NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

Prosjektoppgave

Kandidatens navn:	Rainer Storheil
Fag:	TTK4720 Navigasjon, fartøystyring og robotteknikk, fordypningsemne, Teknisk Kybernetikk
Oppgavens tittel:	Innvevd programvare for kybernetisk håndleddsprotese
Oppgavens tekst:	Forskning har frembrakt ny teoretisk viten om hvordan en motorisert håndleddsprotese bør utformes for å tilby brukeren maksimal nytteverdi. En ønsker å lage en fysisk protese etter disse prinsippene for å evaluere den reelle nytten. Protesen vil ha et mikrokontrollerbasert styresystem, og målet for denne oppgaven er å utvikle og realisere programvaren for systemet basert på en foreliggende funksjonsspesifikasjon.

1. Gi en oversikt over kommunikasjonsgrensesnitt og styringsstategier i eksisterende protesesystemer, og å relatere den foreliggende oppgaven til disse systemene ved å påpeke likheter og ulikheter. Oversikten baseres på tilgjengelig litteratur og annen informasjon.

2. Analysere den foreliggende problemstillingen og beskriv på bakgrunn av dette en modulær programvarearkitektur tilpasset problemet.

3. Foreta en detaljert konstruksjon av de arkitektoniske elementene (modulene) i systemet. Beskriv tester som kan utføres på ferdige moduler og system for å konstatere hvorvidt spesifikasjonen er oppfyllt.

4. Implementer og prøv ut deler av (evt. hele) programvaresystemet så langt tiden strekker til. Dersom det blir knapt med tid legges vekten på implementasjon og utprøving de essentielle algoritmene, mens lavnivå for I/O etc. utsettes.

Oppgaven gitt:	15.01.2005
Besvarelsen levert:	17.06.2005

Utført ved Institutt for teknisk kybernetikk

Veileder: Øyvind Stavdahl

Trondheim, den 17.06.2005

Faglærer: Geir Mathisen

Forord

Denne mastertoppgaven er et ledd i en større prosjekt (NRWD) der målet er en kommersiell protese. Denne protesen skal utvikles med utgangspunkt i nyere forskning. Denne oppgaven skal utrede et styresystem for denne protesen.

Masteroppgaven er laget for Institutt for Teknisk Kybernetikk ved NTNU, Trondheim. Oppgaven er et ledd i Øyvind Stavdahl's prosjekt om å få ferdigstillt en håndleddsprotese. Motivasjon er at denne er basert på optimal kinematikk for en slik protese med 1 frihetsgrad.

Arbeidet mot målet vil fortsette med ferdigstilling av en prototyp og vil forhåpentligvis resultere i en ny type protese som letter hverdagen betraktelig for personer som har mistet deler av armen.

Jeg ønsker å rette en takk til veileder Øyvind Stavdahl for hans enorme engasjement og evne til å skape entusiasme rundt dette prosjektet. Det har vært en særdeles lærerik prosess og virkelig gitt mersmak. Jeg vil også takke "partneren" min, Håkon Skjelten for hans hjelp underveis. Gutta borte på hangaren må også få sin takk for enorm hjelp og støtte til det meste. En spesiell takk går også til kokkene ved Grenaderen som har gjort det mulig for meg å kombinere denne oppgaven med en av mine største interesser i livet, kokkelering.

Trondheim 17. juni 2005

Sammendrag

Denne rapporten er et ledd i et større prosjekt, The NTNU Revolute Wrist Device (NRWD). NRWD er en håndleddsprotese med en frihetsgrad som opererer rundt en optimal akse **k**. Prosjektet har også som et mål og kunne integreres i en birolle i et modulært protesesystem styrt av en sentral enhet. I tillegg skal protesen være slik oppbygd at man enkelt kan tilføye egenskaper uten å måtte skifte ut hele systemet. Med utganspunkt i en utarbeidet spesifikasjon for dette prosjektet, blir det designet og implementert en innvevd programvare for protesen i denne rapporten.

Oppgaven tar først for seg litt historikk rundt håndleddsproteser og forskningen gjort på dette området. Historikken leder helt fram til dagens kommersielle proteser. Dette er et forskningsområde det ikke er viet alt for mye oppmerksomhet. Forskningen har ofte konsentrert seg om selve hånden, mens håndleddet har blitt utelatt. Håndleddet er derfor ofte enkle løsninger som ikke gir særlig stor nytteverdi.

Videre blir det også sett litt på styreprinsipper innenfor denne forskningen. Her har det derimot blitt gjort en god del forskning, men med økt funksjonalitet følger også økte belastninger for brukeren.

Oppgaven tar videre for seg selve styresystemet. Teknikken som er brukt i oppgaven er et såkalt “top-down” design. NRWD blir delvis brutt ned fra hele programvaren, via modulene den må inneholde og deres klasser, til design og implementasjon av disse. Til slutt er det satt opp en kommunikasjonsprotokoll for NRWD, og også innspill til en kommunikasjonsprotokoll for den modulære armprotesen er listet opp.

Siden det ikke er ferdig noen maskinvare og arbeidet med en kommunikasjonsprotokoll for den modulære armprotesen akkurat har startet, er ikke programvaren fullstendig. Av den grunn er det heller ikke blitt implementert på en mikrokontroller, men lagt klart for at dette kan gjøres.

Videre arbeid vil nå bestå i å samkjøre maskinvaredelen og programvaredelen til en fungerende enhet. Deretter kan en prototyp konstrueres, og vha. målinger på denne kan regulatorer designes og tunes. Når kommunikasjonsprotokollen til hele armsystemet er klart, kan også dette implementeres og prototypen kan testes fullstendig.

Planen er å ha en prototyp ferdig sommeren/høsten 2005.

1 Innledning

For mange mennesker i dag er en protese et godt hjelpemiddel i hverdagen. Stadig utvikling og forbedringer av proteser gjør livet deres lettere. Men utviklingen kan vel aldri helt erstatte deres tap. Proteser forbedres stadig i form av lengre batterikapasitet, mindre vekt, mindre størrelse og større funksjonalitet. Men dette kan være en veldig vanskelig kombinasjon. Ofte skjer forbedringene på et og et punkt av gangen.

Denne rapporten er skrevet som en del i et større prosjekt, the NTNU Revolute Wrist Device (NRWD). Dette prosjektet baserer seg på en doktoravhandling[1], som konkluderer med en optimal vinkel for et håndledd med en frihetsgrad. Målet med prosjektet er en ferdig håndleddsprotese som kan bedre hverdagen til en bruker. Det er også satt som et klart mål at protesen skal kunne benyttes som en del av en større protese. Dvs. en protese som i seg selv har flere enheter, f.eks. albue, håndledd og en hånd. NRWD må derfor være tilpasset dette systemet. Flere forskningsmiljø samarbeider per dags dato om et slikt system.

Denne oppgaven går simultant med en annen oppgave[2]. Det skal her designes en programvare for protesen, mens den andre oppgaven designer maskinvaren. Det kunne her vært en fordel om en ferdig prototyp i form av maskinvare og konstruksjon var ferdig før design av programvare startet. Dette hadde muliggjort en tidlig implementasjon og testing av systemet underveis. Siden samarbeidet om et modulært protesesystem heller ikke har pågått lenge, mangler en del vesentlig informasjon i forhold til tilpassing mot dette systemet.

Det vil i denne rapporten bli gitt en kort gjennomgang av protesers historie og utvikling. Det vil også bli sett på viktige styreprinsipper, før man kommer til selve styresystemet til NRWD.

Oppgaveteksten er gjengitt på neste side.

MASTEROPPGAVE

Kandidatens navn: Rainer Storheil

Fag: Teknisk Kybernetikk

Oppgavens tittel(norsk): Innvevd programvare for kybernetisk håndleddsprotese

Oppgavens tittel(engelsk): Embedded software for a cybernetic wrist prosthesis

Oppgavens tekst:

Forskning har frembrakt ny teoretisk viten om hvordan en motorisert håndleddsprotese bør utformes for å tilby brukeren maksimal nytteverdi. En ønsker å lage en fysisk protese etter disse prinsippene for å evaluere den reelle nytten. Protesen vil ha et mikrokontrollerbasert styresystem, og målet for denne oppgaven er å utvikle og realisere programvaren for systemet basert på en foreliggende funksjonsspesifikasjon.

1. Gi en oversikt over kommunikasjonsgrensesnitt og styringsstategier i eksisterende protesesystemer, og å relatere den foreliggende oppgaven til disse systemene ved å påpeke likheter og ulikheter. Oversikten baseres på tilgjengelig litteratur og annen informasjon.

2. Analysere den foreliggende problemstillingen og beskriv på bakgrunn av dette en modulær programvarearkitektur tilpasset problemet.

3. Foreta en detaljert konstruksjon av de arkitektoniske elementene (modulene) i systemet. Beskriv tester som kan utføres på ferdige moduler og system for å konstatere hvorvidt spesifikasjonen er oppfyllt.

4. Implementer og prøv ut deler av (evt. hele) programvaresystemet så langt tiden strekker til. Dersom det blir knapt med tid legges vekten på implementasjon og utprøving de essentielle algoritmene, mens lavnivå for I/O etc. utsettes.

Oppgaven gitt: 15. januar 2005

Besvarelse leveres: 20. juni 2005

Besvarelse levert: 17. juni 2005

Utført ved: Institutt for teknisk kybernetikk

Veileder: Øyvind Stavdahl

2 Proteser

En protese er definert ved et den skal erstatte en tapt legemsdel. Mange blander også feilaktig inn ortoser i kategorien for protoser. Ortoser er derimot definert ved at den avlaster eller støtter en eksisterende kroppsdel. Proteser finnes i mange utgaver og for mange ulike deler av kroppen. Det mest vanlige er proteser i form av hender eller føtter, men etterhvert har det også kommet flere og flere proteser som til en hvis grad kan erstatte både syn og hørsel. I motsetning til hva mange tror er det ikke ulykker som er hovedårsaken til avhengigheten til proteser, men de fleste er derimot født med et manglende lem.

