

Styresystem for kybernetisk h ndledd

Geir Turtum

11. desember 2013

Oppgavetekst

Instituttet har sammen med University of New Brunswick, Canada, stått sentralt i utviklingen av et motorisert protesehåndledd med unike kinematiske egenskaper. Det foreligger et maskinvaredesign og en prototyp av styresystemet, og dette systemet skal nå ferdigstilles og programmeres slik at protesen kan brukes i forskningssammenheng. Sentrale systemegenskaper inkluderer kommunikasjon over en CAN-buss ved hjelp av protokollen PDCP, noe signalbehandling og styring av en børsteløs DC-motor.

1. Gi en kort oversikt over prosjektets bakgrunn og nåværende status, der du legger vekt på systemets tilstand relativt dets spesifikasjoner og tiltenkte anvendelse.
2. Foreliggende HW-design er basert på en enkelt mikrokontroller som skal stå for både motorkommutering, regulering, signalbehandling og kommunikasjonsprotokoll(er). Foreta en vurdering av om dette designet er egnet, med spesiell vekt på avbruddshåndtering og sanntidsaspekter. Gjør om nødvendig praktiske målinger for å underbygge din konklusjon.
3. Foreslå på bakgrunn av punkt 2 en maskin- og programvarearkitektur for systemet. Eksisterende løsninger bør gjenbrukes i den grad det er mulig og hensiktsmessig.
4. Implementer og test systemet så langt tiden tillater det.

Sammendrag

Denne prosjektoppgaven tar for seg design og utvikling av styresystemet til håndleddsprotesen NRWD(NTNU Rotary Wrist Device). NRWD ble i sin tid skapt på bakgrunn av doktorgradsavhandlingen[2] til Øyvind Stavdahl.

Det har blitt skrevet flere prosjekt- og masteroppgaver[6, 3] om NRWDen hvor alle har hatt som mål å realisere en prototype av protesen, men på grunn av ulike årsaker ikke har fullført. Denne oppgaven baserer seg i hovedsak på arbeidet til Andreas Kråkenes [3]. Han har produsert maskinvare lagt ut på flerlagskort, samt skrevet programvare for testing av maskinvaremodulene, noe som la et meget godt grunnlag for videre arbeide.

I denne prosjektoppgaven har styresystemet blitt ferdigstilt og det har blitt gjort anbefalinger til endringer i maskinvare der man har sett muligheter for forbedringer.

Kommunikasjonen med protesen går over standard analoge og digitale elektroder og med den åpne protokollen PDCP(Prosthetic Device Communication Protocol). Implementeringen av protokollen er gjort av tidligere NTNU studenter[4, 5] mens undertegnede har stått for integreringen av protokollen med protesens styresystem.

Forord

Jeg vil takke katten min Tom.

Nomenklatur

- NRWD - NTNU Rotary Wrist Device
- PDCP - Prosthetic Device Communication Protocol
- CAN - Controller Area Network
- PWM - Pulse Width Modulation
- UNB - University of New Brunswick
- ISR - Interrupt Service Routine
- LDO - Low Dropout Regulator
- BLDC motor - Brushless DC electric motor

Innhold

1	Introduksjon	7
1.1	Bakgrunn for oppgaven	7
1.2	Tidligere arbeid	8
1.3	Mine bidrag	8
1.4	Utviklingsmetodikk og fremgangsmåte	9
1.5	Rapportens oppbygning	10
2	NRWD Spesifikasjoner	11
2.1	Wrist Joint Function, WJF	11
2.2	Wrist Motor Function, WMF	12
2.3	Wrist Servo Function, WSF	12
2.4	Wrist Communication Function, WCF	14
2.5	Wrist Power Function, WPF	15
2.6	Proximal Attachment Function, PAF	15
2.7	Distal Attachment Function, DAF	16
3	Mekanikken	18
4	Interrupt diskusjon	20
4.1	Bakgrunn	20
4.2	Et interrupt drevet system	20
5	Maskinvaretesting og diskusjon	23
5.1	Mikrokontroller	23
5.2	Spenningsforsyning	24
5.3	Posisjonsmåling	24
5.4	Motordriver	26
5.5	Strømovertvåking	27
5.6	CAN	28
5.7	Diverse	28

6	Programvare implementasjon	29
6.1	Motor	29
6.2	Current	30
6.3	Position	30
6.4	Controller	30
6.5	Analog	32
6.6	PDCP and can	32
6.7	Main	33
7	Diskusjon	35
8	Konklusjon	36
A	Innhold CD	38
B	Funksjonsspesifikasjon NRWD	39
C	Diverse	40

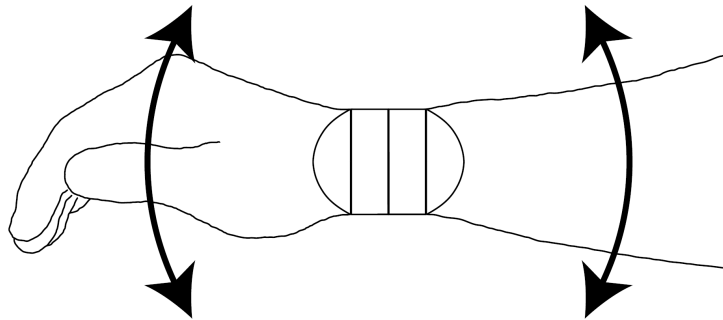
Kapittel 1

Introduksjon

1.1 Bakgrunn for oppgaven

Ideen bak protesen kommer fra Øyvind Stavdahls doktorgradsavhandling [2] fra 2002. I den viste eksperimenter med friske h ndledd at en vinkel mellom lengdeaksen til underarm og rotasjonsaksen til h ndleddet gjorde flere bevegelser enklere for brukeren   utf re.

For   teste disse resultatene i praksis trenger man en 1-akse h ndleddsprotese med design som muliggj r valgfri rotasjonsakse, se figur 1.1. Dette ble bygget som en del av mastergraden til Arthur Zink[1] ved University of New Brunswick, Canada. Etter at mekanikken var ferdigstilt trengte man et styresystem for   styre protesen.



Figur 1.1: Vinkel mellom h ndprotese og underarm kan varieres. Kilde: Styresystem for kybernetisk h ndleddsprotese[3]

Det ble ogs  funnet ut at dette var en god anledning til   teste ut den  pne protesestyringsprotokollen PDCP, som da var under utvikling hos UNB. Som krav i funksjonsspesifikasjonen til NRWDen ble det satt at den skulle

kommunisere ved hjelp av denne protokollen. I tillegg skulle den kunne styres med de mer konvensjonelle analoge og digitale elektrodene.

1.2 Tidligere arbeid

Realisering av styresystemet til protesen har hittil vært påbegynt i to mastergradoppgaver. Den siste av disse var skrevet av Andreas Kråkenes i 2011[3] kom langt på vei til å fullføre designet. Hovedgrunnen til at protesen enda ikke har blitt fullført skyldes at det første designet innholdt en alvorlig arkitekturfeil som hindret kommunikasjonen mellom de to mikrokontrollerne. Det ble forsøkt ordnet. Det endte med at man gikk man over til en løsning hvor en mikrokontroller skulle styre hele systemet.

Da prosjektet ble overtatt av undertegnede, var alt av hardware designet og lagt ut på prototypekort. Programvare for å teste deler av systemet var ferdigskrevet, men koden var av varierende kvalitet og mye funksjonalitet var ikke implementert. Flest mangler var knyttet til de ulike regulatorene og implementasjonen av PDCP protokollen.

To mastergradsstudenter ved NTNU[4, 5] laget i 2012 en implementasjon av PDCP-protokollen. UNB hadde tidligere laget en implementasjon basert på PIC mikrokontrolleren, mens denne var designet for ATMELs AVR mikrokontrollere. Implementasjonen var designet for at det skulle være enkelt å overføre koden til ny maskinvare, noe som gjorde at den egnet seg godt for dette prosjektet.

