Högskoleingenjörs uppsats i Elektronik
Institutionen för systemteknik, Linköpings universitet, 2018

Simulering och implementering av ett elektriskt motorsystem för minimering av hastighetvariationer

Oskar Jonsson och Andreas Öhman

LINKÖPING UNIVERSITY

Högskoleingenjörs uppsats i Elektronik

Simulering och implementering av ett elektriskt motorsystem för minimering av hastighetvariationer

Oskar Jonsson och Andreas Öhman

LiTH-ISY-EX-ET–18/0479–SE

Handledare: **Kristoffer ekberg**
ISY, Linköpings universitet
Bo Winqvist
Saab Dynamics
Oscar Nilsson
Saab Dynamics

Examinator: **Mattias Krysander**
ISY, Linköpings universitet

Institutionen för systemteknik
Linköping University
581 83 *Linköping*

Sammanfattning

Syftet med examensarbetet är att utse en lämplig ersättare för en motor i en av Saab Dynamics produkter. Den tidigare motorn är en synkronmotor av typen hysteresmotor och nu söks en billigare lösning med liknande prestanda. Applikationen kräver att motorn från stillastående ska kunna accelereras till 600 varv per minut på maximalt 0,5 sekunder. Efter 0,5 sekunder ska medelhastigheten per varv maximalt avvika med 0,2 promille. I utveckling av en ny lösning ingår val av motortyp, styrelektronik och givare samt design av regleralgoritmer.

I rapporten beskrivs ett antal motorer som kan vara lämpliga ersättare till den tidigare motorn samt deras styrkor och svagheter. Sedan följer en undersökning och implementation av en av de lämpliga kandidaterna, en borstlös likströmsmotor. Val av givare och styrteknik för den motortypen beskrivs och motorns lämplighet undersöks i en modell skapad i Mathworks Simulinkmiljö där ett reglersystem implementeras. Simuleringar visa att den valda motorn är lämplig för en praktisk implementation. Slutligen beskrivs konstruktionen av en testuppställning som innefattar motor, givare och styrteknik implementerad i en processor som ställer ut strömmen till motorn för att uppnå önskad hastighet. Uppställningen uppfyller kravet i testkörningar och i den bästa körningen är hastighetsvariationerna 7,4 gånger mindre än det specificerade kravet.

Tack

Vi vill tacka Saab Dynamics för möjligheten att utföra vårt examensarbete i Karlskoga. De har varit ett intressant projekt, lärorikt och inte minst roligt att se arbetsplatsen. Även är vi tacksamma för att de bidragit med boende som gav oss möjligheten att utföra arbetet i Karlskoga.

Stort tack till våra handledare Bo Winqvist och Oscar Nilsson. Där Bo Winqvist har bidragit med mycket kunskap och vägledning genom projektets gång och Oscar Nilsson introducerade oss till Saab samt har hjälpt till med mycket administrativt.

Vi vill även tacka vår examinator Mattias Krysander och handledare Kristoffer Ekberg på Linköpings Universitet.

Tack till våra fäder som hjälpt till med transport från och till Karlskoga.

Linköping, Juni 2018
Oskar Jonsson och Andreas Öhman

Innehåll

Notation

Förkortningar	**Betydelse**
BLDC	Borstlös likströmsmotor
PMDC	Permanetmagnetiserad likströmsmotor
PM	Permanetmagnet
RPM	Varv per minut

Variabler	**Betydelse**
$SYST\_FREQ$	Antal interna avbrott per sekund
$TIMER1$	Räknar antalet pulser för pågående rotation
ENX_COUNT	Totala antalet pulser över en rotation
POS_INCR	Radianer per puls
fas_pos	Rotorns vinkel $[rad]$
$tmr1$	Antal pulser vid det pågående systemavbrott
$tmr2$	Antal pulser vid det föregående systemavbrottet
$delta\_tmr$	Antal pulser mellan två systemavbrott
$MY2\_PI\_3$	$2\pi/3$
$rotor\_speed$	Hastigheten på rotorn $[rad/s]$
$INVK_t$	Inversen av motorns moment konstant
K_P	Proportionell försträknings konstanten
K_I	Integrerande försträknings konstanten
$ROTOR\_SPEED\_SETPOINT$	Den begärda hastigheten på motorn $[rad/s]$
Tq_dem	Det begärda momentet beräknat med regulatorn $[Nm]$
$ipeak$	Amplituden på strömmen $[A]$
$curr\_ph\_a$	Strömmen för fas a $[A]$
$curr\_ph\_b$	Strömmen för fas b $[A]$
SC_IPHASE_UPH	Konstant för att få korrekt spänning till D/A
$TIMER2$	Räknar upp varje klockpuls i processorn
$temp\_timer$	Array med föregående klockpulser vid varje puls på ett varv
$temp1$	Lagrar antalet klockpulser vid start av avbrottet
$pulses$	Vilken puls rotorn befinner sig vid på pågående rotation
$clock\_timer$	Differensen mellan pulses(n) och pulses(n-12) i klockpulser

$timer\_per\_rot$	Tid för en rotation(Matlab)$[s/varv]$
rot_per_sec	Antal rotationer under en sekund(Matlab) $[varv/s]$
$e(t)$	Reglerfelet
u_0	Önskad utsignal
$y(t)$	Systemets utsignal
K_i	Integrerande förstärkning
K_p	Proportionell förstärkning
e_x	Inducerad spänning i fas x $[V]$
i_x	Strömmen i fas x $[A]$
V_x	Spänningen i fas x $[V]$
R	Fasresistans $[R]$
L	Fasinduktans $[L]$
θ	Rotor-vinkel $[rad]$
K_t	Moment-konstanten $[mNm/A]$
K_e	Motemk-konstanten $[rad/V]$
ω	Vinkelfrekvens $[rad/s]$
T_e	Totala momentet $[Nm]$
J	Rotorns tröghet $[gcm^2]$
β	Viskösa friktionskonstanen $[Nm/\omega]$
ω_{nl}	Varvtal vid ingen last $[rad]$
T_{nl}	Moment vid ingen last $[Nm]$

Figurer

Tabeller

1

Introduktion

Saab Dynamics önskar att ett projekt utförs för att undersöka möjligheter att ersätta en motor i en av deras produkter. Den använda motorn i produkten är en hysteresmotor som är en typ av synkronmotor. Denna motor används för att hålla en konstant hastighet med låga hastighetsvariationer under en rotation. Då denna motortyp är ovanlig vill de att ersätts med ett annat alternativ. Hysteresmotorns egenskaper är att den har en mjuk start men speciellt att den har väldigt litet hastighetsfel under rotation, den användes kommersiellt i bland annat skivspelare och klockor [17].

Flertalet motortyper har undersökts för att finna en lämplig kandidat med små hastighetsvariationer. I detta fall låg fokus på varför dessa variationer skapas och hur de kan minskas för få en så jämn rotation som möjligt. Variationerna som uppstår när motorn roterar beror på att vridmomentet inte är konstant. Amplituden för variationerna varierar på typen av motor och dess kvalité. Rapporten avser därför att finna en lämplig ersättare till den tidigare motorn som sedan skall modelleras, simuleras och implementeras för att studera om det är möjligt att hålla en jämn hastighet med en billigare lösning.

1.1 Syfte

Syftet med denna rapport är att hitta en lämplig ersättare för hysteresmotorn till en av Saab Dynamics produkter. Hysteresmotorn håller ett konstant varvtal med små variationer. Lastfallet för motorn består av ett tröghetsmoment och friktion mot luft och lager. För att uppnå kraven med den ersättande motorn antas det nödvändigt att använda någon typ av reglering och givare för att beräkna varvtal. Motorns effektområdet är kring en watt.

1.2 Mål

Målet är att utse en lämplig motor där förutsättningar för jämn rotation finns med implementation av en regulator. Den valda motorn blir även grunden för valet av givare, då kommuteringsättet som användas för motorn kan kräva olika förutsättningar i form av återkoppling. Ett reglersystem behöver implementeras för att motorn skall klara kraven. Kraven är att motorn måste kunna accelerera upp från stillastående till 600 RPM på 0,5 sekunder med ett maximalt hastighetsfel på 0,2 ‰ per rotation.

1.3 Avgränsningar

Med tanke på projektets längd och det låga lastfallet avgränsas projektet till motorer drivbara med likström.

Motortypen asynkronmotor kommer inte att inkluderas i rapporten då denna motor främst används vid större lastmoment och anses därför inte lämplig för denna uppgift [12]. De synkronmotorer som är lämpliga inom detta område kommer att innefattas i rapporten vilket är den borstlösa likströmsmotorn. Det finns flertalet avancerade specialgjorda synkronmotorer men dessa har vanligen en hög kostnad [18].

Då motorn skall hålla en konstant hastighet och inte utsätts för några lastförändringar kommer rapporten inte utvärdera systemet vid last eller hastighetsförändringar förutom vid uppstart av motorn.

1.4 Frågeställningar

Då målet är att finna en lämplig ersättare för hysteresmotorn strävar denna uppsats efter att besvara följande frågeställningar.

- Vilken motortyp har jämnast friktion och jämnt elektromekansiktmoment över ett varv och minst hastighetsstörningar?
- Vilken givare kan mäta motorns rotationshastighet med tillräckligt hög precision och med hög frekvens?
- Vilket styrsystem är mest lämpligt för att uppnå kraven med den valda motorn?

Med frågeställningarna i åtanke, väljs en lämplig motor och varvtalsgivare. Komponenterna används sedan tillsammans med en lämplig reglerstrategi för att uppfylla kraven.

2
Studie

I det här kapitlet presenteras den studie som gjordes inför valet av motor och medbehövd teknik. Olika motortyper studeras, specifikt om deras egenskaper är relevanta för att uppnå en jämn konstant hastighet. Avgörande för motorn är momentrippel och kommuteringsätt som studeras i kapitlet.

2.1 Elektriska motorer

I en typisk motordrift tillförs ström till motorn för att generera ett magnetiskt fält vid statorn och ett vid rotorn. Magnetfälten skapar en kraft mellan varandra och ett vridmoment uppstår som får rotorn att rotera. Skiftningar i magnetfälten sker för att bibehålla momentet. Om det är rotorn eller statorn som har ett växlande magnetfält beror på typen av motor. Exempelvis så har likströmsmotorer generellt ett statiskt magnetfält vid statorn tillfört från antingen likström eller en permanentmagnet och strömriktningen i rotorn växlas kontinuerligt.

2.1.1 Likströmsmotorn

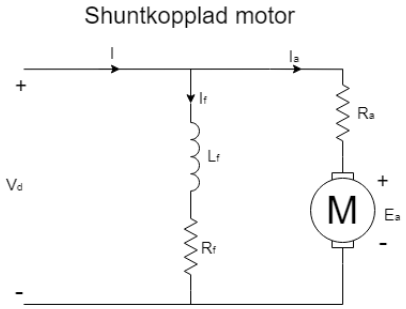
Likströmsmotorn är en vanlig elektrisk motor och har stor mångsidighet, med olika möjligheter för magnetisering av motorn som till exempel separat-, serie-, shunt-, kompound- och permanentmagnetsmagnetisering. De olika typerna av likströmsmotorer skiljer sig i karaktär och har olika egenskaper. Likströmsmotorn är vid de flesta typer av magnetisering enkel att styra och har därför användningsområden där stor variation i varvtal eller exakt kontroll av motorn behövs [6].

Likströmsmotorn består av en rotor, stator och en kommutator. Rotorn och statorn har varsin lindning som alstrar magnetiska fält. Samverkan mellan dessa två fält skapar ett moment som får rotorn att rotera. För att bibehålla momentet skiftas strömmens riktning

i rotorlindningen två gånger per varv via en kommutator som ligger mot ledare så kallade borstar som rotorlindningarna är kopplade till, detta gäller för en motor med två poler. En motor med fler poler kommer kommuteringen att ske ytterligare gånger per rotation [12][6].

Shuntmagnetiserad motor

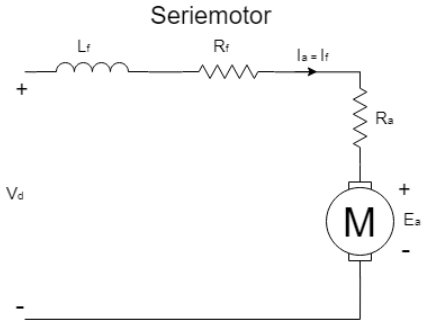
Shuntmotorn har parallellkopplad rotor och stator. Det medför att strömmen i statorn inte påverkas av den last som läggs på motorn, detta går att se i Figur 2.1 där rotorn och statorn har två separata strömmar. Med en konstant spänningskälla kommer statorn i motorn att ha nära konstant magnetiskt fält vilket gör den lätt att hastighetsreglera [12][7].



Figur 2.1: *Kopplingsschema över en shuntmotor.*

Seriemagnetiserad motorn

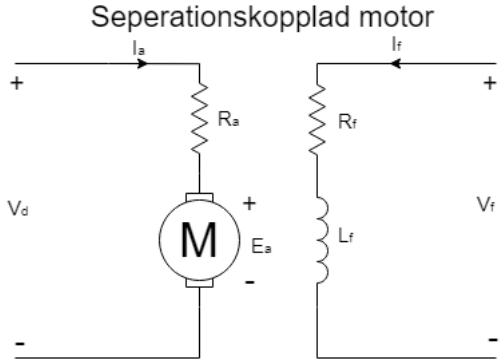
I en serielindad likströmsmotor är de två lindningarna i stator och rotor kopplade i serie. Eftersom stator- och rotorströmmen är den samma kommer det magnetiska flödet att förändras i takt med rotorströmmen. I och med detta faller varvtalet kraftigt i takt med ökad belastning. I Figur 2.4 illustreras hur en seriemotor är kopplad och att samma ström passerar genom både stator och rotor. Fördelen med en seriemotor är att den har högt startmoment samt högt vridmoment vid låga hastigheter, den är dock svår att hastighetsreglera. Det finns också risk för rusning när den är lastfri eller om lasten tas bort [12][7][6].



Figur 2.2: *Kopplingschema över seriemotorn. Notera att det går samma ström genom bägge lindningarna.*

Separatmagnetiserad motor

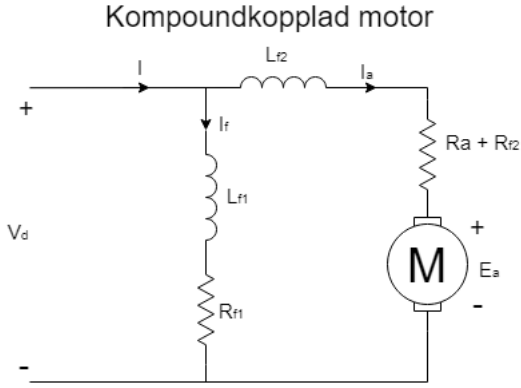
På separatmagnetiserade motorer har rotorlindningen och statorlindningen separata spänningskällor se Figur 2.3. Med separata spänningskällor har statorn en egen strömkälla och därav ett stadigt magnetiskt fält. Den har liknande karaktäristik med shuntmotorn [7].



Figur 2.3: *Kopplingschema över serperationsmotorn. Både fältlidningen och rotorlindningen har separata spänningskällor vilket ger en stabil ström i fältlindningen*

Kompoundsmagnetiserad motor

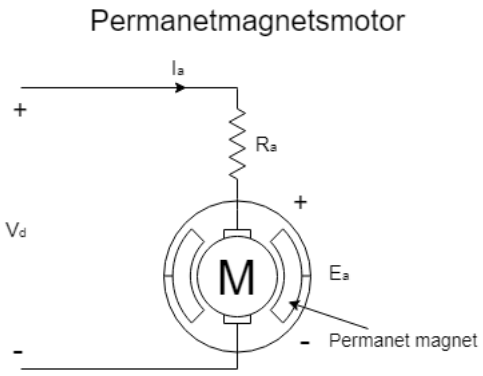
Kompoundmotorn är parallellkopplad som en shuntmotor men har en extra statorlindning i serie med shuntkopplingen, se koppling i Figur 2.4. Detta medför att motorns karaktäristiska egenskaper blir ett mellanting av serie- och shuntmotorn. Denna motortyp används för att optimera specifika egenskaper [12][7].