Av protesene vi finner tilpasset til et menneskets arm, kan man klart se en sammenheng mellom protesen og armens sentrale ledd. Men jo flere ledd man mangler, jo større og mer kompliserte proteser får man. Dette fører ofte til at protesens nytteverdi blir langt mindre enn de ekstra belastninger proteser medfører. Derfor velger mange protesebruker enkle proteser. Dette gjør de enkle å bruke, men der mindre av en normal arms funksjoner er tilstede. For mange er også utseende veldig viktig. Mange ganger kan det være langt mere praktisk med en protese som viker fra utseende til en normal arm, men som er mere funksjonell. Dette har ikke hatt stort gjennomslag akkurat av den grunn at folk vil se mest mulig "normal" ut. Det vil her bli lagt vekt på de mest vanlige former proteser, siden det er disse som stort sett blir brukt.

Dette kapittelet vil ta for seg proteser og gjøre leseren kjent med dens verden. Det vil spesielt bli lagt vekt på overkroppsproteser siden denne oppgaven omhandler håndleddet. Det vil bli sett nærmere på det generelle rundt proteser, hva som finnes av proteser og hvordan disse virker. Det vil også bli nevnt litt rundt videre forskning og etiske spørsmål som dukker opp underveis.

2.1 Protesetyper

Ikke alle potensielle protesebruker velger å bruke en protese. Mange som med tiden har lært seg å leve uten en protese føler seg komfortable uten protesen. Den største fordelen med dette er at man tar i bruk kroppens naturlige sensortilbakemeldinger, man kan føle det man berører. Det finnes også en kontroversiell metode kalt The Krukenberg Procedure som kan benyttes av personer med deler av underarmen inntakt. Et inngrep skiller de to bena i armen, ulnar og radius fra hverandre slik at disse kan beveges fra og mot hverandre og danne en gripefunksjon. Fordelene her er som over, følsomhet, samtidig som man kan gripe objekter. Ulempen er derimot at dette ikke ser særlig naturlig ut, og derfor er mange skeptiske til inngrepet.

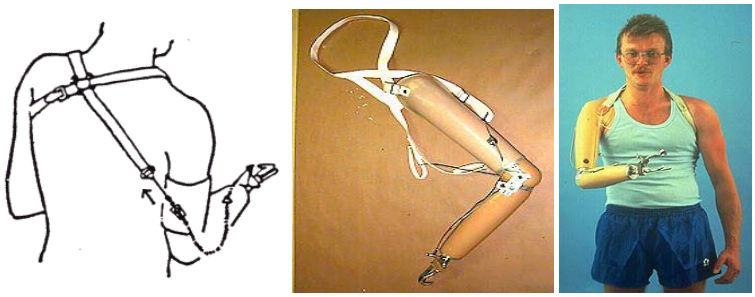
For personer som tar i bruk proteser finnes det en del muligheter. Det vil nå bli kort forklart hva disse går ut på.

2.1.1 Kosmetiske proteser

Kosmetiske proteser blir brukt av den enkle grunn at det skal se naturlig ut. Siden disse protesene er passive, blir det heller ikke store belastningen på brukeren med hensyn på vekt, størrelse og støy. En passiv enhet gjør det også lettere å gjenskape en naturlig hånds utseende siden den ikke vil bli utsatt for bevegelser innad. Etterhvert har også de kosmetiske protesene gått litt bort fra å være helt passive. Ved å ha elastiske fingre, kan brukeren selv forme fingrene til for eksempel å gripe en gjenstand.

2.1.2 Kroppsdrevne proteser

Kroppsdrevne proteser blir som tittelen sier drevet av kroppen selv. Seletøy gjør det mulig for brukeren å styre protesen ved hjelp av eksisterene kroppsfunksjoner. I Figur 2-1 ser vi noen eksempler på slike seletøy. Brukeren kan her åpne og lukke kloen på protesen ved bruk av motsatt skulder. Albueleddet kan også kontrolleres ved hjelp av wire med samme prinsipp. Seletøyet er dermed en forlengelse av kroppen som styrer en protese ved hjelp av kroppens egen mekaniske energi.

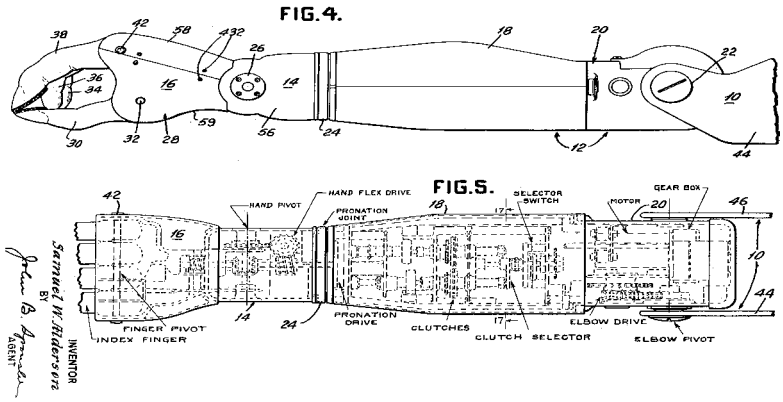


Figur 2-1: Forskjellige eksempel på seletøy

2.1.3 Eksternt drevne proteser

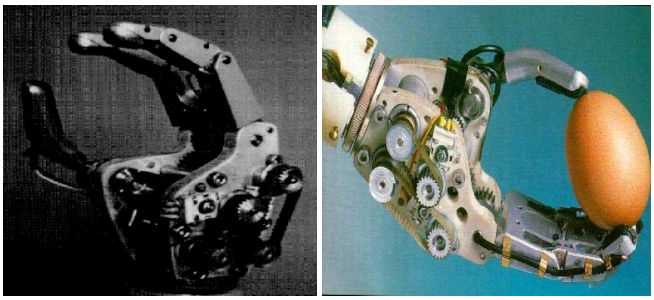
Her blir protesen drevet av en ekstern energikilde. Batterikapasitet var lenge et stort problem for brukerne, men utviklingen på området går raskt og man har i dag godt egnede batterier selv om batteriets kapasitet, størrelse og vekt fortsatt er en vanskelig problemstilling.

Man finner her alt fra proteser designet for en spesiell situasjon til helarmsproteser. Proteser for skulderledd, albueledd, håndledd og hender finnes på markedet. Spesielt innenfor hender er det et stort utvalg. Mye av forskningen innen proteser har da også gått på hender. Man kan her finne proteser spesiallaget for diverse idretter til hender med fem bevegelige fingre. Men de fleste av de mest avanserte protesene egner seg mest til forskning. Størrelse, vekt og komplikasjonsgrad gjør de lite populære blant brukerne.



Figur 2-2: Alderson's IBM-hånd, patentert i 1952

I 1952 patenterte Alderson sin IBM-arm[3]. Denne armen er vist i Figur 2-2. Armen hadde albueledd, bevegelig håndledd i form av rotasjon og fleksjon/ekstensjon og en gripefunksjon. Den var ikke egnet for daglig bruk blant annet pga. vekt og strømforbruk, men den banet vei for videre forskning. Etterhvert kom også den svenske SVEN-hånden[4] som hadde hele seks frihetsgrader, tett fulgt av det britiske protesemiljøet. Britene har hatt stor utvikling innen proteser. Verdt å nevne her er vel Oxford-hendene[5][6], the CyberHand[7] og the Edinburgh Modular Arm System[8].



Figur 2-3: To eksempler på Oxford-hender

2.2 Styreprinsipper for proteser

Proteser blir festet til kroppen ved hjelp av seletøy, hylser eller osseointegrasjon[9]. Seletøy ble forklart i 2.1.2 og anvendt av personer som mangler albueleddet. For personer med albueledd, får man spesiallaget en hylse av en ortopediingeniør som sitter på stumpen. Osseointegrasjon foregår ved å operere inn en metallbit i skelettet. På denne måten ivaretar man mye av armens funksjoner og i tillegg blir pasienten mindre følsom for protesens vekt. Problemet med denne metoden er infeksjoner rundt metallet og ubehaget rundt en operasjon. Dette er derfor en lite brukt metode men forskningen på området for forbedringer pågår stadig.

Når man så har protesen festet til kroppen, må systemet styres. Dette er blitt viet mye forskning og utviklingen er veldig stor på dette området. Det vil her bli gjennomgått endel stryreprinsipper for de ulike protesetypene.

2.2.1 Kroppsdrevet

Som det ble kommet inn på tidligere blir seletøy drevet av kroppens inntakte funksjoner. Dette gjør brukeren i stand til enkelt å kunne vite hvilken tilstand protesen er i siden bestemte bevegelser av for eksempel skulder gir en bestemt bevegelse av protesen. Problemet her er at brukeren må bevege skulderen i uvante posisjoner for bestemte bevegelser for protesen. Har man da en inntakt arm man bruker samtidig vil det kunne oppstå konflikter mellom arm og protese i visse situasjoner. Dette vil selvsagt med tiden bli en vanesak for brukeren, men det vil begrense arbeidsrommet.

2.2.2 Extended Physiological Proprioception - EPP

Dette prinsippet er et meget sentralt prinsipp innen proteseforskning. Det ble innført av Simpson på 70-tallet[10]. Det dreier seg her om proposjonal styrke mellom protese og kroppsbevegelse. Når en protesebruker utøver en kraft mot protesen i form av en bevegelse, vil kraften protesen utøver være proposjonal. Vil en kraft bli utøvd mot protesen, vil den tilsvarende proposjonale kraften også bli utøvd mot brukeren. Det enkleste eksemplet er de tidligere nevnte seletøyene. Flytter brukeren skulderen for eksempel 1 cm framover vil kloen for eksempel åpne seg 1 cm. Hvis man så holder igjen kloen med en viss kraft, vil brukeren kjenne dette ved at skulderen blir dratt bakover. Dette kan utvikles videre ved å tilsette en ekstern kraftkilde, slik at en bevegelse på 1 cm i skulderen kan flytte albuen eksempelvis 5 cm. Et annet eksempel fra idrettens verden kan her illustrere prinsippet. Å bruke en golfkølle er en form for EPP. Man beveger armen i en liten bue, mens køllebladet har en langt større bane. På denne måten forlenger man kroppens funksjon og momentet på ballen blir langt høyere. På grunn av ballens lave vekt, vil ikke brukeren(golfspilleren) kjenne treffet særlig. Men bytter man ut golfballen med noe langt tyngre, vil kraften også kjennes på kroppen. På denne måten har man en lukket sløyfe.

