CHALMERS

Testanläggning för automatisk spänningsvariation

Test unit for automatic voltage variation

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

JOHN PERSSON
VICTORIA PETERSON

Institutionen för Signaler och system
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2016

Förord

Detta examensarbete om 15 HP har utförts på institutionen för Signaler och system på Chalmers tekniska högskola och är den avslutande delen av högskoleingenjörsutbildningen Mekatronik (180 HP). Arbetet har under tio veckor utförts på Consat Engineering AB under våren 2016 med Manne Stenberg som examinator och Göran Hult som handledare.

Vi skulle vilja tacka alla inblandade på Consat och Chalmers som delat med sig av tips och erfarenheter under arbetets gång. Ett särskilt tack vill vi ge till Magnus Lindén och Johan Rubensson på Consat som hjälpt oss med konstruktion och feedback. Vi vill även tacka våra handledare på Chalmers och Consat, Göran Hult respektive Jonas Williamsson.

Göteborg, juni 2016

John Persson
Victoria Peterson

Sammanfattning

För att spara in på utvecklingskostnader automatiseras idag de tester som utförs i allt större utsträckning inom industrin. Consat Engineering AB som projektet utförs åt har idag en hög grad av automatisering vad gäller mjukvarutest via olika kommunikationsgränssnitt av styrenheter. Däremot saknas möjligheten att automatiskt variera matningsspänningen till den styrenhet som skall testas. Tanken med detta projekt var att skapa en testanläggning som automatiskt kan variera matningsspänningen utifrån användarens önskemål. Arbetet har resulterat i en fungerande testanläggning samt denna rapport som behandlar arbetsgången från förstudie till färdig anläggning. Genom att koppla in en spänningskälla som försörjer anläggningen och sedan koppla denna till enheten som ska testas kan användaren skicka ett meddelande via CAN-bussen till testanläggningen med ett önskat spänningsvärde. Kretsen reglerar sedan spänningen till detta värde och klarar av att mata ut 0-32 V med en noggrannhet på 100 mV samt att driva enheter upp till 2 A. För att enkelt kunna se att spänningen som matas ut är vad användaren önskat görs en avläsning av spänningen som sedan skrivs ut på en display samt skickas tillbaka via CAN för eventuell loggning.

Abstract

The tests to verify hardware are today often executed automatically. The automatization regarding tests at Consat Engineering AB, where the project has been performed for, are today widely implemented. However, the possibility to automatically vary the supply voltage to the test object for testing is missing. This report covers the working process from pre study to a working product. The aim of this project was to create a module that could vary the supply voltage based on the user's requests. By connecting a supply voltage to the module and connect the output to the test unit, the user can send a CAN message to the module containing the desired voltage value. The module is then adjusting the output voltage to this value with the precision of 100 mV up to 32 V. To ensure that the output voltage is equivalent to the requested one, the voltage is measured by the microcontroller and displayed on an LCD and sent back to the user by CAN for logging.

INNEHÅLL

Beteckningar

A/D	Analog-to-digital
CAN	Controller Area Network
ESD	Electro Static Discharge
LCD	Liquid Crystal Display
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
OP	Operational Amplifier
PIC	Peripheral Interface Controller
PWM	Pulse Width Modulation
TVS	Transient Voltage Suppressor
Chalmers	Chalmers Tekniska Högskola
Consat	Consat Engineering AB

1. INLEDNING

Detta projekt handlar om att konstruera hårdvara och mjukvara som utökar automatiseringen av hårdvarutester för styrenheter åt Consat Engineering AB. Nedan följer en mer detaljerad redogörelse av projektet.

1.1. Bakgrund

Consat är en ingenjörsbyrå verksamma inom industriell automation, elektronik- och systemutveckling, telematik samt miljö- och energiteknik. Företaget utvecklar bland annat mjukvaror till olika styrenheter inom fordonsindustrin där det ställs mycket hårda krav på testning. Det satsas därför mycket på effektivisering och kvalité kring detta, där ett steg är att automatisera testerna i så hög utsträckning som möjligt.

Consat har idag en hög grad av automatisering vad gäller mjukvarutest via olika kommunikationsgränssnitt av styrenheter. Däremot saknas möjligheten att automatiskt variera matningsspänningen till den styrenhet som skall testas.

1.2. Syfte

Syftet med projektet är att konstruera en hård- samt mjukvara som automatiskt skall variera matningsspänningen till den styrenhet som testas. Denna funktion skall komplettera nuvarande testutrustning där tanken vid projektslut är att Consat skall ha möjlighet att utföra tester av styrenheter helautomatiskt.

1.3. Avgränsningar

Examensarbetet är begränsat till 15 högskolepoäng vilket motsvarar cirka 10 arbetsveckor. I och med tidsbegränsningen kommer endast en prototyp att konstrueras men i mån av tid kan prototypen komma att vidareutvecklas. Det tas inte heller hänsyn till några ekonomiska aspekter i projektet.

1.4. Precisering av uppgift

I projektet skall en befintlig testanläggning kompletteras med ytterligare en funktion. Denna lösning ska automatiskt kunna variera matningsspänningen till en styrenhet. Funktionen kommer innefatta både mjukvara och hårdvara för att kunna utföra detta. Spänningen som ska skickas ut bestäms av användaren som skickar ett meddelande till testanläggningen. En kravspecifikation kommer att upprättas och utefter denna skall det sedan undersökas och bestämmas vilka komponenter som skall användas. Arbetet består således av en teknisk förstudie av nuvarande testanläggningen och potentiella lösningsförslag samt ett utvecklings-arbete av den hård- och mjukvara som tagits fram i den tekniska förstudien. Arbetet redovisas i form av ett prototypkretskort, teknisk rapport samt muntlig redovisning.

2. TEKNISK BAKGRUND

Här presenteras alla teknisk data som använts eller undersökts i projektet och som behövs för att få förståelse för rapportens innehåll.

2.1. BUSMASTER

Busmaster är ett gratis open source-program som kan användas för att simulera bland annat CAN-kommunikation. Med detta kan man skicka och ta emot meddelande samt läsa av eventuella felmeddelanden på bussen. (Kvaser, 2014)

2.2. CAN

CAN, Controller Area Network, är en typ av busskommunikation som lanserades under början på 80-talet av Bosch och används främst inom fordonsbranschen.

I CAN förekommer fyra olika meddelandetyper: datameddelande, meddelandebegäran, felmeddelande och kömeddelande. Ett datameddelande är ren data som skickas, en meddelandebegäran efterfrågar ett meddelande från en annan nod, ett felmeddelande sänds om en nod upptäcker ett fel på bussen och ett kömeddelande skickas om noden får för mycket data på en gång. Dessa datatyper är i sin tur uppdelade i flera delar. Ett datameddelande består exempelvis av tolv olika delar varav en är meddelandets identifierare och en annan är själva meddelandet.

Det speciella med CAN är att det inte finns någon så kallad master eller slave i systemet. Detta gör att vilken nod som helst får börja sända ett meddelande om bussen är ledig. Skulle flera noder börja sända samtidigt jämförs alla meddelandens identifierare för att avgöra prioriteten. Noden med högst prioritet får sedan fortsätta sända medan de andra måste vänta på sin tur. Till skillnad från andra bussystem används inga adresser, utan det är identifieraren som bestämmer vilka noder som ska ta emot meddelandet. Detta medför att flera noder kan ta emot samma meddelande och eftersom noderna är så pass smarta kan de själva avgöra om meddelandet ska tas emot eller ej. En stor fördel med CAN är att man sparar mycket kablage i och med att det endast krävs två kablar för att koppla ihop en nod med en annan. En annan fördel är att det är mycket enkelt att ansluta nya enheter på bussen eftersom noden inte behöver anslutas enskilt till alla de andra noderna, utan det är bara att koppla in den på bussen. (National Instruments, 2014)

2.3. Lågpassfilter

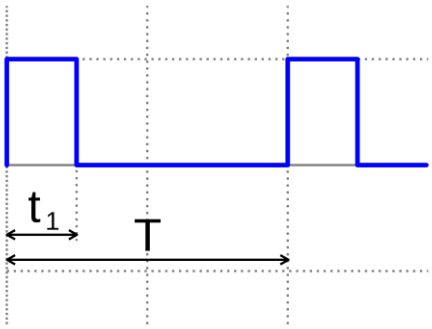
Detta är ett filter som dämpar signaler över en viss bestämd frekvens och släpper igenom signaler under denna frekvens. Det används ofta när man vill omvandla en PWM-signal till en analog spänning. Filtret byggs oftast upp med hjälp av resistorer och kondensatorer. (Texas Instruments, 2008)

2.4. MPLAB

MPLAB är en utvecklingsmiljö från Microchip gjord för deras PIC mikrokontroller. Där finns det möjlighet att konfigurera vilken PIC man programmerar och får då hjälp från programmet med vilka register och kommandon som finns att tillgå. (MPLAB, 2016)

2.5. PWM

Pulsbreddsmodulering (Pulse Width Modulation) används ofta som styrsignal för exempelvis elmotorer. En PWM-signal ser ut som en fyrkantsvåg och skiftar mellan hög och låg signal så snabbt att apparaten som styrs tolkar det som en jämn spänning.



Figur 2.1 – PWM-signal

För att variera denna spänning regleras tillslagstiden, t_1 enligt *Figur 2.1*, alltså den tid fyrkantsvågen är hög. Är tillslagstiden 100% fås max spänning, till exempel 5 V. Är tillslagstiden istället 30% blir spänningen istället 1,5 V om max spänning är 5 V. För att använda PWM-signalen för att göra en D/A-omvandling måste signalen gå genom ett lågpassfilter, detta för att filtrera bort de höga frekvenserna och få kvar medelspänningen. (Wikipedia, 2016)

2.6. Schottkydiod

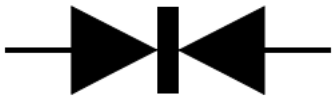
Schottkydioder är en halvledardiod med ett lägre framspänningsfall än många andra dioder.

Figur 2.2 – Symbol för Schottkydiod

Detta leder bland annat till lägre effektförluster och snabbare switchhastigheter vilket gör den lämpad för kretsar med högre frekvenser. Dess snabba switchegenskaper utnyttjas även vid polvändningsskyddsapplikationer då dioden oftast sitter i serie med en säkring och vid fel polaritet leder en stor ström som utlöser säkringen och skapar ett avbrott i kretsen innan någon skada är skedd. (Future Electronics, 2016)

2.7. TVS-diod

TVS-dioder är en typ av diod som används till att skydda känsliga elektriska kretsar eller komponenter mot överspänning, spänningsspikar och ESD (Semtech, 2000).

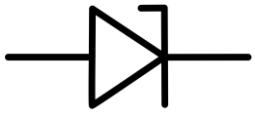


Figur 2.3 – Symbol för en dubbelriktad TVS-diod

Genom att begränsa spänningen vid ett visst värde, så kallad ”breakdown voltage”, och börja leda ström ner till jord skyddar den kretsen mot för höga värden. En stor fördel gentemot andra dioder är dess extremt snabba operationshastighet. Den börjar leda så pass snabbt att det i de flesta datablad är uppgett som att den är extremt snabb snarare än att ett faktiskt siffervärde uppges. TVS dioder är mycket vanligt inom fordonsindustrin av just denna anledning men använda även av hobbyelektriker i kretsar med ESD känsliga komponenter. Dom förekommer både som enkel- och dubbelriktade dioder. (Vishay, 2010)

2.8. Zenerdiod

Zenerdioder har till skillnad från många andra dioder en väl definierad genombrottsspänning, den så kallade Zenerspänningen, där dioden låter ström flyta även i backriktningen. Då Zenerspänningen hålls ganska konstant är Zenerdioder väl lämpad i kretsar för spänningsstabilisering. (Future Electronics, 2012)



Figur 2.4 – Symbol för Zenerdiod

3. METOD

Arbetet inleds med att utforma en grov tidsplan i form av ett Gantt-schema där arbetet delas in i olika kategorier så som informationsinhämtning, konstruktion och dokumentation. Sedan bryts den ner i mindre punkter som kan läggas in i ett ärendehanteringssystem för att eftersträva ett mer verkligt arbete. Efter detta påbörjas arbetet med att analysera den nuvarande testanläggningens funktion samt att kartlägga de olika funktionerna som testanläggningen ska kunna utföra. Utifrån dessa uppgifter ska en kravspecifikation tas fram för hårdvara samt mjukvara och därefter kan informationssökningen påbörjas.

När tillräckligt med information inhämtats startar planeringen av hur de olika delarna i uppgiften ska utföras hårdvarumässigt. Eftersom många bitar förmodligen kan lösas på flera sätt måste flertalet urval ske för att få fram en slutgiltig lösning för varje funktion.

Sedan ska de olika komponenterna dimensioneras och ritas upp på ett kretsschema. Parallellt med detta ska även mjukvaruplaneringen påbörjas. När hårdvaruplaneringen för några av funktionerna är klar ska komponenterna beställas så att mjukvaran kan testas och vidareutvecklas. Detta för att arbetet skall kunna utföras parallellt i så stor utsträckning som möjligt.

De enklare funktionerna ska implementeras först, till exempel skrivning till displayen. Detta är mycket till hjälp vid testning och felsökning för resten av funktionerna. Därefter ska de resterande funktionerna planeras och dimensioneras för att göra konstruktionen komplett. Sedan ska konstruktionen testas och verifieras innan allt flyttas från kopplingsdäcket till det slutgiltiga prototypkretskortet.

4. FÖRSTUDIE

För att kunna utföra arbetet krävs en mer specifik formulering av problemet samt vilka krav som ska ställas på den slutgiltiga lösningen. Denna information hämtades från samtal med Peter Brunnsåker på Consat som regelbundet använder sig av testanläggningen.

