CHALMERS

Prototypframtagning av analoga lägesgivare för linjära ställdon

Prototype production of analog position sensors for linear actuators

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

Nabil Fatmi
Pontus Petersson

Institutionen för signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige, 2014

Prototypframtagning av analoga lägesgivare för linjära ställdon

FÖRORD

Det här examensarbetet är det sista momentet på mekatronikprogrammet, en treårig högskoleutbildning (180 HP) vid Chalmers tekniska högskola. Arbetet är gjort på institutionen för signaler och system och examinator är Bertil Thomas. Arbetet är på 15 HP och har gjorts under 8 veckor på vårterminen 2014.

Vi vill börja med att tacka Marika Stolpe, Teknikområdeschef på Consat Engineering AB, för att hon tillsammans med Sven Svensson, produktutvecklingschef på SKF Actuation System AB, gav oss chansen och möjligheten att genomföra ett relevant examensarbete för mekatronikprogrammet. Vi vill även tacka Jonas Williamsson, handledare på Consat, och Göran Hult, handledare på Chalmers tekniska högskola, för att de stöttat och hjälpt oss under arbetets gång. Sist men inte minst vill vi tacka hela personalgruppen på Consat för att dom välkomnat oss och tagit sig tid att dela med sig av deras erfarenheter.

Nabil Fatmi & Pontus Petersson
Göteborg, 28 Maj 2014

SAMMANFATTNING

SKF Actuation System AB tillverkar olika typer av ställdon. Då ställdonet ofta styrs av ett styrsystem behövs feedback om ställdonets position. För detta används olika typer av lägesgivare, en typ som används är en mekanisk potentiometer som ger en absolut analog utsignal. En annan typ är en inkrementell encoder E2 med ett digitalt pulståg som utsignal. SKF vill undersöka om det är möjligt att ta fram en ny typ av encoder, E4 som kan ersätta den mekaniska potentiometern. Consat Engineering AB har under 2013 gjort en förstudie för en sådan analog lägesgivare. Den nya givaren skall vara absolut, det vill säga att givaren skall veta var ställdonet befinner sig direkt från start och utsignalen skall vara analog precis som för den mekaniska potentiometern.
I detta arbete tas en prototyp eller labbkort för E4 fram, för att verifiera att konceptet från förstudien fungerar. Kretsen skall matas med 12 V eller 24 V DC och innehåller bland annat två hallgivare, en mikroprocessor, ett lågpassfilter och en strömbackup.
Mikroprocessorn räknar ut positionen av ställdonet med hjälp av pulserna som fås från hallgivarna. Processorn levererar en PWM-signal som beror på den aktuella positionen av ställdonet. PWM-signalen filtreras och förstärks i kretsen så att feedbacksignalen blir en analog signal med diskreta steg. Utspänningen ligger mellan 0 V och 10 V där 0 V motsvarar hemmaläge och 10 V motsvarar det maximala ändläget för ställdonet. Strömbackupen är för att hantera den eftergång som kan ske vid ett planerat eller oplanerat spänningsfall.
Resultatet av arbetet visar att det är möjligt att bygga en E4 encoder som uppfyller grundkraven som SKF ställde.

ABSTRACT

SKF Actuation Systems AB manufactures various types of actuators. Actuators are often controlled by a control system that requires feedback about actuator position. Different kinds of position sensors are used to produce the feedback signal. One of the sensors is a mechanical potentiometer that provides an absolute analog output voltage. Another type is an incremental encoder E2, which gives an output in the form of a digital pulse signal.
SKF wants to examine if it is possible to develop a new type of encoder, E4, which can replace the mechanical potentiometer. Consat Engineering AB has during 2013 made a feasibility study for this kind of analog position sensor. The new sensor should be absolute, meaning the sensors output voltage will be correctly directly after power up.

In this study a prototype in the form of a Prototype Board for the E4 is developed to verify that the concept of the earlier feasibility study works. The circuit will be supplied with 12 V or 24 V and contains two Hall-sensors, a microprocessor, a low-pass filter and a power backup. The microprocessor calculates the position of the actuator using the pulses obtained from the Hall-sensors. The processor delivers a PWM signal that depends on the current position of the actuator. The PWM signal is filtered and amplified in the circuit to produce an analog signal. The output-voltage is between 0 V-10 V where 0 V corresponds to the initial position and 10 V corresponds to the maximal/end position. The backup power is to handle planned or unplanned power failure. The conclusion of this study shows that it is possible to build an E4 encoder that meets SKF basic requirements.

INNEHÅLL

1 Inledning 1
1.1 Bakgrund 1
1.2 Syfte 1
1.3 Avgränsningar 1
1.4 Precisering och frågeställningen 2
2 Teknisk bakgrund 3
2.1 PIC 3
2.2 PWM 3
2.3 LP-Filter 4
2.4 FilterPro 4
2.5 LT Spice 5
2.6 Absolut Encoder 5
2.7 Inkrementell Encoder 5
2.8 Linjär encoder 6
2.9 Hallgivare 6
2.10 Rotary encoder 6
2.11 SKF ställdon 6
2.12 Lägesgivare för ställdon 7
2.12.1 Mekanisk potentiometer 7
2.12.2 Inkrementell rotary encoder 7
2.12.3 Ändlägesgivare 8
3 Beskrivning av önskat system 9
3.1 Krav 9
3.2 Konceptbeskrivning 9
4 Metod 14
5 Sammanfattning av förstudie 15
5.1 Filter 15
5.2 Spänningsdelning 16
5.3 Strömbackup 16
5.4 Spänningsreglering 16
5.5 Hallgivarna 16
5.6 Processorn 17
5.7 Metod för D/A-omvandling 17
5.8 Lågpassfiltret och förstärkning 17
6 Genomförande 18
6.1 Analys av förstudie/simuleringar/beräkningar 18
6.1.1 Filter 18
6.1.2 Spänningsdelning 18
6.1.3 Strömbackup 19
6.1.4 Hallgivarna 19
6.1.5 Processorn 19
6.1.6 Metod D/A-omvandling 19
6.1.7 OP och LP-filter 19
6.1.8 Konstruktionen av nytt filter i FilterPro 21
6.2 Val av komponenter 30

6.3 ... 30
6.4 Preliminär krets ... 31
6.5 Mjukvara ... 32
6.5.1 Utvecklingsmiljö ... 32
6.5.2 Övergripande ... 32
6.5.3 Huvudprogram ... 33
6.5.4 Hantering av pulser ... 34
6.5.5 Kalibrering ... 37
6.5.6 Felhantering ... 37
6.5.7 Hantering av spänningsbortfall ... 38
6.5.8 PWM-utgång ... 39
6.5.9 Konfiguration av processor ... 41
6.6 Realisering och tester ... 42
6.6.1 Kopplingsdäcket ... 42
6.6.2 Labbkort ... 44
6.6.3 Testsystem med Labbkort ... 45
6.6.4 Genomförda tester ... 46
7 Resultat ... 48
7.1 Begränsning i hur givaren kan användas ... 48
7.2 Testresultat på framtagen hårdvara ... 48
7.3 Kretsritningen och BOM ... 53
7.4 Mjukvara ... 53
8 Diskussion ... 54
8.1 Direkta risker med framtagen givare ... 54
8.2 Möjliga risker med framtagen givare ... 55
8.3 Diskussion av resultatet ... 55
8.4 Förbättringsmöjligheter allmänt ... 57
8.5 Övriga tankar ... 58
8.6 Förbättringsmöjligheter för Mjukvaran ... 59
9 Slutsatser ... 61
10 Referenser ... 62
Appendix ... i
Bilaga A. Kretsschema ... i
Bilaga B. BOM ... ii
Bilaga C. Mjukvara ... iii

BETECKNINGAR

PIC: Peripheral Interface Controller

OP: Operationsförstärkare

LP-filter: Lågpassfilter

HP-filter: Högpassfilter

PWM: Pulsbreddsmodulering, "Pulse-width modulation"

BOM: Bill of materials

DAC: Digital to Analog Converter

PLL: Phase-Locked Loop

D/A: Digital till analog

A/D: Analog till digital

Picoscope: Instrument för signalanalys på dator

GUI: "Graphical user interface"

I/O: Ingångar/utgångar

E2: Encoder version 2

E3: Encoder version 3

Huvudmatning: 24 V / 12 V-matningen som förser hela kretsen med ström

Huvudström: Strömmen som dras från huvudmatningen

Huvudspänning: Spänningen för huvudmatningen

Internmatning: 5 V-matningen inom kretsen som har en strömbackup

Internström: Strömmen som dras från internmatningen

Internspänning: Spänningen för internmatningen

DC: Likström, "Direct Current"

LED: Lysdiod, "Light Emitting Diode"

Debounce: En teknik för att ignorera "studsen" vid en knapptryckning

Peak to Peak: Amplitudvärdet från topp till botten på en signal

BOR: Brown Out Reset

1 INLEDNING

En analog lägesgivare har tagits fram vid Consat Enginering åt SKF Actuation System. För att få en bild av projektets bakgrund, mål och omfattning beskrivs det utförligt i detta kapitel.

1.1 Bakgrund

SKF Actuation System tillverkar ställdon som är modulärt uppbyggda med stor variation i slaglängder, ställhastigheter och lastförmåga. I dagsläget används två olika system för positionering av ställdonet. Den första lägesgivaren är av encodertyp där fält från magneter som är monterade på drivmotorns axel känns av och mottags som ett pulståg till överordnat system. Den andra är en linjär potentiometer av foiltyp som ger en analog utspänning till överordnat system. Den sistnämnda typen har problem med mekanisk utslitning. För att komma ifrån detta problem vill SKF använda sig av en lägesgivare av encodertyp som i sin tur ger en analog utsignal. En analog signal bedöms kunna anslutas till ett överordnat styrsystem till lägre totalkostnad än pulssignaler. Consat Engineering AB har under 2013 gjort en förstudie för en sådan analog lägesgivare. Syftet var att uppskatta tid och kostnad för att realisera lägesgivaren, kartlägga risker och att ta fram en konceptlösning. Förstudien skall ses som en ramspecifikation av lägesgivarens egenskaper.

1.2 Syfte

Syftet med arbetet är att bevisa att det är möjligt att ta fram en lägesgivare till SKF ställdon enligt SKF krav. Konceptlösningen från förstudien skall realiseras för att verifiera att den tekniska lösningen fungerar. Lägesgivaren ska kunna monteras på de ställdon som SKF producerar i dagsläget utan att behöva göra ändringar på själva ställdonet.

1.3 Avgränsningar

Då projekttiden är begränsad kommer inte den färdiga, fullständigt testade slutprodukten att tas fram. Arbetet kommer att satsa på att ta fram ett labbkretskort med hårdvara. Även mjukvara kommer att tas fram för att visa att konceptlösningen fungerar. Kortet kommer endast att användas i rumstemperatur och inga ytterligare tester kommer att göras.
Inga ekonomiska beräkningar kommer att göras för att givaren skall hålla budget.

Slutprodukten skall få plats i nuvarande plastkåpa för tidigare modeller av encodern, detta kommer att tänkas på, med inga beräkningar kommer att göras.

1.4 Precisering och frågeställningen

Är lösningsförslaget som finns i förstudien realiserbart?
Hur kommer den nya givaren skilja sig mot den analoga potentiometern för användaren?

2 TEKNISK BAKGRUND

Nedan förklaras olika tekniker och begrepp som använts i det här arbetet.

2.1 PIC

PIC är en förkortning till ”Peripheral Interface Controller” och är en typ av mikroprocessorer tillverkade av Microchip Technology. Principen i PIC är att använda I/O som finns på PIC-processorn för att ta emot och/eller skicka data både digitalt och analogt.
Det finns olika serier av PIC där är det olika egenskaper som skiljer mellan de olika serierna, exempelvis:

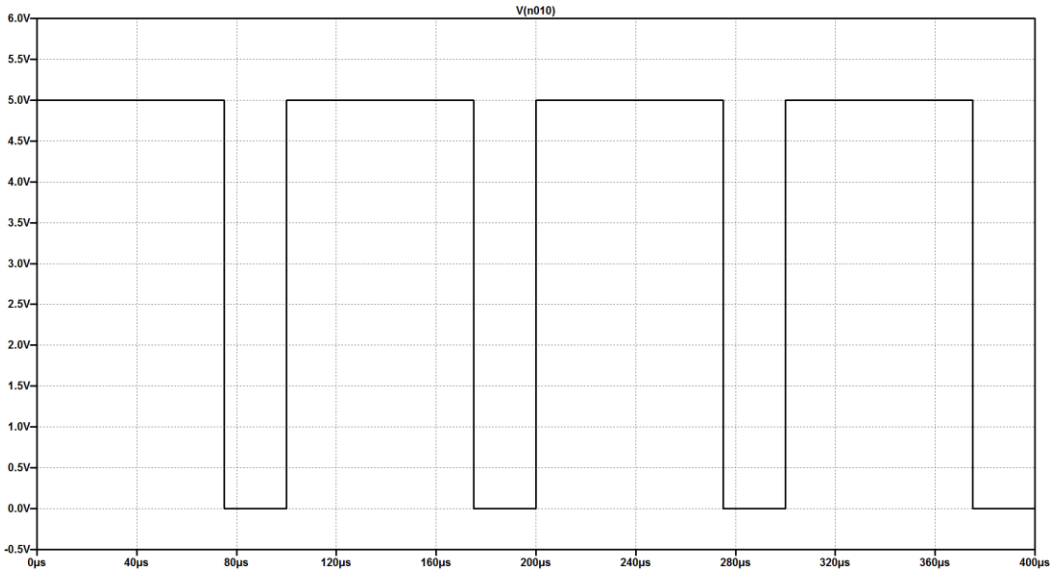
- Antal pinnar och I/O i PIC:en
- Klockfrekvens som bestämmer hur snabb processor ska hantera data.
- Storlek av internminne.

Det finns även andra egenskaper som är viktiga när man väljer PIC och detta kan man läsa i databladet (Microchip, 2012) [1].

2.2 PWM

PWM står för Pulsbreddsmodulering (*Pulse width modulation).* Metoden används för att kunna använda en digital utgång för att skapa en analog signal. Signalen som PWM-utgången genererar har formen av en fyrkantsvåg med en bestämd frekvens (figur 1). Förhållandet mellan hur lång tid pulsen är hög (1) och lång tid pulsen är låg (0) avgör vad utsignalens medelspänning kommer bli. Förhållandet mellan tiden utsignalen är hög och periodtiden kallas för duty cycle eller PWM-värde. Den maximala utspänningen som kan fås är om pulsen är hög hela tiden och den lägsta om signalen är låg hela tiden. Där emellan ger en ökning av signalens duty cycle en linjär ökning av den analoga signalens medelspänning. (PWM Signal and What is it Used For?, 2003) [2].

Ex. För en signal där amplituden är 5 V och pulsen hög (1) under 75 % av cykel och låg (0) under 25 % av cykel [Fig. 1] blir medelvärdet på utsignalen: $5\,V * 75\,\% = 3.75\,V$ detta motsvarar en 75 % duty cycle.



Figur 1 Pulsvåg som genererar 75 % duty cycle.

2.3 LP-Filter

LP-filter eller lågpassfilter är en typ av filter som används för att filtrera bort låga frekvenser. LP-filtret låter alla frekvenser som är lägre än den valda brytfrekvensen att passera och dämpar de frekvenser som är högre än den valda frekvensen.
Det finns andra typer av filter ex. högpassfilter som är motsatsen till lågpassfilter och det finns även bandpassfilter som dämpar alla frekvenser förutom frekvenser inom ett valt intervall.
Framställning av filter kan göras genom att själv beräkna de olika komponenter som ingår i filtret eller genom att använda program ex. FilterPro
Ett lågpassfilter kan byggas med ett motstånd och en kondensator detta kallas då för ett RC-filter. Ett sådant filter är av första ordningen, ordningen på ett filter avgörs av hur många kondensatorer ett filter har. Ju högre ordning ett lågpassfilter har desto brantare dyker amplitudkurvan. Man pratar ofta om brytfrekvens och förenklat är det vid den frekvensen som filtret börjar dämpa. Ett filter av andra ordningen dämpar med 40dB/dekad, tredje ordningen 60dB/dekad och så vidare. Om en förstärkare används i filtret kallas filtret för ett aktivt filter.

2.4 FilterPro

FilterPro är ett program som används för att ta fram filter. Programmet ägs av Texas Instruments och är helt gratis. Programmet ger förslag på design samt BOM lista på komponenter som behövs. Programmet är tillgängligt på Texas Instruments websida och är helt gratis.

2.5 LT Spice

LT Spice är ett program som används för att simulera kretsar. Programmet är väldigt användbart i hårdvarukonstruktionen eftersom det ger möjlighet att göra ett virtuellt test av kretsen. LT Spice ger även möjlighet att göra virtuella mätningar och kontrollera att komponenterna fungerar som det är tänkt.
Det finns några begränsningar när det gäller simulering av kretsar. Ibland saknas komponenter som man vill simulera. Det går att skapa sina egna modeller om användaren har tillgång till tillräckligt data för att skapa rätt komponent. Det finns även andra fall där komponenterna inte går att simulera ex. en PIC.

2.6 Absolut Encoder

En absolut encoder ger positionsinformation om var läshuvudet befinner sig. För att vara absolut krävs att positionen är känd även om systemet har varit utan ström. Positionen är tillgänglig för avläsning så fort encodern får ström igen. En absolut encoder har en viss upplösning, antal diskreta lägen som den kan befinna sig i. Ett sätt för att få reda på vilket läge encodern står är att varje läge har en specifik binär kod, ofta används graykod för detta ändamål. Ett annat sätt är att positionen lagras i ett minne, antingen mekaniskt eller digitalt med hjälp av batteri. Det finns olika tekniker för att avläsa en absolut position, bland annat optiskt, magnetiskt och mekaniskt.

2.7 Inkrementell Encoder

En inkrementell encoder ger ingen positionsinformation i sig själv utan ger bara information om rörelseriktning och hastighet. Ett överordnat system tar hand om informationen från encodern, och kan med hjälp av en referenspunkt att utgå ifrån kan en position fås.
Utsignalen från en inkrementell encoder kan vara av typen Quadrature signaler, dvs. två digitala fyrkants-signaler, A och B som ligger 90 grader fasförskjutna. På grund utav fasförskjutningen kan man på så sätt kunna avläsa vilket håll man rör sig åt. Upplösningen beror på hur tätt detta pulståg kommer, genom att öka densiteten på pulståget kan man öka upplösningen. Genom att använda sig av en tredje referenssignal, ofta kallad Z (från engelska zero), kan överordnat system räkna ut positionen för det som mäts.
Fördelar är att den är billig och väldigt enkel i sin konstruktion.

2.8 Linjär encoder

En linjär encoder används för att mäta position och/eller hastighet och riktning på en linjär rörelse, t.ex. ett ställdon. En linjär encoder kan vara antingen absolut eller inkrementell.

2.9 Hallgivare

En hallgivare är en givare som ger utslag när den påverkas av ett magnetfält genom att använda sig av halleffekten. Halleffekten uppkommer när ström går genom en ledare i ett magnetfält bildas en potentialskillnad(hallspänning) vinkelrät mot strömriktningen. Givarna används inom många områden, i bilar används de i ABS system för hastighetsbestämning av hjul samt för att veta när gnistan skall komma i ett tändningssystem. (Edward, 2006) [3].

2.10 Rotary encoder

En rotary encoder används för att mäta vinkel och/eller vinkelhastighet och rotationsriktning på en axel. En rotary encoder kan vara antingen absolut eller inkrementell.
Är den absolut har varje vinkel en unik diskret kod och hur många olika diskreta lägen som finns kallas för encoderns upplösning.
En inkrementell rotary encoder ger endast information om vinkeländringen och vinkelhastighet. Ett överordnat system får ta hand om informationen för att avgöra läget på axeln.

2.11 SKF ställdon

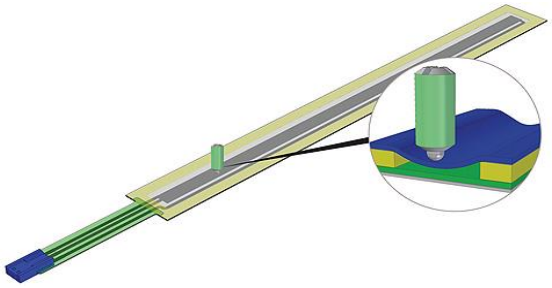
SKF ställdon är en kombination av ett mekaniskt och elektriskt system. Ställdonet tar emot en elektrisk signal och omvandlar denna till en mekanisk linjär rörelse. Styrsignalen kommer ofta från ett styrsystem/reglersystem. Skruven som rör sig linjärt drivs utav en DC motor. Hastigheten av den linjära rörelsen kan variera beroende av stigningen på skruven, matningsspänningen, ställdonets belastning och utväxling mellan motor och skruv.
SKF har en väldigt bredd sortiment av ställdon.
Ställdonen har ofta en feedbacksignal som på något sätt indikerar positionen på skruven, detta används av överordnat system som styr ställdonet. Det kan vara en lägesgivare som säger var i intervallet ställdonet befinner sig. Det finns även ändlägesgivare som bara visar om ställdonet är i något av sina ändlägen.

2.12 Lägesgivare för ställdon

En kort förklaring av hur olika lägesgivare fungerar.

2.12.1 Mekanisk potentiometer

En linjär potentiometer av foiltyp [Figur 2] består av ett membran och en nål. Potentiometerns motstånd beror på var nålen trycker på membranet. Membranet klistras fast i ställdonets chassi. Nålen följer skruvens linjära rörelse och ”släpar” på membranet och på så sätt påverkas motståndet i potentiometern. Denna förändring av resistans används för att ändra spänningen på en analog utsignal. Den analoga utsignalen kan kopplas till ett överordnat system för att på så sätt avläsa ställdonets position.
Som alla mekaniska system påverkas en foilpotentiometer av slitage som kan resultera till en felaktig eller utebliven utsignal med tiden.
Denna lösning används idag på vissa av SKF´s ställdon.



Figur 2 "Regal" potentiometer (Potentiometer, SENSOFOIL® Membrane) [4]

2.12.2 Inkrementell rotary encoder

Då det är en motor som via en växellåda driver ställdonets skruv framåt, är motorns roterande rörelse proportionell med ställdonets linjära rörelse. På så sätt kan man genom att mäta motoraxelns läge veta var ställdonets kolv befinner sig.
På motoraxelns ände sätts en permanentmagnet med lämpligt antal poler som utanpå liggande encoder läser av med hallgivare. Utsignalen blir ett digitalt pulståg med två kanaler, där kanalerna är 90 graders fasförskjutna. Denna signal innehåller bara information om vinkelhastighet och rotationsriktning och överordnat system behöver någon referenspunkt för att kunna bestämma ställdonets position. Denna lösning används idag på vissa av SKF´s ställdon.

2.12.3 **Ändlägesgivare**

Med de båda tidigare sätten vet man var ställdonet befinner sig var än i slaget det är. Ett enklare sätt som är fullt tillräckligt i många tillämpningar är att använda sig av ändlägesgivare. Då fås endast information om ställdonets läge när det befinner sig i sitt bakreläge eller i sitt främre läge. Detta kan åstadkommas med hjälp av en brytare som påverkas mekaniskt direkt av ställdonets skruv.
Denna typ av lägesgivare kan användas för sig självt men också kombineras med de tidigare lägesgivarna för att få extra säkerhet.

3 BESKRIVNING AV ÖNSKAT SYSTEM

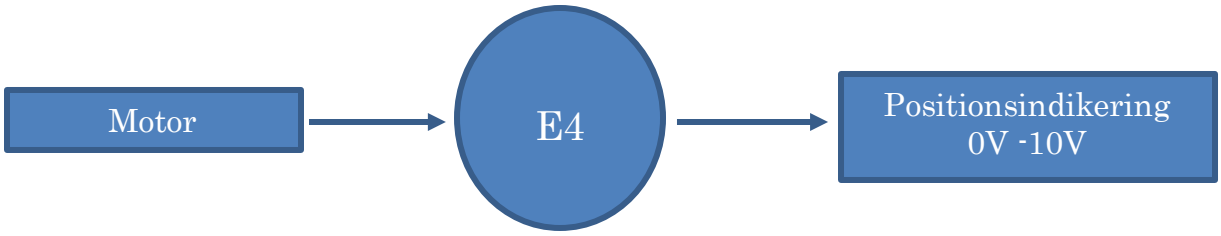
Här presenteras den önskade lägesgivaren och dess krav samt förklaring till hur det skall fungera.

