HET NEUROPLATFORM

DATUM AFSTUDEREN	: 12 NOVEMBER 2012 – 8 MAART 2013
STUDIE	: ELEKTROTECHNIEK
OPLEIDING	: HAAGSE HOGESCHOOL
AFSTUDEERCOACH	: HR. BROEDERS
STUDENT	: JOERI TOM WILLEMSE
LETTERTYPE	: LEXIA I.V.M LEESBAARHEID DISLECTIE
OPDRACHTGEVER	: TU-DELFT
BEGELEIDERS	: WOUTER A.SERDIJN & MARIJN VAN DONGEN

Voorwoord

Op een vakantie in Oostenrijk ben ik in aanraking gekomen met meridianen-therapie dankzij Irene. Door deze therapie is mijn moeder op wonderbaarlijke wijze hersteld van klachten, waarvan ik me het begin nauwelijks kan herinneren. Door deze therapie ben ik geïnteresseerd geraakt in het menselijk lichaam en de werking van het zenuwstelsel.

Via Irene ben ik verder in contact gekomen met neurostimulatie en de verbluffende resultaten hiervan. Daarom wil ik Irene hartelijk bedanken, dat ze mij in contact heeft gebracht met neurostimulatie.

Tijdens mijn stage op de TU-Delft heb ik veel kennis opgedaan dankzij Marijn van Dongen en Wouter Serdijn. Zij hebben mij begeleid in de ontwikkeling van het neuroplatform. Daarnaast wil ik de hele Sectie Electronics bedanken voor de gezellige sfeer en omdat ik altijd bij iedereen terecht kon.

Ik wil meneer Broeders bedanken voor alle adviezen die hij tijdens mijn stage heeft gegeven, het richting geven tijdens mijn stage en alle opbouwende kritiek op mijn verslag.

Ik wens iedereen veel leesplezier en hoop dat dat de neurostimulator nog veel zal bijdrage aan een vruchtbare ontwikkeling in de neurostimulatie.

Samenvatting

Binnen de Biomedical Elektronics Groep van de TU-Delft wordt gewerkt aan nieuwe front-ends voor neurostimulatie/neurosensing. Met neurostimulatie (NS) worden elektrische signalen naar specifieke delen van de hersenen gezonden, om ziektes als Parkinson te onderdrukken. Met neurosensing (NM) worden de hersensignalen gemeten, zodat er bekend wordt wat het effect van neurostimulatie precies is en om te bepalen wanneer er actieve neurostimulatie nodig is.

Helaas zijn deze twee front-ends niet goed te combineren en dat is de belangrijkste reden voor het ontwikkelen van een nieuw platform (neuroplatform). Het neuroplatform kan ook gebruikt worden voor het testen van front-ends en voor het gebruiksgemak van artsen, een niet onbelangrijk onderdeel, omdat er veel projecten gestrand zijn doordat artsen niet met de techniek konden omgaan.

Op het neuroplatform moet het mogelijk zijn om een willekeurige wave-form te genereren, er moeten spanningen gemeten worden, digitale communicatie met instelbare spanningen aanwezig zijn en daarnaast moeten er twee externe interrupts zijn voor het synchroniseren met geluid en beeld, waarbij de gewenste actie binnen 1ms uitgevoerd wordt.

Bij het kiezen van het platform is gekozen voor een basisplatform(T-Minus). T-Minus bevat een microcontroller (ATMega2560) die geprogrammeerd is met AVR Studio 4, vanwege betere simulatiemogelijkheden en snelheid. Voor de willekeurige wave-form is gebruik gemaakt van een Digitaal-Analoog Converter (DAC) en een eerste-orde filter, de DAC was al aanwezig op het platform. Daarnaast kwam de DAC als beste uit het onderzoek. Er zijn 5 verstelbare voedingen aanwezig, twee tussen 0 en 5V, de andere drie moeten ingesteld worden tussen 1 en 4V omdat deze aangesloten zijn op de bi-directionele levelconverter. De bi-directionele levelconverter wordt gebruikt voor vier groepen van vier I/O, waarbij de spanningen bij de front-ends instelbaar zijn. Helaas is er nog wel kortsluiting mogelijk. In de software is alle code binnen een interrupt zo kort mogelijk gehouden, wat ten koste gaat van de gebruiksvriendelijkheid. Bij de functies buiten de interrupts is veel meer rekening gehouden met de gebruiksvriendelijkheid, de naam van die functies is daarnaast ook gebaseerd op functies binnen Arduino.

Aan het eind van deze stageperiode is er een component opgeleverd dat in staat is om analoge spanningen te meten met een ADC , Analoge signalen te creëren met de DAC, digitaal te communiceren via 4x4 instelbare I/O, via externe interrupts binnen 1ms te synchroniseren met bijvoorbeeld een geluidssignaal. Het laagdoorlaatfilter kan nog verder uitgewerkt worden. De externe klok, stroombeveiliging ,de microSD moeten nog geïmplementeerd worden. Daarnaast zijn er nog enkele functies bij de DAC en digitale I/O die verder uitgewerkt kunnen worden.

Inhoudsopgave

1 Inleiding

In de Biomedical Electrinics Group aan de TU wordt gewerkt aan neurostimulatie/neurosensing. Voor de neurostimulatie (NS) worden front-ends gemaakt die via een elektrode, elektrische signalen naar specifieke delen van de hersenen zenden, waarmee ziekte's als Parkinson onderdrukt worden. Met neurosensing (NM) worden de hersensignalen gemeten, zodat er bekend wordt wat het effect van neurostimulatie precies is en om te bepalen wanneer er actieve neurostimulatie nodig is. Tijdens het ontwerp van deze front-ends wordt een hoge prioriteit gegeven aan het energieverbruik en het formaat, dit zijn erg belangrijke onderdelen wanneer deze geïmplanteerd worden.

Helaas zijn deze twee front-ends niet goed te combineren en dat is de belangrijkste reden voor het ontwikkelen van een nieuw platform (neuroplatform). Het neuroplatform kan ook gebruikt worden voor het testen van front-ends en voor het gebruiksgemak van artsen, een niet onbelangrijk onderdeel, omdat er veel projecten gestrand zijn doordat artsen niet met de techniek konden omgaan.

Dit verslag is het naslagwerk van het neuroplatform, prototype 1.0. In hoofdstuk 1, 2 en 3 wordt de projectopdracht en de werkwijze beschreven. In hoofdstuk 4 wordt verder ingegaan op de keuzes en het onderzoek die tot de hardware van het neuroplatform hebben geleid. In hoofdstuk 5 worden de keuzes en de geïmplementeerde functies in de software verder uitgewerkt. In hoofdstuk 9 wordt het platform uitgetest. In hoofdstuk 7 worden de behaalde resultaten en conclusies kort en bondig samengevat. In hoofdstuk 8 worden alle aanbevelingen op een rijtje gezet. In hoofdstuk 10 zijn de bijlage te vinden van de componenten, het geïmplementeerde programma en de onderzochte werking van Eagle.

2 Projectopdracht

Op de TU-Delft wordt gewerkt aan een nieuwe methode om ziektes te bestrijden. De methode is volledig gericht op het manipuleren van de hersenen, dit heeft al een positief resultaat opgeleverd. Er is een apparaatje ontwikkeld(de neurostimulator) dat met de hulp van elektrische signalen delen van de hersenen "doof kan maken" voor bepaalde symptomen, dit wordt al gedaan bij de ziekte van Parkinson, Tinnitus en epilepsie. De onderzoekers in dit project zijn ervan overtuigd dat deze techniek de mogelijkheid bied om nog vele andere ziektebeelden te genezen zoals Gilles-de-la-Tourette, autisme en dystonie.

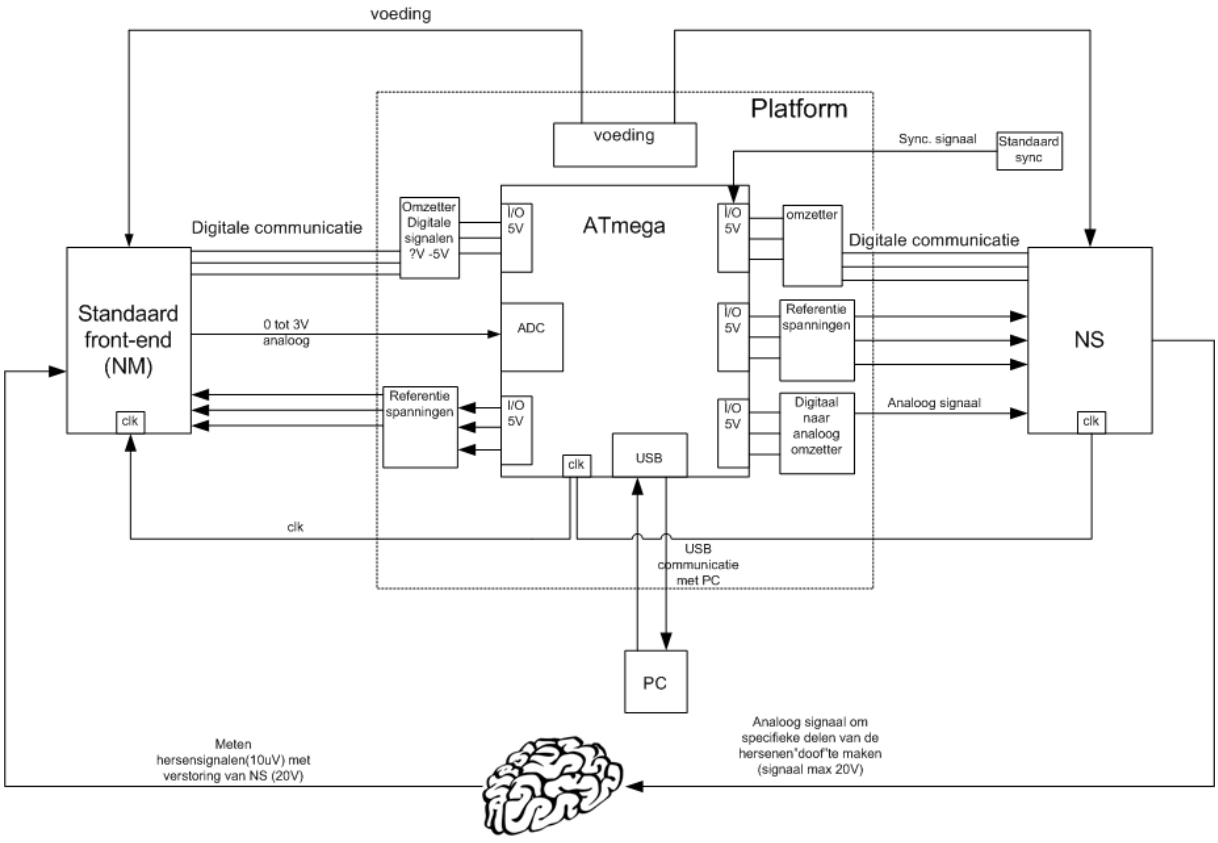
2.1 Probleemstelling

De onderzoeksgroep is bezig met het realiseren van een techniek(neurometer) om elektrische pulsen in de hersenen te detecteren tijdens het “doof maken” van de gewenste hersendelen. Het detecteren van elektrische signalen heeft hierbij als doel, de neurostimulator te laten anticiperen op de aanwezige hersensignalen. Helaas kan de neurostimulator, de gemeten hersengolven niet interpreteren en hier op anticiperen.

Daarnaast zijn de neurostimulator en neurometer niet eenvoudig te gebruiken voor de artsen, die er uiteindelijk mee moeten werken. Wanneer er geen eenvoudige interface voor de artsen gemaakt wordt, kunnen artsen geen gebruik maken van deze techniek en wordt er geen bijdrage geleverd aan de gezondheidszorg.

2.2 Beschrijving van de opdracht

Om de neurostimulator effectief in te zetten, moet er een platform komen, dat de gedetecteerde meetsignalen daadwerkelijk om kan zetten in een ander patroon van elektrische signalen(zie figuur 1). Er moet een eenvoudige gebruikersinterface gemaakt worden zodat artsen goed kunnen werken met de neurostimulator(NS), dit betekent waarschijnlijk een start/stop knop. De wetenschappers hebben een interface nodig om te bepalen bij welke signalen van de neurostimulator, welke actie uitgevoerd moet worden door de neurostimulator(NS). De interface voor de wetenschappers zal bestaan uit C code en eenvoudige functies. De artsen hebben standaard sync signalen nodig om te kunnen zien wat het effect van beeld en geluid is op de hersenen(standaard sync).



Figuur 1: schematisch overzicht "platform neurostimulatie"

2.3 Programma van eisen

Eisen van het platform

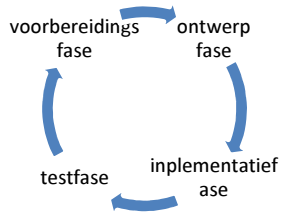
- Er zal een aansluiting voor een standaard front-end gemaakt worden met:
  - Een x aantal I/O voor digitale communicatie waarbij een omzetter nodig is om de spanningslevels compatible te maken(input &output).
  - Een aansluiting op de ADC van de microcontroller(input).
  - Een x aantal niveaus van referentiespanningen aangestuurd door de microcontroller(output).
- Voor de neuronenstimulator zal er nog een extra analoog golfvormig signaal gemaakt moeten worden die aangestuurd kan worden door de microcontroller.
- Er moeten minimaal twee sync. signalen zijn, deze moeten binnen 1 ms synchroniseren (simpele '0'of '1'), zie beschrijving van de opdracht.
- Er moet een I/O beschikbaar zijn voor start en stop knop voor start en stop met een knop.
- De gebruikte I/O's moeten eenvoudig aangeroepen kunnen worden door de programmeur/onderzoeker, met de hulp van functies.
- Het systeem moet 2 dagen achter elkaar actief kunnen zijn, zonder dat dit tot interne onvoorziene situaties leid.

Eisen documentatie

- Er moet een document komen, waarin op een eenvoudige wijze beschreven moet worden wat het platform kan en hoe het werk.
- Er wordt een scriptie gemaakt over "platform neurostimulatie"

3 Werkwijze

In het plan van aanpak is een werkwijze beschreven waarbij elk onderdeel is aangepakt zoals in Figuur 2: module opbouw project omschreven. Over het algemeen is elk onderdeel zoals de software van de externe interrupt, digitale I/O en ADC volgens Figuur 2: module opbouw project uitgewerkt. Soms is er echter van dit model afgeweken door onderdelen die nog niet bekend waren, de hardware die in een keer is geïmplementeerd, en dus niet



Figuur 2: module opbouw project

1. Voorbereidingsfase: onderzoek mogelijkheden en benodigde onderdelen.
2. Ontwerpfase: werk onderzoek uit, kies de beste optie en maak een ruwe schets, hoe dit geïmplementeerd wordt.
3. Implementatiefase: werk de ontwerpfase uit en voer het uit.
4. Testfase: test of het onderdeel aan de voorgeschreven eisen voldoet

Wanneer het onderdeel niet voldoet zal deze syclus herhaald worden met het uitgevoerde werk als input.

4 hardware

In dit hoofdstuk wordt onderzocht hoe de hardware van het platform wordt vormgegeven, welke onderdelen zijn precies nodig en wat zijn de beste mogelijkheden.

Alle componenten zijn via Farnell besteld. Voor het bepalen van de gewenste componenten is gefilterd op functie, beschikbaarheid en prijs. Voor bijna alle componenten wordt SMD gebruikt. Dit zijn componenten die min of meer op de printplaat geplakt worden. Hierdoor kunnen op beide zijden componenten geplaatst worden alhoewel er maar een zijde met de hulp van de oven gesoldeerd kan worden, onderdelen aan de andere kant moeten met de hand gesoldeerd worden. De belangrijkste reden voor het gebruik van SMD is dat er minder ruimte nodig is. SMD componenten zijn kleiner en een gat door de printplaat kost veel ruimte.

4.1 Het platform

Er zijn twee mogelijkheden die als basis kunnen dienden voor het ontwikkelplatform voor neurosimulatie (neuroplatform).