Figur 2.4: *Kopplingsschema över kompundmotorn. Till skillnad från shuntmotorn i figur 2.1 har kompoundmotorn en extra fältlindning som är kopplad i serie med rotorlindningen.*

Permanentmagnetiserad motor

I permanetmagnetsmotorn ersätts statorns lindning med en permanentmagnet, koppling syns i Figur 2.5. Detta eliminerar de negativa effekter vid förändring av strömmen i statorlindningen som kan uppstå i de andra typerna av likströmsmotorer. Därför är sambandet mellan varvtal och moment för denna motor i stort sett konstant och dess momentkurva är väldigt lik en shuntkopplad likströmsmotor men med högre startmoment. När permanetmagnetsmotorerna blev starkare och deras karaktäristik bättre tog de i stort sett över marknaden för borstade likströmsmotorer då de i förhållande till andra likströmsmotorer är mindre för samma effekt [7][6][24].



Figur 2.5: *Kopplingschema över en permanetmagnetsmotor(PMDC), där statorn består av permanetmagneter och rotorn har en lindning med mekanisk kommutation.*

Fördelar och nackdelar med likströmsmotrer

fördelarna med likströmsmotorn är att de flesta typer har högt startmoment, snabb acceleration och är enkla att hastighetsstyra. Det finns olika styrsätt för likströmsmotorer, det vanligaste är att justera rotorspänningen. Ofta används PWM-signaler för denna typ av styrning. Andra sätt att styra är genom justering av rotorresistans eller statorspänning [6]. Ofördelaktigt så slits borstarna och kräver underhåll vilket begränsar likströmsmotorns livstid och pålitlighet under längre användning. Likströmsmotorn har även en relativt hög friktion över en rotation eftersom borstarna alltid har kontakt med kommutatorn. [6][24].

Momentrippel är ett problem för de flesta motorer, likströmsmotorns främsta orsak till momentrippel sker när strömmen skiftar riktning. Detta sker när momentet börjar avta vilket är när rotorns poler ligger i fas med motsatt polaritet i statorn. I detta skede sker en kommutation och borstarna passerar kommuteringsegment för att skifta strömmens riktning och bibehåller vidmomentet. Baserat på antalet poler och motorns utformning kan kommutering ske flertalet gånger per rotation, vilket ökar periodtiden för momentripplet men amplituden blir lägre. Statorn kan också orsaka momentrippel om dess ström förändras, detta kan minimeras om statorn ersätts med en permanentmagnet [6][24].

2.1.2 Stegmotorn

Stegmotorn är en motor som drivs med likström via digitalteknik. Den har minst två faser som är placerade 90° från varandra med nord och sydpol för respektive fas som är 180° från varandra. Med denna typ av teknik är det möjligt att skicka pulser för att få motorn att ta ett steg i taget. Ett enkelt exempel är en stegmotor som har två statorlindningar och där rotorns tvärsnitt är utformat som ett plustecken av järn. I detta exempel är det möjligt att ta 90° per steg. Stegmotorn tar då ett steg genom att en fas strömsätts i taget men det är också möjligt att använda halvstegsprincipen där båda faserna strömsätts samtidigt för att ta 45° per steg [1][6].

Den vanliga typen av stegmotorer som används är av typen hybrid. Rotorn består av en permanentmagnet inlagd i en solid eller laminerad järnkärna (hybrid). Där har till exempel rotorn har 50 tänder och fyra statorfält med vanligen 48 tänder och har då en stegvinkel på 1,8 grader. Beräkningen av stegvinkeln är beroende av antalet statorfält multiplicerat med antalet rotortänder enligt

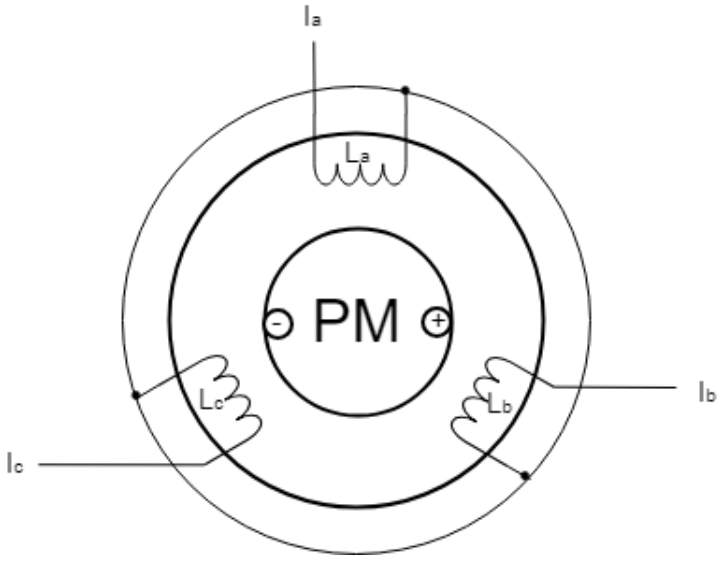
$$sl = \frac{360}{n \cdot p} \tag{2.1}$$

där p betecknar antalet rotor-tänder och n antalet faser [1][6].

Stegmotorns fördel är att den har ett högt vridmoment och det krävs inte några positionssensorer eller någon återkoppling för att styra dess position. Detta beror på att stegvinkeln är känd och motorns styrenhet kan då rotera motorn till en specifik position genom att skicka pulser. Då stegmotorn är borstlös är den också underhållsfri. Nackdelar med stegmotorn är att den har låg effektivitet och drar substantiell energi oavsett belastningen, den saknar också återkoppling för eventuella missade steg. Den har också ett statiskt fel som ökar med lasten på motorn, där felet är icke ackumulerande. Stegmotorns största nackdel är att den har ett momentrippel med hög amplitud då momentet behöver vara högre än friktionen mellan två steg [1][6][24].

2.1.3 Borstlösa likströmsmotorer

Den borstlösa likströmsmotorn (BLDC) är en synkronmotor med en permanentmagnet i rotorn. Motorn liknar en ut- och invänd borstad permanentmagnetlikströmsmotor (PMDC) då permanentmagneten sitter i rotorn och faslindningarna i statorn, illustration över motorns koppling syns i Figur 2.6. BLDC-motorn kommuterar elektroniskt till skillnad från DC-motorn som kommuterar mekaniskt med borstar. Detta medför en längre livslängd än den vanliga DC-motorn då borstarna slits ut med tiden och mindre friktion inuti motorn vilket leder till mindre momentrippel. Dock behöver BLDC-motorn en återkoppling av rotorns position relativt till faserna för att kommutera vid rätt tillfälle. Vanligtvis används halleffektsensorer placerade vid varje faslindning som ger ut en signal då rotorn passerar en fas. Kommutationen sker då genom att transistorer strömsätter faserna beroende på rotorns position. Vanligen har BLDC-motorer tre faslidningar men det finns de med både fler och färre. Fler lindningar leder till mindre momentrippel men ökar motorns inre komplexitet.

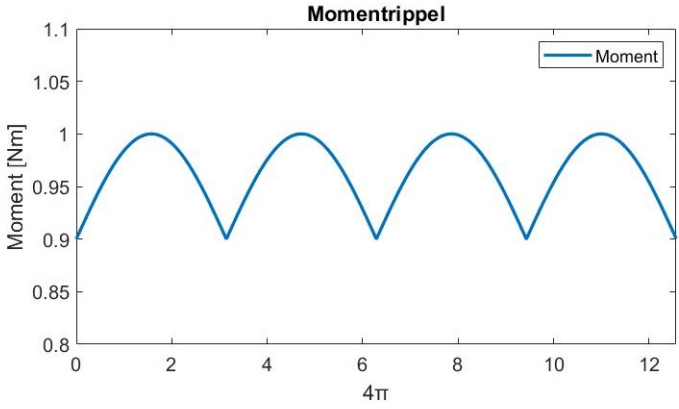


Figur 2.6: *Figuren illustrerar hur en borstlös likströmsmotor med tre statorlindningar och två rotorpoler ser ut. Rotorn finns i mitten och består av en permanentmagnet och omkring rotorn sitter statorn med tre lindningar a, b och c.*

2.2 Momentrippel

Momentet som orsakas av de magnetiska fälten i en elmotor varierar något beroende av rotorns polers position relativt till statorns polers position. Desto längre bort fältens motsatta poler är från varandra desto starkare är vridmomentet. När polerna närmar sig varann minskar kraften något och precis innan en kommutation sker är kraften som minst. Alltså oscillerar motorns vridmoment periodvis med rotorn. En graf över hur ett momentrippel kan se ut illustreras i 2.7. Oscillationerna är relativt små och försumbara för de flesta applikationer.

Momentrippel kan minimeras med fler faslidningar eller fler poler i rotorn. Genom att öka antalet poler i statorn kommer momentripplets frekvens öka och amplitud minska. Detta beror på att rotorns poler attraheras av statorns poler flera gånger per rotation och momentet jämnas ut. I till exempel en likströmsmotor med flera statorfaser kommer kommutation ske flera gånger per varv [24]. Det resulterar i ett jämnare vridmoment över hela varvet. Det går också att dämpa momentripplet med olika styrmetoder [23].

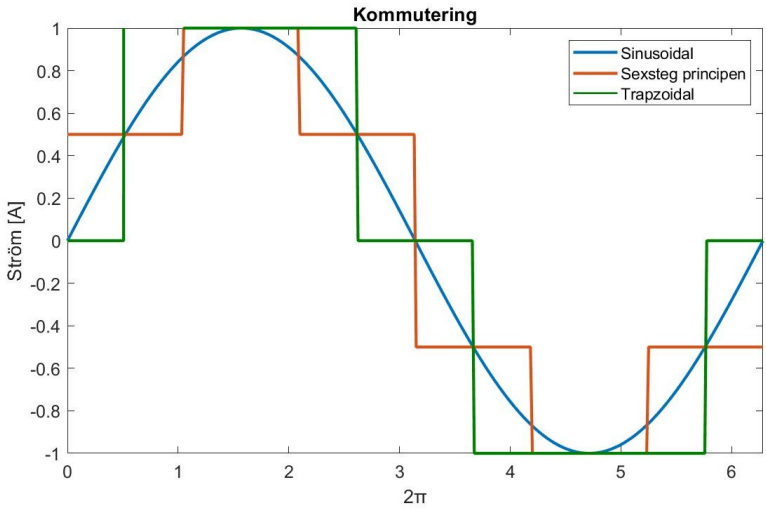


Figur 2.7: *Illustration över hur ett momentrippel kan se ut för en motor. I detta fall är det två rotationer med två rippel per rotation och det sker två kommutationer under ett varv.*

2.3 Styrteknik

Det finns flertalet olika styrtekniker för en borstlös likströmsmotor. Två av dom vanligaste är trapezoidal och sinusoidal [16]. Dessa tekniker har olika fördelar beroende på hur noggrann styrning som eftersträvas. I Figur 2.8 ses metodernas styrsignal för motorn.

Trapezoidal bör vara en ideal styrteknik då en borstlös likströmsmotor i teorin har en inducerad spänning som är trapetsformad. Den inducerade spänningen är dock i praktiken mer lik en sinusvåg vid denna styrning, vilket orsakar momentrippel. Motorn har därför en ojämn hastighet över ett varv, vilket är tydligt vid lägre hastigheter. Vid Trapezoidal kommutation används vanligen sexstegsprincipen. Med denna teknik är tre faser strömsatta samtidigt vilket ger ett högre moment än när endast två av lindningarna är strömsatta i taget som vid vanlig trapzoidal kommutering. Sexstegsprincipen används då strömmarna efterliknar sinsvågor vilket minskar momentrippel, se figur 2.8 [16].

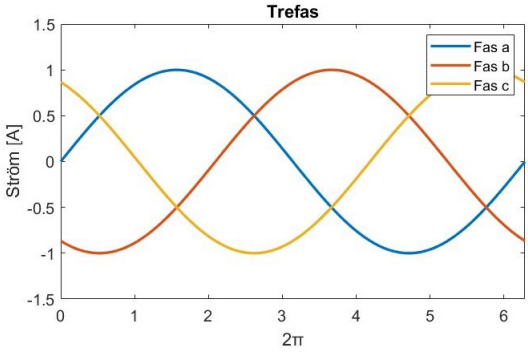


Figur 2.8: *Illustration över hur en fas strömsätts med sexstegsprincipen eller sinusoidal kommutering under en rotation.*

Sinusoidal kommutation är en vanligen använd styrteknik för implementationer av borstlösa likströmsmotorer för att minimera hastighetsvariation då det inte uppkommer momentrippel som vid trapezoidal styrning. Denna kommuteringsteknik strömsätter de tre faserna samtidigt med sinusformade strömmar. Strömmarna är fasförskjutna med 0, $-\frac{2\cdot\pi}{3}$ respektive $-\frac{4\cdot\pi}{3}$ radianer ifrån varandra. Strömmarnas faser kan ses i Figur 2.9 nedan och benämns a, b och c. Tekniken ger ett jämnt moment för exakt hastighetskontroll. Detta kan också ses i ekvationen 3.14 som visar att korrekt strömsättning ger ett felfritt moment. Vid väldigt höga hastigheter kan sinusoidal kommutering tappa sin fördel, då det krävs väldigt hög frekvens från givaren och en processor med hög klockfrekvens för bra representation av strömfaserna [16].

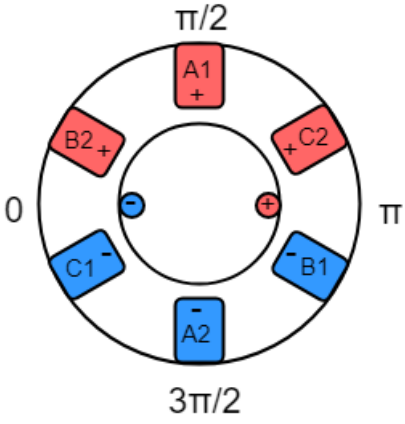
2.3.1 Strömsättning av faserna

För bästa resultat vid användning av styrtekniken sinusoidal kommutering behöver strömfasernas vinkel ligga 90° framför motorn [5]. Som exempel när rotorn är i position 0 bör strömfaserna ligga på vinklarna $\frac{\pi}{2}$, $\frac{-\pi}{6}$ och $\frac{-5 \cdot \pi}{6}$ radianer. I Figur 2.10 kan det ses hur faserna bör vara strömsatta när rotorn befinner sig i denna position.



Figur 2.9: *Illustration av hur de tre strömfaserna a, b och c ligger i relation till varandra med 120° fasförsjutning mellan varandra.*

Rotorns sydpol kommer vid rotation vara omkring 90° framför fas a som i figur 2.9 representeras av den blå grafen. Vid denna strömsättning kommer samtliga av motorns lindningar bidra med ett positivt moment under hela varvet och resulterar i det starkaste möjliga momentet.



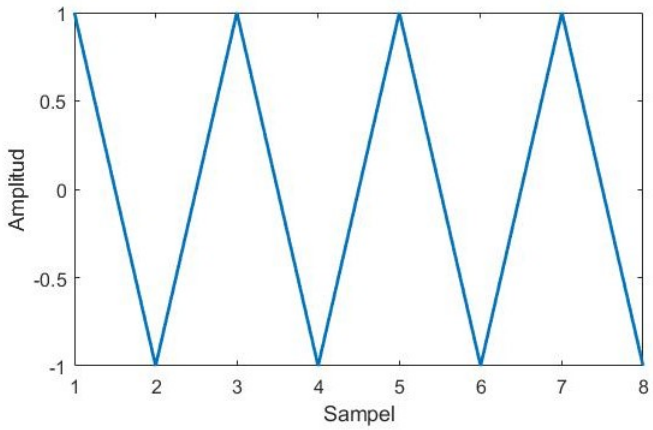
Figur 2.10: *Illustration av hur lindningarna bör vara strömsatta när motorn är i position noll, för att de skall rotera. Varje lindning har två kopplingar som representeras med 1 och 2. Strömmen tillförs till lindningen på den koppling som är röd och lämnar lindningen via den koppling som är blå*

2.3.2 Beräkningsfrekvens

Rotorns rotation under körning kan representeras som en sinusformig våg. Som beskrivet i avsnitt 2.3.1 så måste strömfaserna erhålla samma periodtid som rotorn dock med en fasförskjutning. Periodtiden bestäms av motorns hastighet, om motorn roterar ett varv per sekund blir alltså periodtiden för strömfaserna 1. Strömfasernas vinkel behöver beräknas minst två gånger per period enligt Nyqvistteoremet för att uppnå en sinusformig ström [9]. I Figur 2.11 visas hur en rekonstruktion med 2 sampel per period blir. Typiskt i kontrollsystem används en sampelfrekvens som är 8-10 gånger högre än signalen som skall rekonstrueras [9]. Den önskade hastigheten att hålla är 10 varv per sekund, därför är det rimligt att utföra minst 80-100 beräkningar av stömmarnas faser per sekund. Vid en beräkningsfrekvens på 100 Hz kan momentförändringar uppstå då motorn hinner rotera

$$\frac{2\pi}{10} = \frac{\pi}{5} \tag{2.2}$$

radianer innan nästa beräkning av strömmarna har skett. Utifrån detta kan det antas att en högre beräkningsfrekvens än 10 är att föredra.



