NTNU
Det skapende universitet

Lab-oppsett for proteseforskning

Ådne Solhaug Linnerud

Master i teknisk kybernetikk
Innlevert: Juni 2012
Hovedveileder: Øyvind Stavdahl, ITK
Medveileder: Anders Fougner, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

Masteroppgave

Studentens navn: Ådne Solhaug Linnerud

Fag: Teknisk kybernetikk

Tittel (norsk): Lab-oppsett for proteseforskning

Tittel (English): Laboratory set-up for prosthesis research

Beskrivelse:

Instituttet vil bygge en autentisk armprotese med motorisert håndledd og gripefunksjon basert på kommersielle komponenter fra produsenten *Motion Control*. Protesen skal benyttes til å utvikle et proporsjonalt styresystem basert på moderne sensor- og treningskonsepter. I denne oppgaven skal du utrede funksjonelle krav til systemet, implementere enkelte deler av det, og evt. gjennomføre et begrenset pilotprosjekt for å evaluere systemets egnethet.

1. Beskriv konseptet «proteseguidet trening» som ble innført av *Blair Lock* et al. i 2011. Utred forskjeller og likheter på hhv. klassiske mønstergjenkjenningssystemer og proporsjonale systemer når det gjelder systemtrening. Foreslå om mulig noen prinsipper for bruk av proteseguidet trening alene eller i kombinasjon med andre teknikker i proporsjonale systemer.
2. Gjør begrunnede valg av løsninger basert på resultatene fra punkt 1, og ta fram en funksjonsspesifikasjon for det aktuelle systemet i samråd med veileder.
3. Beskriv alternative løsninger for å kombinere *Motion Control*s komponenter med ITKs eksisterende programvarebibliotek for mønstergjenkjenning og styring som et ledd i å realisere funksjonsspesifikasjonen. Løsningen skal være mest mulig robust og fleksibel, og må kunne fungere i sann tid.
4. Implementer og evaluer utvalgte deler av systemet. Prioriter deler som er spesielt kritiske for den tiltenkte bruken.

Faglærer: 1. aman. Øyvind Stavdahl
Faglig veileder: PhD-kandidat Anders L. Fougner

Trondheim, 16.12.2011

Øyvind Stavdahl

Sammendrag

Institutt for Teknisk Kybernetikk(ITK) ved NTNU har et programvarebibliotek for styring av proteser. Dette biblioteket består av et mønstergjenkjenningssystem som kan brukes til å styre en robothånd. Mønstergjenkjenning gir protesebrukeren mulighet til å styre en protese, men uten den grad av kontroll som er ønsket. Derfor er det i denne oppgaven blitt implementert et proporsjonalt styringssystem som skal gjøre at brukeren kan kontrollere en protese bedre. For å trene opp et slikt system er det utviklet metoder som skal brukes sammen med proteseguidet trening. Det er også utviklet treningssett som brukes for å lagre referanseverdier. For å vise at dette kan brukes på proteser, ble kontroll av en protese fra Motion Control implementert inn i ITKs bibliotek. Resultatene indikerer at dette fungerer bra, men at det å kontrollere flere frihetsgrader simultant er vanskelig. Derfor må det utvikles nye treningssett som kan forbedre kontroll av flere frihetsgrader simultant.

Abstract

Department of Engineering Cybernetics(ITK) at NTNU has a library for controlling prostheses. This library consists of a pattern recognition system which can be used to control a robotic hand. Pattern recognition provides a prosthetic user the possibility to control a prosthesis, but without the desired level of control. This report describes the implementation of a proportional control system which should result in better control of the prostheses. To train such a system there have been developed different methods that works alongside prostheses guided training. It is also developed training sets which is used to create reference values for this training. To show that this system works on prostheses, it was developed control of a prosthesis delivered by Motion Control into ITK's library. Results indicate that this works well, but control of multiple degrees of freedom simultaneously is difficult. New training sets should be developed to strengthen the control of multiple degrees of freedom simultaneously.

Forord

Denne oppgaven er et resultat av min mastergrad ved Teknisk Kybernetikk på Norges teknisk-naturvitenskapelige universitet (NTNU). Masteroppgaven har vært både krevende og utfordrende med mye venting på deler. Men alt i alt er jeg fornøyd med det resultatet som er blitt lagt frem.

Jeg vil rette en takk til Andres Fougner for god veiledning og godt samarbeid. Jeg også takke Øyvind Stavdahl som har kommet med innspill når det har vært nødvendig og for at jeg ble akseptert til denne oppgaven.

Samtidig vil jeg takke mamma, pappa og min søster for støtte gjennom arbeidet med oppgaven. Takk til mine medstudenter; Asbjørn G. Klausen som diskusjonspartner og Trond Suleng for selskap på laben.

Ådne Solhaug Linnerud

Innhold

Figurer

Tabeller

Nomenklatur

ACC	Akselerometer
ADC	Analog til digital omformer
DAC	Digital til analog omformer
DAQ	Data acquisition
DLL	Dynamic link-library
DOF	Frihetsgrader
EMG	Elektromyografi, måling av elektrisk aktivitet når en muskel trekker seg sammen
GUI	Graphical User Interface/Grafisk brukergrensesnitt
ITK	Institutt for Teknisk Kybernetikk
LDA	Lineær diskriminant analyse
NI	National Instruments
NTNU	Norges teknisk-naturvitenskapelige universitet
PGT	Proteseguidet trening
SDK	Software development kit
SEMG	Surface-elektromyografi
SGT	Skjermguidet trening
TCP/IP	Transmission Control Protocol/Internet Protocol

USB	Universal Serial Bus
VI	LabVIEW-fil for kode
XML	Extensible Markup Language

1 Innledning

Innen protesestyring er mønstergjenkjenning mye brukt for å kunne tolke hva protesebrukeren vil at protesen skal gjøre. Mønstergjenkjenning er robust og ytelsen er god, men protesen kan ikke kontrolleres i den grad som er ønskelig. For å kunne kontrollere protesen bedre må nye styringsmetoder innføres. Proporsjonalstyring er et slikt system og er implementert inn i et programvarebibliotek utviklet på Institutt for Teknisk Kybernetikk (ITK). Dette biblioteket er utviklet gjennom en tidligere masteroppgave av Jørn Bersvendsen [1] og undertegnedes prosjektoppgave [2]. Biblioteket er skrevet i programmeringsspråket LabVIEW. Hvis dette er et ukjent språk er det anbefalt å lese Appendiks C i [1].

Denne oppgaven bygger på mange av metodene i biblioteket. Dermed er det anbefalt å lese [1] for å få et overblikk av biblioteket. Informasjon om treningsmetoder er delvis hentet fra [2].

Videreutviklingen av ITKs bibliotek omhandler oppsett av innhenting av sensordata fra en bruker, implementasjon av et nytt styringssystem og muligheten for å styre en protese fra Motion Control(MC). En sentral del av oppgaven er at det skal være mulig å benytte proporsjonalstyring for å kontrollere protesen fra MC. Dette ble implementert inn i ITKs bibliotek, som allerede består av et mønstergjenkjenningssystem for å kontrollere en robothånd. For å benytte proporsjonalstyring er det foreslått ulike metoder som kan brukes for å trene opp de ulike funksjonene en motor kan ha. Basis for alle metodene er proteseguidet trening.

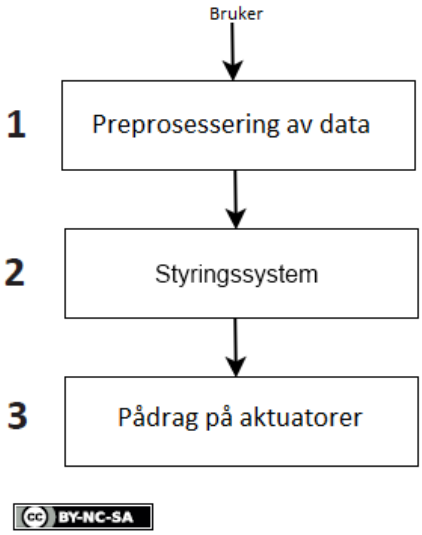
Rapporten er delt inn i 8 kapitler. Kapittel 2 beskriver overordnet struktur for styring av proteser med eksempler for styringssystemer og treningsmetoder for trening av disse styringssystemene. I kapittel 3 beskrives protesen fra MC og ulike metoder for å kunne bruke denne sammen med ITKs bibliotek. Seksjon 4.1 beskriver forskjeller og likheter mellom styringssystemene proporsjonalstyring og mønstergjenkjenning. Seksjon 4.2 beskriver trening av proporsjonalstyring

med proteseguidet trening som basis. Seksjon 4.3 beskriver hvilke funksjoner det er ønsket at ITKs bibliotek skal utvides med. Implementasjon av deler av funksjonene som er ønsket er presentert i kapittel 5. Kapitlene 6 og 7 er henholdsvis resultat og diskusjon. Kapittel 8 inneholder konklusjon og forslag til fremtidig arbeid.

2 Introduksjon

Dette kapitlet beskriver tre sentrale elementer for styring av proteser; innhenting av data, styringssystemer og pådrag på aktuatorer. I tillegg introduseres metoder for systemtrening og ITKs programvarebibliotek.

Losier forslo i [3] en modell for systemoppbygning av ei protese. Denne modellen beskriver hvordan et protesestyringssystem kan være bygd opp, fra innhenting av data fra bruker til styring av protesen. Figur 2.1 viser en forenklet versjon av modellen.



Figur 2.1: Modifisert versjon av Losiers modell, basert på [3, 4]

1. **Preprosessering av data** skjer gjennom tre steg:
   (a) Innhenting av data fra bruker, for eksempel signaler fra elektromyografielektroder
   (b) Filtrering av data fanget opp fra brukeren, for eksempel filtrering av 50 Hz fra strømnettet
   (c) Beregne egenskaper på målt data fra hver sensor, eksempler på egenskaper er gjennomsnittsverdi og varians

2. **Styringssystem** brukes for å gjøre om data fra preprosesseringen til motorfunksjoner med tilhørende verdier. Et eksempel på et styringssystem er mønstergjenkjenning.
3. **Pådrag på aktuatorer** setter pådragene gitt av styringssystemet på aktuatorene til en utførende enhet.

2.1 Innhenting av data fra bruker

Data som hentes inn fra bruker gjennomgår noen transformasjoner for å kunne brukes videre i neste steg. Det samles inn N antall sampler fra brukeren. Disse brukes i egenskapsberegning for hver enkelt sensor.

Elektromyografi

Muskelsammentrekning skyldes at det går en elektrisk puls langs muskelfibrene i begge retninger. Dette kalles for et myoelektrisk signal(MES) og kan måles med elektromyografi(EMG). De EMG-elektrodene som blir brukt i denne oppgaven registrer MES på hudoverflaten, kjent som surface-EMG(SEMG). EMG-elektrodene er bygd opp av en eller flere elektroder som avleser spenningsforskjeller. Fordelen ved bruk av SEMG er at protesebrukeren slipper å få operert inn EMG-elektrodene. En ulempe er at ved måling på hudoverflaten registreres det mange MES samtidig. Dette medfører at finmotorikken blir svekket i forhold til elektroder som er operert inn på hver muskel [5]. Men for de generelle funksjonene, som for eksempel håndleddsfleksjon, vil SEMG være tilstrekkelig.

EMG-elektroder er et eksempel på en sensortype som brukes for innhenting av data i ITKs bibliotek. Disse elektrodene blir beskrevet i seksjon 2.4

Beregne egenskaper fra data

Data hentet fra sensorene gjennomgår egenskapsberegninger for å kunne brukes i styringssystemer. Eksempler på egenskaper er varians og gjennomsnitt. Se [1]

for en nærmere beskrivelse av egenskapene støttet av ITKs bibliotek.

2.2 Styringssystemer

Styringssystemene skal sikre en bruker kontroll over en protese på en enkel og effektiv måte. For å bruke disse systemene kreves det treningsmetoder som instruerer brukeren i hva som skal gjøres. Denne seksjonen beskriver ulike treningsmetoder og styringssystemer.