En EPP posisjonsregulator for et protesesystem, basert på et mikroprosessordesign er gjennomført med gode resultater i nyere tid[11].

2.2.3 Myoelektrisk styring

Dette er en av de mest brukte metodene for styring av proteser. Det ble introdusert av Reiter på 40-tallet[12]. Videre har dette stadig utviklet seg og blitt bedre. Elektromyografi(EMG) går ut på å plassere en sensor over en muskel. Man vil da måle spenningene som går i muskelen. Ved å bruke amplituden på signalet kan man styre en protese. Ved å bruke to motvirkende muskler kan man få styrt en protese i to retninger. Ved å måle på flere muskler og bruk av nevrale nettverk, kan man finne mønster slik at en bruker kan styre en protese med flere ledd. I senere tid har også bruk av IEMG[13], eller implantet EMG gitt veldige gode resultater. I stedet for at sensoren ligger på hudens overflate, opererer man inn sensoren slik at den er i kontakt med muskelfibrene. På denne måten kan man måle langt mere nøyktig og det er også gjort forskning med positivt resultat hvor man kan skille mellom de forskjellige fibrene i

musklene. Dermed har man enda større muligheter innenfor styring av en flerleddsprotese.

Mechanomyografi(MMG)[14] er også et område det har vært en del forskning på i det siste, med gode resultat. Her måler man istedet de mekansike svingningene i muskelen. Dette er enda på forskningsstadiet, men de oppnådde resultater tyder på at dette også vil bli å finne på proteser i tiden som kommer.

2.2.4 SAMS

The Southampton Adaptive Manipulation Scheme (SAMS)[15] er et hierarkisk styresystem som gjør det mulig å styre flere uavhengige bevegelser samtidig som det krever færre grader av styresignaler. Dette er et system utviklet av det britiske protesemiljøet med Peter J. Kyberd i en av hovedrollene. Dette systemet er naturligvis veldig ofte benyttet og gjør utviklingen av bedre proteser langt lettere. The Southampton Hand[15] er en av de første prototypene med dette systemet.

2.2.5 Tanke- og nervestyrt

Her finner man de mest revolusjonerende styresystemene. Ved å koble seg direkte til hjernen eller nervene kan man gjennskape kroppens eget styresystem. Men dette er ikke særlig utviklet enda, selv om forskningen på området er stort. Forskerne støter stadig på store problemer, blant annet innen lovgivning og etikk som vil bli diskutert i neste avsnitt.

Tankestyrte proteser er det området man har kommet lengst på. Her har man faktisk klart å utvikle en funksjonell protese[16]. Arbeidet startet med Dr. Todd Kuiken. Han utviklet en metode hvor man operativt flytter en ubrukt nerve til en funksjonell muskel. En nerve som tidligere gikk til for eksempel en persons biceps, kan etter en amputasjon flyttes til for eksempel personens brystmuskel. Når denne nerven slår rot, måler man signalet som kommer fra hjernen, og man kan ved hjelp av datamaskiner finne algoritmer til å styre en protese. Kuiken testet ut sin metode på dyr med stort hell. Høsten 2004 ble en amerikaner som hadde mistet begge armene i en ulykke, utstyrt med Kuikens tankestyrte protese. Forsøket ser ut til å være vellykket, men det er for tidlig å si noe om resultatet enda.

Figur 2-4: Jesse Sullivan med sin tankestyrte protese

Nervestyrte proteser har ikke kommet like langt, kanskje spesielt pga. streng lovgivning. Forskere har lenge forsøk å koble seg på nerver for å styre proteser på denne måten. Dette har til noen grad lyktes, men man har ikke gode nok resultater til

kommersielle produkt. Men den siste tids gjennombrudd i stamcelleforskning gir håp. Håpet er å implantere mikrochiper inn i kroppen, for så at nerver skal gro gjennom disse. Dette har mislykkes så langt, men ved hjelp av stamceller har de klart dette i laboratorier. Ved å implantere disse inn i kroppen, håpes det at disse vil gro fast i nerven og man gjenskaper kroppens eget styresystem. Men det er langt igjen, og spørsmålet er om man kommer så langt. Eller rettere sagt, får komme så langt.

2.3 Etikk

Det er store muligheter for revolusjonerende fremskritt i proteseforskningen. Men samtidig som man ser store fordeler med denne forskningen, dukker det også opp en del etiske spørsmål. Spesielt ifm. stamcelleforskning går debatten høylytt. I Norge er denne form for stamcelleforskning ulovlig, men vi trenger ikke dra lenger enn til Sverige for å finne en litt mildere lovgivning. I USA kjemper President Bush for å beholde dagens lovgivning, mens Kongressen i mai 2005 gikk inn for å åpne opp lovene. I første omgang går dette på å finne kur for en del sykdommer, blant annet Alzheimers, men proteseforskningen kan også dra nytte av dette.

Videre er mange motstandere av å "tukle" med skaperverket. Det å implantere sensorer i hjernen er meget omdiskutert. Hjernen er omtalt som et av dagens største mysterium og mange er skeptiske til bivirkninger. Forskning på mennesker er generellt et vanskelig tema. Derfor må man også i de fleste land søke om tillatelse til slik forskning.

Et annet stort tema er hva forskningen blir brukt til. Er det i hovedsak for å bedre livskvaliteten til mennesker, eller skape supersoldater for militæret. Det er ingen hemmelighet at det amerikanske forsvaret(DoD) årlig skyter inn millioner av dollar til slik forskning. Mange spør seg hvor grensen må gå. Heldigvis har de fleste land strenge lovgivninger rundt omstidt forskning.

NRWD 3

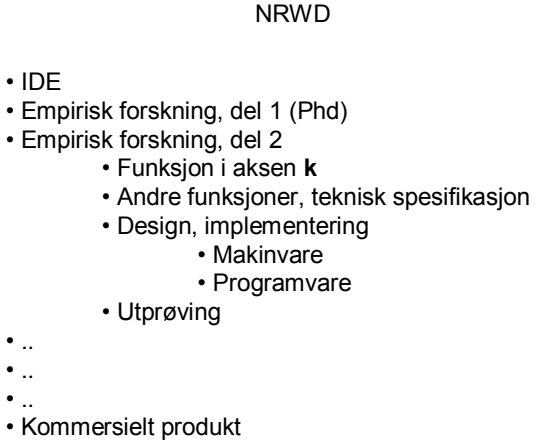
Hovedmålet med denne oppgaven er å utvikle et styresystem for håndleddsprotesen NRWD. Før det gåes inn på selve styresystemet vil bakgrunnen for oppgaven bli gjennomgått. Det vil her dreie seg om prosjektet NRWD og de forskjellige delene av dette prosjektet.

3.1 NTNU Revolute Wrist Device(NRWD)

NRWD er et prosjekt satt i gang som følge av Øyvind Stavdahl's doktoravhandling[1]. I denne avhandling kom han frem til en optimal vinkel for en håndleddsprotese med en frihetsgrad. Deretter ville han utvikle en prototyp med denne vinkelen som utgangspunkt for å kunne se virkningen av sin forskning. Prosjektet ble startet i regi av NTNU og Øyvind Stavdahl.

3.1.1 Prosessen

Som nevnt over er dette et prosjekt basert på en ide av Øyvind Stavdahl. Figur 3-1 viser prosjektets deler.



Figur 3-1: Prosjektplan for NRWD

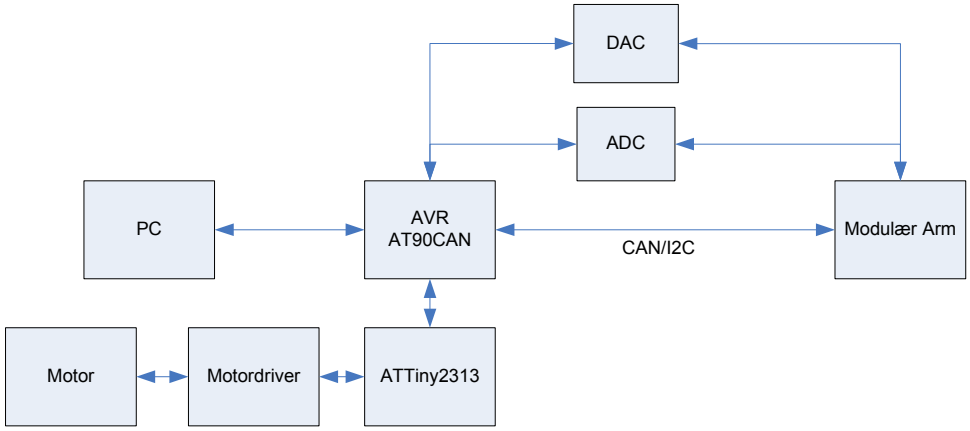
Stavdahl's avhandling ble avsluttet våren 2004[1]. Høsten 2004 ble det utarbeidet en spesifikasjon for NRWD[17]. Våren 2005 pågår det to masteroppgaver som jobber

videre med prosjektet. Den ene tar for seg maskinvaren til protesen[2], mens denne er den andre som tar for seg programvaren. Sommeren/høsten 2005 er det planlagt å ha en ferdig prototyp av NRWD.

Denne oppgaven er da delen som omtales som programvare i Figur 3-1.

3.1.2 Maskinvare

I Figur 3-2 ser vi en forenklet modell av maskinvaren. En nøyaktig versjon av dette



Figur 3-2: Forenklet skisse av maskinvaren

finnes i Håkon Skjeltens hovedoppgave[2]. AVR'en skal altså kunne ta inn og sende ut både analogt og digitalt. Digitalt i form av busser, CAN og I2C. Videre skal det være mulig å koble til en pc via terminal for testing og omprogrammering av protesen. En ATTiny er videre koblet til motorstyringen.

3.1.3 Spesifikasjon

Det ble utarbeidet en spesifikasjon til NRWD i form av en prosjektoppgave høsten 2004[17]. Denne ble senere videreutviklet av Øyvind Stavdahl[18] og komprimert til en tabell. Denne utgjør mye av bakgrunnen for denne oppgaven og også maskinvare-oppgaven. Spesifikasjonen stiller en del bør- og skal-krav til protesen som må tas hensyn til. Det vil ikke bli gjengitt her.