3.1 Krav

Den önskade lägesgivaren av encodertyp är tänkt att ersätta den analoga lägesgivaren av foiltyp. Den nya givaren skall ge motsvarande eller bättre precision.
Utformningen skall vara densamma som de tidigare encodrarna E2 och E3 och läsa av samma permanentmagnet på samma sätt.
Lägesgivaren som detta arbete handlar om ska uppfylla följande krav:

- Lägesgivaren skall vara absolut och ska alltså visa rätt position vid uppstart efter ett spänningsfall.
- Spänningsmatningen ska vara 12 V eller 24 V (±5 %).
- Encodern skall även klara spänningspeakar och dippar.
- Lägesgivaren ska vara monterbar på de ställdonen som idag har en E2 eller E3 encoder.
- Utsignalen ska vara en analog signal (0 V – 10 V) där 0 V visas vid hemmaläge och 10 V visas vid ändläge.
- Kalibrering av ställdonet ska endast behövas vid monterings- och kalibreringsprocessen på SKF och aldrig hos kunden vid en normal användning.
- Slutprodukten ska klara temperaturområdet: -20 °C – +70 °C alternativt -40 °C – +80 °C (se avgränsning).

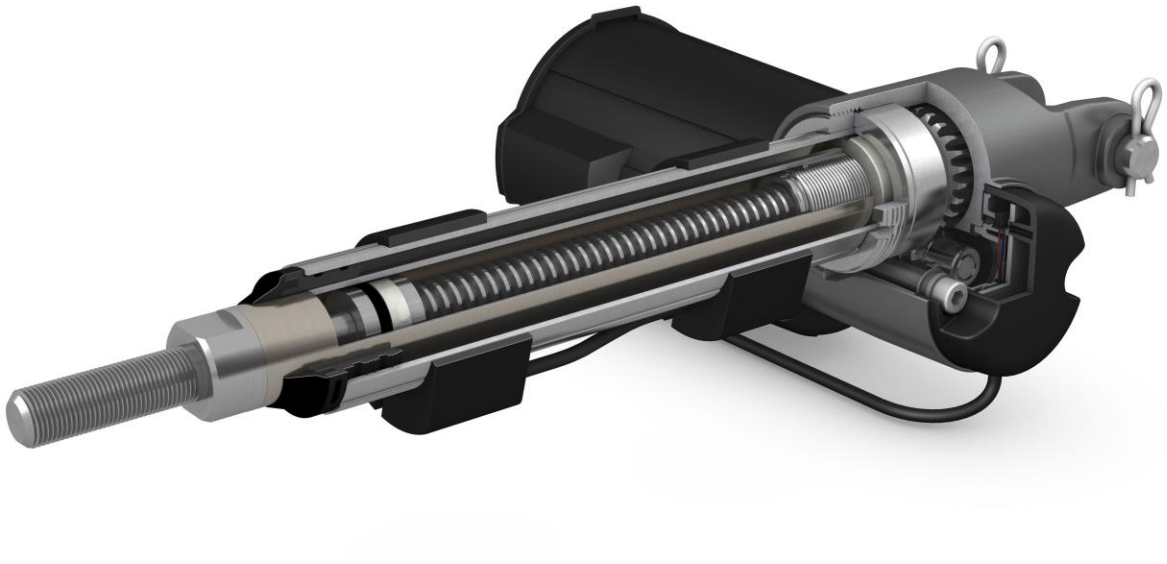
3.2 Konceptbeskrivning



Figur 3 Figuren visar en generell bild av systemet

Den nya lägesgivaren skall ha samma form och använda samma sätt att läsa av positionen som E2 och E3, men utsignalen från den nya encodern skall vara som en analog potentiometer. E4 skall alltså dels ha nya funktioner och samtidigt innehålla samma funktioner som den inkrementella encodern E2.

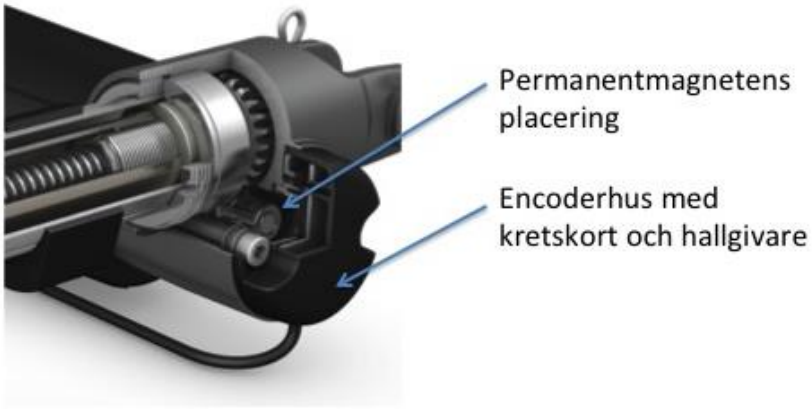
Istället för att pulserna från encodern skickas vidare till ett överordnat system skall behandlingen av pulserna endast ske i encodern. Givaren skall sedan i sin tur skicka vidare en analog utsignal med hjälp av D/A-omvandling. Den analoga signalen används sedan av ett överordnat system för bestämning av positionen på ställdonet.



Figur 4 SKF Ställdon av typen CAT

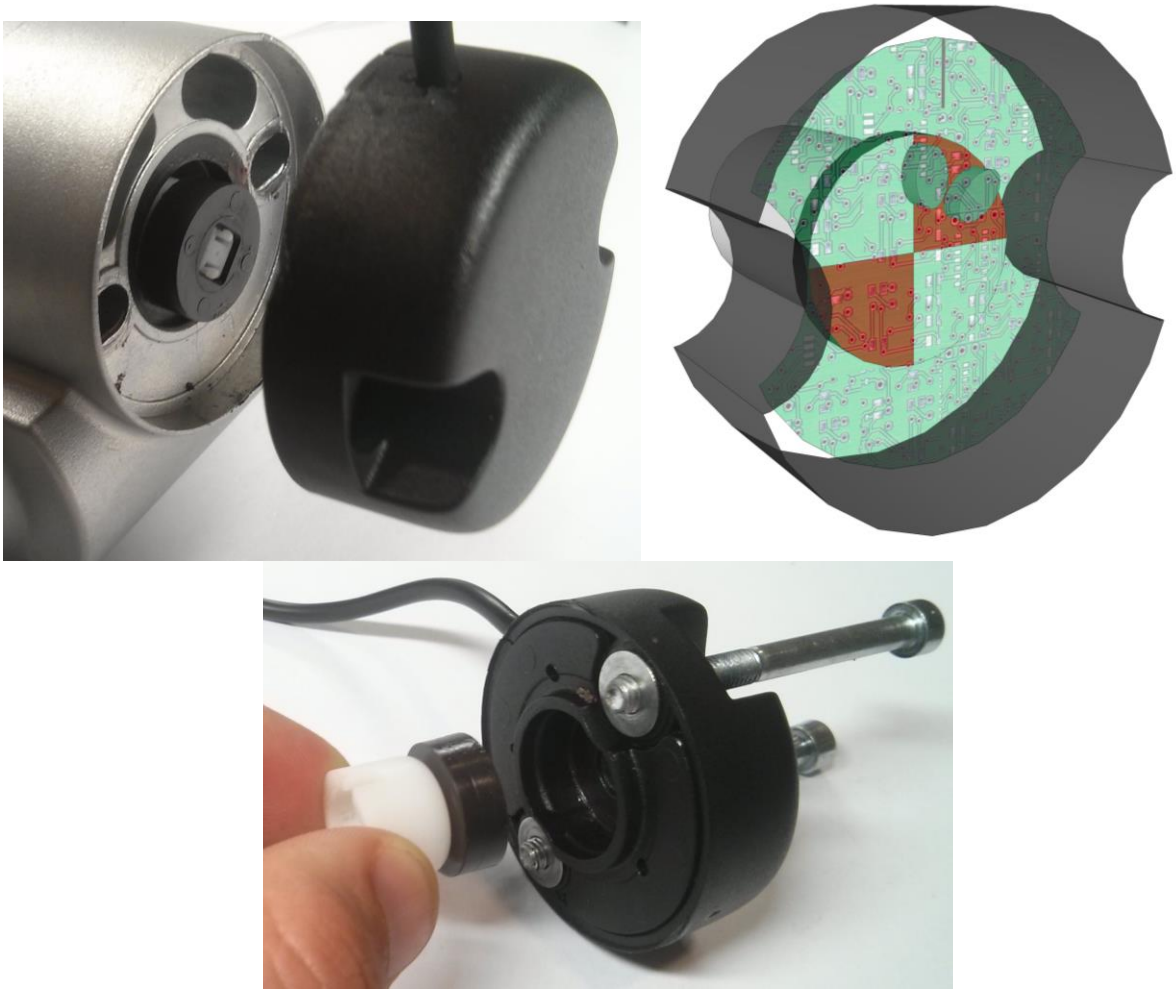
Metoden för att få fram pulserna använder sig av en permanentmagnet och två hallgivare.
På kortsidan på motoraxeln sitter en permanentmagnet med fyra poler, två nord- och två sydpoler. Hur magneten ser ut kan ses i figur 5 och var den sitter monterad kan ses i figur 6 och figur 7.

Figur 5 Permanentmagnet i vekligheten och hur den avbildas i rapporten.



Figur 6 Placering av Encoder och magnet på ställdon.

Utanför magneten sätts encoderhuset fast som ett lock. På kretskortet som sitter i encoderhuset sitter två hallgivare monterade. Givarna är placerade på en plats som gör att de hamnar mitt framför den roterande permanentmagneten.
En skiss över hur givarna sitter i förhållande till magneten kan ses i Figur 7. I figuren är vissa delar av plastkåpan ej utritade. I verkligheten sitter det mellan magneten och kretskortet en plastgavel för att skydda elektroniken mot fett och smuts.

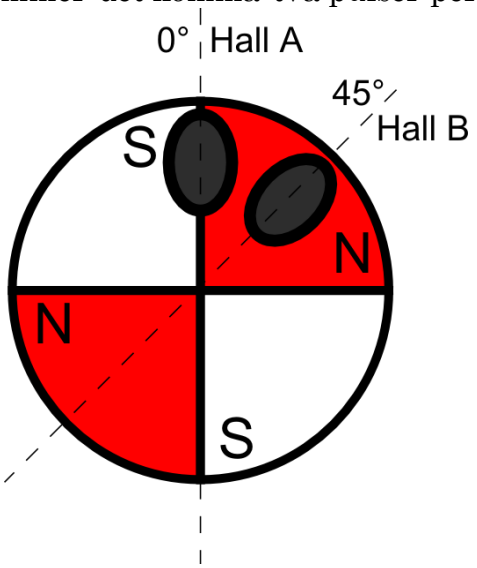


Figur 7 Tre olika vyer av hur magneten sitter i förhållandet till encoder

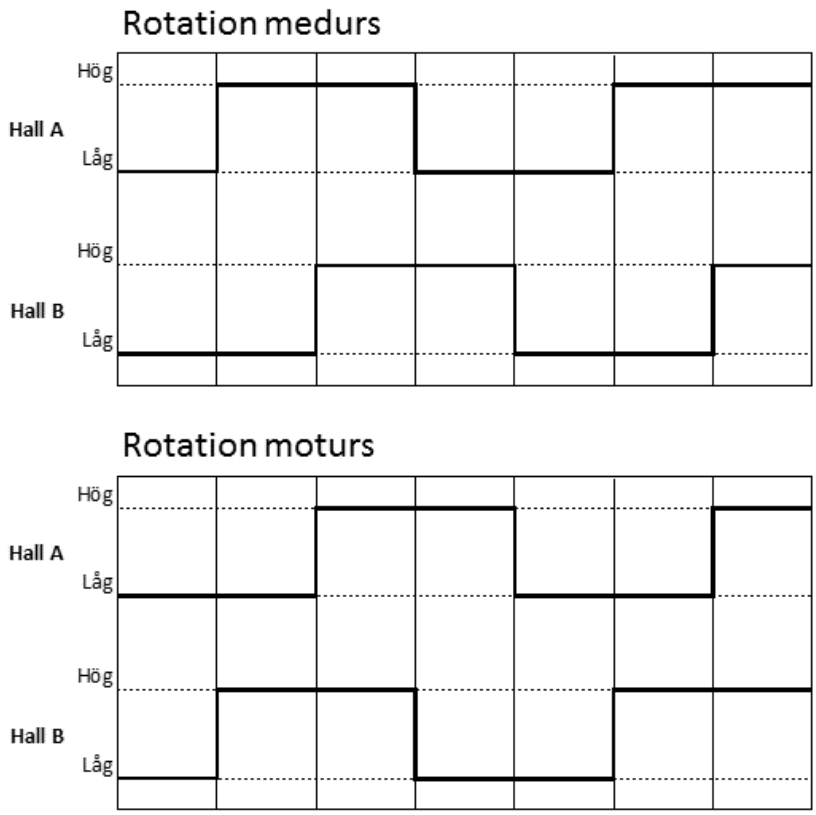
Pulståget

Om hallgivaren ”ser” en nordpol blir utsignalen låg och en sydpol blir den hög. Pulståget som blir när motorn och där med magneten roterar kan ses i figur 9. Genom att lägga hallgivare B 45 grader förskjuten jämfört med hallgivare A fås en 90 graders fasförskjutning mellan pulserna, se Figur 8. Lägg märke till hur vid medurs rotation kommer pulsen först för givare A, medan vid moturs rotation kommer pulsen först för givare B. Detta kan utnyttjas för att bestämma vilket håll ställdonet går åt.

Motorn driver i sin tur skruven så att genom att hålla koll på hur motoraxeln rör sig kan positionen på skruven bestämmas. Maxfrekvensen på pulserna uppskattas till ca 500 Hz i förstudien. Då permanentmagneten har fyra poler kommer det komma två pulser per kanal och varv.



Figur 8 Rätt position av Hallgivarna för att få fram rätt pulståg



Figur 9 Pulståget som genereras vid rotation av magneten

Kalibrering

Då encodern i grund och botten kommer bygga på en inkrementell encoder kommer en nollpunkt att behöva bestämmas, som encodern kan räkna ifrån. Dessutom kan donen variera i slaglängd och utväxling mellan motor och skruv vilket gör att antalet pulser som passerat vid ändläget kommer variera stort. Därför kommer givaren att behöva kalibreras innan leverans.

Eftergångstid

Eftersom encodern skall vara absolut, får den aldrig missa några pulser från magneten. Även om skruven är självhämmande kan ställdonet gå en stund även efter ett spänningsfall. Dessa pulser måste registreras av encodern på något sätt. Därför behövs en strömbackup för att givaren skall kunna registrera pulser från eventuell eftergång på skruven.

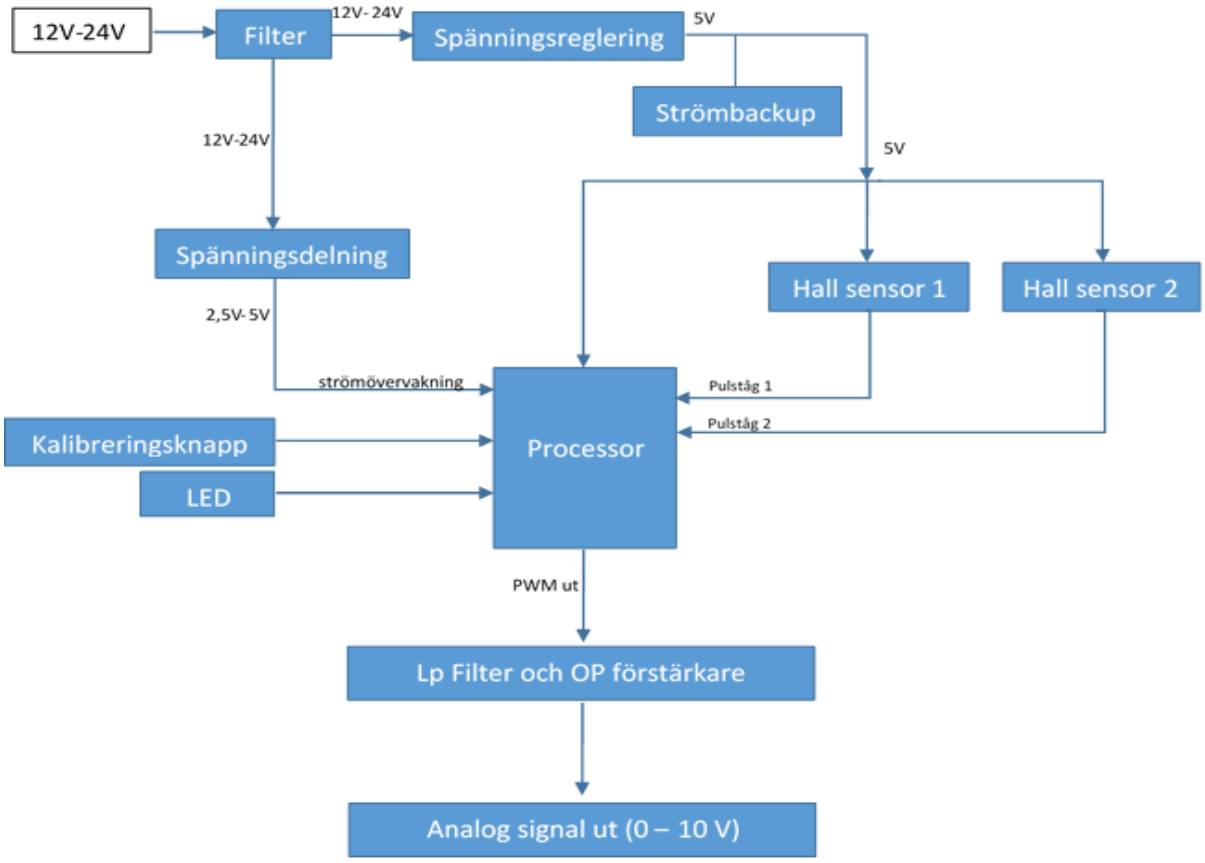
4 METOD

Metoden som används för att lösa uppgiften kan sammanfattas i följande punkter

1. Inläsning av tidigare förstudie gjord av Consat
2. Analys av förstudie samt simuleringar och räkningar som i sin tur bestämmer nästa punkt.
3. Bestämning och beställning av komponenter
4. Testning och verifiering av krets och enkelt mjukvarutest
5. Koda mjukvara
6. Testning av mjukvara
7. Löda ihop ett kort med rätt komponenter och se att det fungerar som det skall
8. Sluttesta med ett riktigt ställdon och mäta upp resultat

5 SAMMANFATTNING AV FÖRSTUDIE

En förstudie gjordes av Consat. Förstudien har haft stort fokus på hårdvara och val av komponenter. Även en tidsuppskattning och kostnadskontroll gjordes. Målet med förstudien är att kontrollera att det är möjligt att designa ett system som uppfyller kraven som SKF satt.
Hårdvaran kan beskrivas enligt följande:



Figur 10 Blockschema som sammanfattar hårdvarukonstruktionen

5.1 Filter

Huvudspänningen ska filtreras först så att det är kontrollerat att inga spänningsdippar eller peakar ska kunna störa eller förstöra resterande komponenterna i kretsen. I förstudie valdes en spole för att dämpa spännings peakar och en kondensator för att dämpa spänningsdippar.

5.2 Spänningsdelning

Processorn ska veta när inmatningsspänningen försvinner. Inmatningsspänningen kan variera mellan 12 V och 24 V och måste därför anpassas för processorn som bara klarar spänningar upp till 5 V. Detta görs genom en enkel spänningsdelning med två motstånd som sedan kopplas till en digital ingång i processorn. Spänningen som spänningsdelas tas efter att inmatningsspänningen filtrerats. En kondensator används även för att filtrera bort störningar.

5.3 Strömbackup

Strömbackupen används för att kunna mata tillräckligt ström på 5 V-bussen för att systemet ska vara igång under en viss tid vid spänningsfall.
En kondensator används för att försörja kretsen och erforderligt värde på den beräknades med hjälp av uppskattad strömförbrukning och uppskattad tid för eftergång på 0.5 s.

5.4 Spänningsreglering

I förstudien användes en spänningsregulator med en referensspänning som krävde en del kringkomponenter. Huvuduppgiften är att omvandla inmatningsspänningen till en stabil 5 V med hög precision. Precisionen i spänningsregleringskomponenterna kommer att påverka den analoga utsignalen.

5.5 Hallgivarna

I förstudien valdes hallgivarna av typen Micronas HAL502SF. De används idag i encoder E2 och E3. Den här typen av givare är redan testade och godkänd av SKF.
Hallgivarna ska vara placerade på framsidan mot magneten och så nära som möjligt. Ingen fysisk kontakt krävs mellan hallgivarna och magnet. De ska också vara förskjutna 45 grader för att kunna generera rätt pulståg. Hallgivarna matas med 5 V spänning som tas efter strömbackupen [Figur 10]. Detta är viktigt för att hallgivarna ska vara igång i en viss tid efter spänningsfall.
Hallgivarna har en inbyggd hysteres, det minskar risken för oscillation där magnetfältet är svagt. Även glapp och vibrationer kan orsaka växling mellan hög och låg Hallgivarna utsignal.
Pulsfrekvensen kan vara som högst ca 500 Hz.

5.6 Processorn

Processorn som föreslogs är PIC12f1822. Kraven som ska den uppfylla är följande:

- ✓ Kunna generera PWM signal med 10 bitars upplösning.
- ✓ Möjlighet att spara data i EEPROM vid spänningsfall.
- ✓ 4st Digitala Ingångar.
  - 2st digitala ingångar för att koppla hallgivarna.
  - 1st digital ingång för strömövervakningen.
  - 1st digital ingång för kalibrerings knapp.
- ✓ 2st Digitala utgångar.
  - 1st digital utgång för PWM generering.
  - 1st digital utgång för LED (används vid kalibrering).

5.7 Metod för D/A-omvandling

I förstudien väljs en processor som kan generera PWM signal med 10 bitars upplösning och med en frekvens som är högre än 1 kHz. 10 bitars ger en signal som har 1024 steg. Varje bit kommer ungefär att motsvara 10 mv i analoga utsignalen. PWM-signalen behöver sedan filtreras i ett LP-filter så att PWM-frekvenserna dämpas medan de låga frekvenserna och DC-värdet ska vara kvar.

5.8 Lågpassfiltret och förstärkning

Förstudien nämner att det behövs ett lågpassfilter för att kunna filtrera bort de oönskade PWM-frekvenserna.
Det behövs också en förstärkning av signalen då den filtrerade utsignalen blir mellan 0 V och 5 V. SKF vill att utsignalen från givaren ska vara mellan 0 V och 10 V. En OP används för att förstärka utsignalen och uppfylla kraven.

6 GENOMFÖRANDE

Här presenteras och förklaras de olika steg samt hur kommer problemet att hanteras och lösas.

6.1 Analys av förstudie/simuleringar/beräkningar

E4 encoder är en produkt som kan ses som utveckling av den tidigare E3 och E2. Fokus i analys och simulering kommer läggas på de delarna som inte är provade innan i E3 och E2.

6.1.1 Filter

Förslaget med en spole och en kondensator från förstudien är samma lösning som användes i E3 och har konstaterats fungerar bra. Inga beräkningar gjordes på denna del av kretsen.

6.1.2 Spänningsdelning

Processorn ska veta när matningsspänningen försvinner och det görs genom att koppla inmatningsspänningen till en digital ingång i PIC 'en. Den digitala ingången är av typen TTL och tolkar en signal som hög om spänningen är mellan 2 V och 5 V. För att läsas som låg skall spänningen ligga mellan 0 V och 0,8 V. (Bilaga för bom och schemaförstudie)
Spänningen in på ingången måste alltså hålla sig inom 2 V - 5 V, men huvudspänningen kan variera mellan 12 V och 24 V. Enligt kraven får huvudspänningen skilja med 5 % så de verkliga värdena på huvudspänningen är 11,4 V och 25,2 V.
Värden på 10 kohm och 2,4 kohm valdes vilket ger följande spänningar:

$$U_{ut} = (\frac{R_2}{R_1+R_2})Uin \qquad \text{Ekv(1)}$$

$$U_{ut@11,4\,V} = \left(\frac{2,4}{2,4+10}\right)11,4\,V = 2,21\,V \qquad \text{Ekv(2)}$$

$$U_{ut@25,2\,V} = \left(\frac{2,4}{2,4+10}\right)25,2\,V = 4,88\,V \qquad \text{Ekv(3)}$$

6.1.3 **Strömbackup**

I förstudien räknades kapacitansen på kondensatorn ut för att kunna hålla en internspänning över 3,5 V i 2 s, alltså en säkerhetsfaktor fyra mot den 0,5 s uppskattade eftergångstiden. Strömförbrukningen på 5 V-sidan uppskattades till 20 mA och tillåtet spänningsfall var 1,5 V. Vilket resulterade i ett kapacitansvärde på 25 mF.