4.1. Analys av nuvarande testanläggning

Dagens testanläggning består av ett antal reläer som kan slås på och av för att skapa kortslutning mellan olika pin-par. Detta görs för att man bland annat ska kunna kontrollera att rätt felmeddelande ges från kretskortet som testas. Kommandot som bestämmer vilka pin-par som ska kortslutas skickas via CAN till en Arduino som sedan styr reläerna på anläggningen. Funktionen som idag saknas i anslutning till testanläggningen är att automatiskt kunna variera matningsspänningen till kretskortet som testas. I dagsläget utför man detta genom att manuellt ställa spänningen med hjälp av vredet på ett spänningsaggregat till önskad nivå.

4.2. Precisering av uppgiften

Användaren ska kunna skicka ett kommando med ett önskat spänningsvärde till testanläggningen. Denna tar emot meddelandet och ska sedan reglera spänningen till önskad nivå. Eftersom den tidigare anläggningen redan använder sig av CAN är det fördelaktigt att utbyggnaden också gör det för att förenkla testningen. För att kontrollera att rätt spänning skickas ut ska värdet läsas av för att sedan återges på en display, värdet ska också skickas tillbaka via CAN för att testningen skall kunna loggas.

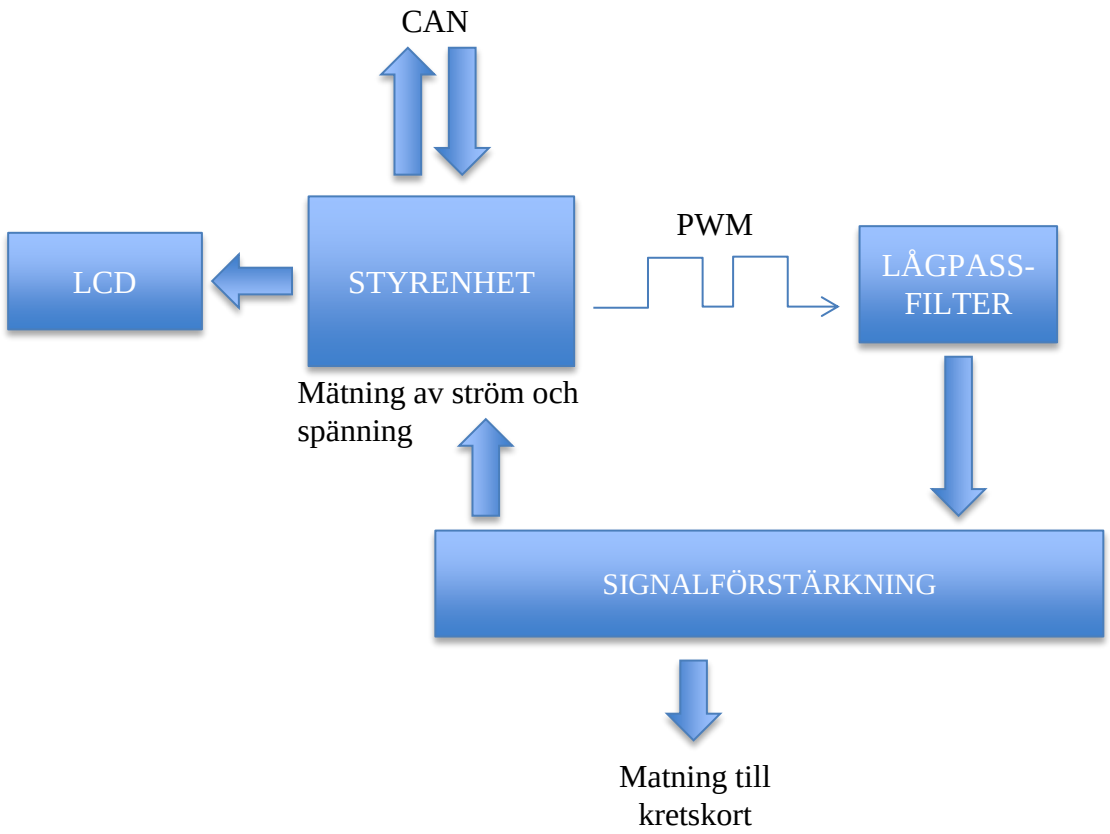
4.3. Kravspecifikation

Nedan följer de specifika krav som ställts på anläggningen.

Nr	**K/Ö**	**Kriterier**	**Kommentar**
1	Krav	CAN-kommunikation	Skicka och ta emot meddelande
2	Krav	Display	Visa aktuellt spänning
3	Krav	Reglera spänning	Noggrannhet 100 mV upp till 20 V
4	Krav	Läsa av utgående spänning	Noggrannhet 100 mV
5	Krav	Polvändningsskydd	
6	Krav	Skydd mot överspänning	
7	Krav	Slutgiltig lösning på prototypkretskort	
8	Önskemål	Reglera spänning	Noggrannhet 100 mV upp till 32 V
9	Önskemål	Läsa av utgående ström	Noggrannhet 10 mA
10	Önskemål	Slutgiltig lösning på riktigt kretskort	
11	Önskemål	Förpacka hårdvara	

5. LÖSNINGSFÖRSLAG

Redan i projektets start bestämdes tillsammans med handledaren på Consat det övergripande sättet att lösa problemet på. Detta sätt var att styra spänningen med en PWM-signal från en styrenhet. Anledningen till detta lösningsförslag var för att få en teknisk höjd på projektet. Problemet hade kunnat lösas med exempelvis ett programmerbart spänningsaggregat men detta ansågs vara en för enkel lösning och valdes därför bort. Då endast slutfunktionen är av vikt för Consat kommer lösningsförslagen istället att ligga mer på komponentnivå. En övergripande bild av den lösning som arbetats fram kan ses i *Figur 5.1*. För ett detaljerat kretsschema över lösningsförslaget se *bilaga 10*.



Figur 5.1 – Blockschema för testanläggningen

5.1. Styrenhet

Inför detta val listades ett antal krav upp som enheten måste uppfylla, dessa var följande:

- Stöd för CAN
- PWM-utgång
- Minst 20 ben för inkoppling av diverse komponenter och funktioner
- A/D-omvandlare

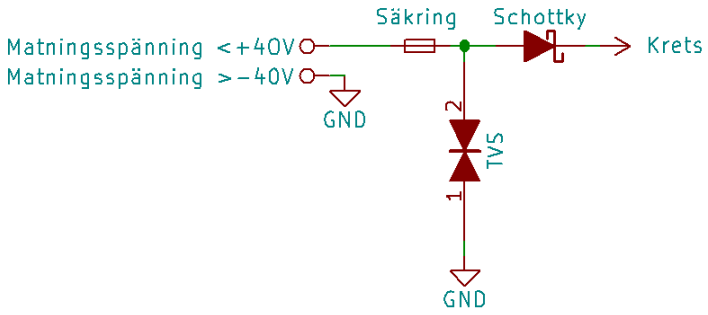
De förslag som först kom upp var Arduino, i och med att den befintliga testanläggningen styrs av en sådan. Det andra alternativet var PIC från Microchip, på grund av tidigare erfarenheter av denna typ. Av samma anledning som nämndes i föregående avsnitt valdes därför PIC. Dels för att mycket redan är implementerat i den befintliga Arduinon, till exempel CAN-kommunikationen vilket leder till att väldigt lite mjukvara behöver utvecklas. Dessutom krävs lite fler kringkomponenter vid användandet av PIC vilket var något som önskades. I mån av

tid ska dock den befintliga mjukvaran i Arduinon slås samman med den nya mjukvaran i antingen PIC:en eller Arduinon.

Nästa steg var att hitta en PIC som passar för ändamålet och som fyller ovanstående krav. I och med att det finns en uppsjö av enheter som uppfyller dessa krav avgränsades urvalet till 8-bit mikrokontrollers av familjen PIC18 vilket gav ett urval på 18 stycken enheter. Till en början valdes den som var billigast, men vid beställningstillfället upptäcktes det att denna inte fanns i lager under de närmsta veckorna så därför föll valet på en snarlik processor, PIC18F2580.

5.2. Polvändings- och överspänningsskydd

För att skydda kretsen mot felkoppling och för höga matningsspänningar konstruerades ett polvändnings- respektive överspänningsskydd. Då kretsen innehåller två olika referensspänningar konstruerades två skydd men med olika dimensioner på komponenterna. Schottkydioden agerar i det här fallet polvändningsskydd och stryper om kontakterna kopplas in fel och leder därmed ingen ström eller spänning vidare till kretsen. TVS-dioden är överspänningsskyddet som vid för höga spänningar blir ledande i backriktningen och utlöser säkringen. Anledningen till att en TVS-diod valdes istället för en annan typ diod, till exempel Zenerdiod som är vanligt förekommande för denna applikation, var för dess snabba operationshastighet och låga spänningsfall kontra andra dioder. Det blir dock ett spänningsfall på omkring 300-400 mV i denna konstruktion men detta anses godtagbart.



Figur 5.2 – Konstruktion av skydd

5.3. Referensspänning till krets

Då kretsen behöver en matningsspänning som är minst lika stor som det värde användaren önskar samtidigt som många komponenter i kretsen endast tål spänningar på upp till 5 V krävs en separat referensspänning på 5 V till dessa komponenter. Det behövs även en 12 V referensspänning för att försörja fläkten till kylningsanordningen, se *kapitel 6.1.10*. Eftersom det endast kommer finnas en matningsspänning till kretskortet behövs en spännings-omvandling från matningsspänning till referensspänning på kretskortet. Omvandlingen kommer ske under höga och varierande matningsspänningar men ska ändå kunna leverera en stabil referensspänning på 5 respektive 12 V med möjlighet att kunna driva enheter upp till 1 A. Två alternativ undersöktes under detta projekt.

5.3.1. Alternativ 1 – Spänningsdelning

Via en spänningsdelning från matningsspänningen kan en referensspänning på 5 respektive 12 V uppnås. Denna lösning är dock inte så flexibel eftersom den alltid kräver ett konstant värde på matningsspänningen.

5.3.2. Alternativ 2 – Spänningsregulator

Med en spänningsregulator anpassad för höga och varierande matningsspänningar kan en hög noggrannhet på referensspänningen uppnås oberoende av matningsspänningens variationer.

Alternativ 2 är den lösning som anammades i detta projekt på grund av dess flexibilitet på matningsspänningen vilket ökar användarvänligheten.

5.4. Programmerare

För att kunna överföra mjukvara till processorn behövdes en programmerare. En PicKit 3 var ett alternativ som snabbt dök upp då denna var enkel och lätt att använda och inte krävde någon omfattande hårdvaruuppkoppling. PicKit 3 gör det också enkelt att framöver programmera om processorn om så skulle behövas. Detta blev också den lösning som användes i projektet utan att andra egentliga alternativ undersöktes.

5.5. CAN-kommunikation

PIC-processorn i projektet har en inbyggd CAN-modul men behöver utöver detta även en CAN-transceiver för att sköta CAN-kommunikationen. Då PIC:en redan är från Microchip valdes CAN-transceivern ut från samma tillverkare för att minska urvalet. MCP2561 valdes eftersom det är en vanligt förekommande modul som är relativt enkel att koppla upp.

5.6. Display

För displayen fanns inte några specifika krav, mer än att det ska finnas plats för det som ska skrivas ut. Trots att det inte tas någon hänsyn till ekonomiska aspekter så valdes displayen baserat på pris i och med att displayer är dyra. Valet föll därför på en enkel alfanumerisk display med 2x16 tecken av modellen DEM 16226 SYH-LY från Display Elektronik. Displayen kan även framöver användas för att kommunicera felmeddelanden för till exempel för låga värden på matningsspänningen.

5.7. Lågpassfilter

Lågpassfiltret konstruerades som ett RC-filter där en resistor kopplas i serie med lasten och en kondensator kopplas parallellt med dessa ner till jord. Detta är en metod som tidigare använts och därför undersöktes inga andra alternativ. Dess funktion är att filtrera bort störningar och jämna ut PWM-signalen som kommer ut från PIC:en innan den skall förstärkas.

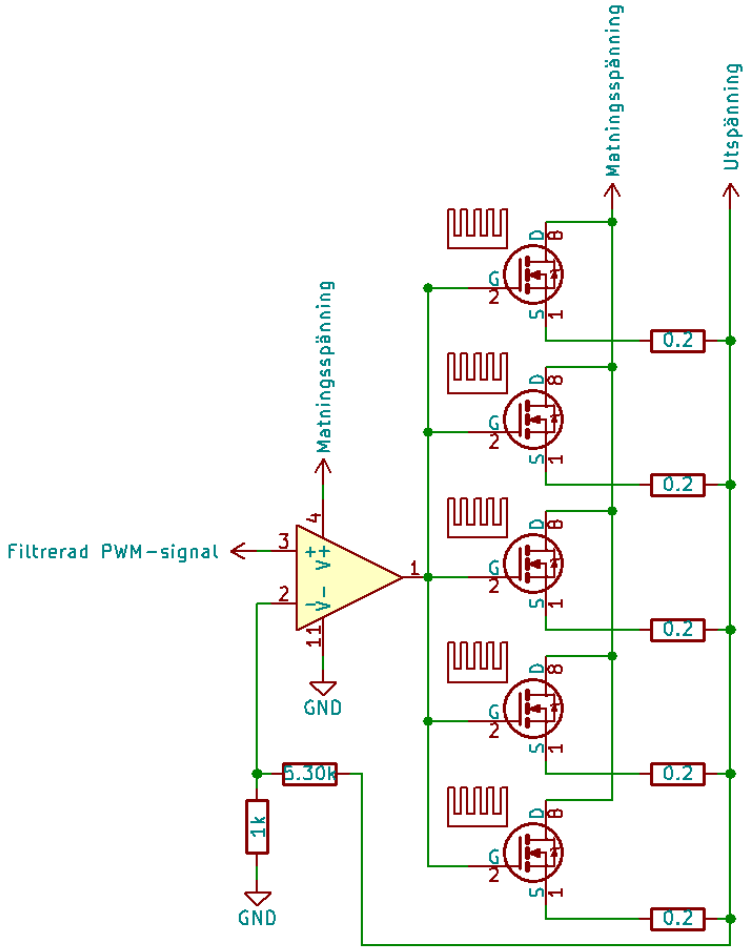
5.8. Signalförstärkning

Efter att PWM-signalen filtrerats skall signalen förstärkas och regleras till det önskade värdet. Spänningen skall förstärkas från 0-5 V till 0-32 V och strömmen skall förstärkas så anläggningen kan driva upp till 2 A. Under projektet testades två olika lösningar att lösa problemet på där båda bestod av en OP som reglerare och signalförstärkare följt av parallellkopplade MOSFET-transistorer som ytterligare signalförstärkning. Anledningen till att MOSFET-transistorer valdes var främst för att dessa är spänningsstyrda och därmed kunde styras direkt från OP:ns utsignal.