Er kan gebruik gemaakt worden van een FPGA. Een groot voordeel bij de FPGA is de mogelijkheid om aparte modules te ontwerpen die hergebruikt kunnen worden, er is nauwelijks externe hardware vereist omdat het mogelijk is alle functies binnen de FPGA te creëren. Een nadeel is dat het ontwerpen van deze modules veel tijd in beslag neemt en de FPGA veel energie verbruikt. Een ander nadeel is dat er geen of nauwelijks kennis beschikbaar is binnen de Sectie, waardoor er weinig ondersteuning mogelijk is.

Er kan ook gebruik gemaakt worden van een basisplatform met microcontroller. Binnen de Sectie van de TU-delft is een basisplatform ontwikkeld met microcontroller (T-minus). De belangrijkste redenen om gebruik te maken van de T-minus is dat de kennis voor ondersteuning aanwezig is. Omdat de T-minus gebruik maakt van een microcontroller is het energieverbruik lager dan bij een FPGA. Daarnaast is het platform compact en makkelijk aan te sluiten door middel van drie connectoren. Om deze redenen is gekozen gebruik te maken van de T-Minus. In figuur 1 zijn alle I/O pinnen te zien. De namen zijn van de pinnen zijn terug te vinden in de datasheet van de ATMega2560.

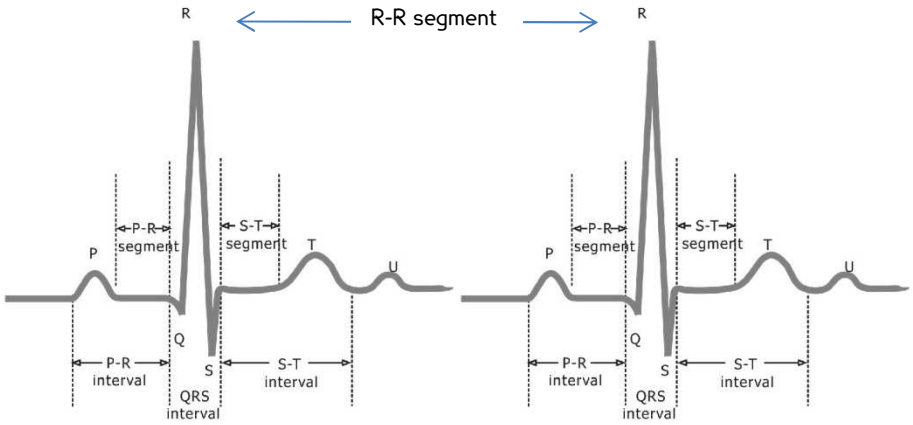
		5V	PF1	PF3	PK1	PK3	PK5	PK7	DAC1	PE3	GND
		GND	PF0	PF2	PK0	PK2	PK4	PK6	DAC0	PE2	5V
GND	5V										
PJ2	PJ1										
PJ0	PA5										
PA3	PA2										
PA0	PA1										
PB5	PA6				T-MINUS ENGINEERING PLATFORM met ATMega2560						
PB7	PA7										
PB0	PB1										
PB2	PB3										
5V	GND										
		5V	PH1	PH3	PH5	PH7	PL1	PL3	PL5	PL7	GND
		GND	PH0	PH2	PH4	PH6	PL0	PL2	PL4	PL6	5V

Figuur 3: T-minus platform met benaming I/O

4.2 Analoge aansturing neurostimulator

Een van de doelen van het neuroplatform is het uittesten van alle mogelijke signalen op het menselijk weefsel. Hiervoor moet het mogelijk zijn om zowel EEG-signalen als ECG signalen te genereren op het neuroplatform.

EEG-signalen bestaan uit 6 hersengolven: Delta, Theta, Alfa, Beta en Gamma. Gamma-golven hebben met maximaal 80Hz de hoogste frequentie. Het ECG is een signaal van het hart, dit signaal heeft een frequentie (RR, zie figuur3) van 10 tot ongeveer 25Hz. Echter het de hoogste frequentie binnen dit signaal is hoger dan de hoogste frequentie binnen de EEG-signalen. Om deze reden worden de EEG-signalen verder buiten beschouwing gelaten, wanneer het mogelijk is om een ECG-signaal te creëren, is het automatisch ook mogelijk om alle EEG-signalen te creëren. Wanneer er componenten met hogere frequenties ontstaan moeten deze weg gefilterd kunnen worden omdat er niet bekend of en hoe het front-end deze kan verwerken



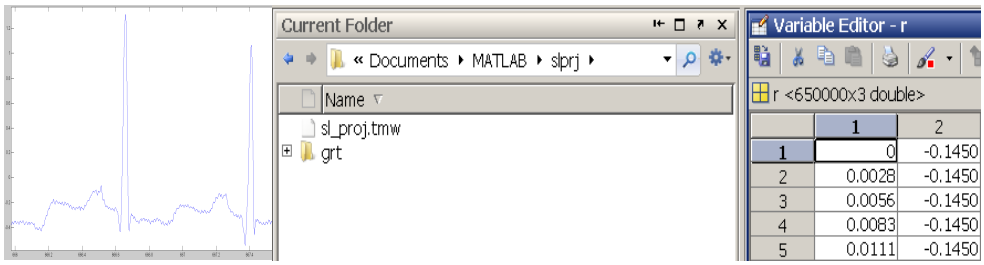
Figuur 4: ECG signaal

Omdat er onduidelijk is uit hoeveel samples het ECG-signaal moet bestaan is er gebruik gemaakt van een ECG-meting op de PhysioNet-webserver.

Om het ECG-signaal op te halen is gebruik gemaakt van het WFDB Software pakket.

Wanneer het WBFDB pakket geïnstalleerd is kan er gebruik gemaakt worden van de functie "rdsamp ('mitdb/101','sigs',1)" binnen MATLAB om een ECG-meting op te halen uit het PhysioNet register. In het ECG-signaal (zie Figuur 5) is te zien dat de eerste rij van variable r oploopt met 0,0028s, dit is de sample-rate. In overleg met de opdrachtgever is gekozen om deze meting verder te gebruiken als maatstaf voor het opwekken van het ECG-signaal.

In het ECG-signaal bleek er gesampled te zijn met ongeveer 357Hz. De sample-rate moet altijd minimaal twee keer zo groot zijn dan de hoogste frequentie van het gewenste signaal, omdat er anders informatie verloren kan gaan. Dit betekent dat het signaal maximaal een frequentie kan hebben van 197Hz.



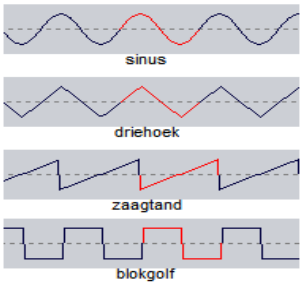
Figuur 5: geïmporteerd ECG signaal MATLAB

Voor de discrete aansturing kan op basis van de bovenstaande gegevens worden vastgesteld dat het gewenste signaal gegenereerd moet zijn met een sample-rate die minimaal een bereik heeft van 0 tot 357Hz. Dit betekent dat het mogelijk is om elk signaal te genereren dat binnen het bereik van 0 tot 197Hz valt. Naast de frequentie moet de spanning kunnen variëren tussen 0 en 5 volt. Dit is een eis die van tevoren door de opdrachtgever is gegeven.

Voor de aansturing van dit signaal zijn 3 verschillende mogelijkheden onderzocht: een frequentie-synthesizer, Puls Width Modulation (PWM), Digital-Analog Converter (DAC).

4.2.1 Frequentie-synthesizer

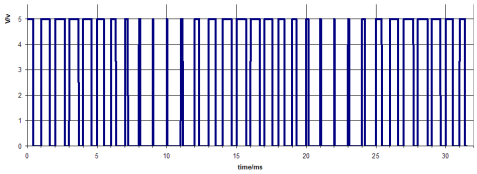
Frequentie-synthesizers zijn IC's die tot hoge frequenties in staat zijn en alleen aangestuurd hoeven te worden wanneer de frequentie of vorm van het signaal aangepast moet worden. Een nadeel van het functiegenerator-IC is dat deze alleen in staat is om sinus-, blok-, driehoek- of zaagtand-golven uit te sturen. Doordat er geen mogelijkheid is om elke willekeurige vorm te maken is een functiegenerator niet de gewenste oplossing.



Figuur 6: frequentie-synthesizer

4.2.2 Puls-Widthe Modulation (PWM)

Met de PWM van de microcontroller is elk signaal te maken. De frequentie is voldoende, maar het signaal heeft nog totaal niet de gewenste vorm. PWM creëert alleen maar pulsen met een aan te passen breedte, precies zoals de naam weergeeft (zie Figuur 7) en geen spanningsvariatie. Voor het opwekken van een analoog signaal is extra filtering vereist. De noodzaak voor de extra filtering maakt van PWM niet de gewenste oplossing.



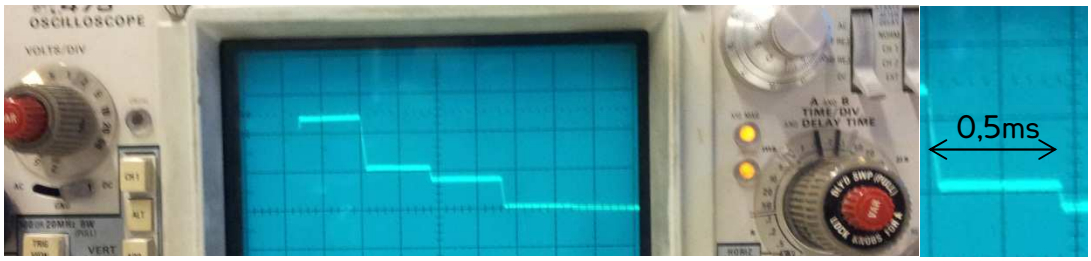
Figuur 7: Puls-Whide Modulation

4.2.3 Digitaal-Analoog Converter (DAC)

De DAC (MCP4725) is een IC, waarmee elke willekeurige vorm te maken is. Het nadeel van de DAC is dat elk stapje real-time aangestuurd moet worden door de microcontroller. Dit kost een hoop CPU-tijd en de maximaal haalbare frequentie is veel lager dan bij het functiegenerator-IC. De DAC zal altijd een discreet signaal uitsturen. Een groot voordeel is dat dit IC al op de print aanwezig is. De sample-rate tussen de verschillende spanning-niveaus wordt gelimiteerd door de microcontroller maar is nog steeds ruim voldoende.

De DAC is in staat om binnen 6µs naar elke willekeurige spanning te schakelen, heeft een High-Speed mode waarbij communicatiesnelheden mogelijk zijn tot en met 3,4Mbps en werkt tussen 0 en 5V met een resolutie van 0 tot 4095 (12 bits). De communicatie tussen de microcontroller en de DAC wordt uitgevoerd via I^2C.

Er is ook een meting gedaan om vast te stellen of de sample-rate van de DAC ook in praktijk hoog genoeg is.



De DAC bleek tijdens de metingen een maximale sample-rate te genereren van 20kHz.

4.2.4 conclusie

Van beide IC's bleek de DAC het meest geschikt omdat de snelheid voldoende is en het verlies aan capaciteit ruimschoots opweegt tegen het maken van elke willekeurige stabiele golfvorm, die relatief eenvoudig in te voeren is. De discrete vormen kunnen gladgestreken worden met een laagdoorlaatfilter.

	PWM	DAC(MCP4725)	Functiegenerator
Al aanwezig	ja	ja	Nee
Neemt weinig capaciteit van de microcontroller in	ja	nee	ja
Is in staat tot alle gewenste vormen	ja	ja	nee
vloeiend signaal	slecht	redelijk	goed
Is in staat tot een hoge frequentie	redelijk	redelijk	goed
Signaal is direct te gebruiken voor neurostimulator	slecht	redelijk	goed

Tabel 1: vergelijking PWM, DAC en functiegenerator

4.3 Filtering

Omdat er gekozen is om het gewenste signaal met samples op te bouwen ontstaan er componenten met hogere frequenties. Het is niet op voorhand bekend of deze hogere frequenties automatisch weg gefilterd worden door de front-ends of deze hoge frequenties voor bijwerkingen zorgen in de hersenen. Om er zeker van te zijn dat de hogere frequenties geen problemen veroorzaken, wordt er ook gebruik gemaakt van een laagdoorlaatfilter (reconstructiefilter).

Om dit filter te realiseren is gekozen voor een simpel RC-filter. De waardes van de weerstand en condensator zijn bepaald met de hulp van een simulatie (zie Figuur 8). Voor het signaal is gebruik gemaakt van de metingen in MATLAB (zie hoofdstuk 4.2). De metingen zijn gedaan met de hulp van Multisim. Tijdens het opbouwen van de simulatie bleek dat de meest geschikte signaalgenerator een continu signaal genereerde van de samples, in plaats van een discreet signaal. Om toch een discreet signaal te genereren is er een for-lus in MATLAB geprogrammeerd die tussen elke sample een extra sample creëert, die wat tijd betreft bijna gelijk is aan de volgende sample en wat spanning betreft, gelijk is aan de vorige sample (zie Tabel 2).

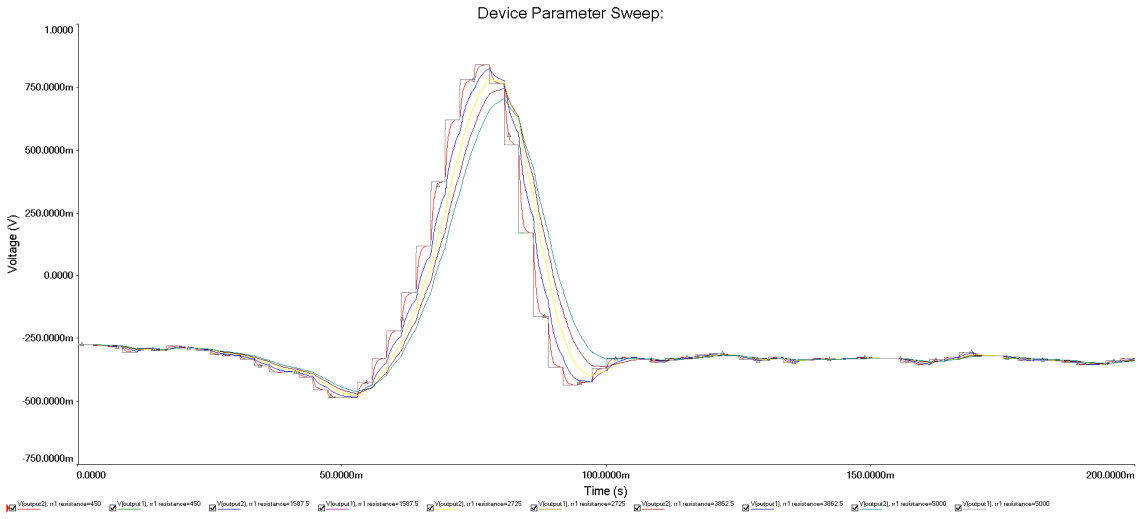
For s=1:s+2;101 d=d+1 index(s,1)=r(d,1) index(s,2)=r(d,2) index(s+1,1)=r(d+1,1)-0,0001	s en d worden gebruikt voor de telling in de loop, index is de file waar het bewerkte ECG signaal in opgeslagen wordt en r is de file waar het originele signaal opgeslagen is. Hetgeen binnen haakjes bepaalt de

index(s+1,2)=r(d,2) end	plek in de sheet van index of r (rij_verticaal, rij_horizontaal)

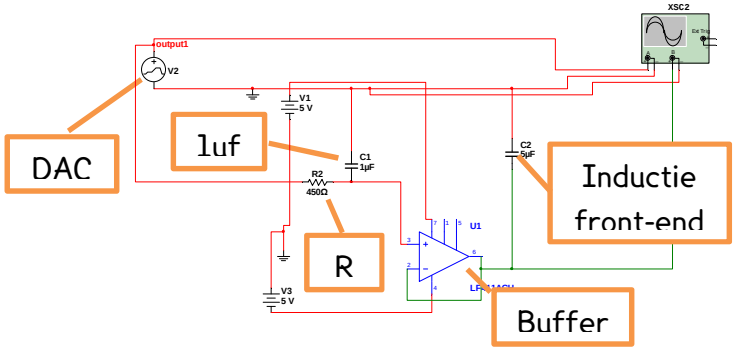
Tabel 2: for-loop in MATLAB voor discreet maken signaal

Door de extra samples wordt er een signaal gegenereerd van 357Hz dat overeenkomt met wat de DAC kan genereren.