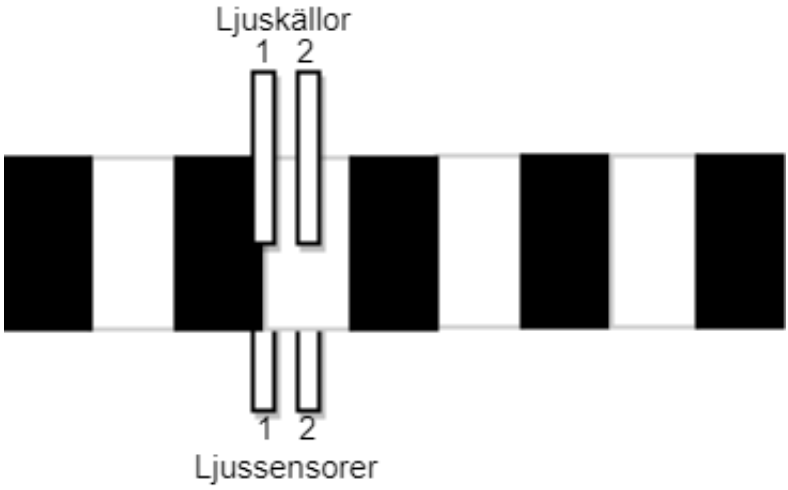
Figur 2.11: *Illusterar hur en sinusvåg blir om den representeras med två sampel per period, under fyra perioder.*

2.4 Pulsgivare

En optisk pulsgivare kan användas för att mäta position och rotationshastighet på en roterande axel. Pulser genereras via en ljuskälla, ljussensor och en cylindrisk skiva på motorns axel se figur 2.12. Den cylindriska skivan har slitsar med exakt samma diameter som avståndet mellan slitsarna för att kunna generera en puls som är hög lika länge som låg. När ljuskällan skickar ljus genom slitsarna som tas upp av ljussensorn på andra sidan av skivan, skickas en hög signal och vid avsaknad av ljus skickas en låg signal. Resultatet är en digital representation av pulserande ettor och nollor, se figur 2.13.

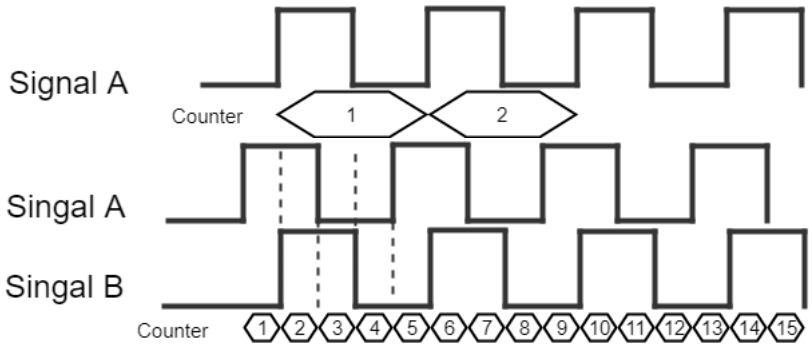
Denna frekvensen av pulser beror på antalet slitsar i skivan och rotationshastigheten. Detta

ger möjlighet att beräkna hur lång tid det tar mellan två pulser för att slutligen beräkna en rotationshastighet [4][8][22].



Figur 2.12: *Illustration av en optisk pulsgivare. Den skiftande rektangeln representerar en del av den roterande skivan fäst på rotorns axel, där vitt representerar slitsar och svart är avståndet mellan två slitsar. Det är två uppsättningar av ljuskällor och ljussensorer för att möjliggöra quadrature encoding.*

För att öka noggrannheten går det att öka antalet pulser genom att använda två set med pulsgeneratorer, alltså två ljuskällor och två ljussensorer som är placerade med ett avstånd från varandra så att de pulser de genererar är 90 grader fasförskjutna från varandra. I figur 2.12 går det att se två uppsättningar med sensorer vilket ger två pulssignaler som går att se i figur 2.13. Med två pulssignaler går det att utföra en beräkning av pulserna som kallas quadrature encoding och ger upp till fyra gånger fler pulser per rotation och en betydligt högre noggrannhet av motorns position. Quadrature encoding används främst för att kunna avgöra vilket håll motorn roterar åt, vilket avgörs beroende på om signal A går hög innan signal B eller viceversa se figur 2.13 [8][4].



Figur 2.13: *Skillnaden mellan att ha en puls gentemot att använda två och kunna utnyttja quadrature encoding. Med en sekvens av ettor och nollor är det möjligt att generera två pulser per slits i skivan och med quadrature går det få så bra som fyra gånger, där varje uppgång eller nedgång av en av signalerna genererar en puls*

Vid användandet av en optisk pulsgivare representerar varje puls en del av varvet. En ekvation kan användas för att beräkna vinkelskillnaden mellan två pulser baserat på antalet pulser per varv samt vilken encodingstyp som används och beräknas enligt

$$\frac{360}{X \cdot P} = grader/puls \tag{2.3}$$

P är slitsar i skivan och X är antalet pulser per slits som till exempel med quadrature encoding är fyra. Figur 2.13 illustrerar de streckade linjerna hur en slits med quadrature encoding ger upp till fyra pulser.

2.5 Simulink

Simulink är ett modelering- och simuleringsverktyg i Matlab skapat av Mathworks[13]. Simulink gör det möjligt skapa mattematiska modeller för både inbyggda system och analoga kretsar. Det består av ett stort blockbibliotek där det är möjligt att designa, simulera, implementera och testa både tidsinvarianta och tidsdiskreta system för till exempel reglering och signalbehandling [13].

2.6 PI-Regulator

För att kontrollera motorns hastighet behövs ett återkopplat system som kan reglera hastigheten med en styrsignal beroende på differensen mellan den önskade hastigheten och motorns hastighet. Den vanligaste formen av regulator kallas proportionell-regulator och kan betecknas enligt

$$u(t) = K_p e(t) \tag{2.4}$$

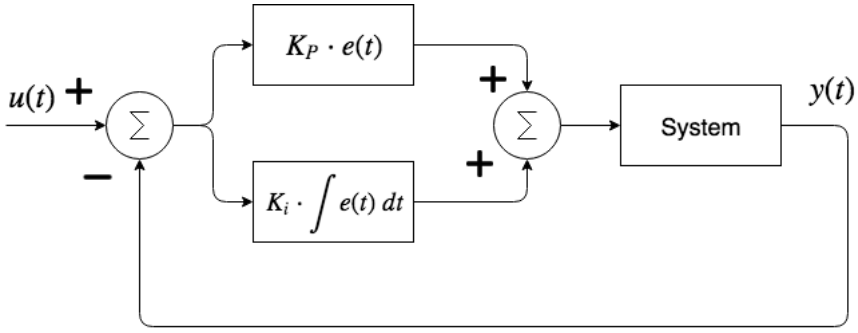
där $e(t)$ är

$$e(t) = u_0 - y(t) \tag{2.5}$$

Där $y(t)$ är systemets utsignal, $u(t)$ den önskade utsignalen från systemet och $e(t)$ är felet eller skillnaden mellan systemets utsignal och den önskade utsignalen. Förstärkningskonstanten K_p bestämmer hur stor proportion av felet som används för reglering. Ett högre värde på K_p ger en snabbare reglering men ökar risken för ett självsvängigt system medans ett lägre värde på K_p ger ett långsammare system. Regulatorn beräknar en styrsignal $u(t)$ som är direkt proportionell mot felet av systemets utsignal, alltså motorns hastighet. P-regulatorn motverkar felet men kan inte eliminera felet helt. I ett stabilt system med P-reglering uppstår vanligen ett stationärt fel. För att motverka detta behöver styrsignalen kontinuerligt öka tills dess att felet närmar sig noll. Detta realiseras genom att lägga till en integrerande del av felet i regulatorn enligt

$$u(t) = K_p e(t) + K_i \int e(t)dt \tag{2.6}$$

Där K_i är förstärkningen för den integrerande delen av regulatorn. En proportionell och integrerande regulator kallas PI-regulator och används ofta till elektriska motorer. I Figur 2.14 kan ett blockschema över en typisk PI-regulator ses. Förstärkningskonstanterna K_p och K_i behöver ställas in. Ofta via en iterativ process för att systemet skall bli stabilt. Om de har för höga värden blir systemet svängigt och instabilt, Vid för låga värden blir systemet långsamt och det stationära felet stort.



Figur 2.14: *Blockschema över en PI-regulator.*

2.6.1 K_p och K_i justering

Att ställa in regulatorförstärkningen till ett system kan vara en lång iterativ process om det utförs för hand. Därför har metoder utvecklats för att snabbare finna förstärkningsvärden för ett önskat beteende av systemet. Zeigler-Nicholsmetoden är en sådan metod som ger en uppskattning av förstärkningsvärden till regulatorn [20]. Metoden brukar dock involvera justering av värdena för hand och är därför ofta en iterativ process.

Zeigler-Nicholsmetoden:

1. Sätt K_i till noll och K_p till ett slumpmässigt värde (exempelvis 1)
2. Öka eller sänk K_p tills dess att systemet precis börjar självsvänga
3. Värdet på K_p benämns nu K_0 och perioden av självsvängningarna T_0
4. Sätt $K_p = 0.5K_0$ och $K_i = \frac{T_0}{1.2}$
5. Om systemet inte har önskat beteende justera K_p och K_i för hand

2.7 Analys

I avsnitt 2.7 jämförs motorernas svagheter och styrkor i syfte att utse en lämplig motor och givare.

2.7.1 Val av motor

Motorvalet grundades på följande punkter.

- Hur och om hastighetsstörningarna går att motverka med olika reglersystem.
- Hur komplicerad styrteknik som krävs för att hastighetsstyra den valda motortypen.
- Hur jämn friktion motorn har under ett varv och hur stort momentripplet är.

Motortyp	**BLDC**	**PMDC**	**Stegmotorn**
Momentrippel	**Lågt**	**Lågt**	**Högt**
Friktion över varvet	**Jämn**	**Jämn fast hög**	**Varierande**
Styrning [kommutering]	**Elektronisk**	**Mekanisk**	**Elektronsik**

Tabell 2.1: *Tabell över de studerade motorernas momentrippel, friktion och deras styrning.*

I tidigt skede räknades stegmotorn bort då den har högt momentrippel och är därför minst lämpad av de studerade motorerna.

Valet bestod då mellan den borstade och den borstlösa likströmsmotorn. Utav dessa valdes den borstade motorn då denna ansågs lättare att styra eftersom den har mekanisk kommutering. Efter närmare studier och kommunikation med handledare på Saab ändrades beslutet. Beslutet ändrades baserat på att den borstlösa likströmsmotorn har mindre friktion över varvet och med sinusoidal kommutering kan momentripplet elimineras. Den borstlösa

motorn som arbetet fortsattes med är av modellen EC-max 22 med märkspänning på 36 volt från företaget Maxon Motors [15]. Detta är en motor som Saab hade på plats och som ansågs lämplig för att kunna uppnå kraven. Denna motor har tre faslindningar i statorn och en permanentmagnet i rotorn med ett polpar.

2.7.2 Val av givare

Valet av givare är baserat på den upplösning som krävs för att kunna reglera hastigheten med en tillräcklig hög noggrannhet för att hastigheten inte skall avvika mer än 0,2 ‰ under en rotation. Utifrån kravet på motorns hastighetsfel per rotation går det att estimera en nedre gräns av upplösningen för en rotation. I uppdraget är det maximala tillåtna hastighetsfelet 0,2 ‰ per rotation vilket resulterar i

$$1 \pm 0.2‰ \tag{2.7}$$

utifrån detta kan en teoretisk lägsta upplösning beräknas till

$$\frac{1}{0,0002} = 5000 \tag{2.8}$$

vilket ger att det krävs minst 5000 pulser under en rotation. Detta är den absolut minsta teoretiska upplösningen som krävs per rotation. I och med detta går det att utesluta många givare och en optisk pulsgivare är den lämpligaste typen av givare.

Den valda optiska pulsgivaren är ENX RIO och har möjlighet att ge upp till 32 768 pulser per rotation med quadrature encoding [14]. Så många pulser per rotation är över det teoretiskt beräknande upplösningen som krävs. Det anses därför mer än tillräcklig för att få en bra representation av signalerna och få en bra regulation.

3
Modell

I avsnitt 3 beskrivs hur en borstlös likströmsmotor kan representeras matematiskt och hur matematiska formler används för att konstruera en modell i programmet Simulink.

3.1 Matematisk modell av BLDC-motorn

En matematisk modell av motorn togs fram och implementerades i Simulink för att göra en preliminär bedömning av motorns lämplighet. Modellen nyttjades för att ge en uppskattning av motorns egenskaper och stegsvar vid hastighetsreglering. Modellen är en förenkling av verkligheten med antaganden att motorns faslindningar är symmetriska och motorn inte har någon hysteres. I figur 3.1 representeras motorns y-kopplade faser och dess impedanser. Det totala momentet kan beräknas utifrån det moment varje fas bidrar med beroende på strömmen. Strömmen beräknas genom att applicera Kirchoffs lag som ger uttrycket

$$V_a - e_a = Ri_a + \frac{di_a}{dt}L \tag{3.1}$$

$$V_b - e_b = Ri_b + \frac{di_b}{dt}L \tag{3.2}$$

$$V_c - e_c = Ri_c + \frac{di_c}{dt}L \tag{3.3}$$

och en omkastning av formlerna ger

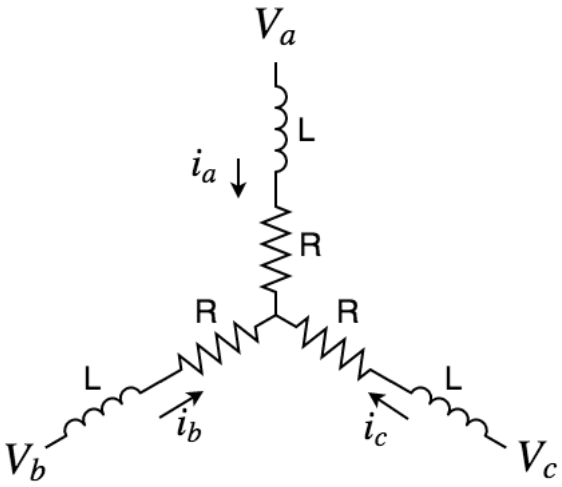
$$i_a = \int \frac{V_a - Ri_a - e_a}{L} dt \tag{3.4}$$

$$i_b = \int \frac{V_b - Ri_b - e_b}{L} dt \tag{3.5}$$

$$i_c = \int \frac{V_c - Ri_c - e_c}{L} dt \tag{3.6}$$

där:

e_x	Inducerad spänning i fas x $[V]$
i_x	Strömmen i fas x $[A]$
V_x	Spänningen i fas x $[V]$
R	Fasresistans $[R]$
L	Fasinduktans $[L]$