5.8.1. Lösningsförslag 1 – OP + N-kanals MOSFET-transistorer

OP:n får den filtrerande PWM-signalen på positiva ingången och skickar denna signal vidare genom fyra parallellkopplade N-kanals MOSFET-transistorer vars uppgift är att förstärka strömmen. Signalen efter transistorerna återkopplas sedan via en spänningsdelning med en

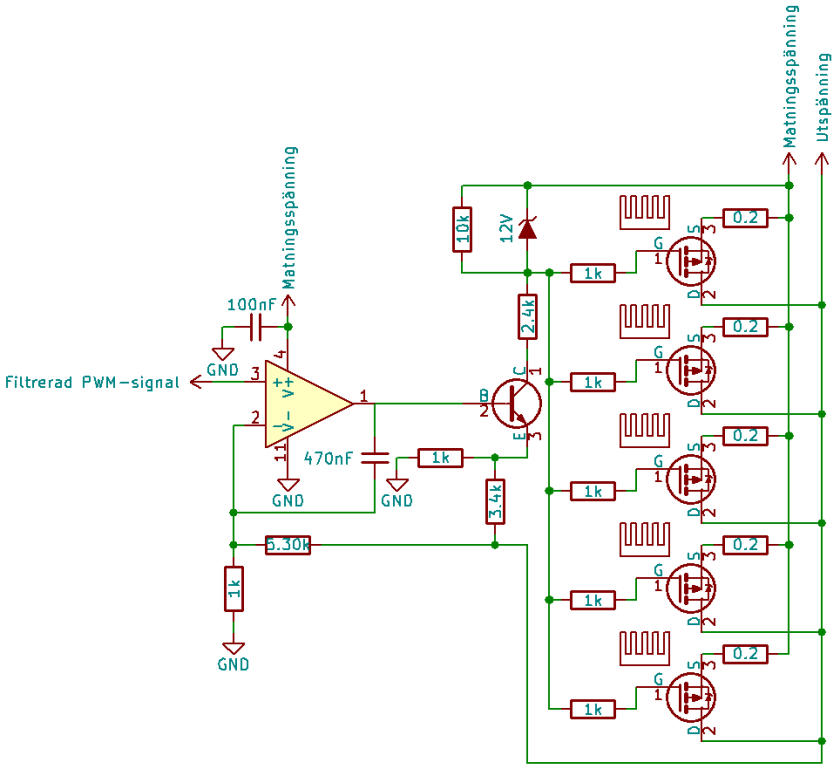
faktor på 6,4 in på OP:s minusingång för att reglera signalen till rätt spänningsnivå, *se Figur 5.3*



Figur 5.3 – OP + N-kanals MOSFET-transistorer

5.8.2. Lösningsförslag 2 – OP + P-kanals MOSFET-transistorer

OP:n får den filtrerade PWM-signalen på positiva ingången och skickar denna signal vidare till basen på en NPN-transistor som skickar signalen vidare in på gate till fem parallellkopplade P-kanals MOSFET-transistorer. Signalen efter transistorerna återkopplas sedan två gånger, först en inre regleringsloop via en spänningsdelning med kvoten 4,3 in på NPN-transistorns emitterben och en yttre regleringsloop via en spänningsdelning med en kvot på 6,4 in på OP:ns minusingång för att reglera signalen till rätt nivå, *se Figur 5.4.*



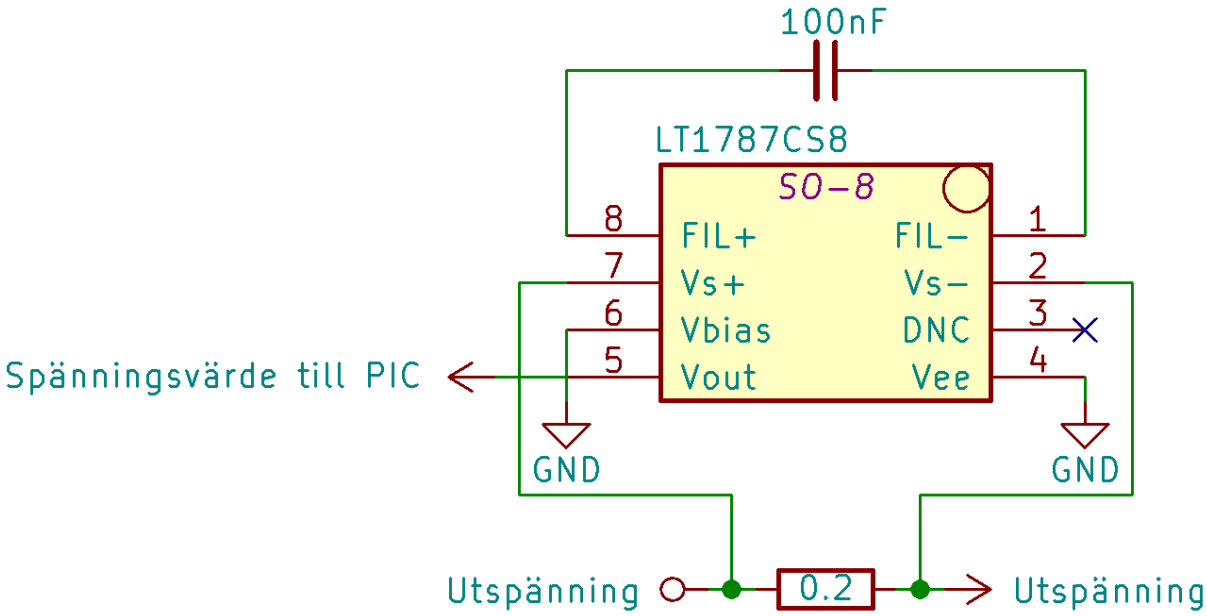
Figur 5.4 – OP + P-kanals MOSFET-transistorer

Först användes alternativ 1 som lösning varefter alternativ 2 ersatte denna och blev lösningen som monterades på det slutgiltiga kretskortet. Läs *kapitel 6.1.8* för händelseförlopp.

5.9. Spännings- och strömmätning till PIC:en

För att kunna återge de exakta värdena på utspänningen och drivströmmen till testobjektet på displayen samt skicka tillbaka dem via CAN för eventuell loggning krävs en återmatning till PIC:en. Då PIC:en har en inbyggd A/D-omvandlare användes denna till att för av spänningen. Såväl utspänningen som drivströmmen har för höga värden för att direkt återmatas till PIC:en och då måste spänningen och strömmen anpassas till de nivåer som PIC:en klarar av utan att störa noggrannheten i mätningen.

För att återge spänningsnivån används en spänningsdelning med en kvot på 8. Detta då PIC:en endast tål spänningar upp till 5 V och den maximala utspänningen på 32 V blir 4 V in till PIC:en. Anledningen till att kvoten är satt högre än nödvändigt är för att den maximala matningsspänningen till kretsen är begränsad till 40 V och skulle vid ett eventuellt hårdvarufel kunna levereras som utspänning om transistorerna leder fullt ut. Detta skulle i sådana fall inte skada PIC:en då 40 V skulle delas ner till 5 V med en kvot på 8 vilket PIC:en klarar av att hantera.



Figur 5.5 – Lösningsförslag, strömmätning

För att återge drivströmmen till testobjektet krävs en separat strömmätning. Genom att använda en strömavkänningsförstärkare som mäter spänningsfallet över ett strömavkänningsmotstånd uppkopplad i anslutning till utspänningen kan värdet kalkyleras i PIC:en till motsvarande strömvärde. Mätningen fås med en noggrannhet på 10 mA vilket motsvarar kraven i *kapitel 4.3*. En strömavkänningsförstärkare LT1787CS8 från Linear Technology valdes som lösning (Linear Technology, 1999).

5.10. Kylning

Kylning för delar av kretsen blev en nödvändighet då det i både 5 V-regulatorn och MOSFET-transistorerna förekommer höga effektförluster på grund av det stora spänningsfallet över dessa i kombination med de höga strömmarna på upp till 2 A. Den enda kylningsmetoden som på förhand var känd och som undersökts var att med hjälp av kylfläns avleda dessa effektförluster, detta är också den standardlösning som används inom elektronik. En kylfläns med fläkt kom i slutändan att användas i detta projekt efter att den tänkta kylflänsen som valts ut visade sig vara för underdimensionerad. Detta efter ett beräkningsfel, se *kapitel 6.1.10*.

5.11. Prototypkretskort

Den slutgiltiga monteringen skulle enligt kravspecifikationen i *kapitel 4.3* ske på ett prototypkretskort. Ett prototypkretskort av standardstorleken 100x160 mm från Roth Elektronik av Epoxipapper med kopparbeläggning valdes ut som lösning.

5.12. Förpackning

I slutfasen av projektet konstaterades att en förpackning av hårdvaran var nödvändig ur både användarvänlighets- och säkerhetssynpunkt men även för att skydda elektroniken på kretskortet. Det finns en rad olika förpackningslådor för småelektronik att tillgå och det togs fram två olika alternativ som diskuterades med personal på Consat.

5.12.1. Alternativ 1 – Aluminiumlåda

En aluminiumlåda har naturligt bra kylningsegenskaper vilket skulle ställa mindre krav på kylflänsen, dock kan temperaturen på lådan bli problematisk ur användarsynpunkt. Lådan riskerar också att bli elektriskt ledande vilket skulle medföra en allvarlig risk vid felanvändning.

5.12.2. Alternativ 2 – Plastlåda

En plastlåda riskerar varken att bli för varm eller elektriskt ledande, dock ställs större krav på kylflänsen vilket troligtvis medför större dimensioner på låda och kylfläns. Lådan är dock, tack vare dess mjuka materialegenskaper, enklare att modifiera och anpassa till anläggningen under konstruktionsarbetet.

Alternativ 2 är den lösning som valdes att gå vidare med i detta projekt då dess egenskaper bidrar till en säkrare arbetsmiljö och då inga krav på dimensioner finns har detta alternativ ingen egentlig nackdel.

6. GENOMFÖRANDE

I detta avsnitt beskrivs det praktiska genomförandet av arbetet, såväl konstruktionen av hårdvaran som utvecklingen av mjukvaran.

6.1. Konstruktion av hårdvara

Under projektets gång har uppkopplingen skett på ett kopplingsdäck för att testa och verifiera alla lösningar innan de slutligen monterats på ett prototypkretskort. Detta gjordes främst för att enklare kunna modifiera och ändra i kretsen under arbetes gång.

6.1.1. Styrenhet

PIC:ens grundkopplingar MCLR, V_{dd} och V_{ss} kopplades upp enligt datablad (Microchip, 2009) varav funktionen sedan verifierades. Oscillatorn dimensionerades efter den valda klockfrekvensen 8 MHz. Två kondensatorer valdes ut efter databladet och en kristall på 8 MHz monterades samt mättes med oscilloskop för att verifiera att rätt frekvens uppnåddes.

6.1.2. Polvändnings- och överspänningsskydd

Schottkydioden monterades på kopplingsdäcket och testades först med polerna rättvända för att sedan kopplas med felvända poler för att verifiera dess funktion. Detta fungerade precis som beräknat.

TVS-dioden och säkringen monterades även de på kopplingsdäcket för att testas. Trots att en supersnabb säkring var inköpt hann TVS-dioden gå upp till 45 V innan säkringen brann upp fastän tanken var att TVS-dioden skulle lösa ut säkringen vid 40 V, det beror troligtvis på att TVS-dioden inte är tillräckligt snabb. En TVS-diod av samma fabrikat fast med andra parametrar testades istället och utlöste säkringen vid 41 V varav denna valdes istället då 41 V inte kan göra någon skada på kretsen.

De båda skydden kopplades sedan upp tillsammans för att se att de inte påverkade varandras funktion negativt vilket de heller inte gjorde. Sist monterades de i anslutning till kretsen för att se att dessa inte påverkade kretsens funktion och det noterades då att det fungerade precis som beräknat.

6.1.3. Referensspänning till krets

Utifrån urvalet i *kapitel 5.3* kopplades en spänningsregulator av typen LM2936 från National Semiconductor (National Semiconductor, 2006) upp på kopplingsdäcket. Denna fungerade mycket bra tills kretsen utvidgades med början på förstärkningssteget. Då detta steg inte kunde leverera den utspänning som efterfrågades började kretsen att felsökas. Olika funktioner kopplades bort från kretsen för att se om någon av dessa gav några störningar och då upptäcktes att CAN-kommunikationen störde spänningsregulatorn som då inte gav önskad referensspänning till PIC:en. Det noterades att den dåvarande spänningsregulatorn endast kunde driva enheter upp till 50 mA vilket var för lite för att driva hela kretsen. Denna byttes tillfälligt ut mot en LM7805 från Fairchild (Fairchild, 2014) varefter CAN-kommunikationen och den övriga kretsen inte uppvisade några störningar. Eftersom denna regulator endast tål matningsspänningar på upp till 35 V var detta inte en regulator som planerades att ha i slutversionen men det fungerade tills vidare för att fortsätta driva utvecklingsarbetet framåt.