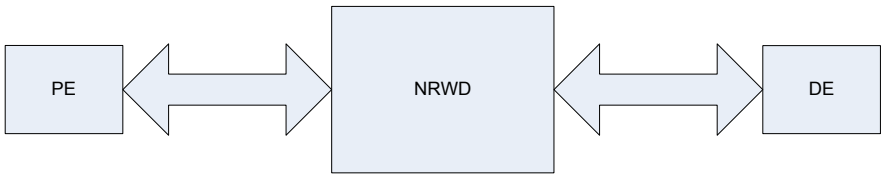
3.1.4 Programvare

Ved starten av oppgaven ble det sjekket ut en del programvare for UML-modellering. Det viste seg at IAR VisualState ville vært et ypperlig verktøy i denne forbindelse. Dette programmet gjør det mulig å kompilere en kode som kan implementeres direkte på en AVR. Desverre viste det seg at NTNU ikke hadde lisens på programvaren og at den er for dyr til å anskaffe ifm. denne oppgaven. Videre ble det besluttet å benytte Rational Rose Realtime siden dette er en programvare som er tilgjengelig på NTNU og det har blitt brukt i undervisningsopplegget tidligere. Men store problemer med installering gjorde dette arbeidet vanskelig. Det ble brukt flere uker på å få løst problemet, men det lot seg ikke fikse. De rette instanser på NTNU ble også koblet inn, uten at en løsning ble funnet. Problemet stod i at programmet ikke klarte kompilere

modellen. All problematikken førte til at programvaren ikke ble benyttet. Å benytte et slikt program kan absolutt anbefales. Dette gjør styresystemet veldig oversiktlig og endringer er lette å ta. Men som sagt er prisen på IAR VisualState veldig høy, så uten en allerede eksisterende lisens kan dette absolutt bli litt for flott.

4 Styresystem

NRWD er tiltenkt å kunne spille en rolle i et modulert protesesystem. Den vil da være en slave for en overordnet master. I tillegg skal håndleddet være i stand til å styre en eventuell analog hånd. Dette overordnede systemet er vist i figuren under.



Figur 4-1: Overordnet system, modulært system

PE: Proximal Electronics. Henviser til elektronikk som ligger nærmere kroppen.

DE: Distal Electronics. Henviser til elektronikk som ligger fjernere fra kroppen.

4.1 Kravspesifikasjon

Spesifikasjonen blir satt opp etter enheter på modulen og krav gitt i spesifikasjonen[18] for NRWD. Det vil her bli gitt i form av usecases fra UML-terminologien.

- Det skal være mulig å kommunisere med enheten. Dette i form av å sende og motta beskjeder og også kunne lese av signaler under testing av enheten.
- Det skal være mulig å styre en analog hånd via enheten. Enheten må dermed ha en egen kontroll for en eventuell analog hånd der det også må være kommunikasjon begge veier.
- Det skal være mulig å omprogrammere enheten. Dette i form av å kunne skifte mellom de flere mulige modus protesen har. Også i form av å kunne videreutvikle enheten.

4.2 Design

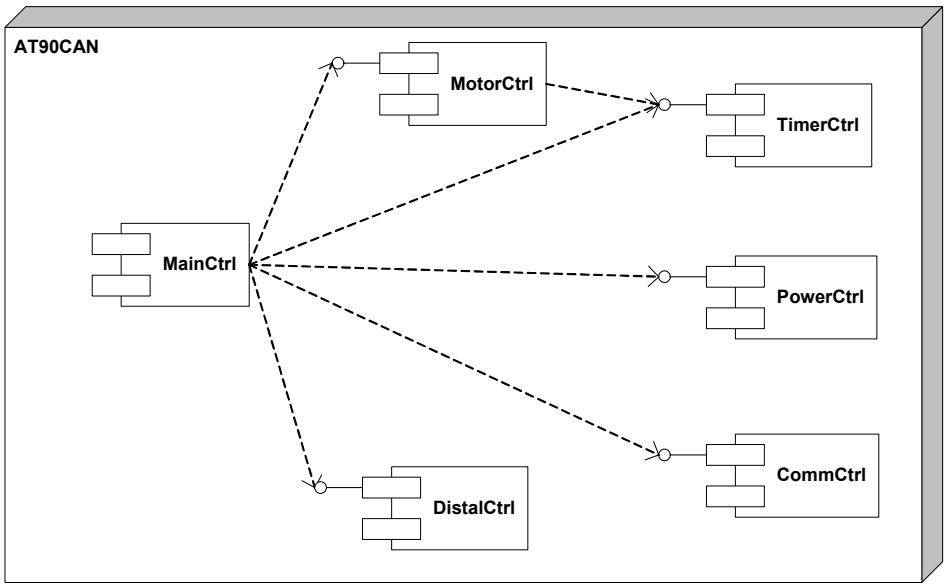
Utfra spesifikasjonen til NRWD har det blitt valgt å dele protesen inn i følgende komponenter:

- **MainCtrl.** Denne komponenten er hovedkontrollen til systemet. Denne tar seg av alt som skjer i protesen.

- **CommCtrl.** Denne komponenten tar seg av all kommunikasjon inn og ut av protesen. Den må dermed ta seg av kontroll over hvilken måte det kommuniseres på enten det er I2C, CAN eller analogt.
- **MotorCtrl.** Denne komponenten styrer selve motoren. Den holder orden på pådrag og retning på motoren, og også hastighet, strømforbruk og posisjon på motoren. Her finner man også de forskjellige kjøremodiene til protesen.
- **PowerCtrl.** Denne komponenten holder et øye med batteriets spenningsnivå.
- **TimerCtrl.** Denne komponenten kontrollerer klokkene til enheten.
- **DistalCtrl.** Denne komponenten skal kunne styre en eventuell analog hånd.

Figuren under viser avhengighetene som et komponentdiagram(UML). Siden dette er avhengigheter betyr det ikke at data ikke kan flyte begge veier. Det er ingen sykler i figuren, noe som betyr at ingen av komponentene er avhengige av hverandre. Dette fører til at programdesign og senere vedlikehold blir mye enklere. Ser at MainCtrl er avhengig av alle komponentene siden denne styrer alt som skjer. DistalCtrl vil kunne styre en eventuellt analog hånd, men dette vil holdes utenfor denne oppgaven siden man ikke vet hva som skal styres.

MainCtrl er avhengig av å kunne hente ut spenningsnivået til PowerCtrl. Den vil spørre regelmessig etter dette og er dermed avhengig av TimerCtrl. Den vil også være avhengig av CommCtrl for å få initialisert kommunikasjonskanalene, få meldinger fra PE og eventuellt svare på disse, samtidig som den får beskjed om hvilken kommunikasjonskanal som benyttes. MainCtrl er også avhengig av MotorCtrl for å kunne vite hastighet, posisjon og strømforbruket til motoren samtidig som den setter modus og styresignaler til motoren. MotorCtrl vil regelmessig hente hastighet, posisjon og strømforbruk på motoren og er derfor avhengig av TimerCtrl. MainCtrl vil også kunne begrense DistalCtrl ved lav spenning på batteriet, samtidig som den vil kunne gi den kommandoer.

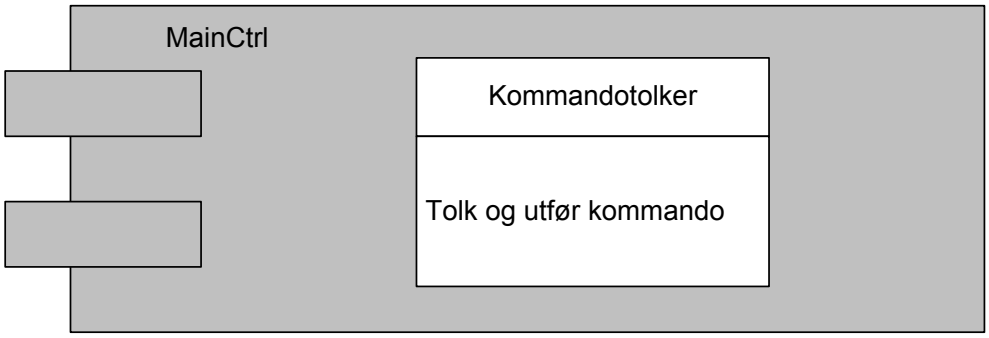


Figur 4-2: Komponentdiagram med avhengigheter

4.2.1 MainCtrl

MainCtrl er som navnet tilsier hovedkontrollen i protesen. Denne skal ha oversikt over alt som foregår i systemet. MainCtrl trenger her følgende klasse:

- **Kommandotolker.** Denne klassen leser inn kommandoer, sjekker at de er gyldige og utfører disse.



Figur 4-3: Klassediagram for MainCtrl

Siden det er her kommandoen skal tolkes er det fornuftig å også implementere kommunikasjonsprotokollen her. Siden denne oppgaven er ment å tilpasses i en prototyp, bør det settes opp en terminal mot protesen, sånn at man enkelt kan kjøre tester og debugge. Derfor er det en fordel at protokollen er tekstbasert. Dette gjør den også lettere å forstå og å sette seg inn i for de som tar seg av videre utvikling av protesen.

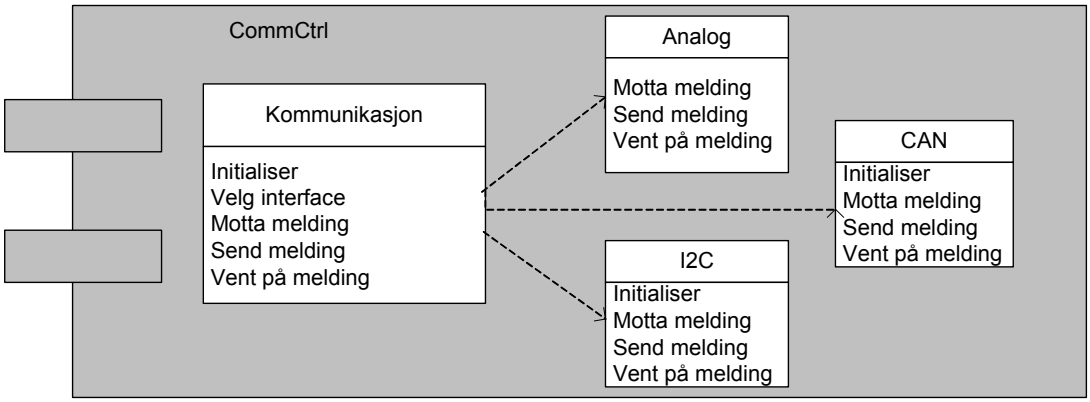
MainCtrl vil også ha kontroll på DistalCtrl hvis en analog hand er koblet til det modulære protesesystemet. MainCtrl vil kunne fortelle PE at en slik protese er tilkoblet og senere gi DistalCtrl de meldinger den skal ha. DistalCtrl vil videre styre dette selv, med en egen MotorCtrl. MainCtrl bør også ha ansvaret for å begrense pådraget til DistalCtrl. Hvis det er lite spenning på batteriet, vil bruk av både håndledd og hånd tappe batteruet raskt og derfor bør MainCtrl ha hovedkontrollen her.

