Expansionskort till en utvecklingsplattform baserad på öppen hårdvara

LUNDS UNIVERSITET
Campus Helsingborg

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för elektroteknik

Examensarbete:
Emanuel Frid
Rasmus Ankersen

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2014

Sammanfattning

Vi har skapat en utvecklingsplattform baserad på OSHW (Open Source Hardware) med en öppen och användarvänlig programmeringsmiljö. Hårdvaran är Arduino-kompatibelt för att ge användaren en möjlighet att snabbt och enkelt utvärdera produktidéer, men är även tänkt att kunna fungera i en slutprodukt. Vi har utvecklat ett expansionskort som är tänkt att anslutas till ett befintligt processorkort.
Arbetet har bestått i att utveckla och designa hårdvaran och satt denna i produktion om fem enheter, samt skrivit ett bibliotek med drivrutiner för att kunna använda hårdvarans funktioner i utvecklingssammanhang.

Nyckelord:
Öppen-källa, Arduino, Elektronikkonstruktion, processtyrning, Mjukvaruutveckling.

Abstract

We have created a development platform based on OSHW (Open Source Hardware) with an open and user friendly development environment. The hardware is Arduino compatible to give the user a possibility to quick and easy evaluate product ideas, but is also intended to be use in a final product. We have developed a expansions card to be used with an already existing micro processor module.

We have designed the hardware and manufactured five units. We have also written a software library which contains functions that supports the use of the hardware for development purposes.

Keywords: Open-source, Arduino, Electrical engineering, Process Control, Software-development.

Förord

Denna rapport beskriver vårt examensarbete för högskoleingenjörsutbildningen inom elektroteknik på Lunds tekniska Högskola/Campus Helsingborg. Vi har bekantat oss med processen från en idé till en färdig lösning. Vi har under arbetets gång fördjupat oss i elektronikkonstruktion och mjukvaruutveckling, samt hur tillverkningsprocessen för dagens elektronikprodukter ser ut. Vi vill tillänga ett stort tack till Per Johansson på Bläckbild för många goda tips under arbetets gång.

Rasmus Ankersen
Emanuel Frid

Innehållsförteckning

1 Inledning

Bakgrund

Vi och företaget (Bläckbild) anser att det fattas en enkel och användarvänlig utvecklingsplattform där man enkelt kan gå från labb-bänken till slutmiljön. Tänkbara användningsområden är hemautomation, enklare industriautomation och övervakning och styrning av mindre processer. Tanken är även att den ska agera som en testplattform för idéer kring IoT (”Internet of things”). Detta är tanken om att allt fler saker ska ha en internetuppkoppling, även de enheter som i vanliga fall inte ansluts till internet (vitvaror, belysning, verktyg och dylikt).

Syfte

Syftet är att skapa en utvecklingsplattform baserad på OSHW (Open Source Hardware) [3] med en öppen och användarvänlig programmeringsmiljö. Hårdvaran ska vara Arduino-kompatibelt för att ge användaren en möjlighet att snabbt och enkelt utvärdera produktidéer, men är även tänkt att kunna fungera i en slutprodukt.

Problemformulering

Något som har önskats av Bläckbild har varit ett utvecklingsverktyg som uppfyller ovan nämnda kriterier och som designas i en formfaktor där elektroniken passar i en standardiserad inkapsling som tillåter användaren att montera denna i nära anslutning av den process som ska kontrolleras. Systemet ska vara modulbaserat med ett processorkort samt ett tillhörande expansionskort som ger användaren möjlighet att välja ett expansionskort som har de funktioner som önskas. Expansionskortet som ska utvecklas ska hantera funktioner som ofta förekommer i processtyrningar. Dessa funktioner skall vara kommunikation, temperaturmätning, digitala in- och utgångar samt möjlighet till anslutning av externa givare. Expansionskortet ska precis som de övriga delarna i systemet bygga på öppen hårdvara och vara kompatibel med processorkortet. Arbetet består i att konstruera och tillverka ett expansionskort till ett processorkort som redan är utvecklat och tillverkat av Bläckbild. Expansionskortets hårdvaruspecifikationer beskrivs i kapitlet ”Hårdvara”. Utöver konstruktionen av hårdvara ska det även skrivas ett Arduino-bibliotek med färdiga funktioner för att kunna använda kortet.

När hårdvaran och mjukvaran är färdig så ska vi sätta upp en enklare process som ska övervakas och styras.

Avgränsningar

Hårdvaran är tänkt att vara en utvecklingsplattform, så någon EMC-testning kommer inte att utföras. Då expansionskortet endast kommer tillverkas i en liten serie, kommer designen inte anpassas för att uppfylla några specifika kostnadskrav. De mjukvarufunktioner som ska implementeras kommer endast stödja de funktioner som expansionskortet utrustas med, således kommer mjukvarubiblioteket inte ha stöd för Wifi-modulen som processorkortet har utrustats med. Drivrutinerna för Wifi-modulen får istället importeras från tillverkarens mjukvarubibliotek om användaren vill använda Wifi-kommunikation

Metodik & arbetsformer

Vi kommer att laborera i skolan och på Bläckbild för testa våra idéer och ta fram en konstruktion som uppfyller våra önskemål. När vi är nöjda med vår konstruktion kommer vi att designa ett kretskort, som sedan ska gå till tillverkning. Vi kommer att använda Cadsoft's Eagle för designen av kortet. Utvecklingen av mjukvara kommer att ske i Arduino's utvecklingsmiljö.
Under arbetes gång kommer vi att publicera arbetet på en blogg och CAD-ritningar samt mjukvara kommer att läggas upp på en Git Hub-sida. Allt för att projektet ska uppfylla kriterierna för öppen hårdvara.

2 Metoder

2.1 Hårdvarukonstruktion

Hårdvarukonstruktionen börjar med en analys av vilka krav som ställs på produkten. Det är här man bestämmer vilka funktioner som ska finnas med, vilken miljö produkten ska användas i och hur den ska användas.

Funktionerna som man bestämmer från arbetets början kan mycket väl komma att ändras under arbetets gång. Detta kan bero på att man inser att vissa funktioner är överflödiga, så att dessa plockas bort, eller att man kommer fram till att nya funktioner behöver läggas till för att uppnå den funktionalitet som man eftersträvar.

Miljön som expansionskortet ska användas i är från början tänkt att vara en industrin-anpassad labbmiljö. Med detta menas att det används spänningsnivåer som är vanligt förkommande ute i industrin, samt att möjligheten för att koppla in extern utrustning (såsom externa givare) skall finnas. Det skulle kunna kalla ett litet steg mot en standardisering, även om det idag finns många olika standarder. Anledningen till detta är att på så sätt få ett så flexibelt system som möjligt, vilket är viktigt då detta är tänkt som ett utvecklingsverktyg.

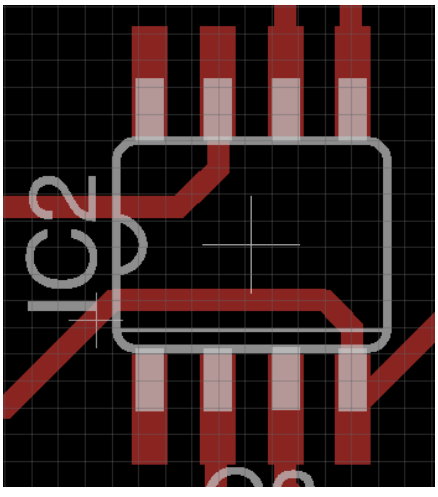
När det är bestämt vilken miljö produkten kommer att befinna sig i och vilka funktioner den ska ha, kan man börja utforma produkten efter dessa parametrar.

I detta fall har varje funktion brutits ner i mindre segment, oftast har detta börjat som en ren blyertsskiss på ett papper, där man ser vilken karaktär av komponent man är ute efter och vilka kringkomponenter denna ska ha för att uppfylla den funktionen man söker. Efter en stunds arbete har man åstadkommit en liten schemaritning för den aktuella funktionen. Det är också i detta stadie som de flesta beräkningar för komponentvärden sker. Detta arbete fortskrider för övriga funktioner tills dess att den önskvärda funktionalliteten har uppfyllts. Det är såklart inte nödvändigt att från grunden rita alla delscheman för hand, utan flera av dessa kan med lätthet ritas direkt under CAD-processen, men skissandet ger en bra förståelse för hur schemat ska utformas.

Då designprocessen ska resultera i ett färdigt kretskort används datorn till hjälp för att utforma detta. Det finns en uppsjö av olika CAD-program för elektronikkonstruktion, allt från gratisversioner till program som kostar omkring 100 000 kronor per licens. Programmet som har använts under arbetet kommer från Cadsoft och heter Eagle. Det är ett CAD-program med den grundläggande funktionaliteten som behövs och är fritt att använda för utvärderingssyfte samt för icke vinstdrivande projekt. Den fria licensen kommer dessvärre med lite begränsningar. Dessa består i en storleksgräns på kortet (80x100mm) samt i en begränsning av antalet signallager på kortet. Dessa begränsningar är i detta fall dock inget större problem.

2.1.1 Val av komponentinkapsling

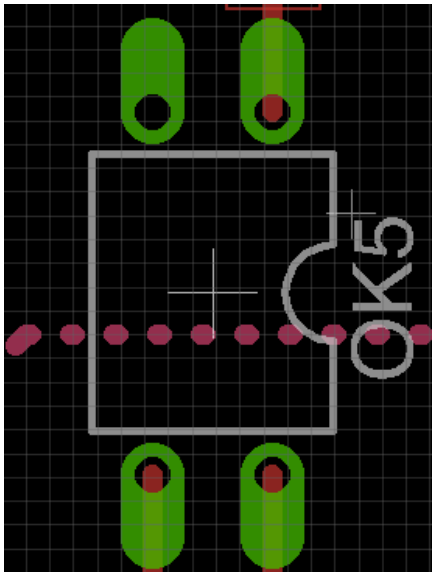
Det fysiska formatet på en komponent är helt beroende på i vilket inkapslingsformat komponenten levereras i. Expansionskortet består till mestadels av ytmonterade komponenter, men det finns även komponenter som är hålmonterade. Ytmonterade komponenter fästs i kopparlagret på vilket de är placerade. Detta sker genom att man har ”paddar” (från engelska pads) som är exponerade kopparytor där komponentbenen läggs mot och lödas fast. I figur 1 visas ett exempel på hur det fysiska formatet för en ytmonterad IC-krets kan se ut.



Figur 1, ytmonterad komponent

I figur 1 visas en skärmdump från Eagle's kortvy. Det röda lagret motsvarar det översta kopparlagret och visar ”paddarna” där komponentbenen kommer att lödas fast. Det vita lagret motsvarar ett silkscreen-tryck som kommer att tryckas på kortet, dock så kommer korttillverkaren att automatisk ta bort den delen av trycket som ligger över ”paddarna”, detta för att det annars hade varit omöjligt att löda fast komponenten om ”padden” hade varit täckt av silkscreen-tryck. Anledningen till att silkscreen-trycket finns där överhuvudtaget är för att underlätta vid monteringen av kortet. Då dessa kort görs i en liten serie kommer de att monteras för hand och det är då till stor hjälp för montören att veta var respektive komponent skall placeras. Om kortet skulle monterats maskinellt hade silkscreen-trycket varit överflödigt.

I figur 2 visas ett exempel på det fysiska formatet för en hålmonterad IC-krets.



Figur 2, hålmonterad komponent

Precis som i föregående exempel är figur 2 en skärmdump från Eagle's kortvy. Denna gång visas istället gröna fält som symboliserar den exponerade kopparn där komponenten ska lödas fast. Till skillnad från föregående exempel är detta en hålmonterad IC-krets, detta betyder att det måste borras hål i kortet som komponentbenen sticks igenom för att sedan lödas fast från undersidan av kortet.

Det finns flera fördelar med att använda ytmonterade komponenter framför hålmonterat, men självklart finns det även några nackdelar att ta i beaktande.

Några fördelar med ytmonterade komponenter:

- Då det inte behövs några genomgående hål i kortet slipper korttillverkaren den extra process borrningen innebär. I större serietillverkningar innebär denna process ett moment som i slutändan kan ta ansenligt mycket tid och på grund av detta leda till en ökad kostnad.

- Komponenter i ett ytmonterat utförande är i regel alltid mindre än om man hade valt en hålmonterad variant. Detta leder till att man kan göra korten mindre eller få in fler komponenter på samma kort. Även här leder detta till en lägre kostnad.

- I dagens monteringsprocess för större serietillverkningar placeras ytmonterade komponenter ut maskinellt av ”pick-&-place”-robotar. Denna utplacering är betydligt snabbare för ytmonterade komponenter då dessa endast behöver läggas på rätt position på kortet, medans en hålmonterad komponent måste passas in så att respektive komponentben förs ner i rätt hål. Detta leder till en snabbare tillverkning, som i sin tur leder till en lägre kostnad.

- Kortare avstånd leder till lägre induktanser och mindre störningskänslighet vid höga frekvenser.

Några nackdelar med att använda ytmonterade komponenter:

- Då ytmonterade komponenter i regel är väldigt små, kan en manuell montering av dessa vara tidsödande och kräva mikroskop. Detta kan även försvåra vid felsökning och reparation.

- Kylningen av komponenterna kan bli ett problem, även i detta fall är detta relaterat till komponentens storlek.

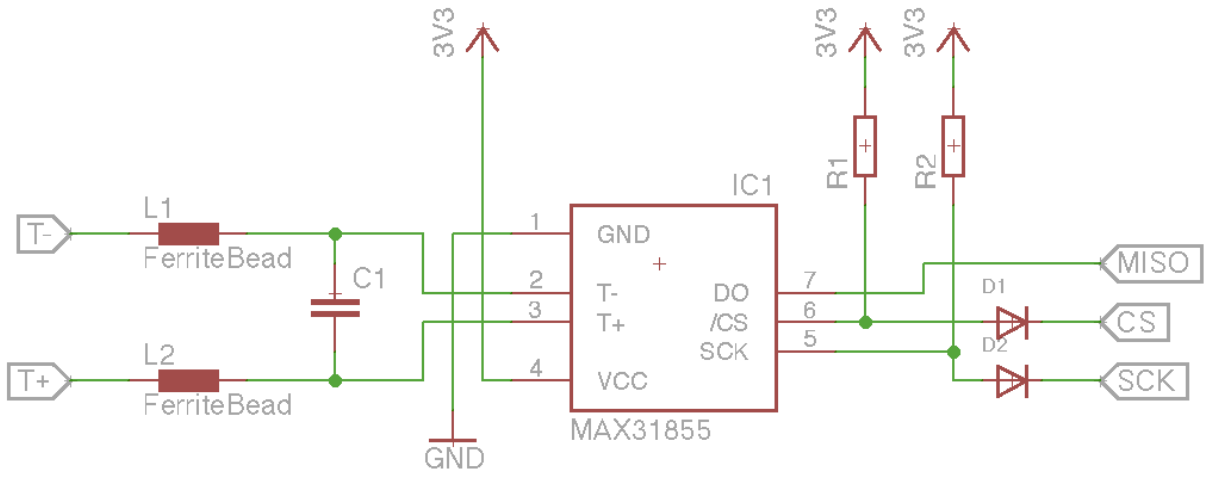
Majoriteten av den elektronik som tillverkas idag innehåller en stor del ytmonterade komponenter. Detta är framförallt av kostnadseffektiva skäl samt att detta ger konstruktören möjlighet att göra enheten väsentligt mycket mindre än om det hade använts hålmonterade komponenter. Dessutom är det en hel del komponenter som enbart finns att tillgå som ytmonterade.