6.1.4 **Hallgivarna**

Förstudien visar ett sätt att koppla hallgivarna och mata dem med 5 volt. Då det är viktigt att hallgivarna sitter i rätt position och med rätt avstånd till permanentmagneten ansågs det lättare att använda sig av E2 för inläsning av pulser i kretsen. Mer om det i rubriken 6.2 Val av komponenter.

6.1.5 **Processorn**

Processorn PIC12f1822 som valdes i förstudien användes i detta projekt. Motiveringen var att antal ben är tillräckligt för de ingångar och utgångar som behövs samt, bra att hållas liten p.g.a. av utrymmesbrist i encoderhöljet.

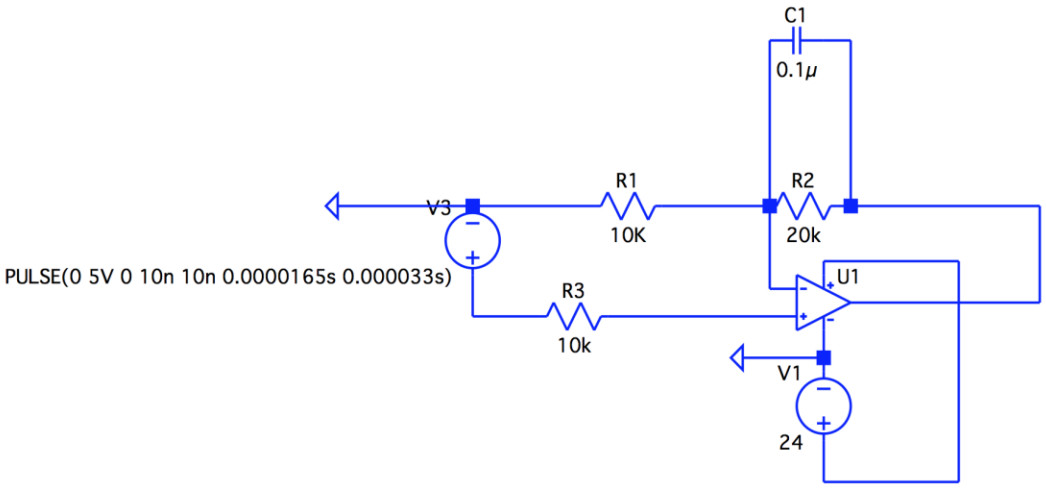
6.1.6 **Metod D/A-omvandling**

Det finns snabbare och bättre sätt än att lågpassfiltrera en PWM-signal för att skapa en analog utsignal. Tillexempel genom att använda en DAC. Denna metod blir dock dyrare och detta fall valdes att inte testas. Istället fortsattes metoden med en lågpassfiltrerad PWM-signal att utvecklas och testas.
För att kunna få en tillräckligt jämn och fin analog utsignal ville en så hög PWM-frekvens som möjligt användas. Därför kollades upp hur hög PWM-frekvens man kunde ha vid 10-bitars upplösning i den valda processorn. Den högsta frekvensen som kunde fås var när 32 MHz klockfrekvens användes, PWM-frekvensen kunde då väljas till 31.125 kHz, vilket ansågs tillräckligt. En tillräckligt låg brytfrekvens får väljas i LP-filtret för att dämpa PWM-frekvensen tillräckligt mycket.

6.1.7 **OP och LP-filter**

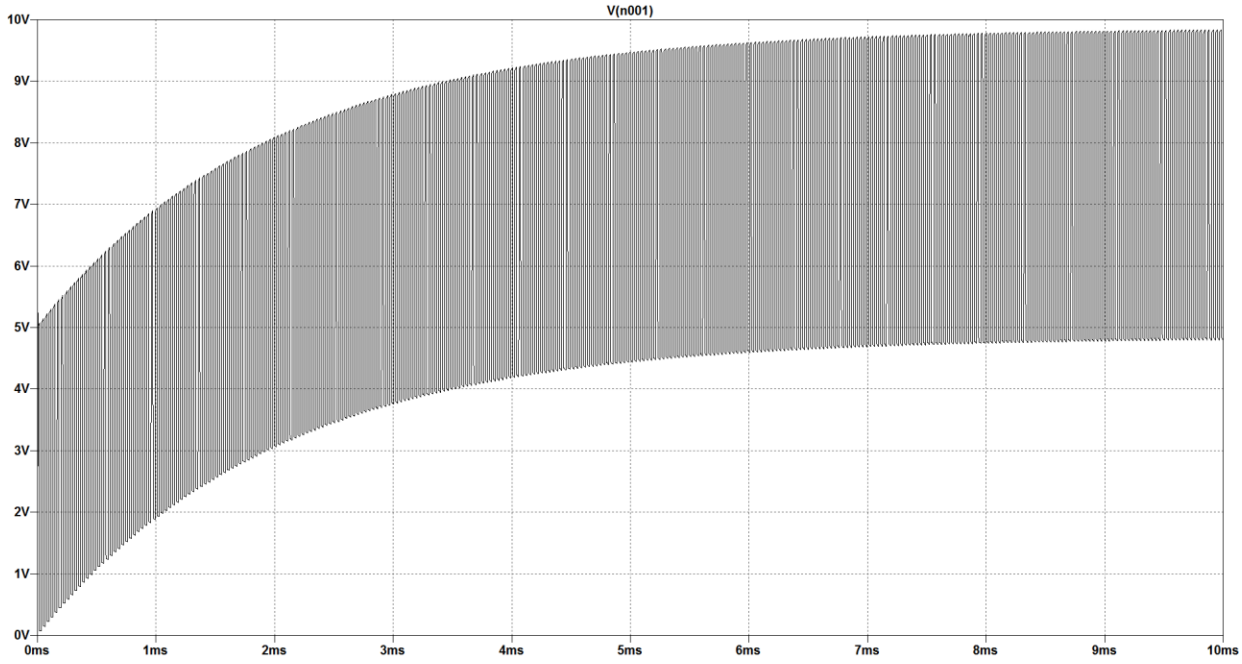
Med hjälp av LT Spice simulerades LP-filtret som var med i förstudien. Målet med denna simulering är att kunna analysera storleken på rippel, fördröjning och förstärkningen av den analoga signalen.

Simuleringen görs genom att skicka ett digitalt pulståg för att simulera en PWM-signal. Pulserna kommer med en frekvens på 31,125 kHz och har en duty cycle på 50 % som skall motsvara 5 V i den analoga utsignalen. OP skall förstärka signalen två gånger.
Simuleringen görs med hjälp av en universell OP som kan skilja sig något mot den riktiga förstärkaren som kommer att användas senare i kretsen.
I figur 11 presenteras kretsschemat.



Figur 11 LT Spice koppling schema för test av OP och filter

Simulering ger följande graf:



Figur 12 Resultat av simuleringen av OP och filtret

Från avläsningen av grafen i figur 12 konstateras att konstruktionen som var i förstudien inte fungerar som det är tänkt. Medelvärdet på utspänningen är ungefär 7,5 V i stället för 5 V som det skall vara. PWM-frekvenserna och dess övertoner dämpas ingenting utan har förstärkningen ett. Orsaken till det är det saknas ett LP-filter och att värdet på komponenterna som valdes för förstärkningen är felaktiga.
För att få ett fungerande filter kan felsökning och ändring av filtret vara ett alternativ men p.g.a. tidsbrist beslutades att ett helt nytt filter tas fram.

6.1.8 **Konstruktionen av nytt filter i FilterPro**

För att ta fram ett nytt filter användes guiden i programmet FilterPro. För att få ett aktivt filter som är enkelt att bygga och valdes filterdesignen till Sallen-key. Sallen-key bygger upp filtret med kaskadkopplade filter av första och andra ordningen. Filtertypen valdes till Butterworth som ger en maximalt jämn amplitudförstärkning av frekvenserna under brytfrekvensen.
En PWM-signal består av DC-värde, PWM-frekvens och en mängd av övertoner med ännu högre frekvenser. Det som vill behållas av denna PWM-signal är främst DC-värdet, alltså medelvärdet av signalen. Alla frekvenser från PWM-frekvensen och uppåt är oönskade och skall dämpas till en acceptabel nivå.
En lämplig dämpning vid PWM-frekvensen räknades ut enligt nedan för att ge programmet den dämpning som behövs för att skapa ett filter.
Då utsignalen är maximalt 10 V och upplösningen på PWM-signalen är ungefär 0,1 % valdes att signalen inte skall skilja mer än ±0,1 % av utsignalens maxvärde 10 V på grund av rippel.
Det motsvarar en utspänning på ±10 mV enligt Ekv(4).

$$10\ V \cdot 0{,}001 = 10\ mV \quad \text{Ekv(4)}$$

Då skall förstärkningen vid 31,125 kHz vara 0,004 gånger enligt Ekv(5) och Ekv(6)

$$\pm 10\ mV = \Delta 20\ mV \quad \text{Ekv(5)}$$

$$\frac{5\ V}{2 \cdot 10\ mV} = 0{,}004\ g\mathring{a}nger \quad \text{Ekv(6)}$$

Programmet vill ha dämpningen i dB och 0,004 gånger är i logaritmisk skala -48 dB enligt Ekv(7).

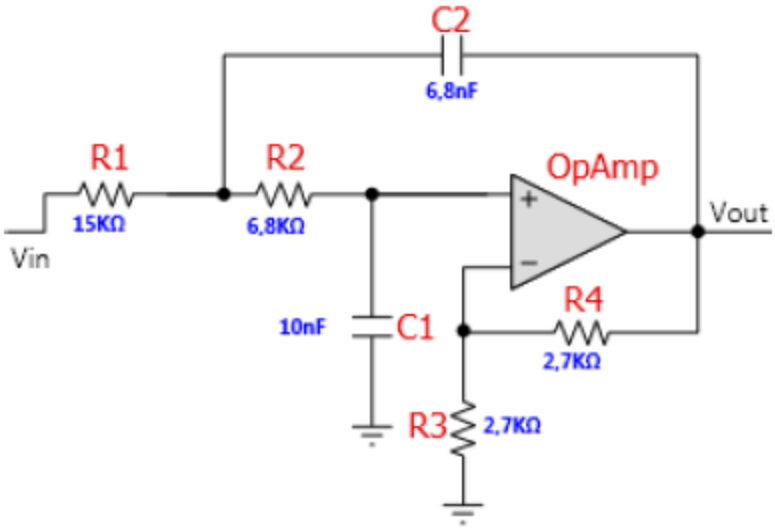
$$20 \cdot \log(0{,}004) = -48\ dB \quad \text{Ekv(7)}$$

Med hjälp av FilterPro framställs ett filter som uppfyller följande filterkrav:

- Dämpa PWM-frekvens 31,125 kHz med -48 dB.
- Två gångers förstärkning av DC-värdet.

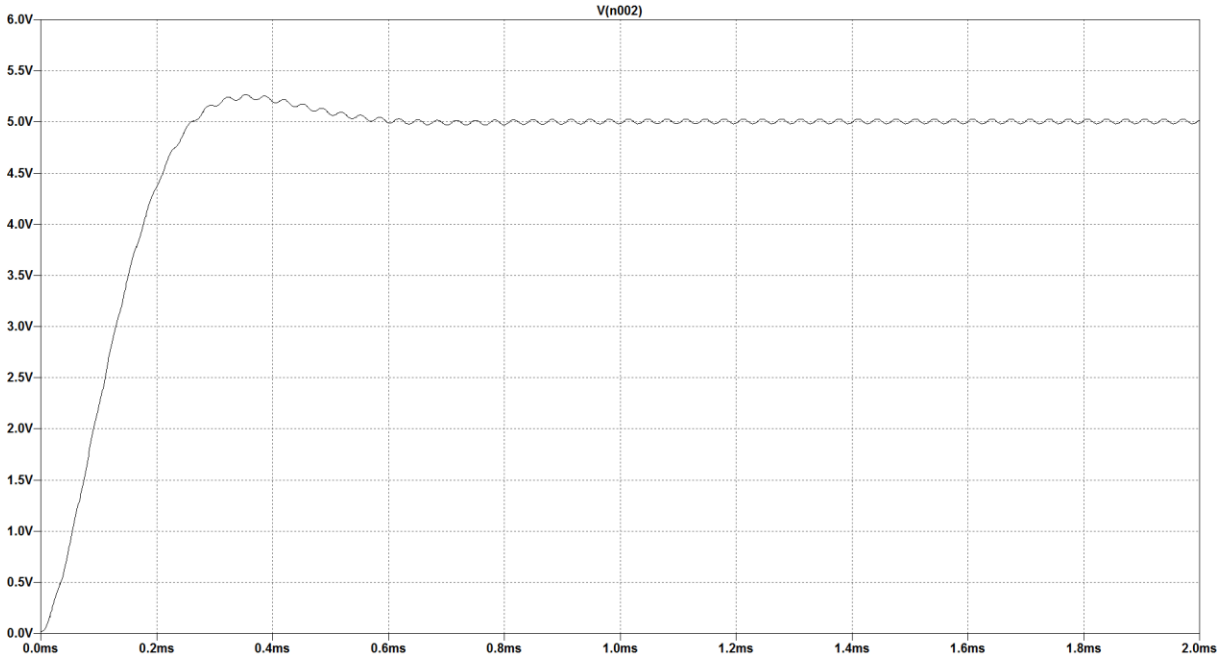
Filter A - Aktivt filter av andra ordningen

Ett filter av andra ordningen togs fram enligt ovanstående inställningar och filtret fick följande värden:



Figur 13 OP och Filer A

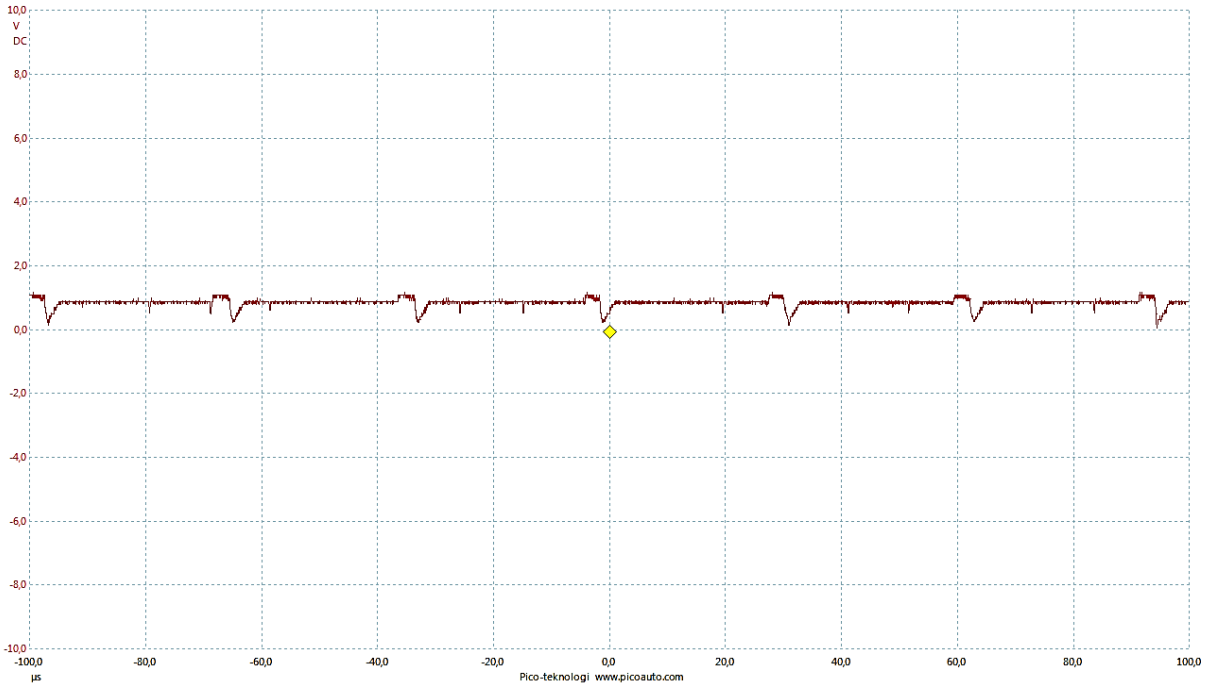
Filtret ritades upp i LT Spice för att kunna simuleras. Grafen i figur 14 visar värden på utgången vid ett stegsvar från 0 % till 50 % av PWM-signalens duty cycle.



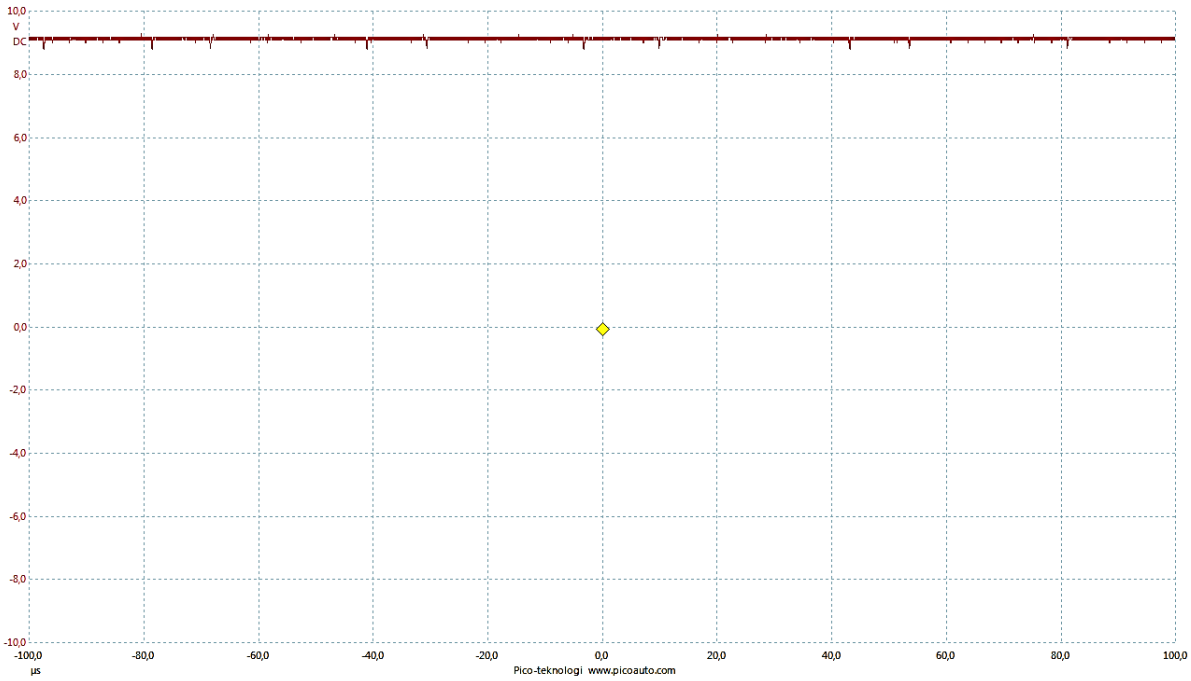
Figur 14 Stegsvar av simulering av filter A

Enligt simuleringen stabiliseras utspänningen vid 5 V och det tar 0.6 ms ungefär, vilket stämmer bra med vad som är önskat. Utsignalen har rippel på ± 27 mV vilket är mer än beräknat.
Eftersom simuleringen görs med universella komponenter byggdes filtret sedan på labbplatta för att testa hur det blir i verkligheten.

En PWM-signal med frekvensen 31,125 kHz genererades med hjälp av PIC:en. PWM-signalen ökades stegvis från 0 % till 100 % samtidigt som signalen analyserades med ett Picoscope.



Figur 15 Utsignalen med medelvärdet 1 V

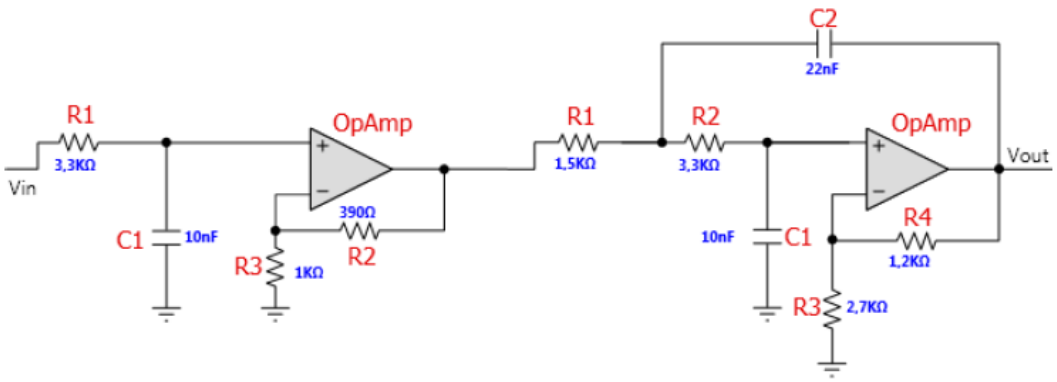


Figur 16 Utsignalen med medelvärdet 9.2 V

I figurerna 15 och 16 visas utsignalerna vid utspänningarna 1 V och vid ca 9.2 V, vid dessa värden var utsignalen som värst och den önskade filtreringen uppnåddes inte. Det finns fortfarande frekvenser inte är tillräckligt dämpade som kommer ut och stör den analoga signalen.
En tänkbar anledning till detta kan tänkas vara att den ofiltrerade PWM-signalen slinker igenom C2 och att OP förstärkaren inte klarar av att reglera bort detta [Figur 13].