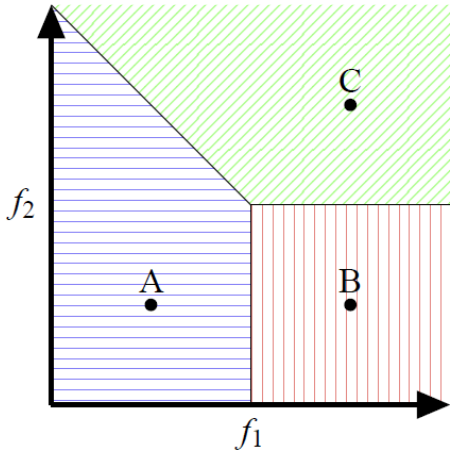
2.2.1 Mønstergjenkjenningssystem

Mønstergjenkjenning bruker kjente data i forhold til en klasse for å klassifisere data som er uidentifisert [6]. Innen protesestyring fastsettes klassene til gitte bevegelser, for eksempel håndledds ekstensjon/fleksjon og underarms supinasjon/pronasjon. Protesebrukeren blir gjennom et treningsprogram bedt om å gjøre en gitt bevegelse for å kunne skape en relasjon mellom en klasse og data som blir målt. Denne relasjonen er en klassifikator som skal gjøre om data til en klasse. Standard treningsmetode for denne klassifikatoren innen protesestyring er lineær diskriminant analyse(LDA), se figur 2.2. LDA deler ulike klasser inn i seksjoner skilt av lineære funksjoner. Ved kjøring av systemet medfører det at et punkt plasseres i en av disse seksjonene. Punktet representerer egenskapsberegningen av sensordata. Innleste data har med andre ord blitt klassifisert til en gitt klasse. Resultatet overføres til protesen ved å sette en konstant hastighet på de elektriske motorene klassen samsvarer med. Dette blir kalt *on-off* regulering [4] og medfører at en motor holder konstant hastighet i en gitt retning eller er i full stopp.

2.2.2 Proporsjonalstyring

Denne seksjonen baserer seg på [4].

Proporsjonalstyring tillater at en eller flere motorfunksjoner kan reguleres proporsjonalt. Dette betyr at vilkårlige verdier vil kunne settes som referanse



Figur 2.2: Egenskapsområdet f1 og f2 delt inn i tre ulike regioner med hver sin klasseid; A, B og C [1]

for en eller flere av motorfunksjonene. I et simultant proporsjonalsystem kan motorfunksjonene til alle frihetsgrader(DOF) styres samtidig.

Treningen av et slik system skaper en relasjon mellom data fra brukeren og referanseverdier. For å skaffe nok referanseverdier, brukes ferdig definerte treningssett til å trene opp systemet. Relasjonen mellom brukerdata og referanseverdiene kan estimeres med estimeringsmetodene beskrevet under gitt lineær mapping.

Lineær mapping

Med system gitt på formen 2.1 hvor F og x er kjent, må A kunne estimeres for å si noe om relasjonen mellom F og x.

$$F = Ax \tag{2.1}$$

hvor:

1. **F** er en vektor med referanseverdier som skal påtrykkes på aktuatorene
2. **x** er vektor med data som skal settes i relasjon til F og som i denne

sammenhengen vil være data tatt opp fra bruker

3. **A** er en matrise som beskriver relasjonen mellom x og F

Lineær regresjon

For å kunne estimere A kan lineær regresjon brukes. Regresjon går ut på å lage en lineær funksjon med minst mulig feil i forhold til et sett med punkter. Gitt mange punkter for F og x kan A estimeres ved å bruke formel 2.2 i for eksempel Matlab.

$$A = x' \backslash F' \tag{2.2}$$

Neurale nettverk

Et neuralt nettverk er en ulineær estimeringsmetode som kan brukes for å estimere A i formel 2.1. Men i [4] konkluderes det med at den enkle metoden lineær regresjon gir nesten like gode resultater som neurale nettverk. Derfor ble ikke denne metoden vurdert videre i denne oppgaven.

Normal lineær estimering

Normal lineær estimering bruker lineær regresjon direkte der referanseverdier for aktuatorene er F og inngangsdata fra bruker er x. Ved kjøring av systemet vil formel 2.1 brukes for å beregne hvilke verdier som skal settes på aktuatorene.

Dekoblet lineær estimering

Dette er en metode som er utviklet av Øyvind Stavdahl og Anders Fougner, men som ikke har blitt publisert enda. Denne estimeringsmetoden dekomponerer

referanseverdiene inn i ulike egenskaper. Disse egenskapene er gjennomsnitt, standardavvik og form. Hver enkelt av disse kan beregnes som vist i formelene 2.3–2.5

$$F_{av} = mean(F) \tag{2.3}$$

$$F_{std} = std(F) \tag{2.4}$$

$$F_{form} = \frac{F - F_{av}}{F_{std}} \tag{2.5}$$

Ved bruk av lineær regresjon som estimeringsmetode blir hver komponent estimert ved bruk av formelene 2.6–2.8.

$$A_{av} = x' \backslash F'_{av} \tag{2.6}$$

$$A_{std} = x' \backslash F'_{std} \tag{2.7}$$

$$A_{form} = x' \backslash F'_{form} \tag{2.8}$$

Kjøring av systemet ved bruk av dekoblet estimering gir F regnet ut i forhold til formel 2.9, der .* er elementvis multiplikasjon

$$F = (A_{form}x).*(A_{std}x) + A_{av}x \tag{2.9}$$

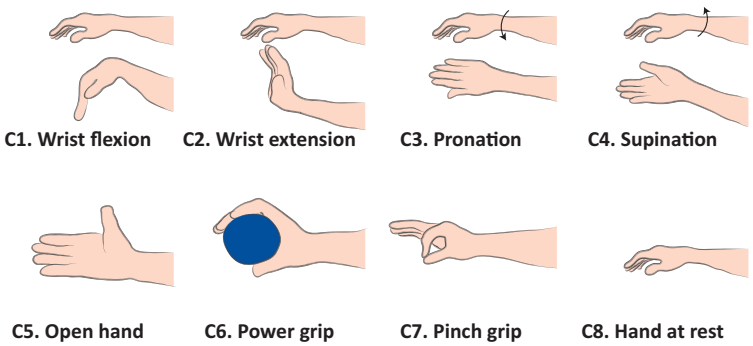
2.3 Metoder for systemtrening

For å trene opp et styringssystem må brukeren instrueres i hva som skal gjøres.

2.3.1 Skjermguidet trening(SGT)

Prinsippet bak SGT er at protesebrukeren får en visuell input fra en skjerm og skal replikere bevegelsen vist på skjermen. Denne inputen kan være i form

av video eller bilder. SGT er den treningsmetoden som er mest brukt for klassifisering hvor MES skal fastsettes til en spesifikk klasse. Populariteten til SGT gjør at denne metoden ofte blir sett på som den eneste måten for å visualisere de bevegelsene protesebrukeren skal gjøre [7]. Et av problemene med SGT er at den krever at brukeren har tilgang til en skjerm store deler av dagen. Men dette begynner å bli et mindre problem siden smarttelefoner er lett tilgjengelig. Likevel er det ønskelig å slippe bruk av eksterne verktøy for å trene opp protesen. Hovedproblemet per i dag er at protesen trenes opp inne. Dette kan gi negative følger siden inneklima og uteklima kan påvirke protesen på forskjellige måter. For eksempel vil det om vinteren være kaldt ute, men varmt inne. Dette vil skape problemer med klassifiseringen fordi måleinstrumentene oppfører seg annerledes i ulike temperaturer og fuktighetsnivåer. Bildene i figur 2.3 viser illustrasjoner av bevegelser som kan bli brukt som visuell input i SGT.



Figur 2.3: Illustasjoner av bevegelser brukt i SGT [8]

2.3.2 Proteseguidet trening(PGT)

PGT er et nytt konsept som ble introdusert i [9]. Dette er et forholdsvis enkelt konsept hvor protesen blir brukt for å trene opp styringssystemet i stedet for et eksternt verktøy, som for eksempel en skjerm. Ved trening vil da protesen gjøre bevegelser basert på forhåndsdefinerte mønstre.

PGT går ut på å bruke protesen som visuell input i læringsprosessen i stedet for bilder eller video. Protesen brukes for å visualisere bevegelser brukeren skal

gjenta med den amputerte armen. Fordelene med bruk av PGT i forhold til SGT er mange. Dersom ytelsen til systemet blir dårlig, det vil si at bevegelser ikke blir klassifisert riktig, kan protesen enkelt trenes opp igjen hvor som helst. Brukeren trenger da ingen eksterne enheter for å trene opp igjen systemet [7]. Siden protesen kan trenes opp hvor som helst, vil PGT muliggjøre et større bruksområde enn SGT. I [9] konkluderes det med at PGT gir brukeren en mer aktiv rolle for å rekalibrere protesen når ytelsen blir dårlig. Det vises også til at brukerne er kapable og villige til å trene opp igjen protesen ved bruk av PGT i stedet for å ta den av.

2.3.3 Mirroring

Treningsmetoden *Mirroring* går ut på at en protesebruker prøver å etterligne den motsatte hånds bevegelse [4]. Denne fungerer bare for personer som har en frisk hånd og en protese. Denne treningsmetoden er hovedsakelig brukt i kraftstyring av proporsjonalsystemer.

2.4 ITKs programvarebibliotek

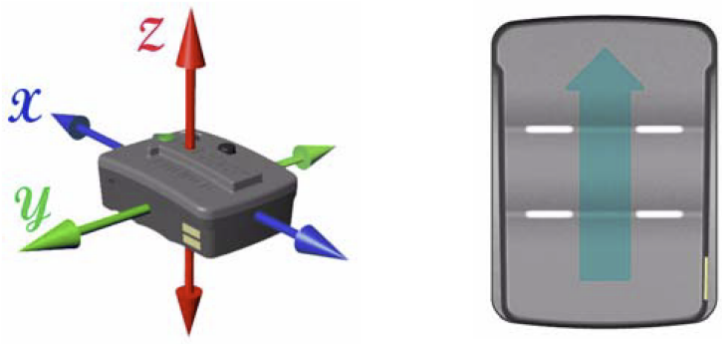
ITKs programvarebibliotek er laget av Jørn Bersvendsen i hans masteroppgave [1] og videreutviklet av undertegnede i prosjektoppgaven [2]. Biblioteket er skrevet i LabVIEW og Matlab og har støtte for innhenting av EMG- og akselerometerdata(ACC).

LabVIEW er et grafisk programmeringsspråk utviklet av National Instruments(NI) og beskrives av filer med ekstensjonen *.vi*. Et LabVIEW-program er bygd opp av en eller flere VIer. En VI kan kalle på andre VIer som kalles subVIer og vises som et ikon. Den viktigste fordelen med LabVIEW er at det er enkelt å lage et grafisk brukergrensesnitt. For en kort introduksjon i LabVIEW se [1, Appendiks C]. Kort oppsummert kalles den grafiske delen og kodedelen av en VI for henholdsvis frontpanelet og blokkdiagrammet.

For å hente inn sensordata brukes en Data acquisition-enhet (DAQ)[1]. Disse enhetene kan både ha innganger og utganger. Inngangene kan brukes til å hente inn analoge signaler fra sensorene og gjøre de digitale, analog til digital-omformer(ADC). Utgangene kan brukes til omgjøring digitalt til analogt, digital til analog-omformer(DAC). NI lager slike DAQer og gjør at oppkobling til en DAQ er godt integrert inn i LabVIEW. Biblioteket har støtte for innhenting av EMG- og ACC-data fra Delsys sine Trigno-sensorer, vist i figur 2.4 [10, 11].