In de simulatie van Figuur 8 zijn meerdere signalen te zien. Het tijd-discrete signaal is het orginele signaal dat de DAC produceert en bij de andere signalen is gebruik gemaakt van een laagdoorlaatfilter (zie Figuur 11). Het signaal dat het dichtst bij de discrete vorm ligt heeft een weerstand van 450 ohm. Bij elk volgende signaal is de weerstand met 1137,5 ohm verhoogd. In de simulatie is te zien dat 450 nog niet ten koste gaat van de spanning; de signalen daarna bevatten minder componenten met hogere frequenties, maar dit betekent ook dat er gegevens van het signaal verloren gaan.



Figuur 8: Simulatie ECG signaal in multisim met laag doorlaatfilter



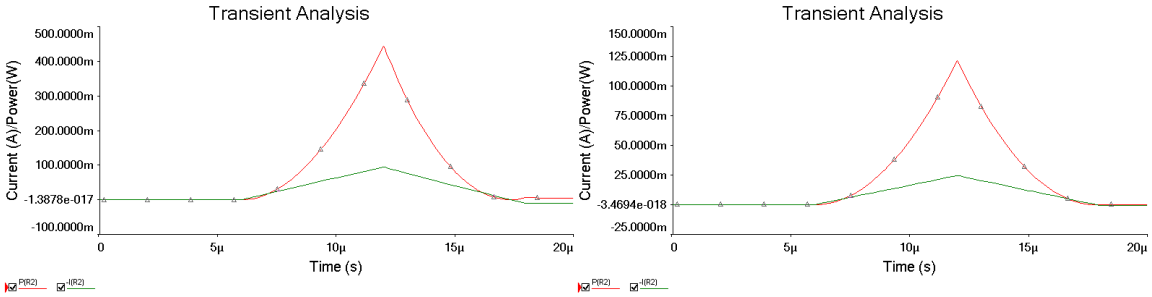
Figuur 9: schakeling met laag doorlaatfilter

Bij de simulatie van de DAC in Figuur 11 is te zien dat er ook gebruik wordt gemaakt van een buffer. Dit wordt gedaan omdat het front-end dat het signaal ontvangt, een ingangscapaciteit heeft. De capaciteit kan zonder buffer, de kantelfrequentie van het laagdoorlaatfilter beïnvloeden.

In de praktijk zal voor R2 een trimmer als instelbare weerstand gebruikt worden. De gebruiker kan de weerstand instellen tussen 50 en 5kΩ.

De keuze voor een trimmer is gemaakt omdat het onbekend is wat voor soort signalen gegenereerd zullen worden en daarmee ook onbekend is welke frequenties weg gefilterd moeten worden voor dat desbetreffende signaal.

Er is ook een simulatie gemaakt van de stroom die door R2 kan lopen (zie Figuur 10). De DAC heeft een settling time van 6µs, van 0 naar 5V en daarna weer 6us van 5 naar 0V.



Figuur 10: simulatie DAC stroom met 50 en 200 ohm

Er is uit een simulatie gebleken dat de stroom hoger werd dan 50mA bij een weerstand lager dan ongeveer 200Ω. De maximale stroom die de buffer van de DAC kan leveren is 50mA, het maximale vermogen van de trimmer is 250mW en daarom is er een weerstand voor de trimmer geplaatst van 150Ω. Dit betekent dat de cutoff frequentie ongeveer tussen 7958Hz en 30,8Hz in te stellen is. Dit is niet ideaal aangezien er een sample-rate van 20kHz met de DAC mogelijk is. Een frequentie van 10kHz kan dus wenselijk zijn. Ondanks deze nadelen is toch gekozen voor 1uf met een trimmer van 5kohm en een extra weerstand van 150ohm, met als enige reden dat deze componenten simpelweg al op voorraad waren. Bij een volgend ontwerp kan er beter gekozen worden voor een 0,1µF condensator en een trimmer van 50kohm, met 50 ohm als laagste waarde.

4.4 Digitale en analoge I/O

Het neuroplatform moet op meerdere manieren kunnen communiceren met de front-ends, omdat er op voorhand niet bekend is welke methode toegepast wordt bij de front-ends. Er moeten mogelijkheden zijn voor analoge en digitale communicatie, met instelbare spanningen voor de digitale I/O. Voor de analoge communicatie is gebruik gemaakt van de analoog naar digitaal converter (ADC) in de microcontroller. Met behulp van een multiplexer kan de ADC op meerdere pinnen aangesloten worden. In dit geval is één pin per front-end voldoende. De ADC-pinnen werken alleen als input, waardoor er geen gevaarlijke situaties kunnen ontstaan en kunnen zonder tussenkomst van een beveiliging aangesloten worden op de front-ends. Er worden 16 pinnen beschikbaar gesteld voor digitale I/O, 8 per front-end. Voor elke meting waarbij dit platform gebruikt wordt kan een andere verdeling van I/O vereist zijn. Daarom is het van belang

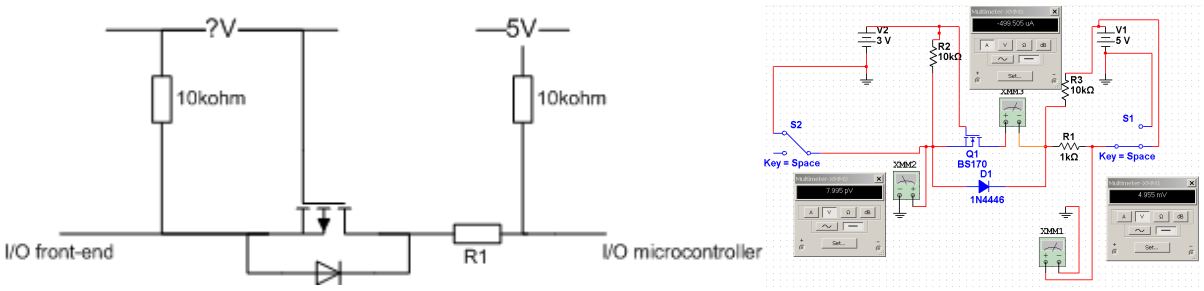
de I/O pinnen zo flexibel mogelijk te maken. Er zijn vier mogelijkheden onderzocht voor het instellen van de digitale I/O.

4.4.1 Bidirectionele level converter met behulp van eigen schakeling

Om te kunnen digitale communicatie mogelijk te maken, moet de I/O van de front-end met een spanning kleiner of gelijk aan 5 volt, de 5V volt van de microcontroller kunnen schakelen en de microcontroller de onbekende spanning van de front-end. Hiervoor is een schakeling bedacht met MOSFET's.

Er kan ook gebruik gemaakt worden van een MOSFET voor bi-directionele level shifting. Hiervoor is maar een select aantal MOSFETs geschikt. Hierbij is het belangrijk dat de treshold lager is dan de laagst gebruikte spanning. In Figuur 11 is een simulatie te zien waarbij de drain en source poort beide als input en output kunnen functioneren, het is zelfs mogelijk om de schakeling te beveiligen tegen kortsluiting door een extra weerstand (R1).

De schakeling zal alleen actief de spanning naar de aarde trekken met de hulp van de MOSFET en de diode. Wanneer de schakeling niet actief naar de nul geschakeld wordt, zullen beide in/uitgangen automatisch weer de gewenste spanning ontvangen via de pull-up. Helaas heeft deze schakeling ook nog een groot nadeel wanneer deze te maken krijgt met capaciteiten. Wanneer er een capaciteit aanwezig is, moet eerst de capaciteit opgeladen worden via de pull-up en dit kan een ongewenste vertraging veroorzaken.



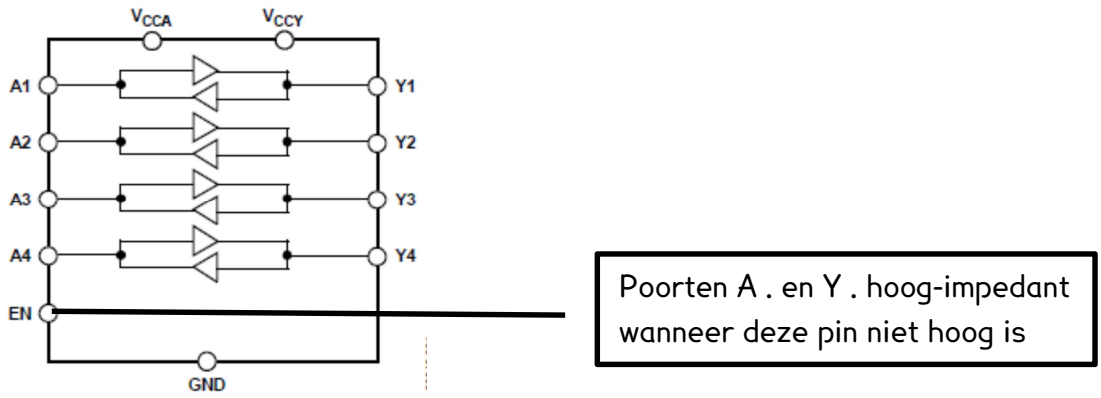
Figuur 11: simulatie eigen schakeling met extra weerstand als kortsluitingsbeveiliging

4.4.2 Elektronische schakelaar

De elektronische schakelaar is een IC dat werkt zoals een relais. Voor deze toepassing is niet gekozen omdat voor elke I/O pin een nieuw IC nodig is. Hierdoor wordt de schakeling veel te groot. Daarnaast is er weer een extra IC nodig voor het schakelen tussen input en output. Deze oplossing bleek ook niet geschikt door het aantal benodigde IC's

4.4.3 Bi-directionele communicatie zonder directie pin (ADG3304BRUZ)

Bij deze chip kan niet bepaald worden in welke richting gecommuniceerd wordt. Er is wel een mogelijkheid om de A en Y poort als hoog-impedant in te stellen (zie Figuur 12, EN). EN kan als beveiliging werken wanneer de te hoge stromen gemeten deze mogelijkheid voegt alleen wat toe als beveiliging ,wanneer er hoge stromen gemeten kunnen worden. Mocht er een goede oplossing worden gevonden voor het meten van de stroom/kortsluiting en een techniek die ervoor zorgt dat input en output van de microcontroller en front-end op elkaar afgestemd zijn, dan is dit de beste oplossing. Zolang er geen oplossing is voor de richting is dit IC niet de beste oplossing voor digitale I/O. Dit IC wordt wel gebruikt voor de communicatie tussen de microSD en de microcontroller via de SPI communicatie. Het SPI protocol voor deze communicatie is zo duidelijk vastgesteld wanneer welke pin als uitgang werkt dat er verder geen problemen kunnen ontstaan, daarnaast kan er niets meer aan dit protocol veranderd worden waardoor menselijk falen geen invloed meer heeft.



Figuur 12: Bidirectionele communicatie zonder directie pin

4.4.4 Bidirectionele communicatie met directie-pin (SN74LVC8T245PW)

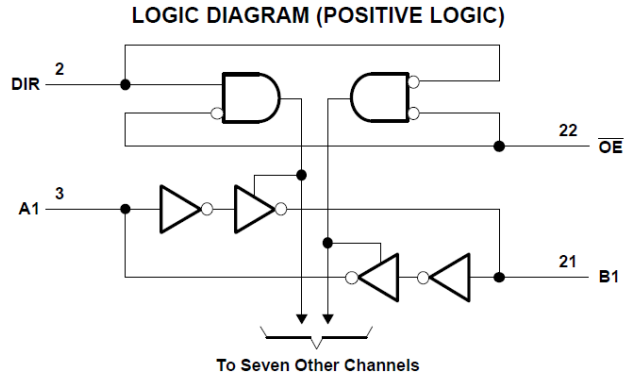
Dit IC heeft één directie-pin (DIR), een enable-pin (OE) en 8 kanalen (zie Figuur 13).

Doordat er duidelijker in te stellen is in welke richting dit IC werkt, is er gekozen voor dit IC.

Ondanks dat de richting bij de microcontroller bekend is betekend dit niet dat de richting automatisch bekend is bij de front-end, er kunnen paar uitgangen gebruikt worden puur voor het aangeven hoe de andere I/O ingesteld zijn of het wordt van tevoren hardware-matig bepaald. Desondanks is kortsluiting en te hoge stromen nog steeds een probleem waar op dit moment geen eenduidige oplossing voor is.

CONTROL INPUTS		OUTPUT CIRCUITS		OPERATION
$\overline{OE}$	DIR	A PORT	B PORT	
L	L	Enabled	Hi-Z	B data to A bus
L	H	Hi-Z	Enabled	A data to B bus
H	X	Hi-Z	Hi-Z	Isolation

(1) Input circuits of the data I/Os are always active.



Figuur 13: Biderecionele communicatie met directie pin

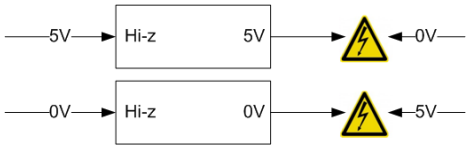
4.4.5 Conclusie

Ondanks dat er nog kortsluiting mogelijk is bij de SN74LVC8T245PW, is hier toch voor gekozen omdat de omdat dit component geen vertraging ondervindt tijdens het schakelen.

Om ervoor te zorgen dat de digitale I/O's nog enigszins flexibel te configureren zijn, is er gebruik gemaakt van vier kanalen per IC, in plaats van de beschikbare acht kanalen. Bij elk IC is de DIR poort aangesloten op een pull-down waardoor de richting van de I/O altijd duidelijk is. Een ander voordeel van een pull-down is dat de poort aan de front-ends in eerste instantie altijd hoog impedant is, waardoor er geen hoge stromen/kortsluiting kan ontstaan. Om de directie om te draaien is er een shunt per directieport (DIR) die een verbinding kan maken tussen 5V en de DIR poort. Hierdoor moet de I/O van de microcontroller als input functioneren. Voor de microcontroller is altijd bekend in welke richting de bi-directionele levelconverter ingesteld staat. Dit komt omdat er op elke DIR poort een input van de microcontroller aangesloten is. Er zijn vier DIR poorten aanwezig met vier DIR inputs. Elke DIR poort stuurt vier digitale I/O aan. Dit betekend dat er in totaal 16 digitale I/O beschikbaar zijn. Wanneer deze is ingesteld als output kan er via deze digitale uitgangen bepaald worden in welke richting de poort ingesteld moet staan, de hardware is hier geschikt voor wanneer er een beveiliging tegen kortsluiting is, in de software moet dit nog geïmplementeerd worden.

4.5 Circuit beveiliging

Tijdens het onderzoek naar de I/O tussen de T-Minus en de front-ends bleek dat er nog steeds een gevaarlijke situatie kan ontstaan wanneer er alleen gebruik gemaakt wordt van de level-converter. Deze situatie kan ontstaan wanneer de I/O pin van de microcontroller als uitgang ingesteld staat en de daaraan gekoppelde I/O van de front-end ook als uitgang ingesteld staat. Zolang beide pinnen hoog staan of beide pinnen laag staan is er nog geen probleem, maar wanneer er een pin hoog staat en een pin laag, ontstaat er een kortsluiting tussen de pinnen zie Figuur 14.



Figuur 14: kortsluiting

Er is onderzocht of de referentiespanning (ADP123AUJZ-R7) en de level-converter (SN74LVC8T245PW) al over ingebouwde beveiligingen tegen kortsluiting beschikken.

In de datasheet van de ADP123AUJZ-R7 bleek een beveiliging te zijn vermeld. Helaas is de stroom waarbij de referentie spanning afschakelt veel te hoog. Daarnaast zouden dan alsnog niet alle combinaties beveiligd zijn want de microcontroller wordt niet gevoed door de referentievoedingen en is daardoor niet beveiligd.

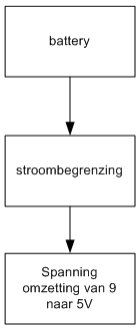
In de datasheet van de SN74LVC8T245PW is een zin gevonden die niet volledig eenduidig is" *The VCC isolation feature ensures that if either VCC input is at GND, all outputs are in the high-impedance state*", Hierdoor is onduidelijk of dit betekent dat wanneer er kortsluiting ontstaat of wanneer de voedingen verkeert zijn aangesloten. Wanneer de zin over kortsluiting gaat is ook nog niet bekend bij welke stroom deze afschakelt en hoe snel.