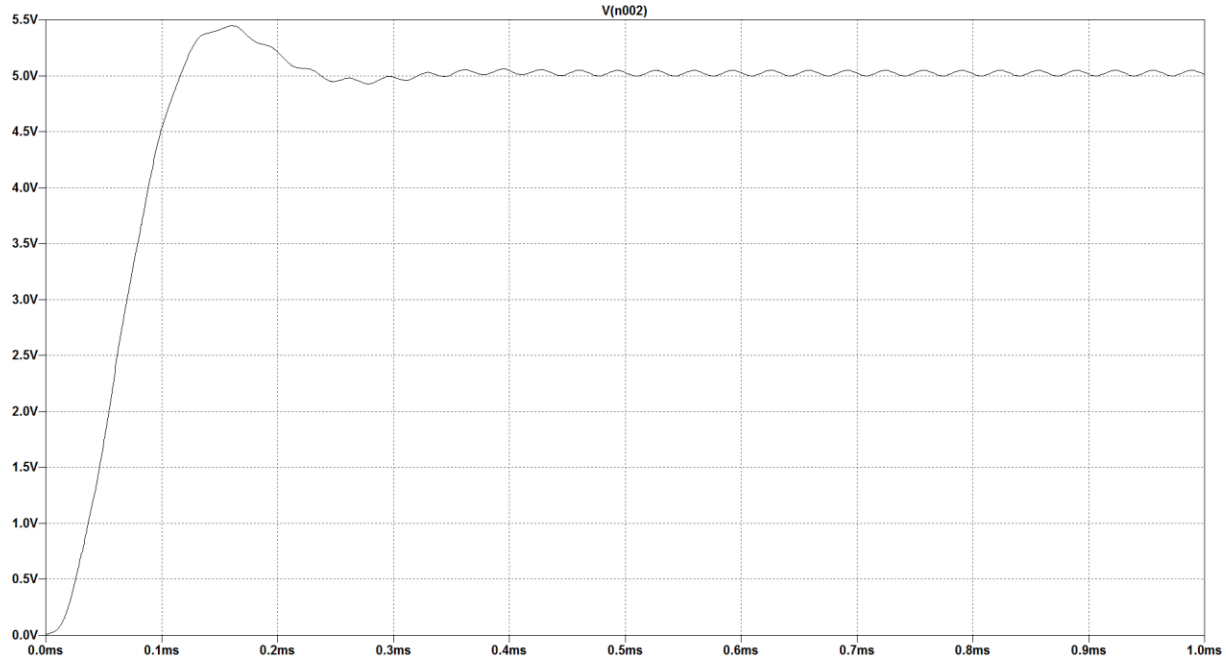
Filter B - Aktivt filter av tredje ordningen

Ett nytt filter togs fram med FilterPro. Problemet med filter A försökte lösas genom att använda ett filter av tredje ordningen istället så att signalen redan är filtrerad en gång innan den kommer fram till C2.
Filtret ska uppfylla filterkraven och målet är att få bättre prestanda än filter A. Även här användes samma inställningar i FilterPro, det vill säga Sallen-key design och Butterworth typ.
I figur 17 visas hur det nya filtret är uppbyggt och vilka värden komponenterna har:



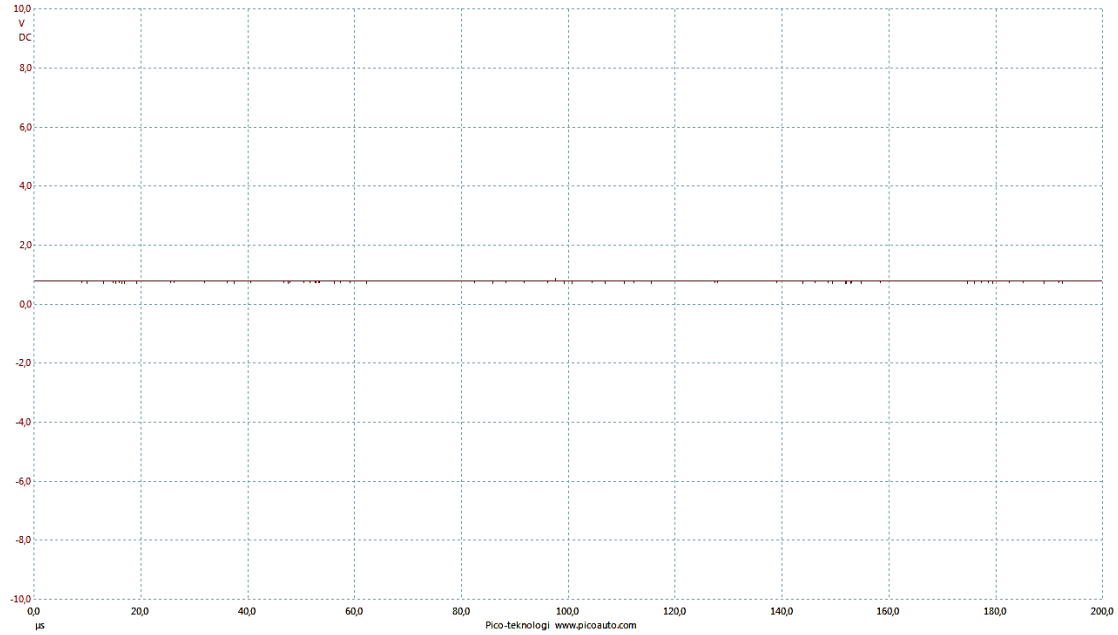
Figur 17 Filter B

En likadan simulering som gjordes för filter A i LT Spice gjordes även för filter B och resultatet kan ses i följande graf:

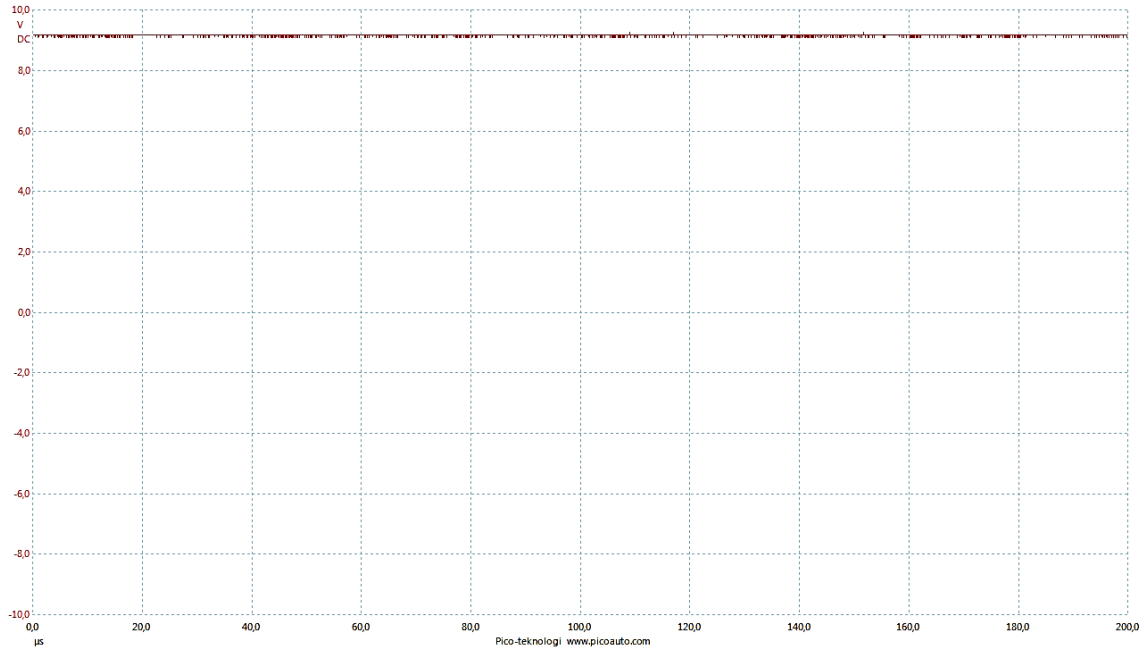


Figur 18 Stegsvar av simulering av filter B

Resultatet av simuleringen för filter B är att systemet blir snabbare än filter A. Dock blir ripplet större i filter B och är nu ±0,048 V.
Åter igen byggdes filtret på en labbplatta och provades på samma sätt som filter A med hjälp av ett Picoscope.
Följande grafer visar det nya filtrets beteende vid 1 V och 9,2 V som var problematiskt för filter A.
Valet av spänningen 1 V och 9,2 V görs för att kunna jämföra utsignalens värde mot värden för filter A.



Figur 19 Picoskop bild av utsignal med medelvärdet 1 V

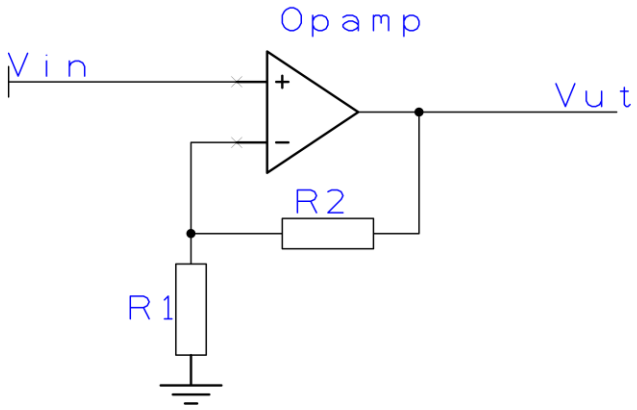


Figur 20 Utsignal med medelvärdet 9.2 V

Filter B uppfyller kraven för uppgiften, dock finns förbättringsmöjligheter som tas upp under nästa rubrik.

- **Förbättringsmöjligheter för Filter B**

De komponenter som styr förstärkningen i en OP-förstärkarkrets är motstånden R1 och R2 som är i figur 21.



Figur 21 Förstärkning med hjälp av OP och motstånd

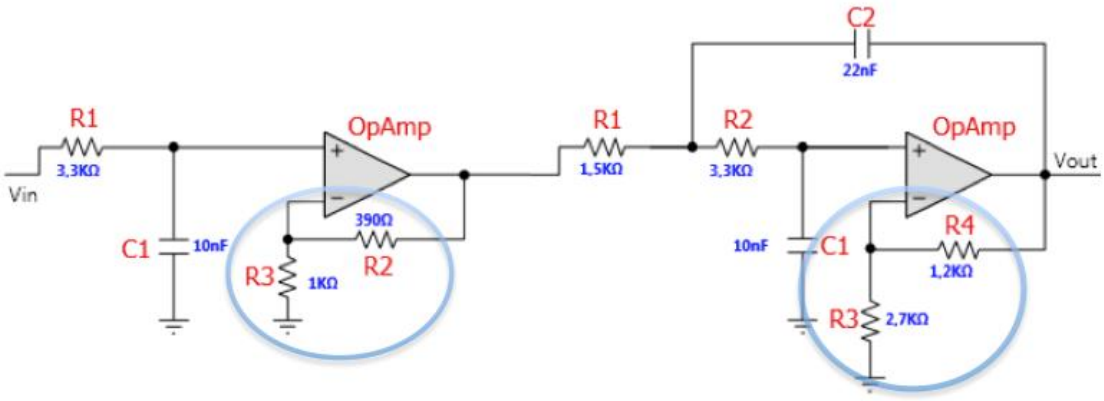
Värdet på R1 och R2 beräknas med hjälp av följande formel:

$$Förstärkning\ (gånger) = 1 + \frac{R_2}{R_1}\quad (gånger) \qquad \text{Ekv (8)}$$

Om förstärkningen 2 vill erhållas mellan Vut/Vin i figur 21 skall då värden på R_1 och R_2 vara samma, enligt ekvation 8 och ekvation 9.

$$1 + \frac{R_2}{R_1} = 2\ gånger$$

$$R_1 = R_2 \qquad \text{Ekv (9)}$$



Figur 22 Filter B

Filter B [Figur 22] består av 2st seriekopplade OP-förstärkare, där signalen filtreras och förstärks vid både första och andra OP-förstärkaren. Varje gång förstärks signalen ungefär 1,414 gånger så att den totala förstärkningen ungefär blir:

$$1{,}414 * 1{,}414 = 1{,}999396\ \text{gånger}$$

Men i filter B blir inte förstärkningarna riktigt 1,414 vid respektive förstärkningsställe utan:

$$1 + \frac{390}{1000} = 1{,}39 \text{ gånger}$$

och

$$1 + \frac{1200}{2700} = 1{,}44 \text{ gånger}$$

dvs.

$$1{,}39 * 1{,}44 = 2{,}0078 \text{ gånger}$$

vilket innebär att förstärkningen skulle bli 2,0078 och inte riktigt 2 även om motstånden skulle ha exakt de värden de är gjorda för.

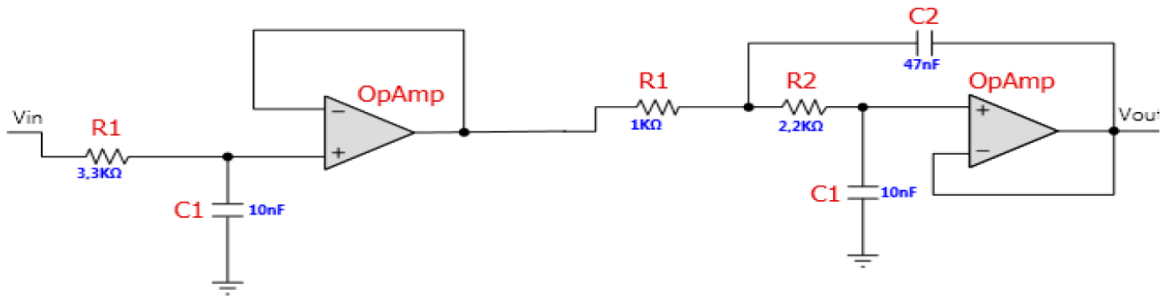
För att få en förstärkning så nära två som möjligt bör de förstärkningsstyrande motstånden vara högprecisionsmotstånd. Genom att endast ha förstärkningen vid den första OP-förstärkaren behövs färre komponenter och dessutom sparas två högprecisionsmotstånd in, vilket innebär en lägre kostnad och sparar plats.

Filter C - Aktivt filter av tredje ordningen med färre komponenter

För att kunna spara plats och komponenter ska ett nytt filter tas fram. Här förklaras de olika steg som gjordes för att få fram filtret.

- **Tredje ordningens filter** med engångsförstärkningen

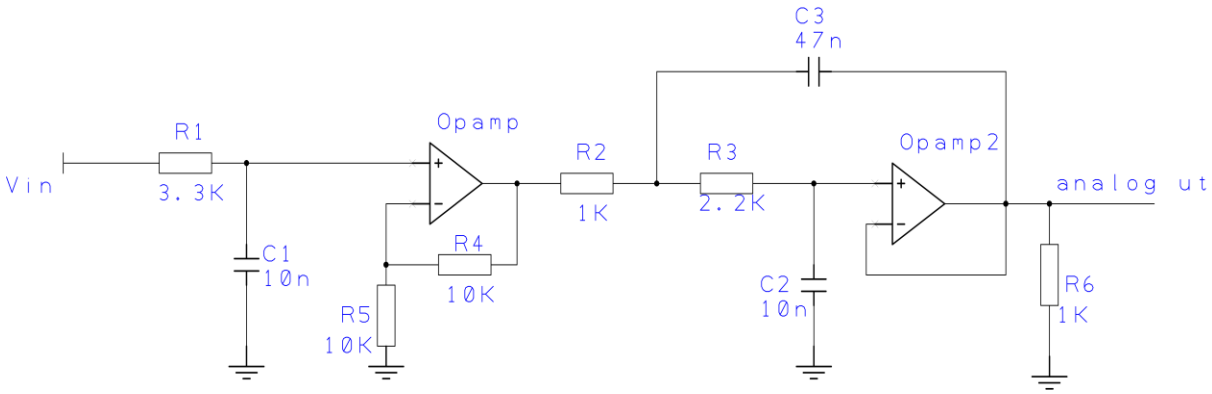
FilterPro ställer automatiskt in var förstärkningen skall ske och i de flesta fall delas förstärkningen upp till att göras på två ställen. Användaren kan inte välja var i kretsen förstärkningen skall ske.
För att kunna få fram den önskade konstruktionen togs ett filter med en gångers förstärkning tas fram med FilterPro med samma inställningar som för filter B.
Nedan kan filtret med en gångers förstärkning ses.



Figur 23 Filter C

Därefter läggs förstärkningen manuellt i kretsen vid den första OP-förstärkaren. En förstärkning skapas som beskrivet tidigare med två motstånd, R_1 och R_2 enligt figur 21. Motstånden väljs till samma värde för att få två gångers förstärkning, värdet valdes till 10 kohm. Komponenterna ska ha 0.1 % noggrannhet.

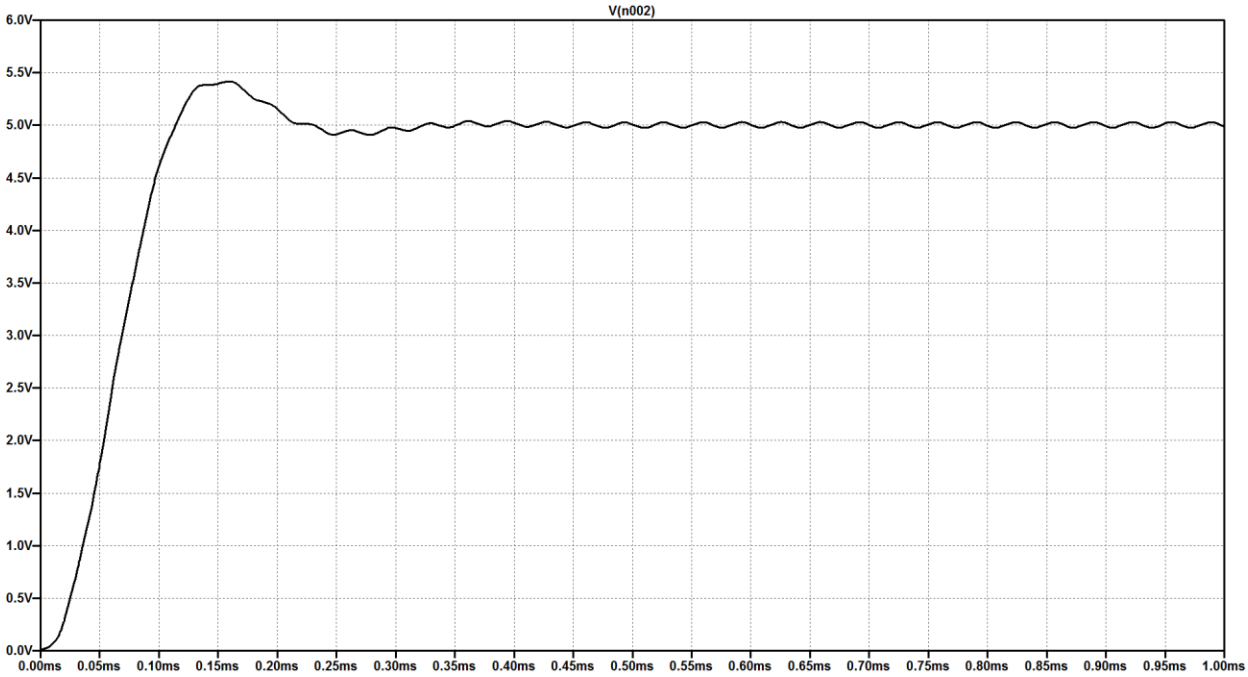
Förstärkningen läggs vid den första OP-förstärkaren, för där påverkar den inte filtrets karaktär.
I figur 24 kan ses hur Filter C, det slutgiltiga filtret ser ut.



Figur 24 Filter C med två gånger förstärkning

Förutom att motstånden R5 och R4 har tillkommit för förstärkningens skull, har även motståndet R6 tillkommit. Anledningen till detta är att när utspänningen från OPamp2 är mycket liten så har utgången svårt att absorbera signalströmmar som kommer genom C3. Utgången beter sig då som en strömgenerator med en källström på 50 uA. Ett motstånd ner till jord bättrar då på OP:ns förmåga att absorbera strömmar som kommer genom C3.

Även filter C simuleras i LT Spice för att kontrollera att det fungerar som det är tänkt. Resultatet av simuleringen presenteras i grafen [Figur 25]



Figur 25 Stegsvar av Filter C

Enligt simuleringen blir:

Stigtid = 0.0975 ms

Rippel = ±30 mV

Utsignalen stabiliseras ungefär vid 0.35 ms

Simuleringen visar att filtreringen och förstärkning fungerar. Inga andra slutsatser dras från simuleringen eftersom den görs med modellerade komponenter. OP och filter ska testas i kretsen med alla andra komponenter. Under kapitel 7 Resultat kan filtrets verkliga värden ses.

6.2 Val av komponenter

BOM-listan som var med i förstudien uppdaterades med de ändringar som gjorts i konstruktionen. Uppdateringen ska omfatta de komponenter som ingår i OP- och filterkonstruktionen. Alla komponenterna som saknas hos leverantören eller som har långa leveranstider ersattes med andra komponenter som har samma egenskaper och uppfyller samma funktioner.

Alla komponenter valdes till hålmonterade komponenter. Fördelar med hålmonterat är att det är lätt att byta ut, enkelt vid felsökning.

I kretsen är det R_4 och R_5 (figur 24) som bestämmer förstärkningen på utsignalen. R_4 och R_5 ska ha 0,1 % noggrannhet för att få en bra noggrannhet på förstärkningen. Resten av komponenter som ingår i designen väljs med 1 % noggrannhets.

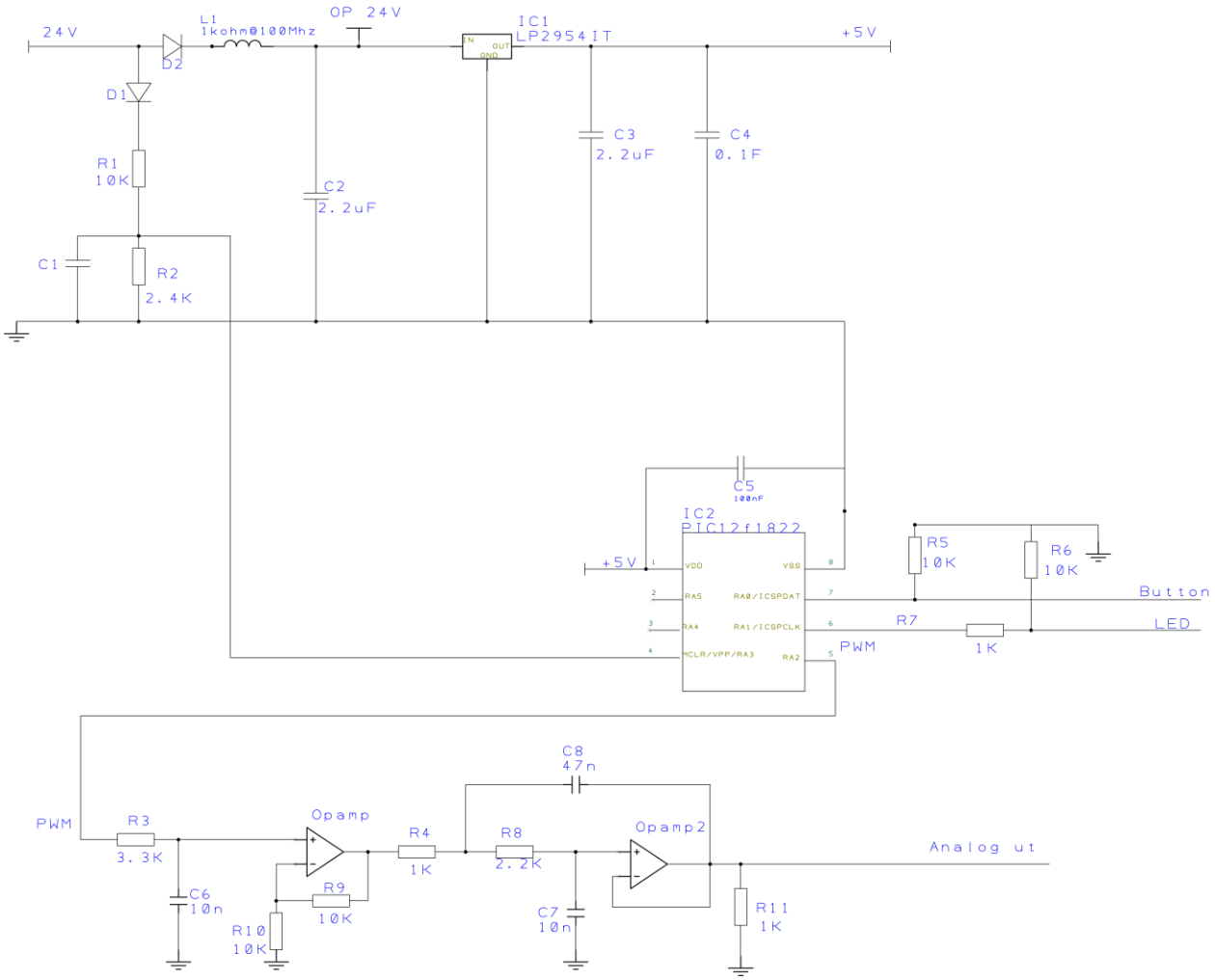
P.g.a. tidsbrist byttes spänningsregulatorn ut till en annan typ eftersom den preliminära regulatorn inte kunde levereras i tid. Den nya regulatorn LP2954IT har 1,2 % noggrannhet.

Enligt förstudien skulle kondensatorn med 25mF användas som strömbackup. Den byttes till en kondensator med 0.1 F istället. Motiveringen är att 25 mF och 0.1 F kondensator ingår i samma grupp, d.v.s. att de har samma storlek och prisskillnaden är väldigt liten. Den största fördelen med detta byte är att vid spänningsfall kan 0.1 F försörja kretsen i betydligt mer tid än 0.25 mF.

Då det är mycket viktigt hur hallgivarna sitter i förhållande till permanentmagneten valdes i detta arbete att använda en E2 encoder för att få fram pulser till testerna. Detta medför att hallgivarna och deras kringkomponenter inte beställdes eller testades.

6.4 Preliminär krets

Schemat presenterar en ny version av kretsen. Den innehåller alla ändringar som gjordes med hjälp av analys och simuleringar. Hallgivarna och deras kringkomponenter är inte med i kretsen.



Figur 26 Preliminär krets

6.5 Mjukvara

Hur mjukvaran utformas är väldigt viktigt för att lägesgivaren alltid skall veta var den är även om strömmen försvinner. Vissa värden skall alltid sparas ner för att finnas tillgängliga vid nästa uppstart. I det här kapitlet tas det upp hur mjukvaran fungerar i stort och hur vissa funktioner är uppbyggda, om koden vill läsas i detalj finns den i Bilaga C.

6.5.1 Utvecklingsmiljö

Mjukvaran för detta projekt är skrivet i Microchips MPLAB X IDE (Integrated Development Environment) som är utvecklingsmiljön för deras mikrokontroller. Programmet har bland annat en inbyggd simulator som kan användas för felsökning och är även bra för att ta reda på hur lång tid en funktion tar att exekvera. Språket som koden är skriven i är C.
För att kompilera koden användes kompilatorn XC8, gratisversionen. Den fungerar för Microchips 8-bitars PIC. XC8 är fri att använda i kommersiellt syfte, skillnaden mellan gratis versionen och Pro versionen är kodoptimeringen men då program inte är så stort kommer inte detta vara något problem.
Under utvecklingen användes versionshanteringsprogrammet Git. Detta möjliggör att enkelt kunna gå tillbaka till tidigare versioner av mjukvaran och samtidigt ha backup på allt man har gjort. Git är ett program med öppen källkod. Git är kommandoradsbaserat och för att enklare kunna använda sig av Git användes det separata GUI-verktyget TortoiseGit.
För överföring av kod till mikroprocessorn användes en Pickit 3 som också görs av Microchip.