Efter en del research valdes en LM317HV från Texas Instrument (Texas Instruments, 2000) som regulator i den slutgiltiga versionen då den kan hantera matningsspänningar på upp till

60 V och samtidigt leverera referensspänningar på mellan 1,25 till 57 V med en noggrannhet på 0,01%, samt driva strömmar upp till 1,5 A. referensspänningen regleras enkelt med två motstånd som kopplas upp omkring regulatorn. Detta gör även LM317HV till en väldigt flexibel regulator som kan användas både som 5 V- och som 12 V-referensspänningskälla.

5 V-regulatorn kopplades upp enligt datablad med en keramisk kondensator på 100nF samt en elektrolyt på 10µF. Två resistorer kopplades upp likt en spänningsdelning och dimensionerades till $R_1 = 240\ \Omega$ respektive $R_2 = 733\ \Omega$ efter beräkningar och kalibrering.

$$V_{ref.\ 5\,V} = 1{,}25 * \left(1 + \frac{R_2}{R_1}\right) + I_{adj} * (R_2) \qquad (6.1)$$

R_1 angavs i databladet till 240 Ω och I_{adj} utlästes vid 25 °C till 53 µA (Texas Instruments, 2000) medans R_2 dimensionerades enligt *ekvation 6.1*. Denna ekvation gav att $R_2 = 715\ \Omega$ vilket resulterade i en utspänning på 4,89 V. Detta orsakade problem för PWM-signalen som helst behöver 5V som spänningsreferens för att kunna skicka ut önskat värde. Därför kalibrerades R_2 om till 733 Ω vilket gav en referensspänning på 5 V + < 0,25 % som en godtagbar felmarginal.

Dimensioneringen av 12 V-regulatorn följde samma procedur där R_2 först beräknades till 2,05 kΩ men sedan kalibrerades till 2,1 kΩ för att ge 12 V referens.

6.1.4. Programmerare

Den valda programmeraren behövde endast en anslutningskontakt samt sammankoppling med MCLR-porten och berörda programmeringsportar på PIC:en (Microchip, 2013). En 6-polig stiftlist kopplades upp och kommunikationen med PIC:en testades. Denna kontakt valdes sedan att inte monteras som anslutningsmöjlighet på utsidan av lådan då stiftlisten skulle kunna ta skada vid ytlig montering. Det finns dock en möjlighet till framtida mjukvaruuppdateringar eftersom kontakten finns kvar på kretskortet.

6.1.5. CAN-kommunikation

Can-transceivern kopplades upp enligt datablad (Microchip, 2013) och en D-subkontakt monterades. De två 60 Ω resistorerna agerar termineringsmotstånd och visade sig i efterhand vara överflödiga då Consat har en adapter för detta som monteras mellan honan och hanen på D-subkontakten. Dessa motstånd fick dock sitta kvar för att eliminera riskfaktorn ifall dessa adaptrar inte alltid finns tillgängliga. Funktion störddes inte heller av att ha dubbelterminering och därför kommer inte Consat att påverkas av detta när de följer sina normala rutiner.

6.1.6. Display

Displayen kopplades upp enligt databladet (DISPLAY Elektronik GmbH, 2008) där alla 8 programmeringsben kopplades till PIC:en för att testa dess funktion och även testa den dittills utvecklade mjukvaran. Efter hand utvecklades mjukvaran ytterligare och därmed kunde LCD kopplas om så att endast 4 programmeringsbara ben användes vilket frigjorde mer plats på kretskortet och möjliggjorde att fler funktioner kan kopplas till PIC:en framöver. Till en början kopplades inte LED-bakgrundsljuset in. Det ansågs inte behövas men efterhand som förutsättningarna för förpackningen av hårdvaran förändrades valdes denna funktion att läggas till för att förbättra användarvänligheten.

6.1.7. Lågpassfilter

Lågpassfiltret är av RC-karaktär och är kopplad direkt efter det ben där PWM-signalen skickas ut från PIC:en. Vid uppkoppling noterades att signalen inte blev tillräckligt stabil efter ett filter. Ett ytterligare filter sattes då i serie med föregående filter vilket gav en stabil signal. Storleken på resistorerna sattes till 2,1 kΩ varefter kondensatorernas storlek räknades ut till 1 µF med hjälp av *ekvation 6.2* och *ekvation 6.3.*

$$PWM_{period} = (PR2 + 1) * 4 * T_{osc} * TMR2\ Prescale\ value \quad (6.2)$$

där

$$T_{osc} = \frac{1}{F_{osc}}$$
$$PR2 = 124$$
$$TMR2\ Prescale\ value = 4$$

$$PWM_{period} = (124 + 1) * 4 * \frac{1}{8 * 10^6} * 4 = 250\ \mu s$$

$$\left.\begin{matrix} \tau = RC \gg PWM_{period} \\ R = 2{,}1\ k\Omega \end{matrix}\right\} \rightarrow C \geq 119\ nF \rightarrow C * \sim 9 \rightarrow C = 1\ \mu F \quad (6.3)$$

6.1.8. Signalförstärkning

Utifrån *kapitel 5.8* började alternativ 1 att konstrueras. En OP av typen LT1413CN8 från Linear Technology (Linear Technology, 2013) som klarade spänningar på upp till 40 V installerades. Därefter kopplades en strömbegränsande resistor in i serie med fem stycken P-kanals MOSFET-transistorer av typen IRF9530NPBF från International Rectifier (International Rectifier, 2004) på OP:ns utgång, varefter en återkoppling till OP:n kopplades in med en förstärkning på 6,4 för att få rätt utspänning. Vid test av alternativ 1 kunde det konstateras att det blev ett spänningsfall på 5 V över transistorerna. Kretsen felsöktes därför grundligt innan det i samråd med handledare på Chalmers och personal på Consat kunde fastställas att detta spänningsfall kommer uppstå i en sådan här typ av lösning. Lösningen som sådan fungerande alltså, men ansågs inte tillräckligt bra då en betydligt högre matningsspänning än nödvändigt behövdes varav beslutet togs att gå vidare med alternativ 2 istället.

Efter uppkoppling av alternativ 2 testades kretsen och det kunde slås fast att lösningen medförde ett mycket lågt men godtagbart spänningsfall över transistorerna, dock betedde sig utspänningen inte som önskat. En mycket omfattande felsökning sattes igång där varje komponent och koppling granskades för att identifiera eventuella fel. När detta inte gav något resultat söktes hjälp från handledaren på Chalmers som kunde konstatera att systemet självsvängde vilket också hade misstänkts tidigare men som det inte hittats någon lösning på. Eftersom det finns flera förstärkningsfaktorer över OP:n, NPN-transistorn samt MOSFET-transistorerna finns det en hög risk att självsvängning kan uppstå och detta var något som inte reflekterats över när denna lösning konstruerades. En möjlig lösning på problemet instruerades av handledaren på Chalmers. Denna bestod av en kaskadreglering där utspänningen återkopplades i en inre loop till NPN-transistorn och en yttre loop som återkopplades till OP:n. En kondensator sattes även mellan OP:ns utgång och minusingång för

att stabilisera signalen. Efter dessa förändringar testades kretsen igen med lyckat resultat, se *Figur 7.1* – Test före kalibrering. Efter detta beslutades det att lösningen var godtagbar och kunde härifrån kalibreras för att nå mer exakta värden på utspänningen.

6.1.9. Spännings- och strömmätning till PIC:en

Efter slutsteget kopplades en spänningsdelare upp för att kunna avläsa den faktiska utspänningen till PIC:en, dels för reglering av utsignalen men även för skrivning till displayen och för att skickas vidare med CAN. Två resistorer av storleken 20 kΩ respektive 140 kΩ kopplades upp för att uppnå en kvot på 8 för att skicka en hanterbar signal till PIC:en. Anledningen till de höga värdena på resistorerna är för att de även ska vara strömbegränsande.

Tanken var även att ha en strömavkänningsförstärkare kopplad från utspänningen tillbaka till PIC:en. Men på grund av tidsbrist i slutet av projektet är denna krets ej uppkopplad hårdvarumässigt men mjukvaran är utvecklad och alla nödvändiga komponenter är beställda och finns för montering i ett senare skede.

6.1.10. Kylning

Från urvalet i *kapitel 5.10* skulle en kylfläns väljas ut som kunde avleda effektförlusterna från MOSFET-transistorerna i signalförstärkningen. Ur *ekvation 6.4* kunde förlusteffekten räknas ut och därefter kunde arbetet med att hitta rätt kylfläns börja.

$$P = U * I \rightarrow 32 * 2 = 64\ W \tag{6.4}$$

Den kylfläns som valdes var en SK481 100 mm från Fischer Elektronik med en termisk resistans på 2,8 °C/W (Fischer elektronik, 2014). Med hjälp av *ekvation 6.5* räknades den temperatur som skulle uppstå vid maximal effektförlust över en transistor ut. Denna dividerades sedan med kvoten 5 eftersom det är det antalet transistorer som parallellkopplas i lösningen och därmed så många transistorer som effekten fördelas över.

$$T_J = P * \frac{(R_{JC}+R_{HA}+R_{CH})}{5} + T_A = 64 * \left(\frac{1{,}32+2{,}8+0{,}25}{5}\right) + 25 = 80{,}936°C \tag{6.5}$$

$T_J = Temperatur\ i\ transistor$
$P = Effekt$
$R_{JC} = Termisk\ resistans\ transistor$
$R_{HA} = Termisk\ resistnas\ kylfläns$
$R_{CH} = Termisk\ resistans\ mellan\ transistor\ och\ kylfläns$
$T_A = Temperatur\ i\ rummet$

I dessa beräkningar begicks ett misstag där kylflänsens termiska resistans också dividerades med 5 vilket gav ett missvisande resultat eftersom effekten inte fördelas över 5 kylflänsar. Detta påpekades av handledaren på Chalmers varav en ny beräkning gjordes enligt *ekvation 6.6* och det konstaterades att kylflänsen inte skulle göra någon nytta då transistorerna skulle brinna upp vid dessa temperaturer.

$$TJ = 64 * \left(\frac{1{,}32+0{,}25}{5} + 2{,}8\right) + 25 = 224{,}296°C \tag{6.6}$$

En kylfläns med fläkt, A80856-001 från Intel, tillhandhölls då av Chalmers (Intel, 2014). Denna testades och visade sig avleda de förlusteffekter som uppkom varav denna senare användes i den slutliga lösningen.

6.1.11. Prototypkretskort

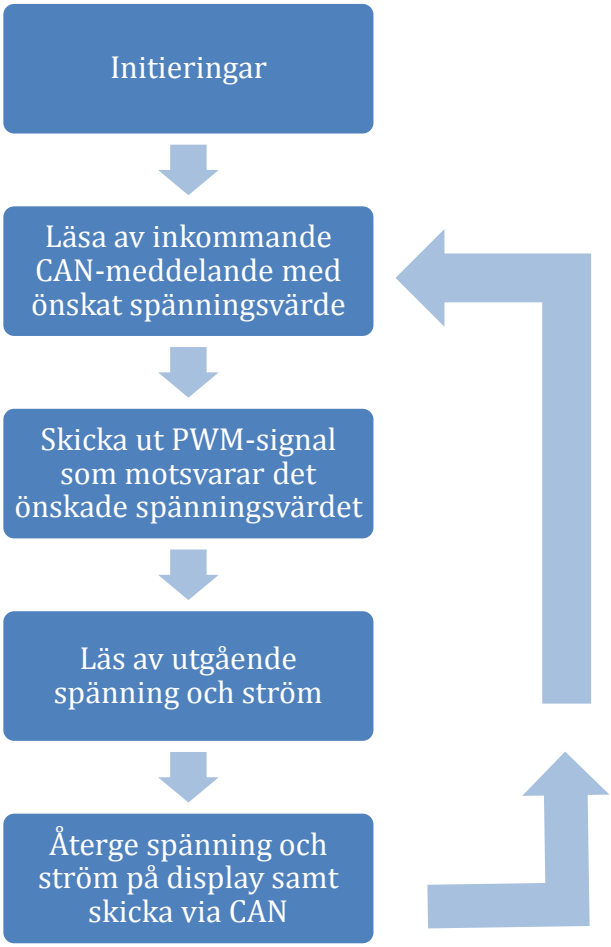
Efter att alla funktioner testats och verifierats på kopplingsdäcket genomfördes en konstruktionsgenomgång med personal på Consat för att upptäcka brister i konstruktionen och diskutera hur monteringen på kretskortet skulle läggas upp. Det beslutades att alla komponenter förutom de som behöva kylning skulle lödas fast på ett kort medans spänningsregulatorerna samt MOSFET-transistorerna skulle lödas fast på ett eget mindre kort i direkt anslutning till kylflänsen. Anledning till detta var att det större kortet då kunde placeras en bit ifrån kylflänsen och därmed skulle påverkas mindre av värmeutvecklingen omkring kylflänsen som skulle kunna påverka anläggningens funktion negativt. Efter montering testades alla funktioner igen för att verifiera att allt var rätt monterat och att ingen kallödning hade förekommit.

6.1.12. Förpackning

Utefter urvalet i *kapitel 5.12* valdes en låda ut som dels kunde rymma prototypkretskortet men även kunde ha kylflänsen monterad på kortsidan. Lådan har även ett genomskinligt lock för att slippa behöva montera displayen ytligt på lådan och istället ha den invändigt i direkt anslutning till kretskortet, detta enbart för att lådan skulle vara någorlunda solid på den sida där alla kontakter och displayen först var tänkt att monteras. Lådan modifierades sedan för att kretskort, kylfläns, kontakter och gummifötter skulle kunna monteras.

