning kan en radiostyrd quadrotor med en kamera användas istället för att hela filmteamet åker upp med kameran i en fullstor helikopter. I stort sett de flesta sammanhang där det normalt sett hade använts stora helikoptrar kan små quadrotorer ersätta dessa. Räddningsuppdrag, polishelikoptrar, listan kan göras lång med tillämpningar på eldrivna luftfarkoster.

Referenser

[1] B. Thomas, *Modern Reglerteknik*. 113 98 Stockholm: Liber AB, 2008, vol. 4.

[2] S. Lundell, "Undersökning av inställningsmetoder för pid-regulatorer," Master's thesis, Chalmers University of Technology, 2012.

[3] L. T. Högskola, *Föreläsning 9*. [Online]. Available: http://www.control.lth.se/media/Education/EngineeringProgram/FRT110/2012/F09-slides6.pdf

[4] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar, "Influence of aerodynamics and proximity effects in quadrotor flight," in *Proceedings of the International Symposium on Experimental Robotics*, June 2012.

[5] M. Halloram and S. O'Meara, "Wing in ground effect craft review," The Sir Lawrence Wackett Centre for Aerospace Design Technology Royal Melbourne Institute of Technology, DSTO Aeronautical and Maritime Research Laboratory, PO Box 4331,Melbourne, Victoria, 3001, Australia, Tech. Rep. AR-010-831, Februari 1999.

[6] ´Arduino´. Arduino mega. [Online]. Available: http://arduino.cc/en/Main/arduinoBoardMega

[7] 'Arduino'. Arduino uno. [2 Mar 2014]. [Online]. Available: http://arduino.cc/en/Main/arduinoBoardUno

[8] 'Adafruit'. Adafruit ultimate gps logger. [Online]. Available: https://learn.adafruit.com/adafruit-ultimate-gps-logger-shield/overview

[9] J.-M. Irazabal, *AN102116-01 I2C Manual*, Philips Semiconductors, 2008.

[10] InvenSense, *MPU-9150 Product Specification*, 4th ed., InvenSense Inc., Sunnyvale, California, USA, 2013.

[11] TutorialsPoint, *Learning Wi-Fi*. [Online]. Available: http://www.tutorialspoint.com/wi-fi

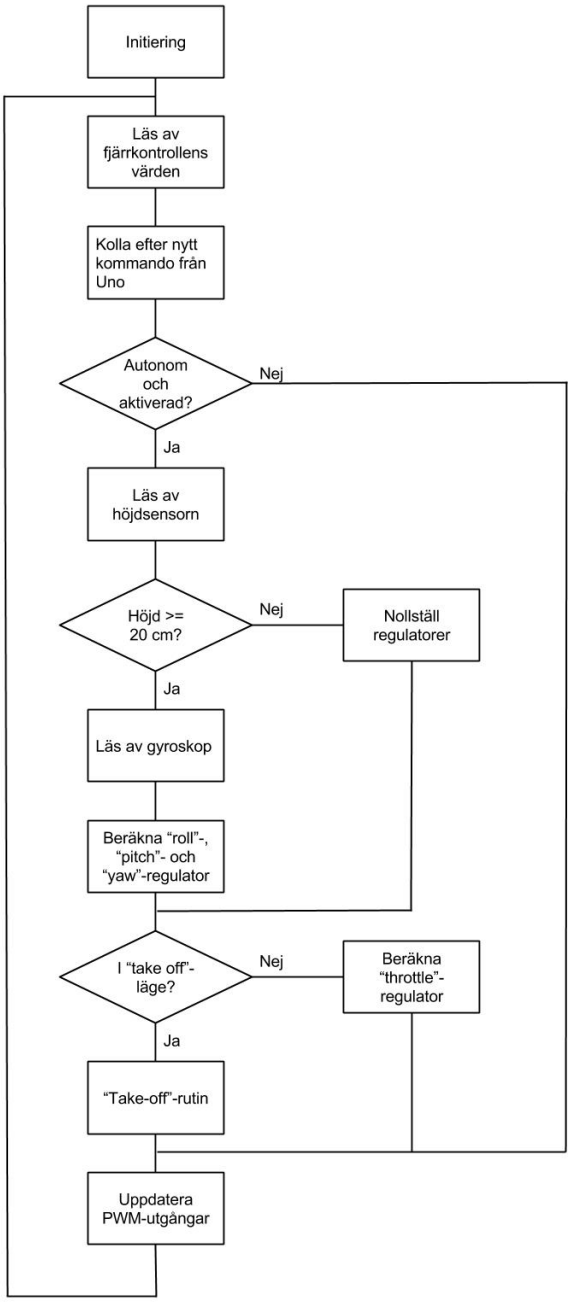
[12] J. F. Kurose and K. W. Ross, *Computer Networking*, 5th ed. Boston, Massachussetts 02116: Pearson Addison-Wesley, 2010.