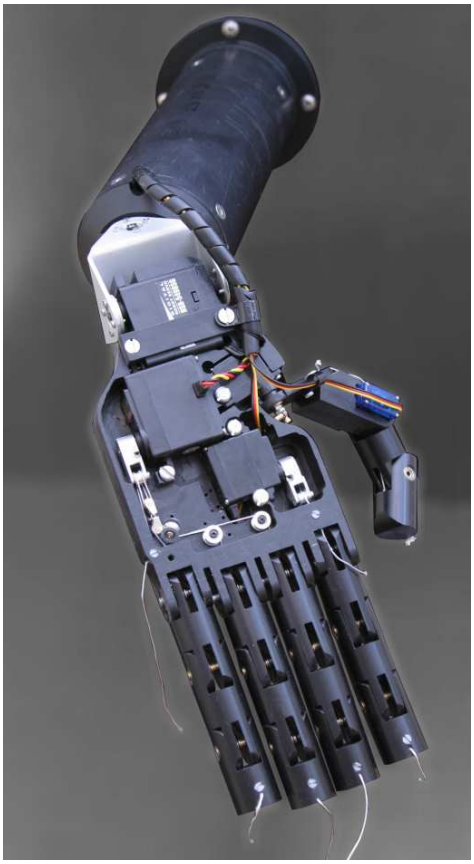
Delsys sitt system består av en basestasjon og 16 sensorer. En enkelt sensor inneholder en EMG-elektrode og et tre-akset akselerometer. Sensordata blir trådløst overført til basestasjonen som igjen sender signalet videre til en DAQ med 16 innganger. Dermed støtter den innhenting fra 4 Trigno-sensorer(4 EMG + 12 ACC).

Biblioteket har støtte for bruk av styringssystemet mønstergjenkjenning. Dette brukes for å kontrollere en robothånd utviklet av Kristian Håkonsen i hans masteroppgave [12], vist i figur 2.5.



Figur 2.4: En Trigno sensor med elektroder og akselerometer [10]

[1] http://www.ni.com/data-acquisition/what-is/



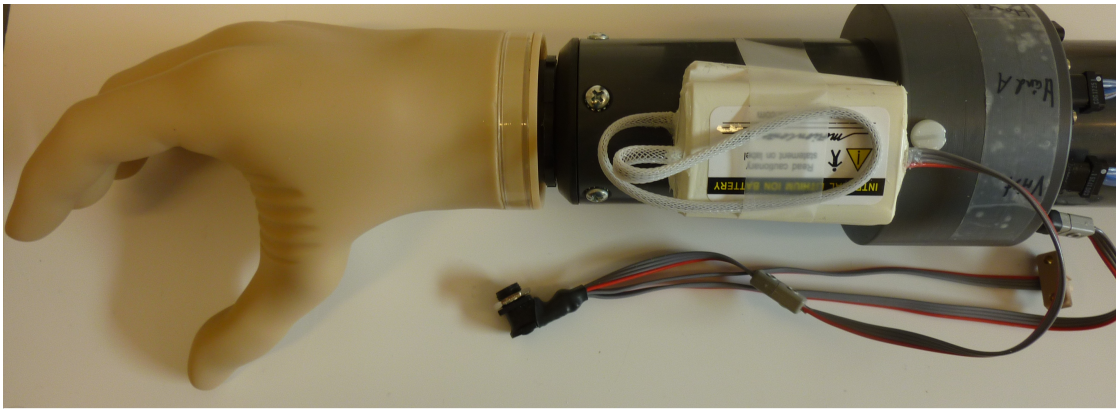
Figur 2.5: Robothånd [12]

3 Motion Control protese

Motion Control(MC) er et firma som utvikler proteser. En slik protese har ITK nå kjøpt inn for bruk innen forskning. Denne protesen er en del av grunnlaget for denne masteroppgaven og brukes for å teste proporsjonalstyring. I følgende seksjoner vil protesen beskrives med dokumentasjon fra leverandøren [13, 14, 15, 16], personlig kontakt[17] og eksperimentering som grunnlag.

3.1 Oppbygning

Protesen har to DOFs, ProWrist og ProHand, og blir styrt av Triad-elektroder. Enheten blir levert med mange komponenter. Disse komponentene er blitt montert sammen i et stativ laget av mekanisk verksted på ITK, se figur 3.1.



Figur 3.1: Motion Control protese i stativ fra ITK

Protesen er bygd opp av følgende komponenter:

Triad-elektroder for avlesning av EMG er vist i figur 3.2a. En elektrode har en tre-ledet kabel hvor den midterste ledningen er et koblingspunkt som er likt for alle elektroder. Den markerte ledningen på figur 3.2a er koblet til batteriet. Den siste brukes for å sette en spenning i forhold til muskelaktivitet med felles koblingspunkt som referanse.

ProWrist er en fritt roterende motor som tillater supinasjon og pronasjon, enheten er vist i figur 3.2b. Den har tre ledninger; Wrist, Hand A og Hand B. Batteriet må kobles til ledningen Wrist for at protesen skal få strøm. Det kan også kobles til et trådløst adapter for kommunikasjon mellom ProWrist og en PC samt et sett med to Triad-elektroder. Til Hand A og Hand B kobles det et sett med Triad-elektroder på hver av ledningene. Hand A og B blir videreformidlet til ProHand. Se [13] for Motion Control sin guide for oppkobling av ProWrist.

ProHand er selve hånden og består av en motor som sørger for åpning og lukking av protesehånden. Enheten er vist i figur 3.2c.

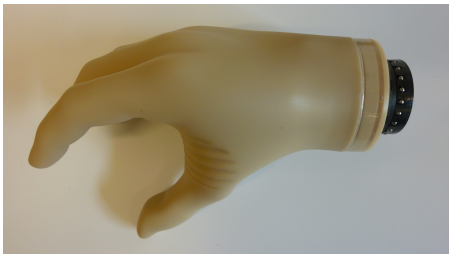
Bluetooth-adapter som sørger for trådløs kommunikasjon mellom en PC og Pro-enhetene. Denne kommunikasjonen brukes for å sette parametere på hver av Pro-enhetene, for eksempel *gain* og *threshold*.

T-kabel brukes for å få tilgang til å sette parametere på ProHand via Bluetooth. I [13] vises hvordan T-kabelen brukes.

Batteri og lader følger også med.

(a) Triad-elektrode

(b) ProWrist

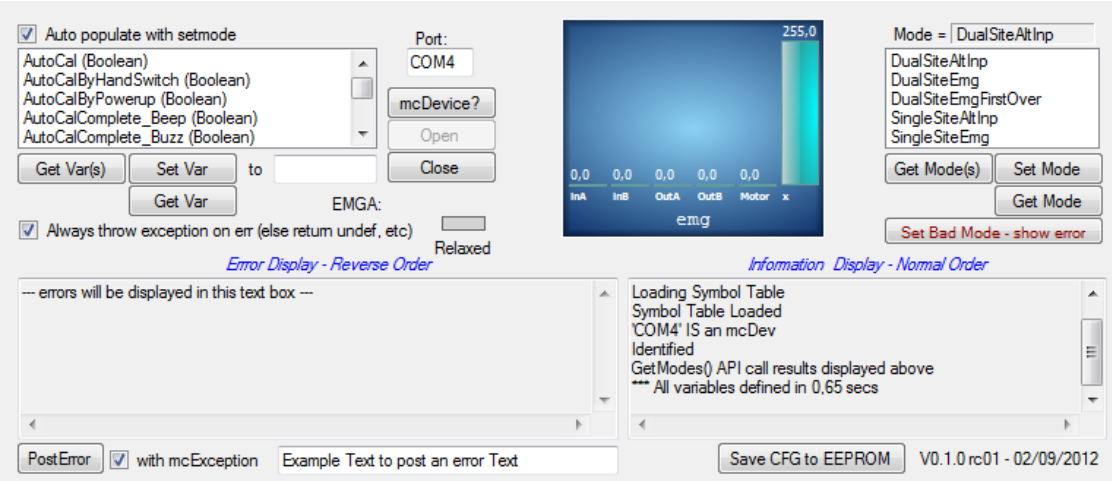


(c) ProHand

Figur 3.2: Deler til Motion Control sin protese

3.2 Tilgjengelig programvare

For å håndtere kommunikasjon med protesen trådløst er det utviklet et Dynamic-link library (DLL) som implementerer nødvendig funksjonalitet. Figur 3.3 viser grafisk grensesnitt for et program skrevet i C#. Programmet bruker den nevnte DLLen for å teste funksjonalitetene til protesen. Dokumentasjonen for DLLen, se [16], beskriver hvilken klasse som skal brukes for å oppnå kommunikasjon med protesen.



Figur 3.3: Motion Control programvare

Følgene felter i C#-programmet er verdt å legge merke til:

Port beskriver hvilken port hver av Bluetooth-adapterene er tilkoblet. Denne porten vil være unik for hver Pro-enhet.

McDevice sjekker om enheten som er koblet til porten er en MC protese.

Open sørger for å åpne porten slik at programmet mottar data fra protesen, mens **Close** stenger porten.

Mode bestemmer hvor mange av de Triad-elektrodene koblet til enheten som skal brukes og hvordan signal fra elektrodene blir filtrert. *Singelsite* mode betyr at bare en Triad-elektrodene er til disposisjon, mens *Dualsite* mode betyr at to elektroder er tilgjengelige. *AltInp* betyr at det ikke er noen filtrering av signalene

fra Triad-elektrodene. EMG betyr at signalene fra Triad-elektrodene blir filtrert før de settes på motoren. Her brukes *DualsiteAltInp*, altså to Triad-elektroder per Pro-enhet og ingen filtrering.

Funksjoner kan sette parametere på protesen. Eksempler på parametere er *gain* og *threshold*. Ved å justere verdiene for disse endres responsen til motoren i forhold til et signal fra Triad-elektrodene. Motorpådrag og input fra Triad-elektroder blir vist som verdier mellom 0 og 255 i brukergrensesnittet(GUIet). For å lagre parameterne som default på protesen kan knappen **Save CFG to EEPROM** aktiveres. Denne knappen gjør at parameterne lagres i non-volatile minne i Pro-enheten.

3.3 Styringsmetode

Pro-enhetene styres av Triad-elektrodene. Spenningen lest av fra elektrodene oversettes til en strøm som settes på motorene[17]. Dette medfører at motorene er hastighetsstyrt uten påført belastning, men kraftstyrt ved belastning. Ved bruk av *Dualsite* mode kobles hver elektrode til en motorretning. Dermed vil en spenning fra den ene elektroden medføre at motoren dreier med klokka, spenning fra den andre elektroden medfører at motoren dreier mot klokka.

3.4 Motion Control i ITKs bibliotek

For å kunne utnytte MC protesen som styringsenhet i ITKs bibliotek har følgende metoder hvert vurdert.

Bruk av Dynamic-link library

Det er gunstig å kunne bruke MC-DLLen i ITKs bibliotek for å bestemme hva protesen skal gjøre. Dette kommer av at det finnes en trådløs tilkobling som gjør det enklere å koble protesen til en PC. Figur 3.4 viser hvordan kommunikasjon

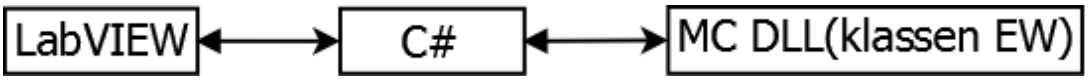
mellom DLLen og LabVIEW kan gjøres. Denne kommunikasjonen foregår i to steg.

Det første steget er sending og mottak av data mellom C# og DLLen. C#-programmet som har blitt skrevet, bygger på programmet fra MC men er laget uten GUI. Grunnen til at DLLen ikke kan benyttes direkte inn i LabVIEW er at DLLen benytter seg av en *delegate*-metode for å registrere mottatte data fra protesen. En *delegate*-metode tillater asynkront kall av en overordnet funksjon lenger ned i programmet. Det er ikke funnet en løsning for å bruke en VI som denne *delegate*-funksjonen.