2.1.2 CAD

Designen påbörjas genom att schemat ritas. När detta görs så väljs komponenter ur ett bibliotek och dessa förs in i schemat. I biblioteket ligger en schemasymbol för komponenten, samt ”footprinten”. Den schematiska symbolen är den man är mest van vid att se och är i princip universell för alla komponenter av samma typ, ”footprinten” däremot motsvara det fysiska formatet för komponenten i fråga och varierar många gånger mellan olika komponenter trots att de är av samma karaktär. De allra flesta passiva standardkomponenter (såsom motstånd, kondensatorer, dioder etc.) finns redan i Eagle’s grundbibliotek, men för en del komponenter är det nödvändigt att skapa desigen själv. I databladet för den aktuella komponenten ges ofta ett förslag på hur designen ska utformas för det fysiska formatet vid monteringen.

När en komponent har valts ur biblioteket så placerars den ut på schemat. Efter att ha placerat ut de komponenter som ska användas, finns det möjlighet att ange namn och attribut för respektive komponent. Attributen sätts till aktuellt komponentvärde (resistans, kapacitans etc.). Att ange komponentvärden redan i schemat är till stor hjälp när det sedan skall skapas en BOM-lista för beställning av komponenter. Sedan att är det dags att definiera kopplingarna mellan de olika komponenterna. De olika kopplingarna mellan komponenterna kallas för ”nets” och är grundläggande för att kunna översätta schemat till en färdig kretskortsdesign.

I figur 3 visas ett exempel på hur en del av schemat kan se ut.

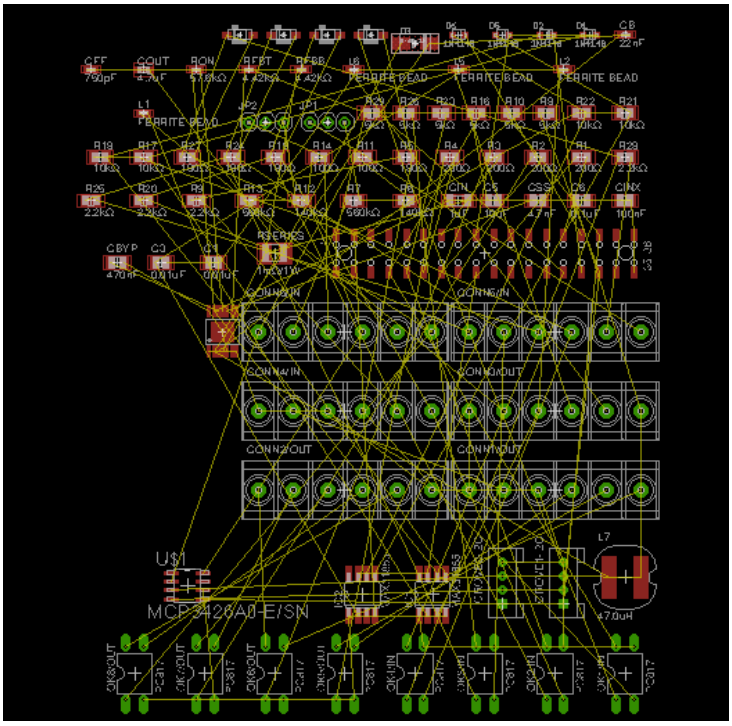


Figur 3, schematisk vy

I höger och vänstersidorna av schemat visas små lådor med text i. Dessa kallas för ”labels” och används för att knyta ihop alla delscheman med varandra. Alla anslutningar med samma ”label” får en faktisk anslutning till varandra även om det inte går att se några ”nets” på schemat. Detsamma gäller för spänningsmatning och jord. Detta gör schemat betydligt mycket mer överskådligt. Fullständigt schema över expansionskortet finns under ”Bilaga, Hårdvara”

När alla komponenter är utplacerade och alla anslutningar är dragna är schemat färdigt, och arbetet kan gå vidare till den fysiska utformningen av kretskortet.

Vid övergång från schemavyn till kortvyn i Eagle så ligger till en början komponenterna utspridda över arbetsytan.



Figur 4, kortvy i Eagle

I figur 4 visas komponenternas fysiska utformning till skillnad från schemasymbolerna som tidigare visades i schemanvyn. De gula linjerna i bilden kallas ”airwires” och symboliserar anslutningarna mellan komponenterna. Nu börjar arbetet med att placera ut komponenterna på kortet, men först definieras storleken på kortet då det ska placeras i en befintlig inkapsling. När formatet på kortet stämmer placeras andra platsspecifika detaljer och komponenter ut. I detta fall gäller det även skruvplintarna, stiftlisten till processorkortet samt ett par skruvhål för att montera fast kortet i sin inkapsling. När dessa komponenter ligger på rätt position kan arbetet med att placera ut resterande komponenter fortsätta.

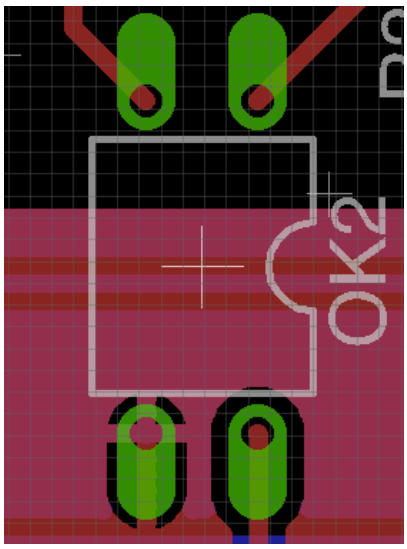
När det kommer till att placera ut komponenterna är det till stor hjälp att ha schema nära till hands. Då kan det med lätthet utrönas vilka passiva komponenter som hänger ihop och till vilken funktion dessa hör. Det är oftast bäst att placera ut grupper av komponenterna med hänsyn till vilken funktion dessa har, samt att hålla avståndet och ledningsbanorna mellan dessa så korta som möjligt. Ofta behövs det flyttas runt och roteras komponenter flera gånger innan placeringen blir rätt.

Efter att utplaceringen är avklarad kan ledningsbanorna börja dras på kortet. Då kretskortet kommer att ha två signallager finns det möjlighet att dra ledningsbanor på både ovansidan och undersidan. När ledningsbanorna ska dras på kortet är de gula linjerna (”airwires”) i föregående bild till stor hjälp, då dess visar vilka anslutningar som behöver göras. Det kan ta lite tid att dra alla ledningsbanor då dessa måste dras med hänsyn till tidigare dragna ledningsbanor och komponenter. Lyckligtvis finns det även möjlighet att använda undersidan av kortet för att dra ledningsbanorna, detta är något som behövs göras ganska ofta då ledningsbanorna korsar varandra.

När det görs en övergång av en ledningsbana mellan ovan- och undersidan eller tvärt om, används vior. Detta är precis som namnet antyder en koppling via det ena lagret till det andra. Vid tillverkning av större serier av kretskort är det fördelaktigt att hålla antalet vior nere, detta för att tillverkningen av en via innebär en borroperation per via, vilket i sin tur är en tidskrävande åtgärd.

Något som behöver anslutas till väldigt många komponenter är jordreferensen. Om jordreferensen hade dragits till alla komponenter via ledningsbanor så hade det i slutändan blivit väldigt många ledningsbanor. Dessutom hade problem kunnat uppstå då de smala ledningsbanorna på kortet faktiskt både har en resistans och induktans, även om den är låg så kan den inte försummas. Detta hade kunnat leda till potentialskillnad mellan olika jordpunkter på kortet, vilket i sin tur hade kunna leda till att elektroniken inte hade fungerat speciellt bra.

Lösningen på detta problem är att göra ett jordplan. Detta görs genom att fylla alla tomma utrymmen med koppar som ansluts till jord. Detta kopparplan flyter fram till ett ställbart isolationsavstånd till övriga ledningsbanor och ”paddar”. Detta läggs på både det övre och det undre lagret. I figur 5 visas hur jordplanet har fyllt ut tomrummet och samtidigt gett en jordanslutning.



Figur 5, jordplan

Figur 5 visar en av optokopplarna på ingångsidan. Då galvanisk separation mellan den externa ingången och processoringången är önskvärt, har det här valts att begränsa jordplanets utbredning på den externa ingångssidan. Detta för att här få ett större isolationsavstånd då högre externa spänningsnivåer kan förekomma.

När det väl har dragits en ledningsbana mellan de punkterna som ska anslutas, försvinner den gula linjen som symboliserade anslutningen. Eagle håller räkningen på hur många resterande anslutningar som behöver dras, vilket är till stor hjälp för att se när arbetet med ledningsbanorna är färdigt.

När alla anslutningar är dragna så går det att använda ett verktyg i Eagle som heter ERC, detta står för Electrical Rule Check och kontrollerar så att alla anslutningar har gjorts på ett korrekt sätt.

Det finns även annat verktyg som heter DRC, detta står för Design Rule Check. Detta verktyg är till för att kontrollera layouten uppfyller kraven som tillverkaren ställer på designen för att kunna tillverka kortet. Det kan till exempel vara minsta genomförbara avstånd mellan två ledningsbanor, minsta storlek på vior eller avstånd mellan en ”pad” och ett borrat hål. Det kan vara ganska många parametrar att ställa in, så lyckligtvis erbjuder de allra flesta tillverkare en nerladdningsbar fil som innehåller alla dessa regler. Denna fil kan läsas in i DRC:n och låta denna kontrollera designen av sitt kort. Ibland upptäcks fel som kan behöva rättas till.

När design är färdig återstår bara två saker. Det ena är att exportera sin design till GERBER-filer och det andra är att generera en BOM-lista.

GERBER-filerna definierar exakt hur de olika lagren på kortet ska tillverkas så att dimensionerna blir rätt och varje lager får en egen GERBER-fil. Tillverkaren behöver inte bara det båda kopparlagrena utan även silkscreen-lagret, lödmask-lagret och borrfilen.

Lödmasken består av ett skyddande lager av lack som läggs över hela kortet förutom på de ytor där komponenterna ska lödas fast. Detta är speciellt viktigt när det används ytmonterade komponenter där avstånden mellan komponentbenen oftast är litet. Om det saknas lödmask finns det risk att lödtennet flyter ut över två eller flera komponentben med kortslutning som följd.

Borrfilen definierar var hålen ska borras och vilken dimension det aktuella hålet har.

Det finns många insticksmoduler till Eagle som kan ge väldigt användbara funktioner. Ett av dessa insticksmoduler kan generera en BOM-lista. BOM står för Bill Of Materials och är en lista över de komponenter som används. Denna lista kan exporteras som en CSV-fil som i sin tur kan öppnas med något tabellhanteringsprogram. När BOM-lista har skapats påbörjas arbetet med att leta efter inköpsställen till komponenterna.

Listan som Eagle kan exportera innehåller bara komponentnamnet, värdet, och vilken typ av komponent det är, så den behöver kompletteras. Det som BOM-lista behöver kompletteras med kan vara information så som tillverkarens id för den specifika komponenten och försäljarens produkt-id. På så vis är det säkert att det alltid blir samma komponent varje gång komponenter beställs.

Många försäljare av komponenter har dessutom möjlighet att importera BOM-listan och skapa en varukorg utifrån denna. Detta underlättar avsevärt om det är många komponenter som behöver beställas.

Digikey valdes som leverantör av komponenter och Seeedstudio för tillverkning av kretskortet.

När kretskort och komponenter hade levererats till Bläckbild så monterades kortet och en grundläggande funktionstestning av kortet genomfördes.

2.2 Mjukvaruutveckling

Mjukvara till projektet är skriven i programmeringsspråket C/C++ och Arduinos egen utvecklingsmiljö är använd för att skriva, felsöka och verifiera kod.
Även en enkel texteditor för C/C++ (Notepad++) har använts för att skriva biblioteket.

Samtliga funktioner och program är testade mot expansionkortet och dess komponenter för att se att fullständig funktionalitet uppnås.
Programmeringen är objektorienterad med en konstruktor för klassen RWino som man använder för att kalla på alla funktioner.
Genom att göra koden objektorienterad görs programmen lättförståliga och lättanvända, genom enkla anrop till det skapade objektet görs operationer och syntax lätthanterlig.

2.3 Källkritik

De referenser som använts under arbetets gång kommer ifrån utvecklarna av de kommunikationsprotokoll som används samt ett webutvecklingsverktyg som användes vid konstruktionen av den switchade spänningsomvandlare. Det ligger i utvecklarnas och tillverkarnas intresse att tillhandahålla korrekt information. Dessutom är dessa protokoll implementerade i det utvecklade mjukvarubiblioteket och fullgod funktionalitet finnes. Den switchade spänningsomvandlaren är konstruerad utifrån den information som tillhandahölls från Texas Instrument och fungerar på en önskvärt sätt. Utifrån detta kan källorna antas som tillförlitliga.

3 Hårdvara

3.1 Temperaturmätning

En funktion som ofta krävs vid processtyrning är temperaturmätning. Expansionskortet har därför utrustats med möjligheten att mäta två separata temperaturer.

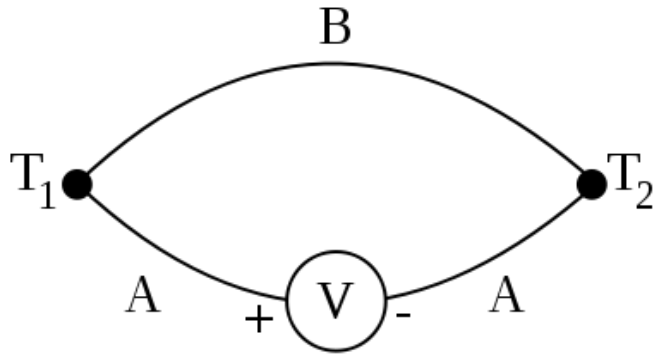
3.1.1 Termoelement

På expansionskortet används termoelement för att mäta temperatur. Termoelementet består av två olika metaller som är sammansvetsade i sensorspetsen och utnyttjar den termoelektriska effekten.

Den termoelektriska effekten, även känd som Seebeck effekten, bygger på att två olika sammansvetsade metaller skapar elektrisk spänning som beror på differenstemperaturen mellan det ”varma” och ”kalla” lödstället.

Med det ”varma” lödstället menas sensorspetsen där de båda metallerna är sammansvetsade och det ”kalla” lödstället är den punkten där man mäter spänningen.

I figur 6 nedan visas de två olika metallerna A & B, de två olika lödställena T1 & T2 samt spänningen V.