1.3 Mine bidrag

Prosjektet ble startet med gå gjennom den overleverte maskinvaren og medfølgende driverkode modul for modul. Detaljer rundt hvordan de ulike modulene ble testet er beskrevet i kapittel 5.

Den overleverte koden var av varierende kvalitet. Koden var strukturert på et fornuftig vis, med en c-fil for hver maskinvare modul. Denne strukturen ble beholdt ut prosjektet. Selve driverkoden trengte en del mer arbeid. Det var skrevet nok til å teste basisfunksjonaliteten til hver modul, men måtte utvides med tilleggfunksjonalitet før den kunne brukes.

Under testing av maskinvaren ble det funnet muligheter for forbedringer i spenningsregulator og posisjonssensormodulene. Kapittel 5 oppsummerer anbefalte endringer i maskinvaren.

Det første styresystemet som ble designet til NRWDen benyttet seg av to mikrokontrollere for å håndtere alle oppgavene. Denne arkitekturen gikk

man bort fra da man fikk problemer med kommunikasjonen mellom de to kontrollerne og man gikk over til å bruke en mikrokontroller. Som en del av oppgaven ble det gjort grundige undersøkelser om den valgte mikrokontrolleren hadde nok ressurser til å styre systemet på egen hånd. Diskusjonen rundt dette er gjort i kapittel 4.

Ifølge funksjonsspesifikasjonene skal protesen ha minst tre forskjellige styringsmoduser, maks fart i valgt retning, posisjons- og hastighetsstyring. Koden til de forskjellige regulatorene ble alle skrevet fra bunnen av. For detaljer om hvordan de ulike regulatorene ble implementert, se kapittel 6.

På slutten av prosjektet ble styringssystemet integrert med kommunikasjonsprotokollen PDCP. Implementasjonen av protokollen var allerede gjort, jobben besto av å skrive om maskinvare abstraksjonslaget. Hvorfor og hvordan dette ble gjort, er beskrevet i kapittel 6.

1.4 Utviklingsmetodikk og fremgangsmåte

Hvordan dette prosjektet best skulle styres falt klart for meg fra begynnelsen av. Det er et maskin- og programvare system som allerede da det ble overtatt var delt opp i klare moduler. Som nevnt tidligere var det meste av maskinvare designet ferdig, mens den gjensto å ferdigstille programvaren. Det var også mye programvare som var skrevet av tidligere studenter som skulle flettes inn i prosjektet for ikke å reproducere arbeide som var gjort tidligere. Det tenkes spesielt på lavnivå implementasjonen av PDCP protokollen som ble utviklet av Andrzej Zamojski [5] og Andreas Nordal[4] samt programvare driverne til NRWDen skrevet av Andreas Kråkenes.

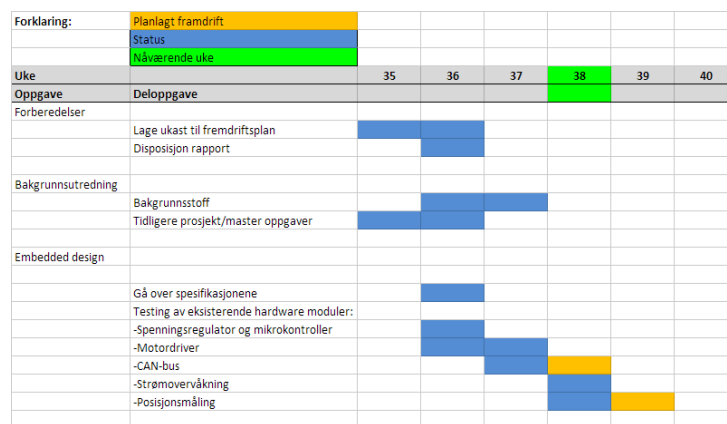
Metoden min sin hovedlinje var i størst grad arbeide på prosjektet modul for modul, forsikre meg om at alt fungerte atskilt, for til slutt å sette modulene sammen til et komplett styresystem. Kort oppsummert var planen:

- Sette seg inn i bakgrunnsstoff for oppgaven.
- Test hver maskinvare modul for seg, samt tilhørende programvare driver.
- Sjekk om maskinvare og programvare oppfyller kravene satt av funksjonsspesifikasjonen.
- Fullfør driverkoden til modulen som blir testet.
- Sett seg inn i implementasjonen av PDCP gjort av tidligere NTNU studenter.

- Integrere PDCP med styringssystemet.

Hvordan hver modul ble testet blir gått gjennom i kapittel 5.

Verktøyet som ble brukt for å budsjettere tiden og sørge for at man kom i mål med prosjektet til riktig tid var, på anbefaling fra veileder, Gantt diagrammet. Figur 1.2 viser framgangsplanen for de første ukene av prosjektet og hvorvidt planen ble overholdt. Gantt diagrammet var et nyttig verktøy å bruke, ikke bare ga det en klar oversikt over hvilke oppgaver man hadde igjen, men også hvorvidt man greide å overholde de tidsfrister man hadde satt.



Figur 1.2: Gantt diagram

Utviklingen av programvare ble gjort på en windows maskin med Atmel Studio 6. For versjonskontroll av kode og dokumenter falt valget på Git.

1.5 Rapportens oppbygning

Skrive om hva som kommer når?

Kapittel 2

NRWD Spesifikasjoner

Dette kapitlet går gjennom funksjonsspesifikasjonene til NRWDen punkt for punkt. Spesifikasjonene beskriver kravene til systemet i sin hellhet fra de fysiske dimensjonene, kommunikasjons grensesnittet opp mot protesen, oppløsningen på de ulike sensorene, maks vinkel hastighet til protesen og så videre.

Delkapittlene som følger tar for seg hver sin del av funksjonsspesifikasjonene. De beskriver hvilke av kravene som er oppfylt, de som enda ikke er implementert og de som krever mer arbeid. Dette kapitlet beskriver systemet i sin nåværende tilstand og er ment for å hjelpe neste person som skal arbeide på prosjektet til å få en rask oversikt over hva som er gjort og hva som må gjøres. For noen av kravene vil statusen være kommentert som "Ikke relevant", dette gjelder de spesifikasjonene som omhandler de rent mekaniske sidene av systemet.

Ikke tatt med i dette kapitlet er de overordente funksjonsspesifikasjonene til NRWDen. De kan leses i det originale spesifikasjons dokumentet vedlagt denne oppgaven, se vedlegg B.

2.1 Wrist Joint Function, WJF

Beskriver det roterende leddet i protesen og hvordan festeanordningen til underarm og eventuell håndleddsprotese skal være manuelt justerbar.

Nr	Beskrivelse	Status
WJF-01	The NRWD joint shall be a single, simple revolute joint which axis of rotation can be placed at an attitude with respect to the forearm and terminal device as specified in [1], Equations (7.4) and (7.5).	Ikke relevant
WJF-02	The joint axis should be manually adjustable to other attitudes than that specified in WJF-01.	Ikke relevant
WJF-03	The joint shall enable an angular excursion of at least 180 degrees. The excursion should be unlimited.	Ikke relevant

2.2 Wrist Motor Function, WMF

Beskriver kravene til protesens vinkelhastighet og moment. Viktig å merke seg at det her settes krav til at motoren skal være beskyttet mot overopphetning i maskinvare.