6.5.2 Övergripande

Innan programmeringen sattes igång gjordes en konfigurering av processorn och ett enkelt testprogram gjordes för att testa att hårdvaran fungerade som det var tänkt. Utgångar sattes och ingångar lästes. Även tester för att rätt klockfrekvens användes och att PWM-utgången fungerade. Mer om konfigurationen av processorn finns senare i kapitlet.

Val av struktur av programvaran och metod.

Ett tidigare, liknande projekt har gjorts åt SKF Actuation Systems med en likadan mikroprocessor. Kodstrukturen som användes i det projektet efterliknades så mycket det var möjligt. Efter möte med ansvarige för det tidigare projektet gjordes valet av programvaror, men då projektet hade några år på nacken fanns nu nyare men motsvarade versioner.
Pickit 3 ger också möjlighet att i realtid kunna pausa exekveringen av kod i mikroprocessorn och kolla vad som finns i vissa register. Detta konstaterades

inte behöva användas då programmet är ganska simpelt. Felsökning gjordes i princip genom att använda en LED och oscilloskop.
I det tidigare projektet fanns det önskemål från SKF om att inte använda avbrottshantering. I detta projekt gjordes samma val, dels för att efterlikna det förra projektet samt att mjukvaran blir mer lättöverskådlig.

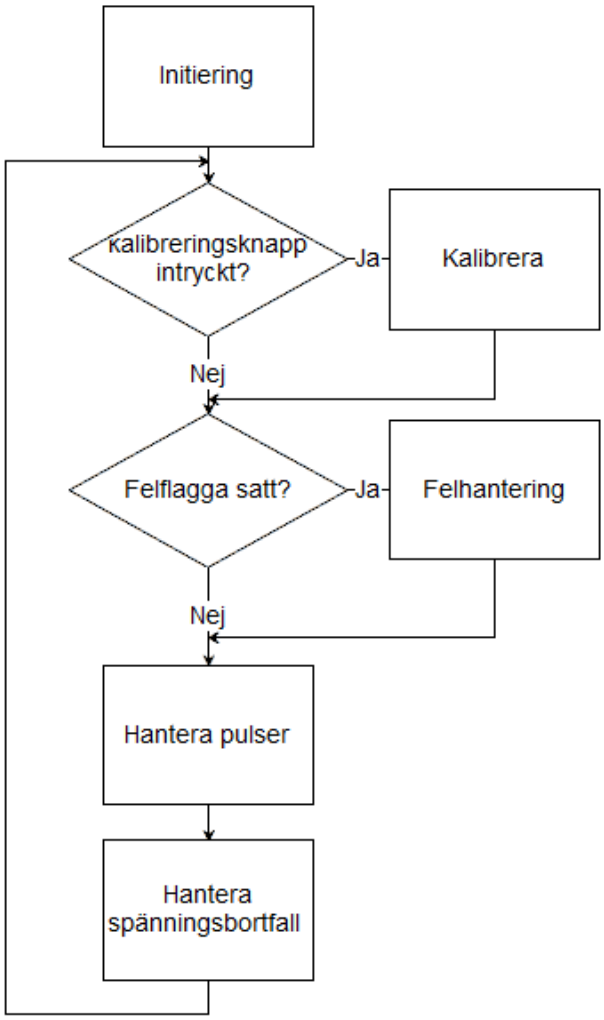
Huvudfunktionerna

För att kunna lösa uppgiften stolpades upp vilka funktioner som var nödvändiga och utvecklades allt eftersom. Dessa togs fram utefter kraven.

- Initiering
- Hantering av pulser
- Felhantering
- Kalibrering
- Skrivning/Läsning EEPROM
- Strömbortfall

6.5.3 Huvudprogram

I figur 27 nedan kan ett flödesschema över mainslingan ses. Viktigt är att ”hanteringen av pulser” alltid sker minst 2000 gånger per sekund alltså med en frekvens av minst 2 kHz. Detta för att inte missa några pulser från givarna. Enligt tidigare uppskattades pulserna komma med 500 Hz, men eftersom att signalen är låg och hög under en pulsperiod måste den kollas med en 1 kHz frekvens. Dessutom är fasförskjutningen 90 grader mellan kanalerna vilket gör att tillståndet kan ändras med en frekvens på 2 kHz.

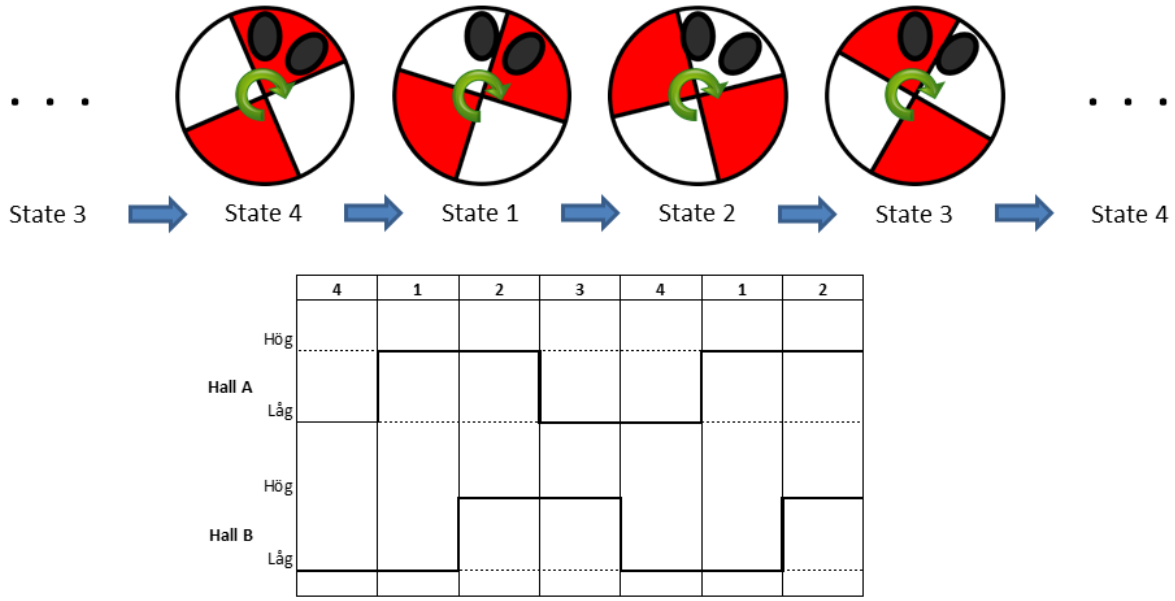


Figur 27 Flödesschemat för huvudprogrammet

6.5.4 **Hantering av pulser**

Pulserna genereras av en permanentmagnet som har fyra poler, två syd och två nord. Då permanentmagneten roterar kan fyra olika tillstånd uppkomma. Dessa fyra tillstånd kommer två gånger per varv, se förklarande figur 28.

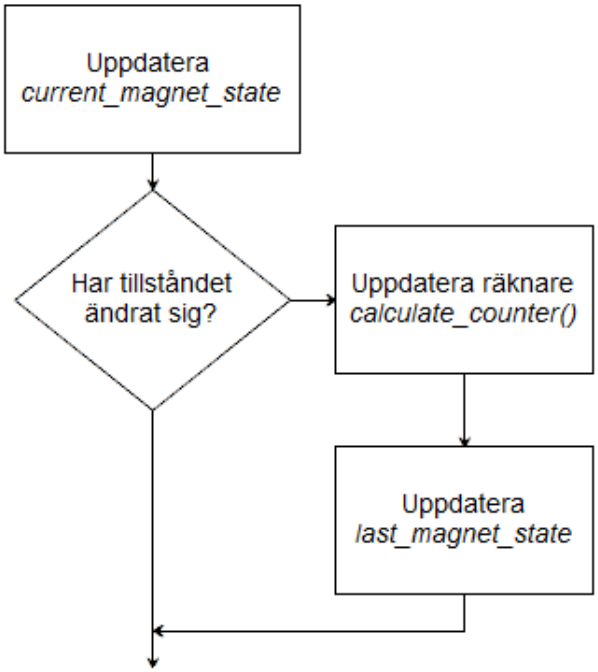
I figur 28 visas hur de olika fyra tillstånd hänger ihop med magnetens läge. Figuren visar hur pulståget förändras när magneten roterar medurs. Roterar magneten moturs kommer de olika tillstånden i omvänd ordning.



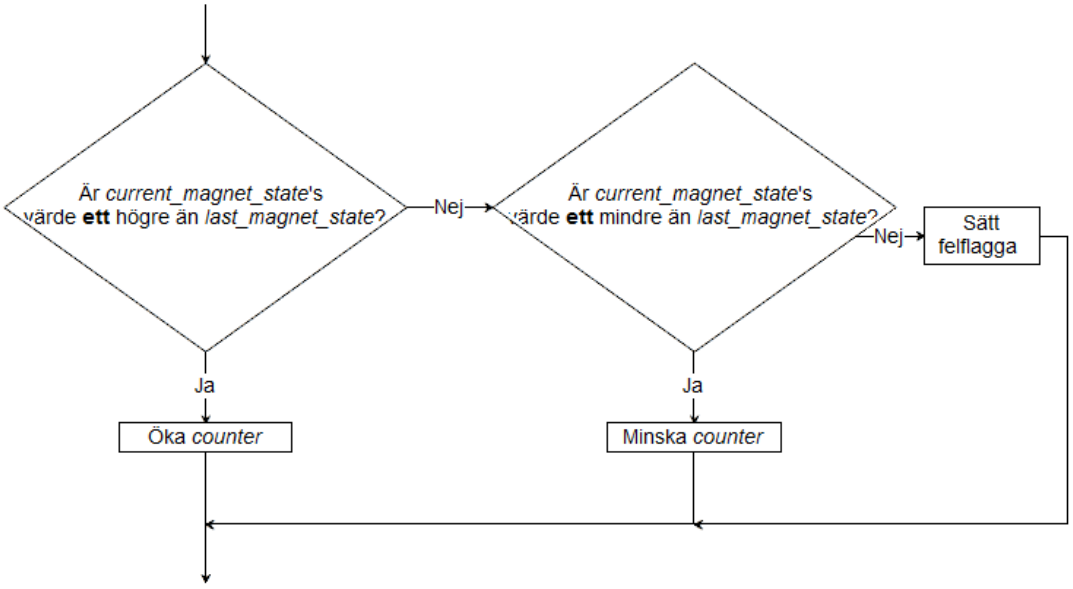
Figur 28 Olika tillstånd med motsvarande pulståg från hallgivare A och B

För att kunna hålla koll på var ställdonet befinner sig relativt en nollpunkt användes två tillståndsvariabler, *current_magnet_state* och *last_magnet_state*. De olika tillstånden gavs olika nummer och beroende på vilken ordning de kommer ökas eller minskas värdet på en absolut räknarvariabel, *counter*. Variabeln *counter* kommer alltså att räkna hur många tillstånd från nollpunkten ställdonet befinner sig.

Funktionerna som tar hand om pulserna heter *manage_movement* och *calculate_counter*. I figuren nedan visas hur funktionen fungerar i stort. Varje gång i main-loopen körs funktionen *manage_movement*, detta sker med minst en frekvens av 2 kHz. Genom att jämföra nuvarande magnettillstånd med föregående vet man om ställdonet har rört sig eller inte. Har ställdonet rört sig, alltså *current_magnet_state* skiljer sig från *last_magnet_state*, anropas funktionen *calculate_counter*. Där ökas eller minskas *counter* beroende på vilket håll magneten har vridit sig. Skulle magneten av någon anledning ha hoppat två tillstånd så sätts en felflagga, *errorflag* för att indikera att en eller flera pulser har missats.



Figur 29 Flödesschema för hantering av tillstånd. manage_movement()



Figur 30 Flödesschema för calculate_counter()

6.5.5 **Kalibrering**

Då ställdonen kan ha olika längd, motor monterad på höger respektive vänster sida kommer värdet på *counter* att kunna variera väldigt mycket. Hemmaläget kommer alltid att vara 0. Däremot kommer ändläget kunna vara dels negativt eller positivt men också kunna variera stort i absolutbelopp. Det längsta ställdonet (700mm slaglängd) tillsammans med den största utväxlingen kan *counter* få ett värde på 11000. Detta är den verkliga upplösningen som ställdonets position kan bestämmas med. För att kunna få ut en analog signal skalas detta värde om till ett 10-bitars tal som i sin tur styr PWM-signalens duty cycle. Ett tal på 10 bitar kan anta värden mellan 0-1023, alltså 1024 olika lägen, detta blir den högsta möjliga upplösningen som kommer att kunna läsas på den analoga utgången ur encodern.
Det fanns inga krav på hur kalibreringsförfarandet skulle se ut, utan endast vad som skulle ske och att en LED och en knapp fanns till förfogande. Därför utformades en enkel sekvens för hur ett kalibreringsförfarande skall se ut.

1. Håll inne knappen tills LED lyser konstant (cirka 2 sekunder)
   Led blinkar initialt när knappen hålls inne.
2. Släpp knapp (Detta blir ditt hemmaläge, *counter* nollställs)
   LED fortsätter att lysa konstant
3. Kör ställdonet till önskat ändläge
4. Tryck på knappen, led startar att blinka
5. Släpp knappen, LED slocknar, värden sparas och kalibreringen är klar

Anledningen till att man behöver hålla inne knappen I två sekunder är att man inte skall komma in i kalibreringsläget av misstag. På så sett förhindras att skriva över gammal kalibreringsdata av misstag.

Det som sparas undan till EEPROM är *counter*-värdet vid ändläge, alltså det högsta värdet counter kan ha.
Där efter körs funktionen *calculate_PWM_factor* som används för att räkna ut skalfaktorn *PWM-factor*. Denna skalfaktor används sedan för att skicka rätt duty cycle till PWM-utgången.

Funktionen *calculate_PWM_factor* använder maxvärdet för counter och räknar ut en skalfaktor. Notera att vid negativt värde på *counter* blir också *PWM_factor* negativ.

$$PWM\_factor = \frac{1023}{\max\_value\_counter}$$

6.5.6 **Felhantering**

Målet är att encodern aldrig skall missa några pulser eller räkna fel. Mjukvaran kan på vissa ställen se om det är någonting som är fel, och sätter då en felflagga som är en global variabel *errorflag*.

De olika typer av fel som kan indikeras är:

WDT

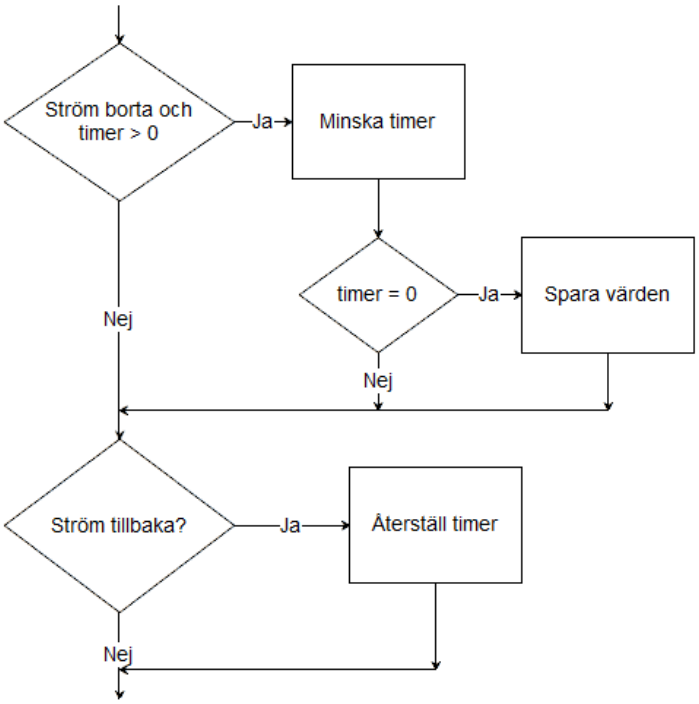
Om processorn skulle hänga sig, sker en reset av processorn. Om en reset har skett p.g.a. WDT indikeras detta av PIC:en genom att sätta en statusbit låg vid nästa uppstart. Denna bit kollas vid initieringen och är den låg sätts *errorflag*.

Felläsning

Om läsningen hoppar ett över ett tillstånd kommer felflaggan att sättas.
Om föregående tillstånd, *last_magenet_state* är 2 och nuvarande, *current_magnet_state* är 4, då har något gått snett och encoderns *counter* stämmer inte längre. Man kan bara hoppa ett tillstånd åt gången.

6.5.7 Hantering av spänningsbortfall

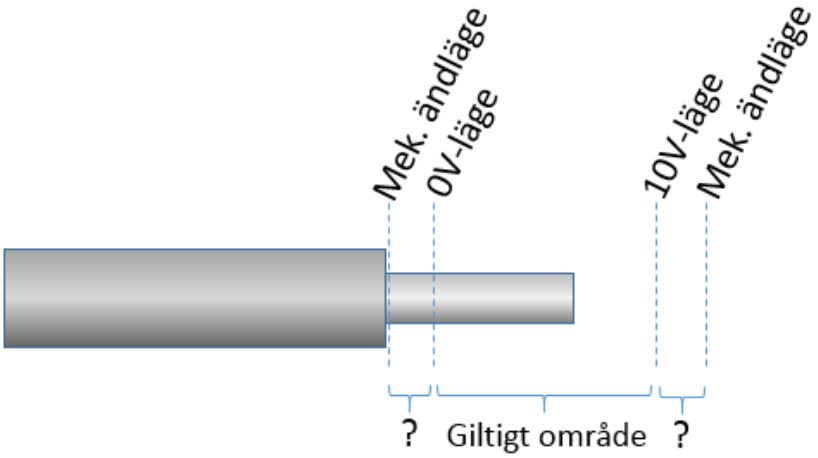
Encodern matas med en spänning på 12V-24V, om den spänningen försvinner har kretsen en strömbackup på 5V-sidan som klarar att driva sensorerna och mikroprocessorn en stund till för en eventuell eftergång på ställdonet. Tiden mellan att strömmen försvinner och att programmet sparar undan värdena är 4 sekunder. Tiden 4 sekunder valdes för att en kondensator på 0,1F valdes till labbkortet, detta gav möjlighet till en längre eftergångstid än de 0,5s som nämndes i förstudien.
Allt hantering av spänningsbortfall sköts av funktionen *mangage_powerloss*. Tiden styrs av en timer-variabel som heter *powerloss_timer.*
Om strömmen är borta minskas *powerloss_timer* varje gång funktionen körs. Om strömmen kommer tillbaka inom 4 sekunder återställs timern och allt fungerar som vanligt. Hinner däremot 4 sekunder gå kommer funktionen att spara undan aktuella värden till EEPROM för följande variabler: *last_magnet_state, counter* och *errorflag*.
När skrivningen till EEPROM är klar kommer processorn att fortsätta köra som vanligt.
Antingen kommer strömmen tillbaka, då fortsätter encodern precis som att ingenting har hänt och timern i *manage_powerloss* återställs.
Om strömmen däremot inte kommer tillbaka kommer processorn att så småningom stängas av utav en BOR. När strömmen sedan kommer tillbaka efter en BOR kommer de gamla värden läsas in i initieringen som vanligt.



Figur 31 Flödesschema för hantering av spänningsfall. Mangage_powerloss()

6.5.8 PWM-utgång

För att styra vad som skall skickas ut på PWM-utgången används ett par funktioner. Den ena, *calculate_PWM_factor* används vid initieringen för att räkna ut *PWM_factor* och den andra, *manage_pwm_output* används för att översätta värdet på *counter* till motsvarande duty cycle.

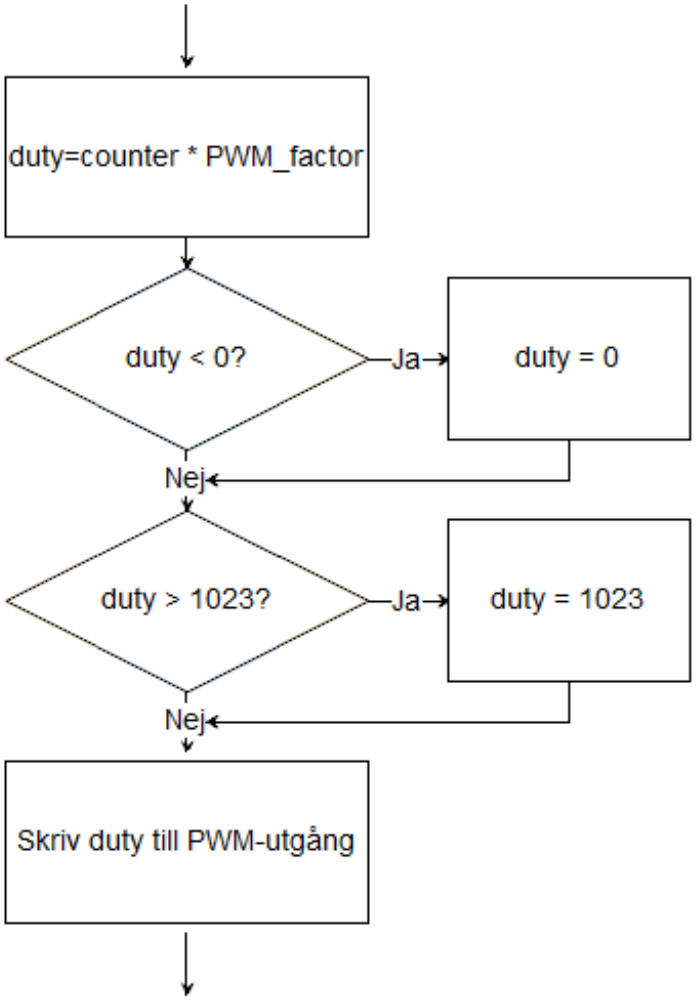


Figur 32 Ändläge på ställdon

Utanför kalibrerat område

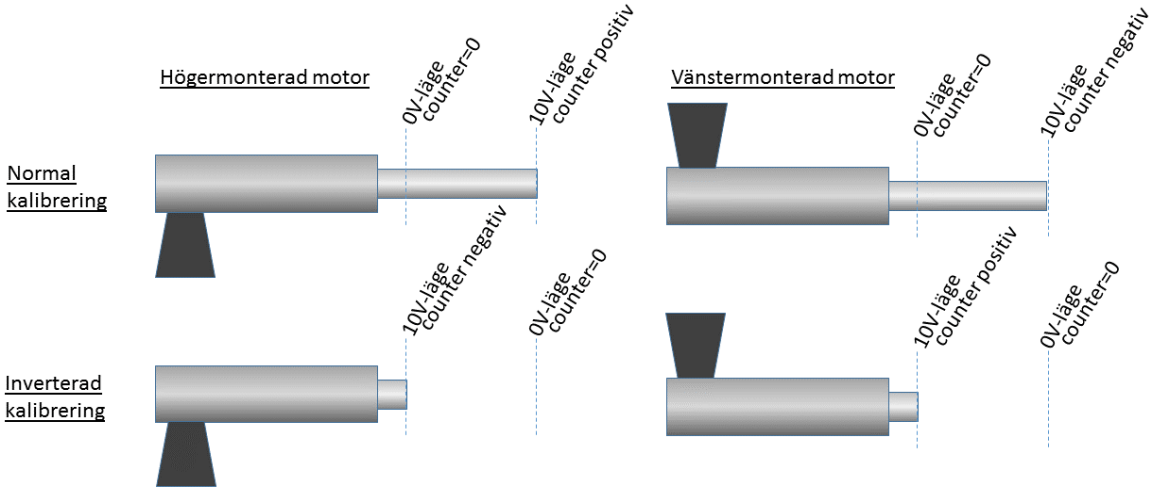
Av olika anledningar kommer ställdonet kunna hamna utanför sitt område det är kalibrerat för. Frågan om vad som skall visas på utgången om ställdonet går förbi noll eller ändläget uppkom. I kravet fanns inget önskemål om vad som

skulle hända. Lösningen som valdes var att fortsätta visa 0 V om man befann sig på 0 V-sidan, respektive 10 V om man befann sig på 10 V-sidan.



Figur 33 Hantering av PWM signal, Manage_PWM_output()

Hur löses problemet med höger/vänstermonterad motor?