Er is gebeld naar Texas instruments. Uit het gesprek werd duidelijk dat er sinds kort alleen ondersteuning wordt gegeven aan studenten die studeren voor hun PhD of nog verder zijn.

In de praktijk is echter gebleken dat er in de SN74LVC8T245PW geen geschikte beveiliging aanwezig is.

4.5.1 Stroombegrenzer

Er kan gebruik gemaakt worden van een stroombegrenzer parallel aan een condensator die de stroompieken opvangt. Helaas is een stroombegrenzing niet ideaal omdat deze alleen optimaal werkt bij precies de gewenste stroom, iets lager of hoger en er komt een ongewenste spanning over de stroombegrenzer te staan, dit zou wel kunnen wanneer de stroombegrenzer geplaatst wordt voor de spanning omzetter omdat een beetje verlies aan spanning dan geen probleem hoeft te zijn is (zie Figuur 15).

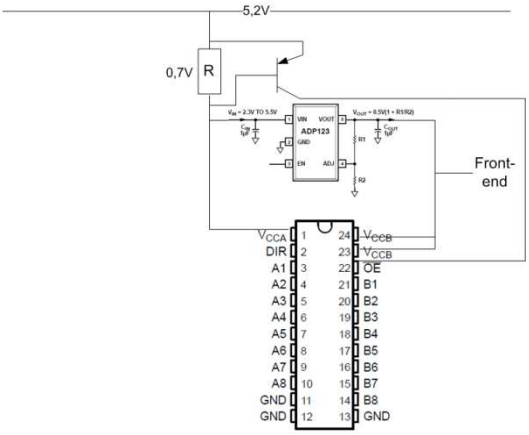


Figuur 15: stroombegrenzing flowchart

4.5.2 Beveiligingen met behulp van de OE-ingang

De level-converter heeft een ingang genaamd OE, zolang de benodigde spanning niet op deze ingang aangeboden wordt zullen alle I/O doorgangen een hoge impedantie hebben. Om de OE-ingang te gebruiken als beveiliging moet de stroom worden gemeten en afhankelijk daarvan de spanning op 0 of 5V geschakeld. De stroom kan met de hulp van een transistor en een weerstand gemeten worden (zie Figuur 16).

Een nadeel is dat de maximale spanning zal afnemen, op dit moment is de spanning in praktijk 5,03V. De voedingsspanning mag iets hoger want de chips zijn bestand tegen een spanning, tot 5,5V. Er zijn ook IC's waarbij de voedingsspanning rond de 5,2 ligt dit kan net het verschil maken aangezien de stroom al veel te hoog is bij een spanning van 0,7V.



Figuur 16: stroombeveiliging met OE ingang en transistor

4.5.3 Weerstanden

Om de stroom te beperken kan ook gebruik gemaakt worden van weerstanden, De maximale spanning kan gedeeld worden door de maximale stroom en de uitkomst is de minimale weerstand.

Bij een gewenste stroom van 40mA en een spanning van 5V zou dit een weerstand van minimaal 125ohm betekenen $\left(R = \frac{U}{I}\right)$.

Een nadeel bij het front-end is dat de spanning kan varieren tussen 1 en 5V. Dit betekend dus dat de stroom onbedoeld veel lager kan zijn bij 1V

4.5.4 beveiligingsIC

Tijdens het onderzoek naar een geschikte kortsluitbeveiliging is er ook een IC gevonden die de gewenste stromen kan beveiligen. Dit IC kan erg interessant zijn bij het uitwerken van een goede stroombeveiliging.

4.5.5 Conclusie

In overleg met de opdrachtgever is uiteindelijk gekozen voor de beveiliging met weerstanden, omdat er niet veel tijd meer was voor het project en de beveiliging misschien al aanwezig was. Helaas een te lage weerstand gekozen (100ohm) waardoor de beveiliging niet voldoet. Achteraf was het beter geweest om gebruik te maken van de mogelijkheid in hoofdstuk 4.5.2.

4.6 Basiscomponenten

In dit hoofdstuk wordt verder ingegaan op de componenten die geen groot vooronderzoek nodig hadden, bij paar van deze componenten zijn wel meerdere mogelijkheden maar de juiste keuze is bij deze componenten relatief eenvoudig te filteren.

4.6.1 LDO

Op het neuroplatform moeten vijf variabele referentiespanningen aanwezig zijn. Deze spanningen moeten elke willekeurige spanning kunnen genereren tussen 1,8 en ~5V. De referentiespanningen zullen nauwelijks belast worden en moeten met de hand ingesteld kunnen worden.

Voor deze toepassing zijn een aantal chips onderzocht: MCP1727, ADP1715, ADP123

Alle drie de chips zijn instelbaar door middel van een trimmer.

Het grootste en belangrijkste verschil is de dro-pout spanning. De ADP123 is gekozen omdat deze met 85mV de laagste drop-out spanning heeft. Dit is erg belangrijk omdat voor deze LDO's de spanning al naar 5V is, dezelfde spanning is ook gewenst als LDO output.

4.6.2 Trimmers (poteniometers)

Voor het afstellen van de referentiespanning (ADP123AUJZ) worden trimmers gebruikt. In de datasheet wordt aangegeven de instellingen met weerstanden groter dan 200kohm onnauwkeurig wordt. De weerstand moet verder zo groot mogelijk zijn opdat er geen hoge stromen ontstaan. De weerstand moet dus zo hoog mogelijk worden zonder dat de referentiespanning onnauwkeurig wordt. Er is daarom gekozen voor een weerstand met een factor 2 lager(100kohm).

Het is erg belangrijk dat de trimmer meerdere turns heeft om nauwkeurig af te stellen. Voor een groot bereik moet de trimmer een lage minimale weerstand hebben (zie H2.3).

Een productie-afwijking van de trimmer is geen probleem omdat de deze toch nog ingesteld moet worden op de gewenste weerstandsdeling, de productie-afwijking is zodanig klein dat de gewenste waarden nog steeds binnen het bereik van de referentiespanning liggen (zie hoofdstuk 6.2).

4.6.3 Condensators

Op het neuroplatform worden condensators gebruikt ter stabilisatie van de voedingen. Hierbij was het soort condensator en de capaciteit terug te vinden in de datasheet van de voeding. Ter voorkoming van een rimpel op de voedingsspanning wordt bij elke actieve component een condensator tussen de voeding en aarde geplaatst.

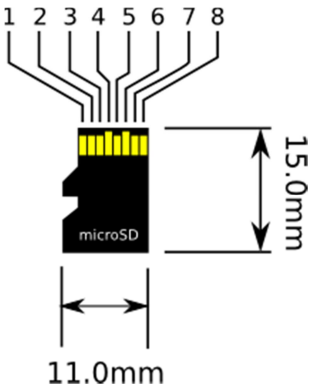
Er worden alleen keramische condensatoren gebruikt. Deze condensators hebben in vergelijking met een tantaalcondensator een kleine interne weerstand, zijn goedkoper en nauwkeuriger. Binnen de condensatoren is gekozen uit X5R/X7R omdat deze condensatoren nauwkeurig genoeg zijn voor de gewenste applicaties, zoals in de datasheet van de twee voedingen (L7805ACD2T-TR, ADP122AUJZ) beschreven.

Minimum temperature[1]		Maximum temperature[1]		Capacitance change permitted[1]	
X	-55 °C	2	+45 °C	**A**	±1.0%
Y	-30 °C	4	+65 °C	**B**	±1.5%
Z	+10 °C	**5**	+85 °C	**C**	±2.2%
		6	+105 °C	**D**	±3.3%
		7	+125 °C	**E**	±4.7%
		8	+150 °C	**F**	±7.5%
		9	+200 °C	**L**	+15% / -40%[2]
				P	±10%
				R	±15%
				S	±22%
				T	+22%/-33%
				U	+22%/-56%
				V	+22%/-82%

Figuur 17: codering keramische condensator

4.6.4 microSD

Het is niet altijd gewenst om een computer te moeten gebruiken. Daarom is het gewenst om een extra methode te hebben voor het invoeren en opslaan van gegevens. Na onderzoek bleek de microSD het ideale medium te zijn omdat deze klein is en al een geïntegreerd communicatieprotocol heeft. Met de hulp van een bidirectionele level converter die 5V omzet naar 3,3V is er alleen nog een houder voor de kaart nodig. Uiteraard moet er wel het nodige aan programmering gedaan worden. Het programmeren en solderen van de houder moet nog worden uitgevoerd.



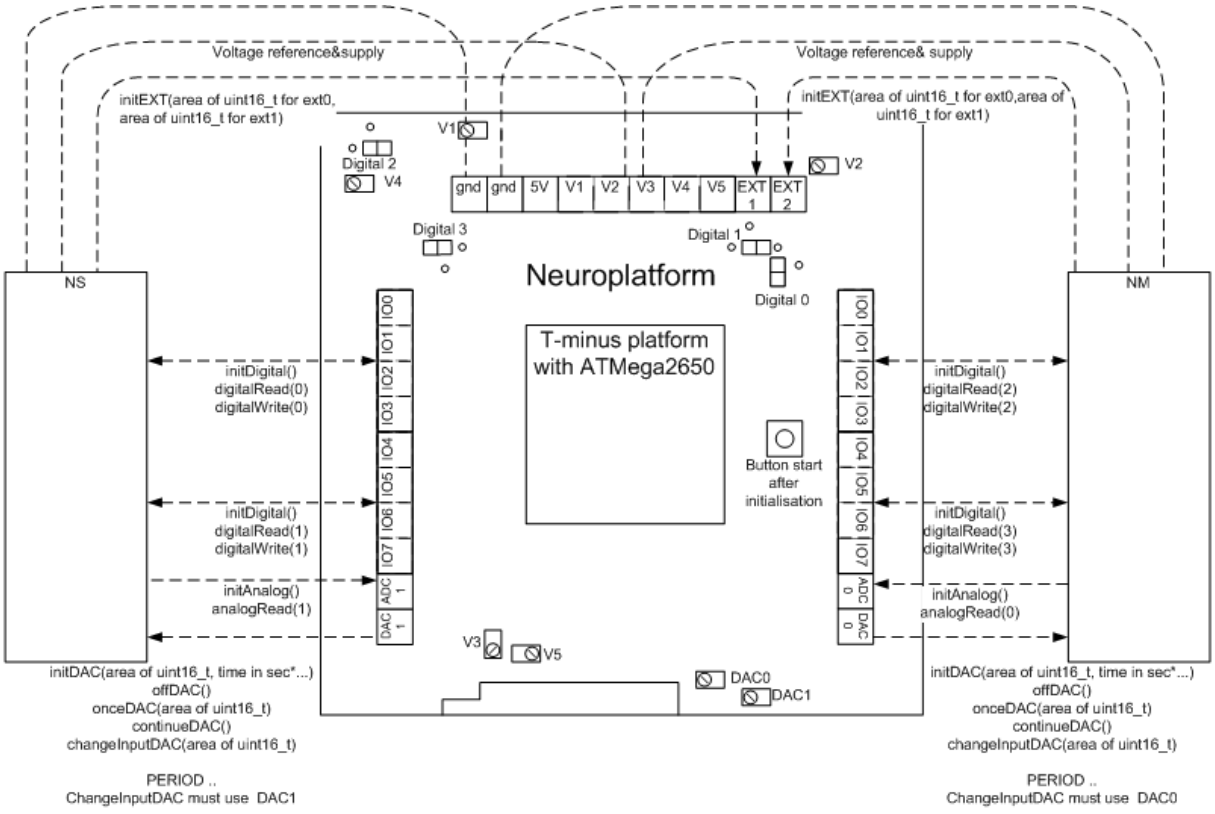
microSD Card						
Pin No.	SD Mode			SPI Mode		
	Name	Type	Description	Name	Type	Description
1	DAT2	I/O/PP	Data Line [Bit 2]	RSV		Reserved
2	CD/DAT3	I/O/PP	Card Detect / Data Line [Bit 3]	CS	I	Chip Select
3	CMD	PP	Command/Response	DI/MOSI	I	Data In/Master Out Slave In
4	Vdd	S	Power	Vdd	S	Power
5	CLK	I	Clock	SCLK	I	Clock
6	Gnd/Vss	S	Ground	Gnd/Vss	S	Ground
7	DAT0	I/O/PP	Data Line [Bit 0]	DO/MISO	O/PP	Data Out/Master In Slave Out
8	DAT1	I/O/PP	Data Line [Bit 1]	RSV		Reserved

5 Software

Voor het neuroplatform is gebruik gemaakt van de T-minus. Dit platform werkt met een microcontroller en moet dus geprogrammeerd worden. Omdat er niet van tevoren precies vastgesteld kan worden hoe de gebruiker de programmering wil, is er gebruik gemaakt van functies. Dit betekent dat de gebruiker zelf wel moet programmeren, maar het is veel eenvoudiger gemaakt en er kan veel gebruik gemaakt worden van voorbeelden.

Aan het begin van het programmeren zijn eerst de prioriteiten bepaald. Omdat de timing erg belangrijk is zullen alle programma's die in een interrupt kunnen komen zo kort mogelijk zijn en zo veel mogelijk onderdelen in initialisaties berekenen. Hierom wordt er veel met pointers gewerkt, aangezien dit de snelste methode is voor het verkrijgen van de gegevens.

In Figuur 18 zijn alle functies te zien en voor welk onderdeel deze gemaakt zijn.



Figuur 18: Overzicht neuroplatform

5.1 AVR-studio VS Arduino

Het basisplatform van de TU-delft (T-Minus) werkt met de microcontroller ATMega2560. De microcontroller kan via een USB-aansluiting geprogrammeerd worden met AVR-studio en Arduino. Arduino herkent de USB poort automatisch maar voor het programmeren met AVR is er nog een extra programma nodig: HappyJTAG2. HappyJTAG2 is momenteel alleen beschikbaar voor AVR-studio4, dit betekent dat er gekozen moet worden tussen AVR-studio4 en Arduino.

Beide software pakketten zijn gratis te downloaden en met beide pakketten wordt er geprogrammeerd in C.

De software-omgeving Arduino is in tegenstelling tot AVR studio erg makkelijk in gebruik doordat:

1. Er is meer informatie op internet beschikbaar voor Arduino dan voor AVR-studio.
2. Arduino heeft een eigen bibliotheek met veel bruikbare voorbeelden. (zie Figuur 20)
3. Het opslaan van programma's in Arduino is erg overzichtelijk doordat Arduino geheugenruimte creëert: Sketchbook (zie Figuur 19).
4. In Arduino zijn functies beschikbaar die het programmeren vereenvoudigen.

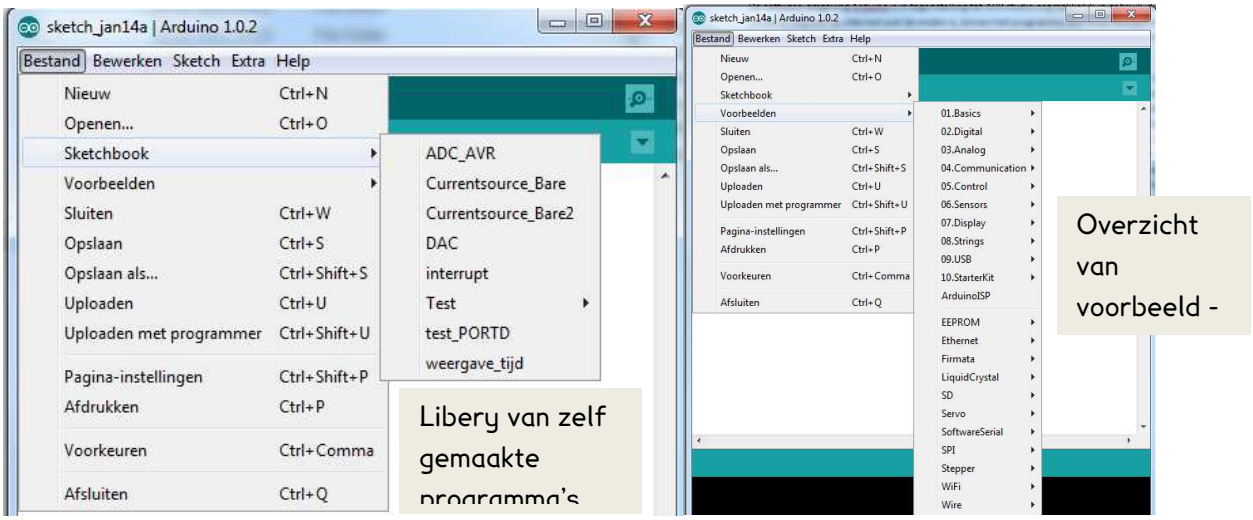
Een groot nadeel is dat er geen eenvoudige toegang is tot de interne libraries en sourcecode van de functies in Arduino. Hierdoor is er geen volledige controle over de microcontroller. Een voorbeeld hiervan is de functie Serial.print(); de werking van deze functie is niet te achterhalen, aangezien er geen toegang is tot de definitie.