Nr	Beskrivelse	Status
WMF-01	The wrist joint (output of the gear train) shall have a maximum angular velocity of at least 1.4 rad/s (81 deg/s). The maximum velocity should be as high as possible.	Høyeste målte vinkel hastighet er målt til 2.72 rad/s (156 grader/sekund)
WMF-02	The wrist joint should have a maximum torque of at least 34,3 mNm.	Ikke målt, men ifølge målinger gjort av Zink [1] er stall torque på 59.6 mNm.
WMF-03	The motor shall have a maximum mechanical output power of at least TBC W.	NA
WMF-04	The motor shall be protected from overheating. The protection should be implemented by hardware.	Dette har blitt implementert i strømovertvåkingsmodulen.

2.3 Wrist Servo Function, WSF

Beskriver kravet til posisjonssensoren og hvilke regulatorer som systemet skal implementere.

Nr	Beskrivelse	Status
WSF-01	The movements of the wrist joint shall be controllable according to the following modes: <ul style="list-style-type: none"> • On/Off-mode • Position mode • Velocity mode 	Alle kontrollerne er Implementert i controller.c
WSF-01-01	In On/Off-mode the motor shall be at rest or run at maximum speed (open-loop) in one direction according to a given setpoint.	Implementert ved bruk av PI-kontroller som kjører motoren nærmest mulig maksimum tillatte strømforbruk.
WSF-01-02	In position mode the joint angle shall be proportional to a given angular setpoint.	Implementert med PI-kontroller.
WSF-01-02-01	The WSF shall include an absolute position sensor. This sensor shall provide no less than 10 bits resolution per revolution.	En 12-bits magnetisk enkoder er brukt.
WSF-01-03	In velocity mode the joint angular velocity shall be crudely proportional to a given velocity setpoint.	Implementert med PI-kontroller.
WSF-01-03-01	The WSF shall include a velocity sensor or estimator.	Hastighet er estimert ved hjelp av hall elementene.
WSF-02	The movements of the wrist joint should be controllable according to the following modes: <ul style="list-style-type: none"> • Torque mode • Impedance mode 	Ikke implementert.
WSF-02-04	In torque mode the motor torque shall be proportional to a given torque setpoint.	Ikke implementert.
WSF-02-04-01	The WSF should include means for monitoring motor current.	Implementert, se strømovertvåkningsmodulen.
WSF-02-05	In impedance mode the mechanical impedance of the joint shall be determined by a given impedance setpoint	Ikke implementert.
WSF-03	WSF shall provide an ¹³ interface to WCF through which the modes (described in WFS-01 to WFS-02) can be selected and relevant parameters can be set, and through which WSF can report relevant state variables TBD.	Koden er lagt opp for at dette enkelt kan bli implementert når kommunikasjons spesifikasjonen er klarlagt.

2.4 Wrist Communication Function, WCF

Beskriver hva slags kommunikasjonsgrensesnitt systemt skal implementere.

Nr	Beskrivelse	Status
WCF-01	The NRWD shall have a two-wire Proximal Communication Interface (PCI) and a two-wire Distal Communication Interface (DCI).	Implementert ved CAN-bus.
WCF-02	The PCI shall be configurable so that it implements two (0V, 7,2V) analog input lines or a bidirectional twowire CAN interface with the PDCP protocol(Losier,2009).	Implementert ved CAN-bus og PDCP-protokollen.
WCF-02-01	The analog input lines shall be able to sample both lines at a rate of 1 kHz. The sampling rate should be as high as 2 kHz.	Implementert i analog.c
WCF-03	The DCI shall be configurable so that it implements two (0 V, 7.2 V) analog output lines or a bidirectional two-wire CAN interface with the PDCP-protocol.	Implementert ved CAN-bus og PDCP-protokollen. Implementasjonen av PDCP-protokollen er ikke ferdig.
WCF-04	The WCF shall be configurable to an all-Analog mode, with the PCI as an analog input interface and the DCI as an analog output interface.	Analog output interface ikke implementert i verken HW eller SW.
WCF-05	The WCF shall be configurable to an all-digital mode, with the PCI and the DCI acting as bidirectional CAN bus interfaces.	Implementert. CAN buss trenger bare et tilkoplingspunkt.
WCF-07	The WCF should be configurable to a hybrid mode in which the PCI acts as two analog input lines while DCI acts as a bidirectional CAN interface (cf. WCF-01)	Ikke implementert i HW.
WCF-10	The WCF shall include a serial interface for downloading software and for debugging/diagnostic purposes.	Implementert ved bruk av JTAG.
WCF-11	WCF shall implement an interface to WSF according to WSF-03.	Se WSF-03.

2.5 Wrist Power Function, WPF

Beskriver kraftforsyningen til systemet.

Nr	Beskrivelse	Status
WPF-01	The WPF shall accept external power in the form of an unregulated two-wire DC supply.	Implementert ved brukt av intern spenningsregulator.
WPF-02	The NRWD shall tolerate and run normally when powered with a voltage in the range (6 V, 12 V). The range of usable voltages should be as wide as (5 V, 18 V).	Valgte spenningsregulator opererer normalt innenfor området (7V, 18V). Det er anbefalt å gå over til en Low Droupout Regulator for å tilfredstille kravet.
WPF-03	The NRWD shall tolerate supply voltage in the range (0 V, 12 V) without exhibiting unpredictable behaviour and without getting damaged.	Implementert ved bruk av brown out deteksjon internt på AVRn. [todo]
WPF-04	The NRWD should automatically limit its motor current to a level that does not reduce the supply voltage below the interval given in WPF-01.	Ikke implementert.

2.6 Proximal Attachment Function, PAF

Beskriver festeanordningen og grensesnittet mellom NRWDen og underarmen.

Nr	Beskrivelse	Status
PAF-01	The PAF shall comprise two parts, the proximal of which is adapted to be permanently attached to the forearm socket and the distal permanently attached to the wrist unit. The parts must 'mate' to form a mechanically stable connection while also being detachable.	Ikke relevant
PAF-01-01	The proximal part of the PAF shall be hollow to allow access to the space within the socket proximally to the wrist.	Ikke relevant
PAF-02	PAF disconnection should be possible with hand or a simple tool, e.g. a screwdriver.	Ikke relevant
PAF-03	The PAF shall include a four-wire electric coupling, preferably mechanically integrated with the PAF itself.	Ikke implementert i prototype hardwaren.
PAF-03-01	The PAF electrical coupling shall include at least two power supply wires/contacts capable of transferring a constant current of 4 A per wire.	Ikke implementert i prototype hardwaren.

2.7 Distal Attachment Function, DAF

Beskriver festeanordningen og grensesnittet mellom NRWDen og en eventuell håndprotese.

Nr	Beskrivelse	Status
DAF-01	The DAF should comprise two parts, the proximal of which is permanently attached to the wrist and the distal permanently attached to the terminal device. The parts must mate to form a mechanically stable connection while also being detachable.	Ikke relevant
DAF-02	The DAF shall include a four-wire electric coupling, preferably mechanically integrated with the DAF itself.	Ikke implementert i prototype hardwaren.
DAF-02-01	The DAF electrical coupling shall include at least two wires/contacts capable of transferring a constant current of 2 A per wire, and these wires shall be connected to the power supply wires from PAF.	Ikke implementert i prototype hardwaren.
DAF-02-02	The DAF electrical coupling should be rotatable without twisting the wires.	Ikke implementert i prototype hardwaren.