6.2. Utveckling av mjukvara

Utvecklingen av mjukvaran har skett i Microchips egen utvecklingsmiljö MPLAB. Valet av miljö gjordes utifrån tidigare erfarenheter av detta program och för att det fanns mycket hjälp att tillgå kring denna programvara. Innan utvecklingen startade gjordes ett enkelt flödesschema för huvudprogrammets funktion enligt *Figur 6*.1. Utvecklingen skedde sedan i den ordning som gynnade resten av processen. Det första som valdes att göras var att skriva koden för displayen i och med att man med hjälp av den enklare kan felsöka och skriva ut värden från till exempel en A/D-omvandling.



Figur 6.1 – Grovt flödesschema för mjukvara

6.2.1. LCD

När valet av LCD var klart kunde mjukvaran kring denna börja konstrueras. Enligt databladet (DISPLAY Elektronik GmbH, 2008) gjordes de initieringar som krävdes för att den skulle fungera. Här gjordes bland annat inställningar för om en eller två rader på displayen skulle användas och om den skulle köras i 4- eller 8-bit mode. Valet föll på två rader samt 4-bit mode. Det senare valdes för att spara in på antalet använda ben på mikrokontrollern i och med att detta läge kräver fyra färre sladdar. Sedan skapades en funktion för att rensa displayen samt en funktion för att skriva ut en text på displayen. Ytterligare en funktion lades senare till, en som specifikt skriver ut det avlästa ström- och spänningsvärde på displayen.

6.2.2. PWM

För att initiera PWM-signalen krävs periodtiden (PR2). Oscillatorns frekvens (F_{OSC}) är sedan tidigare bestämd till 8 MHz. Frekvensen på PWM-signalen valdes till ett godtyckligt värde (4 kHz) och Timer 2 Prescale beräknades till 4 enligt *ekvation 6.7* så att PR2 blev mindre än 255 som är registrets storlek. Utifrån detta kunde sedan periodtiden räknas ut och med

ekvation 6.7 blev då periodtidens värde 124, vilket sedan initierades i mjukvaran. (Microchip, 2009)

$$PR2 = \frac{Fosc}{4*TMR2_{Prescale}*PWM_{Frequency}} - 1 \qquad (6.7)$$

Nästa steg var att sätta upp en funktion för att kunna räkna ut pulsbredden utifrån önskad spänning. PWM-modulen i den valda processorn har en upplösning på tio bitar, vilket betyder att pulsbredden kan varieras i steg från 0 till 1023. Eftersom spänningen ska kunna varieras med 0,1 V noggrannhet upp till 32 V betyder detta att 0,1 V motsvarar ungefär 3. Ekvationen som räknar ut bitvärdet för pulsbredden ser ut enligt:

```
bitVärde = (önskadSpänning/maxSpänning)*maxBitVärde
```

Ett önskat spänningsvärde på exempelvis 12,4 V skulle då ge bitvärdet 396 och en pulsbredd på 39%.

6.2.3. A/D-omvandling

För att kunna återge den spänning och ström som matas ut från anläggningen måste en A/D-omvandling ske. Initieringen av denna funktion består av val av referensspänning, inställning av de ben som fysiskt kopplats in, hur A/D-omvandlingsresultatet ska sparas samt längden på acquisition time, delayen som krävs mellan A/D-omvandlingarna.

Sedan byggdes en funktion som utför själva omvandlingen. Beroende på vilket ben som ska läsa av spänningen vid omvandlingen anges det kanalnummer som motsvarar det önskade benet. Kanal 0 (AN0) används för att läsa av utgående spänning till kretskortet, kanal 1 (AN1) för utgående ström. Efter detta sätts biten som anger att omvandlingen ska starta och efter att denna är klar sparas resultatet undan i en egen variabel för att senare kunna omvandlas till ett decimalt värde. Vid det tillfälle då strömmen ska läsas av mäts i själva verket spänningen där 1 V motsvarar en 1 A. Strömmen räknas sedan ut i mjukvaran med hjälp av Ohms lag.

Kravet för återgivning av ström och spänning är 10 mA respektive 100 mV. I och med att A/D-omvandlaren i mikrokontrollern har en upplösning på 10 bitar fås då en noggrannhet på 5 mA samt 30 mV vilket uppfyller dessa krav. För att få en noggrannare återgivning kan en metod kallad oversampling användas. Denna lösning valdes dock bort för att avgränsa arbetet i och med att den upptäcktes sent i projektet.

6.2.4. CAN

Den sista funktionen som skapades var CAN-kommunikationen. Efter att ha studerat ett antal exempelprogram för CAN och PIC påbörjades uppbyggnaden av funktionerna. För att testa kommunikationen användes programmet Busmaster. Kommunikationen fungerade inte alls till en början så utvecklingen av CAN började om på nytt, fast då med ett färdigt bibliotek med funktioner för initiering, skicka och ta emot meddelande. Denna gång gick det att ta emot meddelande i PICen, dock med ett felmeddelande i Busmaster som följd. Detta felmeddelande var tyvärr inte så pass tydligt att det gick att läsa ut var det egentliga felet var. Men efter felsökning med oscilloskop kunde felet fastställas till att PICen inte gjorde rätt när den skulle bekräfta meddelandet. Hur detta problem skulle lösas listades aldrig ut eftersom bekräftelsedelen är något som ska ske automatiskt, så det var bara att felsöka koden en gång till. Parallellt med detta upptäcktes ett hårdvarufel gällande CAN-kommunikationen och när

felet var åtgärdat fungerade det även att skicka meddelande från PICen, dock med ett felmeddelande. Åtgärden som gjordes för att försöka lösa problemet var att ännu en gång börja om från start med CAN-funktionerna och noggrant följa databladet. Helt plötsligt började kommunikationen att fungera felfritt. Det lades dock ingen tid på att försöka hitta vad det exakta felet var i och med att den tiden inte fanns.

De funktioner som i slutändan var implementerade var följande:

- *CANInitialize*, denna funktion initierar och ställer in de ben som används för CAN, ID-filter för meddelande som tas emot, format på meddelanden som skickas och tas emot samt baud rate timing för kommunikationen.
- *MakeTxMessage*, här delas de decimala värdena som ska skickas upp i två variabler, en för heltalsdelen och en för decimaldelen.
- *CANTransmit*, denna funktion skickar ström- och spänningsvärdet som lästs av innan utmatningen. Här sänds datan enligt tabellen nedan.

D0	D1	D2	D3
Spänning, heltalsdel	Spänning, decimaldel	Ström, heltalsdel	Ström, decimaldel

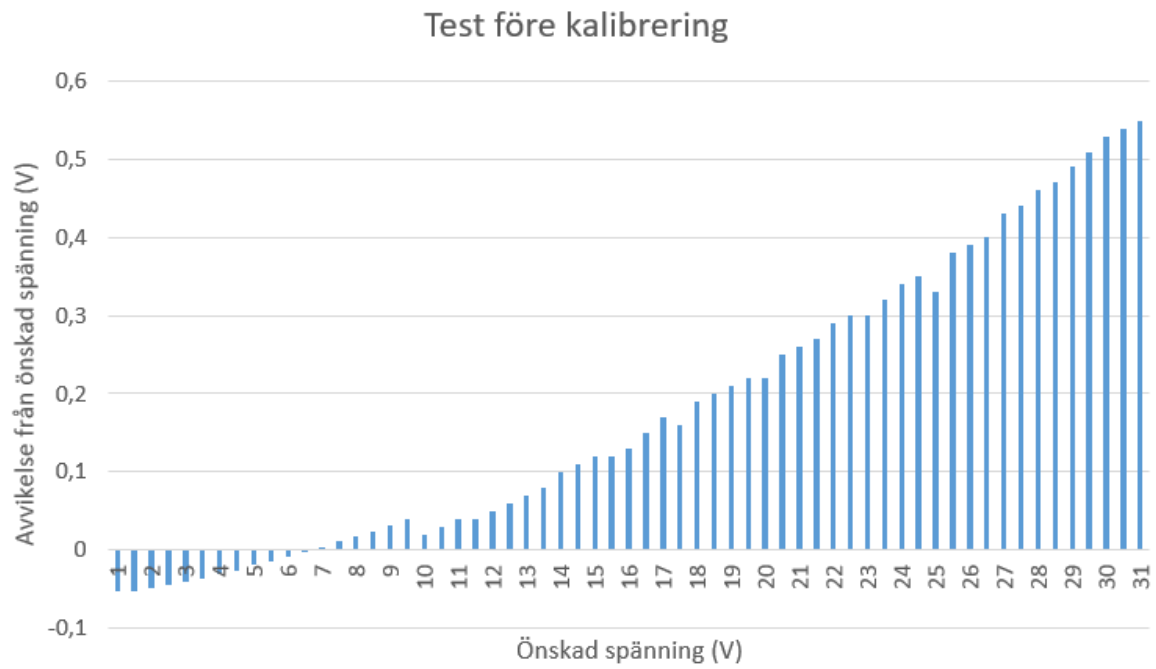
Exempel, avläst ström = 2,3 A, avläst spänning = 24,1 V

D0	D1	D2	D3
24 (0x18)	1 (0x01)	2 (0x02)	3 (0x03)

- *CANReceive*, det enda denna funktion gör är att spara undan mottaget meddelande i variabler.

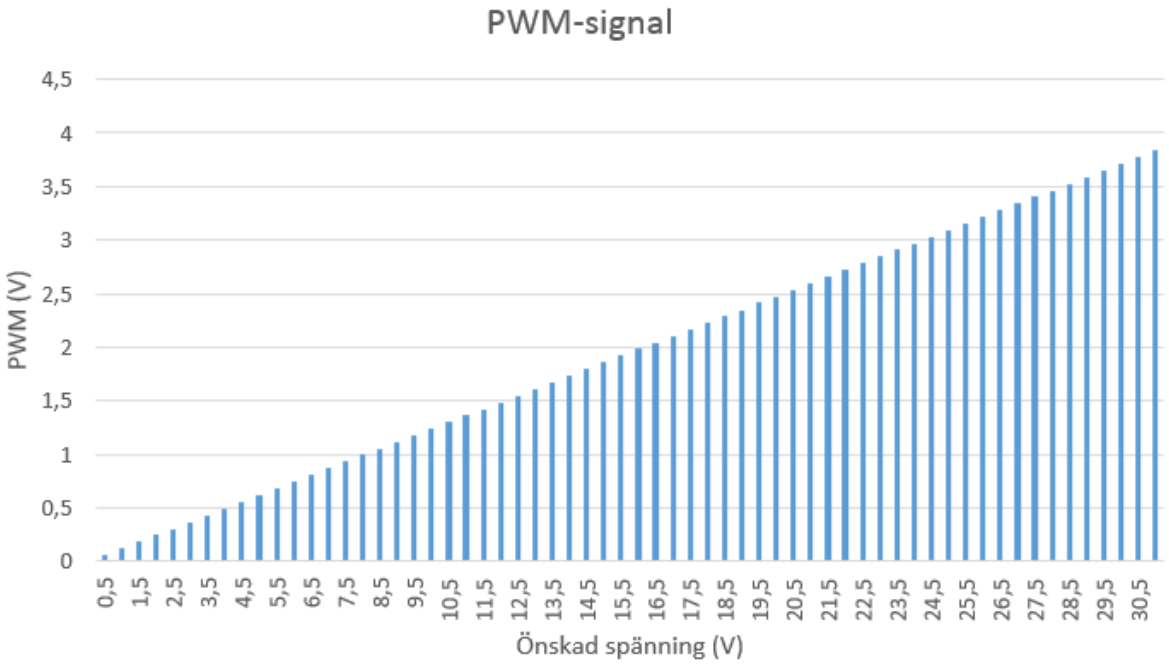
7. TEST OCH VERIFIERING

För att verifiera anläggningen gjordes flertalet tester. Det första testet som gjordes var att testa spänningsvärden från 0-31 V med steg om 0,5 V. Anledning till att testet inte genomfördes inom hela spänningsspannet (0-32 V) var för att det vid testtillfället endast fanns tillgång till en spänningskälla på 32 V. I *Figur 7.1* redovisas resultatet med fokus på utspänningens avvikelse från det önskade spänningsvärdet.



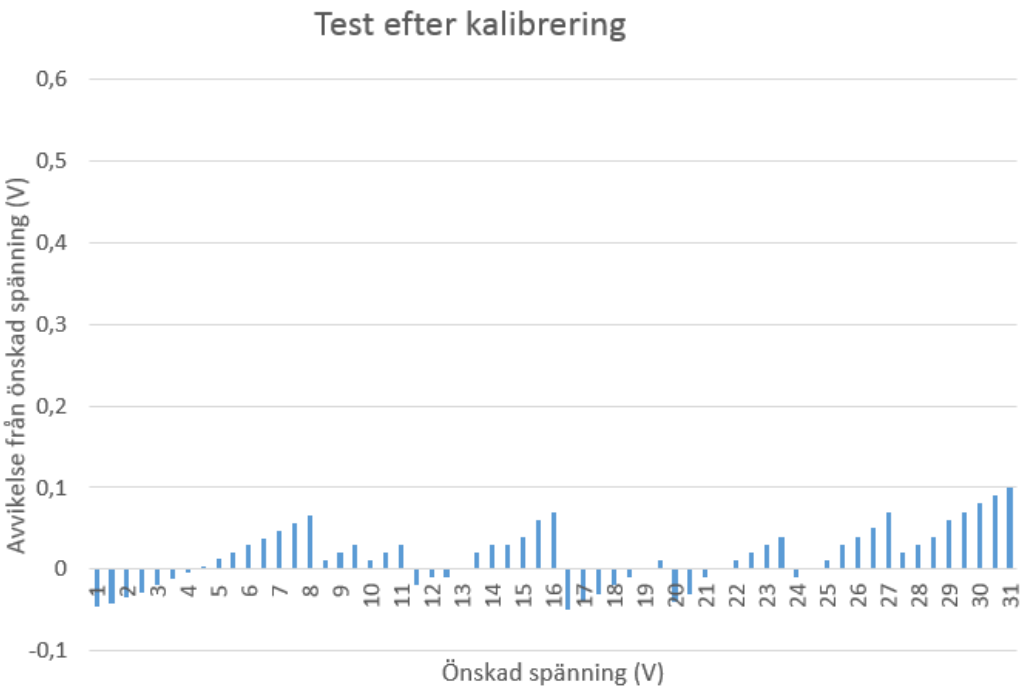
Figur 7.1 – Test före kalibrering

Här upptäcktes det att felet ökar i takt med att spänningen ökar. Eftersom mätningar även gjordes på PWM-signalen, före förstärkningssteget, se *Figur 7.2*, kunde det fastslås att problemet låg i hårdvaran eftersom signalen ökar precis som den ska.