Een ander nadeel is dat de functies van Arduino veel meer tijd in beslag nemen dan wanneer deze direct worden ingesteld via de registers. De registers zijn niet eenvoudig uit te lezen.

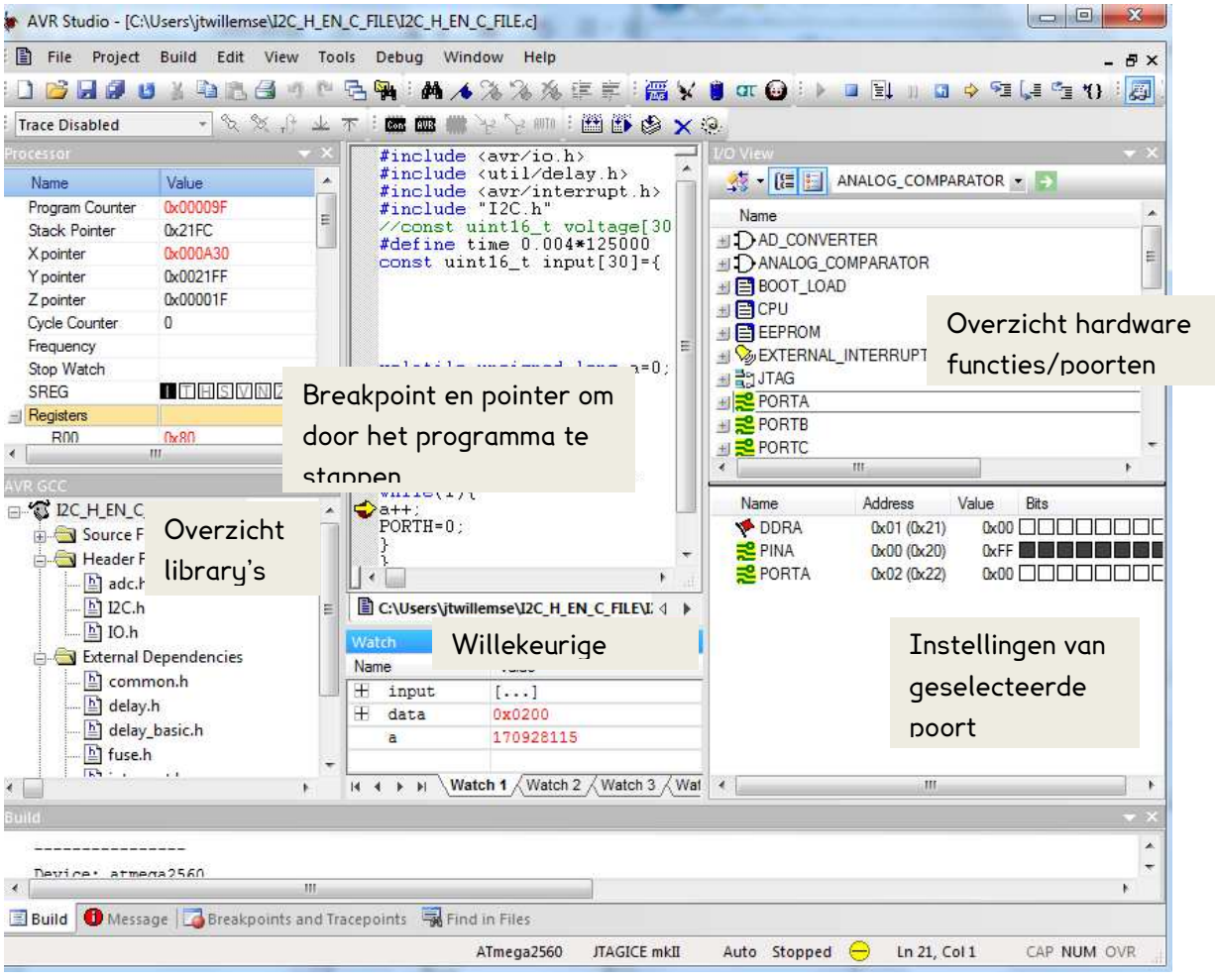
Er is geen mogelijkheid voor simulatie en/of doorlopen van het programma in stapjes of met breakpoints (zie figuur3).

Er is wel een mogelijkheid om met de hulp van een virtuele COM-poort gegevens uit te lezen, maar dit zorgt weer voor een extra belasting van de processor. De geschreven software in Arduino is niet makkelijk te porten naar AVR omdat een hoop functies van Arduino niet bekend zijn in AVR.

Van AVR naar Arduino zal daarentegen nauwelijks problemen opleveren.



Figuur 19: Arduino opslag programma's in Sketchbook en voorbeeldprogramma's



Figuur 20: AVR-studio simulatiemogelijkheden

AVR studio is gemaakt voor mensen die al iets verder zijn in het programmeren van microcontrollers. Het programmeren via AVR is minder eenvoudig, door minder voorbeelden en functies. Toch is gekozen om verder te werken met dit programma omdat de controle over de microcontroller groter is. Bij AVR-studio is alle library-code wel in te zien, kan de waarde in elk register uitgelezen worden zonder extra software en zijn er meer simulatie mogelijkheden. Door functies voor de gebruikers van het neuroplatform te maken wordt AVR alsnog gemakkelijk in gebruik, zonder de nadelen van Arduino. Daarnaast is het programma heel makkelijk te porten naar Arduino.

mogelijkheden	Arduino	AVR-studio
gratis	Ja	Ja
Mogelijkheid tot toevoegen van liberies	Ja	Ja
simulatie	Nee	Ja
Breakpoints, door programma stappen	Nee	Ja
Weergave, waarde van willekeurige variabele	Nee	Ja
Weergave van registers	Nee	Ja
Voorbeeldprogramma's	Ja	Nee
Informatie op internet	Goed	Redelijk
Snel en eenvoudig te programmeren	Goed	Slecht
Toegang tot interne liberies/sourcecode	Slecht	Goed
Porten naar ander programma	Slecht	Goed
Eenvoudige interface	Goed	Slecht
Efficiënt gebruik van processor	Slecht	Goed
installeren	Goed	Redelijk
Programma opslaan en openen	Goed	Redelijk

Tabel 3:Arduino VS AVRstudio

5.2 Softwareontwerp

In dit hoofdstuk wordt verder ingegaan hoe het geïmplementeerde programma is opgebouwd, welke keuzes zijn gemaakt bij het ontwerp van de functies en hoe elke functie geïmplementeerd moet worden.

5.2.1 Commentaar

In het programma is voor elke functie een informatieblok gecreëerd. In dit blok wordt gemeld wat de functie doet, hoe de functie werkt en wanneer deze aanwezig is, hoe de prescaler ingesteld wordt.

5.2.2 I/O & ADC

Bij de I/O en ADC is er geen gebruik gemaakt van interrupts. Voor de naam van de functies is Arduino als inspiratiebron gebruikt omdat de functies ongeveer hetzelfde werken en ten minste één van de beoogde gebruikers graag met Arduino werkt.

In Voorbeeldprogramma 1 zijn alle mogelijkheden gegeven. Er zijn een paar onderdelen in een kleiner lettertype geschreven. Dit is puur voor de weergave en heeft geen invloed op het programma.

Voorbeeld programma

```
#include <avr/io.h>
#include <avr/interrupt.h>//enable interrupts
#include "neurostimulatie.h"//maak gebruik van de voorgeprogrammeerde functies

Int a=0;//variabele is alleen voor het voorbeeld om digitalRead() en analogRead te
lezen.
int main{
initAnalog();
initDigital();
while(1){
a=digitalRead(0);//hardware-matig output->digital read = altijd 0
a=digitalRead(1);
a=digitalRead(2);
a=digitalRead(3);
a=analogRead(0);//geeft een waarde terug van maximaal 12bits
a=analogRead(1);
a=digitalWrite(0,0x0A);//wordt alleen uitgevoerd wanneer de jumper verbinding maakt
tussen 5V                    //en de directie poort(hardware-matig op output)
a=digitalWrite(1,0x0A);
a=digitalWrite(2,0x0A);
a=digitalWrite(3,0x0A);
}
}
```

Voorbeeldprogramma 1: analogRead & digitalRead/Write

Omdat er geen gebruik gemaakt is van interrupt bij de ADC, is er in de functie van readAnalog een while-loop aanwezig. Dit vertraagt het rekenproces wel, toch is gekozen voor deze methode omdat het belangrijker is dat er geen vertraging in de Interrupt Service Routines (ISR) voorkomt.

Er is gekozen om de 16 digitale I/O in te stellen in 4 groepen van 4 bits. Voor het bepalen van de groep wordt gebruik gemaakt van 0 t/m 3, omdat in het digitale domein geteld wordt vanaf 0 en niet vanaf 1. Het splitsen van de 16 I/O in 4 groepen is gedaan omdat de directie van de pinnen binnen een groep hardware-matig altijd hetzelfde is. Voor het softwarematig instellen van de groepen is per groep gebruik gemaakt van één extra input. Deze inputs zijn verbonden met de directie-poort van de bi-directionele level converter (zie hoofdstuk 0). Doordat de digitale I/O automatisch ingesteld wordt aan de hand van de directie-poort, kan er geen kortsluiting ontstaan bij de microcontroller, door menselijk falen. Daarnaast is het een stuk eenvoudiger voor de

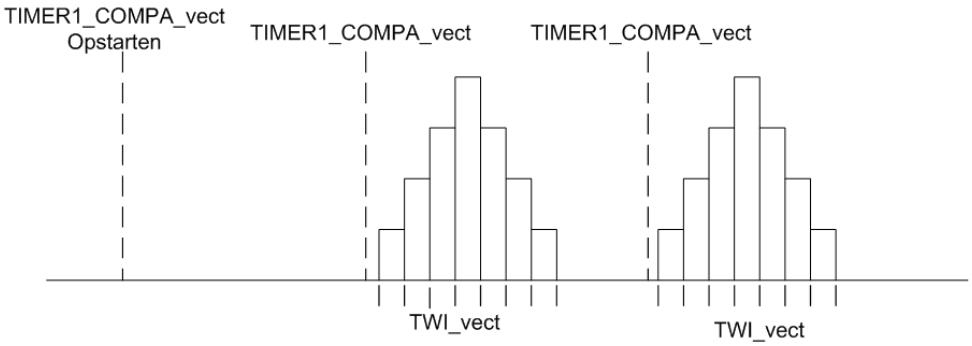
gebruiker. De gebruiker hoeft de directie van de pinnen niet in te voeren. Wanneer digitalRead en digitalWrite voor dezelfde pin gebruikt worden zal er geen gevaarlijke situatie ontstaan bij de microcontroller, behalve dat er maar één functie actief gebruikt wordt.

Het is ook mogelijk om de 4 inputs die de directie meten als outputs te programmeren, de vier pinnen kunnen dan actief de directie van de poort veranderen. Dit is nog niet geïmplementeerd en ook niet aan te bevelen zolang er nog geen goede methode is om te schakelen tussen input/output voor de front-ends en een betere methode is gevonden voor de kortsluitbeveiliging.

Voor de directie van de digitale I/O is wel een ISR geinitialiseerd (ISR(PCINT2_vect)) . Deze interrupt is echter alleen als beveiliging en zal niet gebruikt worden zolang de jumpers alleen geplaats/verwijderd worden wanneer de microcontroller uit is. Deze beveiliging werkt ook alleen voor de microcontroller en niet voor de front-ends.

5.2.3 DAC

De Digitaal Analoog Converter werkt met interrupts en werkt zo anders dan analogWrite dat de namen van de functies niet gebaseerd zijn op Arduino. Tijdens het programmeren is ervan uitgegaan dat de meest bruikbare vorm, de vorm van een periodiek-signaal is. In Figuur 21 is te zien dat TIMER1_COMA gebruikt wordt voor de herhaling van het signaal en de TWI interrupt gebruikt wordt voor het schrijven naar de DAC, via I^2C.



Figuur 21: DAC signaal

De timer is gebruikt omdat hiermee het interval tussen elke periode nauwkeurig bepaald kan worden (tijd van timer mag nooit korter zijn dan het signaal). De timer zal met de hulp van pointers het signaal updaten (data=data2). Daarnaast zorgt de timer voor de TWI-interrupt enable. De TWI-interrupt is speciaal gemaakt voor het I2C-protocol en wordt geactiveerd wanneer de I2C-bus gereed is voor transmissie.

Op dit moment is er maar één DAC goed getest. Daarnaast is de software nu zo gemaakt dat er maar één DAC tegelijk gebruikt kan worden. Er is aandacht aan maar één DAC besteed met het idee van een neurostimulator (heeft DAC nodig) en een

neurosensor (heeft geen DAC nodig). Daarnaast heeft de beschikbare tijd ook een rol gespeeld.

In dit programma is de bitrate gebruikt om de sample-rate te bepalen. Wanneer er twee DAC's gebruikt worden mogen de signalen van beide DAC's niet overlappen, omdat dit de snelheid nadelig kan beïnvloeden. Binnen de datatransmissie van een sample mag er geen onderbreking plaatsvinden. De samples van de twee DAC's kunnen wel na elkaar gewijzigd worden maar dit zal ook ten koste gaan van de snelheid aangezien de communicatie na elke sample herstart moet worden en de samples op elkaar moeten wachten. Wanneer er maar een DAC gebruikt wordt zal er per signaal een start zijn, moet de dataverbinding opgebouwd worden met de DAC, kan de informatie verzonden worden (zie input van Voorbeeldprogramma 2) en kan de dataverbinding afgesloten worden.

Voor uitgebreide informatie over het instellen van de sample-rate, zie hoofdstuk 6.1.2.

In Voorbeeldprogramma 2 is gegeven hoe de DAC geinitialiseerd moet worden en het signaal gewijzigd kan worden.

Voorbeeldprogramma 2

```
#include <avr/io.h>
#include <avr/interrupt.h>//enable interrupts
#include "neuroplatform.h"

#define time 0,02* 125000//0,02 is de tijd in seconde
const uint16_t input[10]={
0x0FFF,0x0000,0x0FFF,0x0000,0x0FFF,0x0000,0x0FFF,0x0000,0x03FF,0x0034};//signaal
const uint16_t input2[10]={
0x0FFF,0x0000,0x0FFF,0x0034,0x0FFF,0x0030,0x0FFF,0x0034,0x0FFF,0x0000};//signaal

int main{
initDAC(input,time);
sei();
while(1){
changeInput(input2);//veranderd het actieve signaal, altijd buiten een interrupt
}
}
```

neuroplatform.h

```
#define PERIOD 10//instellen van het aantal stapjes in een puls/signaal(zie Figuur 21, TWI_vect)
```

neuroplatform.c

```
void changeInputDAC(const uint16_t* input){
	d=0;
	if (data==data2){
		if(data!=changeData){
			changeData[0]=DAC1;//dit bepaald welke DAC er gebruikt wordt
			for(t=1; t<=TOTAAL; t=t+2){//mask input for right settings
				changeData[t]= getal=(((0x0F00))&input[d])>>8;
				changeData[t+1]=getal= (input[d]&0x00FF);
				d++;
			}
			data2=changeData;//set changeData as next active signal
		}
		else {
		changeData2[0]= DAC1;//dit bepaald welke DAC er gebruikt wordt
			for(t=1; t<=TOTAAL; t=t+2){
				changeData2[t]= getal=(((0x0F00))&input[d])>>8;
				changeData2[t+1]=getal= (input[d]&0x00FF);
				d++;
			}
		data2=changeData2;
		}
	}
}
```

Voorbeeldprogramma 2: DAC

5.2.4 Extern interrupt

De inputs EXT1 (PD2) en EXT2 (PD3) borduren voort op de DAC in de voorgeprogrammeerde basisinstellingen (zie Figuur 18). De inputs (PD2 en PD3) zullen de interrupts (INT2 en INT3) activeren wanneer er een spanning tussen 1 en 5 volt op de inputs aangesloten wordt. Er is gekozen om beide interrupts tegelijk te initialiseren omdat deze interrupts precies dezelfde functie hebben, deze interrupts nergens anders voor worden gebruikt en geen veiligheidsrisico vormen. Daarnaast wordt de code compacter en het geheel daardoor overzichtelijker.