Figur 3.1: *Schema över Y-kopplad BLDC-motors faser.*

Den inducerade spänningen e är en motspänning som uppstår vid varje fas när motorns ankare har en rörelse relativt till det inducerade magnetfältet. Motorn agerar då som en generator och en motspänning uppstår. Strömmarna vid sinusoidal kommutation kan uttryckas enligt

$$i_a = A \sin(\omega) \tag{3.7}$$

$$i_b = A \sin(\omega - \frac{2\pi}{3}) \tag{3.8}$$

$$i_c = A\sin(\omega - \frac{4\pi}{3}) \tag{3.9}$$

Den inducerade spänningen kan uttryckas för sinusoidal kommutation som

$$e_a = K_e\omega\sin(\theta) \tag{3.10}$$

$$e_b = K_e\omega\sin(\theta - \frac{2pi}{3}) \tag{3.11}$$

$$e_c = K_e\omega\sin(\theta - \frac{4pi}{3}) \tag{3.12}$$

Det totala momentet motorn producerar är beroende av den tillförda strömmen ges av

$$T_e = \frac{e_a i_a + e_b i_b + e_c i_c}{\omega} \tag{3.13}$$

Från formlerna (3.7)-(3.13) fås då

$$T_e = K_t(i_a\sin(\theta) + i_b\sin(\theta - \frac{2\pi}{3}) + i_c\sin(\theta - \frac{4\pi}{3})) \tag{3.14}$$

där K_t är momentkonstanten som ideallt är lika med K_e. Vilket kan förenklas till

$$T_e = 1,5K_t i_{peak} \tag{3.15}$$

Eftersom summan av strömmarna är lika med noll enligt Kirchhoffs strömlag kan strömmarnas relation till varandra uttryckas

$$i_c = -i_a - i_b \tag{3.16}$$

eller

$$A\sin(\omega t - \frac{4\pi}{3}) = -A\sin(\omega t) - A\sin(\omega t - \frac{2\pi}{3}) \tag{3.17}$$

Det totala producerade momentet kan också beräknas från summan av alla momentförluster i motorn och det utvunna momentet till lasten T_l. Momentförlusterna uppstår från den viskösa friktionen β och rotorns tröghet J vilket ger ekvation

$$T_e = T_l + J\frac{d\omega}{dt} + \beta\omega \tag{3.18}$$

En omskrivning av formeln för att beräkna rotationshastigheten ger

$$\omega = \int \frac{T_e - T_l - \beta\omega}{J} dt \tag{3.19}$$

Den viskösa friktionen beräknades utifrån momentet vid tomgångsgallet T_{nl} enligt

$$\beta = \frac{T_{nl}}{\omega_{nl}} \tag{3.20}$$

där:

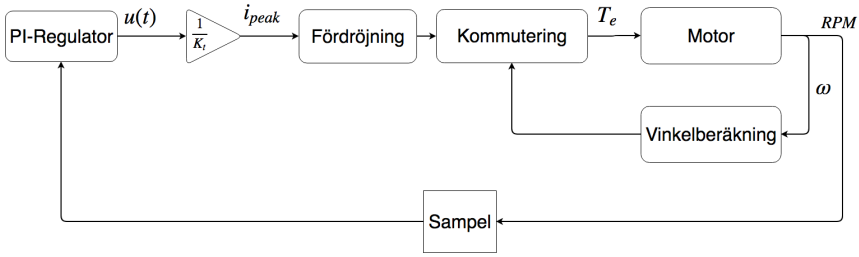
θ	Rotor Vinkel $[rad]$
K_t	Moment-konstanten $[mNm/A]$
K_e	Motemk-konstanten $[rad/V]$
ω	Vinkelfrekvens $[rad/s]$
T_e	Totala momentet $[Nm]$
J	Rotorns tröghet $[gcm^2]$
β	Viskösa friktionskonstanen$[Nm/\omega]$
ω_{nl}	Vinkelfrekvens vid tomgångsfallet $[rad]$
T_{nl}	Moment vid tomgångsfallet $[Nm]$

3.2 Simulink-Modell

En modell skapades i Simulink som följde formler från avsnitt 3.2 och en förenklad representation av den kan ses i Figur 3.2 nedan. Modellen är uppdelad i flera block som beskrivs vidare i följande sektioner. Målet med modellen är att ge en preliminär uppskattning av motorns beteende och att skapa ett reglersystem till motorn som kan uppnå kraven.

I modellen är hastigheten återkopplad till en PI-regulator som reglerar strömmen i_{peak} för en önskad hastighet. Systemet har en fördröjning efter regulatorn som representerar en uppskattning av tiden det tar för processorn att skicka ut en styrsignal från dess att den börjar beräkna styrsignalen. Efter fördröjningen beräknas det totala momentbidraget från strömmen. I nästa block *Motor* representeras motorns hastighet beroende på moment och motorns egenskaper. Hastigheten beräknas i RPM för regulatorn och i vinkelfrekvens för att beräkna vinkeln till sinussignaler. Sampel-blocket tar in värdet av hastigheten och håller

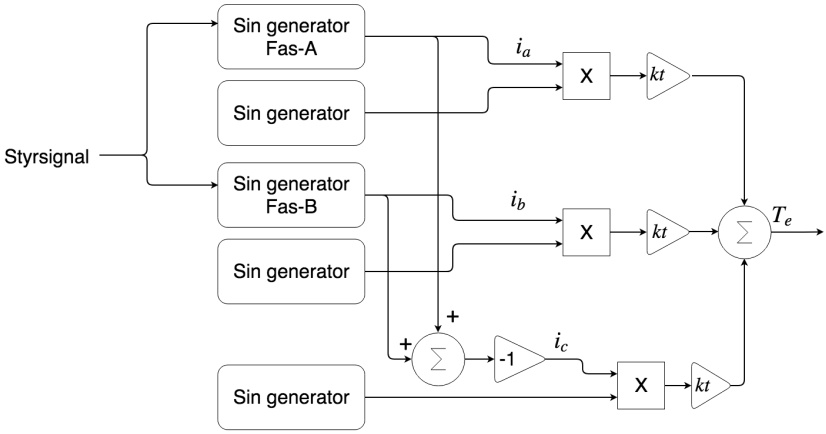
värdet konstant som utsignal under en samplingsperiod och tar sedan ett nytt värde. Detta representerar den frekvens som PI-regulatorn beräknar nya styrsignaler för den verkliga implementationen.



Figur 3.2: *Förenklat blockschema över hur kommutering sker i modellen.*

3.2.1 Kommutering

Kommuteringsblocket beräknar det drivande momentet T_e utifrån formeln (3.14) det vill säga det totala momentet strömmen bidrar med. Blockets innehåll kan ses i Figur 3.3.



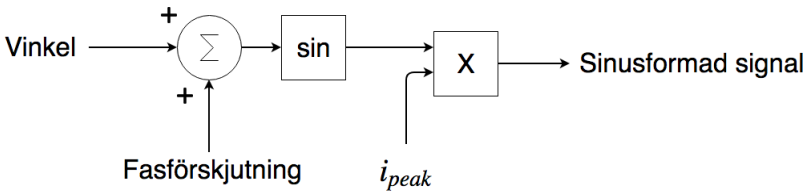
Figur 3.3: *Blockschema över hur T_e beräknas i modellen utifrån i_{peak}.*

Blocket innehåller två sin generatorblock för fas-A och fas-B, tre sin generatorblock som representerar de inducerade spänningarnas faser. Fas-C beräknas från inversen av summan från fas-A och fas-B.

3.2.2 Sin-generering

För att generera en dynamisk sinussignal som skall representera bland annat strömmen skapades ett block som tar in vinkel, fasförskjutning och amplitud. I figur 3.4 kan blockets innehåll för strömgenereringen ses där vinkeln beräknas kontinuerligt beroende av hastigheten som resulterar i sinusignalens vinkelhastighet. För fas-b och fas-c adderas en

fasförskjutning till vinkeln innan den går till sinusfunktionen. Styrsignalen (i_{peak}) kommer ifrån PI-regulatorn och bestämmer amplituden av sinussignalen.



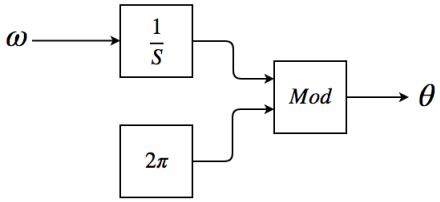
Figur 3.4: *Blockschema över hur sinussignaler genereras i modellen.*

3.2.3 Beräkning av θ

Eftersom som strömmens periodtid är densamma som rotorns varvtid beräknas periodtiden som ett resultat av motorns hastighet. I modellen integreras vinkelhastigheten som ger den totala vinkeln och utifrån den kan en vinkel fås från resten av den totala vinkeln och 2π enligt

$$\theta = \int \omega \, dt \; mod \; 2\pi \tag{3.21}$$

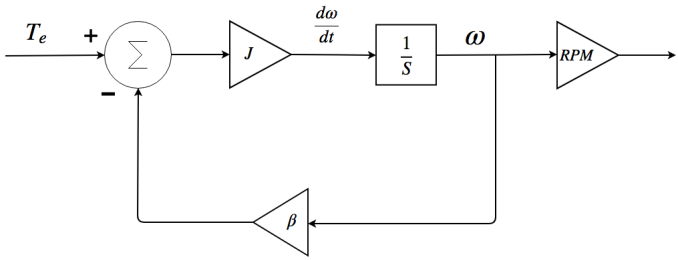
I Figur 3.5 kan beräkningen av θ i Simulink ses.



Figur 3.5: *Blockschema över hur vinkeln beräknas till blocket sinus-generering.*

3.2.4 Motor

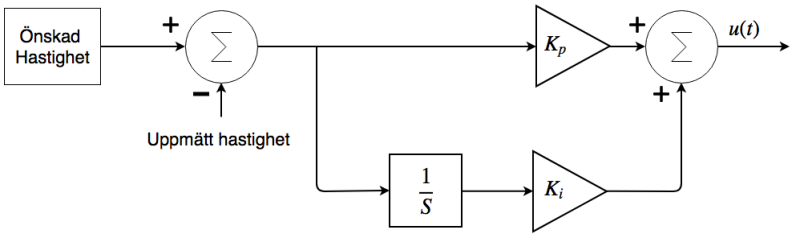
Motorns hastighet beroende av momentet modelleras enligt formel (3.19) och kan ses i Figur 3.6. Eftersom lasten enbart består av ett tröghetsmoment räknades den med i rotorns tröghet. Blocket RPM i figuren räknar om vinkelfrekvensen till varv per minut för en tydligare representation av hastigheten.



Figur 3.6: *Blockschema över hur motorn modelleras i Simulink.*

3.2.5 PI-regulator

Blocket för PI-regulatorn konstruerades efter formel (2.6) och dess innehåll kan ses i figur 3.7. Blocket beräknar differensen av den önskade hastigheten och hastigheten som kommer från motor-blocket. Sedan summeras en proportionell förstärkning av felet och en integrerad förstärkning av felet som resulterar i en styrsignal.



Figur 3.7: *Blockschema över hur PI-regulator modelleras i Simulink.*

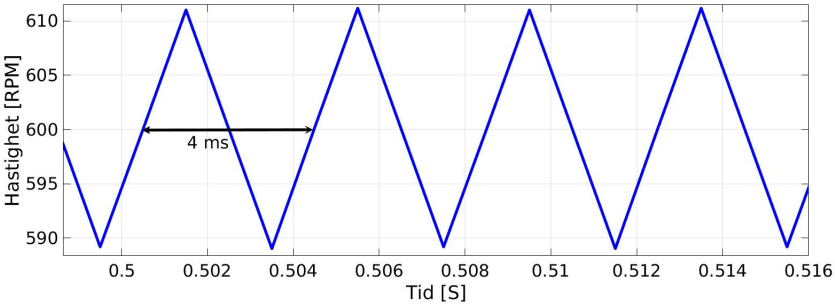
4

Simulering

I detta avsnitt presenteras simuleringar av modellen med syfte för en preliminär utvärdering av motorn innan systemet implementerades i en testbänk.

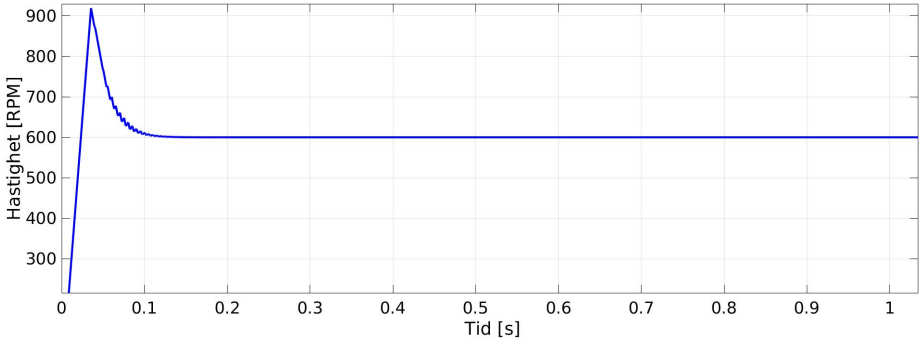
4.1 Regulatorförstärkning

För att finna värden till K_p och K_i användes Zeigler-Nicholsmetoden som är beskriven i avsnitt 2.6.1. Den proportionella förstärkningen K_p sattes till 0,1 och K_i till noll vilket resulterade i ett instabilt system. Sedan justerades den K_p iterativt tills dess att systemet började självsvänga vilket var vid 0,0018 och K_0 sattes därför till 0,0018. I Figur 4.1 kan självsvängningarna ses som har en amplitud på 10 RPM och en periodtid t_0 på 4 ms. Utifrån Zeigler-Nicholsmetoden sattes därför K_p och K_i till 0,0009 och 0,0033. Detta resulterade i ett stabilt men för långsamt system som inte klarade av kravet på att vara uppe i rätt hastighet efter 0,5 sekund. Den integrerande förstärkningen K_i justerades därför för hand tills att ett önskat resultat uppnåddes.



Figur 4.1: *Resultat från modellens simulering då* $K_p = 0,0018$ *och* $K_i = 0$*. Ur figuren kan självsvängningarnas periodtid på 4 ms ses.*

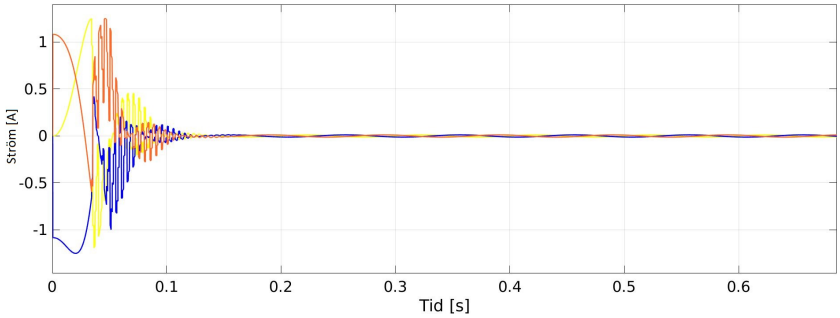
Den integrerande förstärkningen K_i justerades till ett värde på 0,033 som resulterade i stegsvaret som kan ses i Figur 4.2. Vid 0,17 sekunder uppnåde systemet hastighetskravet och vid 0,5 sekunder var hastighetsfelet så litet som $4 \cdot 10^{-10}$ RPM vilket är väl inom kravet.



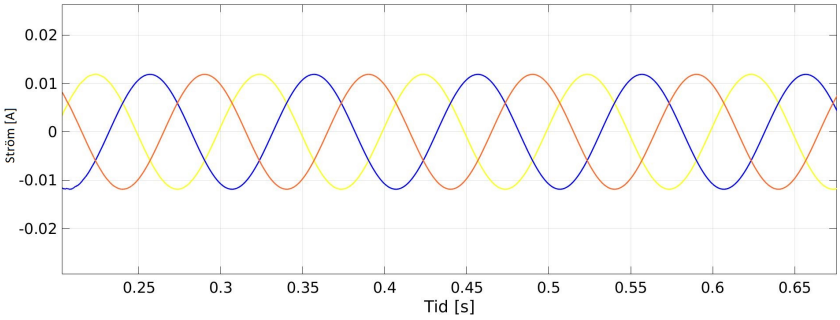
Figur 4.2: *Resultat från modellens simulering då* $K_p = 0,0090$ *och* $K_i = 0,033$

4.2 Moment och Strömmar

Resultaten från simuleringarna studerades genemot formlerna under stycke 3.2 för att se att modellen är korrekt uppbyggd. Nedan i Figur 4.3 och Figur 4.4 ses fasströmmarna från en simulering där de har stabiliserat sig vid 0.5 sekunder. Det går även se att de är korrekt fasförskjuta med 120° genemot varandra eftersom avståndet mellan varje kurva är lika stort. Detta bekräftar att strömmens faser är korrekt modellerade.