[13] A. El-Rabbany, *Introduction to GPS:the global positioning system*. Norwood, Massachussetts 02062: Artech house, 2002, 'ISBN 1-58053-183-0'.

[14] B. Shahian and M. Hassul, *Control System Design Using MatLab*. Englewood Cliffs, New Jersey 07632: Prentice-Hall, Inc., 1993, vol. 4.

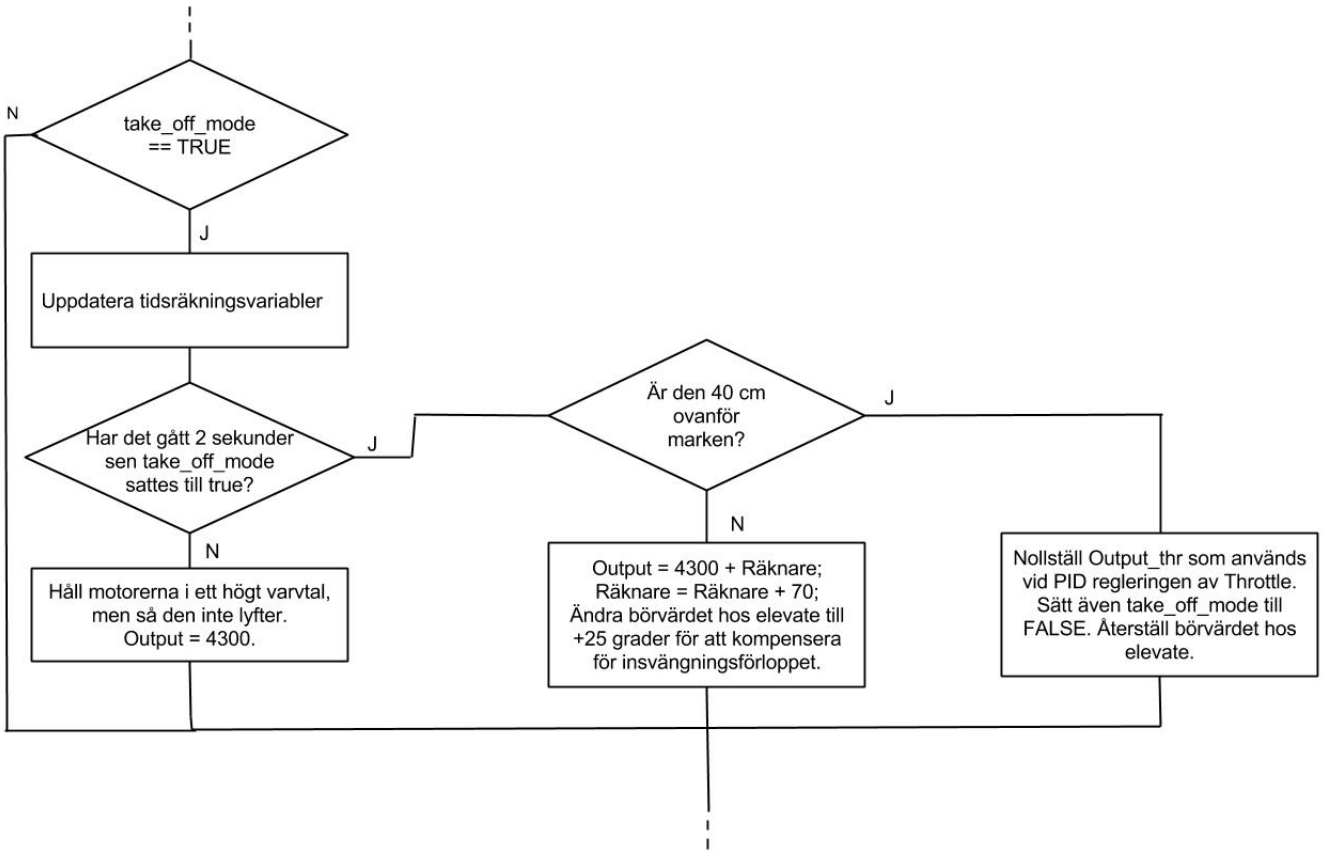
[15] B. Beauregard. Improving the beginner's pid. [3 Jun 2014]. [Online]. Available: http://brettbeauregard.com