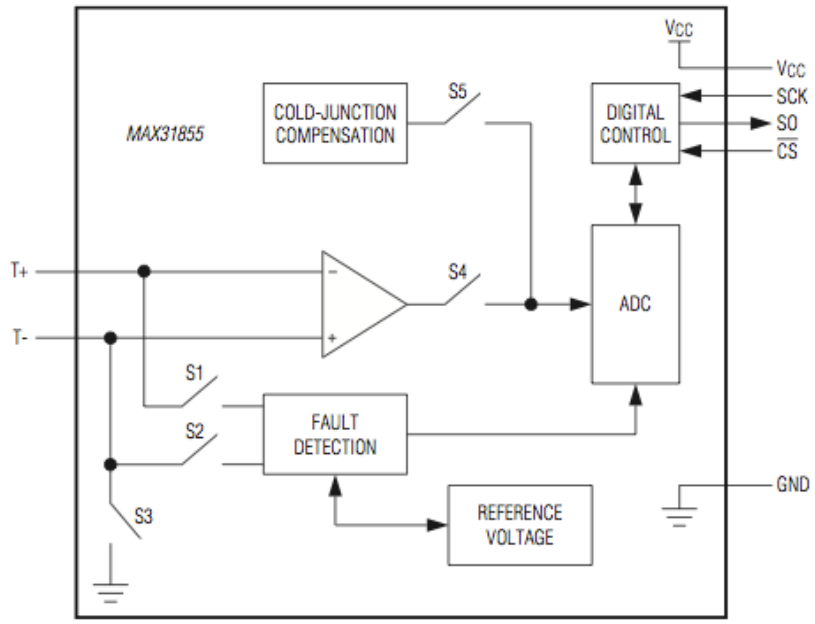
Figur 6, Seebeck-effekten

I detta fall ligger det kalla lödstället i skruvplinten där termoelement kopplas in, och ett termoelement av typen K har valts. De olika typerna av termoelement definieras av vilka olika metaller man använder. En K-givare består av metallegeringarna Chromel och Alumel.
Chromel är en legering av krom-nickel och alumel är en legering av aluminium-nickel.

Spänningen som ett termoelement alstrar är väldigt låg, i detta fall handlar det om 41 µV/°C, så någon typ av förstärkning krävs för att kunna använda termoelementet. Utöver förstärkningen behövs också temperaturen på det ”kalla” lödstället mätas, då spänningen som termoelementet alstrar är beroende på differenstemperaturen mellan det båda lödställena, dessutom måstes värdet samplas och konverteras till ett digital värde.

3.1.2 MAX31855

Det finns flera olika integrerade kretsar som löser uppgiften, här har det valts en krets från Maxim Integrated som heter MAX31855. I figur 7 visas ett blockschema över de olika funktionerna i MAX31855.



Figur 7, blockschema över MAX31855

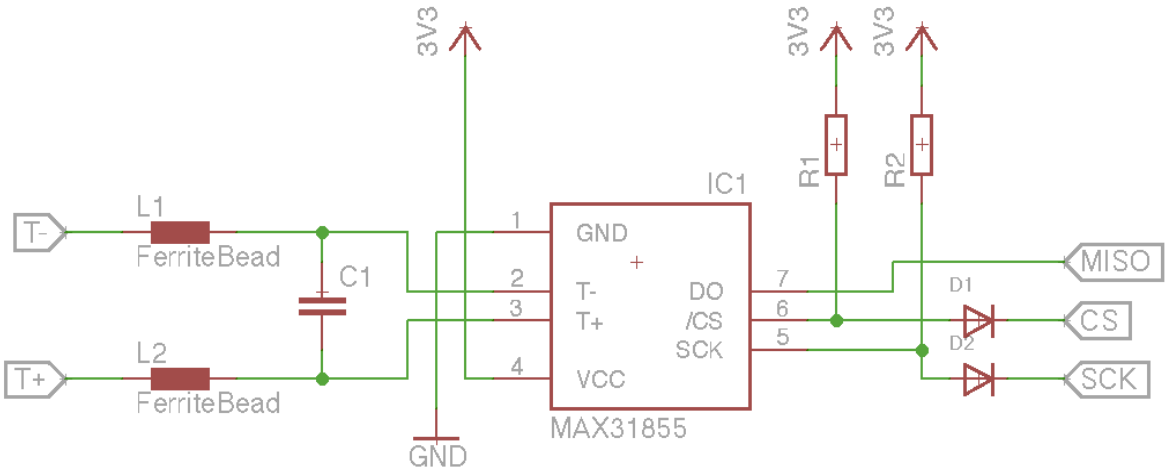
Termoelementet ansluts till T+ och T- som i sin tur är anslutna till en differentialförstärkare. Utspänningen från differentialförstärkaren kan anslutas till en A/D-omvandlare via den interna switchen S4. Den interna temperaturen mäts direkt på kiselplattan i kretsen och kan kopplas till AD-omvandlaren via den interna switchen S5. Anledningen till att kretsens interna temperatur mäts är för att man antar att denna temperatur är densamma som det ”kalla” lödställets temperatur. Det är således viktigt att kretsen placeras i nära anslutning till det ”kalla” lödstället så att temperaturen mellan dessa två inte skiljer sig för mycket.

Utöver temperaturmätningen finns även en feldetektering som kan detektera om termoelementet inte är anslutet, kortslutet till jord eller kortslutet till matningsspänningen.

Mätproceduren sköts av en digital kontroll som kopplar de olika switcharna till A/D-omvandlaren, sammanställer mätvärdet samt kommunicerar med processorn via SPI. Hur SPI fungerar beskrivs grundligare i kapitlet om mjukvara.

3.1.3 Konstruktion

Då spänningen som termoelementet alstrar är väldigt låg, blir ingången till termoelementförstärkaren känslig för brus. För att åtgärda detta ligger det en ferritpärla monterad i serie med varje ingång till förstärkaren samt en 10nF kondensator parallellt med ingångarna. Ferritpärlorna blockerar högfrekvent brus och kondensatorn absorberar transienter som annars skulle ha förstärkts av termoelementförstärkaren och påverkat mätvärdet.



Figur 8, schematisk vy över termoelementförstärkaren

Till höger av schemat i figur 8 visas in- och utgångarna från termoelementförstärkaren. Den matas med 3.3V och ingångarna också är avsedda för 3.3V. Då processorn arbetar med 5V krävs en nivåkonvertering för att inte riskera att förstöra ingångarna på termoelementförstärkaren.

Nivåkonverteringen är löst genom att använda dioder och pull-up motstånd. Då någon av processorns anslutna utgångar sätts höga backspänns dioden och ingångens nivå sätts till 3.3V genom pull-up motståndet. När samma utgång går låg så sätts utgången till diodens framspänningsfall. I detta fallet går det väldigt lite ström igenom dioden (0.33mA), detta resulterar i ett lågt framspänningsfall. Ur databladet för dioden (1N4148) kan det utläsas att framspänningsfallet för den ovan nämnda strömmen bli cirka 0.4V vilket är under den nivån (0.8V) som termoelementförstärkaren kräver för att uppfatta ingångssignalen som låg.

På expansionskortet sitter det två stycken termoelementförstärkare med passiva komponenter enligt schema ovan. Spänningsmatningen på 3.3V kommer från processorkortet där en spänningsstabilisator konverterar processorkortets matningsspänning på 5V till 3.3V.

Vid framtagning av drivrutiner till kortet upptäcktes att termoelementförstärkarens interna temperatur hade ett konstant offset fel på ungefär 7-8 °C. Efter en lång felsökning så upptäcktes det att matningsspänningen till termoelementförstärkaren hade ett spänningsrippel på 180mV peak to peak. Då detta kunde vara en av anledningarna till mätfelet så eftermonterades en 10µF kondensator parallellt mellan 3.3V och jord i nära anslutning till kretsen för att på så vis minska inspänningsripplet. Detta löste problemet och resulterade i en korrekt mätning av temperatur.

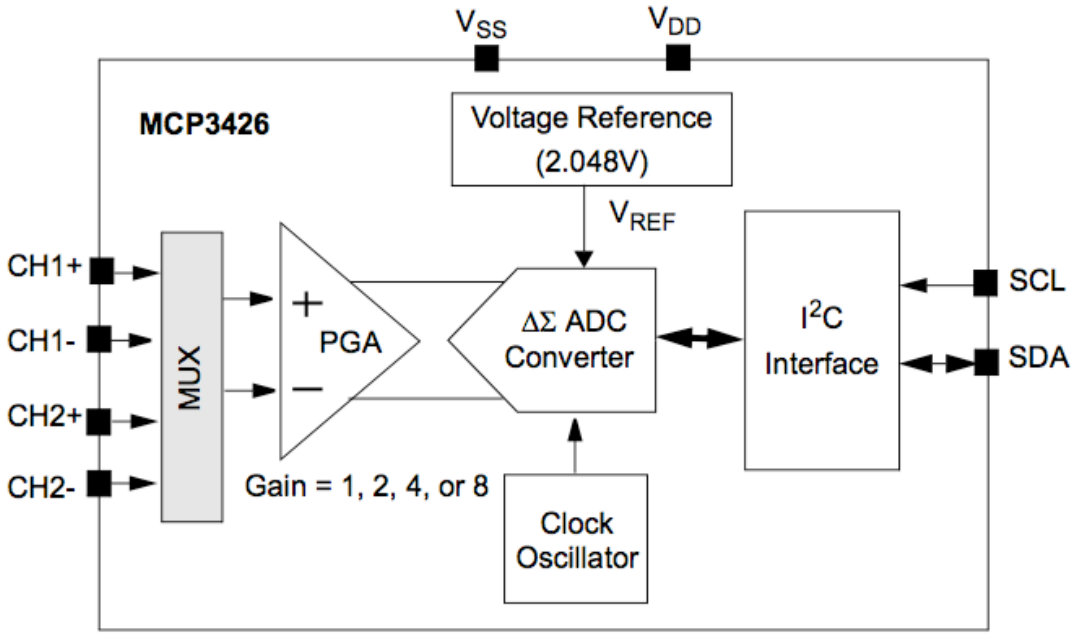
3.2 A/D-Omvandlare

Det finns många tillfällen då man behöver använda sig av analoga signaler. Många sensorer och givare presenterar mätvärdet analogt, antingen genom en varierande resistans, spänning eller ström. För att kunna använda dessa signaler i digitala sammanhang krävs en digital översättning av spänningsnivån. Detta sker med en A/D-omvandlare (Analog/Digital) som mäter spänningen och översätter detta till ett digital värde.

Det finns flera olika metoder att mäta den analoga signalen på. De två vanligaste teknikerna är successiv approximation och integrering.

3.2.1 MCP3426

På expansionskortet har det dock valts en A/D-omvandlare som använder en annan teknik som kallas Delta-Sigma[6]. Detta ger en relativt långsam mätning men resulterar i en hög precision. A/D-omvandlaren är tillverkad av Microchip och heter MCP3426[7]. I figur 8 visas ett blockschema hämtat från databladet över MCP3426: s olika funktioner.



Figur 9, blockschema över MCP3426

I högersidan av blockschemat i figur 9 visas de två ingångskanalerna (CH1 & CH2) till A/D-omvandlaren. Varje kanal består av ett differentialpar där den analoga insignalen utgörs av skillnaden mellan + och -. Detta möjliggör mätningar där signalen inte behöver förhålla sig till A/D-omvandlarens jordreferens, vilket kan vara väldigt användbart om insignalen inte har samma jordpotential som A/D-omvandlaren.

Efter ingångskanalerna sitter en multiplexer som möjliggör ett val mellan att mäta från kanal 1 eller 2.

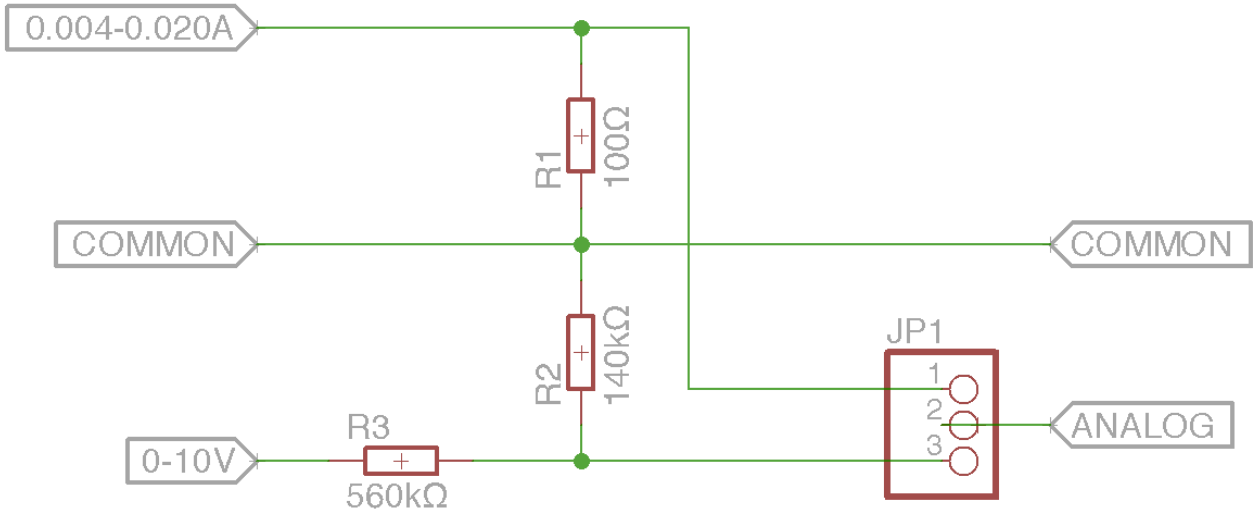
A/D-omvandlaren är även utrustad med en PGA (Programmable Gain Amplifier). Detta är en programmerbar differentialförstärkare som möjliggör en förstärkning av signalen innan den översätts till ett digital värde. Detta kan vara användbart vid mätning av små signaler. Förstärkningen kan ställas till 1,2,4 eller 8 gångers förstärkning och sätts via A/D-omvandlarens I^2C-kommunikation. Efter en eventuell förstärkning så går signalen vidare till Delta-Sigma-konverteringen. Det är här som insignalen översätt till ett digital värde. Konverteringen behöver en klockfrekvens och referensspänning, vilket denna A/D-omvandlaren har internt. Många A/D-omvandlare kräver en extern referensspänning, något som alltså inte behövs i detta fall.

Längst till vänster i blockschemat visas I^2C enheten som tar hand om kommunikationen mellan A/D-omvandlaren och processorn. Hur I^2C fungerar beskrivs kapitlet ”Mjukvara”.

3.2.2 Konstruktion

Inom industrin används ofta givare vars utsignal varierar mellan 4-20mA eller 0-10V. Expansionkortet är tänkt att arbeta på samma nivåer som är vanligt förekommande ute i industrin. Detta för att det då finns möjlighet att använda det stora utbudet av givare som finns.

Det behövs således funktioner för att mäta både spänning och ström. I figur 10 visas ett schema över hur detta är konstruerat på expansionskortet.