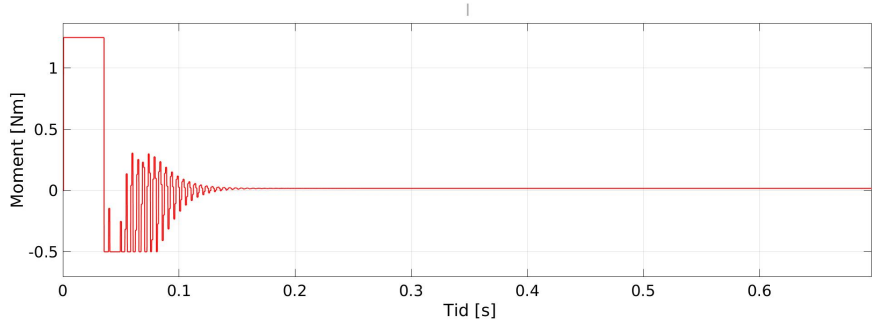


Figur 4.3: *Simulering av de tre strömfaserna i modellen.*



Figur 4.4: *Strömfaserna i simuleringen efter att de har stabiliserat sig.*

I Figur 4.5 kan det beräknande momentet från simuleringen ses. Det går att se hur momentet blir stabilt utan variationer efter 0,5 sekund som önskat. Momentet beräknades enligt ekvation (3.14).



Figur 4.5: *Det drivande momentet T_e efter systemet stabiliserat sig och konstant hastighet uppnås, där det går att se att momentet blir stabilt.*

4.3 Sammanfattning

Resultatet från simuleringarna uppnådde de krav som är ställda på motorn. När motorns hastighet hade stabiliserat sig uppförde strömmarna och momentet som förväntat enligt formlerna. Ur simuleringarna kan det fastslås att en PI-regulator och sinusoidal kommutering kan användas för att styra en borstlös likströmsmotor. De framtagna reglervärdena från simuleringen kommer användas i testbänken.

5
Testbänk

I detta kapitel beskrivs hur den praktiska implementationen är uppbyggd och hur den används för att nå de ställda kraven. Först beskrivs de ingående komponenterna och hur de används. Programmeringen av processorn beskrivs stegvis och kapitlet avslutas med resultatet för de tester som utfördes. Målet med testbänken är att få motorn med styrteknik att uppnå en konstant hastighet med en maximal avvikelse på 0,2 ‰ per varv och inom en halv sekund efter start.

5.1 Implementation av styrsystem

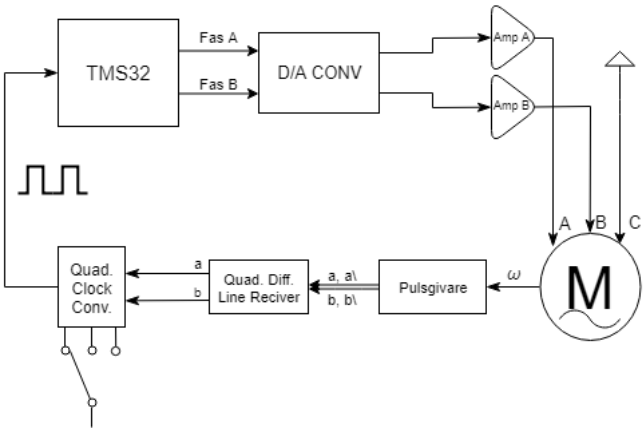
I avsnitt 5.1 beskrivs de väsentligaste komponenterna som använts och vad de använts för, se Figur 5.1 för blockschema över implementationen.

Processorn TMS320C30

Processorn i implementationen är av modellen TMS320C30 med 32 bitars register. Till denna processor finns en emulatormiljö som gör det möjligt att till exempel stega igenom kod eller ta ut data från minnet efter användning.

Quadrature clock converter/LS7184

LS7184 är en komponent som tar in signalerna A och B från pulsgivaren och konverterar signalerna till en signal. Med komponenten är det möjligt att välja upplösning per varv, mellan 1, 2 och 4 multiplicerat med antalet slitsar per rotation. Exempelvis väljar man 2 så ger LS7184 ut $2 \cdot 8192$ pulser per rotation. Vid inställning på 4 använder den quadrature encoding och får upp till fyra gånger fler pulser per rotation [11].



Figur 5.1: *Blockschema över den praktiska implementationen. Återkopplingen sker i form av pulser till processorn TMS32. Längst ner till vänster ses det hur de är möjligt att välja antalet pulser per rotation i återkopplingen med komponenten Quadrature Clock Converter.*

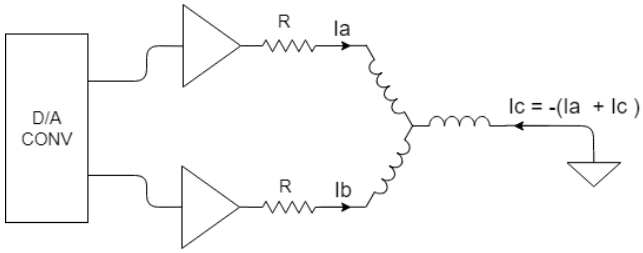
Quadruple Differential Line Reciver/LS33A-SP

Differential line receiver används ofta för att eliminera störningar som kan uppkomma på pulsgivarens signaler. I detta fall används den då pulsgivaren ger ut differentiella signaler.

Komponenten tar in signal A och B från pulsgivaren samt dess inventerade signaler A-invers och B-invers. Den skickar ut signal A-ut och B-ut som är differensen mellan de inkomna signalerna och deras inverser. På ut-signalerna har störningarna från A och B eliminerats och de kan användas riskfritt i resterande komponenter [21]. De uppkomna störningarna är identiska för både signal A och A-invers vilket gör det möjligt att eliminera störningar genom att att låta utsignalen vara differensen mellan A och A-invers.

Förstärkare LM675

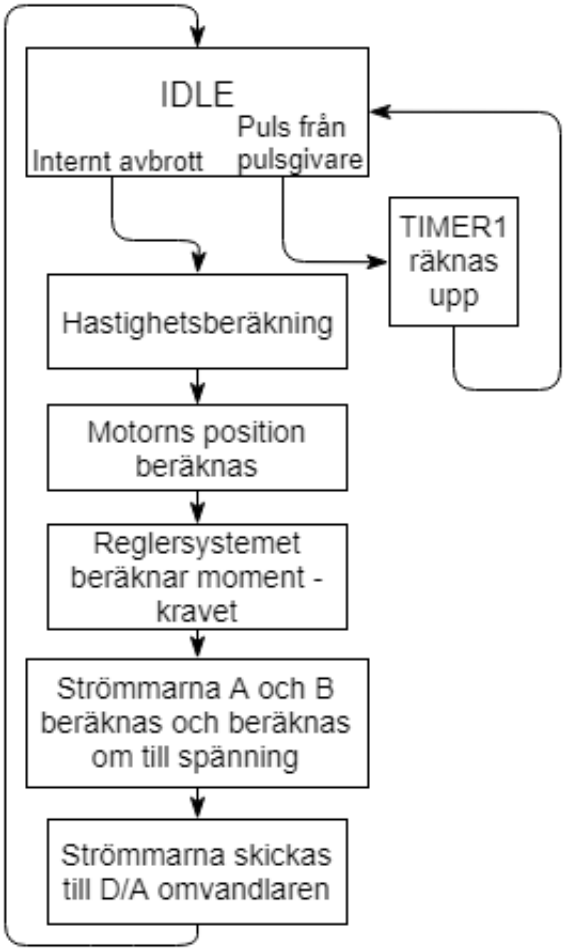
LM675 är en operationsförstärkare som kan leverera upp till tre ampere. Två stycken sådana användes för att driva ström till motorn. I Figur 5.2 kan ett förenklat schema över förstärkarnas koppling ses. De beräknade strömmarna omvandlas till analoga signaler i D/A-omvandlaren och går vidare till två förstärkare. Fas-A och fas-B går igenom varsin förstärkare medan fas-c är kopplad till fas-A's och fas-B's gemensamma signaljord vilket ger en korrekt ström till fas-C enligt formeln 3.13.



Figur 5.2: *Kopplingschema över förstärkarsteget. De tre induktanserna representerar lindningarna i motorn och hur de skapas. Resistorn R finns för att kunna beräkna korrekt spänning i processorn.*

5.2 Implementering i C

Detta stycke beskriver hur implementationen i C fungerar. Vilka avbrottsrutiner som används och varför. Hur beräkningar av av fasens position sker, strömmar, hastighetsberäkningar samt hur regulatorn används och implementerades. Figuren 5.3 visar flödesschema över implementationen.



Figur 5.3: *Flödesschema över programmet som används för styrning av motorn.*

Tabell 5.1: *Tabell över de variabler som används i mjukvaran till processorn för att utföra beräkningar.*

$SYST\_FREQ$	Antal interna avbrott per sekund
$TIMER1$	Räknar antalet pulser för pågående rotation
ENX_COUNT	Totala antalet pulser över en rotation
POS_INCR	Radianer per puls
fas_pos	Rotorns vinkel $[rad]$
$tmr1$	Antal pulser vid det pågående systemavbrott
$tmr2$	Antal pulser vid det föregående systemavbrottet (n-1)
$delta\_tmr$	Antal pulser mellan två systemavbrott
$MY2\_PI\_3$	$2\pi/3$
$rotor\_speed$	Hastigheten på rotorn $[rad/s]$
K_P	Proportionell försträknings konstanten
K_I	Integrerande försträknings konstanten
K_t	Motorns moment konstant [Nm/A]
$ROTOR\_SPEED\_SETPOINT$	Den begärda hastigheten på motorn $[rad/s]$
Tq_dem	Det begärda momentet beräknat med regulatorn $[Nm]$
$ipeak$	Amplituden på strömmen $[A]$
$curr\_ph\_a$	Strömmen för fas a $[A]$
$curr\_ph\_b$	Strömmen för fas b $[A]$
SC_IPHASE_UPH	Konstant för att få korrekt spänning till D/A

5.2.1 Avbrottsrutiner

Implementationen i C drivs av en intern avbrottsrutin, som är tidsbaserad. För att beräkna pulser från pulsgivaren används en dedikerad hårdvara i processorn.

Den dedikerade hårdvaran används för att räkna upp variabeln $TIMER1$ ett steg, detta sker för att veta hur långt den pågående rotationen kommit. $TIMER1$ har ett förinställt maximum till antalet pulser per varv, vilket sätts med konstanten ENC_COUNT. När $TIMER1$ överskrider maximum har en rotation skett och variabeln nollställs.

Det interna avbrottet sker periodvis. Avbrotten sker med en frekvens satt av $SYSTEM\_FREQ$ som kan sättas till valfri frekvens beroende på hur ofta beräkningar behöver ske. Vid varje internt avbrott sker beräkningar av hastighet, strömmarnas amplitud och fas. Väsentlig data såsom hastigheten, de beräknade strömmarna och det krävda momentet loggas för att kunna läsas av när den pågående körningen har slutförts.

5.2.2 Logger-programmet

Logger-programmet är ett program som SAAB framtagit för att kunna läsa av data i TMS320C30:s minne vid körning med dess emulator. Data sparas som flyttal i processorns minne. Lagringsminnet är på totalt 128K*4 bytes. Det är valbart hur många olika variabler som önskas lagra. Med fler variabler minskar antalet loggningar som kan sparas per variabel, då minnet är begränsat.

En logg kan läsas av från minnet och sparas som en objekt-fil när ett test är slutfört. Loggen översätts sedan via ett program på datorn till en data-fil som kan läsas in i matlab för slutgiltiga beräkningar.

5.2.3 Beräkning av radianer per puls

För att använda sinusoidal kommutering behöver vinkeln till strömmarna beräknas, för detta måste avståndet mellan två pulser räknat i radianer vara känt, och ekvation (2.3) används för att beräkna följande

$$POS\_INCR = \frac{2\pi}{(X \cdot P)} \tag{5.1}$$

där är P antalet slitsar i pulsgivaren per rotation och X är en faktor på 1, 2 eller 4 beroende på vilken upplösning som är satt i LS7184. Vid lägst upplösning är avståndet cirka $7,66 \cdot 10^{-4}$ radianer per puls och vid högst är den $1,91 \cdot 10^{-4}$ radianer per puls.

5.2.4 Hastighetsberäkning

Motorns hastighet beräknas vid varje systemavbrott. I början av systemavbrottet sparas först det nuvarande antalet pulser $TIMER1$ från den pågående rotationen i en temporär variabel $tmr1$. Förra avbrottets puls antal finns sparat i variabeln $tmr2$. Differensen av dessa ger antalet pulser som gått mellan de två senaste systemavbrotten enligt

$$delta\_tmr = tmr1 - tmr2 \tag{5.2}$$

Efter att motorn roterat ett varv blir $tmr1$ mindre än $tmr2$ eftersom ENC_COUNT nollställs och detta inträffar en gång per rotation och medför att $delta\_tmr$ blir negativ. Om $delta\_tmr$ är negativ kommer det resultera i en negativ vinkelhastighet efter slutberäkningen. För att undvika detta används följande if-sats

if ($delta\_tmr < 0$)

$$delta\_tmr = delta\_tmr + ENC\_COUNT \tag{5.3}$$

If-satsen åtgärdar så att korrekt antal pulser beräknas på ett varv och att hastigheten inte blir negativ när dessa fall uppstår. Med antalet pulser mellan två avbrott kan vinkelhastigheten beräknas med POS_INCR radianer per puls och antalet systemavbrott per sekund ($SYST\_FREQ$) genom

$$rotor\_speed = delta\_tmr \cdot POS\_INCR \cdot SYST\_FREQ \tag{5.4}$$

5.2.5 Beräkning av fasen

Fasen ges av produkten från POS_INCR som beräknades i ekvation (5.1) och antalet räknade pulser från det pågående varvet som fås från $TIMER1$ och ger ekvationen

$$fas\_pos = TIMER1 \cdot POS\_INCR \tag{5.5}$$

Denna beräkning sker vid varje systemavbrott för att strömmarna ska få korrekt fas.

5.2.6 Implementation av reglersystem

För att reglera motorns hastighet används samma reglerteknik som i Simulink- modellen, en PI-regulator. Reglerfelet beräknas med $rotor\_speed$ och $rotor\_speed\_setpoint$ där $rotor\_speed$ subraheras från $rotor\_speed\_setpoint$ enligt

$$rotor\_speed\_err = rotor\_speed\_setpoint - rotor\_speed \tag{5.6}$$

Reglerfelet används för att PI-regulatorn ska kunna beräkna det begärda momentet Tq_dem, ekvationerna (2.4), (2.5) och (2.6) används för att beräkna Tq_dem. PI-regulatorn implementerades enligt

$$rotor\_speed\_err\_I = rotor\_speed\_err\_I + \frac{rotor\_speed\_err}{SYST\_FREQ} \tag{5.7}$$

$$Tq\_dem = K\_i \cdot rotor\_err\_I + K\_p \cdot rotor\_speed\_err \tag{5.8}$$

5.2.7 Beräkning av strömmen till faserna

Det begärda momentet räknas om till strömmarnas amplitud för att uppnå detta moment. Strömmarnas amplitud beräknas med inversen av momentkonstanten K_t. Momentkonstanten multipliceras med det beräknande momentet enligt

$$ipeak = \frac{Tq\_dem}{K_t} \tag{5.9}$$

Efter att amplituden på strömmen är bestämd, beräknas sedan faserna av strömmarna. Vinkeln fås från tidigare beräkning (5.5) och en fasförkjutning på $\frac{\pi}{2}$ läggs till för att strömmarna skall få korrekt fas i relation med rotorn. Strömmarna a och b beräknas enligt

$$curr\_ph\_a = ipeak \cdot sin(fas\_pos + \frac{\pi}{2}) \tag{5.10}$$

$$curr\_ph\_b = ipeak \cdot sin(fas\_pos - MY2\_PI\_3 + \frac{\pi}{2}) \tag{5.11}$$

Fas a och b beräknas om till spänning för att sedan översättas från digital form till analog form via en D/A-omvandlare. Strömmarna konverteras om med konstanten SC_IPHASE_UPH vars värde kommer från resistansen som sitter efter förstärkaren i kretsen som kan ses i Figur 5.2.