MainCtrl må ha klart hvilken kanal det kommuniseres på. Ved busser kan kommunikasjon med PE sørge for velging av modus osv., men ved analogt er dette ikke mulig. Det bør derfor settes opp en plan for hvordan protesen skal kjøre ved analog kommunikasjon. Dette må gjøres etter ønsker fra brukeren, hvilken modus som ønskes brukt.

4.2.2 CommCtrl

Denne enheten skal ta seg av meldingsflyten inn og ut av AVR'en. Dermed må denne enheten ha følgende klasser:

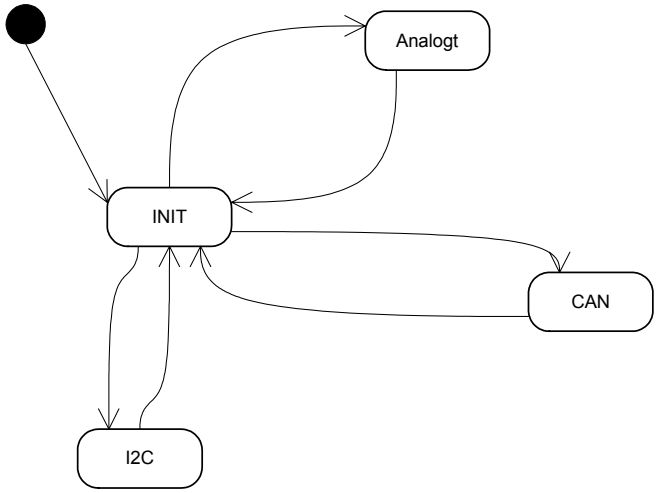
- **Kommunikasjon.** Denne klassen skal kunne sende, motta og vente på meldinger.
- **I2C.** Denne klassen initialiserer bussen og sender, mottar og venter på meldinger fra PE.
- **CAN.** Denne klassen initialiserer bussen og sender, mottar og venter på meldinger fra PE.
- **Analog.** Denne klassen initiliserer kanalen og sender, mottar og venter på meldinger fra PE.



Figur 4-4: Klassediagram for CommCtrl

Det skal være mulig å kommunisere med PE enten via CAN-buss, I2C-buss eller analogt. Når strømmen på protesen settes på, vil MainCtrl be CommCtrl om å initialisere kommunikasjonskanalene. CommCtrl vil her lytte til alle tre kanaler med et visst tidsrom, der den venter på bekreftelse fra PE. Kommer det inn et signal via en buss vil denne bli satt. Etter et forhåndsatt antall tidsskritt uten signal, vil protesen bli satt til å kjøre analogt. Dette kan vises i et tilstandsdiagram som vist i Figur 4-5. Når dette er gjort vil CommCtrl gi beskjed tilbake til MainCtrl om at initialisering er ferdig.

MainCtrl får samtidig beskjed om hvilken kanal det kommuniseres på, slik at den kan styre deretter.

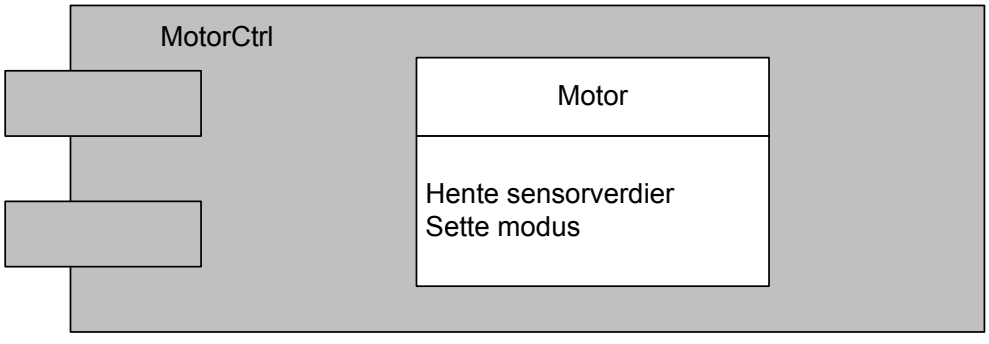


Figur 4-5: Tilstandsdiagram for kommunikasjonskanaler

4.2.3 MotorCtrl

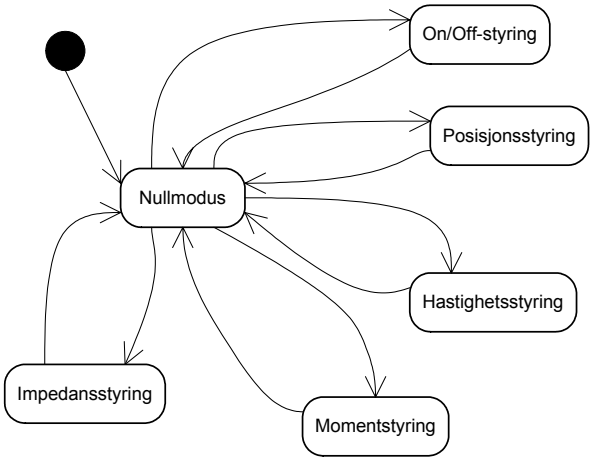
Denne enheten inneholder de ulike modus og henter ut verdier fra sensorene. Her har vi følgende klasse:

- **Motor.** Denne klassen har funksjoner som henter inn posisjon, hastighet og strømforbruket til motoren. Klassen har også oversikt over hvilken modus protesen kjører på.



Figur 4-6: Klassediagram for MotorCtrl

MotorCtrl inneholder funksjoner som kan hente ut motorens hastighet, posisjon og strømforbruk. Disse verdiene blir hentet ut regelmessig etter forhåndssatte tidsskritt. Videre vil de bli brukt i regulatorene motoren må ha for de forskjellige modusene. Motorens modus vises i tilstandsdiagrammet i Figur 4-7.



Figur 4-7: Tilstandsdiagram for modus

Nullmodus. Denne modusen har null i pådrag til enhver tid. Det vil si at ingenting skjer med motoren i denne modusen. Protesen vil automatisk stå i denne modusen helt til andre beskjeder blir gitt.

On/Off-styring. Denne modusen kjører motoren med settpunktet direkte uten regulering.

Posisjonsstyring. Denne modusen bruker tilbakekobling fra posisjonen til å flytte protesen til valgt posisjon. Pådraget må også reguleres slik at protesen får en mest mulig jevn(naturlig) bevegelse.

Hastighetsstyring. Denne modusen bruker tilbakekobling fra hastighet til å flytte protesen etter ønsket hastighet. Også her må pådraget reguleres slik at protesen får en jevn bevegelse.

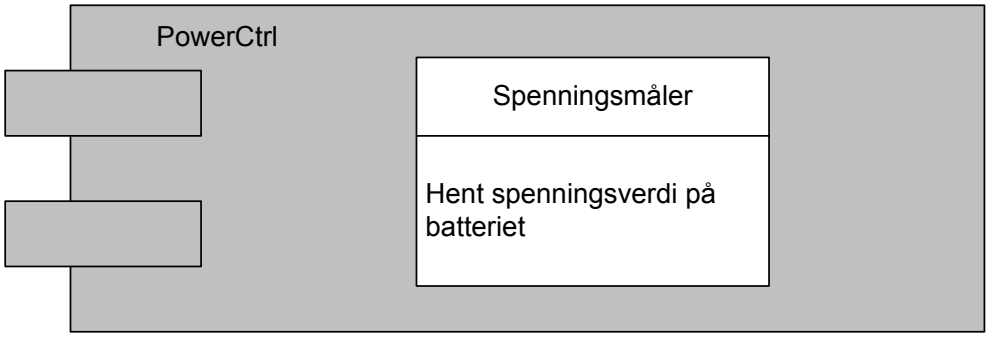
Momentstyring. Denne modusen bruker tilbakekobling fra strømforbruket i motoren til å drive protesen med ønsket moment.

Impedansstyring. Denne modusen bruker tilbakekobling fra hastighet og posisjonen til motoren til å drive protesen med ønsket impedans.

4.2.4 PowerCtrl

Denne enheten tar seg av måling av spenningen på batteriet. Den har følgende klasse:

- **Spenningsmåler.** Denne klassen må ha funksjoner for å hente ut spenningsnivået til batteriet.



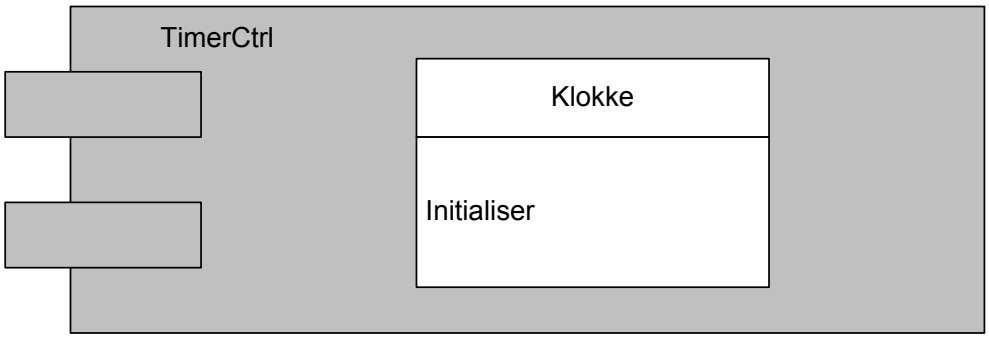
Figur 4-8: Klassediagram for PowerCtrl

Når forespurt om spenningsnivå på batteriet skal denne klassen automatisk sende verdien tilbake til MainCtrl.