Figur 10, val mellan ström och spänningsmätning

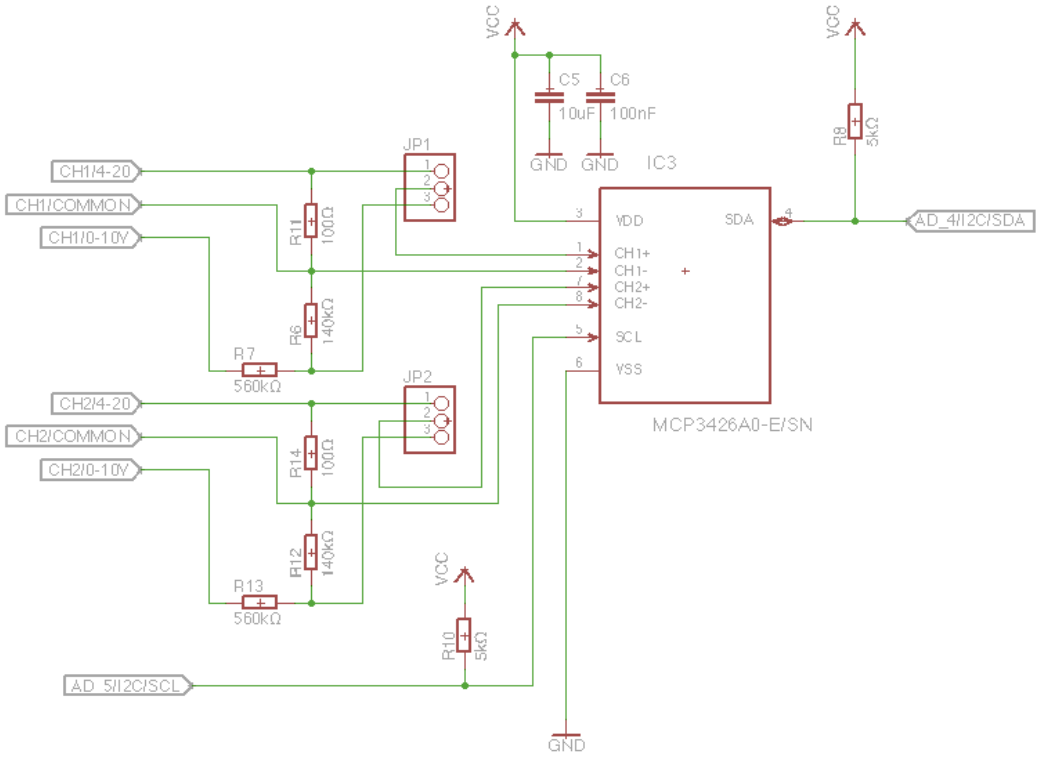
Valet mellan ström och spänningsmätning sker genom att en bygel som sitter monterad på en trepolig stiftlist flyttas ett steg. Om bygeln sätts mellan stift 1 och 2 kommer A/D-omvandlaren mäta en 4-20mA strömsignal och om bygeln sätts mellan stift 2 och 3 så kommer A/D-omvandlaren att mäta spänning.

Strömmätningen sker genom att strömsignalen leds genom ett shuntmotstånd med hög precision (0.1%). Spänningsfallet över shuntmotståndet kommer att vara proportionell mot strömmen som flyter igenom motstånden. Då A/D-omvandlare jämför insignalen mot sin interna referensspänning (2.048v) så finns det endast möjlighet att mäta signaler som är ±2.048V, alltså måste shuntmotståndet dimensioneras efter detta.

Oftast är det önskvärt att inte ha en allt för hög resistans på ett shuntmotstånd då detta ger upphov till ett spänningsfall på strömslingan. Detta spänningsfall måste givaren klarar av leverera. Då det är önskvärt att det maximala värdet på strömsignalen ska motsvara ett spänningsvärde som ligger nära det maximala värdet som kan mätas med A/D-omvandlare blir motståndsdimensioneringen i detta fall väldigt enkelt. Om det väljs ett motstånd på 100Ω kommer 20mA att motsvara ett spänningsfall på 2V vilket är väldigt nära referensspänningen, dessutom är 100Ω ett standardvärde så utbudet är stort och priset är relativt lågt, även för ett precsionsmotstånd.

Spänningsmätningen är avsedd för en signal mellan 0V och 10V. Då insignalen till A/D-omvandlaren inte får vara högre än referensspänningen behövs en spänningsdelning. När det ska mätas spänning är en hög ingångsimpedans att föredra, då ett högt strömuttag kan påverka mätkällan. Spänningsdelarens två motstånd dimensioneras så att spänningsfallet över det motståndet som spänningen mäts ligger nära A/D-omvandlarens interna referens (2.048V) vid den maximala spänningen som spänningssignalen kan anta (10V). Vad spänningsdelaren resulterar i är en nerskalning av spänningsignalen. I detta fall är en skalfaktor på 5:1 önskvärt, då en femtedel av 10V motsvarar 2V, vilket är tillräckligt nära A/D-omvandlarens begränsning på maximal insignal.

I figur 11 visas ett schema över hur den analoga mätkretsen är konstruerad på expansionskortet.



Figur 11, schemavy över A/D-omvandlaren med tillhörande passiva komponenter

Längst till vänster i schemat i figur 11visas ingångarna, med tillhörande shuntmotstånd och spänningsdelare.

När det kommer till kretsar som hanterar analoga signaler så är det är av största vikt att se till så att matningsspänningen till kretsen är så stabil och brusfri som möjligt. I detta fall är två stycken kondensatorer placerade i nära anslutning till matningsspänningens ingång. Den ena av dessa två kondensatorer är en tantalkondensator med en kapacitans på 10µF och den andra är en keramisk kondensator som har en kapacitans på 100nF. Kondensatorn på 10µF absorberar de långsamma större svängningarna medan kondensatorn på 100nF absorberar de små snabba svängningarna. Denna lösning är rekommenderad av tillverkaren och nämns i databladet för A/D-omvandlaren, men detta är i princip ett krav för att få stabila och pålitliga mätvärden.

I figur 11 visas även A/D-omvandlaren I^2C -kommunikationskanal (SDA), samt klockan till dessa (SCL). Båda dessa har även pull up-motstånd på 5kΩ för att få väldefinierade logiska nivåer.

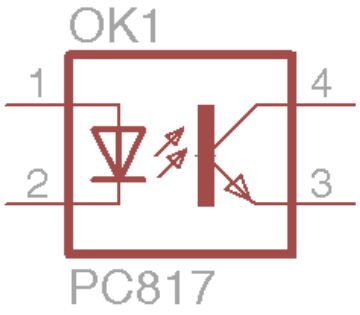
3.3 Digitala in- och utgångar

Då expansionskortet är tänkt att använda som en del i ett 24V system krävs åtgärder för att skydda ingångar från överspänning, då dessa är konstruerade för 5V. Det kan även vara intressant att ha galvanisk separation mellan in- och utgångarna på kortet och processorns in- och utgångar. Detta kan vara bra om man exempelvis har anslutit ett relä eller någon annan form av induktiv last på utgången då dessa vid frånslag kan generera en tillräckligt hög spänning för att förstöra processorns utgång. Likaså är det önskvärt att skydda processorns ingångar från överspänningar.

3.3.1 Optokopplare

Lösningen på ovanstående problem ges av en optokopplare per in- och utgång. Dessa erbjuder galvanisk separation och ger dessutom möjlighet att hantera större strömmar än vad processorns in- och utgångar klarar av.

I figur 12 visas principen för en optokopplare.

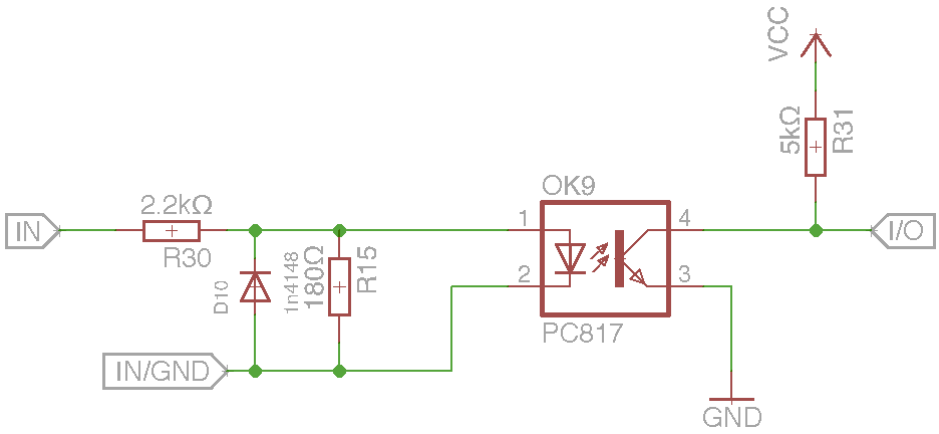


Figur 12, optokopplare

Ingång 1 och 2 är kopplade till en lysdiod och ingång 3 och 4 är kopplade till en fototransistor. När lysdioden tänds så sluter transistorn kretsen genom ingång 4 ner mot jord (3).

3.3.2 Konstruktion

De digitala ingångarna på expansionskortet är konstruerade enligt figur 13.



Figur 13, galvaniskt isolerad ingång baserad på optokopplare

På ingångssidan av optokopplaren är två motstånd samt en skyddsdiod placerad. Det första motståndet (2.2kΩ) är ett strömbegränsningsmotstånd. För att driva en lysdiod med en högre spänning än framspänningsfallet krävs det att man begränsar strömmen, i detta fall används ett motstånd för detta. Motståndet dimensioneras enligt följande:

$$\frac{Vin - Vfd}{Id} = R$$

Vin är matningsspänningen, Vfd är framspänningsfallet för dioden och Id är den strömmen som dioden behöver.

Framspänningsfallet i detta fall ligger på 1.2V och en diodström på 10mA. Då ingången är dimensionerad för 24V blir motståndet enligt följande:

$$\frac{24 - 1{,}2}{0.01} = 2280\Omega$$

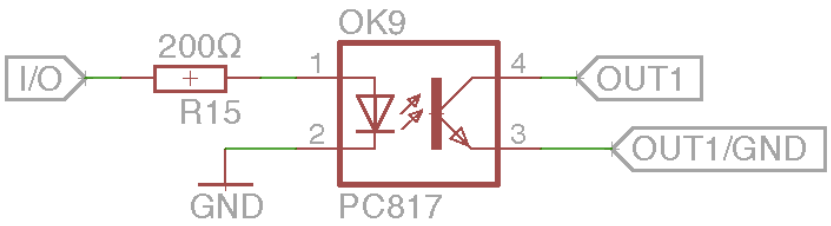
Då det är fördelaktigt att använda standardvärden på motstånd, både med hänsyn till pris och tillgänglighet, så väljs här ett värdet på 2.2kΩ.

Då spänningsfallet över detta motstånd blir 22.8V och strömmen som flyter igenom motståndet ligger på 10mA, blir effekten som motståndet utvecklar 228mW. Detta kan tyckas vara försumbart lite, men då det används ytmonterad motstånd så måste detta tas i beaktande. En vanlig effekttolerans på standardmotstånd i storleken (0805) är 125mW. Ytmonterade motstånd är i regel alltid mindre än hålmonterade och kan på grund av sin storlek inte avge speciellt mycket effekt utan att bli överhettade och i värsta fall börja brinna. I detta fall väljs ett motstånd som klarar av 500mW för att vara på den säkra sidan.

I föregående schema över optokopplare så visas ytterligare ett motstånd parallellt med lysdioden. Detta motstånd är till för att få bättre definierade logiska nivåer, lite som ett pull-down motstånd. Detta motstånd ligger på 180Ω och har som uppgift att dela ner spänningen. Då inspänningen når 15.9V blir spänningsfallet över motståndet 1.2V, vilket är framspänningsfallet för dioden och fototransistorn börjar leda.

Till höger i schemat så visas ett pull-up motstånd på 5kΩ samt en av processorns ingångar. Pull-up motståndet har som uppgift att ge mer väldefinierade logiska nivåer och är nödvändigt när det används öppen kollektor. Fototransistorns utgång är också av typen öppen-kollektor, med detta menas att den är transistorn sluter kretsen ner mot jord. När transistorn inte leder så kommer processorns ingång att känna av de 5V som ges genom pull-up motståndet. På så vis blir ingången inverterad, men detta är lätt att kompensera i mjukvaran.

De digitala utgångarna på expansionskortet är något enklare i sin design, men bygger fortfarande på principen med att använda optokopplare för att få galvanisk separation mellan processorns känsliga elektronik och externt inkopplade enheter. I figur 14 visas schemat för hur de digitala utgångarna är konstruerade.



Figur 14, galvaniskt isolerad utgång baserad på optokopplare

Till vänster i schemat som visas i figur 14 visas hur en av processorns utgångar är ansluten till optokopplarens ingång via ett strömbegränsningsmotstånd. Detta motstånd har precis som tidigare i uppgift att begränsa strömmen genom dioden. I detta fall är motståndet väsentligt mycket lägre än vad det var när optokopplaren användes som en digital ingång. Detta beror på att spänningen från processorn utgång endast är 5V. Precis som tidigare blir utgången av typen öppen-kollektor, det vill säga att kretsen sluts ner mot jord.

3.4 Switchad DC/DC-omvandlare

Expansionskortet är dimensionerat för 24V-nivåer, således borde ju också matningsspänningen vara 24V. På både processorkortet och expansionskortet ligger spänningsnivåerna på 5V och 3.3V. Det behövs alltså en konvertering av den externa matningsspänningen från 24V ner till 5V.

3.4.1 Linjär spänningsstabilisering

Det finns flera sätt att lösa ovanstående problem. Det allra enklaste hade varit att använda integrerad spänningsstabilisator. Dessa finns både med ställbar och fast utspänning, och kräver endast ett litet antal externa passiva komponenter för att fungera.

En integrerad spänningsstabilisator ser till så att spänningsfallet över den alltid är skillnaden mellan matningsspänningen och den önskade utspänningen. Denna metod är att föredra om applikationen inte kräver några större strömmar och/eller om matningsspänningen inte skiljer sig allt för mycket från den önskade utspänningen. Ett stort problem med integrerade spänningsstabilisatorer är att dessa utvecklar en effekt som värsta fall kan bli ganska hög, med krav på rejäl kylning som följd av detta.

Om en integrerad spänningsstabilisator hade valts till expansionskortet, där matningsspänningen är 24V, den önskade utspänningen 5V och en önskad utström från spänningsstabilisatorn på 1A. Förlusteffekten från spänningstabilisatorn blir således:

$$(24 - 5) * 1W = 19W$$

Detta är en alldeles för hög förlusteffekt i sammanhanget och hade ställt krav på en stor kylfläns monterad på spänningsstabilisatorn.

3.4.2 Switchad nerspänningsomvandlare

Det är i detta fall önskvärt med en switchad nerspänningsomvandlare då detta ger en betydligt högre verkningsgrad än om det hade använts en integrerad spänningsstabilisator.

Förenklat går det att säga att en switchad nerspänningsomvandlare skickar ut ström i pulser genom en induktor som i sin tur jämnar ut strömmen. Lasten ansluts efter induktorn och parallellt med en filterkondensator. För att räkna ut längden på pulserna som ska skickas genom induktorn mäter en reglerkrets utspänningen och kompenserar pulslängden.

3.4.3 Konstruktion

Det finns idag många olika reglerkretsar på marknaden. I detta fall har det valts en från Texas Instrument, dels för att deras utbud är väldigt stort men framförallt för att de erbjuder ett väldigt bra utvecklingsverktyg för att dimensionera switchade spänningsomvandlare. Utvecklingsverktyget heter Webench[5] och låter användaren ange parametrar såsom högsta och lägsta inspänning, önskad utspänning och ström. Webench ger därefter en mängd olika förslag baserade på Texas Instruments olika reglerkretsar, med tillhörande passiva komponenter och totalpris. Utöver detta går det även att få fram mycket relevant data om spänningsomvandlare, såsom verkningsgrad, termiska simuleringar och kompletta BOM-listor med alla komponenter som krävs.

För att dimensionera spänningsomvandlare behövs det information om hur mycket ström spänningsomvandlaren ska leverera. Detta görs lättast genom att gå igenom databladen för respektive strömförbrukande komponent och se hur mycket ström dessa maximalt kan dra. I tabell 1 följer strömförbrukningen för respektive komponent.