Det siste steget er sending og mottak av data mellom LabVIEW og C#. En metode for å opprette kommunikasjon er bruk av TCP/IP. En annen metode bygger på funksjoner som leser og skriver til stacker. Det er laget et C# program som mottar data fra DLLen og setter dette på en stack. I LabVIEW brukes en VI for å lese fra denne stacken. Testing av denne metoden viser at DLLen til MC har delt minne. Dette betyr at initialisering av to objekter av samme klasse i DLLen medfører at objektene overskriver hverandre. Dermed må det enten to prosesser til for å snakke med begge Pro-enhetene samtidig eller så må DLLen lastes inn i minne flere ganger. Kode for C#-programmet og VIen er lagt ved masteroppgaven.

Videre informasjon og eksperimentering viste at motorpådrag ikke kan settes via DLLen[17]. Grunnen til dette er at protesen ikke er laget for at motorpådraget skal kunne settes utenfra. Protesen er konstruert som et lukket system og skal dermed fungere uten ekstra tilkoblinger.

Bruk av DLLen i biblioteket er utelukket siden motorpådrag ikke kan settes.



Figur 3.4: Kommunikasjon mellom LabVIEW og MC-DLLen

Endre firmware

For å kunne bruke de trådløse egenskapene til protesen og samtidig kunne sette motorpådrag må det lages ny firmware til hver av Pro-enhetene. Dette er den eneste endringen som trengs da alle andre nødvendig funksjoner allerede er støttet. Ved å endre firmware må det også lages en ny DLL for å kunne hente ut og sende data til Pro-enhetene. Siden det trådløse aspektet er viktig er dette et godt alternativ for å koble protesen sammen med ITKs bibliotek.

Bruk av en DAQ

En annen måte er å erstatte en av de enhetene som er koblet til protesen med noe som kan kontrolleres. Siden protesen opprinnelig blir kontrollert av Triad-elektroder vil det være naturlig å bytte ut disse med andre enheter. Ved å koble til ledningene fra Triad-elektrodene til utgangskanaler på en DAQ istedet, vil det være mulig å kontrollere strømmen som blir satt på motoren gjennom en VI i LabVIEW. Dette gjøres ved å sette ulike spenninger på utgangskanalene.

4 Systemtrening og funksjoner

4.1 Forskjeller og likheter i systemtrening

Denne rapporten beskriver to aktuelle styringssystemer; mønstergjenkjenning og proporsjonalstyring. Felles for de begge er at data fra bruker skal omgjøres til en verdi som beskriver en referanseverdi for motorfunksjoner. I mønstergjenkjenning er denne verdien en klasseid, mens i proporsjonalstyring beskriver verdien referanseverdien direkte. Bruk av proporsjonalstyring har med andre ord større oppløsning enn mønstergjenkjenning fordi pådrag som settes på motorene kan gis vilkårlige verdier, mens en klasse er fastsatt til en verdi.

Treningsmetodene til de to styringssystemene er litt forskjellige. Der mønstergjenkjenning kan beskrives av et bilde for hver klasse i SGT, må det i proporsjonalsystemer brukes animasjoner for å instruere brukeren i hva som skal gjøres. PGT fungerer på samme måte for de to systemene fordi protesebrukeren skal gjøre det samme som protesen.

Trening ved mønstergjenkjenning består i å fortelle brukeren om å gjøre gitte bevegelser. Ved bruk av mønstergjenkjenning som styringssystem er det begrenset hvor mange klasser som brukes. Dermed skal data fra sensorene transformeres til et lite sett med verdier. For proporsjonalstyring kan en bevegelse beskrives av mange verdier. Derfor må det mange referanseverdier til for å kunne skape god nok relasjon mellom data fra bruker og motorreferanser. Det er her treningssett kommer inn for å kunne definere hva referanseverdiene skal være.

Videre i rapporten er det satt fokus på proporsjonalstyring.

4.2 Proteseguidet trening i proporsjonalsystemer

Ved bruk av PGT i proporsjonalsystemer er det interessant å kunne trene opp systemet med PGT uten modifikasjoner. I proporsjonalsystemer finnes tre ulike motorfunksjoner som er ønskelig å kunne trene opp. Disse funksjonene er posisjon, hastighet og kraft. Men hver motorfunksjon gir ulike begrensninger ved bruk av PGT. Det er for eksempel vanskelig å se på en protese hvor hardt den klemmer.

Posisjon

Posisjonen til et objekt er relativt enkel å oppfatte. PGT innebærer at protesebrukeren skal følge posisjonen til protesen under trening. Dette medfører at PGT vil fungere alene uten hjelp av andre metoder. Posisjonsstyring vil være den enkleste metoden å bruke.

Hastighet

En motor kan være fritt roterende eller ha endepunkter. Fritt roterende motorer gjør det mulig å se hastighetsreferansen som er påtrykt motoren. En motor med endepunkter vil være annerledes fordi det må sikres at motoren ikke står i et av endepunktene. Hvis en hastighet er satt på en motor som står i sitt endepunkt vil ikke brukeren kunne oppfatte hva referanseverdien er. Dette gjør at det enten må være mulig å detektere om motoren er i sitt ytterpunkt eller at treningssettet må bli designet slik at det ikke er mulig å gå utover endepunktene. Ved å ha mulighet til å lese av posisjon eller hastighet på en motor vil disse problemene kunne unngås. Hvis en motor står i et av endepunktene, men blir påtrykt en hastighetsreferanse ulik 0, vil det være mulig å lese av hastigheten på motoren. Den avleste hastigheten brukes som referanse istedet for den referanseverdien som blir påtrykt motoren.

Dette betyr at for en fritt roterende motor fungerer PGT helt fint alene, men for en motor med endepunkt må PGT kombineres med en avlesning av en eller flere

motorfunksjoner.

Kraft

Et av problemene med PGT for å trene bruk av kraft er at det ikke er mulig å se hvor hardt en protese klemmer. Dersom dette er av interesse kan ikke PGT brukes alene. For å kunne fortelle brukeren hvor mye kraft som brukes, må det skapes en relasjon som gjør at det er mulig å føle hvor stor kraften er.

PGT og SGT

Ved å kombinere PGT og SGT vil det være mulig å vise hvilken bevegelse protesen gjør og illustrere på skjerm hvor stor kraft som brukes. Dette er ingen god metode siden to objekter følges med på samtidig. Kompleksiteten vil bli unødvendig stor og SGT kunne like gjerne bli brukt alene for å illustrere bevegelse og kraft.

PGT og Lyd

For å unngå at brukeren må følge med på to ting samtidig, kan forskjellige sanser brukes. Ved å kunne høre hvor mye en motor jobber, vil blikket være fokusert på protesen og ørene på motorlyden. Men dette vil også være komplisert fordi det kan være vanskelig å høre forskjeller i motorlyden.

PGT sammen med andre objekter

Mirroring ble nevnt tidligere som en metode for å trene opp et styringssystem basert på kraftestimering. Denne vil kunne brukes i dette tilfellet også. For å benytte denne metoden med PGT vil brukeren være avhengig av redskaper som bidrar til at den friske hånden skal kunne oppfatte det som protesen gjør. Dette medfører at protesebrukeren må ha med seg unødvendig mye utstyr for å kunne trene opp styringssystemet. I tillegg må brukeren tenke på at det som gjøres med

den friske hånden også skal gjøres med den amputerte hånden for å kunne ta opp riktig data.

Et hensiktsmessig objekt for å estimere kraften til protesen er en klemmeball. De fleste har en relasjon til hvor hardt noe må klemmes for at det skal deformere seg. En klemmeball vil ikke kunne benyttes for for eksempel pronasjon og supinasjon. Men ved å bruke en klemmeball vil kombinasjonen av denne og PGT kunne brukes til å trene kraftstyring av gripefunksjon.

En sans som kan benyttes for å kjenne kraft er berøring. Poenget med det er at brukeren kan bruke berøring for å estimere hvor hardt protesen roterer eller klemmer. Dette er gunstig siden kroppen er tilgjengelig overalt hvor protesen skal fungere. Men dette medfører at det må lages treningssett som ikke kan skade brukeren, for eksempel ved å klemme for hardt. Treningssettene må ta høyde for dette, men da kan ikke de høyeste klemmeverdiene bli estimert nøyaktig. Men dette er også en av få ulemper med denne metoden.

4.3 Funksjonsspesifikasjon

Tabell 4.1 viser ønskede utvidelser av ITKs bibliotek. Kravene er delt inn i de tre nivåene vist i den modifiserte Losier modellen:

1. Ønskede endringer og muligheter i forhold til innhenting av data fra protesebrukeren
2. Hvilke styringssystemer programmet skal kunne benytte
3. Hvilke enheter som skal kunne vise de gitte bevegelsene

Tabell 4.1: Tabell av ønskede funksjoner

Krav nr.	**Beskrivelse**	**Kommentar**	**Kryssref**
K1	Sensormoduler **skal** være utskiftbare(modulære) og kunne kombineres		

K1.1	Data fra Trignobasen **skal** overføres via USB		
K1.1.1	**Skal** fungere i sanntid med minst 4×EMG og 4×3×ACC (4 Trignosensorer)		
K1.1.2	**Bør** fungere med alle sensorene i sanntid		
K1.1.3	**Bør** fungere uten bruk av andre programmer for å lese fra USB		
K1.2	Data fra Motion Control sensorer **bør** være tilgjengelige i sanntid	Avhenger av mulige løsninger av K3.2	K3.2
K1.3	Det **bør** være enkelt å legge til nye typer sensorer	EMG og ACC brukes nå, men blant annet kraftmåling vil kunne legges til	K1
K2	Det **skal** være mulighet for bruk av forskjellige typer styringssystemer		
K2.1	Styringsmodulene **skal** være modulære for enkelt å kunne legge inn nye styringsmetoder		
K2.2	Det **skal** være mulighet for fortsatt bruk av mønstergjenkjenning		
K2.3	Proporsjonalstyring **skal** kunne nyttes som styringssystem		
K2.3.1	**Skal** ha mulighet til å styre kraft, hastighet og/eller posisjon	Alle tre vil være mulig å styre enkeltvis, og i kombinasjon med hver andre	
K2.3.2	**Skal** ha treningsmetode for bruk av hastighets- eller posisjonsstyring		

K2.3.3	**Bør** ha en treningsmulighet for styring av kraft		
K2.3.4	Det **skal** være mulig å kunne velge ulike treningssett	Dette er for å se hvilke treningssett som gir best resultat	
K2.3.5	Treningssettene **skal** være enkle å følge	Dette medfører at bevegelsene i treningsettet må være naturlige	
K3	Koblinger mellom ITKs bibliotek og ulike proteser/robothender **skal** være modulær		
K3.1	**Skal** fortsatt være mulighet for å bruke robothånden		
K3.2	**Skal** ha mulighet for å kobles opp mot Motion Controls protese		
K3.2.1	**Skal** fungere i sanntid		
K3.2.2	**Skal** ha funksjoner for å sette kraft, hastighet, og/eller posisjon		
K3.2.3	**Bør** kunne utnytte alle funksjonene protesen har		

5 Implementasjon

Dette kapittelet omhandler endringer gjort i ITKs bibliotek i forhold til funskjonsspesifikasjonen i tabell 4.1.

5.1 Bruk av LabVIEW

For å kunne si noe om hva slags metoder som er brukt i implementasjonen av funksjonsspesifikasjonen kreves det en gjennomgang av sentrale datatyper for LabVIEW.

Et **cluster** er en samling av forskjellige datatyper kapslet inn i en datatype. Et eksempel er vist i figur 5.1a hvor flere datatyper er kapslet inn i datatypen *Test cluster*.

En **Refnum** er en referansekontroll som brukes som innganger i subVIer for å tillate endringer av det objektet referansen peker til. Dette blir det samme som pekere i programmeringsspråket C. Figur 5.1b viser eksempel på en slik *refnum*, med *Test cluster* som datatype. En måte å lage en *refnum* på er å lage den rett fra referansen til et objekt. Et eksempel er å lage en referanse av *Test clusteret*. Denne metoden gjør at objektet *Test cluster* ikke er inkludert i *refnumen* som vist i figur 5.1b. Ved å endre på clusteret vil denne metoden gjøre at *refnumen* ikke er kompatibel med objektet lenger. Derfor må det lages nye *refnumer* for alle subVIer hvis det skjer en endring vi clusteret. En annen måte er å lage en *refnum* rett fra objektet. Denne metoden er illustrert i figur 5.1b. En slik *refnum* oppdateres automatisk hvis det blir gjort endringer i clusteret. Dette bidrar til å kunne lage innganger til subVIer dynamiske.