Figur 7.2 – Mätning av PWM-signal

För att korrigera detta fel gjordes kalibreringar i återkopplingen till operationsförstärkaren samt korrigeringar i mjukvaran. I *Figur 7.3* ses testet som gjordes efter kalibreringen med ett bättre resultat. Här ligger spänningensvärdena inom gränsen för godkänd noggrannhet.

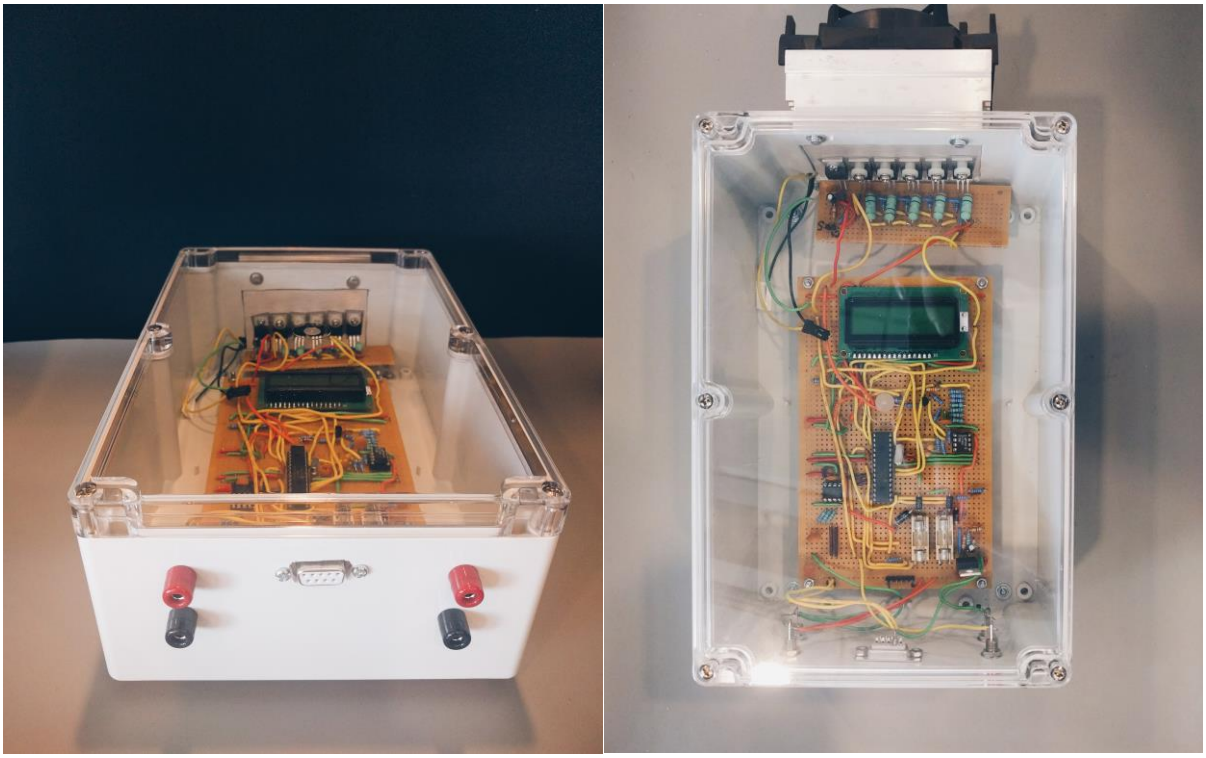


Figur 7.3 – Test efter kalibrering

8. RESULTAT

Målet med detta projekt var att konstruera en testanläggning för automatisk spänningsvariation. Med hjälp av kravspecifikationen kan det konstateras att de krav som satts är uppfyllda. Dessutom har alla önskemål förutom ett uppnåtts. Spänningen kan regleras upp till 32 V, hårdvaran är förpackad i en låda med fastmonterade kontakter och kretsen är monterad på ett prototypkretskort. Det önskemål som inte uppnåtts är strömmätningen. Lösningen är dock framarbetad men på grund av tidsbrist har den inte testats och implementerats.

Genom att koppla in en spänningskälla som försörjer anläggningen och sedan koppla denna till enheten som ska testas kan användaren skicka ett CAN-meddelande till testanläggningen med ett önskat spänningsvärde. Kretsen reglerar sedan spänningen till detta värde och den klarar av att mata ut 0-32 V samt driva enheter upp till 2 A. För att enkelt kunna se att spänningen som matas ut är motsvarande vad användaren önskat görs en avläsning på spänningen som sedan skrivs ut på en display samt skickas tillbaka via CAN för eventuell loggning. Då förstärkningen av spänningen i kretsen inte är helt konstant fås ett fel beroende på spänningen som ska matas ut. I *Figur 7.3* – Test efter kalibrering syns ett diagram som visar avvikelserna. Den slutgiltiga lösningen visas i *Figur 8.1*.



Figur 8.1 – Slutgiltig lösning

9. DISKUSSION

Det finns flera saker som kan förbättras och vidareutvecklas. Mycket av dimensionering i hårdvaran har gjorts via ekvationer hämtade från datablad, informationssökning och sedan tidigare inhämtad kunskap. Dessa ekvationer är oftast anpassade efter ideala förhållanden och är därför inte helt applicerbara i verkligheten. Det har allt som oftast i projektet och främst inom hårdvarukonstruktionen tagits fram teoretiska värden ur dessa ekvationer som sedan behövts dimensionerats om och kalibrerats via fysiska tester på kopplingsdäcket. Till en början fanns en tro eller okunskap i att dessa teoretiska värden skulle ligga hyfsat nära verkligheten och därmed inte behövas kalibreras om i någon större utsträckning. Detta har under projektets gång dock visats sig vara en allt större del av hårdvarukonstruktion att hela tiden testa och göra små ändringar i hårdvaran utefter att den hela tiden utvecklas med nya funktioner.

Många komponenter i kretsen är också kraftigt överdimensionerade mot vad de faktiskt ska användas till. Detta var något som efter missödet med den första spänningsregulatorn togs i beaktan, att det alltid är bra att överdimensionera och ha en bra marginal gentemot teoretiska värden, då elektriska komponenter visat sig kunna variera i beteende och utförande ganska kraftigt.

Utifrån detta har lärdomen dragits att det vid beställning av komponenter är bra att beställa dels den komponenten vars värden är uträknande med även beställa likadana komponenter men med omkringliggande värden. Då finns möjlighet att testa och kalibrera fram den lösning som fungerar bäst i praktiken.

Senare i projektet dök det upp en frågeställning om matningen till testanläggningen, där det diskuterades huruvida den kunde matas direkt från ett 230 V vägguttag. Framöver kan det uppstå behov av att utveckla anläggningen med en sådan möjlighet. Under detta projekt ansågs det lättare ur arbetssäkerhetssynpunkt att genomföra utvecklingsarbetet med lägre spänningar. Det skulle också bli nödvändigt att genomgå speciella utbildningar för att få handskas med sådana höga spänningar vilket det inte fanns tid för i detta projekt.

För att förbättra noggrannheten på spänningen skulle en algoritm kunna tas fram utifrån de testvärden som finns och sedan implementeras i mjukvaran för att reglera spänningen ännu noggrannare. Något som också skulle kunna förbättrats mjukvarumässigt är AD-omvandlingen. Den är inte särskilt noggrant på grund av avrundningsfel vid vissa värden. Detta hade kunnat åtgärdas med att antingen använda en mikrokontroller med högre upplösning på omvandlingen eller lösas i mjukvaran med en metod som kallas oversampling.

REFERENSER

Display Elektronik GmbH, 2008. *DEM 16226 SYH-LY,*
http://www.display-elektronik.de/filter/DEM16226SYH-LY.pdf
[Använd 25 04 2016].

Fairchild, 2014. *LM7805*
https://www.fairchildsemi.com/datasheets/LM/LM7805.pdf -
[Använd 23 05 2016].

Fischer elektronik, 2014. *SK481 Heatsink.*
http://www.farnell.com/datasheets/1332092.pdf
[Använd 23 05 2016].

Future Electronics, 2016. *What is a Schottky Diode?.*
http://www.futureelectronics.com/en/diodes/schottky-diodes.aspx
[Använd 23 05 2016].

Future Electronics, 2012. *What is a Zener Diode?*
http://www.futureelectronics.com/en/diodes/zener.aspx
[Använd 13 06 2016].

Intel, 2014. *Thermal Design Guide.*
http://download.intel.com/design/intarch/designgd/27370403.pdf
[Använd 23 05 2016].

Internal Rectifier, 2008. *Power MOSFET.*
Available at: http://www.farnell.com/datasheets/44031.pdf
[Använd 11 05 2016].

International Rectifier, 2004. *N-MOSFET.*
https://www.elfa.se/Web/Downloads/ta/_e/evIRF9530N_Data_E.pdf?mime=application%2Fpdf
[Använd 11 05 2016].

Kvaser, 2014. *Open source software*
https://www.kvaser.com/support/open-source-software/
[Använd 24 05 2016].

Linear Technology, 1999. *High Side Current Sense Amplifiers.*
Available at:
https://www.elfa.se/Web/Downloads/g_/ds/lt1787_eng_ds.pdf?mime=application%2Fpdf
[Använd 23 05 2016].

Linear Technology, 2013. *Single Supply, Dual Precision Op Amp.*
https://www.elfa.se/Web/Downloads/93/_e/xb642793_e.pdf?mime=application%
[Använd 11 05 2016].

Microchip, 2009. *PIC18F2580*
http://ww1.microchip.com/downloads/en/DeviceDoc/39637d.pdf
[Använd 20 04 2016].

Microchip, 2013. *http://ww1.microchip.com/downloads/en/DeviceDoc/20005167C.pdf.*
http://ww1.microchip.com/downloads/en/DeviceDoc/20005167C.pdf
[Använd 23 05 2016].

Microchip, 2013. *In-Circuit Debugger/Programmer User's Guide.*
http://ww1.microchip.com/downloads/en/DeviceDoc/52116A.pdf
[Använd 23 05 2016].

MPLAB, 2016. *Microchip.*
http://www.microchip.com/mplab/mplab-x-ide
[Använd 2016 05 19].

National Instruments, 2014. *Controller Area Network (CAN) Overview.*
http://www.ni.com/white-paper/2732/en/
[Använd 25 04 2016].

National Semiconductor, 2006. *LM2936.*
Available at: http://datasheet.octopart.com/LM2936Z-5.0-National-Semiconductor-datasheet-10554171.pdf
[Använd 23 05 2016].

Semtech, 2000. *TVS Diode Application Note.*
http://www.semtech.com/images/promo/What_are_TVS_Diodes.pdf
[Använd 23 05 2016].

Texas Instruments, 2000. *LM317HV*
http://www.ti.com/lit/ds/symlink/lm117hv.pdf
[Använd 23 05 2016].

Texas Instruments, 2008. *Texas Instruments*
http://www.ti.com/lit/an/spraa88a/spraa88a.pd
[Använd 22 04 2016].

Wikipedia, 2016. *Pulsbreddsmodulering.*
https://sv.wikipedia.org/wiki/Pulsbreddsmodulering
[Använd 23 05 2016].

Vishay, 2010. *Transient Voltage Suppressors for Automotive Electronic Protection.*
http://www.vishay.com/docs/88490/tvs.pdf
[Använd 23 05 2016].