Figur 34 Illustration hur det ser ut med Höger/vänstermonterad motor

Ställdonet finns i två olika utföranden, antingen roterar magneten medurs när ställdonet körs utåt, eller i det andra utförandet roterar magneten moturs. Detta bestäms av vilken sida av ställdonet motorn är monterad på. Detta kommer påverka om *counter* är positiv eller negativ vid sitt ändläge efter kalibreringen.
I båda fallen skall 10 V visas på den analoga utgången när ställdonet är i sitt ändläge.
Detta har lösts genom att tillåta *PWM_factor* bli negativ när den räknas ut.
När sedan skalningen görs för att skriva *duty_cycle* blir resultatet positivt då både *counter* och *PWM_factor* är negativa.

6.5.9 Konfiguration av processor

Här tas övriga funktioner upp som finns inbyggda i PIC:en som aktiverats och konfigurerats. Konfigurationen av processorn gjordes enligt följande:

Konfigurationsregister

I XC-kompilatorn används ”pragma config statements” för att göra dessa inställningar. Dessa kan enkelt skapas med Configuration Bits-verktyget som hittas i MPLAB X under Window→PIC Memory Views→Configuration Bits. Inställningarna som önskas väljs och ut får man färdiggenererad källkod. Denna kod lades sedan i filen *Config_bitar.c* som inkluderades i projektet.
Nedan visas de viktigaste av inställningarna som gjorts, resterande inställningar är valda till default-värden.

```
// CONFIG1
#pragma config FOSC = INTOSC      // Oscillator Selection (INTOSC oscillator: I/O function on CLKIN pin)
#pragma config WDTE = ON        // Watchdog Timer Enable (WDT enabled)
#pragma config MCLRE = OFF        // MCLR Pin Function Select (MCLR/VPP pin function is digital input)
#pragma config BOREN = ON       // Brown-out Reset Enable (Brown-out Reset enabled)
#pragma config CLKOUTEN = OFF    // Clock Out Enable (CLKOUT function is disabled. I/O or oscillator function on the CLKOUT pin)

// CONFIG2
#pragma config PLLEN = ON       // PLL Enable (4x PLL enabled)
#pragma config BORV = HI            // Brown-out Reset Voltage Selection (Brown-out Reset Voltage (Vbor), high trip point selected.)
```

Klockfrekvens

Som nämnt tidigare vill en så hög PWM-frekvens som möjligt användas, därför valdes klockfrekvensen till den högsta möjliga 32 MHz. Detta fås genom att använda den interna oscillatorns 8 MHz signal som multipliceras med 4 genom att använda processorns PLL-funktion.
För att välja intern oscillator och PLL görs val i Config-registren.
För att välja intern klockfrekvens 8 MHz behöver följande två registerbitar ställas in:

```
OSCCONbits.IRCF  = 0b1110;      //Set to the 8 MHz HFINTOSC
OSCCONbits.SCS   = 0b00;        //Clock determined by FOSC in Config1
```

Brown out timer

Om spänningsmatningen till processorn understiger en viss spänningsnivå kommer processorn att utföra en BOR för att processorn inte skall exekvera kod utanför sitt giltiga arbetsområde. Det finns två spänningsnivåer att välja på för när reset skall ske. Vid användning av PLL måste den högre av nivåerna användas. Detta konfigureras i Config-registren enligt tidigare.

Watch dog timer

Om programmet skulle hänga sig i en oönskad loop används en WDT.
Den fungerar så att en timer nollställs hela tiden i programmet, skulle denna timer inte nollställas på den konfigurerade timer-tiden så utför processorn en WDT-reset vilket resulterar i att felflaggan tänds. På detta sätt kommer användaren informeras om ett fel av denna typ inträffat.

Master clear enable

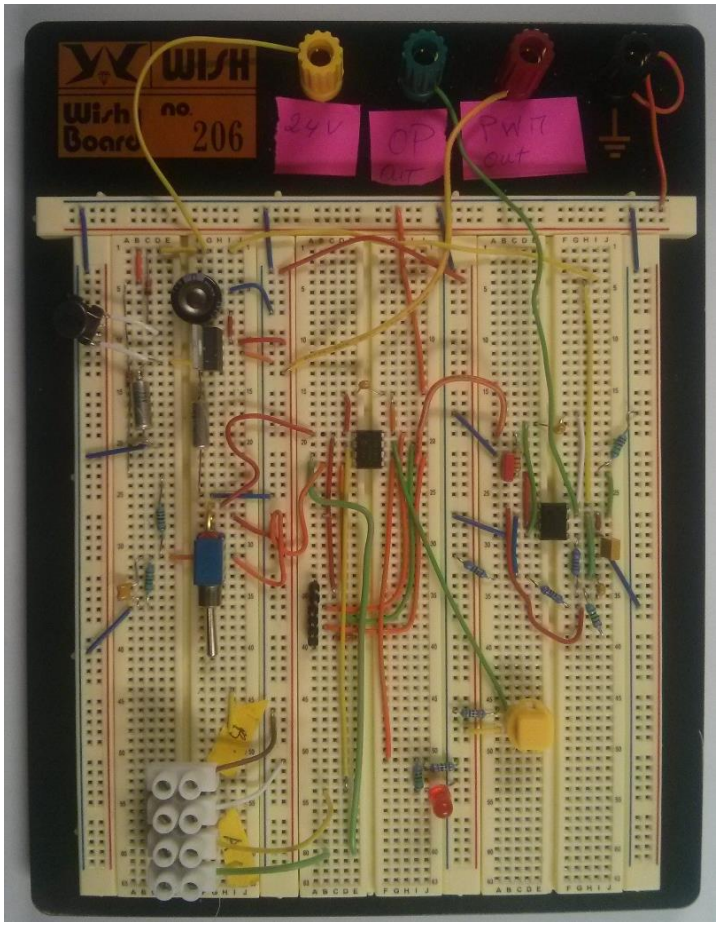
Eftersom alla pinnar kommer att användas som I/O behöver funktionen MCLRE stängas av. Pinnen blir då en ingång som kan användas. Detta görs enligt ovanstående konfiguration i register *CONFIG1*.

6.6 Realisering och tester

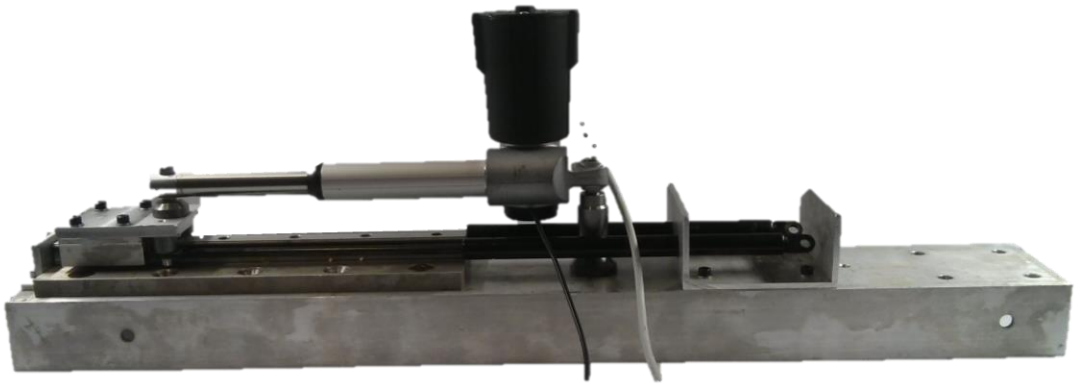
Här förklaras hur testningen av kretsen genomförts.

6.6.1 Kopplingsdäcket

I ett första steg byggdes kretsen på ett kopplingsdäck för att enkelt kunna felsöka och göra mätningar. Att bygga på ett kopplingsdäck gör också att det är väldigt lätt att byta komponenter om det behövs. Hur kopplingsdäcket ser ut kan ses i figur 35.
Kretsen kopplades enligt den preliminära kretsschemaritningen under rubrik 6.3 Preliminär krets.
För att generera ett pulståg användes ett ställdon av typen CAT med en monterad E2 encoder, allt sittandes på en testrigg. Detta kan ses i figur 36.



Figur 35 Kopplingsdäck med alla komponenter



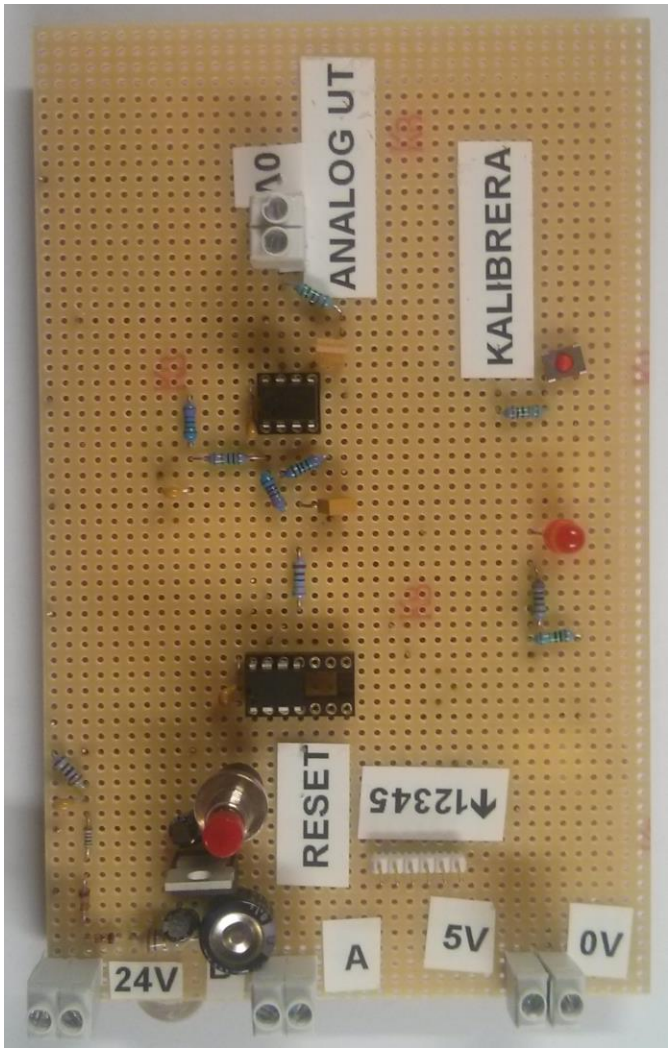
Figur 36 Ställdon CAT i rigg med monterad E2-encoder

I kretsen på kopplingsdäcket kontrollerades:

- Att spänningsdelningen levererar 2 V - 5 V till kretsen vid de olika inspänningarna.
- Att spänningsregleringen ger 5 volt ut.
- Att strömbackupen kan försörja kretsen vid spänningsfall.
- Att PIC:ens I/O inställningar stämmer med det som är i mjukvaran.
- Att förstärkningen och filtret fungerar enligt uppsatta krav.
- Att kalibreringen fungerar som den skall.
- Att grundfunktioner i mjukvaran fungerade, så som spara/läsa till EEPROM.

6.6.2 **Labbkort**

När kretsen på kopplingsdäcket fungerade och komponenterna var bestämda, löddes de hålmonterade komponenterna fast på ett laborationskretskort som kan ses i figur 37. Målet med detta kretskort var att kunna ha en hållbar krets med fastlödda komponenter så att ingenting skulle riskera att lossna.



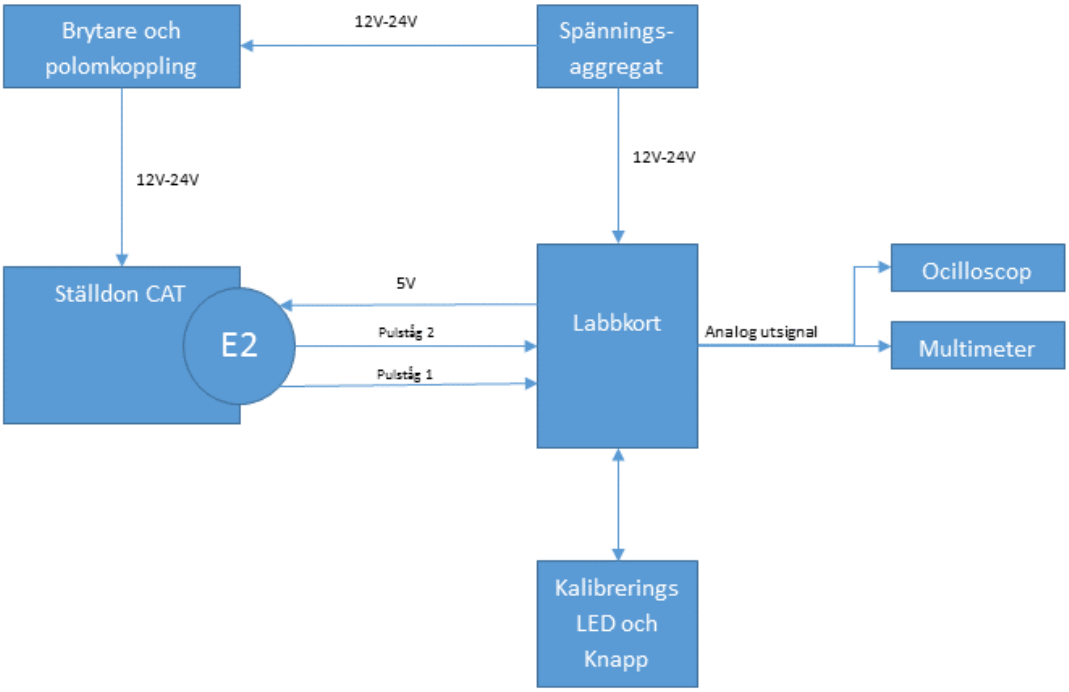
Figur 37 Labbkortet användes vid utveckling och testning av mjukvaran

Labbkortet användes vid utveckling och testning av mjukvaran.
För att förenkla felsökning och utvecklingen lades det till delar på kretsen som inte skall sitta på den slutgiltiga kretsen.
Tillagda funktioner för utvecklingssyfte är:

- En resetknapp - för att starta om processorn, detta eftersom att bryta huvudspänningen till kortet inte bryter internspänningen då den har strömbackup.
- LED och tryckknapp - För att enkelt kunna kalibrera givaren utan en extern kalibreringskontroll.
- Stiftlist - För att programmera processorn med mjukvara sattes även en stiftlist för att kunna koppla in programmeraren Pickit 3.

6.6.3 Testsystem med Labbkort

Labbkortet användes dels för test av mjukvara men också för att kunna testa så att hela systemet fungerar ihop. Dessutom användes Labbkortet för att ta de mätvärden som presenteras under resultatdelen. Dessa mätvärden jämfördes med resultatet av de tidigare gjorda simuleringarna.
Testerna av hela systemet gjordes genom att använda ett ställdon av typen CAT monterat i en testrigg. På ställdonet satt en E2-encoder som genererar ett pulståg vid rörelse. Kretsen matades med ett spänningsaggregat som levererar 12 V / 24 V och E2 matades med 5 V från kretsens internspänning.
För att inte fysiskt behöva trycka på labbkortet för mycket byggdes även en fjärrkontroll för att kunna trycka på kalibreringsknappen samt att kunna läsa av LED, denna kan ses i figur 39. Figur 38 förklarar hur kretsen kopplas till ställdon och oscilloskop för att kunna analysera utsignalen.



Figur 38 Illustrering av testkoppling

Figur 39 Kalibreringsfjärrkontroll

6.6.4 Genomförda tester

Tester som gjordes på hela testsystemet var följande:

Stegsvar för analog utgång

Ett likadant stegsvar som i LT Spice simulerades för filter C gjordes också på den verkliga kretsen för att verifiera att filtret håller sig till de uppsatta kraven. Testet utfördes genom att göra ett steg på PWM-utgången från en duty cycle på 0 % till 50 %. Stegsvaret användes för att kunna analysera signalens stabilitet, snabbhet, rippelsstorlek.

Rippel för analog utgång

Genom stega igenom alla 1024 möjliga PWM-värden och titta på den analoga utsignalen i oscilloskopet kunde storleken på signalens rippel avgöras. Oscilloskopets funktion som mäter signalens peak to peak användes för att mäta spänningsskillnaden mellan den lägsta respektive högsta spänningsnivån på signalen.

Max och min för analog utgång

Kravet var att utsignalen skulle vara mellan 0 V-10 V därför togs mätvärden vid 0 % PWM-värde och vid 100 %. Mätningarna gjordes med hjälp av oscilloskopet och multimeter.

Givarens linjäritet

Även utsignalens linjäritet testades genom att registrera hur utsignalen ändras när PWM-värdet ökas stegvis från 0 till maxvärdet som är 1023 med steg om 50.

Livslängd vid strömavbrott

I detta test har tiden uppmäts för hur lång tid det tar för internspänningen att gå från 5 V till 2,8 V.
I de olika testerna har laddningstiden för kondensatorn varierats.

Klara av strömbortfall

För att testa att encodern inte glömmer bort sin senaste position vid en omstart av processorn gjordes många olika tester för att få encodern att komma bort sig. Detta gjordes genom att dra ut 24 V -kabeln ur spänningsaggregatet vid olika tillfällen och sedan starta igång encodern igen för att se så att det fortfarande visade rätt position. Att rätt position fortfarande visades gjordes genom att mäta med skjutmått på ställdonet.

Klara längre pass av körning utan att räkna fel

Ställdonet har kalibrerats i olika slaglängder som mätts upp med skjutmått och körts fram och tillbaka flertalet gånger för att sedan mätas med skjutmått igen.

Klara av både höger- och vänstermonterad motor

Alla tester har provats på ett ställdon med högermonterad motor. För att testa så att allt fungerar med även med en vänstermonterad motor har inverterad kalibrering använts. Hur detta går till illustreras i figur 34. Det gör genom att platserna för hemmaläge och ändläge byter plats vid kalibreringen. Då snurrar permanentmagneten åt det andra hållet när ställdonet kör mot ändläget och från encoderns sida upplevs detta precis som att det är ett ställdon med vänstermonterad motor.

Klara av att köras med hög pulstågsfrekvens (400 Hz)

Genom att köra ett olastat ställdon med högt motorvarvtal testades att encodern kunde ta emot pulser med hög frekvens utan att räkna fel.
Pulståget från ställdonet mätte 400 Hz i pulsfrekvens vilket motsvarar ett motorvarvtal på 12000 rpm. Motorn kördes både framåt och bakåt under testet.

7 RESULTAT

Här sammanfattas resultat som fås av testerna.

7.1 Begränsning i hur givaren kan användas

Då laddningen av kondensatorn har blivit en flaskhals i konstruktionen har en användningsbegränsning lagts till.

Användningsbegränsningen lyder:
Ändring av position får endast ske då givaren är och har varit inkopplad till huvudspänningen i minst 1 minut, samt under utsatt tid för eftergång.

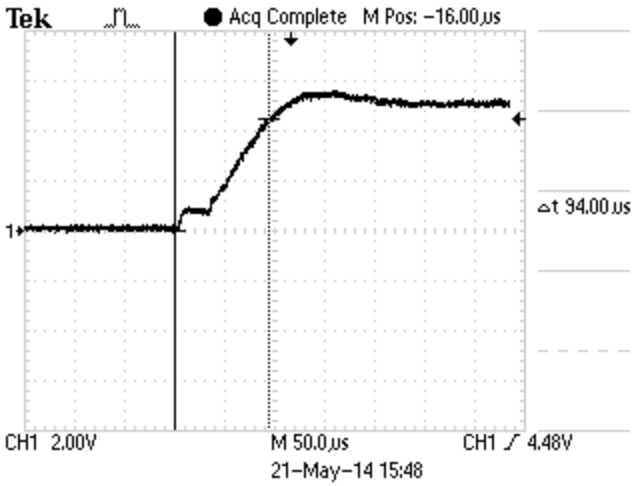
Om positionen ändras och givaren används innan 1 minut har gått kommer givaren inte att klara ett eventuellt spänningsbortfall. Mer om detta tas upp i diskussionsdelen.

7.2 Testresultat på framtagen hårdvara

Här presenteras resultat av stegsvar och resultat av tester som gjordes med ställdon.

Stegsvar för analog utgång

Det slutgiltiga filtret blev filter C enligt kapitel 6.1.8.
Följande graf visar resultatet för det verkliga stegsvaret:



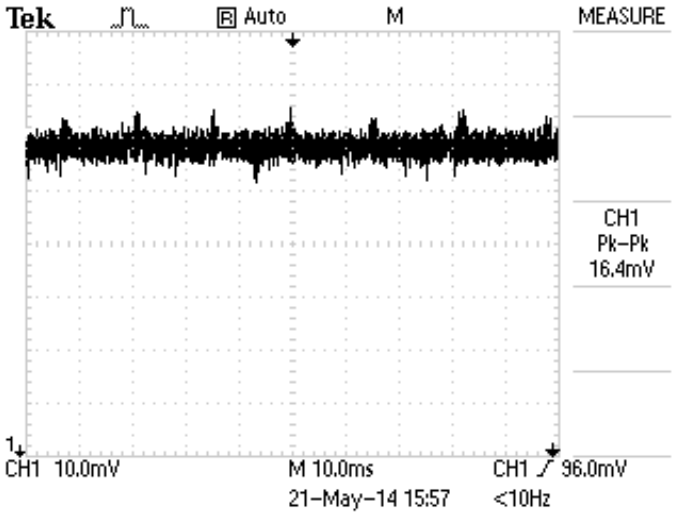
Figur 40 Stegsvar. Från 0 % till 50 % duty cycle på filter C.

Enligt grafen är:

- Stigtiden är: 0,094 ms.
- Signalen stabiliseras efter ungefär 0,25 ms.
- Vid 0,7 V stannar utsignalens spänningsnivå i cirka 30 us, detta tas upp i diskussionen.

Rippel för analog utgång

För att läsa av storlek på signalens rippel zoomades utsignalen in för att få en bättre upplösning och därmed få ett noggrant värde. Resultatet presenteras i följande graf:



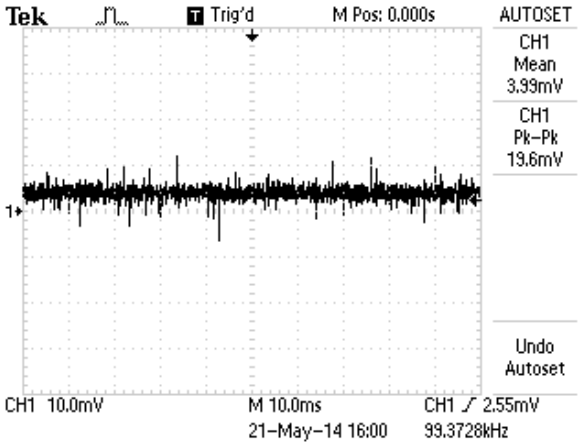
Figur 41 Analys av utsignalens rippel

Enligt grafen är:

- Peak to peak för utsignalens rippel är 16,4 mV, alltså ± 8,2 mV.
- Frekvensen på de högsta topparna i signalen är ca 50-100 Hz.

Max och min för analog utgång

Figur 42 nedan visar den analoga utsignalen vid 0 % i duty cycle, alltså den lägsta utspänningen som kan visas av givaren. Detta sker när ställdonet befinner sig i sitt hemmaläge.

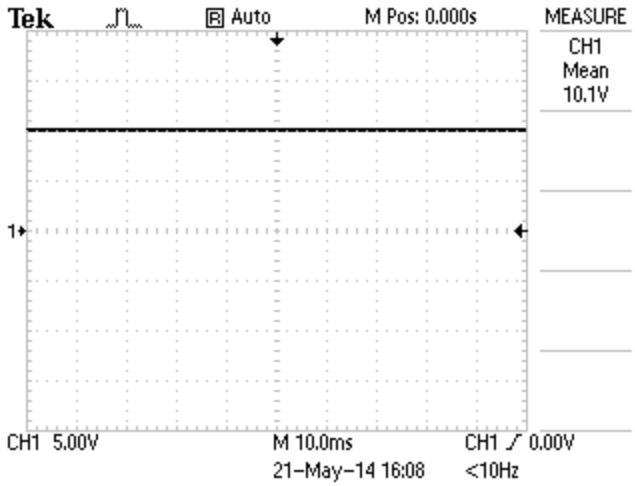


Figur 42 Analog utsignal vid 0 % i duty cycle

Enligt grafen är:

- Signalens spänning när ställdonet står i sitt hemmaläge är ca 4 mV.
- Vid nolläget ökar utsignalens rippel något och dess, ”peak to peak”, är 19.6 mV, alltså ± 9,8 mV.
- Frekvensen på de högsta topparna i signalen är ca 50-1000 Hz.