En **Dynamic event handler** håndterer kode når et event skjer, for eksempel når en verdi endrer seg.

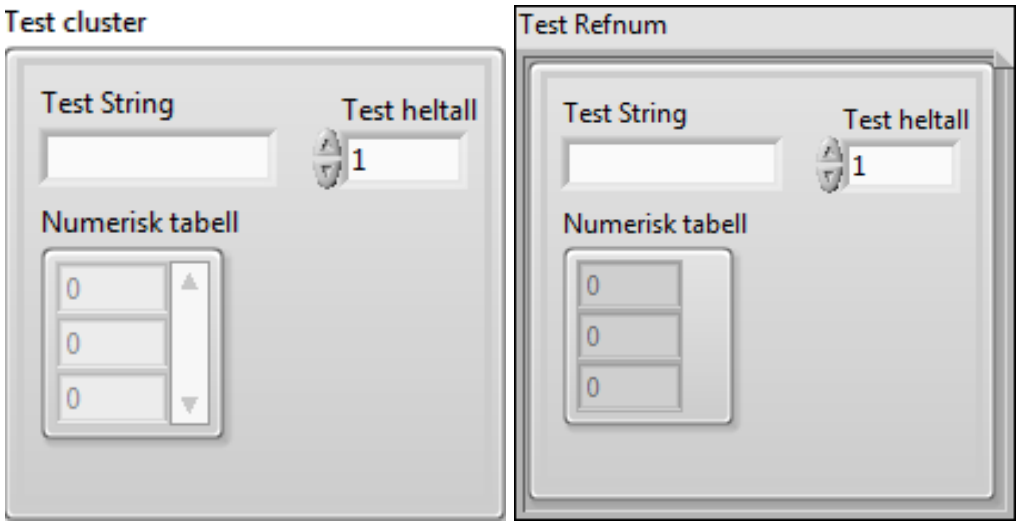
En **Ring** er en liste valgbare verdier og gir ut hvilken indeks som er valgt. Eksempelvis 0 hvis første element er valgt og 1 hvis andre element er valgt.

Et eksempel på dette er vist i figur 5.1c.

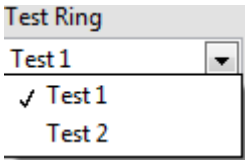
Et **MathScript** er en blokk som kan kjøre Matlab kode inne i LabVIEW.

En **Switch case** er en blokk som velger hva som skal kjøres basert på en bestemt verdi. Dette er det samme som en standard if-else operator.

Filekstensjon **.ctl** brukes for filer med egendefinerte objekter. Slike egendefinerte objekter kan lages direkte inn i en VI, men hvis objektet brukes flere steder i et program må eventuelle endringer gjøres på alle steder objektet er brukt. Ved å lage objektet som en ctl-fil trengs bare endringer i denne filen for å oppdatere alle instanser av objektet i programmet. Å lage en ctl kan gjøres ved å klikke på et objekt og si *Make Typedef*. Det er lurt å lage en typedef for alle egendefinerte clustere som brukes for at det skal være enklere å gjøre endringer. Alle clustere som har blitt programmert inn i eksisterende kode, er lagret som *typedefs*.



(a) Eksempel på cluster **(b)** Eksempel på *refnum*



(c) Eksempel på ring

Figur 5.1: Eksempler på datatyper for LabVIEW

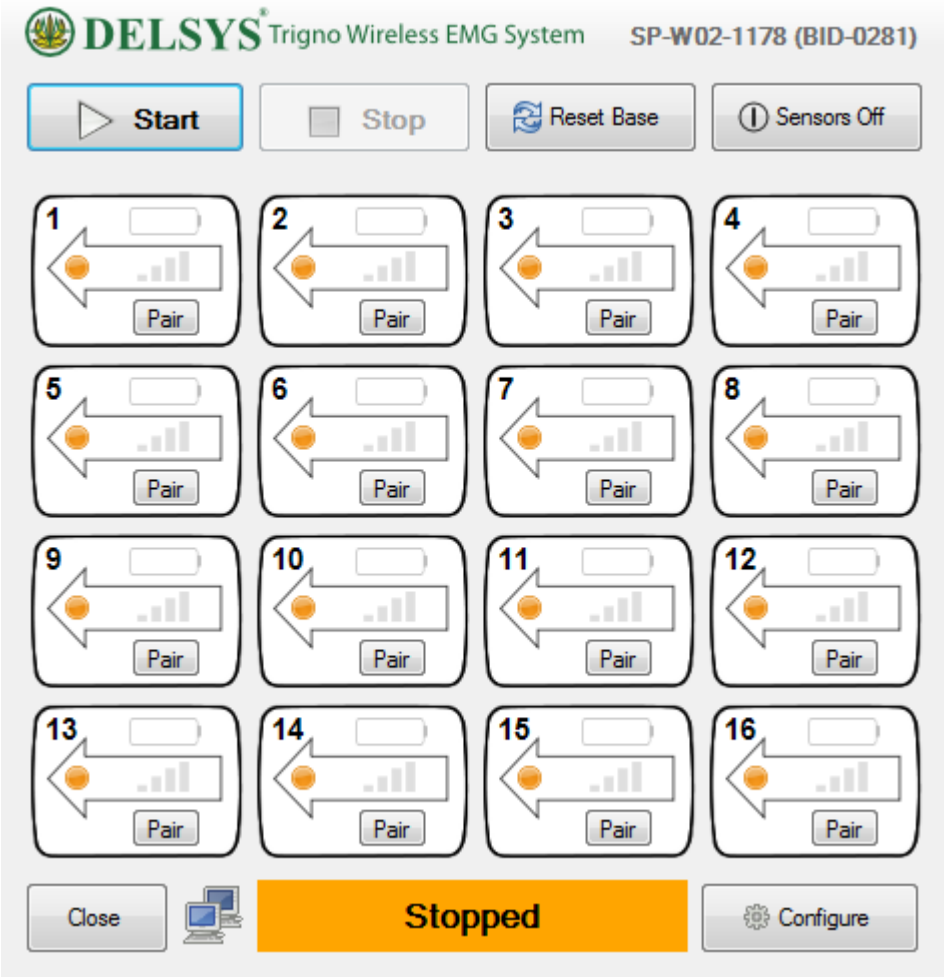
5.2 Endringer i innhenting av data fra bruker

Denne seksjonen tar for seg hvordan sensorendringer er implementert i forhold til funksjonsspesifikasjonens krav om sensorendringer.

5.2.1 Overføring av data over USB fra Trigno-basen

ITKs bibliotek har støtte for uthenting av data fra Trigno-sensorene ved bruk av en DAQ. Delsys har nå gjort det mulig å hente ut data i tredjeparts programvare over USB i sanntid. I denne sammenheng har de utviklet et Software Development Kit (SDK) [11]. Denne SDKen inneholder et program kalt *Trigno Analog Output*, som er vist i figur 5.2. Programmet mottar data over USB og videreformidler dette over TCP/IP. Et hvert tredjeparts program vil da kunne aksessere sanntidsdata over USB gjennom TCP/IP-koblingen. Delsys har vedlagt eksempelkode for noen programmeringsspråk, blant annet LabVIEW. For LabVIEW er det laget en hovedVI som viser data hentet fra VIene vist i figur 5.3.

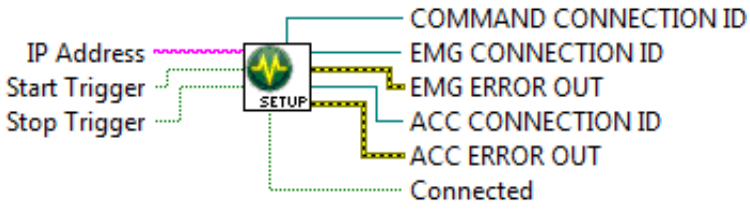
Setup-VIen i figur 5.3 initialiserer TCP/IP koblingen mellom *Trigno Analog Ouput* og tredjeparts programvare. Det som trengs er IP-adressen til den PCen Trigno-basen er koblet til. Normalt vil dette være den samme PCen som kjører tredjeparts programvare. IP-adressen kan da være 127.0.0.1. Trigno-basen har en *trigger*-port som kan brukes for å synkronisere med annen hardware. For å spesifisere om dette skal brukes, settes *Start Trigger* og *Stop Trigger* til henholdsvis *true* eller *false*. Dette betyr at hvis *Start Trigger* er satt til *true*, må Trigno-basen motta en puls på *trigger*-porten for å sende data. I ITKs bibliotek er det ikke noe hardware som må synkroniseres mot Trigno-basen. Dette betyr at begge *trigger*-verdiene vil være satt til *false*. VIen setter opp tre TCP/IP-porter for å sende og motta data. Disse er kommando- , ACC-, og EMG-porten. Kommandoporten brukes for å sende kommandoer til *Trigno Analog Ouput*. De viktigste kommandoene er *Start* og *Stop* som styrer om data mottatt over USB skal legges ut på TCP/IP-portene eller om de skal forkastes. ACC-porten brukes for å hente ut ACC-data og EMG-porten henter ut EMG-data.



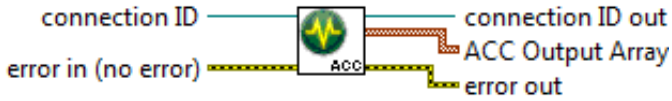
Figur 5.2: *Trigno Analog Ouput*

ACC-VIen i figur 5.3 mottar en sample av akselerometerdata med en samplingsrate på 148,1 Hz. En ACC-sample tilsvarer 192 bytes (16 sensorer $\times$ 3 akser per sensor $\times$ 4 bytes per akse).

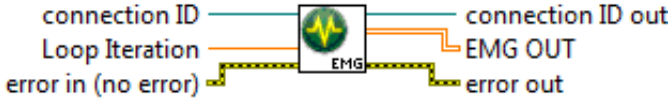
EMG-VIen i figur 5.3 mottar EMG-data. Samplingsraten for EMG er 2000 Hz slik at det må hentes ut flere EMG-sampler enn ACC-sampler. Forholdet mellom samplingsratene er $\frac{2000}{148,1} \approx 13,5$. Loop Iteration brukes for å hente ut 13 og 14 EMG-sampler annen hver gang, slik at ACC- og EMG-data kan hentes ut samtidig. Datatabellen inneholder 16 kolonner og 13 eller 14 rader. En EMG-sample tilsvarer 64 bytes (4 bytes per sensor $\times$ 16 sensorer)



(a) Åpne porter til *Tringo Analog Ouput*



(b) Hente ACC-data



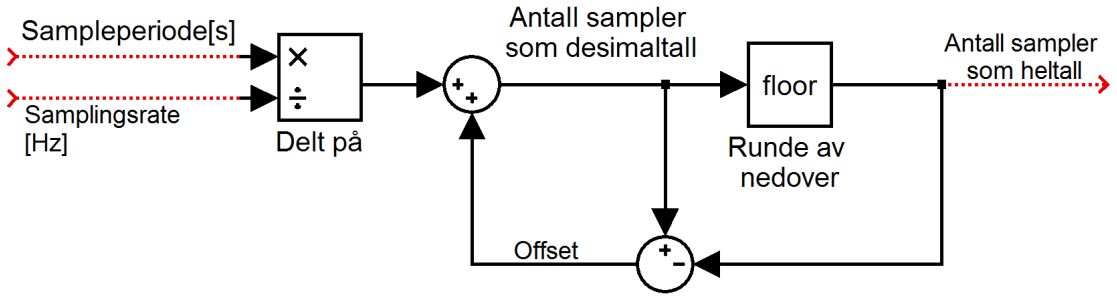
(c) Hente EMG-data

Figur 5.3: Delsys SDK-VIer

I ITKs bibliotek hentes det alltid ut en viss mengde sensorsampler. Dette fungerte fint tidligere da data ble hentet fra en DAQ. Grunnen til dette er at den har fast samplingsrate og fast antall sampler som skal hentes ut. Dette gjør at samplingsraten til ACC og EMG ikke har noe å si. For fast antall sensorsampler kan det ikke hentes ut data fra sensorer med ulik samplingsrate.