FIGURFÖRTECKNING

BILAGOR

Bilaga 1 – Programkod main.c

```
/*
 * File:   main.c
 */

#define _XTAL_FREQ 8000000

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

// CONFIG1H
#pragma config OSC = HS         // Oscillator Selection bits (HS oscillator)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enable bit (Fail
                                    Safe Clock Monitor disabled)
#pragma config IESO = OFF       // Internal/External Oscillator Switchover
                                    bit (Oscillator Switchover mode
                                    disabled)

// CONFIG2L
#pragma config PWRT = OFF        // Power-up Timer Enable bit (PWRT
                                    disabled)
#pragma config BOREN = BOHW      // Brown-out Reset Enable bits (Brown
                                    out Reset enabled in hardware only
                                    (SBOREN is disabled))
#pragma config BORV = 3          // Brown-out Reset Voltage bits (VBOR
                                    set to 2.1V)

// CONFIG2H
#pragma config WDT = OFF         // Watchdog Timer Enable bit (WDT
                                    enabled)
#pragma config WDTPS = 32768     // Watchdog Timer Postscale Select bits
                                    (1:32768)

// CONFIG3H
#pragma config PBADEN = ON       // PORTB A/D Enable bit (PORTB<4:0>
                                    pins are configured as analog input
                                    channels on Reset)
#pragma config LPT1OSC = OFF     // Low-Power Timer 1 Oscillator Enable
                                    bit (Timer1 configured for higher power
                                    operation)
#pragma config MCLRE = ON        // MCLR Pin Enable bit (MCLR pin
                                    enabled; RE3 input pin disabled)

// CONFIG4L
#pragma config STVREN = ON       // Stack Full/Underflow Reset Enable
                                    bit (Stack full/underflow will cause
                                    Reset)
#pragma config LVP = ON          // Single-Supply ICSP Enable bit
                                    (Single-Supply ICSP enabled)
#pragma config BBSIZ = 1024      // Boot Block Size Select bit (1K words
                                    (2K bytes) boot block)
#pragma config XINST = OFF       // Extended Instruction Set Enable bit
(Instruction set extension and Indexed Addressing mode disabled (Legacy
mode))
```

```
// CONFIG5L
#pragma config CP0 = OFF            // Code Protection bit (Block 0
                                    (000800-001FFFh) not code-protected)
#pragma config CP1 = OFF            // Code Protection bit (Block 1
                                    (002000-003FFFh) not code-protected)
#pragma config CP2 = OFF            // Code Protection bit (Block 2
                                    (004000-005FFFh) not code-protected)
#pragma config CP3 = OFF            // Code Protection bit (Block 3
                                    (006000-007FFFh) not code-protected)

// CONFIG5H
#pragma config CPB = OFF            // Boot Block Code Protection bit (Boot
                                    block (000000-0007FFh) not code
                                    protected)
#pragma config CPD = OFF            // Data EEPROM Code Protection bit
                                    (Data EEPROM not code-protected)

// CONFIG6L
#pragma config WRT0 = OFF       // Write Protection bit (Block 0 (000800
                                    001FFFh) not write-protected)
#pragma config WRT1 = OFF       // Write Protection bit (Block 1 (002000
                                    003FFFh) not write-protected)
#pragma config WRT2 = OFF       // Write Protection bit (Block 2 (004000
                                    005FFFh) not write-protected)
#pragma config WRT3 = OFF       // Write Protection bit (Block 3 (006000
                                    007FFFh) not write-protected)

// CONFIG6H
#pragma config WRTC = OFF       // Configuration Register Write Protection
                                    bit (Configuration registers (300000
                                    3000FFh) not write-protected)
#pragma config WRTB = OFF       // Boot Block Write Protection bit (Boot
                                    block (000000-0007FFh) not write
                                    protected)
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit (Data
                                    EEPROM not write-protected)

// CONFIG7L
#pragma config EBTR0 = OFF      // Table Read Protection bit (Block 0
                                    (000800-001FFFh) not protected from
                                    table reads executed in other blocks)
#pragma config EBTR1 = OFF      // Table Read Protection bit (Block 1
                                    (002000-003FFFh) not protected from
                                    table reads executed in other blocks)
#pragma config EBTR2 = OFF      // Table Read Protection bit (Block 2
                                    (004000-005FFFh) not protected from
                                    table reads executed in other blocks)
#pragma config EBTR3 = OFF      // Table Read Protection bit (Block 3
                                    (006000-007FFFh) not protected from
                                    table reads executed in other blocks)

// CONFIG7H
#pragma config EBTRB = OFF      // Boot Block (000000-0007FFh) protected
                                    from table reads executed in other
                                    blocks


#include <xc.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
```

```
#include "lcd.h"
#include "pwm.h"
#include "adc.h"
#include "can.h"

#define V_OUT       0
#define C_OUT       1
#define V_IN        2

void main(void)
{
    float desiredVoltage, voltOutRead, currentOutRead, voltInRead;
    unsigned int bitValADC;

    // Init PIC - PWM, CAN, ADC, LCD, clear LCD //
    PwmInit();
    CANInitialize();
    ADCInit();
    LcdInit();
    LcdClear();

    while(1)
    {
                // Read CAN and convert CAN-message for PWM //
        desiredVoltage = CANToVoltage(CANReceive());

                // adjust the dutycycle
        if (desiredVoltage > 8 && desiredVoltage < 15) {
            PwmDutyCycle(desiredVoltage*0.99);
        }
        else if(desiredVoltage > 15) {
            PwmDutyCycle(desiredVoltage*0.985);
        }
        else {
             PwmDutyCycle(desiredVoltage);
        }

                // AD-conversion, outgoing voltage and Current //
        voltOutRead = GetADCValue (DoADC (V_OUT), V_OUT);

        //currentOutRead = GetADCValue (DoADC (C_OUT), C_OUT);
        //voltInRead = GetADCValue (DoADC (V_IN));

                // Convert ADC to floats and write to LCD //
        PrintValues(voltOutRead, 1.2);

                // Send current voltage and Current with CAN //
        CANTransmit(MakeTxMessage (voltOutRead, desiredVoltage));
    }
    return;
}
```

Bilaga 2 – Programkod can.c

```
/*
 * File:   can.c
 * Created on April 14, 2016, 3:13 PM
 */

#include <p18f2580.h>
#include <stdlib.h>
#include <math.h>
#include "xc.h"
#include "can.h"
#include "lcd.h"

static void CANSetOperationMode(char mode);

static void CANSetOperationMode(char mode)
{
    CANCON &= CAN_RESET_MODE;                    // clear previous mode
    CANCON |= mode;                              // set new mode

    while( CANGetOperationMode() != mode );    // Wait till desired mode is
set
}

void CANInitialize()
{
    // PIN Settings
    TRISBbits.RB2 = 0;              // TX (RB2) - output
    TRISBbits.RB3 = 1;              // RX (RB3) - input

    LATBbits.LATB2 = 0;             // Clear pins
    LATBbits.LATB3 = 0;

    TXB0CONbits.TXPRI1 = 1;         // Highest Transmit Prio, level 3
    TXB0CONbits.TXPRI0 = 1;

    // Put module into Configuration mode.
    CANSetOperationMode(CAN_MODE_CONFIGURATION);

    // Transmit mode
    TXB0SIDLbits.EXIDE = 0;         // Standard format will be transmitted
    RXF0SIDLbits.EXIDEN = 0;        // Filter will only accept standard ID
messages

    // Baud Rate Settings
    // 500kHz, 75% Sample Point
    BRGCON1bits.SJW0 = 0;           // Syncro Jump Width (Tq) = 1, SJW = 00
    BRGCON1bits.SJW1 = 0;
    BRGCON1bits.BRP5 = 0;           // Baud Rate Prescaler = 1, BRP =
000000
    BRGCON1bits.BRP4 = 0;
    BRGCON1bits.BRP3 = 0;
    BRGCON1bits.BRP2 = 0;
    BRGCON1bits.BRP1 = 0;
    BRGCON1bits.BRP0 = 0;

    BRGCON2bits.SEG2PHTS = 1;       // Phase Segment 2 Freely programmable
    BRGCON2bits.SAM = 0;            // Bus line is sampled once at the
sample point
```

```
    BRGCON2bits.SEG1PH2 = 0;        // Phase Seg 1 (Tq) = 3, SEG1PH = 010
    BRGCON2bits.SEG1PH1 = 1;
    BRGCON2bits.SEG1PH0 = 0;
    BRGCON2bits.PRSEG2 = 0;         // Prop Time (Tq) = 2, PRSEG = 001
    BRGCON2bits.PRSEG1 = 0;
    BRGCON2bits.PRSEG0 = 1;

    BRGCON3bits.WAKDIS = 1;         // Wake-up disabled
    BRGCON3bits.WAKFIL = 0;         // CAN bus line filter is not used for
wake-up

    BRGCON3bits.SEG2PH2 = 0;        // Phase Seg 2 (Tq) = 2;, SEG2PH = 001
    BRGCON3bits.SEG2PH1 = 0;
    BRGCON3bits.SEG2PH0 = 1;

    // Receive mode
    RXB0CONbits.RXM1 = 0;           // Receive only valid messages with
standard identifier
    RXB0CONbits.RXM0 = 1;
    RXB0CONbits.RXB0DBEN = 1;
    RXB0CONbits.FILHIT0 = 0;            // Acceptance filter 0 (RXF0)
    RXF0SIDH =((RX_ID>>3) & 0xFF);
    RXF0SIDL = ((RX_ID<<5) & 0xE0);
    RXB0SIDH = ((RX_ID>>3) & 0xFF);     // Receive ID
    RXB0SIDL = ((RX_ID<<5) & 0xE0);

    // Mode select
    ECANCONbits.MDSEL0 = 0;         // Mode 0
    ECANCONbits.MDSEL1 = 0;

    // Put module into Normal Mode
    CANSetOperationMode(CAN_MODE_NORMAL);
}

Message CANReceive()
{
    Message rxMessage;

    if (RXB0CONbits.RXFUL)              // If message received
    {
        PIR3bits.RXB0IF = 0;            // Clear received flag
        rxMessage.DLC = RXB0DLC;        // Number of bytes
        rxMessage.VoltageL = RXB0D1;    // Data
        rxMessage.VoltageH = RXB0D0;
        PrintValues(RXB0D0,RXB0D1);
        RXB0CONbits.RXFUL=0;             // Receive buffer is ready to
receive a new message
    }
    return rxMessage;
}

Message MakeTxMessage (float voltOutRead, float currentOutRead)
{
    Message txMessage;
    voltOutRead = round(voltOutRead*10)/10;     // Round float to nearest
.0
    int voltH = (int)voltOutRead;               // Get int part of float
    float voltL = voltOutRead - voltH;          // Get decimal part of
float
    voltL *= 10;                                // Decimal part to "int"
```

```
    currentOutRead = round(currentOutRead*10)/10; // Round float to nearest
.0
    int curH = (int)currentOutRead;               // Get int part of float
    float curL = currentOutRead - curH;           // Get decimal part of
float
    curL *= 10;

    txMessage.DLC = 4;                        // Data to transmit
    txMessage.VoltageH = voltH;               // xxxH = int part of value
    txMessage.VoltageL = voltL;               // xxxL = decimal part of value
    txMessage.CurrentH = curH;
    txMessage.CurrentL = curL;
    return txMessage;
}

void CANTransmit(Message txMessage)
{
    TXB0SIDH = ((TX_ID>>3) & 0xFF);     // Transmit ID
    TXB0SIDL = ((TX_ID<<8) & 0xE0);

    TXB0DLCbits.TXRTR = 0;              // No Transmitter Remote
Transmission Request

    TXB0DLC = 0x00;                     // Reset DLC
    TXB0DLC |= txMessage.DLC;           // Set new DLC

    TXB0D3 = txMessage.CurrentL;        // Message Data
    TXB0D2 = txMessage.CurrentH;
    TXB0D1 = txMessage.VoltageL;
    TXB0D0 = txMessage.VoltageH;

    TXB0CONbits.TXREQ = 1;              // Ready to transmit
}

float CANToVoltage (Message rxMessage)
{
    float volt = rxMessage.VoltageH;
    float voltDec = (float)rxMessage.VoltageL/10;
    volt += voltDec;
    return volt;
}
```

Bilaga 3 – Programkod can.h

```
/*
 * File:   can.h
 */

#include "xc.h"

typedef struct {
    unsigned char IDH;
    unsigned char IDL;
    unsigned char DLC;
    unsigned char VoltageH;
    unsigned char VoltageL;
    unsigned char CurrentH;
    unsigned char CurrentL;
 }Message ;


#define CANGetOperationMode() (CANSTAT & 0xe0)

#define CAN_MODE_NORMAL             0x00
#define CAN_MODE_CONFIGURATION      0x80
#define CAN_RESET_MODE              0x1F

#define TX_ID           0x111
#define RX_ID           0x123

//volatile Message txMessage;
//volatile Message rxMessage;

/*  CANInitialize
 *  Initializes the CAN settings, Pin settings, Baud Rate, RX/TX mode
 *
 *  Input: -
 *
 *  Output: -
 */

void CANInitialize();

/*  CANReceive
 *  Handle the received message and save it to rxMessage
 *
 *  Input: -
 *
 *  Output: -
 */

Message CANReceive();

/*  MakeTxMessage
 *  Composes a message to transmit based on input and store it in txMessage
 *
 *  Input: Voltage and Current value (float)
 *
 *  Output: -
 */

Message MakeTxMessage (float voltOutRead, float currentOutRead);

/*  CANTransmit
```

```
 *  Transmits the message stored in txMessage
 *
 *  Input: Voltage and Current value (float)
 *
 *  Output: -
 */

void CANTransmit(Message txMessage);

/*
 * CANToVoltage
 *
 * Composes float value based on received CAN message
 *
 * Input: Integer part and decimal part of desired voltage value
 *
 * Output: Float value
 */

float CANToVoltage (Message rxMessage);
```