5.2.8 Hastighetsutvärdering

För att utvärdera systemet har en extra pulsgivare och en skiva med tolv slitsar på rotorn använts. Vid varje rotation kommer alltså tolv pulser ges ut från givaren. Denna pulsgivare är kopplad till en annan processor än den som styr motorn. Processorns uppgift är att logga antalet klockpulser som skett mellan två slitsar på skivan.

Denna processor använder endast externa avbrott som kommer från pulsgivaren. Vid dessa avbrott sker beräkningar av hur lång tid som passerat i form av klockpulser. Den passerade tiden loggas för att kunna läsas av. Pulsgivaren kan max ge tolv pulser per rotation och beräkningsfrekvensen beror på motorns hastighet då frekvensen av avbrott beror på hastigheten.

Tabell 5.2: *Tabell över de variabler som används i mjukvaran till processorn som utvärderar motorns hastighet samt variabler som användes för de avslutande beräkningarna i Matlab.*

$TIMER2$	Räknar upp varje klockpuls i processorn
$temp\_timer$	Array med föregående klockpulser vid varje puls på ett varv
$temp1$	Lagrar antalet klockpulser vid start av avbrottet
$pulses$	Vilken puls rotorn befinner sig vid på pågående rotation
$clock\_timer$	Differensen mellan pulses(n) och pulses(n-12) i klockpulser
$timer\_per\_rot$	Tid för en rotation(Matlab) [s/varv]
rot_per_sec	Antal rotationer under en sekund(Matlab) [varv/s]

För att få extra noggrannhet beräknas hastigheten tolv gånger på ett varv. När ett avbrott sker sparas antalet klockpulser $TIMER2$ i variabeln $temp1$, därefter räknas $pulses$ upp och om de blir högre än tolv nollställs den.

Antalet klockpulser för en rotation beräknas

$$clock\_timer = temp1 - temp\_timer(pulses) \tag{5.12}$$

$$temp1 = temp\_timer(pulses) \tag{5.13}$$

Ekvationen beräknar differensen av tiden då en puls från givaren kommer och tiden då samma puls återkommer. Värdet på $temp1$ sparas i arrayen $temp\_timer(pulses)$ tills nästa gång pulsen återkommer nästa rotation. Genom att $temp1$ sparas vid varje puls under ett varv kan beräkningen av hastigheten ske tolv gånger per rotation. Antalet beräknade pulser $clock\_timer$ loggas i slutet av avbrottet för att kunna läsas av när testet är slutfört.

Slutliga beräknignar i Matlab

De loggade värderna för $clock\_timer$ läses av med hjälp av loggerprogrammet och beräkningar kan slutföras i Matlab. För att beräkna hastigheten läses de loggade värdena av från

processorn och beräknas om till hastighet. Processorns klockfrekvens är 8MHz och med de loggade klockpulserna för varje rotation kan hastigheten beräknas med

$$time\_per\_rot = \frac{clock\_timer}{8 \cdot 10^6} \tag{5.14}$$

med $time\_per\_rot$ är det möjligt att beräkna hastigheten per sekund. Detta görs genom beräkna kvoten av ett och $time\_per\_rot$ enligt

$$rot\_per\_sec = \frac{1}{time\_per\_rot} \tag{5.15}$$

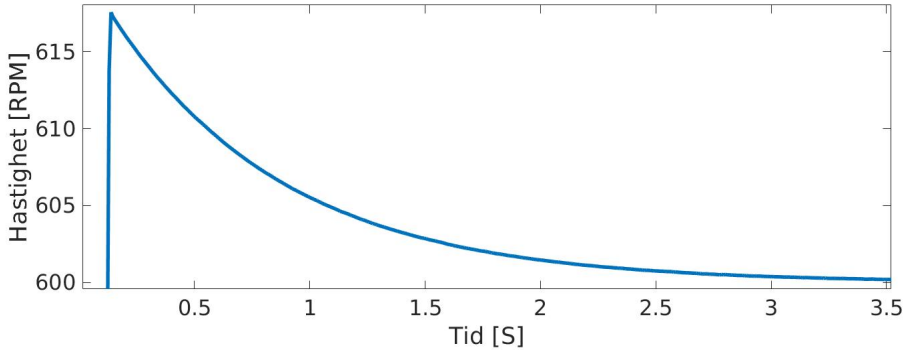
med denna beräkning är hastighetskontrollen färdig då antalet rotationer per sekund enkelt kan beräknas om till vinkelhastighet eller varv per minut.

5.3 Resultat

I detta avsnitt presenteras resultatet från olika tester. Tester gjordes för att finna optimala inställningar till styrtekniken för ett så litet hastighetsfel som möjligt. Testerna innefattar inställningar av reglervärden, beräkningsfrekvens och pulsgivarens upplösning.

5.3.1 Regulator Justering

Regulatorförstärkningen i testbänken sattes först till de värden som var framtagna från simuleringen av modellen. Resultatet är ett stabilt system som kan hålla en hastighet runt 600 RPM men ett litet fel. Dock så får systemet en översläng vid uppstart och går ner till 600 RPM för långsamt som kan ses i Figur 5.4. Vid 0,5 sekund ligger motorns hastighet på cirka 611 RPM vilket inte uppnår kraven.



Figur 5.4: *Motorns stegsvar då regulatorförstärkningen var satt till simuleringens bästa reglerförstärkning.*

Eftersom regulatorförstärkningen inte gav ett önskat resultat användes Ziegler-Nicholsmetoden för att ta fram nya värden. Den proportionella förstärkningen K_p sattes först till 0,1 och K_i till noll. Detta gav ett instabilt system och K_p sänktes tills att systemet blev stabilt.

Gränsen mellan ett instabilt och ett stabilt system låg någonstans mellan 0,018 och 0,017 därför sattes K_0 till 0,018.

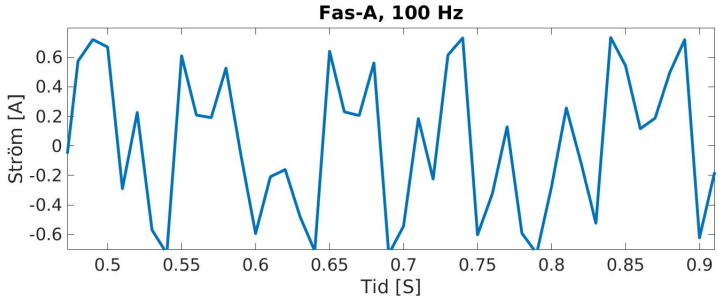
Periodtiden T_0 av självsvängningarna låg på cirka 0,05 sekunder. Regulatorförstärkningarna K_p och K_i sattes därför till 0,009 och 0,042. Systemet blev dock fortfarande för långsamt och K_i ökades tills att kraven på systemet blev uppnådda. Efter det justerades både K_i och K_p något förhand för att minska det stationära felet ytterligare. Resulterade K_p och K_i blev 0,0092 och 0,35 som kan ses i Tabell 5.1. Med dessa värden nådde motorn kravet på hastighetsfelet efter 0.25 sekunder. Det största uppmätta felet efter 0,5 sekund låg på cirka 0,034 ‰ vilket är runt en faktor på 6 mindre än kravet.

Tabell 5.3: *Tester för olika reglervärden.*

K_p	0.009	0.009	0.0092
K_i	0.012	0.042	0.35
Resultat	För långsamt	För långsamt	Inom kraven

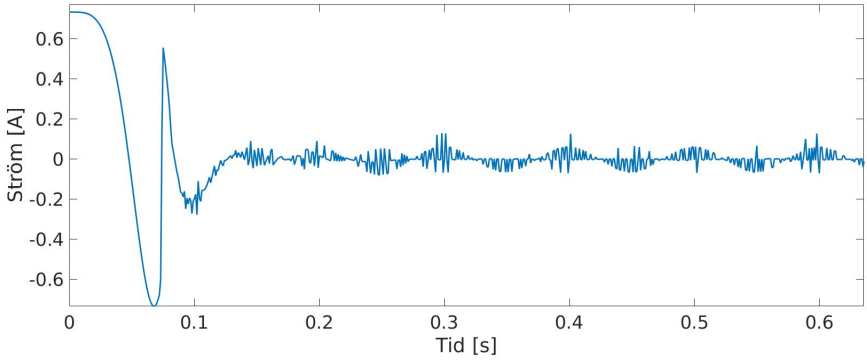
5.3.2 Test av beräkningsfrekvens

I avsnitt 2.3.2 diskuterades frekvensen av beräkningar som behövs för att konstruera en sinusformad ström till motorn beroende på motorns hastighet. Här presenteras resultatet från tester av beräkningsfrekvenser mellan 100 och 4000 Hz. Samtliga tester hade samma K_p- och K_i-förstärkning på 0,0092 och 0,35 samt upplösning inställd på 32768 pulser per varv. I Figur 5.5 kan den beräknade strömmen till fas-A ses vid en beräkningsfrekvens på 100 Hz. En antydan av en sinusformad ström med en period på 0,1 sekunder kan urskiljas men det resulterade hastighetsfelet som kan ses i Tabell 5.4 uppnår inte kraven.



Figur 5.5: *Den beräknade strömmen genom fas-A vid beräkningsfrekvensen 100 Hz.*

I Figur 5.6 kan den beräknade strömmen till fas-A ses vid en beräkningsfrekvens på 1000 Hz. Resultatet är en mer sinusformad ström än vid beräkningsfrekvensen 100 Hz.



Figur 5.6: *Den beräknade strömmen till fas-A vid en beräkningsfrekvens på 1000 Hz*

I Tabell 5.4 kan det ses vilka beräkningsfrekvenser som resulterat i ett hastighetsfel inom kraven. Både för låga och för höga beräkningsfrekvenser resulterar i ett hastighetsfel som ligger utanför kraven. Optimalt resultat gavs vid en beräkningsfrekvens på 2000 Hz.

Tabell 5.4: *Tabell över resultat vid olika beräkningsfrekvenser. Testerna utfördes med samma K_p och K_i värde samt upplösning.*

Beräkningsfrekvens [Hz]	**100**	**200**	**500**	**1000**	**1500**	**2000**	**4000**
Största fel [RPM]	45,88	8,87	0,08	0,03	0,06	0,02	6,88
Största fel [‰]	76,472	14,776	0,122	0,039	0,095	0,027	11,471

Hastighetsfelet för beräkningsfrekvenserna 500, 1000, 1500 och 2000 Hz låg under 0,2 ‰ över en rotation och därför gjordes fem ytterligare tester för varje beräkningsfrekvens. I Tabell 5.5 framgår det största hastighetsfel motorn hade över ett varv för respektive beräkningsfrekvens.

Tabell 5.5: *Tabell innehållandes det största hastighetesfel respektive beräkningsfrekvens gav. Fem tester gjordes för varje beräkningsfrekvens.*

Beräkningsfrekvens [Hz]	500	1000	1500	2000
Största fel [‰]	0.1349	0.0424	0.0950	0.0275

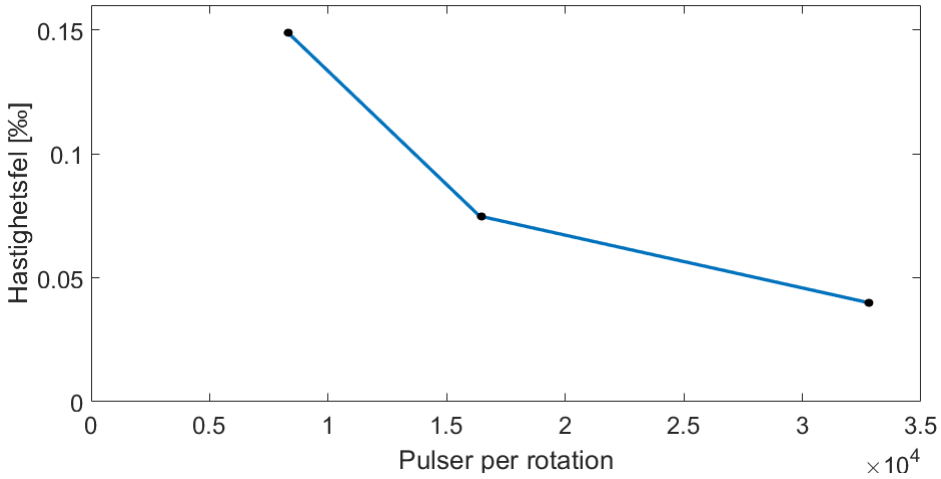
5.3.3 Upplösning

Testerna för olika upplösningar gav resultat som ses i Tabell 5.6. Samtliga tester hade samma K_p och K_i förstärkning på 0,0092 och 0,35. Medelhastigheten skiljer sig ytterst lite mellan de olika upplösningarna men är dock av mindre intresse. Mer intressant är hur den högsta och lägsta uppmätta hastigheten över ett varv skiljer sig.

Tabell 5.6: *Tabell över resultat vid olika upplösningar. Testerna utfördes med samma K_p och K_i värde samt beräkningsfrekvens.*

Upplösning [pulser per varv]	**32768**	**16384**	**8192**
Största fel [RPM]	599,99	599,99	599,99
Största fel [‰]	0,0400	0,0749	0,1499

I Tabellen 5.6 kan en antydan till linjärt samband tolkas mellan det största felet över ett varv och upplösningen. I Figur 5.7 visas en graf med hastighetsfel mot antalet pulser per rotation. Skillnaden mellan de hösta och lägsta hastigheternas fel ligger nära en faktor på två av skillnaden mellan upplösningarna. För detta arbete ligger samtliga resultat inom kravet på felet med en acceptabel marginal. Samtliga tester nådde även den önskade hastigheten inom 0,5 sekunder.



Figur 5.7: *Grafen visar de största hastighetsfelen vid tester av olika upplösningar. Det går att se en antydan till ett linjärt samband mellan felet och upplösningen.*

6

Utvärdering

6.1 Slutsats

I den här rapporten har olika typer av motorer, givare och tillhörande styrteknik studerats för att välja en lämplig motor till en av Saabs produkter. Resultatet blev en borstlös likströmsmotor. En mattematisk modell gjordes av motorn i Matlabs Simulink för simulering och tidig utvärdering av motorn. Simuleringarna av modellen uppnåde det uppsatta kravet på motorns hastighetsfel och motorn testades därefter i en testbänk. Efter regulatorjustering, beräkningsfrekvens- och upplösningstester uppnåddes kraven på motorn även i testbänken.

Valet av motor baserades främst på möjligheten att eliminera hastighetsvariationer under drift. Momentrippel uppstår främst vid kommutering av motorn och ojämn friktion vilket orsakar hastighetsvariationer. Valet av motor blev därför en borstlös likströmsmotor eftersom dess interna friktion är jämn och med sinusoidal kommutation kan momentripplet teoretiskt elimineras.

Sinusoidal kommuteringen krävde också att beräkningar av strömmarnas faser sker med hög frekvens för att motorn skall rotera utan några större variationer. Utifrån de testerna av beräkningsfrekvenserna som utfördes fanns både en övre och en undre gräns där kraven inte uppnåddes. Den undre gränsen ligger mellan 200-500 Hz eftersom hastighetsvariationer uppstår vid låga beräkningsfrekvenser då strömmarna inte får den sinusform som krävs för att eliminera momentripplet. Den övre gränsen ligger mellan 2000-4000 Hz eftersom vid högre beräkningsfrekvenser ökar felet på den beräknade rotorns vinkel vilket leder till att hastighetsberäkningen till regulatorn får ett större fel och i sin tur även då styrsignalen.

Den optiska givaren valdes baserat på att sinusoidal kommutering skulle användas då det krävs återkoppling med hög upplösning för denna typ av kommutering. Motorn i samband med denna givare gav resultat högt över kraven. Vid test med en beräkningsfrekvens på

2000 Hz och högsta upplösningen av ett varv på 32 768 pulser per rotation uppnåddes det minsta hastighetsfelet på 0,027 ‰. Resultatet i det testet blev omkring 7,4 gånger bättre än kravet på 0,2 ‰.