Istedet for antall sensorsampler kan det hentes ut en viss periode med data. Denne perioden er gitt i sekunder. For hver sensortype multipliseres perioden med samplingsraten og resulterer i antall sampler som skal hentes ut fra hver sensortype. Siden en viss mengde sampler hentes ut hver gang, vil en inngang på hver av SDK-VIene bestemme antall sampler som skal hentes ut. Dette er løst ved å hente ut antall bytes per sample $\times$ antall sampler fra TCP/IP-porten. Tabellen som returneres fra hver av VIene vil da være todimensjonal med like mange kolonner som antall sensorer og like mange rader som antall sampler. Et problem er at antall sampler som hentes ut er et heltall. Samplingsraten for ACC er 148,1 Hz. Dette betyr at hvis det er ønsket å hente ut 0,1 sekunder med data tilsvarer dette 14,81 sampler. Problemet ble løst ved å unngå desimaltall og dermed runde antall sampler ned til nærmeste heltall. Dette ble ble gjort for å unngå å måtte vente mer enn de 0,1 sekundene på sensordata. Samtidig vil dette

medføre forsinkelse i uthenting av data hvis det hentes ut 0,1 sekunder med data over en lang periode. Det er derfor ønskelig at gjennomsnittet av antall sampler skal bli 14,81 over denne perioden. En måte å få antall sampler på er å bruke en *offset*-verdi som beregnes av tidligere feil i antall sampler som er hentet ut. Metoden for dette er vist i figur 5.4.

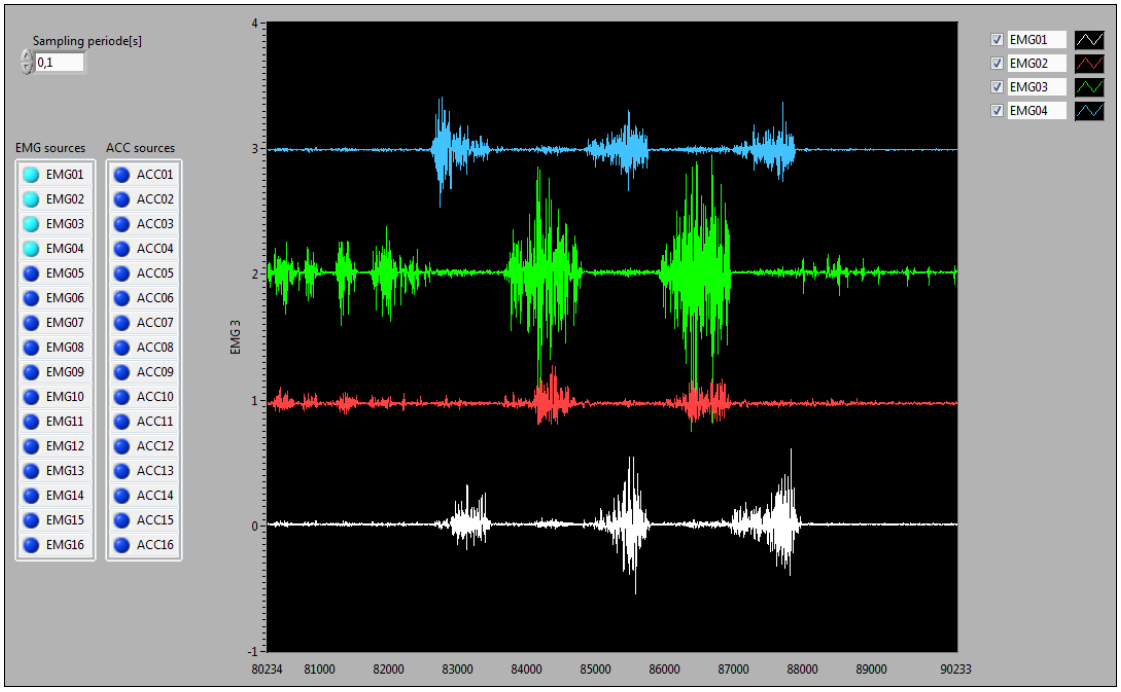


Figur 5.4: Beregning av sample *offset*

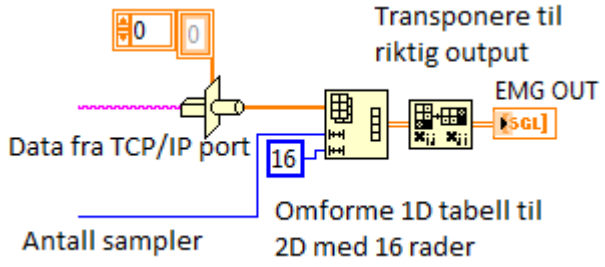
Figur 5.5 viser frontpanelet til en testVI som henter ut data fra Trigno-sensorene. TestVIen benytter de nye metodene for å hente ut data og det kan velges hvilke sensorer som det skal hentes data fra. Samplingsperioden kan varieres mens programmet kjører. Grafen viser måledata både for ACC- og EMG-sensorer. Dette programmet og testprogrammet fra Delsys er brukt for å teste ytelsen for innhenting av sensordata. Hensikten med å teste dette er å verifisere at ytelsen er tilstrekkelig til å implementere løsningen inn i ITKs bibliotek. Normalt er antall sensorer som brukes i hovedprogrammet i ITKs bibliotek fire eller fem stykker. Derfor er et minimumskrav at det skal kunne hentes ut data fra fem sensorer i sanntid. Dette fungerer for begge testprogrammene, så ytelsen er tilstrekkelig. Likevel er det ingen grunn til ikke å kunne hente ut data fra alle sensorer i sanntid.

Testprogrammet til Delsys fungerer med 16 sensorer. Men der kan det ikke velges hvilke sensorer som skal benyttes. Valg av sensorer tilfører en tidsforsinkelse i systemet som gjør at det ikke er mulig å hente ut data fra 16 sensorer i sanntid. For å redusere tidsforsinkelsen er to alternativer vurdert. Det første alternativet er å endre algoritmen som brukes for å velge hvilke data som skal brukes. Denne algoritmen er en enkel loop som henter ut data fra gitte indekser, men kan

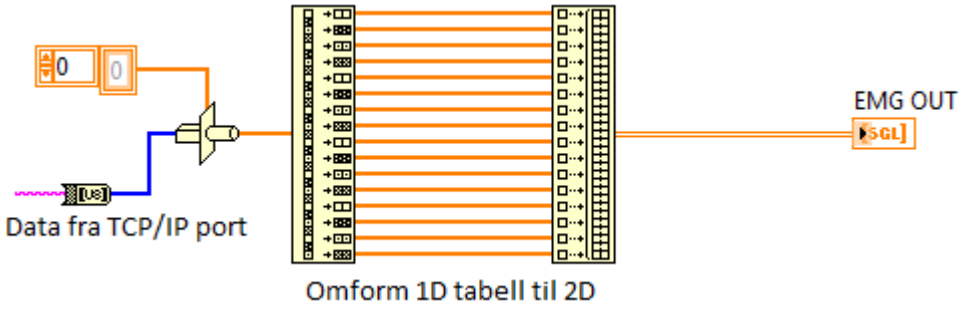
byttes ut med en mer effektiv algoritme for å redusere tidsforsinkelsen. Det andre alternativet er å endre uthenting av data i Delsys sin SDK. Et forslag for å gjøre dette er vist i figur 5.6a satt opp mot originalmetoden i 5.6b. Disse metodene for uthenting skiller seg fra hverandre ved at data fra TCP/IP-porten deles opp. Delsys henter ut data fra hver sensor til en 1D-tabell for så å kombinere alle 1D-tabellene til en 2D-tabell. Den foreslåtte metoden modifiserer data fra porten til en 2D-tabell direkte. Dette gjør at uthenting av data totalt sett bruker kortere tid og tillater uthenting av 16 sensorer samtidig.



Figur 5.5: Frontpanelet til VI for testing av Trigno-sensorene



(a) Modifisert metode i forhold til Delsys SDK



(b) Opprinnelig metode i Delsys SDK

Figur 5.6: Metoder for å hente ut data fra Trigno-sensorene

5.2.2 Egenskapsutregning

I ITKs bibliotek kan det velges hvilke egenskaper som skal brukes for hver enkelt sensortype. Denne muligheten er vist i figur 5.7. For å ta vare på data som skal brukes i egenskapsberegning, brukes det et egendefinert cluster kalt Window vist i figur 5.8a. Dette clusteret inneholder vinduslengde, hvor mye data som er tatt opp i sekunder og en tabell med data og tilhørende ID for hver enkelt sensor. IDen brukes for å bestemme hvilke data i Window-clusteret som skal være med i egenskapsberegningene. For eksempel hvis det er huket av for at en egenskap skal beregnes for EMG alene, velges bare de som har EMG i IDen sin. Denne metoden fungerer når tabellene for alle sensortypene har lik lengde. Men som beskrevet tidligere kan det hentes ut data med ulik lengde på grunn av at samplingsratene er forskjellige. Et eksempel på hvorfor denne metoden ikke fungerer er at hvis en tabell inneholder 200 EMG-sampler og en annen 14 ACC-sampler, vil ACC-tabellen tilpasses EMG-tabellen ved å initialisere

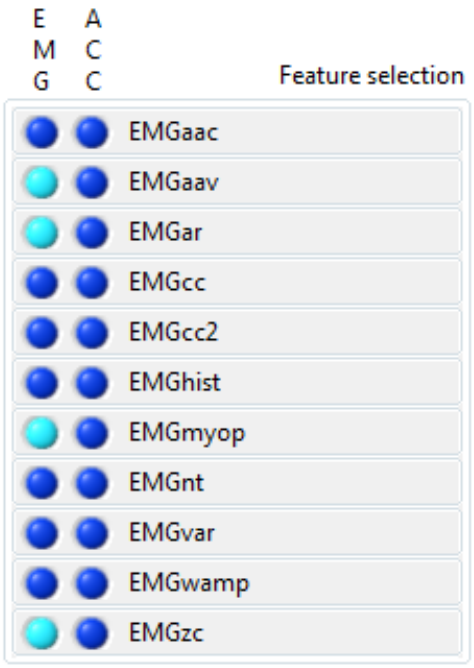
de 186 manglende elementene med null ved sammenslåing. Den mest brukte egenskapen for utregning på ACC-data er gjennomsnitt. Denne går drastisk ned når ACC-tabellen tilpasses og beregningen blir derfor ikke riktig. En måte å korrigere dette på er å tilpasse lengden på data fra sensorene med en VI som tar inn en tabell og hvilken lengde tabellen skal tilpasses til. Men dette betyr at det tilføres mer data enn det som hentes inn, noe som ikke skal være nødvendig. Istedet kan noen små endringer på Window-clusteret gjøres:

- Fjerne vinduslengde og bare bruke antall sekunder med data i stedet
- Gi hver enkelt sensortype en fast ID istedet for at hver enkelt sensor har en unik ID

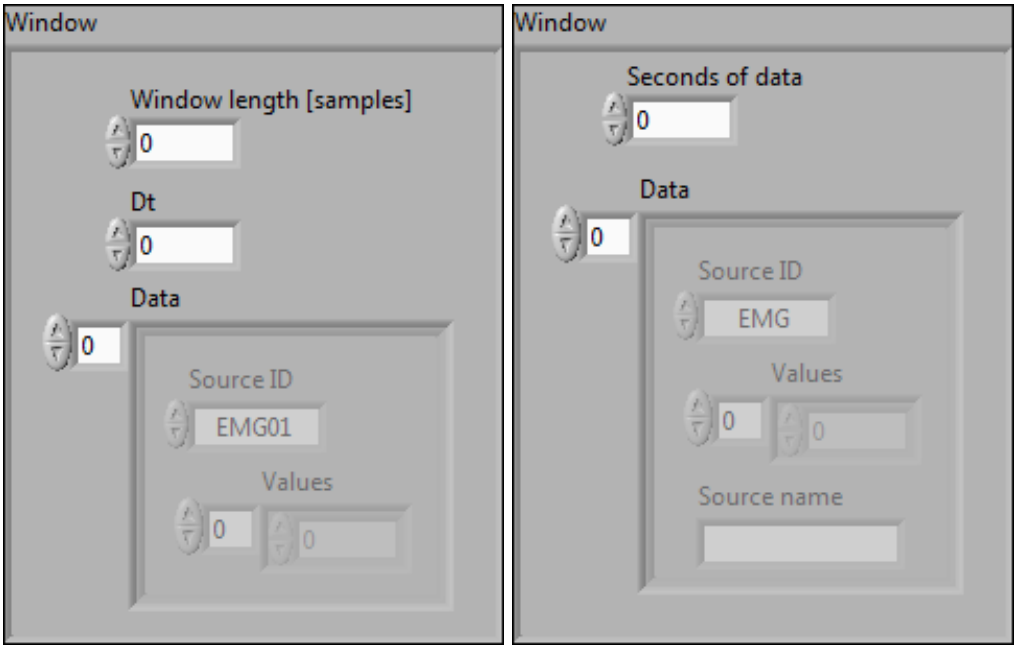
Disse endringene kan gjøres i ctl-filen til Window-clusteret og oppdateringen vil skje i hele programmet. En oppdatert versjon av clusteret er vist i figur 5.8b.