Bilaga 4 – Programkod lcd.c

```
/*
 * File:   lcd.c
 */

#include <xc.h>
#include <stdlib.h>
#include <stdio.h>
#include "lcd.h"

static void LcdCmd (unsigned char cmd);
static void LcdEN ();
static void LcdCursor (unsigned char pos);
static void LcdWrite (unsigned char c);

void LcdInit ()
{
    TRISAbits.RA4 = 0;          // LCD connected pins to output
    TRISAbits.RA5 = 0;
    TRISCbits.RC1 = 0;
    TRISCbits.RC4 = 0;
    TRISCbits.RC6 = 0;
    TRISCbits.RC5 = 0;

    RS = 0;
    EN = 0;
    __delay_ms(40);             // Wait 40ms after power applied
    DB4 = 1;                    // Init command
    DB5 = 1;
    DB6 = 0;
    DB7 = 0;
    LcdEN();
    __delay_ms(6);              // Wait 6ms
    LcdEN();
    __delay_ms(2);
    LcdEN();
    __delay_ms(2);
    DB4 = 0;                    // Set 4 bit mode
    LcdEN();
    __delay_ms(2);
    LcdCmd(0b00101000);         // Set interface length 4-bit, 2-line, 5x8
dots display
    LcdCmd(0b00001100);         // Display On, Cursor OFF, Cursor Blink OFF
    LcdCmd(0b00000110);         // Set entry Mode
    LcdClear();                 // Clear screen
}

 static void LcdCmd (unsigned char cmd)
{
    RS = 0;             // Instructions/command selected
    LcdWrite(cmd);      // Write cmd to LCD
    RS = 1;
}

 static void LcdEN ()
{
    EN = 1;             // Set EN to high
    __delay_us(1);      // Wait for 1us
    EN = 0;             // Set EN to low
}
```

```
static void LcdWrite (unsigned char c)
{
    if(c & 0x80) DB7=1; else DB7=0;         // Set pins to high/low (write
to lcd)
    if(c & 0x40) DB6=1; else DB6=0;
    if(c & 0x20) DB5=1; else DB5=0;
    if(c & 0x10) DB4=1; else DB4=0;
    LcdEN();
    if(c & 0x08) DB7=1; else DB7=0;
    if(c & 0x04) DB6=1; else DB6=0;
    if(c & 0x02) DB5=1; else DB5=0;
    if(c & 0x01) DB4=1; else DB4=0;
    LcdEN ();
    __delay_us(50);
}

void LcdWriteString (char *s)
{
    while(*s != '\0')
    {
        LcdWrite (*s++);                    // write each character
    }
}

void LcdClear ()
{
    RS = 0;
    LcdWrite(0x01);         // Clear command
    RS = 1;
    __delay_ms(1);
}

static void LcdCursor (unsigned char pos)
{
    RS = 0;
    LcdWrite(0x80 + pos);             // Move cursor
    RS = 1;
}

void PrintValues(float volt, float ampere)
{
    char voltC[];
    char ampC[];
    LcdClear();                        // Clear LCD

    LcdCursor(0x00);
    sprintf(voltC, "%.2f V", volt);    // Voltage (Float) to String
    LcdWriteString (voltC);            // Write volt value to LCD

    LcdCursor(0x40);
    sprintf(ampC, "%.2f A", ampere);   // Current (Float) to String
    LcdWriteString (ampC);             // Write current value to LCD
    __delay_ms(90);
    __delay_ms(90);
}
```

Bilaga 5 – Programkod lcd.h

```
/*
 * File:   lcd.h
 */

#define _XTAL_FREQ 8000000

#define RS  PORTAbits.RA4       // LCD register select pin
//#define RW  PORTAbits.RA1
#define EN  PORTAbits.RA5       // LCD enable pin
//#define DB0 Rxx
//#define DB1 Rxx
//#define DB2 Rxx
//#define DB3 Rxx
#define DB4 PORTCbits.RC4       // LCD data bus pins
#define DB5 PORTCbits.RC5
#define DB6 PORTCbits.RC1
#define DB7 PORTCbits.RC6


#include <xc.h>
#include <stdlib.h>
#include <stdio.h>

/* LcdInit
 * Initiates the LCD
 *
 * Input: -
 *
 * Output: -
 */

void LcdInit ();

/*  LcdWriteString
 *  Writes string to LCD through lcdWrite
 *
 *  Input: String
 *
 *  Output: -
*/

void LcdWriteString (char *s);

/*  LcdWriteStringConst
 *  Writes string constant to LCD through lcdWrite
 *
 *  Input: String constant
 *
 *  Output: -
 */

void LcdWriteStringConst(const char * s);

/*  LcdClear
 *  Writes clear command do LCD
 *
 *  Input: -
 *
 *  Output: -
*/
```

```
void LcdClear ();

/*  PrintValues
 *  Prints volt value at first row, ampere value at second row
 *
 *  Input: volt and ampere as floats
 *
 *  Output: -
 */

void PrintValues(float volt, float ampere);
```

Bilaga 6 – Programkod adc.c

```
/*
 * File:   adc.c
 */


#include <xc.h>
#include "adc.h"

#define V_OUT       0
#define C_OUT       1
#define V_IN        2

#define PIN_MAX     1023
#define V_MAX_32    32
#define V_MAX_5     5

#define RESISTOR    0.47

void ADCInit ()
{
    ADCON0bits.CHS = 0b0000;        // Channel 0, default
    ADCON0bits.ADON = 1;            // AD-module ON
    ADCON1bits.VCFG1 = 0;           // Internal voltage reference, AVss
    ADCON1bits.VCFG0 = 0;           // Internal voltage reference, AVDD
    ADCON1bits.PCFG = 0b1100;       // AN0, AN1, AN2 = analog, AN3-AN10 =
digital
    TRISAbits.RA0 = 1;              // RA0/AN0 = input
    TRISAbits.RA1 = 1;              // RA1/AN1 = input
    ADCON2bits.ACQT = 0b101;        // Acquisition time --> 12 TAD
    ADCON2bits.ADCS = 0b100;        // Conversion clock, (FOSC/4) ~10us
    ADCON2bits.ADFM = 1;            // Right justified adc result
}

unsigned int DoADC (int channel)
{
    ADCON0bits.CHS |= channel;          // Channel 0 = Volt out, Channel 1
= Current, Channel 2 = Volt in
    ADCON0bits.GO_DONE = 1;             // Start conversion
    /***** EVENTUELL DELAY *****/
    while (ADCON0bits.GO_DONE);         // Wait for conversion to complete
    unsigned int adc_reading = (ADRESH << 8) | ADRESL;      // Put
conversion result to adc_reading
    adc_reading *= 1.2626;                // Calculate 4 V->5 V
    /*if (channel == C_OUT) {
        adc_reading /= RESISTOR;
    }*/
    return adc_reading;
}

float GetADCValue (float value, int channel)
{
    float result = 0;
    if (V_OUT == channel) {
        result = (value/PIN_MAX)*V_MAX_32;          // Convert result to
voltage
    }
    else if (C_OUT == channel) {
        result = (value/PIN_MAX)*V_MAX_5;
    }
```

```
    return result;
}
```

Bilaga 7 – Programkod adc.h

```
/*
 * File:   adc.h
 */
#include <xc.h>
#include <stdlib.h>
#include <stdio.h>

/*  adcInit
 *  Set configbits for AD-conversion
 *
 *  Input: -
 *
 *  Output: -
 */
void ADCInit (void);

/*  doADC
 *  Do ADC for desired channel
 *
 *  Input: Channel 0 = Volt out, Channel 1 = Current, Channel 2 = Volt in
 *
 *  Output: AD result, 0-1023
 */

unsigned int DoADC (int channel);

/*  getADCValue
 *  Converts result from doADC to float voltage/current
 *
 *  Input: Channel 0 = Volt out, Channel 1 = Current, Channel 2 = Volt in
 *
 *  Output: Voltage/Current as float
 */

float GetADCValue (float value, int channel);
```

Bilaga 8 – Programkod pwm.c

```
/*
 * File:   pwm.c
 */

#define PIN_MAX     1023
#define V_MAX_32      32

#include <xc.h>
#include <math.h>
#include "pwm.h"

void PwmInit ()
{
    PR2 = 0x7C;                         // Set PWM-period: PR2 = 124
    TRISCbits.RC2 = 0;                  // RC2 = output
    T2CONbits.T2CKPS = 01;              // Prescale value = 4, Timer2
    CCP1CON = 0b00001100;               // PWM mode
    T2CONbits.TMR2ON = 1;               // Timer2 is on
}

void PwmDutyCycle (float volt)
{
    if (volt > V_MAX_32)              // If volt is too large, set to valid
value
    {
        volt = V_MAX_32;
    }
    unsigned int pinVal = (int)((volt / V_MAX_32) * PIN_MAX);       //
Calculate pin value
    pinVal /= 2;
    CCPR1L = pinVal >> 2;                                  // 8 MSB to CCPR1L
    CCP1CON = (CCP1CON & 0b11001111) | ((pinVal & 0b0000000011) << 4);
// 2 LSB to CCP1CON <5:4>
}
```

Bilaga 9 – Programkod pwm.h

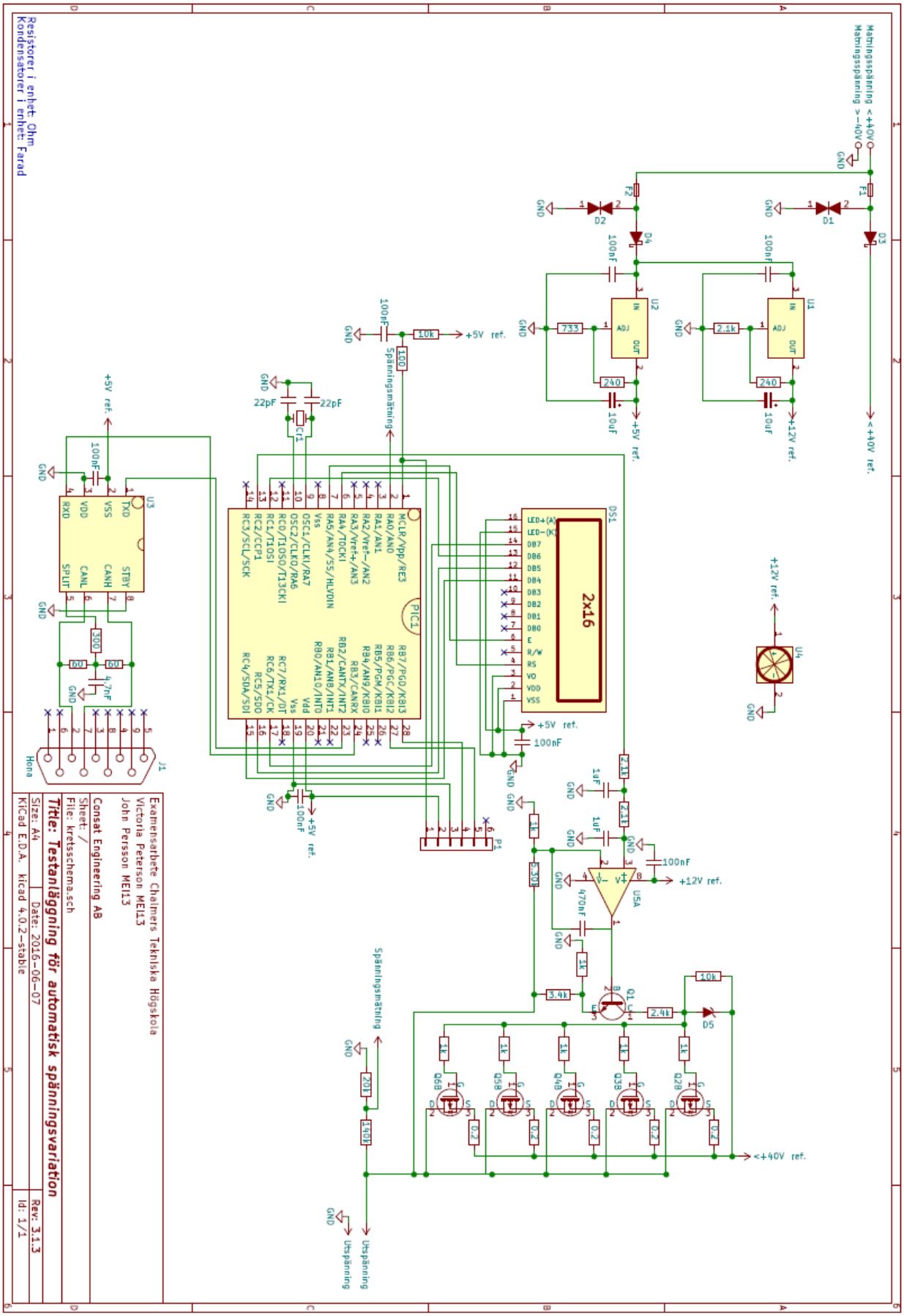
```
/*
 * File:   pwm.h
 */

#include <xc.h>
#include <stdlib.h>
#include <stdio.h>
/*  pwmInit()
 *  Initiate the PWM pins
 *
 *  Input: -
 *
 *  Output: -
 */
void PwmInit ();            // init for PWM

/*
 * dutyCycle
 *
 * Sets the duty cycle based on the input.
 *
 * Input: Desired volt value (float)
 *
 * Output: -
 */

void PwmDutyCycle (float volt);
```

Bilaga 10 – Kretsschema



Bilaga 11 - Komponentlista

Komponenter inkluderade i kretsschemat		
Beteckning	**Tillverkare**	**Serienummer**
Cr1	IQD	LFXTAL003156
D1	Littlefuse	P6KE43CA
D2	Diotec	SB340
D3	Diotec	SB140
D4	Vishay	BZX85C12
DS1	Display Elektronik	DEM 16226 SYH-LY
F2	RND Components	RND 170-00062
F1	RND Components	RND 170-00065
J1	MH Connectors	MHDD09-F-T-B-S
P1	Molex	90120-0926
PIC1	Microship	PIC18F2580-I/SP
Q1	Fairchild	MPSA06
Q2B	IR	IRF9530NPBF
U1	Texas Instrument	LM317HVT/NOPB
U2	Microship	MCP2561-E/P
U4	Intel	A80856-001
U5A	Linear Technology	LT1413CN8#PBF
Komponenter ej inkluderade i kretsschemat		
	Tillverkare	**Serienummer**
	RND Components	RND 170-00001
	Bopla	M2401G
	No Brand (Elfa)	19-1-1.
	No Brand (Elfa)	TO-220 0,30MM SIL RUBBER
	Roth Elektronik	RE318-HP
	Schützinger	IBU 5568 Ni / RT
	Schützinger	IBU 5568 Ni / SW
	Austerlitz Electronic GmbH	IB10