De externe interrupts bestaan uit één initialisatie (initEXT) met als input twee signalen. De signalen worden tijdens de initialisatie omgezet tot een area van uint8_t, die naast het signaal ook de gewenste instellingen bevat. Daarna worden deze signalen apart opgeslagen per interrupt. Door de signalen van tevoren te verwerken hoeft dit niet tijdens de interrupt uitgevoerd te worden, waardoor de interrupt sneller wordt uitgevoerd. In de interrupts kunnen wijzigingen uitgevoerd worden wanneer een andere actie gewenst is dan het veranderen van het signaal (zie Voorbeeldprogramma 3).

Voorbeeldprogramma 3

```
#include <avr/io.h>
#include <avr/interrupt.h>//enable interrupts
#include "neuroplatform.h"//maak gebruik van nieuwe functies
```

```
const uint16_t input1[30]={ 0x0FFF,0x0000,0x0FFF,0x0000,0x0FFF,0x0000,0x0FFF,0x0000,0x0FFF,0x0000};

const uint16_t input2[30]={ 0x0FFF,0x00FF,0x000F,0x0000,0x000F,0x00FF,0x0FFF,0x00FF,0x0FFF,0x0000};

int main{
initEXT();
sei();
while(1);
}
```

Voorgeprogrammeerde interrupts, de input voor de interrupts zijn pin PD2 en PD3

```
ISR(INT2_vect)
{
data2=outputINT2;          //this will change the next form of the DAC after the trigger on PD2,
                    //when you want to do something els, change the code on this place
}
/*********************************************************************************************
*This will be activated when EXT interrupt is initialised and the pin becames high                    *
*********************************************************************************************/
ISR(INT3_vect){
data2=outputINT3;          //this will change the next form of the DAC after the trigger on PD3,
                    //when you want to do something els, change the code on this place
}
```

Voorbeeldprogramma 3: external interrupts

6 Testen

Het neuroplatform moet op meerdere manieren getest worden, zodat er geen onverwachte situaties ontstaan die tot problemen kunnen leiden. Voor het neuroplatform is dit onder te verdelen in twee belangrijke onderdelen: de timing, dit is erg belangrijk voor de reactietijd van de externe interrupt. De hardware, hierbij wordt gecontroleerd of alle componenten naar behoren werken, wat het stroomverbruik is en alle mogelijke spanningen.

6.1 Timing

Een belangrijk onderdeel bij het testen het neuroplatform is de timing. De timing is erg belangrijk omdat door de artsen de eis is gesteld dat het platform een synchronisatiepuls moet verwerken binnen 1ms.

Naast de eis van de artsen is het ook van belang om te weten welke CPU-tijd over blijft voor berekeningen. Dit is van belang voor het uitvoeren van real-time berekeningen en interactie tussen de neurosensor en de neurostimulator.

Voor de synchronisatietijd en de belastings-tijd van de microprocessor moet er eerst bepaald worden in welke onderdelen het programma opgedeeld kan worden, welke onderdelen van het programma voorrang krijgen (bijv. interrupts) en daarmee andere programma's kunnen vertragen en daarmee de maximale belastings-tijd voor andere programma's verkleinen.

6.1.1 Assembly VS Digitale uitgang

Er zijn twee mogelijkheden om de belastings-tijd op de processor (CPU) te bepalen:

1. De assemble-code onderzoeken, Deze code bestaat uit een aantal functies. Van elke functie is de belasting op de CPU bekend in clockcycles. De clockcycles kunnen omgerekend worden naar de tijd dat de CPU belast is.
2. Door een digitale uitgang uit te sturen aan het begin en eind van een functie kan bepaald worden met de hulp van een oscilloscoop, hoe groot de belasting is op de CPU. De tijd die gebruikt kan worden voor overige acties kan bepaald worden door in het hoofdprogramma de digitale uitgang laag te maken en aan het begin van alle overige programma's de uitgang hoog te maken.

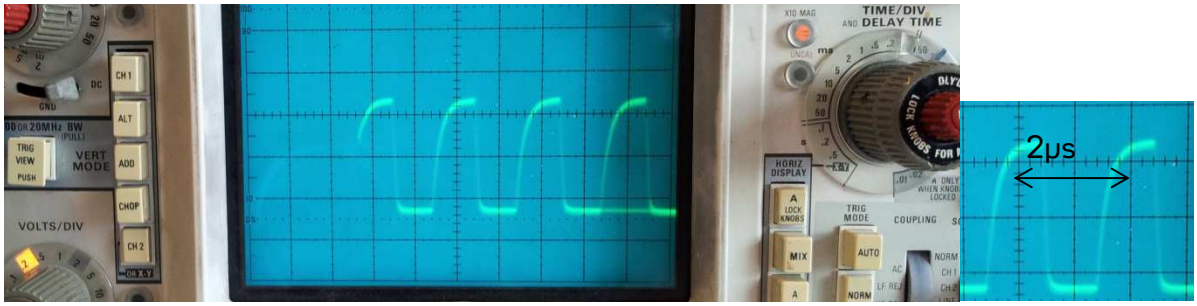
Het meten van de tijd met de hulp van assembly nauwkeuriger dan met een digitale uitgang, doordat er geen vertraging is van de digitale uitgang en onafhankelijk bepaald kan worden van de snelheid van de processor. Toch is er gekozen voor het meten met de hulp van een digitale uitgang. Een digitale uitgang is veel eenvoudiger toe te passen en houd ook rekening met externe factoren die tot vertragingen kunnen leiden.

	Assembly	Digitale uitgang
Praktisch	nee	Ja
theoretisch	ja	Nee
Kleine afwijking	nee	Ja
Houd rekening met externe factoren	nee	Ja
Is eenvoudig toe te passen, voor elke functie hetzelfde	nee	Ja
Is nauwkeurig om te zetten in een andere snelheid van de CPU	ja	Nee

Tabel 4: Assembly VS Digitale uitgang

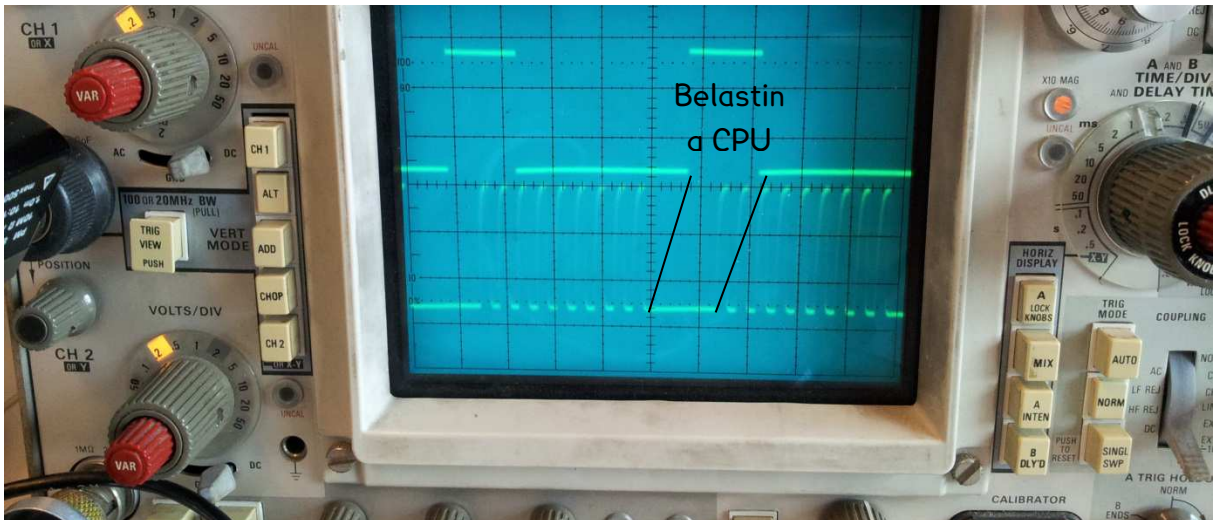
6.1.2 Timing DAC

De DAC wordt aangestuurd via de I^2C, de communicatiesnelheid van deze bus is altijd minimaal 16x lager dan de kloksnelheid van de microcontroller(8Mhz). Dit betekend dat de communicatiesnelheid maximaal 500kbps is(zie Figuur 22)

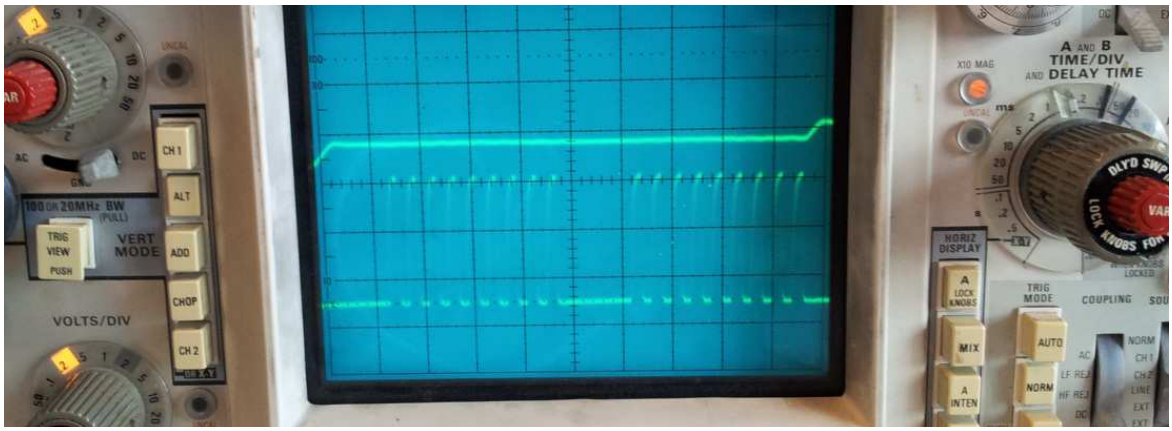


Figuur 22: I^2C communicatiesnelheid

Voor de timing en belasting van de DAC is een meting gedaan(zie Figuur 23). Het bovenste signaal is de DAC met willekeurige waardes. Het onderste signaal geeft de tijd aan dat de microprocessor wordt belast. De microprocessor wordt belast wanneer het onderste signaal hoog is, de tijd dat de CPU niet belast wordt is afhankelijk van de communicatiesnelheid tussen de microcontroller en de DAC, aangezien de CPU niet belast wordt door de transmissie. Wanneer de bitrate afneemt, neemt de sample-rate af en de belasting af.



Figuur 23: Belasting CPU vs bitrate



Figuur 24: communicatie I^2C

De belasting per sample zal altijd gelijk blijven, de belasting is de voorbereiding op de transmissie, aangezien een sample in twee keer verzonden wordt(zie Figuur 24) betekend dit een belasting van 14us per sample. De tijd voor het verzenden van één bit is minimaal 2µs

In figuur 10 is te zien dat de capaciteit op de microprocessor 2X 7µs bedraagt, de tijd dat dat de gegevens verzonden worden is 18x2µs in deze tijd wordt de cpu van de microprocessor niet belast. Zolang de microprocessor met een kloksnelheid van 8MHz werkt zal de capaciteit altijd 14µs zijn per verandering van de spanning. Dit betekend een kleinere belasting bij een lagere sample-rate van de DAC. De sample-rate is in te

stellen met de hulp van de bitrate van de I^2C bus, het is dus handig om een zo laag mogelijke bitrate te gebruiken, waarbij de sample-rate van de DAC nog steeds voldoet. Voor het instellen van de gewenste sample-rate moet eerst de bitrate bepaald worden: $\frac{1/minimale\ gewenste\ snelheid\ in\ Hz - 14us}{18} = bitrate$, wanneer de bitrate bekend is moet bepaald hoe de prescaler en het bitrate-register ingesteld moeten worden, hiervoor is een formule gegeven in de datasheet van de microcontroller(ATMega2560):

$2 * TWBR * 4^{TWPS} = \frac{8Mhz}{bitrate} - 16$ Deze belasting is exclusief het instellen van de DAC, dat bij elke cyclus opnieuw uitgevoerd wordt.

Voorbeeldberekening instellen I2C bij een sample-rate van 357Hz
1/357Hz=2,8ms->(2,8ms-14us)/18=0,1548ms=6458Hz 8Mz/6458Hz=1239Hz->(((1239-16)/2)/4)=152=0x98->TWBR=0x98 14us= voorbereidingstijd van transmissie/belastingstijd CPU , 18= aantal bits voor een sample, 8Mz= kloksnelheid, 2 = twee transmissies nodig voor een sample, 4= prescaler(TWPS)

Tabel 5: voorbeeldberekening I2C bij sample-rate van 357Hz

In deze berekening is de belastingstijd van timer 1 buiten beschouwing gehouden. Er is wel al duidelijk dat de tijd belastingstijd van de timer en het interrupt kort genoeg zijn om geen problemen te veroorzaken bij het behalen van een reactietijd van 1ms. Voor de initialisatie moet de belastings-tijd van initDAC worden gemeten, dit is echter van ondergeschikt belang omdat paar seconden meer of minder tijdens de initialisatie geen effect heeft op de prestaties van het neuroplatform.

6.2 Basis hardware test

In Tabel 6 worden alle onderdelen van het neuroplatform zo uitgebreid mogelijk getest, hierbij is te zien dat alles naar behoren werkt, behalve de verstelbare voeding, in combinatie met de bi-directionele levelconverter. Wanneer spanning hoger wordt dan 4 volt, neemt de stroom explosief toe, de reden hiervoor is onbekend.

testen print		
onderdeel	meting multimeter	stroommeting power supply
voedingsspanning direct van de 7805	5,03V	x
ingang voeding	5,03V	x
voeding1	0,49V-5,03V	stroom neemt explosief toe buiten de range van 0,96-4V(ongeveer 8mA extra) boven 0,8 volt blijft het stroomverbruik enigszins stabiel, het stroomverbruik zal niet afnemen en niet toenemen
voeding2	0,49V-5,03V	||
voeding3	0,49V-5,03V	||
voeding4	0,49V-5,03V	||
voeding5	0,49V-5,03V	||
DAC zonder microcontroller	4-4,17V	||
I/O zonder dat er 5V aangeboden wordt (aan front-end), zonder shunt	0V	
I/O met 5V aangeboden (aan front-end), met shunt	ong. 5v op I/O pin aangeboden, op front-end komt spanning, 5v spanning losgekoppeld na een tijdje gaat spanning weer naar 0V	stroom nog steeds onder 1mA
I/O zonder dat er 5V aangeboden wordt (aan microcontroller), zonder shunt	er leek even 5,04 volt op een pin te staan, toen ik nog een keer keek was het weg	
I/O met 4V aangeboden (aan microcontroller), zonder shunt	spanning aanbieden op front-end, op mic ontvangen 5,04V, langzaam naar beneden omdat daarna geen 0V is aangeboden	stroom schommeld een beetje, stroom 6-7mA
shunt aansluiten		toename van iets minder dan 2mA per shunt

Tabel 6: hartware test

7 Resultaten en Conclusies

Aan het eind van dit project is er een prototype 1.0 van het neuroplatform. Dit platform is in staat om één DAC aan te sturen en er zijn twee ADC ingangen aanwezig waarmee de spanning gemeten kan worden. In totaal zijn er 16 I/O beschikbaar. Deze I/O kunnen per groep van vier als input of output ingesteld worden. Tijdens een meting met de multimeter M970, bleek de basis-spanning niet 5 maar 5,03V te zijn. Of de afwijking in de spanning aan de multimeter ligt is niet bekend. Er zijn 5 instelbare spanningen beschikbaar. Drie van deze spanningen moeten met extra zorg behandeld worden omdat deze de bi-directionele level converter voeden. De level converter zal hoge stromen trekken wanneer de spanning lager wordt dan 1 a 0,8V en hoger dan 4V. Het instellen van de I/O is op dit moment met een pull-down weerstand uitgevoerd en een jumper. Wanneer de jumper een verbinding maakt tussen 5V en de directieport zal de pull down een verbinding maken tussen 0 en 5V, dit heeft als gevolg dat er per jumper iets minder dan 2mA verbruikt wordt. De maximale stromen zijn nu ongeveer 10mA, wanneer deze hoger zijn betekent dit dat een er kortsluiting optreedt bij de I/O van de bi-directionele poort. De software van dit platform is gereed voor gebruik. Hierin zijn nog wel wat verbeteringen mogelijk, met name voor de DAC. Al met al is het gelukt om de minimale specificaties uit te voeren, maar er is nog ruimte voor belangrijke verbeteringen. De enige twee onderdelen die niet uitgevoerd zijn, zijn de output van de klok en de microSD. Hiervan was wel al van tevoren bepaald dat deze geen prioriteit hadden.