Det nye Window-clusteret løser bare litt av problemet, fordi de data som skal brukes i egenskapene settes sammen rett før beregningen gjøres. I seksjon 5.2.3 beskrives det hvordan sensormoduler som er enkle å gjøre endringer på kan lages. Deriblant hva utgangen fra sensor-VIen for innhenting av data fra sensorene kan være. Denne utgangen er en tabell av sensordata hvor data fra hver sensortype samles i en 2D-tabell og settes inn i en tabell av cluster med sensortypeID som indeks. Oversettelse mellom sensordatautgangen og Window-clusteret kan gjøres dynamisk.

Tilbake til problemene rundt egenskapsberegning. Her kan den omvendte oversettelsen gjøres, altså gjøre om Window-clusteret tilbake til sensordata utgangen. Ved å gjøre dette har hver sensortype blitt separert. Egenskapene kan da beregnes ved å oppgi indeks for sensortypen. For eksempel hvis en egenskap bare skal beregnes på EMG, brukes indeksen som samsvarer med EMG. Dersom en egenskap skal beregnes for både EMG og ACC vil egenskapen bli beregnet for EMG først og deretter for ACC.

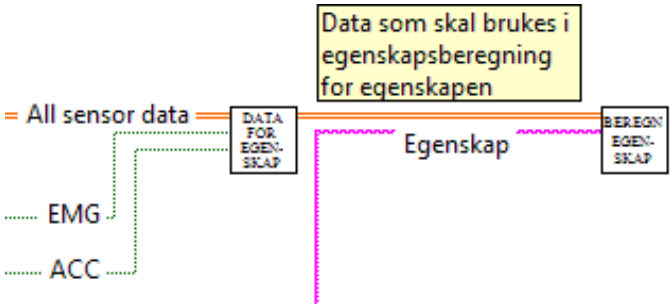


Figur 5.7: Valg av egenskap for sensortyper

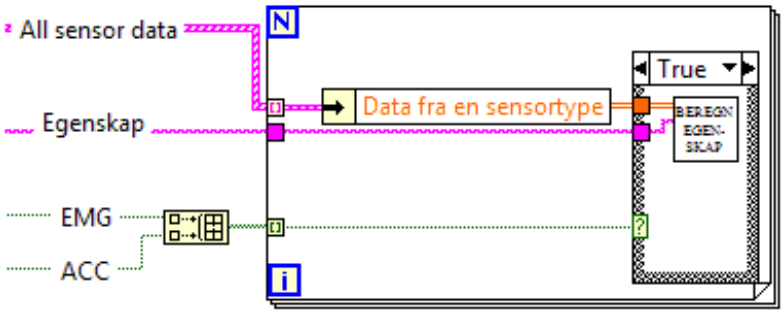


(a) Opprinnelig window cluster **(b)** Modifisert window cluster

Figur 5.8: Gammelt og nytt Window cluster



(a) Opprinnelig metode for beregning av egenskaper



(b) Modifisert metode for beregning av egenskaper

Figur 5.9: Egenskapsberegning

5.2.3 Modularitet

For å kunne gjøre det enkelt å legge inn nye sensorer og gjøre de modulære, må det finnes overordnede metoder for oppsett og uthenting av data fra sensorer. Disse overordnede metodene skal gjøre at nye sensortyper må tilpasses overordnet metode uten at andre endringer i programmet er nødvendig. De overordnede enhetene skal ha en fast inngang som er enkel å endre. Det løses ved å lage et cluster for hver sensortype og *refnumer* av disse clustrene. Inngangen kan da være et cluster av alle disse *refnumene*. Denne typen inngang vil dynamisk oppdateres ved endringer i clustrene til hver sensortype.

Det gjelder også å ha en dynamisk utgang, slik at det å legge til nye sensortyper ikke resulterer i at det må gjøres endringer i resten av programmet. En metode for å gjøre dette er å lage en 1D-tabell av clustere som inneholder 2D-tabeller hvor indeksene er beskrevet av sensortypene. Dette minner om en 3D-tabell, men hovedforskjellen er at data fra hver sensortype er adskilt fra

hverandre. Eksempelvis vil indeks 0 kunne tilsvare EMG og indeks 1 tilsvare ACC. Oversettelse til Window clusteret er da en enkel løkke som tilpasses automatisk med antall sensortyper som er støttet. Dermed vil ikke det å legge til nye sensortyper medføre endringer i oversettelsen til Window-clusteret.

5.3 Endringer i styringssystem

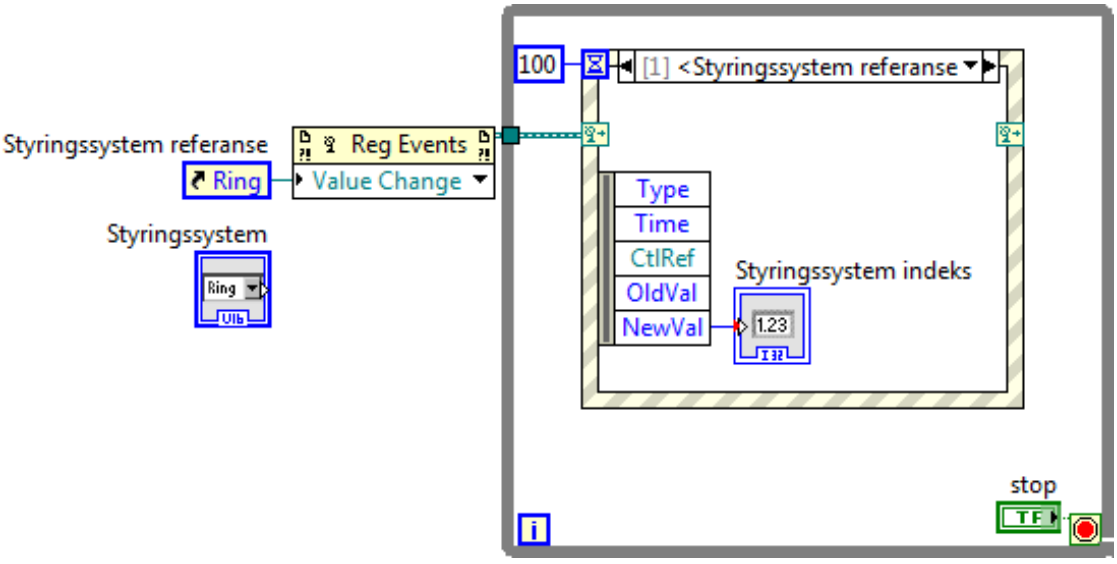
For å kunne velge mellom ulike styringssystemer kan det nyttes en *Ring* som gir ut en verdi som identifiserer hvilket styringssystem som skal brukes. Denne verdien kan da brukes for å bestemme hvilken metode som skal kjøre ved bruk av en *case*-blokk. Hvert styringssystem vil ha en metode inne i denne *case*-blokken.

Et bytte mellom ulike styringssystemer krever at det må gjøres endringer i brukergrensesnittet. For å ha muligheten til å endre GUIet når et nytt styringssystem blir valgt, kan en *Dynamic event handler* brukes. Figur 5.10a viser et system der en *Dynamic event handler* brukes for å vise indeksen til det valgte styringssystemet når typen endres. I GUIet vil det være mulig å aktivere og deaktivere om et objekt skal vises eller ikke. Dette gjøres ved å lage en referanse og velge egenskapen *visible* vist i figur 5.10b. Ved å kombinere de to metodene kan hva som skal vises styres ved valg av styringssystem.

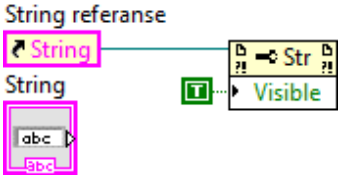
Denne endringen gjør at systemet er klart for at ulike styringssystemer kan benyttes. Dette gjelder så fremt det er laget egne moduler som kjøres ved korrekt indeks for styringssystemet.

Proporsjonalstyring

ITKs bibliotek har allerede støtte for ett styringssystem - mønstergjenkjenning. Det er ønskelig å utvide biblioteket med proporsjonalstyring. Et proporsjonalsystem er avhengig av treningssettet som brukes i opptrening av systemet. Treningssettene kan for eksempel være definert i tekstfiler som kan leses inn ved hjelp av en parser. Tekstfiler kan ha ulike formater, deriblant xml. I LabVIEW er det in-



(a) *Dynamic event handler*



(b) *Visible*-egenskap

Figur 5.10: LabVIEW enheter for GUI endringer

nebygde metoder for å hente inn data fra en xml-fil ved å bruke et cluster som datatype. Ved å bruke xml kan store parsere for å lese inn fra fil unngås. Selve treningssettene og metoden for å lese inn treningssettene er utviklet sammen med Anders Fougner.

Følgende egenskaper er ønskelig for treningssettene.

- **Pådrag**: en tabell som dikterer hvilke verdier som skal settes på aktuatorene.
- **Referanseverdi**: en tabell som brukes i trening av systemet. I mange tilfeller vil denne være identisk til pådragstabellen, siden det er disse pådragene brukeren oppfatter.
- **Sampleperiode**: en verdi som beskriver hvor mye data som skal hentes ut

for hver referanseverdi.

- **Frihetsgrad**: en tabell som sier hvilken DOF som tilhører hver enkelt verdi i pådrag og referanse.
- **Opptakstabell**: en tabell med *booleans* som blir brukt for å angi hvilke verdier i referansetabellen som skal settes i relasjon med sensordata.
- **Annen informasjon**: informasjon som forteller bruker hva som kommer.