Kapittel 3

Mekanikken

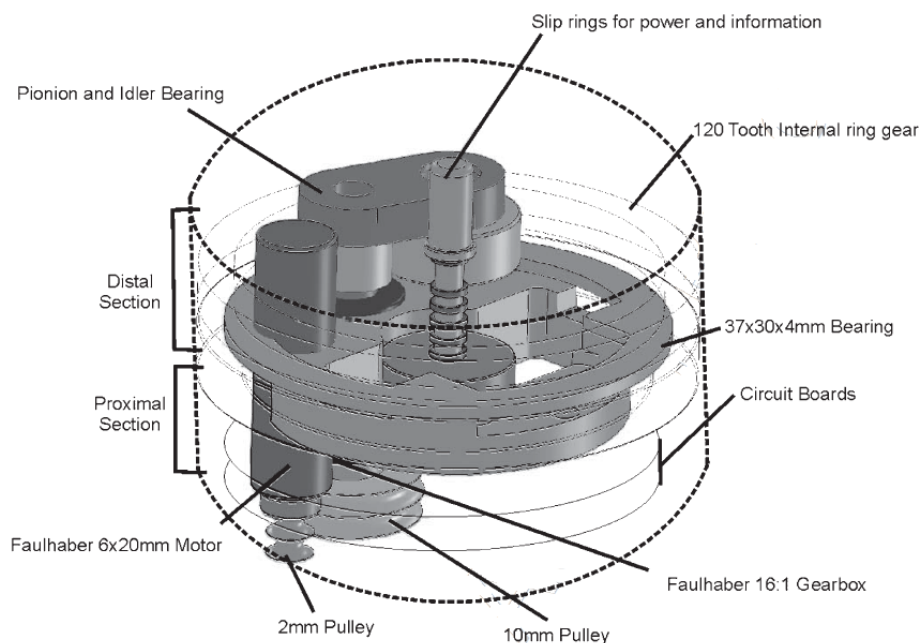
Mekanikken i protesen er et resultat av masteroppgave skrevet ved universitetet i New Brunswick, Canada. Dette ble gjort på oppdrag fra Øyvind Stavdahl for å teste resultatene fra sin doktorgradsavhandling i praksis. Mekanikken består av to seksjoner, den proksimale og distale (henholdsvis seksjonen nærmest underarmen og den lengst unna). De to seksjonene kan rotere fritt mot hverandre ved hjelp av et kulelager. Selve bevegelsen er drevet av en børsteløs DC-motor som er koplet til et girtog med utveksling 120:1 [1].

Under testing av motordriveren ble det funnet at det, ved en forsyningspenning på 12V, måtte til en dutycycle på oppmot 60% for å i det hele tatt bevege mekanikken. Ved rundt 80% dutycycle ble strømmen til motoren kuttet av strømovervåkingsmodulen da strømmen gjennom motoren oversteg maksimum grensen. Denne grensen er satt i maskinvare for å unngå at motoren blir ødelagt på grunn av overopphetning. Dette ga regulatoren et veldig lite rom å jobbe innenfor noe som gjorde implementasjonen av posisjon og hastighetskontrolleren meget vanskelig.

Mekanikken ble tatt med til mekaniske verksted ved NTNU for å se om de kunne løse problemet. Det viste seg at aksen til det minste tannhjulet var bøyd noe som førte til ekstra friksjon i systemet. Tannhjulet det er snakk om er det som er koblet til Faulhaber girboksen i den distale seksjonen, se figur 3.1. Verkstedet fikk bøyd det til slik at det gikk fra å ha en kast på 0.22 mm til 0.03 mm. Dette reduserte friksjonen betraktelig og mekanikken kunne nå beveges med en dutycycle ned mot 40%.

Da mekanikken ble undersøkt ble det også funnet at kulelageret i protesen begynte å bli slitt noe som førte til ytterligere friksjon i systemet.

Om rettingen av giraksen løser problemet på langt sikt vil vise seg med tiden. At problemet i det hele tatt har oppstått tyder på at mekanikken ikke er riktig dimensjonert for de krefter den kan bli utsatt for. En teori på hvordan problemet kan ha oppstått er at man har vridd på den distale



Figur 3.1: Oversikt bilde over den indre mekanikken til protesen. Figur hentet fra [1]

seksjonen, noe som påfører store krefter på den indre mekanikken. Dette tyder på at mekanikken kan ha behov for en ny revisjon før den er robust nok til å kunne brukes i eksperimenter.

For fremtidig bruk er det lagt ved en oversikt over rotasjonshastigheten til protesen ved forskjellige duty cycles etter at mekanikken var rettet av verkstedet, se vedlegg C.1. Denne kan brukes for å se om problemet har gjenoppstått.

Kapittel 4

Interrupt diskusjon

4.1 Bakgrunn

Tidligere forsøk på å realisere styresystemet til NRWDen har brukt to mikrokontrollere. Den ene skulle brukes til motor kommuteringen mens den andre skulle håndtere alle de andre oppgavene som regulering og kommunikasjon. Da systemet ble testet oppsto det problemer med kommunikasjonen mellom mikrokontrollerene og selv etter mye testing fant man aldri årsaken til problemet. Dette førte til at man gikk over til en arkitektur som brukte en enkelt mikrokontroller.

Spørsmålet var da om den valgte mikrokontrolleren, AT90CAN128 kjørende på 16 MHz, var nok til å håndtere alle oppgaver som før skulle blitt utført av to mikrokontrollere. Dette kapitlet tar for seg denne problemstillingen.

4.2 Et interrupt drevet system

Så hva slags type system er det som styrer protesen? Styringssystemet er nesten helt og holdent interrupt drevet. Bortsett fra regulatoren som blir kjørt inni main while loopen er alt annet styrt med interrupts. Motorkommuteringen, kommunikasjonsprotokollen, fartsestimeringen, posisjonsavlesningen og strømovervåkingen blir alle styrt gjennom interrupts. Hver av disse funksjonene krever at tiden mellom et interrupt blir generert og mikrokontrolleren går inn i tilhørende ISR(Interrupt Service Routine) ikke overstiger en viss lengde.

Om et interrupt ikke blir håndtert innenfor tidsfristen en eller flere ganger vil systemet i beste fall bli mindre nøyaktig, i verste fall vil det slutte å fungere. Som et eksempel kan man ta for seg posisjonssensoren. Posisjonen er proporsjonal med dutycyclelen til sensor signalet. Denne leses av ved å

måle tiden mellom positiv og negativ flanke på signalet, hver endring i nivå på signalet genererer et interrupt. Hvis tiden det tar for mikrokontrolleren å behandle interruptet tar lengere tid enn forventet vil posisjonen bli feil avlest.

Som nevnt ovenfor, om ikke systemet har nok resurser til å håndtere alle interruptene som kommer vil man risikere at man får et system som virker tregt og uresponsivt eller slutter å virke. For å kunne gi en garanti om at dette ikke ville skje ble systemet analysert for tidsbruk i interrupt rutiner.

Typen interrupts som systemet skal håndtere kan deles inn i to typer, periodiske og aperiodiske.

De periodiske er som følger:

- HALL_X_vect - Trigger hver gang en av de tre hall sensorene i motoren skifter nivå
- TIMER3_CAPT_vect - Trigger på hver flanke til PWM signalet fra posisjons sensoren
- TIMER0_OVF_vect - Brukes til å beregne hastigheten til motoren

De aperiodiske er:

- CANIT_vect - Trigger hver gang det kommer en CAN melding
- CURRENT_vect - Trigger når maskinvaren slår av strømmen til motoren
- RESET

For å analysere systemet ble varigheten til hver interrupt målt. Et interrupt i mikrokontrolleren består av tre deler. Det er kontekstskiftet hvor mikrokontrolleren lagrer unna stacken før man går inn i ISR'en, arbeidet som blir utført inni rutinen og gjenoppbyggingen av stacken i det man forlater den. Tiden det tar for mikrokontrolleren å utføre de to kontekstskiftene er et fast antall instruksjoner, det faktiske tallet finnes i mikrokontrollerens datablad [8]. Varigheten til hvert interrupt ble målt ved å sette en GPIO-pinne høy ved inngangen av interruptet og lav i det den gikk ut og lese av opptiden på oscilloskopet. Se tabell 4.1 for oppsummering av tidsbruk til de periodisk interruptene.