4.2.5 TimerCtrl

Denne enheten tilbyr en stabil tidsangivelse til systemet. Den inneholder en klasse:

- **Klokke.** Denne klassen inneholder en funksjon for å initialisere klokka.



Figur 4-9: Klassediagram for TimerCtrl

Både MainCtrl og MotorCtrl er avhengig av en klokke for å hente ut forskjellige verdier ved faste tidsskritt. Den skal også kunne synkroniseres med et klokkene i et modulært protesesystem.

4.2.6 DistalCtrl

Denne enheten er lagt inn for å styre en eventuell analog hånd. Hva som skal skje i denne enheten blir ikke utgreid i denne oppgaven, da man ikke vet hva det er som skal styres. Det eneste som blir klargjort er at MainCtrl har funksjoner for å vite om det er en eventuell hånd koblet til. I tillegg er MainCtrl avhengig av å kunne begrense DistalCtrl ved en PowerSave modus. Dette må gjøres for å unngå at begge proteser kjører samtidig og bruker opp for mye av batteriet samtidig.

4.3 Implementering

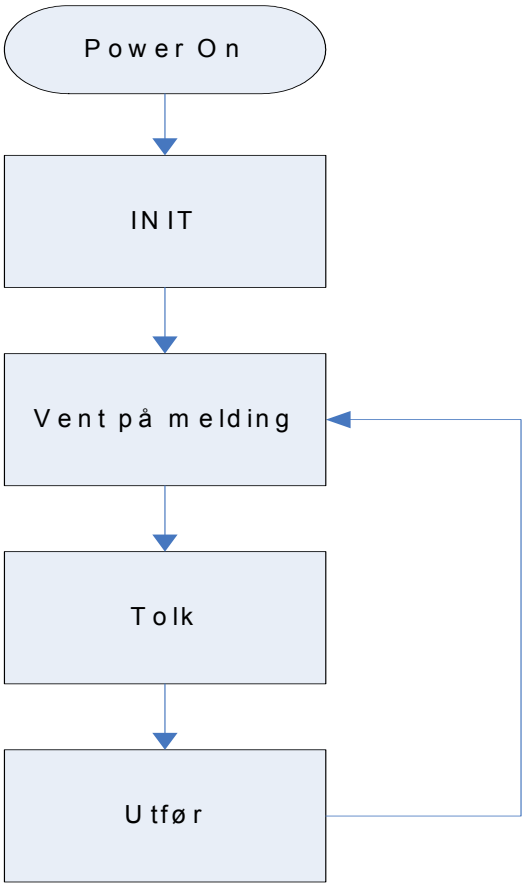
Dette kapittelet vil ta for seg hvordan styresystemet skal implementeres. Det vil her bli forklart hva som skal skje i hver modul, og hvordan dette kan gjøres. Manglende informasjon om kommunikasjonsprotokoll for et modulært protesesystem gjør at systemet ikke er fullstendig. På samme måte mangler en del informasjon angående den ferdige prototypen som gjør at endel verdier må settes senere.

4.3.1 MainCtrl

Hovedløkken til styresystemet vil ligge i denne modulen. Den er vist i Figur 4-10. Idet strømmen blir slått på vil initialisering av all nødvendig periferi starte. Det vil bli gitt kommandoer til alle moduler som er avhengig av initialisering av å starte dette. Hvis systemet kjører analogt vil protesen gå i en forhåndsbestemt modus, der retning og et settpunkt til motoren vil sendes rett til valgt regulator. Ellers vil MainCtrl få beskjed tilbake om at dette er ferdig. Deretter vil MainCtrl vente på et interrupt som tilsier at en melding har kommet inn i AVR's databuffer fra CommCtrl. MainCtrl henter så meldingen for deretter å tolke meldingen og utføre det meldingen tilsier. Når dette er utført, venter MainCtrl på nye meldinger.

MainCtrl må også inneholde en funksjon som sjekker spenningsnivået på batteriet. Det vil med fast tidsintervall sjekkes og MainCtrl avgjør hvilken modus protesen skal kjøre, Normal eller PowerSave. Det gis beskjed til MotorCrtl som styrer dette. Ved PowerSave vil det settes begrensninger på pådraget til motoren, og ved en eventuell analog hånd må det settes opp en algoritme som sikrer at begge motorer ikke kjører samtidig og tapper batteriet helt. Ved at en motor kjører av gangen blir det lettere for brukeren å unngå tomt batteri som tilsier ingen bruksverdi av protesen.

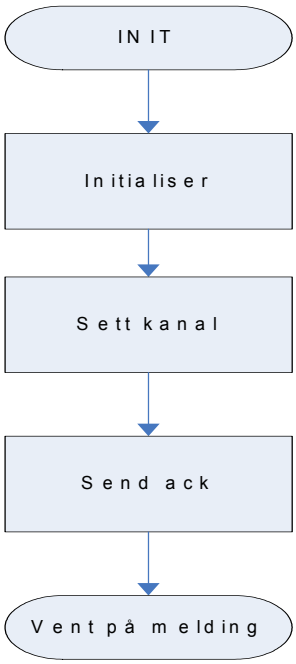
Hvordan begrensningene skal settes er noe som må avgjøres etter en del testing på en prototyp. Dette gjelder også ved hvilke tidskritt de forskjellig målinger skal gjøres.



Figur 4-10: Flytdiagram for hovedløkken

4.3.2 CommCtrl

CommCtrl er grensesnittet mot annen elektronikk. Ved oppstart fås det beskjed om å



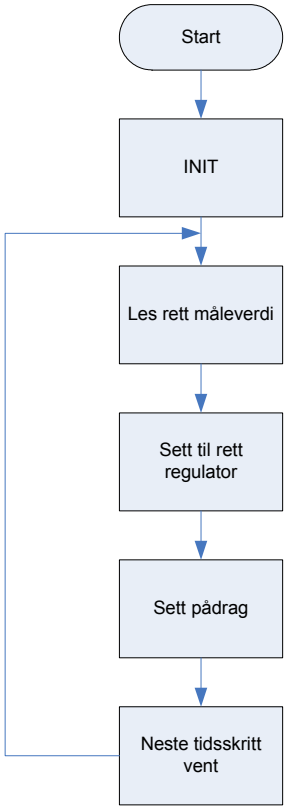
Figur 4-11: Flytdiagram for CommCtrl

initialisere. CAN og I2C bussene blir da initialisert og det må lyttes på de to bussene med et fast intervall. Finner man aktivitet på en av bussene, settes denne bussen til kommunikasjonskanal. Hvis man ikke finner en aktiv buss etter et gitt tidspunkt, vil den analoge kanalen automatisk bli satt. Deretter sendes det bekreftelse til MainCtrl om at initialiseringen er ferdig og hvilken kanal det kommuniseres over. Deretter venter CommCtrl på innkommende meldinger. CommCtrl må implementeres med funksjoner for å sende, motta og vente på meldinger. Når en melding kommer inn legges denne i AVR'ens databuffer. Så settes det en interrupt med et flag slik at MainCtrl vet at en melding har kommet inn. Ved beskjed fra MainCtrl vil også CommCtrl kunne sende beskjeder til PE.

4.3.3 MotorCtrl

I MotorCtrl er det styring av selve motoren som er hovedoppgaven. Dette vil følge flytdiagrammet i Figur 4-12. Det vil ved oppstart bli initialisere nødvendig periferi før

løkken starter opp. Løkken vil lese inn de rette måleverdiene, sette rett regulator og



Figur 4-12: Flytdiagram for MotorCtrl

sette pådrag ved hvert tidsskritt. Hvilken modus protesen er i vil avgjøre hva som gjøres i løkken ved hvert tidskritt.

Det enkleste måten å implementere setting av regulator og måleverdier er ved å bruke pekere. Når MotorCtrl får beskjed om å gå i en bestemt modus vil det automatisk bli satt pekere til rette regulatorer og hvilken måleverdier som skal hentes inn. Dette gjør det også enkelt å utvide programvaren med nye regulatorer og måleverdier i ettertid. Modusene må inneholde følgende:

Nullmodus. Ved oppstart av protesen vil denne modusen automatisk bli satt. Denne modusen gir ut null i pådrag ved enhver tid og ingen peker er her satt til måleverdier. Regulatorpekeren peker til en nullregulator som gir ut 0 i pådrag til en hver tid. Uavhenigig av hvilke kommandoer MotorCtrl får angående styring av protesen, vil den stå stille.

On/Off modus. Ved innkommende kommando om styring av motoren vil det bli satt en peker til On/Off regulatoren som har en forsterkning på 1, dvs si den sender signalet rett gjennom uten regulering. Motoren vil her kjøre med maks hastighet i valgt retning til ingen flere kommandoer kommer.

Posisjonsmodus. Ved innkommende kommando om styring av motoren vil det bli satt en peker til posisjonsregulatoren og en peker til posisjonsmålingen. Motoren vil dermed flytte seg til ønsket posisjon.

Hastighetsmodus. Ved innkommende kommando om styring av motoren vil det bli satt en peker til hastighetsregulatoren og en peker til hastighetsmålingen. Motoren vil her flytte seg etter ønsket hastighet.

Momentmodus. Ved innkommende kommando om styring av motoren vil det her bli satt en peker til momentregulatoren og en peker til målingen av motorens strømforbruk. Motoren vil her flytte seg med ønsket moment.

Impedansmodus. Ved innkommende kommando om styring av mototren vil det her bli satt en peker til impedansregulatoren og pekere til hastighetsmålingen og motorens strømforbruk.

Felles for alle modus er at pådraget settes til null i starten som de vil holde til en kommando om styring kommer inn. Etter at kommandoen er utført settes pådraget igjen til null i avvente på ny kommando.

Regulatorene som skal implementeres for NRWD er følgende:

Nullregulator. Tar ikke inn noen måleverdier. Pådrag alltid satt til null.