8 Aanbevelingen

In hoofdstuk 7 wordt er ingegaan op het instellen van de directiepoort, waarbij een pull-down is gebruikt van 100kohm. Het is aan te bevelen om, indien mogelijk, deze minimaal vijf keer zo groot te maken want op dit moment zijn de pull-ups de grootste belasting van het hele neuroplatform.

Op het neuroplatform zijn twee DAC's aanwezig, waarbij op dit moment alleen de mogelijkheid is om één DAC tegelijkertijd in gebruik te nemen. Wanneer het toch gewenst is om met beide DAC's te werken is aan te bevelen met de hulp van changeInputDAC of TIMER1_COMPA_vect te switchen tussen de twee verschillende DAC's. Bij het gebruik van changeInputDAC zal deze na elke keer dat het signaal uitgevoerd is de input van het signaal switchen met een andere input. Hierbij is het van belang dat er genoeg tijd over is voor changeInputDAC om deze conversie uit te voeren binnen elke cyclus. TIMER1_COMPA_vect zal altijd het signaal starten en hier is dus ook de mogelijkheid om te switchen tussen de input van DAC0 en DAC1. Bij beide opties geldt wel dat er maar één timer gebruikt wordt, om overlap te voorkomen.

In de software wordt bij initDAC nu meteen de DAC/TIMER1_COMP_vect geactiveerd. Dit betekendt dat er geen volledige controle is over het starten van de DAC. Wanneer er een aparte functie gemaakt wordt die alleen de interrupt Timer1_COMPA_vect enabled, zal de DAC precies op dat moment uitgevoerd worden zodra de eerste opstartperiode voorbij is(zie Figuur 21: DAC signaal). Daarnaast kan een functie gemaakt worden die de interrupt één keer enabled en daarna weer disabled, zodat het signaal ook één keer uitgevoerd kan worden.

In hoofdstuk 7 wordt ook gesproken over mogelijke kortsluiting, voor deze kortsluiting is wel een oplossing gevonden in hoofdstuk 4.5.2, deze beveiliging is helaas pas achteraf ontworpen en daarom niet ingebouwd, wanneer de beveiliging uit hoofdstuk 4.5.2 niet toegepast wordt is het alsnog handig om de OE ingang aan te sluiten met een pull-up. Na het opstarten moet de OE ingang met 0V verbonden worden.

Ten behoeven van de kortsluitbeveiliging kan het gebruik van het IC MCP18480 verder onderzocht worden.

In hoofdstuk 7 is ook aangegeven dat drie instelbare spanningen problemen kunnen veroorzaken wanneer deze een te lage spanning geven. Dit is mogelijk te verhelpen door een weerstand in serie te plaatsen, zodat het bereik van de LDO(zie hoofdstuk 0) binnen de spanningen valt die nodig zijn voor de bi-directionele level converter.

Op dit moment is het alleen mogelijk om signalen in te voeren via de computer. Wanneer artsen zelf ook nieuwe signalen willen uitproberen is dit niet de meest gebruiksvriendelijke manier. Daarnaast is er nu geen mogelijkheid om gegevens op te

slaan ter evaluatie. Daarom is het aan te bevelen om de micro-SD alsnog te inplementeren.

9 Bronnen

http://www.arduino.cc/

http://nl.farnell.com/

http://www.ni.com/white-paper/6349/en

http://www.neuro-biofeedback.nl/webtest/index.php/wetenswaardigheden/2-ongecategoriseerd/19-hersenfrequenties

http://www.cvphysiology.com/Arrhythmias/A009.htm

http://en.wikipedia.org/wiki/QRS_complex

http://www.chlazza.net/sdcardinfo.html

10 Bijlage

10.1 Componenten

bestellen	fabrikant-nummer	prijs	site	ordernummer	aantal	totaal
referentiespanningen	ADP123AUJZ-R7	1,15	farnell	1855247	5	5,75
referentiespanningen IC, LDO, 3.3V, 0.3A, 5TSOT	ADP122AUJZ-3.3-R7	0,95	farnell	1858064RL	1	0,95
biderectionele communicatie	SN74LVC8T245PW	1,32	farnell	1236407	5	6,60
SOCKET, MICROSD, PUSH-PUSH, 1.8MM	502774-0891	2,30	farnell	2064063	1	2,30
SANDISK - SD4661 - MEMORY, MICRO SD, 2GB	SD4661	13,76	farnell	1795130	1	13,76
biderectionele poort zonder dir pin(4input)	ADG3304BRUZ	3,20	farnell	1078228	1	3,20
voeding	L7805ACD2T-TR	0,83	farnell	1467759	2	1,66
OP AMP, SINGLE, 1.8V, 1MHZ, 5SOT-23	MCP6L01T-E/OT	0,25		1715852RL	2	0,50
CAPACITOR CERAMIC, 1UF, 16V, X5R, 20%, 0603	MC0603X105M160CT	0,014	farnell	1710292	20	0,28
CAPACITOR CERAMIC, 0.1UF, 25V, X7R, 10%, 0603	C1608X7R1E104K	0,006	farnell	2146271	19	0,11
CAPACITOR CERAMIC 0.33UF, 25V, X5R, 10%, 0603	C1608X5R1E334K	0,023	farnell	1844214	2	0,05
TRIMMER, SMD, 5 TURN 100K	3214W-1-104E	2,47	farnell	988303	5	12,35
TRIMMER, SMD, 5 TURN 5K	3214W-1-502E	2,47	farnell	988261	2	4,94
connector naar microcontroller-bordje	67996-420HLF	0,7	farnell	1918006	3	2,10
TERMINAL BLOCK, 2WAY	CTB0308/2	0,59	farnell	3882652	1	0,59
RECEPTACLE, 2.54MM, VERT, 10WAY	SSM-110-L-SV	2,13	farnell	1668264	2	4,26
SWITCH, SPST, 24V, 160GF, SMD	4-1437565-2	0,55	farnell	2060823	1	0,55
totaal						59,95

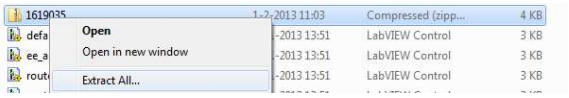
10.2 Werking eagle

10.2.1 Inporteren van farnell

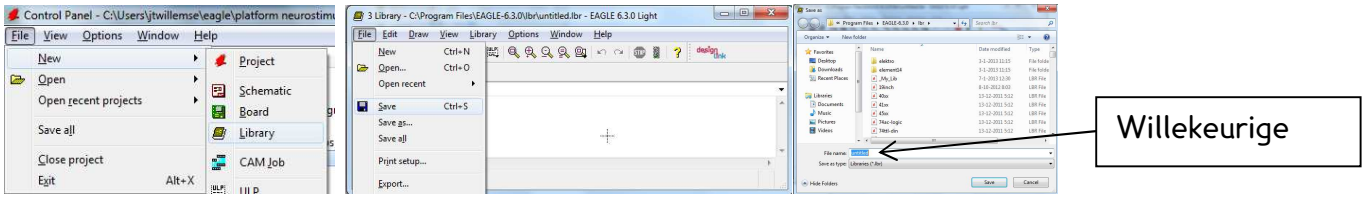
Bij farnell zijn er van sommige componenten eagle tekeningen beschikbaar, deze tekeningen kunnen gedownload worden, zodra er een account is aangemaakt. Wanneer er een tekening is wordt dit weergegeven onder de specificaties van het component(zie figuur)

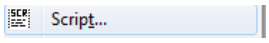
CAD CadSoft_EAGLE (3.21KB) EN

Wanneer deze gedownload is, moet deze opgeslagen en ge-unzipt worden in een willekeurige folder.

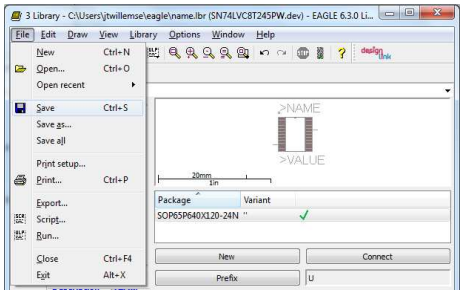


In eagle kan nu een nieuwe bibliotheek gemaakt worden of gebruik gemaakt van een oude bibliotheek. Voor een nieuwe bibliotheek moet deze eerst aangemaakt worden en dan opgeslagen met een willekeurige naam.



In de library kan nu een script geïmporteerd worden via file->script, de gedownloade file moet daarna ingevoerd worden. 

Het device scherm komt naar voren, de laatste stap is opslaan en het component is nu toegevoegd aan de library



10.2.2 Nieuw component maken

Voor het maken van een component moeten drie stappen doorlopen worden, het maken van een package ,het maken van een symbool , het maken van een device(samenvoegen) .

Voor deze stappen gedaan kunnen worden moet eerst een gewenste bibliotheek geopend worden(zie hoofdstuk..).

10.2.3 Package

Voordat er begonnen wordt met het maken van de package, moet er eerst gekeken worden of deze al in de bibliotheek aanwezig is en anders of er connectors zijn met dezelfde pinafstand.

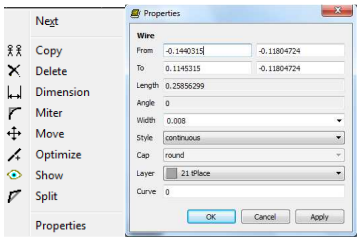
Wanneer er een package aanwezig is in de bibliotheek kan deze in het hoofdscherm met rechter muistoets geselecteerd worden en toegevoegd de bibliotheek die geopend is.

Wanneer er alleen een connector is met dezelfde pinafstanden, moet de bibliotheek waar de gewenste connector aanwezig is geopend worden, dan kan er op het symbool geklikt worden van package en is het mogelijk om de gewenste connector te selecteren. De connector kan dan gekopieerd worden door alles te selecteren met Group . Klik op Copy , rechter muistoets en copy group. Wanneer ctrl toets ingehouden wordt gaat de muis naar de dicht-bijzijnste connectie, dit is erg handig omdat de connecties precies geplakt kunnen worden waar deze gewenst zijn.

Wanneer de eigen bibliotheek geopend is moet package weer geopend worden en een nieuwe naam opgegeven. Met paste wordt de gekopieerde connector nu in de tekening van het nieuwe package geplaatst.

Bij het tekenen moet verder nog rekening gehouden worden met de grit, wanneer deze bij het plaatsen van de connecties(pad of smd) in mm staat kunnen er problemen optreden wanneer het component verbonden moet worden doordat het grit voor componenten bij voorkeur hetzelfde moet zijn. Er moet ook rekening gehouden worden met een kruis dat op de pagina staat, dit is het middelpunt, het punt van de muis bij het plaatsen van het component

Het tekenen in eagle wordt slecht ondersteund, het aan te raden om met het instellen van de grit, of met de hulp van coördinaten de exacte vorm te teken (zie figuur...).



Figuur 25: rechter muistoets, Properties coördinaten

10.2.4 Symbool

Voor het symbool kunnen op de zelfde manier als bij het package onderdelen gekopieerd worden.

De pinnen kunnen met rechter muistoets gedraaid worden en door middel van properties of Name

kan de naam veranderd worden, met properties kunnen ook gewijzigd worden waar de pin voor bedoeld is bijvoorbeeld als I/O of voor power. De vorm is niet echt van belang, dit wordt toch alleen maar gebruikt voor het schema.

10.2.5 Device

Wanneer het symbool en package gereed zijn kunnen de pinnen verbonden worden, dit gebeurd in het device.

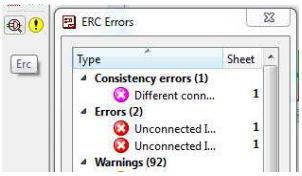
Een device wordt op de zelfde manier aangemaakt als het package en Symbool.

Het symbool kan toegevoegd worden via add en het package kan toegevoegd worden via new.

De pinnen kunnen verbonden worden via connect en er kunnen eventueel nog extra gegevens toegevoegd worden via attribute.

10.2.6 No forward backward notation

Wanneer deze foutmelding weergegeven wordt betekend het dat het board en schema niet meer met elkaar verbonden zijn. Het board en schema moeten altijd tegelijk open en gesaved worden, tenzij er nog geen board gegenereerd is. De foutmelding kan hersteld worden door het board te verwijderen en opnieuw te genereren (met deze knop kan daarna geswitcht worden). Wanneer er al aan het board gewerkt is kan kunnen de verschillen tussen het board en schema aangepast worden, de verschillen zijn te vinden door op erc te klikken. De consistency errors moeten hersteld worden om ervoor te zorgen dat het schema en board weer verbonden zijn.



10.3 Neuroplatform.h

```
#ifndef _NEUROPLATFORM_H_
#define _NEUROPLATFORM_H_

#include <avr/io.h>

//when scrolling throu NEUROPLATFORM.c, the groups are easy to see so you can go fast to the wanted function
/************************groups*/
/********
*comment*
********/
//programm(){}

//this is the amount of changes(integers of area) of voltage in one puls
//(input must be a integer area with (0-4095)
#define PERIOD  10
//this is the amount of uint8_t that is required internally
#define TOTAAL  ((PERIOD*2)+1)
#define DAC0    0xC0
#define DAC1    0xC2

//input(a string of integer that make the voltage vorm(0-5V=0-4095))
// time(the time between each form, must be more than the PERIOD*(time of one change voltage) (time=time in sec*62500))
void initDAC(const uint16_t* input/*area of integer*/,int time/*time in seconds*62500*/);


void offDAC(void);

//change the form of the puls of the DAC
void changeInputDAC(const uint16_t* input/*area of integer*/);

//time is de tijd in seconde*125000(8Mz/64=125000)
//void startI2COnce(const uint16_t* input/*area of integer*/);

//this saves the two wanted signal wen and change it when activating external interrupt one or two
void initExt(const uint16_t* input2/*area of integer*/, const uint16_t* input3/*area of integer*/);


void initDigital(void);

int digitalRead(uint8_t ch/*0-3*/);

void digitalWrite(uint8_t ch/*0-3*/,uint8_t data/*0b00000000-0b00001111*/);

void initAnalog(void);

int analogRead(uint8_t ch/*0-1*/);


#endif
```

10.4 Neuroplatform.c

C:\Users\joerf\Downloads\neuroplatform (1).c zondag 10 maart 2013 14:24

```
#include <avr/io.h>
#include<avr/interrupt.h>

#include"NEUROPLATFORM.h"

/* initials, not in h file becourse it is not alowed to use it other files then this
file**********************initials**/
int i=0;
int a=0;

uint8_t* data;//DAC change signal
//used by ISR(TIMER1_COMPA_vect)

uint8_t* data2;//DAC change signal
//used by ISR(TIMER1_COMPA_vect), ISR(INT2_vect), ISR(INT3_vect) and changeInputI2C

uint8_t changeData[30];//DAC change signal
uint8_t changeData2[30];//DAC change signal
//used by changeInputI2C

uint8_t outputINT2[30];//extern interrupt for DAC change signal
//used by ISR(INT2_vect) and initExt
uint8_t outputINT3[30];//extern interrupt for DAC change signal
//used by ISR(INT3_vect) and initExt

int change=0;//DAC change signal
//used by changeInputI2C
int ready=1;//when this is 1 the input of the DAC can be changed by changeInputDAC,when 0 not
//used by ISR(COMPA_TIMER1_vect) and changeInputDAC

volatile    int t=1;
volatile    int d=0;
volatile    uint8_t getal;
/*functions*************************************************************************************
***functions******functions*/

/*DAC****all the functions for setting the signal
waveform**********************************DAC functions**DAC functions*/

/*****************************************************************************************
*to control the DAC, the first thing is to initialise the I2C(two wire communication->TWI), *
*to initialise the TWI you have to set the registers below this text, when enable the       *
*every time the communication is ready, the TWI interrupt activates                         *
*    - TWAMR(TWI (Slave) Address Mask Register) will set to 0b11111110, there is no active  *
*      mask                                                                                 *
*                                                                                           *
*calculation for the speed(TWBR)/sample rate. Calculation example:                          *
*1/357Hz=2,8ms->(2,8ms-14us)/18=0,1548ms=6458Hz                                             *
*8Mz/6458Hz=1239Hz->(((1239-16)/2)/4)=152=0x98->TWBR=0x98                                   *
*14us=time consuming by microcontroller,18=the amount bits for one sample,8Mz=clock rate    *
*    - TWCR                                                                                 *
*    - TWSR(set prescaller)                                                                 *
*-----------------------------                                                              *
*|TWPS1|TWPS0|division factor|                                                             *
*|-----|-----|---------------|                                                             *
*|  0  |  0  |       1       |                                                             *
*|  0  |  1  |       4       |                                                             *
```