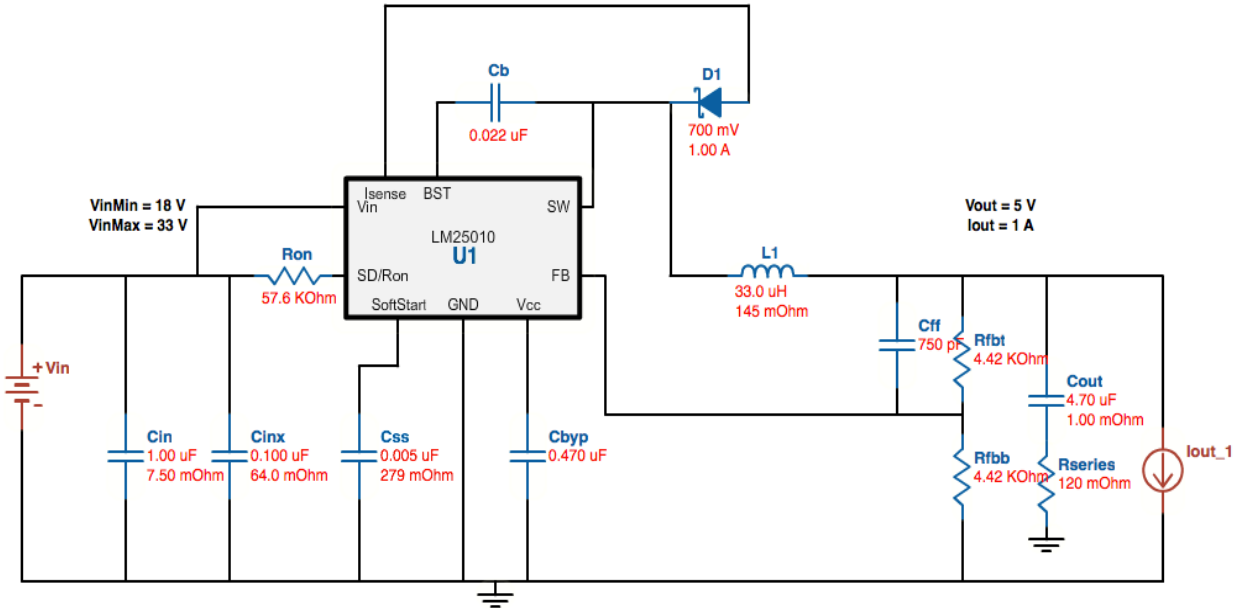
Komponent	Antal	Strömförbrukning per komponent
Processorkort (ej inräknad den externa wifi-modulen)	1	29.5mA
MCP3426	1	180µA
PC817, optokopplare	4	20mA
5kΩ Pull up-motstånd	6	1mA
10kΩ Pull up-motstånd	4	0.33mA
MAX31855	2	1.5mA
CC3000 (Wifi-modul)	1	300mA (endast vid sändning)

Tabell 1, strömförbrukning

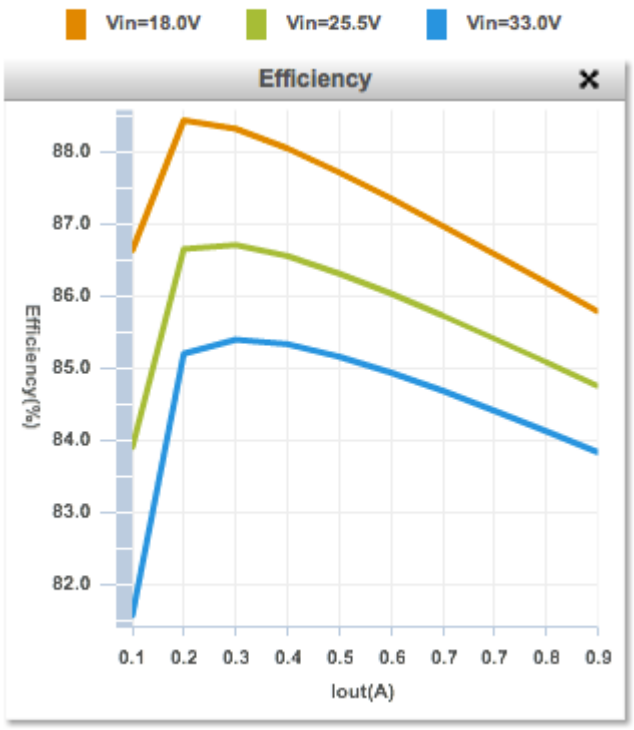
Om det förutsätts att alla komponenterna är aktiva samtidigt, samt att alla utgångar är höga, ger det en total strömförbrukning på 420mA. Utöver detta även önskvärt att ha möjlighet att koppla in några externa sensorer och använda någon typ av display på I2C-bussen, så om spänningsomvandlaren maximala utström väljs till 1A finns det fortfarande en viss kapacitet till extern utrustning.

På expansionskortet används en konstruktion baserad på en reglerkrets som heter LM25010. Det är en integrerad reglerkrets för strömmar upp till 1A och den ger en relativt hög verkningsgrad (84%) samt ett lågt totalpris för både reglerkretsen och de externa passiva komponenterna.

När det väl har valts en reglerkrets genererar Webench ett lösningsförslag med reglerkretsen och alla passiva komponenter som behövs för spänningsomvandlaren utifrån de angivna parametrarna. I figur 15 visas ett schema genererat av Webench utifrån reglerkretsen LM25010.



Figur 15, schema genererat av Webench



Figur 16, förhållande mellan ström och verkningsgrad

I figur 16 visas ett diagram över relationen mellan verkningsgrad och utström.Det går att se att verkningsgraden minskar om utströmmen avviker från 0.2A. Den största minskningen sker då utströmmen sjunker under 0.2A. Denna minskning av verkningsgrad ser drastisk ut, men skalan för verkningsgraden i ovanstående diagram är lite missvisande. Även om verkningsgraden sjunker när utströmmen ändras från det optimala 0.2-0.3A området så är det inget större problem, dels för att minskningen i verkningsgrad inte är speciellt stor och dels för att den aktuella strömmen vid en lägre verkningsgrad är relativt låg och inte ger upphov till några större förlusteffekter.

Då reglerkretsens switchfrekvens är relativt hög (525KHz), ställs höga krav på låga ESR-värden för kondensatorerna i nerspänningsomvandlare. ESR står för ekvivalent serie-resistans och är frekvensberoende. Lyckligtvis så specificerar Webench vilka ESR-värden respektive kondensator ska ha, vilket underlättar avsevärt i utformandet av nerspänningsomvandlaren.

4 Mjukvara

I projektet har det utvecklats en hel del mjukvara för styrning och mätning av expansionkortets funktioner. Det mesta av mjukvaran är utvecklad i Arduinos egen utvecklingsmiljö som är gjord till processorn som används på processorkortet. Det används en hel del kommunikations-protokoll som är standariserat i arduinon för att prata med de olika enheterna på kortet, mycket av dessa funktioner finns redan implementerat i Arduinos egna bibliotek.
Det som är gjort i projektet är att skräddarsy funktioner och operationer till expansionkortet och att inkludera ett lättanvänt mjukvarubibliotek med de viktigaste och mest användbara funktioner som kan tänkas behövas.

4.1 SPI-Protokoll

SPI(”Serial Peripheral Interface”) [2] protokollet är utvecklat av Motorola och är en buss för överföring av data mellan enheter. SPI protokollet tillåter seriell kommunikation mellan en Master och en eller flera Slaves.
Varje enhet har 4 stycken portar som är dedikerade vid användning av SPI bussen.

Slave select (SS): Då enheten är satt till ”Master” används denna port för att välja vilken ”Slave” den vill kommunicera med, är enheten en ”Slave” är porten satt till som en ingång som väljs då ”Mastern” vill kommunicera med sagd enhet.

Serial clock(SCLK): Denna port används för att ta emot klockpulser om enheten är satt som ”Slave” eller för att skicka pulser då den är en “Master”.

Master out/slave in(MOSI): Denna port används för att skicka data om enheten är satt som “Master” och för att ta emot data om den är satt som “Slave”.

Master in/slave out(MISO): Denna port används för att ta emot data om enheten är satt som “Master” och för att skicka data om den är satt som “Slave”.

Figur 17, SPI master/slave

I figur 17 visas hur Master och Slave är sammankopplade.
För överföring av data fungerar SPI genom användning av 8 bitars register. Ett register i varje enhet (Master/Slave) som är länkade via MOSI och MISO vilka tillsammans formar ett 16bitars register som är seriellt skiftat åtta bitar varje gång data överförs. 8bitars registret i mastern skickas över till slaven och slavens register skickas över till mastern där värdet kan avläsas utifrån tills dess att nästa registerskifte sker. En överföring av data via SPI går till på följande sätt:

Mastern sätter SS hög/låg beroende på om enheten är aktivt hög eller låg, till den Slave som den vill kommunicera med. Därefter skickas en bit via MISO/MOSI vid varje klockpuls till dess att det önskvärda antalet bitar har överförts. Olika enheter kan behöva överföra olika mängd data via SPI så antalet bitar som överförs kan ändras beroende på vilken enhet det kommuniceras med.

4.1.1 Implementering av SPI

I projektet används SPI till kommunikation mellan termoelementförstärkarna och AVR- processor, för avläsning av temperatur via ett termoelement av typen" K". SPI-kommunikationen är hårdvaruimplementerad, det vill säga att AVR-processorn har ett inbyggt stöd för att hantera SPI.

Data från termoelementet består av 4 bytes (32bitar) som innehåller termoelementets temperatur som ett 14bitars signerat tal, förstärkarens interna temperatur som är ett 12bitars signerat tal och till sist finns de felbitar som skickas ut för att kunna avgöra om förstärkaren och termoelementet fungerar korrekt. Det finns även 2 bitar som är reserverade från fabrik och inte har någon funktion i denna applikation.
Tabell 2 visar vilka bitar och vad de används till.

Bit	Beskrivning
31-18	14bitars signerat heltal för termoelementets temperatur
17	Reserverad
16	Felbit för att kolla om klockpulser fungerar som de ska 0 normalt
15-4	12Bitars signerat heltal för att läsa av termoelementförstärkarens interna temperatur
3	Reserverad
2	Felbit som kollar om termoelementet är kortsluten mot VCC, 0 normalt
1	Felbit som kollar om termoelementet är kortsluten mot jord, 0 normalt
0	Felbit som kollar om det finns ett inkopplat termoelement, 0 normalt

Tabell 2, utdata från termoelementförstärkare

4.1.2 Funktioner som använder SPI

I mjukvarubiblioteket finns det tre funktioner som utnyttjar SPI

Till termoelementförstärkaren används det två stycken funktioner, den ena är för att hämta ut termoelementets temperatur denna funktion heter ”readTemp”.
Den andra funktionen är ”readInternalTemp” här hämtas istället termoelementförstärkarens interna temperatur.

För att skriva data till SD-kortet används funktionen ”logValue” som sparar data till en fil på SD-kortet.

4.1.2.1 Funktionen *”readTemp”*

Inparameter: uint8_t thermoCouple.

thermoCouple används för att välja vilken av termoelementförstärkare data ska hämtas ifrån, 1 eller 2.

I ”readTemp” är det termoelementets temperatur som avläses, det vill säga de först 14 bitarna av datan som kommer från förstärkaren
Detta görs genom att föra över alla 32 bitar med hjälp av SPI och sedan maska ut de 14 översta bitarna med skiftesoperationer.
Detta 14 bitars tal konverteras sedan till en temperatur i grader Celsius genom att multiplicera 14bitars talet med 0.25 då varje bit är värd 0.25 Grader.
Om det skulle vara en negativ temperatur som uppmäts så görs det en koll om ”Sign” biten är hög eller låg och då returneras en negativ temperatur.

Funktionen returnerar en Double med avläst värde.

4.1.2.2 Funktionen *”readInternalTemp”*

Inparameter: uint8_t thermoCouple.

thermoCouple används för att välja vilken termoelementförstärkare som datan skall hämtas ifrån, 1 eller 2.

I ”readInternalTemp” är det termoelementförstärkarens interna temperatur som skall hämtas, detta är bit 14-4 från förstärkaren.
Dessa bitar hämtas på samma sätt som i funktionen ”readTemp” men skiftesoperationerna görs något annorlunda så att man får ut de aktuella bitarna.
Till skillnad från ”readTemp” så är varje bit värd 0,0625 grader Celsius. Då skall det utmaskade 12-bitars tal multipliceras med 0.0625.
Även här tas det hänsyn till om det skulle vara ett negativt värde med hjälp av första biten i 12-bitars talet.

Funktionen returnerar en Double med avläst värde.

4.1.2.3 Funktionen ”logValue”

Inparametrar: double value, char name[ ].

logValue skriver värdet “value” till en fil på SD-kortet som har namnet “name[ ]”.
Funktionen är tänkt för att lagra värden under en längre tid för att kunna skapa en loggfil med till exempel temperaturförändringar.

Funktionen stödjer filhantering för SD-kort med filsystemen FAT16/32.

4.2 I^2C protokoll

I^2C(”Inter-Integrated Circuit”)[1] protokollet är ett protokoll för seriell överföring av data mellan enheter
I^2C använder sig av två ledare för kommunikation, en av dem används för seriell data (SDA) och den andra för seriell klocka (SCL).

I^2C använder sig av minnesadressing, till skillnad från SPI som använder sig av SlaveSelect. Detta medför att man kan koppla samman ett stort antal enheter på samma buss och kommunicera med dem genom att använda deras adresser.
Beroende på vilka enheter som används så skiljer det sig på hur många adressbitar som används men det finns allt ifrån 3 bitars adressering med 8 olika adress upp till 16 bitars adressering med ~65000 adresser.

Även bussens kapacitans ger vissa begränsningar och I^2C används sällan på större avstånd än några meter. I^2C är främst avsett för kommunikation mellan olika enheter på samma kretskort.
I I^2C finns även felkontroll i form av ACK(acknowlegment) och NACK(None acknowlegment), för att kunna kontrollera att data överförts på ett korrekt sätt.
Detta ger en säker överföring och det det går att kontrollera att data som skickas verkligen kommer fram så som det är tänkt.

Då I^2C används för kommunikation görs det via Master och Slave noder. I^2C är dock ett multi-Master system vilket betyder att det kan finnas mer än en master i taget inkopplat på samma buss, och Slaves kan bli Masters under drift.

Master noder står för klockpulser som används för att överföra data samt adressering av Slaves.

Slave noder använder klockpulser samt adressering för att skicka eller behandla data.

En kommunikations kedja med I^2C går till på följande sätt.
Mastern börjar med att skicka ut en startbit följt av adressering till den slav den vill kommunicera med.
Sedan skickar Mastern om den vill skriva till eller läsa från Slave noden, detta görs med hjälp av en bit som sätts till 0 om den vill skriva eller 1 om den vill läsa från noden.
Om Slaven med adressen som Mastern skickat är inkopplad på I^2C bussen kommer den att svara med en ACK-bit att den är redo för att skicka eller mottaga data.
När förbindelsen är upprättad skickas data över i mängder om en byte i taget följt av en ACK/NACK bit för kontroll.
Detta fortsätter tills dess att mastern har skickat eller tagit emot aktuell data och därefter skickas en STOP bit som avslutar kommunikationen.

4.2.1 Implementering av I^2C

På expansionskortet används hårdvaruimplementerad I^2C för kommunikation mellan ADC (Analog/Digital Converter) och processorn.

Data hämtas från ADC med hjälp av I^2C, i form av 4 byte (32-bitar) där de första 2 byten innehåller data för avläsning och de sista 2 är för kontroll av inställningar på ADCn.
I ADCn går det med hjälp av I^2C att göra inställningar i dess register genom att skriva till ADCn från processorn.
Här finns det ett 8-bitars register som kan ändras för att få enheten att fungera på ett önskvärt sätt.
Inställningarna på ADCn bestämmer följande:

- Vilken kanal som skall användas, i detta fall finns det två kanaler, 1 och 2.