On/Off regulator. Tar ikke inn noen måleverdier. Pådrag settes til maksimal hastighet i ønsket retning. Regulering i form av brukers synsobservering. Ved PowerSave må det settes begrensninger på pådraget. Disse bør settes som resultat av testing på protesen.

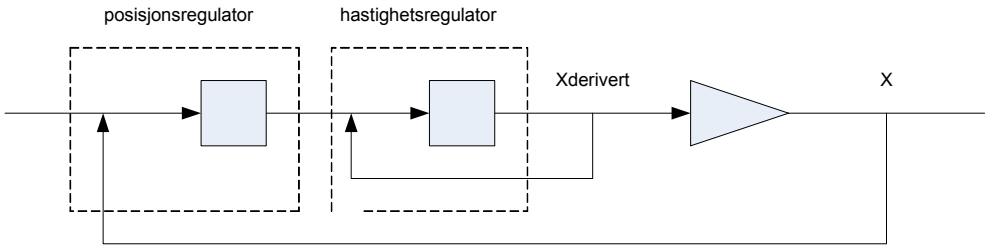
Posisjonsregulator. Tar inn posisjonsmåling. Flytter motoren til ønsket posisjon i ønsket retning. Ved PowerSave må det settes begrensninger på pådraget. Disse bør settes som resultat av testing på protesen.

Hastighetsregulator. Tar inn hastighetsmåling. Flytter motoren i ønsket hastighet i ønsket retning. Ved PowerSave må det settes begrensninger på pådraget. Disse bør settes som resultat av testing på protesen.

Momentregulator. Tar inn strømforbruket til motoren. Flytter motoren med ønsket moment i ønsket retning. Ved PowerSave må det settes begrensninger på pådraget. Disse bør settes som et resultat av testing på protesen.

Impedansregulator. Tar inn hastighetsmåling og strømforbruket til motoren. Flytter motoren med ønsket mekanisk impedans i ønsket retning. Ved PowerSave må det settes begrensinger på pådraget. Disse bør settes som et resultat av testing på protesen.

Når en prototyp er ferdigstillt kan arbeidet med innstillinger av regulatorer starte. Dette arbeidet avhenger veldig av oppbygning av protesen, spesiellt med tanke på gir, tyngde og størrelse. Videre er det også mulighet for at de enkelte regulatorene kan sette pekere til hverandre slik at kaskaderegulering kan benyttes. Ved bruk av pekere kan dette lett utvides hvis ønskelig. Et eksempel på dette vises i Figur 4-13 der regulatoren henter inn både posisjon og hastighet og regulerer motoren etter dette. Ved i tillegg å legge til måleverdien til strømforbruket til motoren i en indre sløyfe kan meget god regulering oppnås. Det er ikke satt på minustegn på tilbakekoblingene i figuren, da dette er opp til design av regulatorer. Figuren skal kun vise at verdiene tilbakekoblet til regulatorene.



Figur 4-13: Skisse for kaskaderegulering

4.3.4 PowerCtrl

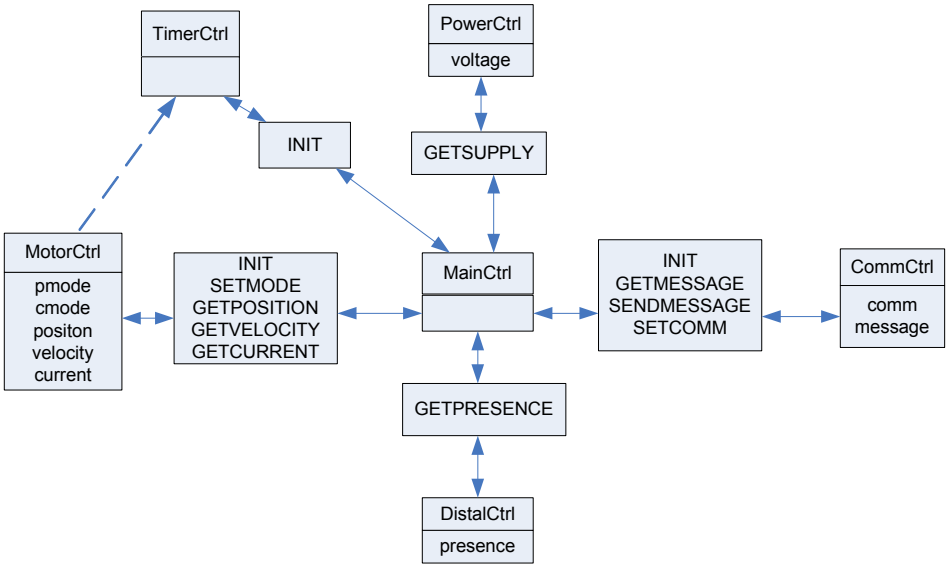
Denne modulen har kun en funksjon, og den henter ut spenningsnivået til batteriet. Ved kommando fra MainCtrl hentes verdien på spenningsnivået og sendes tilbake til MainCtrl.

4.3.5 TimerCtrl

Denne modulen er systemets klokke. Klokkeverdien trengs av MainCtrl og MotorCtrl for regelmessig å hente verdier fra forskjellige instanser. Dette gjelder spenningsnivå på batteri og hastighet, posisjon og strømforbruket til motoren. Er systemet koblet til et modulært styresystem for flere proteser, vil det være mulig å synkronisere alle klokkene i systemet.

4.3.6 Totalt system

Arbeidet over leder til et system som vist i Figur 4-14. Den viser modulene med deres variable. De er knyttet sammen via protokoller som inneholder kommandoer som blir kalt mellom modulene. Det stiplede linjen i figuren antyder at MotorCtrl trenger å vite tellerverdier for når den skal hente ut de forskjellige måleverdiene.



Figur 4-14: Enkel skisse av styresystemet

4.4 Kommunikasjonsprotokoll

Kommunikasjonsprotokollen som må implementeres i MainCtrl skal inneholde alle kommandoer brukt i systemet. I tillegg må det også utarbeides en protokoll for bruk mellom de mulige enheten i en modulær arm. Dette vil ikke bli gjort i denne oppgaven, men det vil bli satt opp en rekke forslag til hva den bør inneholde for at håndledd skal kunne fungere optimalt.

4.4.1 Kommunikasjonsprotokoll for håndleddsprotesen

Dette er protokollen som implementeres i MainCtrl og inneholder alle kommandoene brukt i systemet. Da en del av disse er utarbeidet av Håkon Skjelten i hans maskinvare-oppgave[2], blir det her bare satt opp deler av den fulle protokollen. Dette må samkjøres ved ferdigstilling av prototyp. Her settes kun inn de som er nødvendige for at styresystemet skal bli forståelig.

GETSUPPLY - hente spenningsverdi på batteriet. Denne kommandoen ber om å få spenningsverdien på batteriet. Denne verdien hentes ut ved faste tidskritt.

Tabell 4-1: Detaljer for GETSUPPLY

	Felt	Beskrivelse
Kommando:		
`GETSUPPLY`	`ingen`	
Respons:		
`GETSUPPLY`	`voltage`	spenningsnivået på batteriet

SETMODE - setter kjøremodus til protesen. Denne kommandoen brukes for å sette de forskjellige modus protesen kan kjøres i.

Tabell 4-2: Detaljer for SETMODE

	Felt	Beskrivelse
Kommando:		
SETMODE	pmode	"0" hvis protesen kjører uten begrensninger på pådraget, "1" hvis den kjører med begrensninger.
	cmode	"null" for nullmodus, "onoff" for ON/OFF-modus, "pos" for posisjonsmodus, "vel" for hastighetsmodus, "torque" for momentmodus og "imp" for impedansmodus
Respons:		
SETMODE	ingen	

GETPOSITION - hente posisjon til motoren. Denne kommandoen henter ut posisjonen til motoren.

Tabell 4-3: Detaljer for GETPOSITION

	Felt	Beskrivelse
Kommando:		
GETPOSITION	ingen	
Respons:		
GETPOSITION	position	posisjonen til motoren

GETVELOCITY - hente hastigheten til motoren. Denne kommandoen henter ut hatigheten til motoren.

Tabell 4-4: Detaljer for GETVELOCITY

	Felt	Beskrivelse
Kommando:		
GETVELOCITY	ingen	
Respons:		
GETVELOCITY	velocity	hatigheten til motoren

GETCURRENT - hente spenningsforbruket til motoren. Denne kommandoen henter ut spenningsforbruket til motoren.

Tabell 4-5: Detaljer for GETCURRENT

	Felt	Beskrivelse
Kommando:		
GETCURRENT	ingen	
Respons:		
GETCURRENT	current	strømforbruket til motoren

SETCOMM - setter kommunikasjonskanal. Denne kommandoen setter kommunikasjonskanal for protesen.

Tabell 4-6: Detaljer for SETCOMM

	Felt	Beskrivelse
Kommando:		
SETCOMM	comm	"CAN" for Can-buss, "I2C" for I2C-buss og "analog" for analogt
Respons:		
SETCOMM	ingen	

INIT - initialisering av nødvendig periferi. Denne kommandoen kjører ved oppstart og ber alle instanser initialisere. De forskjellige modulene vil her sende en bekreftelse til MainCtrl om at initialiseringen er ferdig. CommCtrl vil i tillegg gi beskjed hvilken kommunikasjonskanal som benyttes og MotorCtrl vil gi beskjed hvilken kjøremodus den står i.

Tabell 4-7: Detaljer for INIT

	Felt	Beskrivelse
Kommando:		
INIT	ingen	
Respons:		
INIT	ack	initialisering ferdig

GETPRESENCE - sjekker om analog hånd. Denne kommandoen sjekker med DistalCtrl om en analog hånd er tilkoblet.

Tabell 4-8: Detaljer for GETPRESENCE

	Felt	Beskrivelse
Kommando:		
GETPRESENCE	ingen	
Respons:		
GETPRESENCE	presence	"1" hvis tilstede, "0" hvis ikke.

GETMESSAGE - hente innkommende melding fra buffer. MainCtrl henter en innkommet meldig for å tolke denne.

Tabell 4-9: Detaljer for GETMESSAGE

	Felt	Beskrivelse
Kommando:		
GETMESSAGE	ingen	
Respons:		
GETMESSAGE	message	melding lagret i bufferet.