Vi ser av tabell 4.1 at mikrokontrolleren bruker mindre enn 2% av tiden inni en interrupt rutine. Det gir rikelig med tid til å håndtere kontrolloopen som kjører utenfor interruptene. Derimot så garanterer det ikke at alle interrupts vil bli håndtert innenfor rimelig tid. I et tilfelle hvor en eller flere av

Navn	Varighet (μs)	Maks frekvens	Tidsbruk per sekund(μs)
HALL_X_vect	4.5	3600	16200
TIMER3_CAPT_vect	4.8	490	2352
TIMER0_OVF_vect	3.2	61	196
Totalt			18742

Tabell 4.1: Viser varigheten til hvert interrupt, hvor mange ganger de blir kjørt i sekundet og totalt tidsbruk per sekund.

modulene genererer et interrupt samtidig kan dette føre til midlertidige feil i systemet. Siden dette ikke er et hard sanntidssystem, hvor overskridelsen av en tidsfrist hadde knekt systemet, er disse midlertidige feilene akseptable siden med alle de forskjellige ISRene vil problemet være løst ved neste interrupt. Da ISRene opptar under 2% av prosessor syklene og de alle operere ved ulike frekvenser vil kolisjon mellom to eller flere ISR være et sjeldent fenomen.

Det ble også undersøkt om mikrokontrolleren er rask nok til å håndtere kommuteringen av motoren. En den for treig vil den ikke kunne kommutere raskt nok til at motoren oppnår den ønskede hastighet. ATMELs applikasjonsnote AVR452 som tar for seg motor styring av børste løse DC motorer ved bruk av AT90CAN mikrokontrolleren. Den setter en teoretisk maksgrense ved 3'478'000 rpm ved 8 MHz, da maksfarten til NRWD motoren er satt til 3600 rpm og mikrokontrolleren kjører på 16 MHz befinner vi oss godt innenfor kravene.

Kapittel 5

Maskinvaretesting og diskusjon

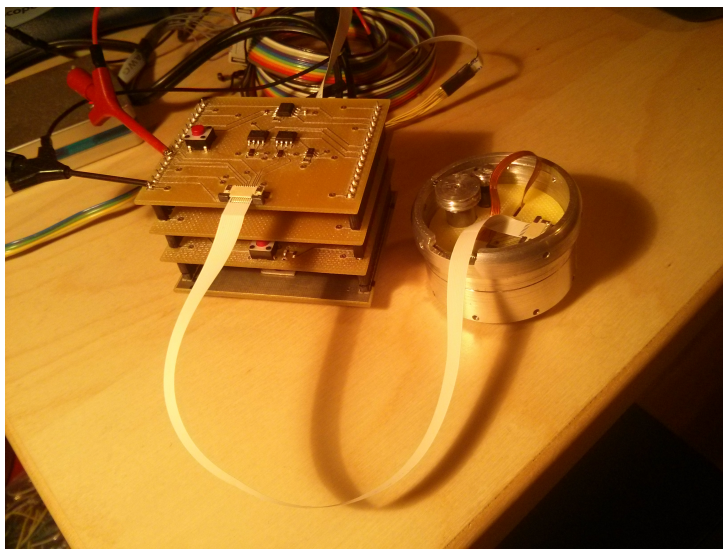
Dette kapitlet går igjennom maskinvaren som dette prosjektet har basert seg på. Det blir forklart hvordan hver modul har blitt testet og eventuelle forslag til forbedringer blir gitt.

Dette prosjektet startet ikke på bar bakke, men bygger videre på arbeidet til flere studenter. Således var det mye dokumentasjon, maskinvare og programvare som måtte gjennomgås før det kunne bygges videre på. Det tenkes her spesielt på arbeidet til Andreas Kråkenes som var siste student som arbeidet på NRWDen. Det var han som sto for utlegg og produksjon av maskinvareprototypen, se figur 5.1, som har blitt brukt under dette prosjektet.

Alt av overlevert maskin- og programvare ble gjennomgått og testet modul for modul for å se om det fungerte i henhold til det som sto i Kråkenes' oppgave [3]. Dokumentasjonen av maskinvaren besto av tekstlige beskrivelser gjort i oppgaven samt vedlagte skjemategninger. Før arbeidet startet ble en kort testplan for hver modul utarbeidet, den ble laget slik at da alt i planen var testet skulle man være sikker på at maskin- og programvaren for valgt modulen oppførte seg som forventet. Under følger resultatene av testingen, samt en diskusjon rundt hver modul. For flere detaljer rundt maskinvaren, se Kråkenes' oppgave[3].

5.1 Mikrokontroller

Styringen av protesen er gjort med en AT90CAN128 ATMEL AVR mikrokontroller. Opplasting av programvare og debugging ble gjort over JTAG via ATMELs programmeringsverktøy, JTAGICE 3. Det ble opprettet kontakt med AVRen og alt fungerte som forventet.



Figur 5.1: Prototypen av NRWD maskinvaren produsert av Andreas Kråkenes, her vist tilkople protese mekanikken.

5.2 Spenningsforsyning

Ifølge funksjonskravet WPF-02 skal NRWDen operere normalt når den blir drevet av en spenningsforsyning mellom 6 og 12V. Spesifikasjonene sier ikke noe om hvilke spenninger som skal brukes internt, men den valgte mikrokontrolleren trenger 4.5V for å kjøre på 16MHz. Det har derfor blitt valgt at den eksterne spenningsforsyningen skal reguleres ned til 5V.

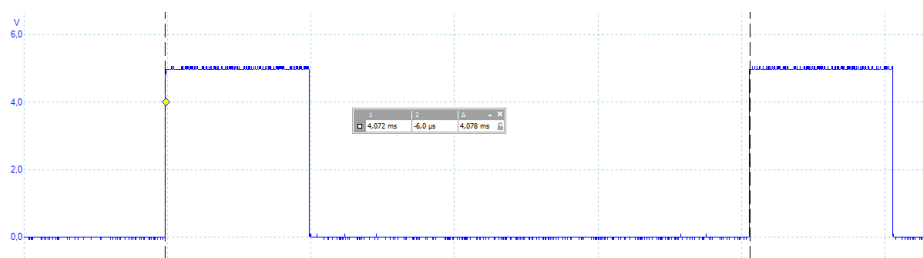
Spenningsregulatoren brukt i NRWDen er en linær regulator som i henhold til databladet gir ut 5V \pm 0.2V ved inngangsspenninger 7-20 V. Det er viktig at NRWDen kan operere på spenninger ned til 6V siden de vanligste batteriene som er brukt i slike proteser gir ut en spenning som kan gå ned til minst 6V. Da differansen mellom inngangs- og utgangsspenning kan være så lav som 1V er anbefales det å gå over til å bruke en LDO (Low Dropout Regulator). En LDO er spesielt designet for å operere under akkurat slike forhold. Et eksempel på en LDO som kan brukes er LF50CDT fra STMicroelectronics.

5.3 Posisjonsmåling

Ifølge funksjonskravet WSF-01-02-01 skal NRWDen være utstyrt med en absolutt posisjonssensor med en oppløsning på minst 10 bit.

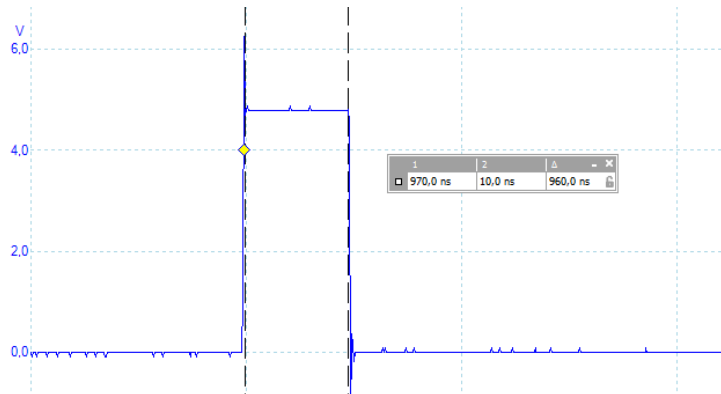
Den nåværende løsningen er en magnetisk enkoder. Den leser av magnet-

feltet til en magnet festet på den distale delen av NRWDen. Sensoren gir ut et PWM signal hvis dutycycle er proporsjonalt med vinkelen til magnetfeltet på sensoren, se figur 5.2. Hver flanke på PWM-signalet generer et interrupt og tiden mellom en stigende og synkende flanke blir målt av mikrokontrolleren.