- Vilken avläsningmetod, antigen genom att kontinuerligt läsa av de strömmade bitarna från ADCn eller genom att göra en engångsläsning (One-Shot).

- Vilken upplösning som skall användas, det vill säga hur många bitars precision som används vid avläsning från ADCn. Det går att välja 12,14 eller 16 bitars upplösning, för olika noggrann mätning.

Val av upplösning påverkar också hur snabbt ADCn klarar av att göra nya mätningar där en lägre upplösning ger en snabbare mätningsfrekvens.

- 12-bitars upplösning ger 240 SPS (samples per second).
- 14-bitars upplösning ger 60 SPS.
- 12-bitars upplösning ger 15 SPS.

Till sist går det även ställa in vilken förstärkning som används vid avläsning av den analoga signalen.

Samtliga inställningar görs i ett konfigurationsregister som finns listat i tabell 3.

Bit	Beskrivning
7	Kontrollbit för att se om registret har blivit uppdaterat med senast skickad data
5-6	Kanalvals bitar, här väljs vilken av kanalerna man vill läsa data från
4	Konverterings inställning, väljs för kontinuerlig eller en-gångs avläsning av ADC
2-3	Val av upplösning 12/14/16 bitar
1-0	Val av förstärkning 1/2/4/8 ggr

Tabell 3, konfigurationsregister för A/D-omvandlare

ADCn använder sig av ett 3 bitars adresserings register som gör det möjligt att koppla in 8st enheter på samma I^2C -buss. Detta kräver dock enheter med olika adresser och enheten som används i detta fall (MCP3426) har en fast adress från tillverkaren och går inte att ändra. Inkoppling av flera enheter med samma adress är inte möjligt.

På expansionskortet finns även två stycken I^2C portar monterats för inkoppling av extern hårdvara som till exempel en LCD-display.

4.2.2 Funktioner som använder I^2C

I mjukvarubiblioteket finns det en funktion som använder I^2C, readFromADC.
Funktionen används för att hämta data från ADCn, för användning vid ström och spännings mätning.

4.2.2.1 Funktionen ”readFromADC”

Inparametrar: uint8_t Channel,Resolution.

Channel används för att välja vilken av de två kanalerna på ADCn som data skall hämtas ifrån, antigen 1 eller 2.

Resolution är val av vilken upplösning som användas då data hämtas, detta värde kan vara 12,14 eller 16 bitar beroende på hur noggrann mätning som önskas.

Funktionen börjar med att sätta konfigurations-registret med de önskade inställningarna av kanal och upplösning.
Data hämtas sedan ut via I^2C där de två första byten(16 bitar) innehåller mätdata från ADCn. Detta värde sparas in i en variabel för att sedan returneras som en double. Det görs även en kontroll så att ADCns register verkligen har uppdaterats med de nya inställningarna innan data returneras.

4.3 Funktionsbibliotek

För att kunna skapa en lättanvänd produkt har det skapats ett mjukvarubibliotek som innehåller de funktioner som kan behövas för att använda expansionkortet tillsammans med AVR-processorn.
En del av funktionerna är till för att läsa av data från enheter som termoförstärkare och ADC, medan andra är till för att konvertera värde eller lagra data till SD-kort.

4.3.1 Konstruktorn

Inparametrar: bool thermoCouple1,thermoCouple2.

Sätts till true om nämnt termoelement är inkopplat, variablerna används sedan till en selfcheck funktion för att se så allt fungerar som det ska.

Först i biblioteket finns konstruktorn för att skapa objektet RWino som sedan används för att kalla på de funktioner som finns med i biblioteket. Här sätts alla in och utgångar till de funktioner de är tänkta att användas till samt initierar de variabler och värden som behövs för att funktionerna ska kunna anropas på ett korrekt sätt.
Då det inte krävs några speciella parametrar för att initiera bilbioteket så lämnas inparametrarna till konstruktorn tom då alla processorns in- och utgångar skall sättas likadant vid varje uppstart.

4.3.2 Funktionen ”readMilliVolt”

Inparametrar: uint8_t channel, resolution.

”channel” är vilken kanal på ADCn ska användas.

”resolution” är med vilken upplösning som ska användas på ADCn.

”readMilliVolt” använder funktionen ”readFromADC” då ADCn används för spännings mätning.
Beroende på vilken upplösning som är vald så bearbetas värdet på olika sätt, då datan från ADCn ändras med upplösningsinställningarna.
Värdet som hämtas med ”readFromADC” (16 bitar) vänsterskiftas 2 steg då 14 bitars upplösning används, 4 steg vid 12 bitars upplösning och lämnas oförändrat då 16 bitars upplösning används.
Varje bit i värdet från ”readFromADC” blir då värt .0625 Volt.
Då spänningsdelning över ADCn används blir omsättningen 5 gånger lägre, vilket måste kompenseras för.
Slutgiltiga beräkningen av spänningen blir då:

$$värde * 0{,}0625 * 5V = spänning$$

Funktionen tar även hänsyn till negativ spänning och ger då ut ett negativt värde.

Returnerar en double med värdet i millivolt.

4.3.3 Funktionen ”readMicroAmp”

Inparametrar: uint8_t channel, resolution.

”channel” är vilken kanal på ADCn som används.

”resolution” är vilken upplösning som används på ADCn.

”readMicroAmp” använder funktionen ”readFromADC” då ADCn används för strömmätning.

Beroende på vilken upplösning som är vald så bearbetas värdet på olika sätt, då datan från ADCn ändras med upplösningsinställningarna.
Värdet som hämtas med ”readFromADC” (16 bitar) vänsterskiftas 2 steg då 14 bitars upplösning används, 4 steg vid 12 bitars upplösning och lämnas oförändrat då 16 bitars upplösning används.
Varje bit i värdet från ”readFromADC” blir då värt 625 µA.
Funktionen tar även hänsyn till negativ spänning.

Returnerar en double med värdet i mikrovolt.

4.3.4 Funktionen ”selfCheck”

”selfCheck” är till för att kontrollera så att termoelementenförstärkaren och ADCn fungerar så som det är tänkt.

För ADCn så hämtas befintlig data ut, då det alltid finns registerbitar som är ettställda kan det kontrolleras om ADCn är ansluten. Om värdet är större än noll så finns det en fungerande kommunikation mellan ADCn och AVR-processorn. Detta sagt så ger denna kontroll bara en indikation på om AVR-processorn har kontakt med ADCn och inte om ADC är ur funktion. Tyvärr finns det ingen inbyggd kontroll i kretsen utan det går bara att kontrollera om I^2C bussens kommunikation med ADCn fungerar.

För termoelementförstärkaren så kontrolleras felbitarna som ligger sist i det 32bitars tal som hämtas ut ifrån termoelementsförstärkaren. Genom att kontrollera de sista 3 bitarna finns det möjlighet att kontrollera om det finns ett anslutet termoelemen, om termoelementet är kortslutet mot jord eller matningspänningen.

Funktionen returnerar en boolsk variabel som är true om det inte upptäcks några fel under kontrollen och false annars.

Funktionen ger även detaljerade felmeddelande via seriemonitorn med vilket fel som har inträffat för att underlätta felsökning. Detta är dock enbart tillgängligt då processorkortet ansluts till en dator via USB.

4.3.5 Funktionen ”readInput”

Inparameter: uint8_t pin.

”pin” är vilken av de fyra optokopplarna som ska kontrolleras. 1,2,3 eller 4 som alternativ.

”readInput” gör en översättning från optokopplarna till ingångar på AVR-processorn då dessa inte har samma numrering. Då optokopplarna är aktivt låga kontrolleras den angivna ingången och om den är låg så finns det en spänning på den valda ingången.

Returnerar en boolsk variabel som är true då ingången är hög false annars.

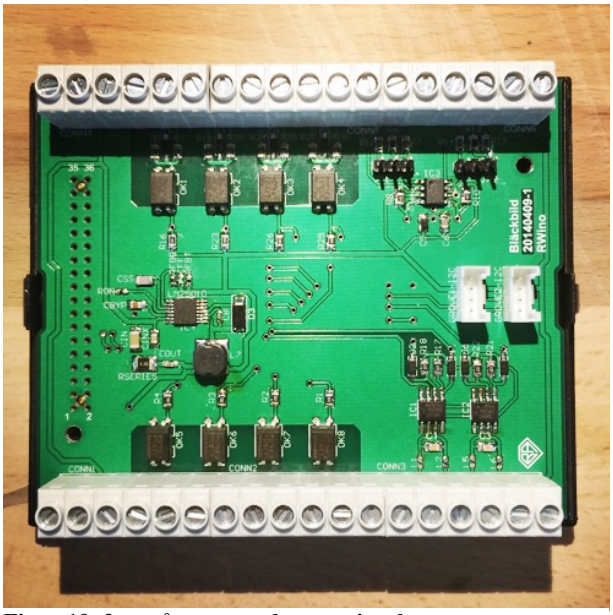
4.3.6 Funktionen ”setOutput”

Inparametrar uint8_t pin, bool state.

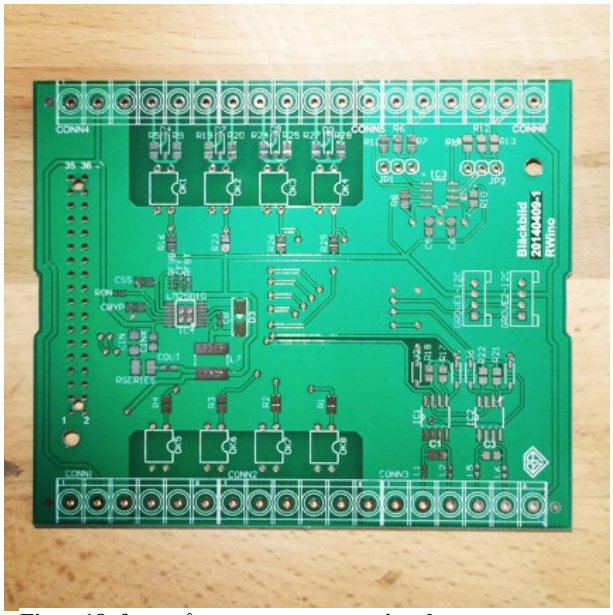
Pin är den utgång som ska ställas 1,2,3 eller 4.
State är vad utången ska sättas till Hög eller Låg. (true,false).

”setOutput” är till för att sätta utgångarna på expansionskortet. De kan antigen vara höga eller låga, även i denna funktion krävs en översättning av AVR-processorns pinnar till optokopplarnas utgångar. Här väljs vilken utgång och vad denna ska sättas till.

5 Resultat



Figur 19, foto på monterad expansionskort



Figur 18, foto på omonterat expansionskort

Figur 18 och 19 visar hårdvaruresultatet.

Hårdvaran resulterade i ett expansionskort som tillsammans med processorkortet installeras i en DIN-standardiserad inkapsling som kan monteras på en DIN-skena. Detta ger möjlighet att installera externa reläer i nära anslutning till modulen, då dessa också kan monteras på en DIN-skena.

Expansionskortet har funktioner som är vanligt förekommande i befintliga processtyrningar, temperaturmätning, digital in- och utgångar och avläsning av givarsignaler på 4-20mA och 0-10V. Dessutom är expansionskortet konstruerat för att fungera för spänningsnivåer på 24V, vilket är en vanligt förekommande spänning i industrisammanhang. Detta ger expansionskortet möjlighet att i vissa fall ersätta ett befintligt styrsystem, till exempel en PLC.

Redan från början var projektet tänkt att bygga Arduinos filosofi med öppen hårdvara med en lättillgänglig och intuitiv programmeringsmiljö. Till skillnad från tidigare hårdvarulösningar som bygger på samma grund är denna lösningen en del i en installationsfärdig helhetslösning där användare enkelt kan ansluta extern hårdvara via skruvplintarna. Även då detta i grund och botten är tänkt som ett utvecklingsverktyg för att snabbt kunna komma igång med utvecklingen av processtyrningen/övervakningen, är detta i många fall fullt dugligt till en färdig processtyrning.

Mjukvarumässigt har det implementerats grundläggande funktioner till expansionskortet. Dessa funktioner möjligör kontroll över in och utgångar, temperaturmätning, logging av mätvärde samt ström och spänningsmätning. Detta ger användaren möjligheter att nyttja dessa funktioner som delar i en större processstyrning/övervakning.

6 Slutsats

Under arbetes gång kan produktens egenskaper förändras. Någon hårdvarufunktion togs bort och någon ny lades till. Från början var tanken att expansionskortet skulle utrustas med en LIN-kommunikationsbuss, efter lite efterforskningar på hur LIN-kommunikationsbussen skulle implementeras började både vi och Per på Bäckbild tvivla på hur stor nytta expansionskortet skulle få av att ha en LIN-kommunikationsbuss. Detta ledde till att denna idé övergavs och istället fokuserades arbetet på ökad funktionalitet för expansionskortet. Istället för att konstruera ett interface för LIN-kommunikation så ersattes denna med en A/D-omvandlare som kan användas för att mäta båda strömsignaler och spänningssignaler. Detta kan vara användbart då någon typ av extern givare för att till exempel mäta flöde eller tryck skall användas, men kan även användas som ett användarinterface där användaren kan ställa någon form av inparameter till processorn via till exempel en potentiometer.

Hårdvarumässigt var det ännu en funktion som fick plockas bort från den ursprungliga planen. Från början var det tänkt att utrusta expansionskortet med en liten signaleringshögtalar (buzzer) som var tänkt att kunna agera som till exempel en varningssignal eller dylikt. Denna valdes inte bort på grund av bristande funktionalitet, då det hade varit användbart att ha en sådan funktion på expansionskortet, utan denna funktion fick tas bort på grund av att processorns anslutningar helt enkelt inte räckte till.

Det har även blivit tydligt att det är väldigt svårt att uppskatta tiden som ett projekt som detta kan ta. Om det beror på projektets natur eller vår bristande erfarenhet är svårt att säga. Själva CAD-arbetet kan gå väldigt fort om komponenterna som söks redan finns i biblioteket, men om det behövs användas komponenter som inte redan finns i biblioteket är det nödvändigt att skapa komponenten från grunden. Detta är ett arbete som beroende på komponenten komplexitet kan vara väldigt tidskrävande.

Något som hade bidragit till att öka känslan av Open Source hos projektet hade varit att dokumentera flera delmoment under designprocessen och publicerat dessa löpande på någon form utav hemsida eller blogg. Detta var en tanke som fanns från början av projektet men som har prioriterats bort under arbetets gång på grund av tidsbrist. Likaså ska scheman, designfiler och källkod publiceras öppet för att användarna ska kunna ta del av dessa, detta är något som kommer att göras inom en snar framtid, dock så kommer designen först att publiceras efter att den har modifieras så att kortet fungerar på ett önskvärt sätt.