-1-

```
*|  1  |  0  |      16       |                                                                *
*|  1  |  1  |      64       |                                                                *
*-----------------------------
*                                                                                 *
*Most of the time we want a periodic signal and be able to change the time between the      *
*pulses, for that the TIMER1_COMPA_vect is used, in this settings, the timer counts until it*
*has the same value as OCR1A, then activate the interrupt TIMER1_COMPA_vect.               *
*TIMER1_COMPA_vect will enable the TWI interrupt that handle the I2C communication         *
*                                                                                          *
*OCR1A= time in seconds(outcome will be changed by the division factor)                    *
*-------------------------------------------------------------------------------           *
*|ADPS2|ADPS1|ADPS0|division factor                                          |             *
*|-----|-----|-----|--------------------------------------------------------|             *
*|  0  |  0  |  0  |No clock source. (Timer/Counter stopped)                |             *
*|  0  |  0  |  1  |1                                                       |             *
*|  0  |  1  |  0  |8                                                       |             *
*|  0  |  1  |  1  |64                                                      |             *
*|  1  |  0  |  0  |256                                                     |             *
*|  1  |  0  |  1  |1024                                                    |             *
*|  1  |  1  |  0  |External clock source on Tn pin. Clock on falling edge  |             *
*|  1  |  1  |  1  |External clock source on Tn pin. Clock on rising edge   |             *
*-------------------------------------------------------------------------------           *
*******************************************************************************************/
void initDAC(const uint16_t* input,int time){
    changeInputDAC  (input);
/*settings I2C*/
TWAMR=0b11111110;//mask register,set like this there will be no mask
//PRR0|=(1<<PRTWI);//??
//PRR0&=~(1<<PRTWI);//??
TWBR=0x98;//speed (bitrate), when use this instate of 0x00 jou get a sample rate of 357Hz
//TWBR=0x00;//this has a sample rate of 20kHz
TWCR=(1<<TWIE);//TWI_vect interrupt enable
TWSR|=(1<<TWPS0);//prescaller(4) works only with TWBR>0, can be 1,4,16,64

/*settings timer*/
OCR1A=time;//compare register
TCCR1A=(1<<COM1A1);//enable compare on register A
TCCR1B=(1<<CS11|1<<CS10|1<<WGM12);////ctc mode(WGM..) and prescaler64(CS..)
//TIMSK1=(1<<OCIE1A);//TIMER1_COMPA_vect interrupt enable
}

/*******************************************************************************************
* offDAC set timer1 off, this means that when there is stil a puls/signal,                  *
*this will finish and that is the last puls/signal of the DAC                               *
*******************************************************************************************/
void continuePulseDAC(void){
TIMSK1|=(1<<OCIE1A);
}
void offPulseDAC(void){
TIMSK1&=~(1<<OCIE1A);
}
void oncePulseDAC(void){
TIMSK1|=(1<<OCIE1A);
TIMSK1&=~(1<<OCIE1A);
}
/*******************************************************************************************
*changeInputI2C convert the given input signal to the right settings, the converted settings*
```

C:\Users\joerf\Downloads\neuroplatform (1).c zondag 10 maart 2013 14:24

```
*will be linked to data2. changedata and changedata2 will switch active state and           *
*resume state, the changes will always be made in the one that is in resume state,          *
*becourse of that the active signal can't become a combination of the old signal and the new*
*signal. input must be an area of PERIOD(see neuroplatform.h) uint16_t                      *
*with each uint16_t between 0-4095(0-5V)
*                                                                                          *
*******************************************************************************************/
void changeInputDAC(const uint16_t* input){
    d=0;
    if (data==data2){
        if(data!=changeData){
            changeData[0]=DAC1;//set the IP adress of the DAC
            for(t=1; t<=TOTAAL; t=t+2){//mask input for right settings
                changeData[t]= getal=(((0x0F00))&input[d])>>8;
                changeData[t+1]=getal= (input[d]&0x00FF);
                d++;
            }
            data2=changeData;//set changeData as next active signal
            //change=1;//switch from changing changeData to changeData2
        }
        else {
        changeData2[0]=DAC1;
            for(t=1; t<=TOTAAL; t=t+2){
                changeData2[t]= getal=(((0x0F00))&input[d])>>8;
                changeData2[t+1]=getal= (input[d]&0x00FF);
                d++;
            }
        data2=changeData2;
        //change=0;
        }
        //ready=0;
    }
}
/* extern
interrupt*******************************************************************************extern
interrupt*/
/*******************************************************************************************
*initExt change signal to signal inputINT2 for interrupt INT2 and                           *
*inputINT3 for interruptINT3. The right settings became saved in the area outputINT2        *
*and outputINT3. The for lus works exactly the same as changeInputDAC.                      *
*DDRD&=~0b00110000 sets the two external interrupts as input,                               *
*in EICRA can set by wtch action the ISR is activated, now it will be activated when        *
*the signal goes from negatieve to active.                                                  *
*EIMSK enables input ISR INT2 and INT3                                                      *
*******************************************************************************************/
void initExt(const uint16_t* inputINT2, const uint16_t* inputINT3){
    int d;
    int t;
    d=0;
    outputINT2[0]=0xC2;
    for(t=1; t<=TOTAAL; t=t+2){
        outputINT2[t]=(((0x0F00))&inputINT2[d])>>8;
        outputINT2[t+1]= (inputINT2[d]&0x00FF);
        d++;
    }
    d=0;
    outputINT3[0]=0xC2;
```

```
    for(t=1; t<=TOTAAL; t=t+2){
        outputINT3[t]=(((0x0F00))&inputINT3[d])>>8;
        outputINT3[t+1]= (inputINT3[d]&0x00FF);
        d++;
    }
    DDRD&=~0b00110000;
    EICRA=(1<<ISC21)|(1<<ISC20)|(1<<ISC31)|(1<<ISC30);
    EIMSK=(1<<INT2)|(1<<INT3);
}

/*ADC
**********************************************************************************************
**********ADC*/


/**********************************************************************************************
*initAnalog initialise the first two pins of portf as input with DDRF and PORTF,            *
*in ADCSRA the ADC will be enabled and the prescaler will be set,the prescaler is set on 128*
*-----------------------------------                                                         *
*|ADPS2|ADPS1|ADPS0|division factor|                                                        *
*|-----|-----|-----|---------------|                                                        *
*|  0  |  0  |  0  |        2      |                                                        *
*|  0  |  0  |  1  |        2      |                                                        *
*|  0  |  1  |  0  |        4      |                                                        *
*|  0  |  1  |  1  |        8      |                                                        *
*|  1  |  0  |  0  |        16     |                                                        *
*|  1  |  0  |  1  |        32     |                                                        *
*|  1  |  1  |  0  |        64     |                                                        *
*|  1  |  1  |  1  |        128    |                                                        *
*-----------------------------------                                                         *
*     VIN · 1024                                                                           *
*ADC= ----------                                                                           *
*       VREF                                                                               *
**********************************************************************************************/
 void initAnalog(){
    //make input for using ADC
    DDRF&=0b11111100;
    PORTF&=0b11111100;
    //reference for ADC, do not change!(connected to 5V)
    ADMUX|=(1<<REFS0);
    //ADC enable
    ADCSRA|=(1<<ADEN)|(1<<ADPS0)|(1<<ADPS1)|(1<<ADPS2);
    //ADC as analog comperator
    ADCSRB|=(1<<ADTS0);
    //ADC start conversion
    ADCSRA|=(1<<ADSC);//

 }

/**********************************************************************************************
*analogRead gives the voltage with a resolution of
**********************************************************************************************/
int analogRead(uint8_t ch/*0-1*/){
  //Select ADC Channel ch must be 0-1
  ch=(ch& 0x01|(1<<REFS0));
   ADMUX=ch;
```

```
   //Start Single conversion
   ADCSRA|=(1<<ADSC);

   //Wait for conversion to complete
    while(!(ADCSRA & (1<<ADIF)));

   //Clear ADIF by writing one to it
   ADCSRA|=(1<<ADIF);

   return(ADC);
}
/*IO*****************************************************************************************
*****digital**IO*/

/*****************************************************************************************
*initDigital                                                                             *
*The pins 4,5,6,7 of portK are connected to the shunts, so they can sense the direction of  *
*the bidirectional levelconverter when initialised as input(DDRK &PORTK). these pins can    *
*also activate an interrupt when the voltage on the pin in changing when                    *
*PCICR(enable interrupt for pins on portK) and PCMSK2(mask for specified pins from port)    *
*are initialised.                                                                           *
*                                                                                          *
*When the K pins are initialised they can be used for setting the right direction:          *
*input or output (PORTH&L).when shunts are gone, microcontroller will be input.             *
*                                                                                          *
*when these pins are set as output they can change the direction of the                     *
*bidirectional levelconverter(this is not programmed                                        *
*                                                                                          *
*PortH and L are used for the digital I/O, ech four pins of the 8 pins are conected bij a   *
*shunt,this will mean they will always be the same direction so initDigital will initialise *
*4pins at a time, four groups with four pins                                               *
*****************************************************************************************/
void initDigital(void){//shunts on 5V=output else input
    DDRK&=0b00001111;
    PORTK&=0b00001111;
    PCICR|=(1<<PCIE2);//interrupt enable interrupt on change for PCINT8-PCINT15
    PCMSK2|=(1<<PCINT20)|(1<<PCINT21)|(1<<PCINT22)|(1<<PCINT23);
    if(PINK&0b00010000){
        DDRL|=0b00001111;
    }
    else{
        DDRL&=0b11110000;
        PORTL&=0b11110000;
    }
    if(PINK&0b00100000){
        DDRH|=(0b00001111);
    }
    else{
        DDRH&=0b11110000;
        PORTH&=0b11110000;
    }
    if(PINK&0b01000000){
        DDRH|=0b11110000;
    }
    else{
        DDRH&=0b00001111;
        PORTH&=0b00001111;
```

```
    }
    if(PINK&0b10000000){
        DDRL|=0b11110000;
    }
    else{
        DDRL&=0b00001111;
        PORTL&=0b00001111;
    }
}
/*************************************************************************************************
* digitalRead will work with the same groups as initDigital, 0b........will mask PIN.,      *
*to devide both ports into two groups,(~DDRH)is a mask so it will only give more            *
*than zero when these pins are input. ch select witch group                                  *
*************************************************************************************************/
int digitalRead(uint8_t ch/*0-3*/){
    if(ch==0){
        return (PINH&0b00001111&(~DDRH));
    }
    if(ch==1){
        return ((PINH&0b11110000&(~DDRH))>>4);
    }
    if(ch==2){
        return (PINL&0b00001111&(~DDRL));
    }
    if(ch==3){
        return ((PINL&0b11110000&(~DDRL))>>4);
    }
}
/*************************************************************************************************
*digitalWrite will work with the same groups as initDigital, 0b........will mask PIN.,      *
*to devide both ports into two groups.  ((DDRH&0b00001111)==0b00001111) will make shure,     *
*when initialised as input, it will not change anything                                      *
*************************************************************************************************/
void digitalWrite(uint8_t ch/*0-3*/,uint8_t data/*0b00000000-0b00001111*/){
    if((ch==0)&&((DDRH&0b00001111)==0b00001111)){
        PORTH|=(data&0b00001111);
        PORTH&=((data&0b00001111)|0b11110000);
    }
    if((ch==1)&&((DDRH&0b11110000)==0b11110000)){
        PORTH|=((data&0b00001111)<<4);
        PORTH&=(((data&0b00001111)<<4)|0b00001111);
    }
    if((ch==2)&&((DDRL&0b00001111)==0b00001111)){
        PORTL|=(data&0b00001111);
        PORTL&=((data&0b00001111)|0b11110000);
    }
    if((ch==3)&&((DDRL&0b11110000)==0b11110000)){
        PORTL|=((data&0b00001111)<<4);
        PORTL&=(((data&0b00001111)<<4)|0b00001111);
    }
}


/*ISR*************************************************************************************************
*****************************************ISR**/

/*DAC*************************************************************************************/
```

C:\Users\joerf\Downloads\neuroplatform (1).c zondag 10 maart 2013 14:24

```
/*********************************************************************************************
*ISR(TWI_vect), this inerrupt will be activated when I2C communication is possible.         *
*TOTAAL must be twice the size +1 of the input area becourse, the first register is used to *
*initialise the communication with the IP adress of the DAC,after that it takes two         *
*registers to sent one sample. the data will set in register TWDR,(1<<TWINT)|(1<<TWEN)      *
*will sent the sample and TWIE will enable the interrupt for the next data.                 *
*the sending itself will not cost any cpu time, the cpu can be used for other things        *
*********************************************************************************************/
ISR(TWI_vect){
        if(i<=TOTAAL){
        TWDR=data[i];//PUT INFORMATION THAT MUST BE SENT IN TWDR REGISTER
        TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWIE);//TWINT&TWEN->SEND TWDR, TWIE->INTERRUPT ENABLE
        i++;
        }
        else{
            TWCR|=(1<<TWSTO);
            i=0;
            }
}

/*********************************************************************************************
*ISR(TIMER1_COMPA_vect) will be activated when TCNT1>=OCR1A.                                 *
*data is a pointer to the active signal,data2 the pointer to the area that will be the next *
*active signal.when ready is one the changeInputDAC can inport a new signal.                *
*TWCR will start the communication with I2C                                                 *
*********************************************************************************************/
ISR(TIMER1_COMPA_vect){
    data=data2;
    //ready=1;
    //i=0;
    TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)|(1<<TWIE);//start communication
}

/*extern interrupt*****************************************************************/
/*********************************************************************************************
*This will be activated when EXT interrupt is initialised and the pin becames high          *
*********************************************************************************************/
ISR(INT2_vect)
{
data2=outputINT2;   //this will change the next form of the DAC after the trigger on PD2,
                    //when you want to do something els, change the code on this place
}
/*********************************************************************************************
*This will be activated when EXT interrupt is initialised and the pin becames high          *
*********************************************************************************************/
ISR(INT3_vect){
data2=outputINT3;   //this will change the next form of the DAC after the trigger on PD3,
                    //when you want to do something els, change the code on this place
}
/*********************************************************************************************
*PCINT1 will be activated when the shunts change,                                           *
*so the digital output/input will also change, this is a safety and is probebly not good    *
*enaf to change the shunts when the program is running, also there is no safety for the     *
*front-end, so by the front-end can shortcut                                                *
*********************************************************************************************/
ISR(PCINT2_vect){
if(PINK&0b00010000){
```

```
        DDRL|=0b00001111;
    }
    else{
        DDRL&=0b11110000;
        PORTL&=0b11110000;
    }
    if(PINK&0b00100000){
        DDRH|=(0b00001111);
    }
    else{
        DDRH&=0b11110000;
        PORTH&=0b11110000;
    }
    if(PINK&0b01000000){
        DDRH|=0b11110000;
    }
    else{
        DDRH&=0b00001111;
        PORTH&=0b00001111;
    }
    if(PINK&0b10000000){
        DDRL|=0b11110000;
    }
    else{
        DDRL&=0b00001111;
        PORTL&=0b00001111;
    }
}
```