Kombinationen av motorn, givaren och kommuteringssättet gav ett önskat resultat över förväntan. Om ett bättre resultat önskats hade ytterligare pulser per rotation varit till fördel. Där till exempel 65 536 pulser per rotation hade resulterat i om kring 0,0135 ‰ enligt våra tester. Med högre upplösning av varvet hade även en högre beräkningsfrekvens varit möjlig för att förbättra resultatet.

6.2 Etisk Diskussion

I dagens samhälle har den miljöpolitiska debatten fått ett ökat tryck då medvetandet om miljöproblemen orsakade av människan har ökat de senaste decennierna. Styrande organ både på nationella och globala nivåer har under en tid styrt industrier och organisationer mot miljövänligare verksamheter. Idealt önskas att en hållbar utveckling uppnås där ekonomiska, sociala och ekologiska behov tillgodoses både i dagsläget och i framtiden. För de ekologiska behoven behövs ett kretsloppssamhälle där naturresurser ska återanvändas, återvinnas och slutligen omhändertas på bästa möjliga sätt. Därför är det viktigt att produkter optimeras genom att använda naturresurer som i bästa fall kan återanvändas och minimera mängden resurser som behövs, speciellt av lagerresurser som exempelvis fossila bränslen och sällsynta metaller.

Inom motorer är olika metallers magnetiska egenskaper nyckel till ett starkt moment och det finns metaller med starkare magnetiska fält än andra. En av dessa metaller är neodym som har högre fältstyrka än många andra metaller. Motorer där neodym eller liknande metall används som legering blir den fysiska storleken på motorn mindre i relation till de motorer där dessa metaller inte används. Metaller med större fältstyrka är därför väldigt eftertraktade vid konstruktion av motorer, för att kunna minska motorers storlek eller öka dess effektivitet [3].

Neodym anses som en sällsynt metall dock så är det inte en naturresurs som det finns brist på, den är snarare svår att utvinna och kommer i små mängder. Att utvinna neodym kan därför ses som ett positivt bidrag till nyttan eftersom det finns i större lager än ovanliga metaller. Sysselsättningen ökar då det är svårutvunnet vilket innebär att det behövs mer personal, alltså drar både den sociala och ekologiska hållbarheten fördel av neodym. Men idag har Kina ett monopol på utvinning av neodym samt många andra sällsynta jordartsmetaller som utvinnas främst i staden Baotou [19][10][2]. Batou har fått skarp kritik angående deras utvinningsprocess då den både påverkar närliggande miljö och samhälle negativt. I utvinningen fås radioaktiva ämnen som biprodukt. Därför kan utvinningen av neodym ses som negativ effekt på nyttan av arbetet mot hållbar utveckling.

Det är alltså viktigt vid design av en produkt att veta vilken typ av material som används och hur den utvinns för att arbeta mot ett hållbart samhälle på en global nivå. Detta är något som vi som ingenjörer kommer bära med oss in i framtiden för att bygga ett mer hållbart samhälle.

Litteraturförteckning

[1] P. Acarnley. *Stepping Motors*: *a guide to theory and practice*. The Institution of Engineering and Technology, London, United Kingdom, 4 edition, 2002. ISBN 9780852964170. Cited on page 7.

[2] Keith Bradsher. China tightens grip on rare minerals. https://www.nytimes.com/2009/09/01/business/global/01minerals.html, 2009-08-31. Cited on page 44.

[3] S. Constantinides. The demand for rare earth materials in permanent magnets. *Arnold Magnetic Technologies*, 2012. Cited on page 44.

[4] Dynapar. The driving force: Magnetic versus optical encoder engines. https://www.dynapar.com/technology/optical-encoders/, 2017-04-05. Cited on page 13.

[5] Mei Motion Engineering. Sinusoidal commutation. http://support.motioneng.com/Software-MPI_04_00/Topics/sinusoidal_commutation.htm, 2017-05-04. Cited on page 11.

[6] A.E Fitzgerald. *Electric Machinery*. McGraw-Hill, 6 edition, 2002. ISBN 0071121935. Cited on pages 3, 4, 6, and 7.

[7] I. Gottlieb. *Practical Electric Motor Handbook*. Newnes, 1 edition, 1997. ISBN 9780750636384. Cited on pages 4, 5, and 6.

[8] National Instruments. Encoder measurements: How-to guide. {http://www.ni.com/tutorial/7109/en/},2018-04-05, 2017. Cited on page 13.

[9] L. Wanhammar H. Johansson. *Digital filters using MATLAB*. Department of Electrical Engineering, Linköping University, 1 edition, 2011. ISBN 9780750636384. Cited on page 12.

[10] Jonathan Kaiman. Baotou is the world's biggest supplier of rare earth minerals and it's hell on earth. https://www.theguardian.com/sustainable-business/rare-earth-mining-china-social-environmental-costs, 2014-03-20. Cited on page 44.

[11] *Quadrature Clock Converterr*. LSI Computer Systems,, 1 2009. http://www.anaheimautomation.com/manuals/ics/L010546Cited on page 31.

[12] T. Franzén S. Lundgren. *Elkraftteknik*. Studentlitteratur, 1 edition, 2002. ISBN 9780750636384. Cited on pages 2, 4, and 5.

[13] Mathworks. Simulink. https://se.mathworks.com/products/simulink, 28-03-2018. Cited on page 14.

[14] *ENX RIO Encoders*. Maxon Motors, 7 2017. https://www.maxonmotor.com/medias/sys_master/root/8827190738974/Datenblatt-ENX16RIO-EN-4218182-Sept-17.pdf. Cited on page 17.

[15] *EC-max* 22. Maxon Motors, 5 2017. https://www.maxonmotor.com. Cited on page 17.

[16] Kelum Akurugoda Gamage Michael Thomas Ratcliffe. Bldc motor power control techniques appraisal. *International Conference on Power Engineering, Energy and Electrical Drives*, 2013. URL https://ieeexplore-ieee-org.e.bibl.liu.se/stamp/stamp.jsp?tp=&arnumber=6635873&tag=. Cited on page 10.

[17] Donald W. Novotny. Hysteresis motor. https://www.accessscience.com/content/hysteresis-motor/35000/, 2014. Cited on page 1.

[18] N. Nyachulue. 3-phase pm synchronous motor. Master's thesis, Helsinki Metropolia University of Applied Sciences, 2015. Cited on page 2.

[19] Rowena Ryan. Baotou is the world's biggest supplier of rare earth minerals and it's hell on earth. https://www.news.com.au/travel/world-travel/asia/baotou-is-the-worlds-biggest-supplier-of-rare-earth-minerals-and-its-hell-on-earth/news-story/371376b9893492cfc77d23744ca12bc5, 2015-04-17. Cited on page 44.

[20] L. Ljung T. Glad. *Reglerteknik, grundläggande teori*. Studentlitteratur AB, 4:7 edition, 2006. ISBN 9789144022758. Cited on page 16.

[21] *QML CLASS V RS-422 Quadruple Differential Line Receiver*. Texas Instruments, 2 2010. http://www.ti.com/lit/ds/symlink/am26ls33a-sp.pdf. Cited on page 32.

[22] D. Tisaj. Design and construction of a tachometer. Master's thesis, Murdoch University, Perth, Western Australia, 2014. Cited on page 13.

[23] Dr. Archana.G.Thosar Vinita S.Bondre. Study of control techniques for torque ripple reduction in bldc motor. *International Conference on Innovations in Power and Advanced Computing Technologies*, 2017. Cited on page 9.

[24] W.H. Yeadon A. Yeadon. *Handbook of Small Electric Motors* (*McGraw-Hill Handbooks*). McGraw-Hill, Two Penn Plaza, New York, 1 edition, 2001. ISBN 9780070723320. Cited on pages 6, 7, and 9.

7

Bilagor

7.1 C-kod för Motorstyrning

```
/*************************************************************/
/****************** C extentions and includes ****************/
/*************************************************************/
#include "c:\program\dsptools\math.h"
#define TRUE 0xFFFF
#define FALSE 0x0
#define ON 0xFFFF
#define OFF 0x0
/*************************************************************/
/************** C30 related variables and data ***************/
/*************************************************************/

#define TIMER0_INC_FREQ 8000000 /* Chrystal frequency = 32MHz  */
 /* --> 32 / 2 / 2  = 8MHz (pulse) */

#define TIMER1_INC_FREQ 8000000 /* Chrystal frequency = 32MHz  */
 /* --> 32 / 2 / 2  = 8MHz (pulse) */

#define CACHE_ON asm (" OR 1800h,ST") /* Enable Cache operation */

#define CACHE_OFF asm (" AND 07FFh,ST") /* Disable Cache operation */

#define Enable asm(" OR 2000h,ST")    /* Enable interrupt */

#define Disable asm(" AND 0DFFFh,ST")  /* Disable interrupt */

#define IDLE_CPU asm(" IDLE ")    /* Idle instruction */

#define TIMER0_INT_ON asm (" OR 100h,IE") /* Enable Timer 0 Int */

#define TIMER0_INT_OFF asm (" AND 0EFFh,IE") /* Disable Timer 0 Int */
```

```
#define EXT2_INT_ON asm (" OR 4h,IE") /* Enable Ext 2 Int */

#define EXT2_INT_OFF asm (" AND 0FFBh,IE") /* Disable Ext 2 Int */

#define PRIM_BUS_INIT  *(unsigned int *) 0x808064 = 0x10f0
 /* Bus operation with 7 waitstates internal OR external RDYN */

#define EXP_BUS_INIT  *(unsigned int *) 0x808060 = 0x0
 /* Bus operation with external RDYN for SPD arbitration */

/********* Timer 0 related variables and data ************/

#define TIMER0_INIT  *(unsigned int *) 0x808020 = 0x2C1
 /* Timer 0 operation from internal clock with pulse output
    at pin TCLK0 */

#define TIMER0_FREQ_INIT  *(unsigned long int *) 0x808028
 /* Initializes Timer 0 to correct frequency */

#define TIMER0_COUNTER  *(unsigned long int *) 0x808024
 /* Counter reg Timer 0 */

/******** Timer 1 related variables and data ***********/

#define TIMER1_IO_ON *(unsigned int *) 0x808030 = 0x206
/* Timer 1 initierad till I/O, pin satt till 1 */

#define TIMER1_IO_OFF *(unsigned int *) 0x808030 = 0x202
/* Timer 1 initierad till I/O, pin satt till 0 */

#define TIMER1_INIT_INT  *(unsigned int *) 0x808030 = 0x2C1
 /* Timer 1 operation from internal clock with pulse output
    at pin TCLK1 */

#define TIMER1_INIT_EXT  *(unsigned int *) 0x808030 = 0x0C1
 /* Timer 1 operation from external clock with pulse output
    at pin TCLK1 */

#define TIMER1_FREQ_INIT  *(unsigned long int *) 0x808038
 /* Initializes Timer 1 to correct frequency */

#define TIMER1_COUNTER  *(unsigned long int *) 0x808034
 /* Counter reg Timer 1 */
/*************************************************************/
/**************** I/O related variables and data *************/
/*************************************************************/
#define CPU_INPORT      *(unsigned int *) 0x804800 /* read 16 bits */
#define CPU_OUTPORT     *(unsigned int *) 0x804802 /* w/r 16 bits  */
#define IO_OUTPORT      *(unsigned int *) 0x805203 /* w/r 16 bits  */
#define IO_INPORT_LOW   *(unsigned int *) 0x805201 /* read low 16 bits  */
#define IO_INPORT_HIGH  *(unsigned int *) 0x805202 /* read high 16 bits  */
#define MUX             *(unsigned int *) 0x805204 /* w MUX channel  */
#define AD_READ_HIGH    *(unsigned int *) 0x805206 /* r 8 bits  */
#define AD_READ_LOW     *(unsigned int *) 0x805207 /* r 6 bits  */
#define SC              *(unsigned int *) 0x805205 /* w start AD conversion  */
#define EOC             *(unsigned int *) 0x805205 /* r bit 8, 1 when AD ready  */
```

```
#define OUT_DA1         *(unsigned int *) 0x805208 /* load DA 1 */
#define OUT_DA2         *(unsigned int *) 0x805209 /* load DA 2 */
#define OUT_DA3         *(unsigned int *) 0x80520A /* load DA 3 */
#define OUT_DA4         *(unsigned int *) 0x80520B /* load Da 4 */
#define UPDATE_DA       *(unsigned int *) 0x80520C /* Uppdate all */

/****************** Logger related variables and data ***********************/
#define EXT1 0x20000 /* start of EXT1 section*/
#define RAM_CELL *( float *) rampoint /* One position in EXT1 section */
#define RAM_FILL  *(unsigned long int *) rampoint /* Memory cell at rampoint */
#define RAM_LENGTH 0x20000 /* Lengh of EXT1 RAM */

#define TSAMPEL  *(unsigned long int *) 0x20000 /* Sampletid, 1 lsb per mikrosekund */
#define NO_OF_VAR  *(unsigned long int *) 0x20001 /* Antal loggade variabler (N) */
#define VAR_LENGTH *(unsigned long int *) 0x20002 /* Lngd hos loggad variabel */
#define LOG_POINT *(unsigned long int *) 0x20003 /* Pekare till sista skriva element */
#define VAR1_START *(unsigned long int *) 0x20004 /* Start variable 1 */
#define VAR2_START *(unsigned long int *) 0x20005 /* Start variable 2 */
#define VAR3_START *(unsigned long int *) 0x20006 /* Start variable 3 */
#define VAR4_START *(unsigned long int *) 0x20007 /* Start variable 4 */
#define LABEL_1 *(unsigned long int *) 0x20008 /* Namn variable 1 */
#define LABEL_2 *(unsigned long int *) 0x20009 /* Namn variable 2 */
#define LABEL_3 *(unsigned long int *) 0x2000A /* Namn variable 3 */
#define LABEL_4 *(unsigned long int *) 0x2000B /* Namn variable 4 */
int rampoint;

/*****************************************************************************/
/****************** Declarations for main and interrupt **********************/
/*****************************************************************************/

#define delay_5us       for (i=1; i<10; i++ )  /* wait 5us */
#define delay_1_5ms     for (i=1; i<3000; i++ ) /* wait 1.5ms */
#define delay_150ms     for (i=1; i<300000; i++ ) /* wait 150ms */
#define delay_10s       for (i=1; i<6000000; i++ ) /* wait 10s */

#define SYST_FREQ       1000  /* Interrupt frequency [Hz] */
#define MY_PI           3.1415926
#define MY_2PI          6.283185
#define MY_2PI_3        2.09439 /* phase offset, 120 deg */
#define MY_4PI_3        4.18879 /* phase offset, 240 deg */
#define MY_PI_2         1.57080 /* phase offset, 90 deg */
#define MY_8_PI         25.13274
#define POS_LIM         6.28   /* + 1 turn */
#define NEG_LIM         18.85  /* - 1 turn */

#define POS_INCR        0.000191747*4 /* radians / encoder step (32768/turn) */
#define ENC_COUNT  32768   /* //32768 Quadrature count of encoder one turn*/
#define NOPOL_2         1.0  /* 2 pole motor */
#define SC_IPHASE_UPH  4.46  /* Scalefactor of power stage [V/A] */
#define SC_TQ_I   36.7  /* Inverse torque constant of motor [A/Nm] */
#define TQ_MAX  0.02  /* Max torque Nm */

float rotor_pos, u_input_dem, u_curr_ph1, u_curr_ph2, u_curr_ph3;
float torque_dem, curr_ph1, curr_ph2, U_delta_time, U_taco;
float ipeak, phase_pos, electrical_pos, avg;

int i, j, count1, count2, count3, delta, delta2, avg_int, Counter_ms;
int slot, tmp, start, count84_2, count84_1, delta84, slot12[12];
```

```
int tmr1_2, tmr1_1, delta_tmr, Int03_flag, Timer0_Int03;
float rotor_speed, rotor_speed_err, rotor_speed_err_I, dt;
float rotor_speed_delta, Tstart, tq_dem, tq_dem_lim, Kp, Ki;
float rotor_speed_setpoint_final;
int sys = SYST_FREQ;
/***************** DECLARATIONS OF PROCEDURES ***********************/
/************************ I/O HANDLERS ******************************/
/********************************************************************/

float  Bip_limit(float Input, float Limit)
{
/* Limits Input to +- Limit */
float Output;

if ((Input <= Limit) & (Input >= -Limit))
Output = Input;
else if (Input > Limit)
Output = Limit;
else Output = -Limit;
return Output;
}

void  Put_DA(float Ana1, float Ana2, float Ana3, float Ana4)
{
#define DA_MASK         0x00000FFF
#define DA_OFFSET       0x00000800
#define DA_SCALE        204.75  /* +- 10 V output */
#define DA_MAX          10.0

unsigned int DA1_word, DA2_word, DA3_word, DA4_word;
float DA1_lim, DA2_lim, DA3_lim, DA4_lim;

DA1_lim = Bip_limit(Ana1, DA_MAX); /* Limit input */
DA2_lim = Bip_limit(Ana2, DA_MAX);
DA3_lim = Bip_limit(Ana3, DA_MAX);
DA4_lim = Bip_limit(Ana4, DA_MAX);

DA1_word = (DA1_lim * DA_SCALE);  /* Scaling and type conversion */
DA2_word = (DA2_lim * DA_SCALE);
DA3_word = (DA3_lim * DA_SCALE);
DA4_word = (DA4_lim * DA_SCALE);

OUT_DA1 =  (DA1_word + DA_OFFSET) & DA_MASK;  /* unipolar conv. */
OUT_DA2 =  (DA2_word + DA_OFFSET) & DA_MASK;
OUT_DA3 =  (DA3_word + DA_OFFSET) & DA_MASK;
OUT_DA4 =  (DA4_word + DA_OFFSET) & DA_MASK;

UPDATE_DA = 0x0; /* Uppdate outputs by dummy write */
}

/************************** Logger setup ****************************/
void  Logger_setup()
{
rampoint = EXT1;
for (j = 0; j < RAM_LENGTH; j++) /* Init memory to int 0 */
{
RAM_FILL = 0x0;
```

```
rampoint++;
}
TSAMPEL = 1000; /* In us, XXXX Hz => XXXXus */
  NO_OF_VAR = 4;
VAR_LENGTH = 16384;
LOG_POINT = 0;
VAR1_START = 0x20100;
VAR2_START = VAR1_START + VAR_LENGTH;
VAR3_START = VAR2_START + VAR_LENGTH;
VAR4_START = VAR3_START + VAR_LENGTH;
LABEL_1 = 0x56415231; /* VAR1 */
LABEL_2 = 0x56415232; /* VAR2 */
LABEL_3 = 0x56415233; /* VAR3 */
LABEL_4 = 0x56415234; /* VAR4 */
}

/************************ Logger store ************************/
void  Logger_store(float Var1, float Var2, float Var3, float Var4)
{

/*if (LOG_POINT < VAR_LENGTH-1)*/

if (LOG_POINT < VAR_LENGTH-1)
{

rampoint = VAR1_START + LOG_POINT;
RAM_CELL = Var1 ;
rampoint = VAR2_START + LOG_POINT;
RAM_CELL = Var2 ;
rampoint = VAR3_START + LOG_POINT;
RAM_CELL = Var3 ;
rampoint = VAR4_START + LOG_POINT;
RAM_CELL = Var4 ;

LOG_POINT = LOG_POINT  + 1;
if (LOG_POINT >= VAR_LENGTH)
 LOG_POINT = 0;

}
}
/*************************************************************/
/****************** CONTROL SIGNAL ***************************/
/*************************************************************/



/*************************************************************/
/****************** Motor position Init **********************/
/*************************************************************/

void correct_pos_motor(float phase_position, float i_peak)
{
curr_ph1 = i_peak*sin(phase_position); /* Current for phase 1 */
curr_ph2 = i_peak*sin(phase_position - MY_2PI_3); /* Current for phase 1 */
u_curr_ph1 = curr_ph1*SC_IPHASE_UPH;
u_curr_ph2 = curr_ph2*SC_IPHASE_UPH;

Put_DA(curr_ph1,U_delta_time,u_curr_ph1,u_curr_ph2);
```

```
}

/*************************************************************/
/****************** INTERRUPT PROCEDURE **********************/
/*************************************************************/
void    c_int09()
{
    /* Main control function. Call frequency: XXXX Hz  from Timer 0*/

Enable;

/* Slow start */


/* Read Timer1 to get position and speed */
tmr1_2 = tmr1_1;
tmr1_1 = TIMER1_COUNTER;
delta_tmr = tmr1_1 - tmr1_2;  /* Speed estimation */
if (delta_tmr < 0) /* Handle counter reset */
{
delta_tmr = delta_tmr + ENC_COUNT;
}
rotor_speed = delta_tmr * POS_INCR *  SYST_FREQ; /* Set speed scale rad/s */
rotor_pos = tmr1_1 * POS_INCR; /* radian */

/* Regulator */

rotor_speed_err = rotor_speed_setpoint_final - rotor_speed;
rotor_speed_err_I = rotor_speed_err_I + rotor_speed_err/SYST_FREQ;

tq_dem  = Ki * rotor_speed_err_I + Kp * rotor_speed_err;
tq_dem_lim = Bip_limit(tq_dem, TQ_MAX);

/* Commutation sine */
ipeak = SC_TQ_I * tq_dem_lim; /* peak current */

electrical_pos = rotor_pos * NOPOL_2; /* Electrical angle */
phase_pos = fmod(electrical_pos, MY_2PI); /* rest modulo MY_2PI */

correct_pos_motor(phase_pos, ipeak);
 /*System freq*/
Logger_store(curr_ph1, rotor_speed_err, rotor_speed, rotor_speed_err_I);
}
/*}*/

void    c_int03() /* Read counter Timer 0. Dummy for external Interrupt */
{
Timer0_Int03 = TIMER0_COUNTER;
Int03_flag = TRUE;
}
/*************************************************************************/
/***************************** MAIN **************************************/
/*************************************************************************/

main()
{
    /* Init CPU */
    CACHE_OFF;
```

```
    PRIM_BUS_INIT;
    EXP_BUS_INIT;

    /* Init logger */
    Logger_setup();

    /* Init timer */
    TIMER0_FREQ_INIT = TIMER0_INC_FREQ / SYST_FREQ; /* Control loop interupt generation */
    TIMER1_FREQ_INIT = ENC_COUNT;  /* Full count for one rev of motor in quadrature mode */
    Int03_flag = FALSE;

    /* Init data */
    count1 = count2 = count84_1, count84_2 = Timer0_Int03 = 0;
    u_curr_ph1 = 0.0;
    u_curr_ph2 = 0,0;
    torque_dem = curr_ph1 = curr_ph2 = U_delta_time = 0.0;
    Counter_ms = 0;
    electrical_pos = 0.0;
    slot = 0;
    start = 0;
    Tstart = 0.1;
    rotor_speed_setpoint_final = 62.83185; /* 10 varv per sekund * 62.8319 */
    rotor_speed_delta = rotor_speed_setpoint_final / (Tstart * SYST_FREQ);


    /* Init regulator */
    Kp = 0.0092;
    Ki = 0.35;
    rotor_speed_err = rotor_speed_err_I = 0.0;
    dt = 1/SYST_FREQ;

    /* Counter intit to get phase position right on motor */
    ipeak = 0.2; /* 200mA peak current */
    rotor_pos = 0.0;
    electrical_pos = rotor_pos * NOPOL_2; /* Electrical angle */
phase_pos = fmod(electrical_pos, MY_2PI); /* rest modulo MY_2PI */
correct_pos_motor(phase_pos, ipeak);


  delay_10s; /* wait for rotor to stop */

    TIMER0_INIT; /* Start timer 0 */
    TIMER1_INIT_EXT; /* Start timer 1 with external clock */
  TIMER1_COUNTER = 8192; /* //8192; 90 ELECTRICAL degrees */
  tmr1_2 = tmr1_1 = 8192; /*//8192; Ensure smooth start */

    TIMER0_INT_ON; /* enable timer interrupt */
    EXT2_INT_ON;   /* enable ext interrupt 2 */
    Enable;
    IDLE_CPU;  /* Sleep and wait for int */

} /* main */
```