Mjukvaran i projektet hade ifrån början en annan utformning än vad som slutprodukten innehåller. Detta då flera av de tänkta funktionerna redan har implementerats av de som levererar hårdvara till de tänkta applikationerna(WiFi, LCD-modul). Då detta inte är någon hårdvara som har utvecklats av oss är det mer aktuellt att mjukvara även levereras från tillverkaren av hårdvaran. Om det skulle ske ändringar i extern hårdvara så kan det behövas uppdateringar i mjukvaran. Detta hade ställt krav på uppdateringar i mjukvaru-biblioteket om biblioteket hade innehållit stöd för den berörda hårdvaran. Då det är ”Open-Source Hardware” så finns oftast mjukvarubibliotek tillgänligt för den externa hårdvaran och kan lätt inkluderas vid behov. Dessa bibliotek hålls updaterade för aktuell hårdvara. Detta gäller även i detta fall. När det kommer till PID regleringen så finns det ett väl fungerande mjukvarubibliotek redan implementerat av Brett Beauregard[4], som är lätt att inkludera vid behov. Då detta bibliotek är bättre eller lika bra som något vi skulle kunna åstadkomma fanns det ingen anledning att implementera detta.

6.1 Framtida utvecklingsmöjligheter

När väl prototypen var färdig och arbetet med mjukvaruutveckligen kom igång upptäcktes en del detaljer som hade vunnit på att ändras om det hade gjort en andra revision av kortet:

- Termoelementförstärkaren behöver bättre filtrering på ingångspäninngen. En 10µF’s och 100nF’s kondensator bör placeras mellan matningsspänningen och jord i nära anslutning till varje termoelementförstärkare. Detta problem nämns även i avsnittet ”Temperaturmätning”. Detta löstes i efterhand genom att löda fast en hålmonterad elektrolytkondensator över spänningsmatningen och jord, mest för att få en fungerande temperaturmätning så att arbetet med mjukvaruutvecklingen kunde fortskrida. Detta är dock något som ska tas med i hårdvarudesignen om det blir aktuellt med en andra revision av kortet.

- Silkscreen-trycket kunde mycket väl ha utökats med information om vad varje plintplats har för funktion. Detta kunde läggas på undersidan av kortet där det finns gott om utrymme. Vid inkoppling av extern hårdvara via plintarna måste användaren i dagsläget konsultera dokumentationen för att vara säker på att ansluta rätt. Detta är inget stort problem för en fast installation, men om enheten används som ett utvecklingssverktyg hade en lättillgänglig plintinformation underlättat avsevärt.

- Expansionskortet kunde utvecklas i symbios med ett nytt processorkort, som förhoppningsvis kunde utrustas med en större processor. På så vis hade flera funktioner fått plats på kortet. Den största begränsningen på expansionskortet är att anslutningarna till processorn är relativt få, då processorn är ganska liten.

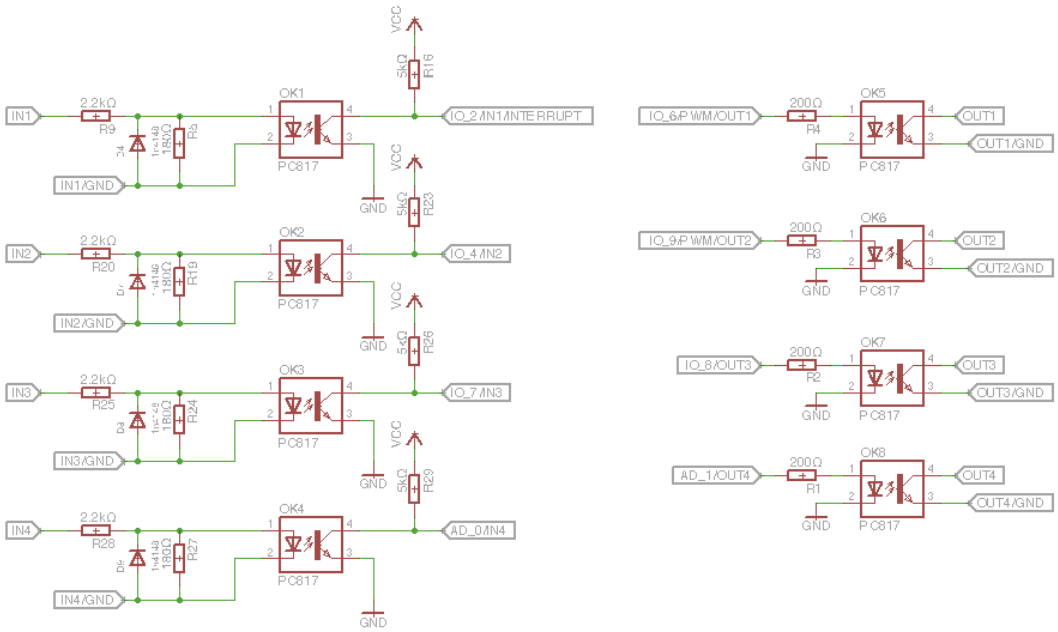
7 Referenser

1. http://www.nxp.com/documents/user_manual/UM10204.pdf (Maj 2014) NXP Semiconductors.
2. http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf (Maj 2014) Freescale Semiconductors.
3. http://www.oshwa.org/definition/ (Maj 2014) Open Source Hardware Association.
4. http://playground.arduino.cc/Code/PIDLibrary (Maj 2014) Arduino PID bibliotek av Brett Beauregard
5. http://www.ti.com/webench (Maj 2014) Texas Instruments web-utvecklingsvertyg
6. http://www.ti.com/lit/an/slyt423/slyt423.pdf (Juni 2014) How delta-sigma ADCs work, Texas Instrument
7. http://ww1.microchip.com/downloads/en/DeviceDoc/22226a.pdf (Juni 2014) Datablad för MCP3426, Microchip