Figur 5.2: Utgangen til posisjonssensoren målt med oscilloskop

Problemet med dette oppsettet kommer ved vinkler nær 0 og 360 grader. Ved disse vinklene vil tiden mellom to flankeendringer være under $1 \mu\text{s}$, se figur 5.3. Som beskrevet i kapittel 4 tar dette interruptet $4.8 \mu\text{s}$ å fullføre. Dette vil si at det andre interruptet som kommer enten ikke vil bli behandlet eller bli behandlet for sent noe som vil gi feil avlest vinkel.



Figur 5.3: Utgangen til posisjonssensoren ved lavest dutycycle

Forløpig er problemet løst i programvare, men en bedre løsning ville vært å lavpassfiltrere PWM-signalet for så å lese av vinkelen med en ADC. Dette løser ikke bare problemet med feil avlesning av vinkel, men det vil også redusere antall interrupts som mikrokontrolleren må behandle. PWM-signalet har en frekvens på 245 Hz, noe som gir 490 interrupts i sekundet. Ved bruk av et lavpassfilter og en ADC vil man bare trenge å kalkulere posisjonen hver gang posisjonsregulatoren skal kjøre. GPIO pinnene på mikrokontrolleren har

allerede mulighet for å aktivere en intern pullup motstand noe som betyr at den eneste eksterne komponenten som kreves for å lage et lavpassfilter er en kondensator med passende verdi.

5.4 Motordriver

Protesens bevegelse er drevet av en børsteløs DC-motor med innebygde Hall sensorer. Ved å lese av Hall sensorene vet man posisjonen til motoraksen og kan utifra det vite hvilket pådragsmønster man skal gi motoren får å få den til å rotere. Utgangen til Hall sensorene er digitale og er koplet opp mot hver sin interrupt linje slik at det blir generert et interrupt hver gang utgangen skifter nivå.

Ifølge funksjonskravet WMF-01 skal protesen minst ha en maksimums omdreiningshastighet på 81 grader/sek. Ifølge Zink[1] er motoren giret ned med en ratio på 800:1, se tabell 5.4. Dette stemmer ikke med protesen brukt i dette prosjektet da diameterene til drivhjulene fra motor til Faulhaber planetary gear ikke er som spesifisert hos Zink. Faktiske målinger viser at den totale gir ration til protesen er på 333:1.

Details of the drive train					
Stage	Stage gear	Cumulative ratio	Velocity after each stage (rpm)	Torque after each stage (mNm)	Notes
ratio					
Motor	1:1	1:1	47000	0.73	Faulhaber 0620 K 006 B
First	5:1	5:1	9400	3.65	Belt
Second	16:1	80:1	587.5	58.4	Faulhaber 08/1K - Planetary Gear
Third	10:1	800:1	58.8	584.0	Ring and pinion 303 stainless steel Pitch: 96 Face width: 3.175 mm Pressure angle: 200

Figur 5.4: Tabellen viser girtoget til motoren slik det originalt var ment, stemmer IKKE med den mekanikken brukt i dette prosjektet. Tabell hentet fra Zink, Design of a compact, reconfigurable, prosthetic wrist[1].

For å oppnå ønsket vinkelhastighet tilsvarer dette en motorhastighet på 4496 rpm, se formel 5.1. Ifølge motordatabladet [9] er makshastigheten til motoren 35600 rpm noe som faller godt innenfor kravet.

$$\frac{81 \text{ grader/sekund}}{360 \text{ grader}} \times 333 \times 60 \text{ sekund} = 4495.5 \text{ rpm} \quad (5.1)$$

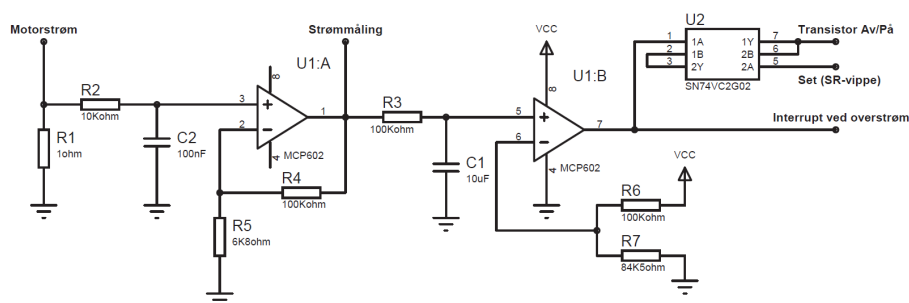
Da motorens hall sensorer genererer 6 interrupt per omdreining vil mikrokontrolleren måtte kunne håndtere minst opptil 3560 interrupts i sekundet, se formell 5.2.

$$\frac{35600 \text{ rotasjon/min}}{60 \text{ sek/min}} \times 6 \text{ interrupts/sek} = 3560 \text{ interrupts/sek} \quad (5.2)$$

En testbenk ble satt opp for å teste om mikrokontrolleren greie å håndtere alle motor sensor interruptene. Et Arduino brett ble brukt for å simulere utgangene til motorens Hall sensorer med en hastighet som tilsvarte 400'000 rpm. Arduino brettet var programmert til å påføre et bestemt antall interrupts som mikrokontrolleren telte. Da antallet registrerte interrupts stemte med antallet sendt ut ved disse hastighetene kunne man være sikker på at mikrokontrolleren ville greie å håndtere kommuteringen av motoren ved de faktiske hastighetene.

5.5 Strømovertvåking

Ifølge funksjonskravene WMF-04 og WSF-02-04-01 skal motoren være beskyttet mot overopphetning i maskinvare og man skal kunne måle hvor mye strøm motoren trekker. Overopphetning skjer når motoren trekker for mye strøm. Utregningen av den maksimalt tillatte strømmen er gjort tidligere av Kråkens[3] ifra motorens datablad[9] og er satt til 145 mA.



Figur 5.5: Skjemategning til strømovertvåkingsmodulen. Figur hentet fra Kråkenes[3]

Motoren får strøm gjennom et 20 KHz PWM-signal som går gjennom den 1 ohms store målemotstanden før det går til jord. Spenningen over målemotstanden blir lavpasfiltrert og forsterket av en operasjonsforsterker før det går inn på en komparator krets som setter en SR-vippe, denne sørger for

å slå av strømforsyningen til motoren når den trekker mer enn den maksimalt tillatte strømmen. SR-vippen kan bli resatt av mikrokontrolleren. Utgangen til operasjonsforsterkeren er også koplet til en ADC-pinne på mikrokontrolleren. Strømmen kan da måles ved å lese av ADCen o

Som vi ser av skjemategningen går motorstrømmen over den 1 ohms store målemotstanden. Deretter blir signalet lavpassfiltrert, motoren er styrt med en 20 kHz PWM signal, og forsterket opp av en operasjonsforsterker. For å få strømovervåkning har utgangen fra operasjonsforsterkeren blitt lagt ut til en av mikrokontrollerens ADC innganger. Strømmen gjennom motoren kan da leses av ved å bruke formel 5.5.

```

/* Multiply K with ADC value to get current in 1/10 uA
* ADCref = 5 V
* Aopamp = 15.7 (opamp gain)
* K = ADCref * (100'000/1024)/(Aopamp) = 31
*/
uint16_t K = 31;
current_mA = (K*ADC)/100; //divide by 100 to get mA

```

For å teste kretsen ble det brukt en variabel strømforsyning. Modulen skrudde av strømmen ved riktig verdi og mikrokontrolleren leste av strømmen med stor nøyaktighet.