7.2 C-kod för extern pulsgivare

```
/*************************************************************/
/****************** C extentions and includes ****************/
/*************************************************************/
#include "c:\program\dsptools\math.h" /* c: new 170816 */
#define TRUE 0xFFFF
#define FALSE 0x0
#define ON 0xFFFF
#define OFF 0x0
/*****************************************************************/
/************* C30 related variables and data ****************/
/*****************************************************************/

#define TIMER0_INC_FREQ 4000000 /* Chrystal frequency = 32MHz  */
/* --> 32 / 2 / 2 / 2 = 4MHz  (square) */

#define TIMER1_INC_FREQ 8000000 /* Chrystal frequency = 32MHz  */
  /* --> 32 / 2 / 2  = 8MHz (pulse) */

#define CACHE_ON asm (" OR 1800h,ST") /* Enable Cache operation */

#define CACHE_OFF asm (" AND 07FFh,ST") /* Disable Cache operation */

#define Enable asm(" OR 2000h,ST")    /* Enable interrupt */

#define Disable asm(" AND 0DFFFh,ST")  /* Disable interrupt */

#define IDLE_CPU asm(" IDLE ")    /* Idle instruction */

#define TIMER0_INT_ON asm (" OR 100h,IE") /* Enable Timer 0 Int */

#define TIMER0_INT_OFF asm (" AND 0EFFh,IE") /* Disable Timer 0 Int */

#define EXT2_INT_ON asm (" OR 4h,IE") /* Enable Ext 2 Int */

#define EXT2_INT_OFF asm (" AND 0FFBh,IE") /* Disable Ext 2 Int */

#define PRIM_BUS_INIT  *(unsigned int *) 0x808064 = 0x10f0
 /* Bus operation with 7 waitstates internal OR external RDYN */

#define EXP_BUS_INIT  *(unsigned int *) 0x808060 = 0x0
 /* Bus operation with external RDYN for SPD arbitration */

/******** Timer 0 related variables and data ***********/

#define TIMER0_INIT  *(unsigned int *) 0x808020 = 0x3C1
 /* Timer 0 operation from internal clock with square output
    at pin TCLK0 */

#define TIMER0_FREQ_INIT  *(unsigned long int *) 0x808028
 /* Initializes Timer 0 to correct frequency */

/******** Timer 1 related variables and data ***********/

#define TIMER1_IO_ON *(unsigned int *) 0x808030 = 0x206
/* Timer 1 initierad till I/O, pin satt till 1 */
```

```
#define TIMER1_IO_OFF *(unsigned int *) 0x808030 = 0x202
/* Timer 1 initierad till I/O, pin satt till 0 */

#define TIMER1_INIT_INT  *(unsigned int *) 0x808030 = 0x2C1
 /* Timer 1 operation from internal clock with pulse output
    at pin TCLK1 */

#define TIMER1_INIT_EXT  *(unsigned int *) 0x808030 = 0x0C1
 /* Timer 1 operation from external clock with pulse output
    at pin TCLK1 */

#define TIMER1_FREQ_INIT  *(unsigned long int *) 0x808038
 /* Initializes Timer 1 to correct frequency */

#define TIMER1_COUNTER  *(unsigned long int *) 0x808034
 /* Counter reg Timer 1 */

/*************************************************************/
/****************** I/O related variables and data ***********/
/*************************************************************/

/*------------------ Logger related variables and data ----------------------*/
#define EXT1 0x20000 /* start of EXT1 section*/
#define RAM_CELL *( float *) rampoint /* One position in EXT1 section */
#define RAM_FILL  *(unsigned long int *) rampoint /*
#define RAM_LENGTH 0x20000 /* Lengh of EXT1 RAM */

#define TSAMPEL  *(unsigned long int *) 0x20000 /* Sampletid, 1 lsb per mikrosekund */
#define NO_OF_VAR  *(unsigned long int *) 0x20001 /* Antal loggade variabler (N) */
#define VAR_LENGTH *(unsigned long int *) 0x20002 /* Lngd hos loggad variabel */
#define LOG_POINT *(unsigned long int *) 0x20003 /* Pekare till sista skriva element */
#define VAR1_START *(unsigned long int *) 0x20004 /* Start variable 1 */
#define VAR2_START *(unsigned long int *) 0x20005 /* Start variable 2 */
#define VAR3_START *(unsigned long int *) 0x20006 /* Start variable 3 */
#define VAR4_START *(unsigned long int *) 0x20007 /* Start variable 4 */
#define LABEL_1 *(unsigned long int *) 0x20008 /* Namn variable 1 */
#define LABEL_2 *(unsigned long int *) 0x20009 /* Namn variable 2 */
#define LABEL_3 *(unsigned long int *) 0x2000A /* Namn variable 3 */
#define LABEL_4 *(unsigned long int *) 0x2000B /* Namn variable 4 */
int rampoint;

/*************************************************/
/****** Declarations for main and interrupt *******/
/*************************************************/

#define SYST_FREQ       2000 /* Interrupt frequency */

float dummy2, temp_timer;
int temp1, dummu = -;
float rotations, counter;
int j;
int temp_timer[12],pulses;
/***************** DECLARATIONS OF PROCEDURES ******************************/

/******************************** Logger setup *****************************/
void  Logger_setup()
{
```

```
rampoint = EXT1;
for (j = 0; j < 0x20000; j++) /* Init memory to int 0 */
{
RAM_FILL = 0x0;
rampoint++;
}
TSAMPEL = 8333; /* 120 Hz */
  NO_OF_VAR = 4;
VAR_LENGTH = 16384; /*gammal: 2048 */
LOG_POINT = 0;
VAR1_START = 0x20100;
VAR2_START = VAR1_START + VAR_LENGTH;
VAR3_START = VAR2_START + VAR_LENGTH;
VAR4_START = VAR3_START + VAR_LENGTH;
LABEL_1 = 0x534c4954; /*SLIT */
LABEL_2 = 0x544f4d31; /*TOM1 */
LABEL_3 = 0x544f4d32; /*TOM2 */
LABEL_4 = 0x544f4d33; /*TOM3 */
}

/************** Logger store ********************/
void  Logger_store(float Var1, float Var2, float Var3, float Var4)
{
if (LOG_POINT < VAR_LENGTH -1)
{
rampoint = VAR1_START + LOG_POINT;
RAM_CELL = Var1 ;
rampoint = VAR2_START + LOG_POINT;
RAM_CELL = Var2 ;
rampoint = VAR3_START + LOG_POINT;
RAM_CELL = Var3 ;
rampoint = VAR4_START + LOG_POINT;
RAM_CELL = Var4 ;

LOG_POINT = LOG_POINT  + 1;
if (LOG_POINT >= VAR_LENGTH)
LOG_POINT = 0;
}
}

/*************************************************************************/
/******************** INTERRUPT PROCEDURE ********************************/
/*************************************************************************/
void    c_int09()
{
Enable;
/*  Dummy */
counter +=1;
if (counter == 2001)
        counter =0;

}

void    c_int03()
{
temp1 = TIMER1_COUNTER;
/* Opto interrupt EXT2_INT */
```

```
/* >>>>>>>>>>>>>>>>>>> Er kod hr <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Ett anrop fr varje slits i skivan */
dummy += 1;
pulses +=1;

if( pulses < 12)
{
pulses = 0;
}

clock_timer = temp1 - temp_timer(pulses);
temp_timer(pulses) = temp1;


Logger_store(pulses, rotations, clock_timer, dummy);
}

main()
{
    /* Init CPU */
    CACHE_OFF;
    PRIM_BUS_INIT;
    EXP_BUS_INIT;

    /* Init logger */
    Logger_setup();

    /* Init timer */
    TIMER0_FREQ_INIT = TIMER0_INC_FREQ / SYST_FREQ; /* 2000 Hz interupt generation*/
    TIMER1_FREQ_INIT = 0xffffffff;  /* all 1 in counter*/

    /* Init data */
    /* Vad ni vill initiera */
rotations = 0;
dummy2 = 0;
pulses = 0;
dummy = 0.0;
counter = 0.0;

    TIMER0_INIT; /* Start timer 0 */
    TIMER1_INIT_INT ; /* Start timer 1 with internal clock */

    TIMER0_INT_OFF; /* Disable timer 0 interrupt */
    EXT2_INT_ON;    /* Enable ext interrupt 2 */
    Enable; /* Enable all */

    IDLE_CPU;  /* Sleep and wait for int */

} /* main */
```