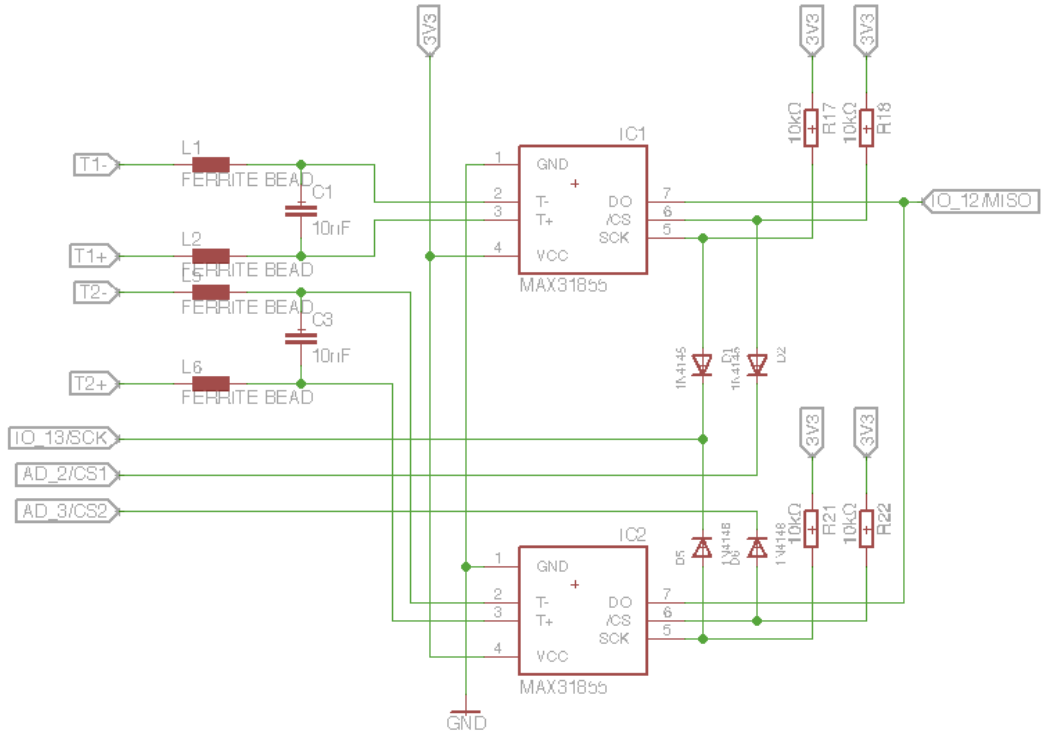
8 Bilagor

8.1 Hårdvara

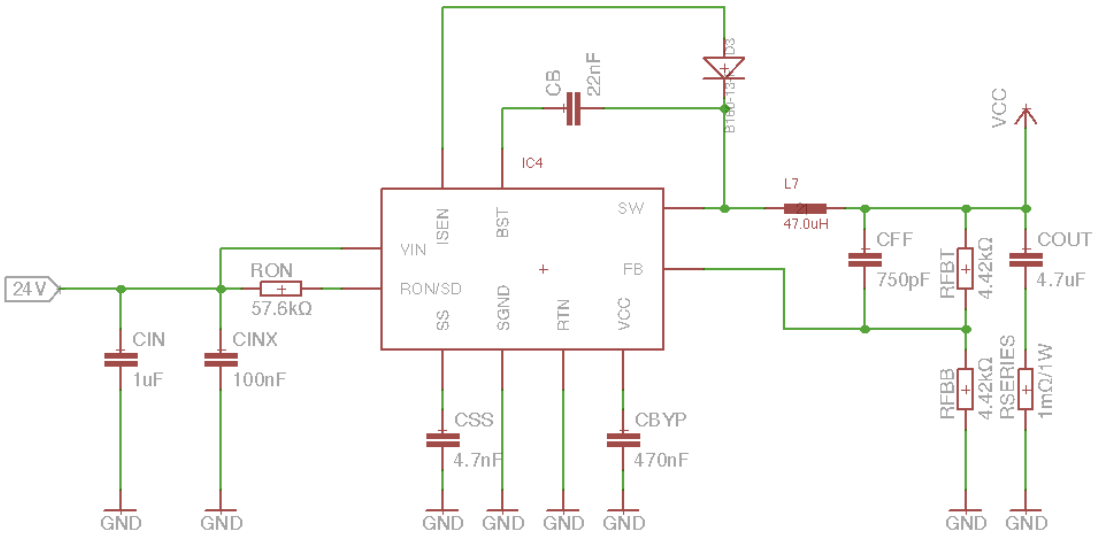
Schema, optiska in- och utgångar



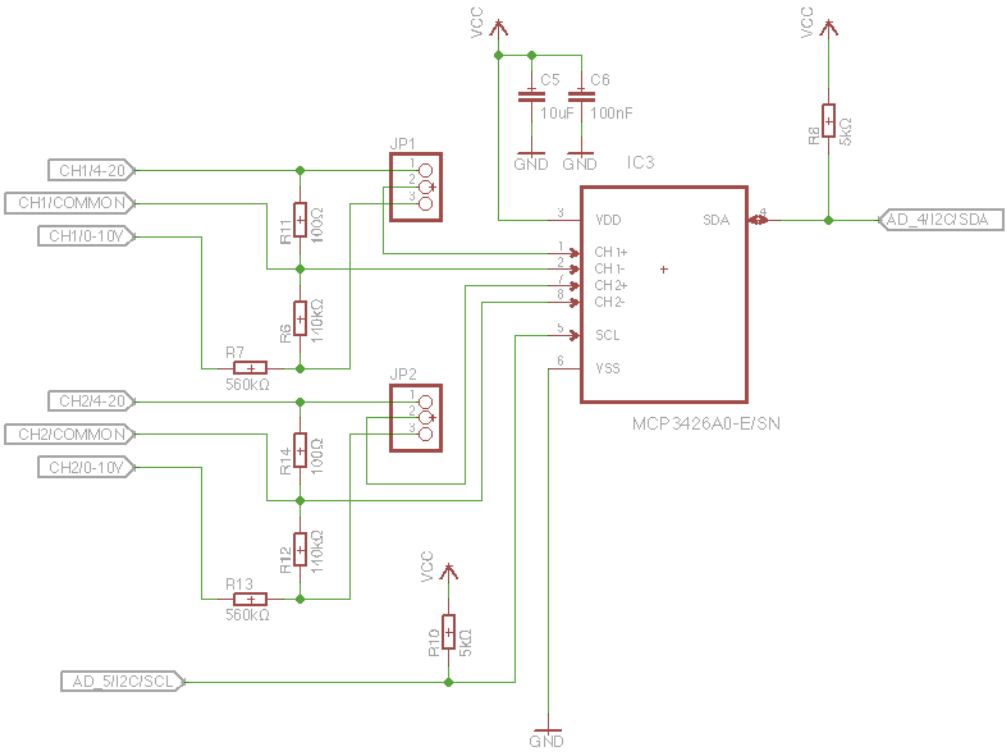
Schema, temperaturmätning



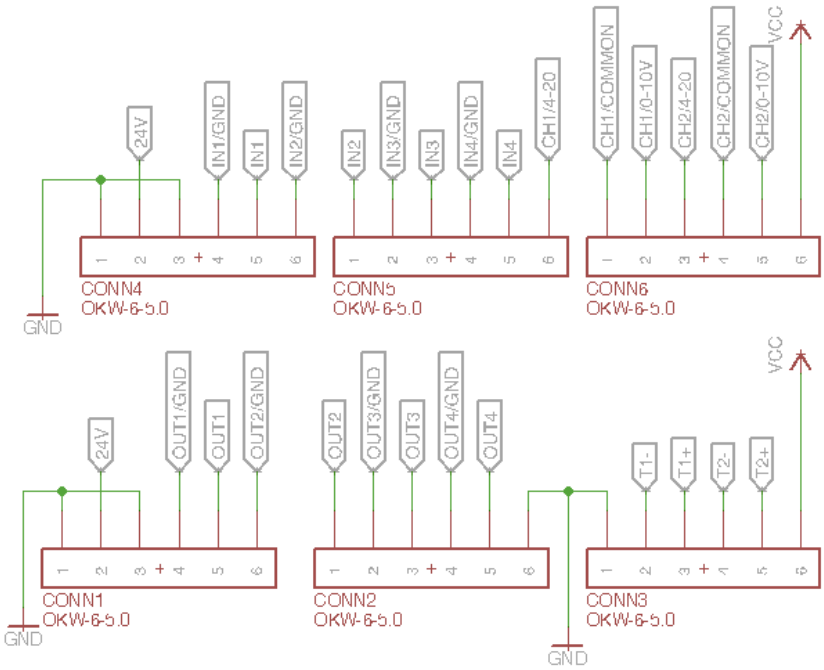
Schema, switchad DC/DC-omvandlare



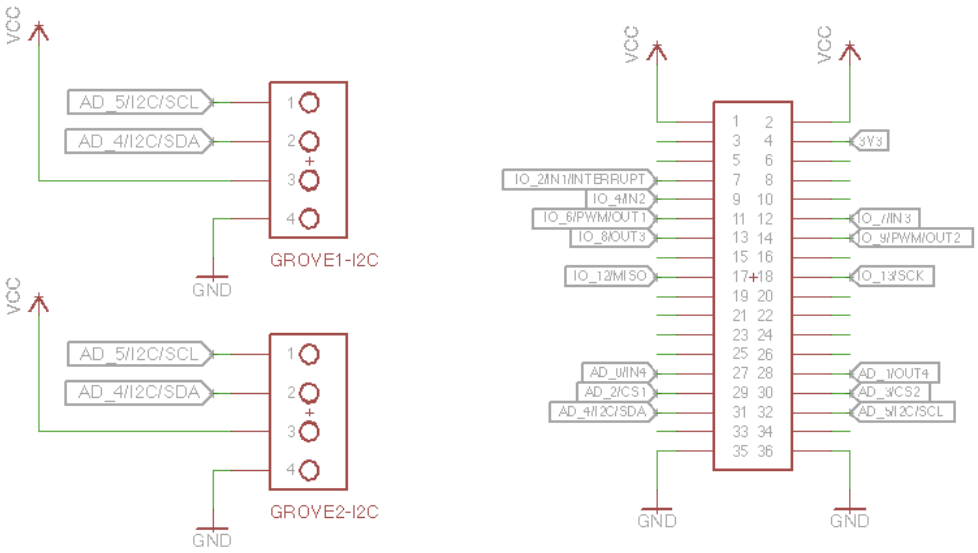
Schema, A/D-omvandlare



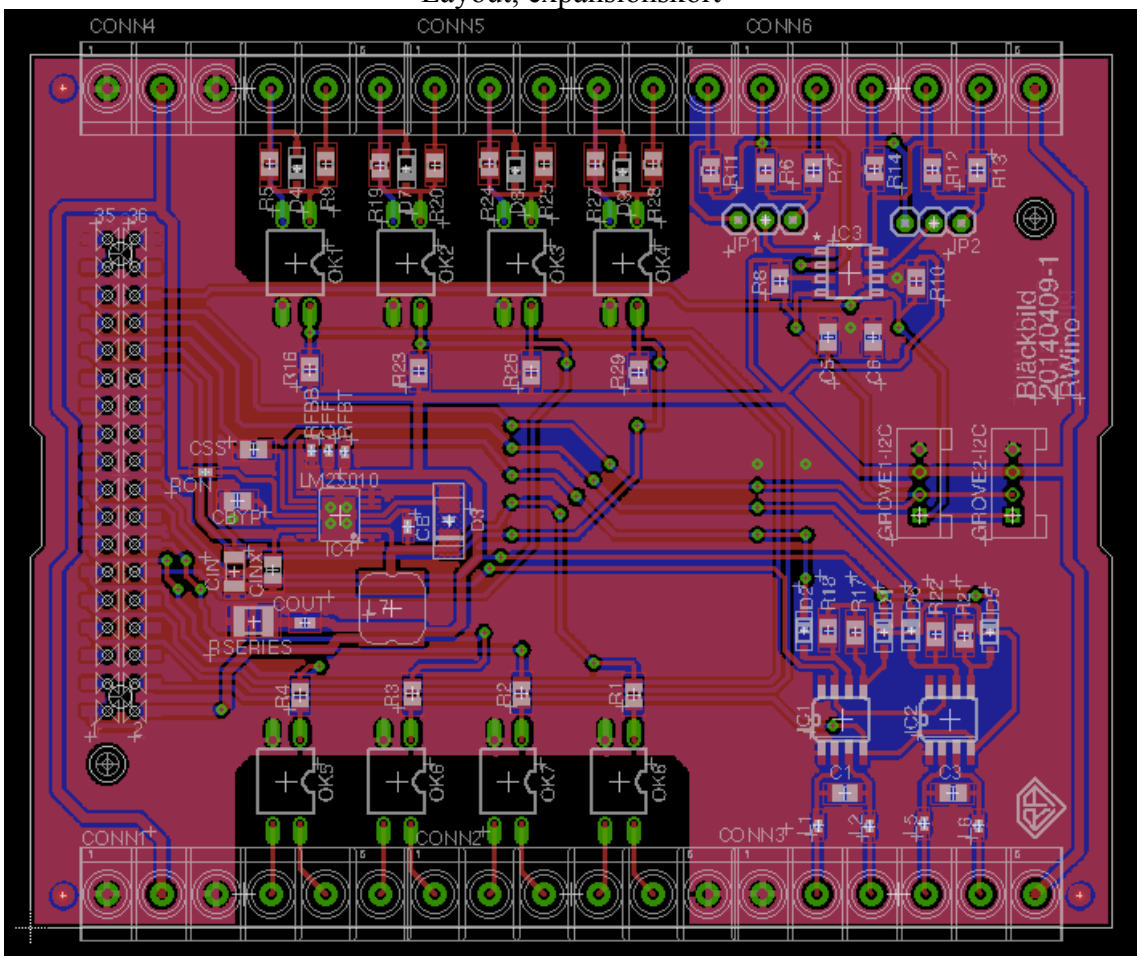
Schema, skruvplintar



Schema, stiftlist till processorkort och extern I^{2}C anslutning



Layout, expansionskort



8.2 Mjukvara

8.2.1 Header

```
/*
 * RWino.h
 */

#if (ARDUINO >= 100)
 #include "Arduino.h"
 #include "SPI.h"
 #include "SD.h"
 #include "Wire.h"
#else
 #include "WProgram.h"
#endif


class RWino {
public:

          RWino(bool thermoCouple1, bool thermoCouple2);
     bool selfCheck();
          double readTemp(uint8_t thermoCouple);
          double readInternalTemp(uint8_t thermoCouple);
     void logValue(double value, char name[]);
          uint16_t readFromADC(uint8_t channel, uint8_t
resolution);
          double readMilliVolt(uint8_t channel, uint8_t
resolution);
          double readMicroAmp(uint8_t channel, uint8_t
resolution);
          bool readInput(uint8_t pin);
          void setOutput(uint8_t pin, bool state);

private:
          bool thermo1, thermo2;
};
```

8.2.2 Source

```
/*
 * RWino.cpp
 */


#include "RWino.h"
#include "SPI.h"
#include "SD.h"
#include <avr/pgmspace.h>
#include <util/delay.h>
#include <stdlib.h>
#include <Wire.h>

RWino::RWino(bool thermoCouple1, bool thermoCouple2) { //
constructor, parameters to see if there are anny
thermocupples connected

        thermo1 = thermoCouple1;
    thermo2 = thermoCouple2;



        SD.begin(9);                          // Enables
use of processor Micro-SD card and wich pin is used
        Wire.begin();                         // Enables
the I2C interface

}

bool RWino::selfCheck(){    // checks if the adc is
working, and if the specified thermocupples are connected.
Serial prints in case of errors. Returns false if the
selfCheck fails

        bool controll = true;
        uint32_t check = 0;
        byte k;
    uint32_t value = 0;
    byte result = 0;

        Wire.requestFrom(0x6C, 4); // requests 4 bytes
from the ADC

        for(int i =0 ; i<4 ; i++){
```

```
                k = Wire.read();

                value |= k;
                }
                if(value == 0) {
                Serial.println("ADC not responding");
// if the value of the ADC return is 0 there is most likely
something wrong with the adc.
                controll = false;
        }

        if(thermo1){
                        pinMode(A2,OUTPUT);
                        // Transfer pin as output
                        digitalWrite(A2,LOW);
                // Sets the active and inactive
thermocouples
                        digitalWrite(A3,HIGH);
                        delay(1);

                                for(int i = 0;
i<4;i++){                               // Loops
thuough the bytes in memory, on the last 8 bits are needed
so no need to use the first 3

                                        result =
SPI.transfer(0x00);
                                }
                                if(result & 0x0001){
                                                //
if bit 0 is high no attached thermocouple

        Serial.println("Thermocouple 1: open circuit");
                                        controll =
false;
                                }
                if ((result & 0x0002)|(result & 0x0004)) {
        // if bit 2 or 3 are high either short circuit to
GND or VCC
                  Serial.println("Thermocouple 1: short
circuit, to GND or VCC");
                                        controll =
false;
                }
           digitalWrite(A2,HIGH);

        }
```

```
                if(thermo2){
                        pinMode(A3,OUTPUT);

        // Transfer pin as output
                        digitalWrite(A3,LOW);
                                                        //
Sets the active and inactive thermocouples
                        digitalWrite(A2,HIGH);
                        delay(1);


                                        for(int i = 0;
i<4;i++){                                       // Loops
thuough the bytes in memory, on the last 8 bits are needed
so no need to use the first 3

                                                result =
SPI.transfer(0x00);
                                        }
                                        if(result & 0x0001){
                                                        //
if bit 0 is high no attached thermocouple

        Serial.println("Thermocouple 2: open circuit");
                                                controll =
false;

                                        }if ((result &
0x0002)|(result & 0x0004)) {            // if bit 2 or 3 are
high either short circuit to GND or VCC
                  Serial.println("Thermocouple 2: short
circuit, to GND or VCC");
                                                controll =
false;
              }
           digitalWrite(A3,HIGH);

        }

return controll;
}

double RWino::readTemp(uint8_t thermoCouple ){ // function
to read the temperature of the selected thermoelement
returns a double with the value converted to Degrees C.

                        uint32_t t;
```

```
                        byte result;
                        SPI.begin();
                        SPI.setBitOrder(MSBFIRST) ;


                if(thermoCouple==1){
                        pinMode (A2,OUTPUT);
                // Transfer pin as output
                        digitalWrite(A2,LOW);
                // Sets the active and inactive
thermocupples
                        digitalWrite(A3,HIGH);



                }
                else{
                        pinMode (A3, OUTPUT);
                // Transfer pin as output
                        digitalWrite(A3,LOW);
                // Sets the active and inactive
thermocupples
                        digitalWrite(A2,HIGH);


                }
                        delay(1);
                        // the cupple needs some time
to settle (300Microseconds)

                         t = SPI.transfer(0x00);
                                        // Transfers
the first byte of the cuple

                                    for(int i = 0;
i<3;i++){                           // Loops
thuough the rest of the bytes and adds them to a 32bit int

                                        result =
SPI.transfer(0x00);
                                        t = t << 8;
                                        t = t |
result;
                                    }
```

```
                                        t >>=18;

        // Shifts the 32 bit int to the external
temperature bits (the 14 MSBs)
                                        double temperature =
((t & 0x1FFF)*0.0625);     // Turns the 13-Bit  integer
into a temperature were each bit is worth 0.25C

                                        if(t&0x2000)
temperature *=- 1;                              // If the
sign bit is 1 turns the temperature negative

        digitalWrite(A2,HIGH);
        // Stops communication with thermocuple

        digitalWrite(A3,HIGH);
    return temperature;

}
double RWino::readInternalTemp(uint8_t thermoCouple ){
                  // Function to get internal temperature
of the desired thermocuple

                            uint32_t t;
                            byte result;
                            SPI.begin();

                            // Initialise the SPI hardware
protocoll
                            SPI.setBitOrder(MSBFIRST) ;

        // Transfer data with the most significant bit
first



                  if(thermoCouple==1){
                            pinMode (A2,OUTPUT);
                  // Transfer pin as output
                            digitalWrite(A2,LOW);
                  // Sets the active and inactive
thermocupples
                            digitalWrite(A3,HIGH);



                  }
```

```
                    else{
                              pinMode (A3,OUTPUT);
                    // Transfer pin as output
                              digitalWrite(A3,LOW);
                    // Sets the active and inactive
thermocupples
                              digitalWrite(A2,HIGH);


                              delay(1);
                              // the cupple needs some time
to settle (300Microseconds)
                    }

                               t = SPI.transfer(0x00);
          // Transfers the first byte of the cuple

                                        for(int i = 0;
i<3;i++){// Loops thuough the rest of the bytes and adds
then to a 32bit int

                                                  result =
SPI.transfer(0x00);
                                                  t = t << 8;
                                                  t = t |
result;
                                        }

                                        t >>=4;

          //Removes the bottom 4 bits

                                        double temperature =
((t & 0x7FF)*0.0625);      // Takes the relevant 11 bits
and turns them into the correct temperature
                                        if(t&0x800)
          temperature *=- 1;                              //
If the sign bit is 1 turns the temperature negative

          digitalWrite(A2,HIGH);
          // Stops communication with thermocuple

          digitalWrite(A3,HIGH);
     return temperature;
}
```

```
void RWino::logValue(double value,char name[ ]){
                  // Prints the Double value to a file on
the SD-Card with the desired name on the file as input

         File myFile;
         pinMode(10,OUTPUT);

         myFile = SD.open(name, FILE_WRITE);
         myFile.print(value);
         myFile.println();
    myFile.close();

}

uint16_t RWino::readFromADC(uint8_t channel,uint8_t
resolution ){ // Function to call to get the current value
on the adc, with resolution and channel as parameter
(options are 12,14,16 and 1,or 2);

         byte config = 0x10;

                  // Sets the configuration with one-shot
mode (change to 0 if you want continuous)

         if(channel == 2){

                  // Set the register to the second
channel on the adc if 2 is selected as imput (1 else)
         config+=0x20;
}

         switch ((int)(resolution)) {

         // Sets the value of the configuration register
depending on what resolution is selected

         case 14:
                  config += 0x04;
         break;

         case 16:
                  config +=0x08;
         break;

  }
```

```
        Wire.beginTransmission(0x6C);

        // Start communication with ADC to set
configuration register
        Wire.write(config);
    Wire.endTransmission();

        Wire.requestFrom(0x6C, 3);

        // Requests 3 data bytes from the adc (3rd byte
is to check the registers on the device)
        uint16_t value =0;
        byte k;

        for(int i =0 ; i<3 ; i++){
                k = Wire.read();

                if(i<2){ value <<=8;
                value |= k;
                }
        }

        if(k!=config){

                // Control to make sure imput register
is the same as the devices's
                return NULL;
        }

                else{
                        return value;
                }

}

double RWino::readMilliVolt(uint8_t channel,uint8_t
resolution){      // Function for reading voltage 0-10V
from the ADC

        uint16_t volt = readFromADC(channel,resolution);
        double result;

        if(volt !=NULL){

                switch ((int)(resolution)) {
```

```
                case 12:
                        volt <<=4;
                break;

                case 14:
                        volt <<=2;
                break;

                }
          result = (volt*0.0625*5);          // Converts
the integer to a milli volt value 1:5 voltage divider
 }

        if(volt & 0x8000){                              //
Check sign bit
                volt =~ volt;
                result = (volt*0.0625*5);
                result *=-1;
        }
        return result;
        }


double RWino::readMicroAmp(uint8_t channel,uint8_t
resolution){  // Fuction to read Micro Ampares from the ADC

        uint16_t amp = readFromADC(channel,resolution);
                                                //Gets the
value from the ADC to use for conversion
        double result;

        if(amp !=NULL){

                switch ((int)(resolution)) {

        // Switch to determin wich resolution is beeing
used

                case 12:
                        amp <<=4;
                break;

                case 14:
                        amp <<=2;
                break;
                }
```

```
          result = (amp*0.0625*10000);

         // Converts the integer to a value in milli Amps
 }

         if(amp & 0x8000){

                  // Checks the sign bit and converts the
reading to a negative output if True

                  amp =~ amp;
                  result = (amp*0.0625*10000);


                  result *=-1;
         }
         return result;


         }
bool RWino::readInput(uint8_t pin){
                  // Function to read input from
optocupples, returns true if pin is high, and false else.


         switch ((int)(pin)) {

                  case 1:

                           if(digitalRead(2)==LOW) return
true;
                           else return false;

                  case 2:

                           if(digitalRead(4)==LOW) return
true;
                           else return false;

                  case 3:

                           if(digitalRead(7)==LOW) return
true;
                           else return false;

                  case 4:
```

```
                        if(digitalRead(A0)==LOW) return
true;
                        else return false;

                }

}
void RWino::setOutput(uint8_t pin,bool state){
          //Function to set output pins high or low
depending on input

         switch ((int)(pin)) {

                  case 1:

                           if(state)digitalWrite(6,HIGH);
                           else digitalWrite(6,LOW);

                           break;

                  case 2:

                           if(state)digitalWrite(9,HIGH);
                           else digitalWrite(9,LOW);

                           break;

                  case 3:

                           if(state)digitalWrite(8,HIGH);
                           else digitalWrite(8,LOW);

                           break;

                  case 4:

                           if(state)digitalWrite(A1,HIGH);
                           else digitalWrite(A1,LOW);

                           break;

                  }

}
```