Figur 43 nedan visar den analoga utsignalen vid 100 % i duty cycle, alltså den högsta utspänningen som kan visas av givaren. Detta sker när ställdonet befinner sig i sitt ändläge.



Figur 43 Utsignalen vid 100 % duty cycle

Enligt grafen är:

- Signalens spänning när ställdonet står i sitt ändläge är ca 10,1 V.
- Ripplet var det samma som vid de övriga PWM-värdena förutom vid 0 % enligt ovan.

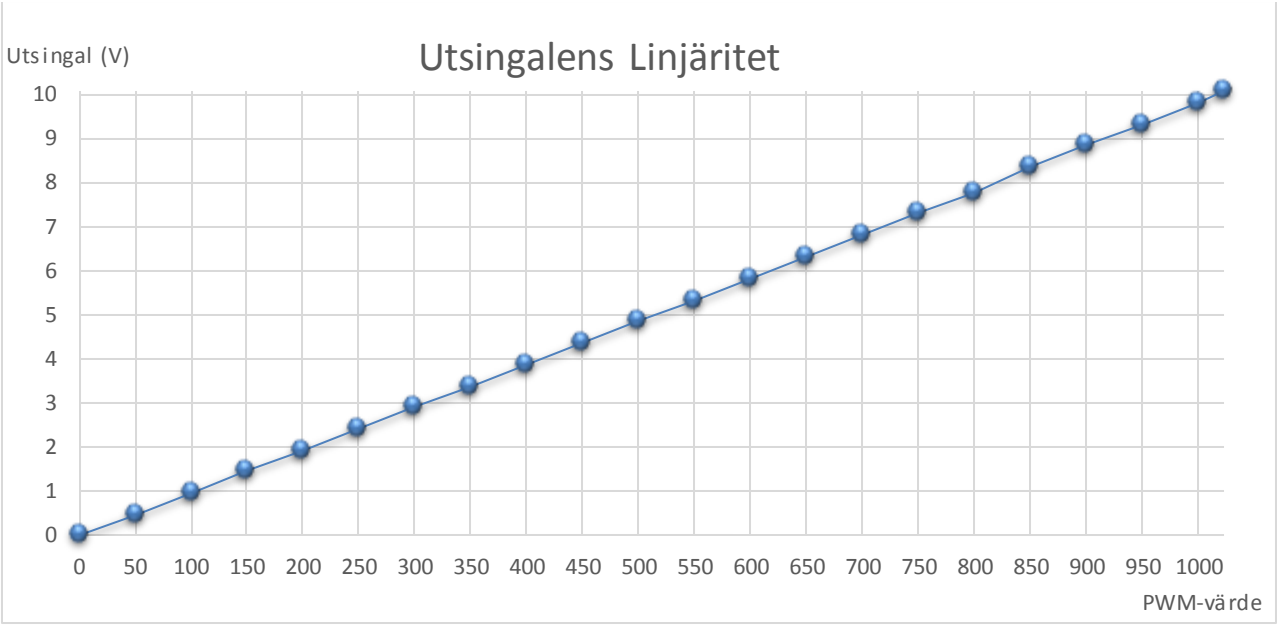
Givarens linjäritet

Tabellen visar hur utsignalen ändrar sig när PWM-värdet ökar från 0 till 1023. (ökningen av PWM-värdet simulerar att ställdonet rör sig). Tabellen visar även motsvarande position (i procent) som ställdonet ska ha för varje PWM-värde.

Tabell 1 Mätvärden för utsignalen vid ökat PWM-värde och motsvarande ställdonsposition

Ställdons position	PWM-värde	Utsignal (V)
0,00 %	0	0,004
4,89 %	50	0,475
9,78 %	100	0,952
14,66 %	150	1,45
19,55 %	200	1,93
24,44 %	250	2,41
29,33 %	300	2,91
34,21 %	350	3,39
39,10 %	400	3,88
43,99 %	450	4,35
48,88 %	500	4,85
53,76 %	550	5,34
58,65 %	600	5,82
63,54 %	650	6,31
68,43 %	700	6,8
73,31 %	750	7,3
78,20 %	800	7,78
83,09 %	850	8,35
87,98 %	900	8,84
92,86 %	950	9,33
97,75 %	1000	9,82
100 %	1023	10,1

Grafen nedan illustrerar utsignalens spänning som funktion av PWM-värdet.



Figur 44 Graf som visar utsignalens spänning som funktion av PWM-värdet

Livslängd vid strömavbrott

Tiden mättes från att huvudspänningen försvann till att internspänningen gick under 2,8 V.

- 16 sekunder om kondensatorn var fullt laddad
- 13 sekunder om kondensatorn var laddad i 2 minuter
- Mindre än 4 sekunder om kondensatorn var laddad i 1 minut

Övriga tester

Här konstateras att de övriga testerna i kapitel ”6.5.4 Genomförda tester” som gjordes klarades utan problem.

- Klarar av strömbortfall
- Klarar längre pass av körning utan att räkna fel (ett par timmar)
- Klarar av både höger- och vänstermonterad motor
- Klarar av att köras med hög pulstågsfrekvens (400 Hz)

7.3 Kretsritningen och BOM

Bilaga A innehåller en komplett ritning av kretsen. Kretsritningen innehåller det nya filtret C och alla andra komponenter som ändrades. Kretsen innehåller även hallgivarna och deras kringkomponenter. Hallgivarnas konstruktion är tagen från förstudien. Inget test kunde göras för att bekräfta att de fungerar. Det förutsätts att de redan är provade i tidigare encodermodeller E2 och E3.

Bilaga B innehåller BOM-listan av de komponenter som ingår i kretsen.

7.4 Mjukvara

Källkoden för mjukvaran kan ses i sin helhet i Bilaga C.

8 DISKUSSION

Under arbetets gång har det uppkommit flera tänkbara felscenarion som kan inträffa. Vi har löst de uppkomna problemen genom att justera hårdvara eller genom att hantera felsituationen i mjukvaran. Ett problem som måste åtgärdas innan givaren kan börja användas i verkligheten är att spänningsövervakningen görs med en digital ingång, detta tas upp i diskussionen nedan. Det kan finnas andra situationer som inte vi har tänkt på under detta arbete som kan resultera till E4-encodern kommer att visa fel utsignal. Fler tester bör göras av någon utomstående för att hitta eventuella brister i konstruktion och mjukvara.

8.1 Direkta risker med framtagen givare

Här beskrivs några direkta risker som vi ser med givaren.

Spänningsövervakning med digital ingång

Huvudspänningen övervakas som bekant av PIC:en och läses av en digital ingång av typen TTL på PIC-processorn. Att läsa en analog signal med en digital ingång kan innebära problem. Eftersom området mellan 0,8 V och 2 V är odefinierat område för en TTL-ingång, kommer en spänning i detta område kunna tolkas som både en logisk etta och nolla. Detta skapar en risk för att encodern kommer bete sig okontrollerat och i värsta fall att fel position indikeras. Särskilt misstänks att problem kommer uppkomma om huvudspänningen ligger på ett värde mellan ca 4 V och 11 V i 4 sekunder eller mer. Detta har aldrig skett under testerna då vi endast testat med huvudspänningen i de tre lägena 12 V, 24 V och ingen spänning.
En lösning på detta problem kan vara att använda sig av en analog ingång istället, då slipper man problemet med att signalen kan bli odefinierad och man kan alltid vara säker på vad statusen på huvudspänningen är.

Felindikation

Att använda den analoga utsignalen för att indikera fel genom att sätta spänningen på konstant 5 V kan tänkas vara farligt i vissa lägen om inte överordnat system hanterar det på ett bra sätt. I framtiden kan det vara bra att ha en egen pinne för att indikera fel eller ett annat sätt.
Ett alternativt sätt skulle kunna vara att använda 0,5 V -10 V för att visa positionen på ställdonet och att 0 V skulle motsvara ett fel. Detta skulle också vara bra om encodern av någon anledning skulle sluta att fungera och inte ge någon analog utspänning alls. Dessa fel skulle då enkelt kunna uppfattas av ett överordnat system som hanterar felen på lämpligt sätt.

8.2 Möjliga risker med framtagen givare

Mätinstrument

Mätinstrumenten som använts för att mäta spänningar har varit ett oscilloskop samt en multimeter. Alla mätvärden i rapporten bygger på dessa instrument och information om när instrumenten kalibrerats känns ej till.

Hallgivare

Den slutgiltiga BOM-listan och kretsschemat innehåller även hallgivarna och deras kringkomponenter. Hallgivarna med kringkomponenters konstruktion är tagen från direkt från förstudien och inga tester har gjorts på dessa för att bekräfta att konstruktionen fungerar. Det förutsätts att de redan är provade i tidigare encodermodeller E2 och E3.

8.3 Diskussion av resultatet

Här diskuteras resultaten från resultatkapitlet.

Begränsning i hur givaren kan användas

Då laddningen av kondensatorn har blivit en flaskhals i konstruktionen har en användningsbegränsning lagts till.

Användningsbegränsningen lyder:
Ändring av position får endast ske då givaren är och har varit inkopplad till huvudspänningen i minst 1 minut, samt under utsatt tid för eftergång.

Anledningen till att givaren måste varit spänningsatt i minst en minut innan den används är för att kunna garantera att kondensatorn kan försörja kretsen i minst fyra sekunder. När huvudmatningen försvinner tar det 4 sekunder innan mjukvaran sparar undan de aktuella värdena i långtidsminnet EEPROM. Om kondensatorn ej hunnit ladda upp sig tillräckligt, kommer processorn att dö innan 4 sekunder och därmed sparas inte de aktuella värdena undan. Givarens position är då ej längre giltig.
En möjlig lösning är att minska uppladdningstiden är att minska tiden för eftergång från 4 sekunder som programvaran ser ut nu, till 0,5s som föreslogs i förstudien. Samma typ av problem kvarstår dock men uppladdningstiden har förkortats till en bråkdel.
En ytterligare förbättring skulle kunna vara att använda en *dynamisk tid för eftergång*. Detta tas upp senare i diskussionen.
Notera även att användningsbegränsningen innebär att positionen på ställdonet ej ändras när givaren är avstängd eller innan uppladdningstiden är slut.

Max och min för analog utgång

Enligt kraven skall givaren visa 0 V-10 V beroende på position. Men när ställdonet befinner sig hemmaläget visas inte 0 V utan 4 mV istället. Motsvarande problem uppkommer för ändläget då istället för 10 V ligger utspänningen på 10.1 V.
Vid hemmaläget ökade även ripplets storlek och frekvenser jämfört med vid de andra PWM-värdena. Förklaringen till detta vet vi inte om varför de uppkommer, men alla tre problemen har tagits upp med SKF och de kommer inte att innebära några problem.
Anledningen till att 4 mV visas är att nuvarande OP inte kan leverera 0 V på sin utgång, detta kan läsas i databladet för LM258 (Fairchildsemi, 2010) [5].
Anledningen till att det visas 10,1 V istället för 10 V kan bero på noggrannheten på instrumenten vi använder, vid test av andra multimetrar har spänningen uppmätts till 9,99 V.

Stegsvar för analog utgång

Vid 0,7 V stannar utsignalens spänningsnivå i cirka 30 us och skapar en terasspunkt innan spänningen går vidare upp mot sitt börvärde. Detta kan bero på OP:ns kretsutformning, detta kan ses i databladet för LM258 (Fairchildsemi, 2010) [5]. Vid spänningar lägre än 0,7 V på utgången slutar en transistor inuti OP:n att leda, för att kunna reglera strömmen på utgången använder då förstärkaren endast en strömregulator på 50 uA. Detta kan vara anledningen till att steget har en liten terrasspunkt vid 0,7 V.
I övrigt stämmer det verkliga stegsvaren mycket bra med det simulerade stegsvaret för filter C. Stigtiden är nästan identisk men däremot är ripplet på signalen mycket mindre i verkligheten. Mer om det under nästa punkt.

Rippel på analog utgång

När dämpningen vid 31,125 kHz bestämdes till -48 dB, byggde detta på att ripplet som skapades av PWM-frekvenserna skulle vara max ±10 mV på utgången. På alla spänningsnivåer utom 0 V på utsignalen uppmättes ett ”peak to peak”-värde på 16,4 mV, alltså ± 8,2 mV. Detta är till och med lite bättre än vad vi räknat fram att det skulle vara. Dessutom är frekvensen på de högsta topparna som gav ”peak to peak” värdet ca 50-100 Hz. Detta tyder på att ripplet skapat av PWM-frekvenserna är mindre än ±8,2 mV.
I simuleringen av filter C ficks rippel med storleken ±30 mV vilket är ganska mycket mer än det beräknade. Filtret provades ändå och visade sig vara mycket bättre i verkligheten. Anledningen till detta kan vara att modellen för OP:n i simuleringen inte var tillräckligt bra. Eftersom att resultatet i verkligheten stämde med uträkningarna gjordes ingen förbättrad simulering.

8.4 Förbättringsmöjligheter allmänt

Det finns några förbättringsmöjligheter som vi sammanfattar i följande punkter.

Ett mer noggrant filter

Stigtiden för filtret var i början av projektet ett problem, då det i starten av arbetet användes filter av första ordningen. Ett filter av första ordningen är mycket långsammare jämfört med ett filter av tredjeordningen om de skall ha samma dämpning vid en viss frekvens i stroppbandet. Därför valdes en relativt låg men acceptabel förstärkning på -48 dB för PWM-frekvensen. Detta värde användes sedan också för alla efterkommande filter. Resultatet blev ett väldigt snabbt filter och systemet behöver inte vara så snabbt då ändring av utsignalen kommer att ske i samma takt som ställdonet ändrar position. Därför skulle filtret kunna använda lägre brytfrekvenser och på så sätt få en större dämpning till kostnaden av att bli lite långsammare. Då skulle en signal med mindre rippel fås.

Hur man skulle kunna förlänga eftergångstiden till 30 sekunder.

Önskemål har under projektets gång kommit fram till att SKF vill ha längre tid för eftergång, upp emot 30 sekunder. Detta önskemål var inte med när förstudien gjordes och har inte heller behandlats i arbetet. Nedan har vi radat upp några möjliga sätt att förlänga livslängden på. Naturligtvis kan samma åtgärder användas för att få en mindre kondensator och behålla samma tid för livslängd.

- Skippa PLL
  För att få en så hög PWM-frekvens som möjligt valdes den högsta klockfrekvensen på processorn. För att kunna köra i den högsta klockfrekvensen krävs att PLL används. Om PLL används måste BOR-spänningen vara inställd på 2,8 V. Skulle PLL avaktiveras kan BOR-spänningen ställas till en lägre nivå och processorn kommer då att leva längre tid med samma kondensator.
  Man kan stänga av PLL genom att köra intern klockfrekvens på 16 MHz istället. Då ändras dock PWM-frekvensen som gör att vi måste ändra om hela LP-filtret. Dessutom om samma brytfrekvenser behålls kommer utsignalen att få mer rippel.
  Ett annat sätt är att stänga av PLL genom att använda extern oscillator istället för den interna och fortsätta köra i 32 MHz. PWM-frekvensen bibehålls och LP-filtret behöver inte göras om. Samma precision på den analoga utsignalen bibehålls. Dock krävs det fler komponenter för att skapa en extern oscillator, vilket tar mer plats på kretskortet.
- Större eller fler kondensatorer
  Genom att använda en större eller fler kondensatorer kommer internspänningen kunna hålla en tillräckligt hög internspänning en längre tid. Då det är platsbrist i encoderhuset kan dock detta bli problematiskt då denna lösning kommer att kräva mer fysisk plats.
- Minska strömförbrukningen
  Ett annat sätt är att försöka få ner strömförbrukningen i kretsen. Ett alternativ är att köra processorn i en lägre klockfrekvens då den kommer

att dra mindre ström. En granskning av kretsen och komponenternas datablad kan behöva göras för att se var strömförbrukningen är hög.

Dynamisk tid för eftergång

Under den tidigare rubriken ”**Begränsning i hur givaren kan användas**” diskuterades hur uppladdningstiden kunde förkortas genom att minska eftergångstiden. Här presenteras tanken om en dynamisk tid för eftergång som skulle kunna kombinera önskemålet om en eftergångstid på 30 sekunder med en kort uppladdningstid. Genom att använda sig av en dynamisk eftergångstid skulle man kunna låta tiden för eftergång bero på hur mycket energi kondensatorn har för tillfället. Genom att garantera en minsta eftergångstid kan en motsvarande erforderlig uppladdningstid räknas ut och införas i användningsbegränsningen. Genom att låta nivån på internspänningen bestämma när värdena skall sparas till EEPROM kan eftergångstider upp emot 30 sekunder fås om energi till det finns i kondensatorn. Eftergångstiden kommer på det här sättet att variera mellan den bestämda minsta tiden exempelvis 0,5 sekunder upp emot 30 sekunder eller högre beroende på kondensator mm.

8.5 Övriga tankar

Här presenteras övriga saker som vi har diskuterat under arbetets gång.

Möjlighet till högre upplösning

Den verkliga informationen om vad ställdonets position är har ofta mycket högre upplösning än vad som kan visas på den analoga utgången. Detta då värdet på *counter* skalas ner till upplösningen av ett 10-bitars tal. Det verkliga värdet på counter skulle kunna användas genom att skippa D/A-omvandling och istället skicka en digital signal till ett överordnat system. Detta skulle kunna ske med seriell kommunikation. Dock måste processorn bytas ut till en modell med fler ben, då alla ben på nuvarande processor är upptagna.

Upptäckande av ändrat läge vid avstängd encoder

Ett tänkbart scenario skulle vara att ställdonet har ändrat läge medan ställdonet är avstängt. Detta är inte tillåtet enligt användningsbegränsningen, men om det skulle hända ändå kan felhanteringssystemet i mjukvaran upptäcka detta i 1 fall av 4. Förklaring följer: I och med att det är giltigt att hoppa ett tillstånd åt vartdera håll samt att stå kvar i samma tillstånd, så kan felhanteringssystemet bara upptäcka att läget ändrats om tillståndet har ändrats två steg, alltså hoppat över ett tillstånd.

Totalkostnad

Arbetet har avgränsats från att beräkna pris på slutprodukten. Dock har priserna på komponenterna som använts alltid kollats och jämförts med förstudiens priser. De delar som bytts ut från BOM-listan i förstudien har bytts till komponenter med motsvarande eller lägre pris. Därför kommer givaren vi

tagit fram att hålla samma eller lägre totalkostnad än det kostnadsförslag som gjordes i förstudien.

8.6 Förbättringsmöjligheter för Mjukvaran

På grund av tidsbegränsning finns det delar kvar på mjukvaran att förbättra. Mjukvaran fungerar dock som den ser ut nu och kan användas som den är.
Förbättringsmöjligheterna diskuteras här nedan.

- char calibrate_button_is_pressed(void) (button debounce)
    - Om man vill ha bättre debounce i befintlig funktion måste man öka delay_us(); eller öka antalet for-loopar. Men då tar den får lång tid för att köras och pulser riskeras att missas.
    - Istället bör funktionen ändras så att den ökar en räknare varje gång den körs istället, som den är nu tar den onödigt lång tid at exekvera pga. att den använder en for-loop.
    - Om funktionen fixas till så kan den användas överallt där makrot CALIBRATE_BUTTON används i nuläget. Som den ser ut kan den bara användas där läget inte är tidskritiskt.
- En debounce-funktion även för den andra ingången, POWER, alltså den ingången som kollar huvudmatningen, vore bra(men ej nödvändig) att ha för att filtrera bort ev. flimmer.
- ”save went well”-flagga kan användas för att förhindra fel
    - Försök har gjorts för att få till en sådan funktion men har inte fått det att fungera.
    - Tanken bakom detta är att om processorn stängs av pga. Brown out timer innan den har hunnit spara undan alla värden så skall detta synas vid nästa uppstart och sätta en felflagga. Processorn kan under tiden den levt tagit emot pulser och ökat räknare utan att sedan hunnit spara undan detta innan reset.
    - Ett tänkbart scenario då detta skulle kunna ske är om ställdonet och encodern blir spänningssatt samtidigt. Ställdonet börjar köra och encodern fungerar som det skall. Dock försvinner huvudmatningen ganska snart efter start, så att kondensatorn inte har hunnit att ladda upp sig ordentligt. Encodern lever vidare för att ta hand om en viss eftergång på ställdonet, men processorn dör innan 4 sekunder har gått och missar därför att spara undan sina värden. Vilket resulterar i att signalen från givaren ej längre är giltig. En ”save went well”-flagga hade i detta fallet indikerat ett fel. Om flaggan ej används kommer E4 att starta i tron om att ställdonet står i den gamla positionen.
      Dock kommer ett sådant fel upptäckas i ett fall av fyra genom att använda ”Upptäckning av ändrat läge vid avstängd encoder” som togs upp tidigare i diskussionen.

- delay_us och delay_ms
    - Fördröjningsfunktionerna är ej helt kalibrerade utan har ett ungefärligt värde på fördröjningen.

    - Funktionerna ger endast rätt tidsfördröjning vid användning av klockfrekvensen 32 MHz, de är alltså klockfrekvensberoende.

- Variabeln calibrate_timer (används endast i funktionen calibrate())
    - calibrate_timer används för att man skall behöva hålla inne knappen i två sekunder för att komma in i kalibreringsläget. Dessa två sekunder är beroende på hur långa delay_ms man har för att få LED att blinka.
    - calibrate_timer används också som en liten delay så att man inte ska hoppa ur kalibreringsläget när man väl kommit in. Detta används för att konstruktionen på funktionen calibrate_button_is_pressed(void) är dålig och inte ger tillräcklig debounce på knappen.
- void manage_powerloss(void)
    - Om huvudmatningen är till skrivs ändå nya värden till powerloss_timer och LED släcks varje varv i main-loopen. Detta är lite onödigt resursmässigt, men gör ingenting att det görs så som programmet ser ut nu. Men är värt att tänka på om man vill utveckla programvaran då det skulle kunna vara dumt att släcka leden om man vill ha den på.

9 SLUTSATSER

Slutsatsen som kan dras från detta arbete är att det är möjligt att bygga en E4 som omvandlar ställdonets position till en analog utsignal. Vissa delar av lösningsförslaget i förstudien fick ändras, men konceptet fungerade i stort. Spänningsövervakningen måste ändras till att använda en analog ingång eller ett alternativt sätt istället för en digital ingång. Givaren kan användas precis som den mekaniska potentiometern med ett undantag, att användningsbegränsningen nedan måste följas.

”Ändring av position får endast ske då givaren är och har varit inkopplad till huvudspänningen i minst 1 minut, samt under utsatt tid för eftergång.”