A Flödesschema för Arduino Mega



Figur A.1: Flödesschema för Arduino Mega

B Flödeschema för “take off”-rutinen



Figur B.1: Flödesschema för Take-off rutinen

C Specifikationer för Arduino-kort

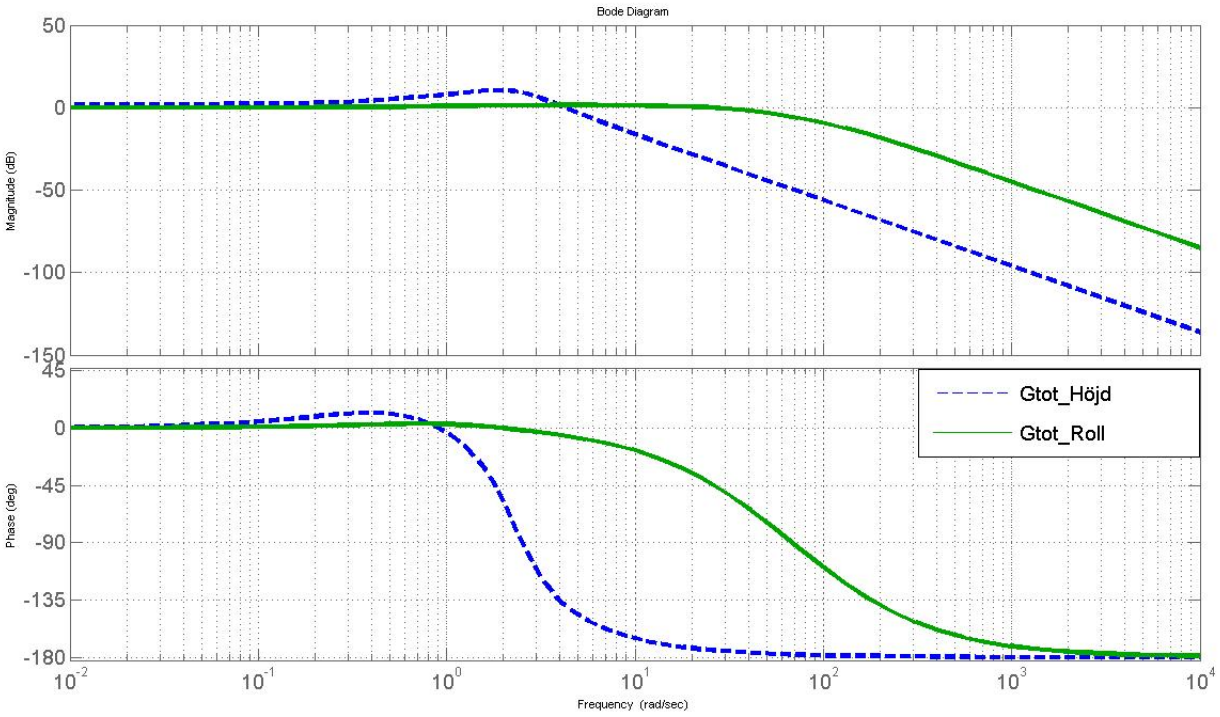
MCU	ATmega2560
Arbetsspänning	5V
Inspänning	ATmega2560
Inspänning (gränser)	ATmega2560
Digitala I/O pins	54
Analoga I/O pins	16
Likström per I/O pin	40 mA
Likström för 3.3V pin	50 mA
Flashminne	256 KB
SRAM	8 KB
EEPROM	4 KB
Klockfrekvens	16 MHz

Tabell C.1: Specifikationer för Arduino Mega

MCU	ATmega328
Arbetsspänning	5V
Inspänning	ATmega2560
Inspänning (gränser)	ATmega2560
Digitala I/O pins	14
Analoga I/O pins	6
Likström per I/O pin	40 mA
Likström för 3.3V pin	50 mA
Flashminne	32 KB
SRAM	2 KB
EEPROM	1 KB
Klockfrekvens	16 MHz

Tabell C.2: Specifikationer för Arduino Uno

D Bodediagram



Figur D.1: Bodediagram för höjdregleringssystemet och roll-regleringsystemet

E Gantt-schema för projektet