SENDMESSAGE - gi beskjed til CommCtrl om å sende melding. MainCtrl gir CommCtrl beskjed om å sende en melding ut av systemet. Dette kan også gjelde videresending av meldinger til DistalCtrl.

Tabell 4-10: Detaljer for SENDMESSAGE

	Felt	Beskrivelse
Kommando:		
SENDMESSAGE	message	melding som skal sendes videre.
Respons:		
SENDMESSAGE	ingen	

4.4.2 Kommunikasjonsprotokoll for modulært armsystem

Denne protokollen skal kunne gjelde for alle protesesystemer som blir koblet til armsystemet. Dette arbeidet har startet, men er langt fra ferdig. Det er derfor ikke hensiktsmessig å lage en slik for denne oppgaven, siden den mest sannsynlig vil bli endret i senere tid. Men det settes her opp en del forslag om hva denne protokollen må kunne inneholde mhp. håndleddet. Dette gjelder ved bruk av busser som kommunikasjonskanal. Når denne er utarbeidet, må den så implementeres i MainCtrl slik at håndleddet snakker samme språk som PE.

Det må sendes ut bekreftelse på mottatte meldinger slik at alt fungerer som det skal. Brukeren må forsikres om at det han ønsker blir gjort.

Sjekke om det er liv i håndleddet. Det må finnes en kommando for å finne ut om et håndledd er koblet til og om dette er operativt. På denne måten slipper man å bruker båndbredde på bussen ved å sende ut meldinger som ingen har bruk for. Det kan også være hensiktsmessig å sende ut denne meldigen med jevne mellomrom. På denne måten vet man til enhver tid om håndleddet er koblet til. Dette kan også benyttes av håndleddet til å avgjøre hvilken kommunikasjonskanal som benyttes. Ved å lytte på sine to busser ved initialisering avgjør håndleddet hvilken kanal som benyttes. Kommer en kommando som spør om protesen er der, vet håndleddet hva det kommuniseres over.

Sette kommunikasjonskanal. Ved eventuell endring av kommunikasjonskanal må dette videreformidles til enhetene i det totale systemet.

Sette styremodus på motoren. Det må kunne gis beskjed til håndleddet om å gå i en styremodus. Dette bør sendes ut en gang for hver endring av modus, og ikke være inkludert i starten av hver styrekommando. Dette er redundant siden man ved hver kommando bruker båndbredde på å sette modus. Ved utsending av modus vil håndleddet stå i denne modus til annen beskjed blir gitt.

Gi styresignaler til håndleddet. Det må utarbeides en måte å gi håndleddet beskjed om hvordan det skal flytte seg. Som nevnt over blir modus satt først for å unngå redundant meldingsflyt. Videre må det gis beskjed om hvordan protesen skal flytte seg i den forhåndsvalgte modusen.

Hentet ut ønsket informasjon. Det må være mulig for PE å hente ut verdier fra håndleddet, kanskje spesielt når det gjelder posisjon og modus. Ved eventuelle feil kan PE ha mistet oversikt over hvilken modus håndleddet kjører i og det kan på denne måten bli sendt ut ugyldige kommandoer. Ved å kunne hente ut modus kan slike feil rettes opp. Det kan også være ønskelig for PE å vite posisjon, hastighet, stømforbruk på motor, spenningsnivå på batteriet osv.

Synkronisere klokker. Det må finnes kommandoer som synkroniserer klokkene i totalsystemet slik at alle enheter opererer likt.

Sjekke om anaolg hånd er tilkoblet. Hvis en analog hånd er tilkoblet håndleddet må PE kunne vite om dette for å sende ut styresignaler for hånden. Dette kan igjen hindre unødvendig bruk av båndbredde.

Gi styresignaler til analog hånd. Hvis en analog hånd er koblet på må den kunne gi håndleddet beskjed om hvordan denne skal styres i form av retning og styrke.

5 Konklusjon

Det viste seg veldig vanskelig å konstruere en fungerende programvare for NRWD på grunn av mangel på informasjon. Siden verken maskinvare eller mekanisk konstruksjon var ferdig ble det vanskelig å kunne teste ut programvare. Det ble derfor lagt vekt på å klarlegge systemet slik at det enkelt kan implementeres ved ferdigstilling av de ovenfor nevnte delene.

Det ble også veldig vanskelig å kunne ferdigstille en kommunikasjonsprotokollen, da det modulære systemet NRWD var tiltenkt å skulle være en del av, ikke var særlig definert. Det ble derfor i stedet satt opp en del punkter som håndleddets programvare er avhengig av for å kunne fungere optimalt. Dette gir igjen en del huller i programvaren.

Oppgaven gikk ut på å starte implementasjon av styresystemet. Dette arbeidet ble påbegynnt, men fikk en rask slutt. Med så mye manglende informasjon, ble det ingen sammenheng i koden. Det ble derfor lagt til side. For en person som skal sluttføre systemet er det enklere å starte fra grunnen av med oppbygningen og ikke ha noen deler her og noen deler der. Måten personer implementerer på er svært forskjellig og derfor overlates alt til videre utvikling.

6 Videre arbeid

Det gjenstår en god del arbeid før en prototyp er ferdigstillt. Når maskinvaren er ferdig og en protyp konstruert, kan man begynne ferdigstillingen. Videre er man avhengig av en kommunikasjonsprotokoll for det modulære systemet NRWD er tiltenkt å være en del av. Det er uansett mulig å gjøre unna en del arbeid før dette skjer.

Programvaren må først og fremst samkjøres med maskinvaren. Dette innebærer samkjøring av funksjoner og funksjonskall. Deretter kan testing av disse kjøres. Videre må det bestemmes hvor ofte rutinemessige funksjonskall skal kjøres. F.eks. hvor ofte skal MainCtrl kontrollere spenningsnivået på batteriet.

I påvente av kommunikasjonsprotokoll for det modulære systemet, kan det lages en enkel, midlertidig protokoll som gjør det mulig å teste kjøring av protesen. Det blir da mulig å måle verdier på motoren, som igjen gjør det mulig å kunne starte design og instilling av regulatorer. Det vil da også kunne bestemmes hvor ofte man henter ut måleverdier fra motoren.

Ved slutten av ferdigstillingen av prototypen bør det implementeres en alternativ kommunikasjonsprotokoll som ikke er tekstbasert. Dette vil gjøre kommunikasjon med eksterne enheter mer effektivt.

Litteraturliste 7

[1] Øyvind Stavdahl, Optimal Wrist Prosthesis Kinematics. (Doktoravhandling, NTNU, 2002).

[2] Håkon Skjelten, Innvevd maskinvare for kybernetisk håndledd. (Masteroppgave, NTNU, 2005).

[3] Samuel W. Alderson, Shoulder harness for artificial arms. US Patent 2,592,842. (1952).

[4] Christian Almström, En electronic control system for a prosthetic hand with six degrees of freedom. Reaserch Laboratory of medical electronics, Chalmers University of technology, Göteborg, Sweden, 1977.

[5] Peter J. Kyberd, Owen E. Holland, Paul H. Chappell, Simon Smith, Robert Tregidgo, Paul J. Bagwell, Martin Snaith, MARCUS: A Two Degree of Freedom Hand Prosthesis with Hierarchical Grip Control. *IEEE Transactions on Rehabilitation Engineering,* Vol. 3, No. *1*, March 1995

[6] P. J. Kyberd, J. L. Pons, A Comparison of the Oxford and Manus Intelligent Hand prostheses. *Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan, September 14-19, 2003*

[7] M. C. Carrozza, P. Dario, F. Vecchi, S. Roccella, M. Zecca, F. Sebastiani, The CyberHand: on the design of a cybernetic prosthetic hand intended to be interfaced to the peripheral nervous system. *Proceedings of the 2003 IEEE/RSJ Intl. Conference on intelligent Robots and Systems, Las Vegas, Nevada, October 2003.*

[8] D. J. Gow, W. Douglas, C. Geggie, E. Monteith, D. Stewart, The Development of the Edinburgh modular arm system. Rehabilitation Engineering Services, Princess Margaret Rose Orthopaedic Hospital, Edinburgh, Scotland, UK.

[9] P. I. Branemark, Osseointegration: Biotechnological Perspective and Clinical Modality. Chapter 1 i *Osseointegration in Skeletal Reconstruction and Joint Replacement*, Eds., Branemark, et.al., Quintessence Books, pp. 1-24, 1997.

[10] D. C. Simpson, The choice of control system for the multimovement prosthesis: extended physiological proprioception (e.p.p.). In *Proceedings, 4th International Symposium on External Control of Upper-Extremity Prostheses and Orthoses,* p. 146-150.

[11] Yiorgos A. Bertos, Craig W. Heckathorne, Richard F. ff. Weir, Dudley S. Childress, Microprocessor based e.p.p. position controller for electric-powered upper-limb prosthesis. *Proceedings - 19th International Conference - IEEE/EMBS Oct. 30 - Nov. 2, 1997 Chicago, IL. USA*

[12] R. Reiter, Eine neue Elektrokunsthand. *Grenzgebiete der Medizin*, 1(4): 133-135.

[13] R. F. ff. Weir, P. R. Troyk, G. DeMichele, T. Kuiken, Implantable Myoelectric Sensors (IMES) for Upper-extremity Prosthesis Control - Preliminary Work. *Proceedings of the 25th Annual International Conference of the IEEE EMBS, Cancun, Mexico, September 17-21, 2003.*

[14] J. Silva, T. Chau, A. Goldenberg, MMG-Based Multisensor Data Fusion for Prosthetic Contol. *Proceedings of the 25th Annual International Conference of the IEEE EMBS, Cancun, Mexico, September 17-21, 2003.*

[15] P. J. Kyberd, P. H. Chapell, The Southampton Hand: an intelligent myoelectric prosthesis. *Rehabil Res Dev. Nov 1994, 31(4), pp. 326-334.*

[16] www.mala.bc.ca/~soules/medi402/brown/cyborg.htm

[17] Rainer Storheil, Optimal elektrisk håndleddsprotese med en frihetsgrad. (Prosjektoppgave, NTNU, 2004

[18] Øyvind Stavdahl, *Functional Requirements Specification for the NTNU Revolute Wrist Device (NRWD) v0.0.* (Teknisk kybernetikk, NTNU, Trondheim)