10 REFERENSER

[1] *Microchip.* (2012). Hämtat från 8/14-Pin Flash Microcontrollers with XLP Technology: http://ww1.microchip.com/downloads/en/DeviceDoc/41413C.pdf

[2] *National Instruments*, 3268KQ0M. (2003). Hämtat från http://digital.ni.com/public.nsf/allkb/294E676237526566862 56DB800508989

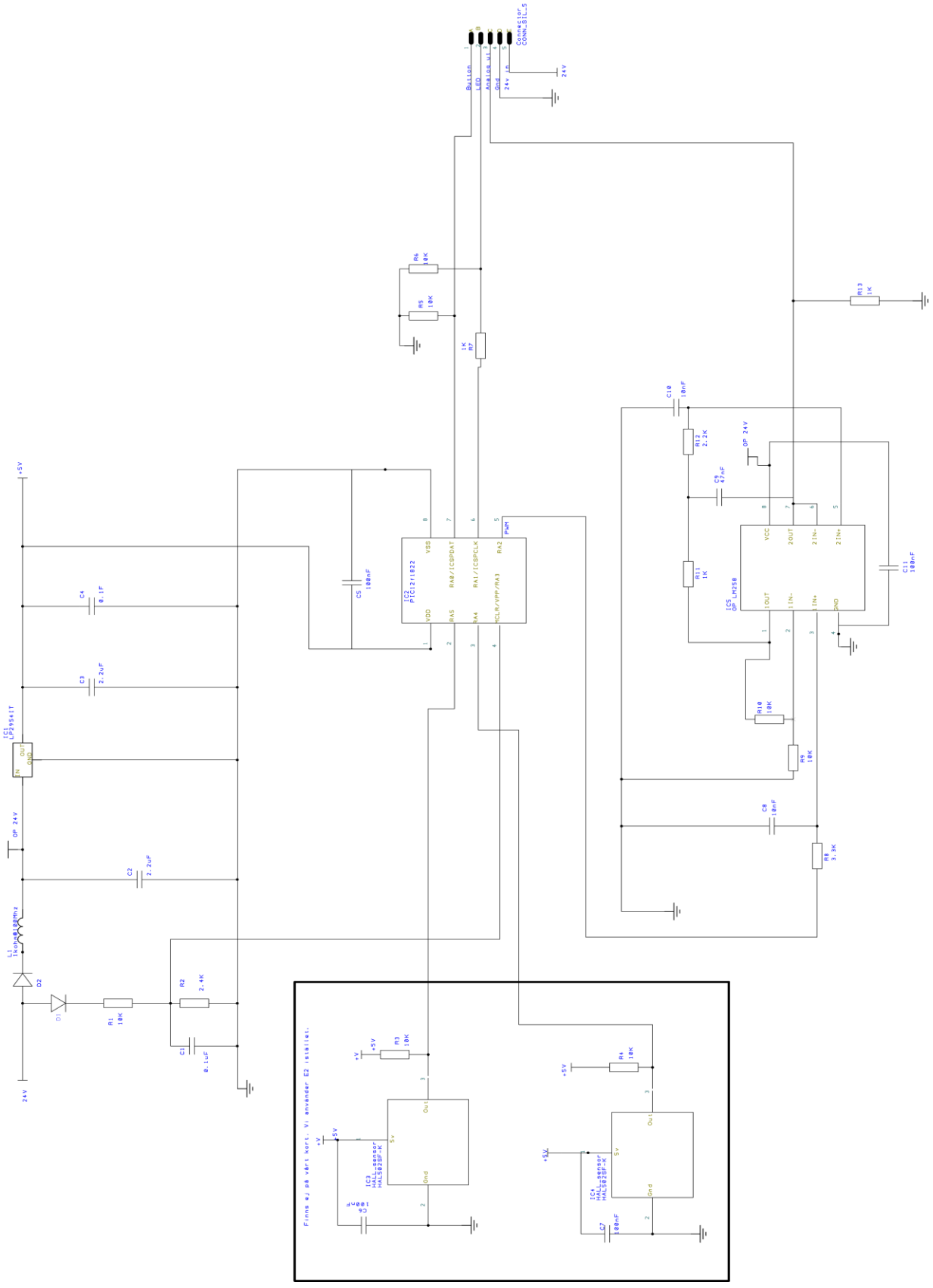
[3] Edward, R. (2006). *Hall-Effect Sensors : Theory and Application (2nd Edition).* Burlington, MA ,USA.

[4] Potentiometer, SENSOFOIL® Membrane. (u.d.). tastatur. Hämtat från Regal Components AB: http://www.tastatur.de/fileadmin/user_upload/Downloads/EN/ProductInfo_Sensofoil_US_V1.4.pdf

[5] *Fairchildsemi.* (2010). Hämtat från LM258A Dual Operational Amplifier: http://www.fairchildsemi.com/ds/LM/LM258.pdf

APPENDIX

Bilaga A. **Kretsschema**



Bilaga B. **BOM**

Position	Komponent	Värdet
C1	Kondensator	0,1 uF
C2	Kondensator	2,2 uF
C3	Kondensator	2,2 uF
C4	Kondensator	0,1 F
C5	Kondensator	0,1 uF
C6	Kondensator	0,1 uF
C7	Kondensator	0,1 uF
C8	Kondensator	10 nF
C9	Kondensator	47 nF
C10	Kondensator	10 nF
C11	Kondensator	0,1 uF
D1	Diod	40 V/0.2 A
D2	Diod	40 V/0.2 A
IC1	Spänningsregulator	MCP2954IT
IC2	Mikroprocessor	PIC12F1822-I/SN
IC3	Hall-sensor	HAL502SF-K
IC4	Hall-sensor	HAL502SF-K
L1	EMC-Filter	1 kOhm@100 MHz
R1	Resistor	10 k, 1 %
R2	Resistor	2,4 k, 1 %
R3	Resistor	10 k, 1 %
R4	Resistor	10 k, 1 %
R5	Resistor	10 k, 1 %
R6	Resistor	10 k, 1 %
R7	Resistor	1 k, 1 %
R8	Resistor	3,3 k, 1 %
R9	Resistor	10 k, 0,1 %
R10	Resistor	10 k, 0,1 %
R11	Resistor	1 k, 0,1 %
R12	Resistor	2,2 k, 1 %
R13	Resistor	1 k, 1 %
P1	Kontaktdon	1.25 mm / 5-pole

Bilaga C. **Mjukvara**

MAIN.C

```
/*
 * File:   main.c
 * Author: nabil.fatmi
 *
 * Created on den 1 april 2014, 12:24
 */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include "system.h"
#include "user.h"

int counter ;
char errorflag = 0;
char last_magnet_state;
double PWM_factor;
unsigned int powerloss_timer = POWERLOSS_TIMER_VALUE_500ms;
unsigned int powerloss_multip_timer = 8;
char previous_button_state = 0;


int main(void)
{
    INTCONbits.GIE = 0;                 //Disables all interrupts
    ConfigureOscillator( );             //Set WDT and Clockfreq(32Mhz)
    InitIO( );                          //Pin Output/Input select
    ConfigurePWM( );                    //Set PWMfreq(31.25khz)
    CALIBRATE_LED = 0;                  //Make sure Led is off
    errorflag = read_char_EEPROM(EEPROM_ADRESS_ERRORFLAG); //Old errorflagvalu
e
    if(STATUSbits.nTO == 0)             //if WDT-reset occurred
    {
        errorflag = 2;  //if WDT-reset occurred, set errorflag
    }

    /*
     * Get old values of last_magnet_state and counter from EEPROM
     */
    last_magnet_state = read_char_EEPROM(EEPROM_ADRESS_LAST_MAGNET_STATE);
    counter = read_int_EEPROM(EEPROM_ADRESS_COUNTER);
     /*
     * Get stored maxvalue of counter and calculate PWM_factor
     */
    calculate_PWM_factor();

    manage_pwm_output();
    StartPWM();

    /*
     * mainloop: Dont use any long delays or EEPROM-
read or writes inside here
     * they take too much time.
     * manange_movement has to be executed in a minumumfreq of 1000hz
     * (exeption in powerloss when we have to save our values)
     */
    counter=0;
    while (1)
    {
      write_dutycycle(counter);
      delay_ms(5000);
```

```
            counter= counter+50;
             CLRWDT();
          }
        return (EXIT_SUCCESS);
    }
```

SYSTEM.H

```
/*
 * File:   system.h
 * Author: nabil.fatmi
 *
 * Created on den 1 april 2014, 13:35
 */
void ConfigureOscillator(void);
void InitIO(void);
void ConfigurePWM(void);
void StartPWM (void);
```

USER.H

```
 /*
 * File:   user.h
 * Author: nabil.fatmi
 *
 * Created on den 1 april 2014, 13:36
 */
#define CALIBRATE_LED LATAbits.LATA1
#define CALIBRATE_BUTTON PORTAbits.RA0
#define HALL_A PORTAbits.RA4
#define HALL_B PORTAbits.RA5
#define POWER PORTAbits.RA3
#define MAGNET_STATE_1 1
#define MAGNET_STATE_2 2
#define MAGNET_STATE_3 3
#define MAGNET_STATE_4 4
#define EEPROM_ADRESS_COUNTER 0
#define EEPROM_ADRESS_MAXVALUE 2
#define EEPROM_ADRESS_ERRORFLAG 4
#define EEPROM_ADRESS_LAST_MAGNET_STATE 5
#define POWERLOSS_TIMER_VALUE_500ms 60000
#define ERROR_INDICATE_VOLTAGE 5
#define DUTY_CYCLE_MAX_VALUE 1023
#define MAX_OUTPUT_VOLTAGE 10.0


void delay_ms(int ms);
void delay_us(int us);
void write_dutycycle ( int );
char check_current_magnet_state  (void);
void calculate (int pos2);
void calculate_PWM_factor(void);
void manage_pwm_output(void);
void manage_movement (void);
void calibrate (void);
void write_char_EEPROM(unsigned char adress, char data);
char read_char_EEPROM(unsigned char adress);
void write_int_EEPROM(unsigned char adress, int data);
int read_int_EEPROM(unsigned char adress);
void manage_powerloss(void);
void manage_error(void);
char calibrate_button_is_pressed(void);
```

CONFIG_BITAR.C

```
/*
 * File:   Config_bitar.c
 * Author: nabil.fatmi
 *
 * Created on den 1 april 2014, 12:53
 */

#include <xc.h>

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

// CONFIG1
#pragma config FOSC = INTOSC    // Oscillator Selection (INTOSC oscillator: I/O function on CLKIN pin)
#pragma config WDTE = ON        // Watchdog Timer Enable (WDT enabled)
#pragma config PWRTE = ON       // Power-up Timer Enable (PWRT enabled)
#pragma config MCLRE = OFF      // MCLR Pin Function Select (MCLR/VPP pin function is digital input)
#pragma config CP = OFF         // Flash Program Memory Code Protection (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Memory Code Protection (Data memory code protection is disabled)
#pragma config BOREN = ON       // Brown-out Reset Enable (Brown-out Reset enabled)
#pragma config CLKOUTEN = OFF   // Clock Out Enable (CLKOUT function is disabled. I/O or oscillator function on the CLKOUT pin)
#pragma config IESO = ON        // Internal/External Switchover (Internal/External Switchover mode is enabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable (Fail-Safe Clock Monitor is disabled)

// CONFIG2
#pragma config WRT = OFF        // Flash Memory Self-Write Protection (Write protection off)
#pragma config PLLEN = ON       // PLL Enable (4x PLL enabled)
#pragma config STVREN = ON      // Stack Overflow/Underflow Reset Enable (Stack Overflow or Underflow will cause a Reset)
#pragma config BORV = HI        // Brown-out Reset Voltage Selection (Brown-out Reset Voltage (Vbor), high trip point selected.)
#pragma config LVP = OFF        // Low-Voltage Programming Enable (High-voltage on MCLR/VPP must be used for programming)
```

SYSTEM.C

```
/*
 * File:   system.c
 * Author: nabil.fatmi
 *
 * Created on den 1 april 2014, 13:30
 */

#include <xc.h>            /* HiTech General Includes */

#include "system.h"

void ConfigureOscillator(void)
{
    OSCCONbits.IRCF = 0b1110;       //set to the 8 MHz HFINTOSC
    OSCCONbits.SCS  = 0b00;         //Clock determined by FOSC in Conf. Word 1

    WDTCONbits.WDTPS = 0b01000;     // 1:8192 (Interval 256 ms typ)
}
void InitIO(void)
```

```
{
    TRISAbits.TRISA0 = 1;   //DI - Calibrate button
    TRISAbits.TRISA1 = 0;   //DO - Calibrate LED
    //TRISAbits.TRISA2 = 0; //PWM out done in startPWM().
    TRISAbits.TRISA4 = 1;   //DI - HAL sensor1
    TRISAbits.TRISA5 = 1;   //DI - HAL sensor2
    ANSELA =0 ;
}
void ConfigurePWM(void)
{   PR2 = 255;          //Load the PR2 register with the PWM period value .
    T2CON = 0;          //Postscaler = 0 ; Prescaler = 0; Timer2 on bit is off

    CCP1CONbits.CCP1M = 0b1100;      //PWM mode: active-high;
    CCP1CONbits.P1M = 0b00;          //Single output;
}


/*
 * Start PWM with this function to send a complete duty cycle and
 * period on the first PWM output.
 */
void StartPWM (void)
{
    PIR1bits.TMR2IF = 0; //Clear Timer2 IF
    T2CONbits.TMR2ON = 1;//Start Timer2
    while ( !PIR1bits.TMR2IF ) //Wait until overflow
    {
    }
    TRISAbits.TRISA2 = 0; // PWM-output on
 }
```

USER.C

```
/*
 * File:   user.c
 * Author: nabil.fatmi
 *
 * Created on den 1 april 2014, 13:18
 */
#include <xc.h>
#include "user.h"
#include "system.h"

extern int counter;
extern char errorflag;
extern char last_magnet_state;
extern double PWM_factor;
extern unsigned int powerloss_timer;
extern unsigned int powerloss_multip_timer;
extern char previous_button_state;
/*
 * Delay that takes ms milliseconds to do. Value in forloop is for
 * 32Mhz clockfreq
 */
void delay_ms(int ms)
{   for(int j =0;j<ms;j++)
    {CLRWDT();
        for(int i=0;i<600;i++)
        {
        }
    }
}
/*
 * Delay that takes us microseconds to do. Number of NOP(); is for
 * 32Mhz clockfreq
 */
```

```
34.  void delay_us(int us)
35.  {
36.      NOP();
37.      NOP();
38.      NOP();
39.  }
40.  /*
41.   * Writes PWM-dutycycle, takes in a value between 0-1023 decimal
42.   */
43.  void write_dutycycle ( int dvalue )
44.  {
45.      CCP1CONbits.DC1B = dvalue;
46.      CCPR1L = dvalue >> 2;
47.  }
48.  /*
49.   * Check Hallsensors A and B and returns wich state the magnet is in
50.   */
51.  char check_current_magnet_state (void)
52.  {
53.      char i;
54.      if ( HALL_A == 1 && HALL_B == 0 )
55.      {
56.          i = MAGNET_STATE_1;
57.      }
58.      else if ( HALL_A == 1 && HALL_B == 1 )
59.      {
60.          i = MAGNET_STATE_2;
61.      }
62.
63.      else if ( HALL_A == 0 && HALL_B == 1 )
64.      {
65.          i = MAGNET_STATE_3;
66.      }
67.
68.      else if ( HALL_A == 0 && HALL_B == 0 )
69.      {
70.          i = MAGNET_STATE_4;
71.      }
72.      return i ;
73.  }
74.
75.
76.  /*
77.   * Compares the input char(current magnet state) with last magnet state
78.   * There has to be a differens between them before this function is executed
79.   * If there is a legal order it increase/decrease counter,
80.   * else errorflag is to be set
81.   */
82.  void calculate_counter (char current_magnet_state)
83.  {
84.      if (last_magnet_state == MAGNET_STATE_1)
85.      {
86.          if (current_magnet_state == MAGNET_STATE_2)
87.          {
88.              counter = counter + 1;
89.
90.          }
91.          else if (current_magnet_state == MAGNET_STATE_4)
92.          {
93.              counter = counter - 1;
94.
95.          }
96.
97.          else
98.           {
99.              errorflag = 1;
```

```
100.              }
101.          }
102.          else if (last_magnet_state == MAGNET_STATE_2)
103.          {
104.              if (current_magnet_state == MAGNET_STATE_3)
105.              {
106.                  counter = counter + 1;
107.              }
108.              else if  (current_magnet_state == MAGNET_STATE_1)
109.              {
110.                  counter = counter  - 1;
111.              }
112.              else
113.               {
114.                  errorflag = 1;
115.               }
116.          }
117.          else if (last_magnet_state == MAGNET_STATE_3)
118.          {
119.              if (current_magnet_state == MAGNET_STATE_4)
120.              {
121.                  counter = counter + 1;
122.              }
123.              else if (current_magnet_state == MAGNET_STATE_2)
124.              {
125.                  counter = counter  - 1;
126.              }
127.              else
128.               {
129.                  errorflag = 1;
130.               }
131.          }
132.          else if (last_magnet_state == MAGNET_STATE_4)
133.          {
134.              if (current_magnet_state == MAGNET_STATE_1)
135.              {
136.                  counter = counter + 1;
137.              }
138.              else if (current_magnet_state == MAGNET_STATE_3)
139.              {
140.                  counter = counter  - 1;
141.              }
142.              else
143.              {
144.                  errorflag = 1;
145.              }
146.          }
147.      }
148.      /*
149.       * Get stored maxvalue of counter and calculate PWM_factor
150.       * NOTE: that if max_puls is negative, then also PWM_factor gets nagative
151.       */
152.      void calculate_PWM_factor()
153.      {
154.          double max_puls = read_int_EEPROM(EEPROM_ADRESS_MAXVALUE);
155.          PWM_factor = ( 1023 / max_puls );
156.      }
157.
158.      /*
159.       * Scales the counter to a number between 0-1023 and writes this number to
160.       * PWM-dutycycle.
161.       * if counter is outside the range, then set to min- or max-value.
162.       * NOTE: that if counter is negative then also PWM_factor is negative
163.       */
164.      void manage_pwm_output(void)
165.      {
```

```
166.        int duty;
167.        duty=counter*PWM_factor; //NOTE: if negative values
168.        if(duty<0)
169.            duty=0;
170.        if(duty>1023)
171.            duty=1023;
172.        write_dutycycle (duty);
173.    }
174.
175.    /*
176.     * This function has to be executed in a minumumfreq of 1000hz else
177.     * pulses can be missed.
178.     */
179.     void manage_movement (void)
180.    {   char current_magnet_state;
181.        current_magnet_state = check_current_magnet_state ();
182.        if (last_magnet_state != current_magnet_state)
183.        {
184.            calculate_counter ( current_magnet_state );
185.            last_magnet_state = current_magnet_state;
186.            manage_pwm_output();
187.        }
188.    }
189.    /*
190.     * Calibrate function
191.     * #1 Hold button for (calibrate_timer*2*delay_ms(50)) seconds until led stops
192.     * flash and shines constant.
193.     * (if the time is less then exit calibrate().)
194.     * #2 Release button to reset counter, this is home.
195.     * this also reset the errorflag.
196.     * #3 You can now drive the acuator to you maxposition
197.     * push the button when you are at maxposition.
198.     * (you cannot do this until calibrate_timer has reached value 0)
199.     * #4 Led flashes to view that you push the button.
200.     * #5 Store counter value of the maxposition in EEPROM and
201.     * calculates the PWM_factor and updates the PWM-output.
202.     */
203.    void calibrate (void)
204.    {
205.        double max_value_counter;
206.        int calibrate_timer = 20;
207.        //#1
208.        while ( calibrate_button_is_pressed() == 1 && calibrate_timer != 0)
209.        {
210.            CLRWDT();
211.            CALIBRATE_LED = 0;
212.            delay_ms(50);
213.            CALIBRATE_LED = 1;
214.            delay_ms(50);
215.            calibrate_timer--;
216.        }
217.        if(calibrate_timer == 0)
218.        {
219.            while(calibrate_button_is_pressed() == 1)
220.            {
221.                CLRWDT();
222.            }
223.            //#2
224.            PWM_factor = 1;
225.            errorflag = 0;
226.            write_dutycycle(0);
227.            counter = 0; //Set counter to zero- this is home
228.            calibrate_timer = 30000;
229.            //#3
230.            while ((CALIBRATE_BUTTON == 0) || (calibrate_timer != 0))
```

```
231.             {
232.                 CLRWDT();
233.                 manage_movement();
234.                 if(calibrate_timer > 0)
235.                 {
236.                     calibrate_timer--;
237.                 }
238.             }
239.             //#4
240.             while ( calibrate_button_is_pressed() == 1)
241.             {
242.                 CLRWDT();
243.                 CALIBRATE_LED = 0;
244.                 delay_ms(50);
245.                 CALIBRATE_LED = 1;
246.                 delay_ms(50);
247.             }
248.             //#5
249.             write_int_EEPROM(EEPROM_ADRESS_MAXVALUE,counter);
250.             /*
251.             * calculate_PWM_factor() be used instead of the two next lines.
252.             * But to save time from a unnessasery EEPROM read, it isn't used here.
253.             */
254.             max_value_counter = counter;
255.             PWM_factor = ( 1023 / max_value_counter );
256.             manage_pwm_output();
257.         }
258.
259.         CALIBRATE_LED = 0 ; //clear the LED before exit
260.     }
261.     /*
262.      * Writes one byte to a EEPROM-adress
263.      */
264.     void write_char_EEPROM(unsigned char adress, char data)
265.     {
266.         EEADRL  = adress;         //Write the adress to EEADRL-register
267.         EEDATL = data;           //Write the byte to EEDATL-register
268.
269.         EECON1bits.WREN = 1;    // Allow write data to EEPROM (Must be reset by SW
)
270.         //Required Sequence to write data to EEPROM
271.         EECON2=0x55;         // Key#1 (used to guard against accidental writes)
272.         EECON2=0xaa;         // Key#2 (used to guard against accidental writes)
273.         EECON1bits.WR=1;     //Start writing
274.
275.         while (PIR2bits.EEIF==0); //Wait for HW to be done
276.
277.         EECON1bits.WREN=0;  // Forbid write data to EEPROM (Required reset)
278.         PIR2bits.EEIF=0;    // Write done, reset EEIF to be able to read EEPROM.
279.     }
280.     /*
281.      * Read one byte from a EEPROM-adress
282.      */
283.     char read_char_EEPROM(unsigned char adress)
284.     {
285.         EEADR=adress;         //Write the adress to EEADRL-register
286.         EECON1bits.EEPGD= 0; // Access to data in EEPROM
287.         EECON1bits.CFGS = 0; // Access to Flash program or data in EEPROM
288.         EECON1bits.RD   = 1; // Begin read from EEPROM
289.         return EEDATA;       // Return data
290.     }
291.     /*
292.      * Write a int to a EEPROM-adress(start adress, (int takes two adresses))
293.      */
```

```
294.    void write_int_EEPROM(unsigned char adress, int data)
295.    {
296.        unsigned char high_char = data >> 8;  //get the 8 HSB
297.        unsigned char low_char = data & 0xFF; //mask out the 8 LSB
298.        write_char_EEPROM(adress, high_char);
299.        write_char_EEPROM(adress+1, low_char);
300.    }
301.    /*
302.     * Read a int from a EEPROM-adress(start adress, (int takes two adresses))
303.     */
304.    int read_int_EEPROM(unsigned char adress)
305.    {
306.        int result;
307.        unsigned char high_char = read_char_EEPROM(adress);
308.        unsigned char low_char = read_char_EEPROM(adress+1);
309.        result = high_char;
310.        result = result << 8;
311.        result = result + low_char;
312.        return result;
313.    }
314.
315.    /*
316.     * If Power supply is gone, a timer starts to count down and led turns on.
317.     * When counter is zero the current values is stored to EEPROM.
318.     * If power supply comes back the counter gets its initial values again.
319.     */
320.    void manage_powerloss(void)
321.    {
322.        if(( POWER == 0) && (powerloss_multip_timer !=0 ))
323.        {
324.            CALIBRATE_LED = 1;
325.            powerloss_timer--;
326.            if(powerloss_timer == 0)
327.            {
328.                powerloss_timer = POWERLOSS_TIMER_VALUE_500ms;
329.                powerloss_multip_timer--;
330.            }
331.            if ( powerloss_multip_timer == 0 )
332.            {
333.                CALIBRATE_LED = 0;
334.                write_int_EEPROM(EEPROM_ADRESS_COUNTER,counter);
335.                write_char_EEPROM(EEPROM_ADRESS_ERRORFLAG, errorflag);
336.                write_char_EEPROM(EEPROM_ADRESS_LAST_MAGNET_STATE, last_magnet_sta
te);
337.                CALIBRATE_LED = 1;
338.            }
339.        }
340.
341.        if ( POWER == 1) //Note: done every main-
loop if POWER = 1 (waste of time)
342.        {   CALIBRATE_LED = 0;
343.            powerloss_timer = POWERLOSS_TIMER_VALUE_500ms;
344.            powerloss_multip_timer = 8;
345.        }
346.
347.    }
348.
349.    /*
350.     * Stores the errorflag to EEPROM,
351.     * set PWM-output at a dutycycle that gives ERROR_INDICATE_VOLTAGE Volt
352.     * Stays here until CALIBRATE_BUTTON is pressed
353.     * (to reset errorflag in calibrate())
354.     */
355.    void manage_error(void)
356.    {
357.        write_char_EEPROM(EEPROM_ADRESS_ERRORFLAG, errorflag);
```

```
358.         write_dutycycle(ERROR_INDICATE_VOLTAGE*(DUTY_CYCLE_MAX_VALUE/MAX_OUTPUT_VO
LTAGE));
359.         while(!CALIBRATE_BUTTON)
360.         {
361.             CLRWDT();
362.         }
363.     }
364.
365.     /*
366.      * Debounce for the CALIBRATE_BUTTON
367.      * you can modify i-
value in forloop and delay_us() to get at better debounce.
368.      */
369.     char calibrate_button_is_pressed(void)
370.     {
371.         char current_button_state = CALIBRATE_BUTTON;
372.         char settled = 1;
373.         for(char i = 0; i < 5 ; i++)
374.         {
375.             if(current_button_state != CALIBRATE_BUTTON)
376.             {
377.                 settled = 0;
378.             }
379.             delay_us(10);
380.         }
381.         if(settled)
382.         {
383.             previous_button_state = current_button_state;
384.         }
385.         return previous_button_state;
386.     }
```