5.6 CAN

Ifølge funksjonskravet WCF-02 skal protesen være styrt over PDCP-protokollen som baserer seg på CAN for å sende og ta imot meldinger. Prototypekortet er utstyrt med en CAN-tranceiver som er koplet opp mot mikrokontrolleren. CAN modulen ble testet med et USB til CAN modul som gjorde det enkelt å sende meldinger fra PCen. Mikrokontrolleren ble testet ved hastigheter opp til 125 kbit hvor alt fungerte som forventet.

5.7 Diverse

Diskuter tilkoblingskontakter her

Kapittel 6

Programvare implementasjon

Dette kapitlet beskriver hvordan programvaren er bygget opp og satt sammen. I tillegg diskuteres modulariseringen av koden og fallgruver ved programmering av et interrupt drevet system. Underkapittelene beskriver hva hver modul gjør og hvordan den forholder seg til resten av koden.

Koden kan deles opp i to hoveddeler. Driverkoden til NRWDen som er maskinvareavhengig og høy nivå laget(HLL) til PDCP-protokollen. Driverkoden er igjen delt opp i deler etter de ulike maskinvaremodulene. En oversikt over programvaren kan ses i figur 6.1. Her vises alle de ulike modulene og hvem som inkluderer hvem.

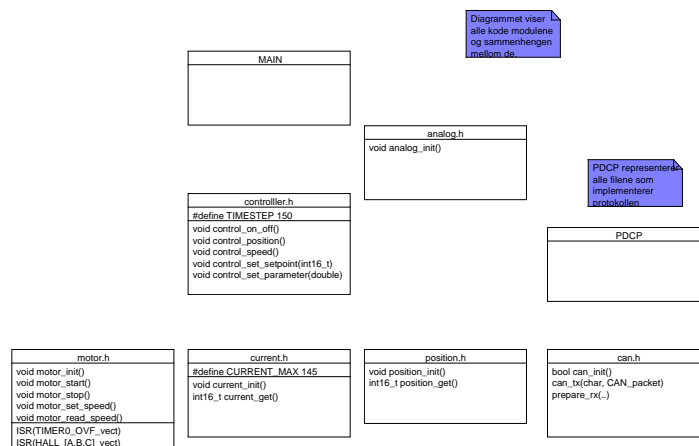
Under følger detaljer rundt hver modul.

6.1 Motor

Motor-modulen sørger for kommuteringen av motoren og vinkelhastighetsavlesning.

Pådraget til motoren settes ved `motor_set_speed()`. Denne funksjonen setter dutycyclen til PWM-signalet som driver motoren. Siden strømmen levert av dette signalet varierer med forsyningsspenningen setter man ikke vinkelhastigheten direkte. Kontrolering av farten til protesen skjer i controller modulen.

Etter at duty cyclen er satt vil Hallsensor interrupt rutinene automatisk ta seg av kommuteringen av motoren. En timer overflow interrup rutine vil kontinuerlig oppdatere vinkelhastigheten til protesen slik at den til enhver tid kan leses av med `motor_read_speed()`.



Figur 6.1: Klassediagram som viser innholdet i og avhengighetene mellom de ulike programvare modulene.

6.2 Current

Current-modulen bruker mikrokontrollerens ADC til å lese av strømmen som går gjennom motoren. Interruptrutinen, `CURRENT_vect`, som blir kjørt når maskinvaren skruer av strømmen til motoren ligger ikke her, men i main c-filen. Dette er fordi `CURRENT_vect` benytter seg av flere av modulene til å rydde opp i systemet etter at motorstrømmen har blitt skrudd av.

6.3 Position

Position-modulen setter opp en timer til "input capture mode". Den er satt opp til å generere et interrupt ved endring i flanken på signalet fra posisjons-sensoren. Ved en positiv flanke starter timeren og ved negativ flanke blir telleren i timeren lagret. Utifra verdien til telleren blir vinkelposisjonen til protesen regnet ut og lagret i en global variabel. Denne kan til en hver tid bli lest ut av `position.read` funksjonen.

6.4 Controller

Ifølge funksjonskravet WSF-01 skal bevegelsen til protesen være kontrollerbar gjennom de følgende måtene:

- On/Off-mode

- Position mode
- Velocity mode

Disse finnes implementert i `controller.c` som henholdsvis `control_on_off()`, `control_position()` og `control_speed()`. Utfordringen ved designet av regulatorne var at strømmen til motoren automatisk ble slått av hvis den oversteg en forhåndsbestemt maksgrense. En regulator som ikke hadde tatt hensyn til dette vil ikke ha gitt en god brukeropplevelse da protesemotoren ville skrudd seg av med gjevne mellomrom.

Funksjonskravet WSF-01-01 sier at On/Off-mode skal være kjørt i en open-loop kontroller. På grunn av overnevnte grunner har den istedenfor blitt kjørt i en PI-kontroller. Denne regulerer strømmen gjennom motoren tett opptil den maksimale grensen for å oppnå høyest mulig vinkelhastighet.

Fartsregulatoren løser begrensingen i motorstrøm ved å bruke en dobbel PI-regulator. Den ytre sløyfen tar inn vinkelhastighet og setter strøm referansen til den indre sløyfen. Den indre setter pådraget til motoren basert på strømreferansen og sørger samtidig for at strømmen ikke går over den maksimalt tillate verdien.

Posisjonsregulatoren fungerer på samme måte som fartsregulatoren, bortsett fra at den ytre sløyfen tar inn vinkelposisjonen til protesen.

I alle regulatorne er utregningen av integralleddet uavhengig av frekvensen som regulatoren kjører på. Dette betyr at om frekvensen til regulatorløkken blir endret så må regulatorne justeres. Frekvensen til regulatorløkken er styrt med en busy-wait forsinkelse funksjon, se figur 6.2. Denne typen forsinkelsesfunksjoner registrerer ikke når man er inni en interruptrutine. Dette betyr at regulatorløkken blir forsinket med så mye tid man bruker inni hver av interruptrutinene. Da mikrokontrolleren bruker mindre enn 2% av tiden sin inne interruptrutiner vil dette ikke medføre noen problemer for regulatoren.

```
while (1)
{
    control_controller ();
    _delay_ms (TIMESTEP);
}
```

Figur 6.2: Main while loop

Verdien til P- og I-leddet til de tre regulatorne ble funnet ved å prøve å se hva som virket bra. De gir god regulering, men det kan med fordel brukes mer tid for å få enda bedre respons.

Funksjonskravet WSF-03 setter krav til at man skal kunne skifte både regulatortype og vedien til regulatorparametrene gjennom PDCP-protokollen. Bytte av regulatortype gjøres gjennom `control_set_mode()` funksjonen. For å bytte parametere må man først velge regulator gjennom nevnte funksjon, deretter kan man sette P- og I leddet ved `control_set_parameter()` funksjonen.

6.5 Analog

Funksjonskravet WCF-04 sier at NRWDen skal kunne styres ved å bruke analoge elektroder. Dette er den normale måten å styre myoelektriske proteser. Videre sier WCF-02-01 at de analoge linjene skal kunne samples ved minst 1 kHz. Myoelektriske signaler har en båndbredde på 500 Hz.

Analog-modulen ble ikke ferdig implementert i dette prosjektet. Hadde den blitt ferdig hadde den hatt de samme oppgavene som PDCP-modulen, ta imot eksterne signaler og styre protesen.

Da kommunikasjonen ville vært rent analogt ville den ikke hatt like rik funksjonalitet som PDCP-protokollen. I sin enkleste form ville den kunne styre protesen i on/off-modus.

6.6 PDCP and can

Når protesen er ferdig skal den kunne styres gjennom PDCP-protokollen. PDCP-protokollen er en åpen kilde protese kommunikasjonsprotokoll. Den har blitt til ved UNB i Canada og har som ambisjon og bli den nye standarden for protesekommunikasjon.

I dette prosjektet har det blitt brukt en NTNU produsert PDCP-implementasjon. Den var oppdelt i to lag, et høynivå[4] lag som implementerte selve funksjonaliteten til protokollen og et maskinvare abstraksjonslag[5] som var kodet for å være lett å tilpasse til ny maskinvare.

I masteroppgaven[5] knyttet til koden hevdes det at koden skal være lett å tilpasse til en mikrokontroller med intern CAN-kontroller. Dette var ikke min erfaring. Det ble brukt mye tid på å forsøke og kode om maskinvarelaget til å passe med AT90CAN mikrokontrolleren. Etter å ha jobbet mye med koden er min konklusjon at den er godt egnet om man skal skifte AVR mikrokontroller, men fortsatt holde seg til den eksterne CAN kontrolleren MCP2515 som koden har blitt designet rundt.

Da man ikke greide å tilpasse det eksisterende maskinvare abstraksjonslaget ble det skrevet et nytt et som implementerte basis funksjonaliteten som krevdes for å teste ut PDCP-protokollen. Denne koden finnes i `hal-c` filen.

Bruken av den interne CAN-kontrolleren er gjort med et åpent CAN biblioteket skrevet for AT90CAN mikrokontrolleren. Biblioteket består av filene can.c og can.h.

6.7 Main

Main starter med å initialisere de ulike modulene og velge kontrollertype. Det er i main-c filen at callback funksjonen fra PDCP-modulen er plassert, se kode 6.3.

```
/* callback_incoming_can */
/*
 * @fn          void callback_incoming_can(struct socket *so)
 * @brief       Callback function from HLL
 */
void callback_incoming_can(struct socket *so){

    struct can_msg *msg = sock_pull(so);
    switch(msg->function_code){

        case REQUEST_BIND:

            break;
        case RESPONS_BIND:

            break;
        case REQUEST_GET_INFO:

            break;

        ...
    }
```

Figur 6.3: Callback funksjonen som blir kalt fra PDCP-modulen hver gang den motar en melding.

Da dette prosjektet ble skrevet fantes det en PDCP-spesifikasjon[7], men ingen eksempel kode som viste hvordan de ulike PDCP-funksjonskodene skulle brukes. Det har blitt laget skjelettkode for å håndtere de ulike PDCP-funksjonskodene. Når eksempel kode blir gjort tilgjengelig kan funksjonaliteten til de ulike kommandoene skrives ferdig.

Det som i hovedsak manglet ved overtagelsen av prosjektet var å skrive ferdig programvare til NRWDen. Det var skrevet drivere til de fleste modulene hvor koden var av varierende kvalitet, skrevet mer for å teste maskinvaren enn for å kunne brukes i det ferdige produktet.

Modulariseringen av koden ble beholdt da den allerede var delt opp i fornuftige deler.

Systemet er for det meste drevet av interrupts, både interne og eksterne. Motorkommuteringen, posisjonsmålingen, motagelser av CAN-meldinger, fartsmåling og så videre skjer alle inni interruptrutiner. Når koden som kjører til en hver tid kan bli avbrutt av en interrupt rutine er det meget viktig å kode slik at det ikke oppstår race[?] kondisjoner. Den enkleste måten å forhindre dette på er å sørge for best mulig separasjon mellom modulene, så ikke en modul kan endre på variabler brukt av en annen modul.

Det nederste laget er driverne til de forskjellige hardware modulene, `current.h` for strømovervåkning, `position.h` for posisjonssensoren, `motor.h` for motorstyring og `can.h` for det nederste laget av PDCP protokollen.

Selve implementasjonen av PDCP protokollen er en porting av programvare skrevet av Andreas Nordal og Andrzej Zamojski i sine masteroppgaver [4, 5].

Kapittel 7

Diskusjon

Kapittel 8

Konklusjon

Det har i denne oppgaven blitt utviklet programvare for å styre NTNUs Rotary Wrist Device, en 1-akse håndleddsprotese. Koden har blitt skrevet for å kjøre på maskinvare laget av tidligere NTNU studenter.

Systemet har blitt klargjort for å kunne bruke den åpne protese kommunikasjonsprotokollen PDCP. Implementasjonen av PDCP-protokollen for AVR hadde blitt gjort tidligere på NTNU. I denne oppgaven ble koden tilpasset for å kjøre på ATMELs AT90CAN128 mikrokontroller.

Det har blitt skrevet programdrivere for alle de forskjellige maskinvaremodulene lagt ut på prototypekortet. For å styre bevegelsen til håndleddet har det blitt skrevet og testet tre ulike regulatorer. En fartsregulator, en posisjonsregulator og en av/på-regulator som kjører protesen ved maksimum vinkelhastighet i valgt retning.

Da prosjektet ble overtatt var friksjon i mekanikken til protesen for stor til at den kunne brukes. Den ble tatt med til NTNUs mekaniske verksted hvor de fikk rettet en girakse som ga betydelig reduksjon av friksjon.

Alt av overlevert maskinvare har blitt nøye testet. Forslag til forbedring av maskinvaren har blitt dokumentert.

Videre arbeide vil bestå i å fullføre integrasjonen med PDCP-protokollen. Dette avhenger av at det blir publisert flere detaljer rundt bruken av protokollen. Før protesen skal brukes i eksperimenter vil man ha stor nytte av å miniaturisere kretsen slik at den får plass inni protesen. Under miniaturiseringen av kretsen anbefales det å utbedre maskinvaren med de anbefalinger gjort i denne oppgaven.

Arbeidet på en analog kommunikasjonsmodul, som et alternativ til PDCP-modulen, har blitt påbegynt.

Bibliografi

- [1] Arthur Zinck, *Design of a compact, reconfigurable, prosthetic wrist*. 2011.
- [2] Øyvind Stavdahl *Optimal wrist prosthesis kinematics: Three-dimensional rotation statistics and parameter estimation*. Department of Engineering Cybernetics, Norwegian University of Science and Technology. 2002. PhD Thesis.
- [3] Andreas Kråkenes, *Styresystem for kybernetisk h ndleddsprotese*. Master i teknisk kybernetikk 2011.
- [4] Andreas Nordal, *Implementasjon og testing av en  pen bussprotokoll for armproteser*. Master i teknisk kybernetikk 2012.
- [5] Andrzej Zamojski, *Design, Implementation and Testing of Low-level Layers of the PDCP for the AVR Platform*. Master i teknisk kybernetikk 2012.
- [6] Inge Brattbakken *Embedded control system for cybernetic wrist prosthesis* Master of Science in Engineering Cybernetics 2010.
- [7] Yves Losier *Prosthetic Device Communication Protocol for the AIF UNB Hand Project* UNB Hand Project Systems Integrator 2012.
- [8] *8-bit AVR Microcontroller with CAN controller AT90CAN128. Datablad*. Atmel
- [9] *Brushless DC-Servomotor, series 0620 B. Datablad* Faulhaber

Tillegg A

Innhold CD

Tillegg B

Funksjonsspesifikasjon NRWD

Tillegg C

Diverse

Duty cycle	Vinkelhastighet
80	156 grader/sekund
65	94 grader/sekund
50	44 grader/sekund

Tabell C.1: Vinkelhastighet til NRWDen ved ulike duty cycles, 12V spenningsforsyning