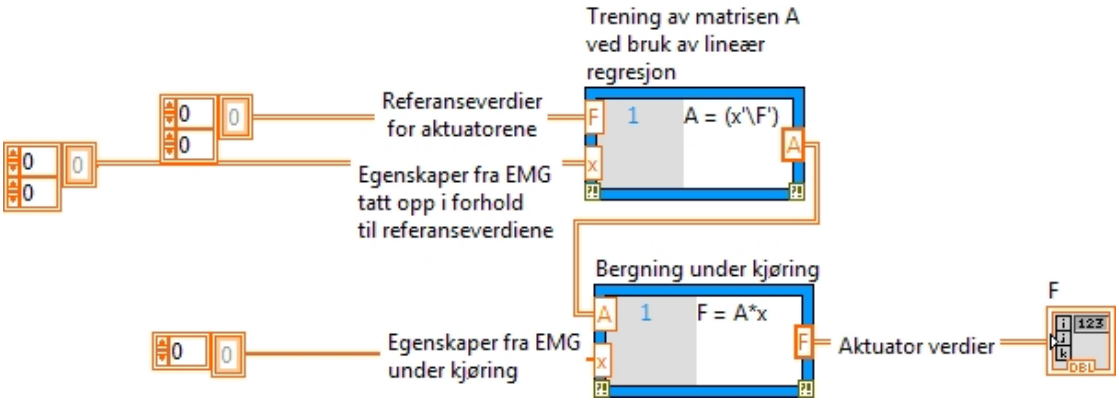
Ved å lage et cluster med verdiene over, vil det å lese fra og skrive til en xml fil være enkelt. Da gjenstår det å definere hvilket verdiområde referanseverdiene og pådragene skal ligge i. En frihetsgrad består av et midtpunkt og to endepunkter. Midtpunktet kan for eksempel defineres som 50, med 0 og 100 som endepunkter. Denne verdien kan være gjeldene for hver enkelt av de tre mulige motorfunksjonene; posisjon, hastighet og kraft. Eksempelvis vil en hastighetverdi som er 0 bety at det holdes maksfart i en retning og 50 bety at motoren står stille. Dette er litt selvmotsigende, men hva denne verdien er betyr ikke så mye. Det som betyr noe er at oversettingen til motorfunksjonene blir riktig.

Trening

Trening av et proporsjonalsystem bruker treningssettene for å instruere bruker i hva som skal gjøres. Samtidig tas det opp data fra bruker for å kunne sette referanseverdier i relasjon med oppsamlet data. Denne relasjonen lages ved å anvende estimeringsmetodene gitt for normal og dekoblet lineær estimering, se kapittel 2. For å gjøre dette enkelt kan Matlab-koden benyttes direkte ved å bruke en Mathscript-blokk. Mulig implementering for normal estimering ved bruk av denne metoden er vist i figur 5.11.

Filtrering

For å kunne styre en enhet med proporsjonalstyring er det nødvendig med filtrering av de estimerte verdiene.



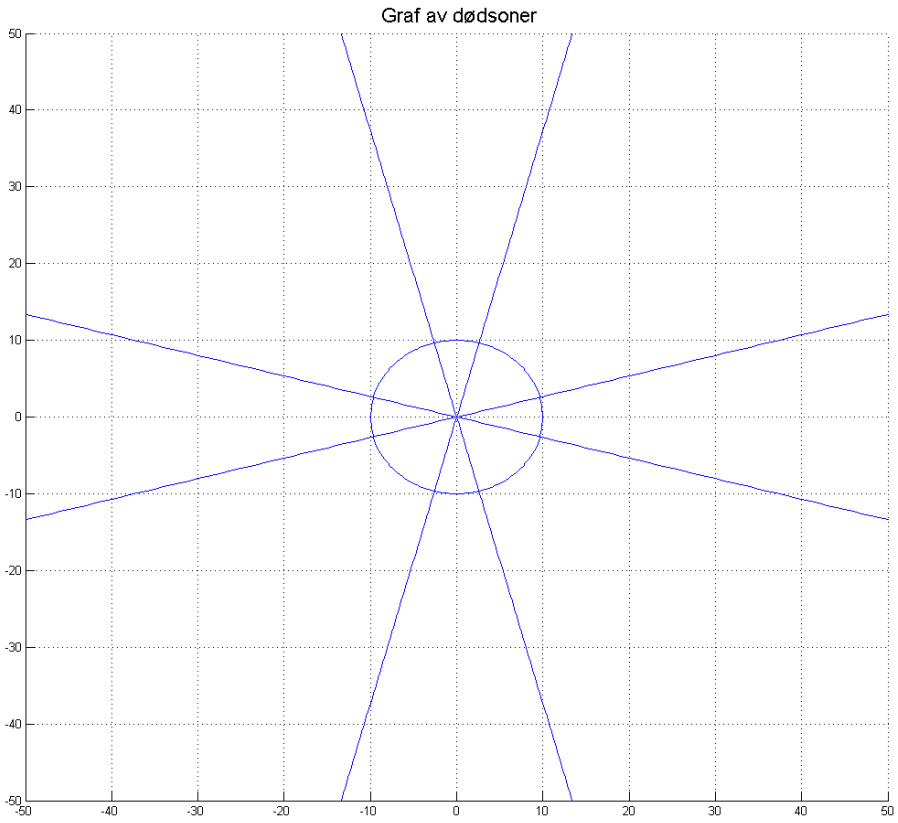
Figur 5.11: Kjøring av Matlab-kode i LabVIEW

Et aktuelt filter definerer dødsoner rundt aksene og origo. Hver akse i denne sammenheng er en DOF. Hvis et punkt havner nær en av aksene blir verdiene for de andre aksene satt til null. I tillegg vil alle verdiene settes lik null hvis radius til et punkt er mindre enn en gitt r0. Dette er for å sikre et hvileområde rundt origo. Områdene rundt aksene skal tillate bruk av bare en DOF. Den todimensjonale versjonen av dette er vist i figur 5.12. I det tredimensjonale tilfellet vil det være en kule rundt origo og koner rundt hver akse.

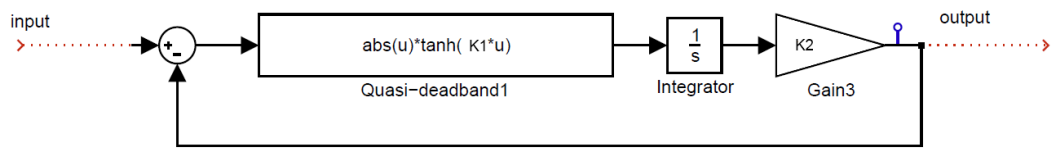
Ved bruk av proporsjonalstyring må det filtreres bort hurtige endringer i pådraget som skal på aktuatorene. Hurtige endringer gjør at protesen for eksempel ikke holder en konstant hastighet. For å straffe hurtige endringer i aktuatorene brukes filteret vist i figur 5.13, der K1 og K2 er justerbare for å finne en passende respons for filteret.

5.4 Endringer i utførende enheter

For å gjøre utførende enheter modulære brukes samme metode som for modularitet i sensorer. Det vil si at for hver utførende enhet lages et cluster hvor data for hver enhet blir lagret. I tillegg er det nødvendig med overordnede metoder hvor hver utførende enhet blir brukt. For å kunne velge hvilken utførende enhet som skal benyttes brukes samme metodikk som for valg av styringssystem, altså ved hjelp av en *Ring*.



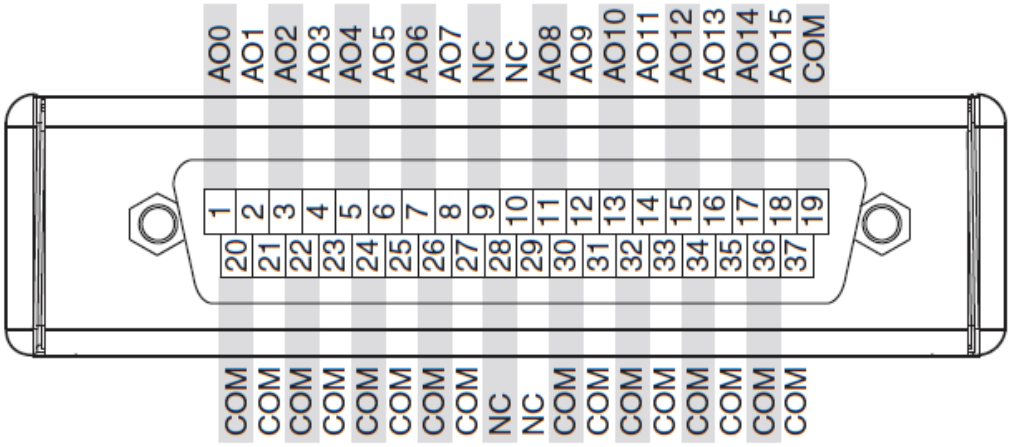
Figur 5.12: Filter for bruk av dødsoner



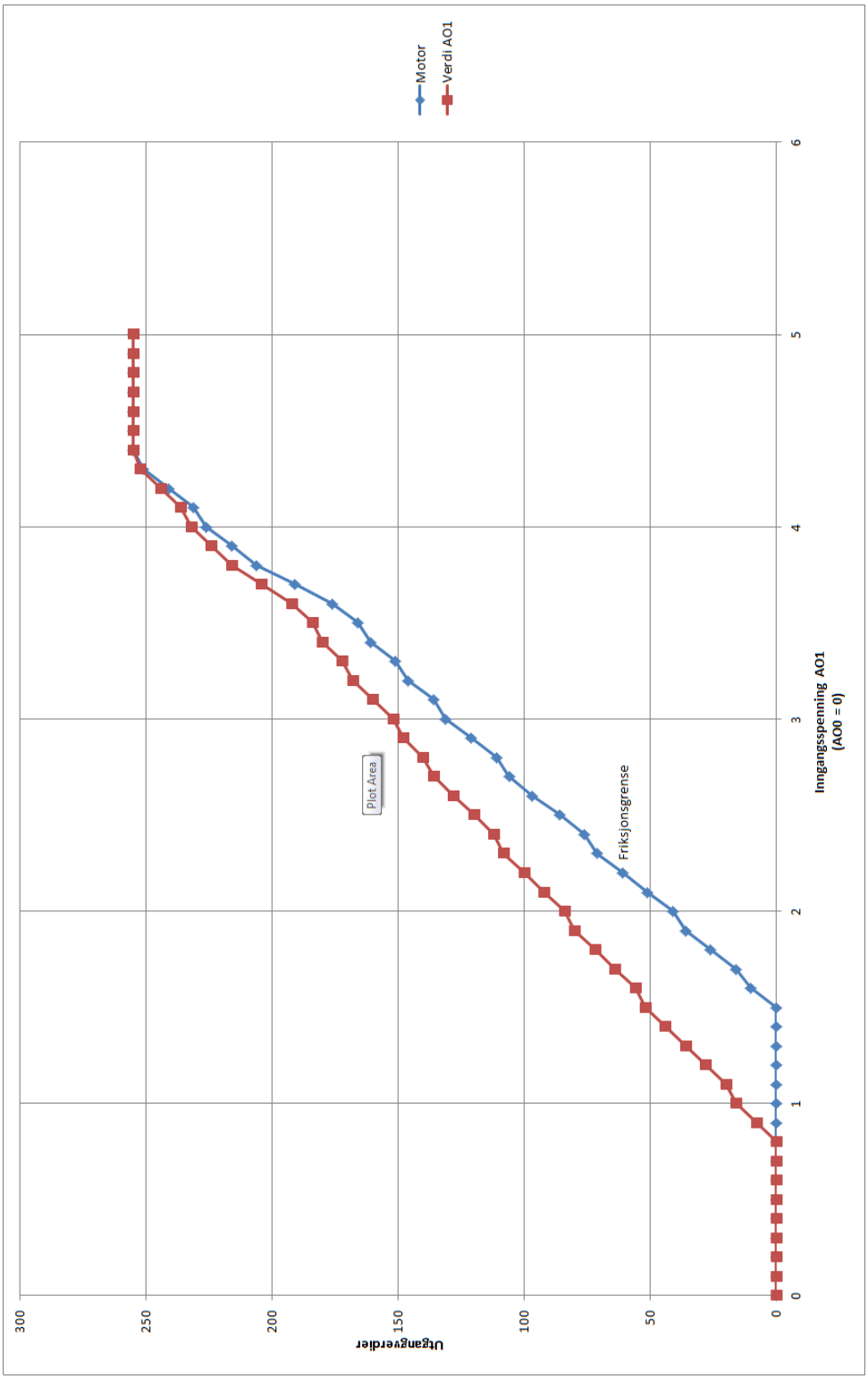
Figur 5.13: Filter for å fjerne hurtige endringer i pådraget, hentet fra [18]

Motion Control protesen

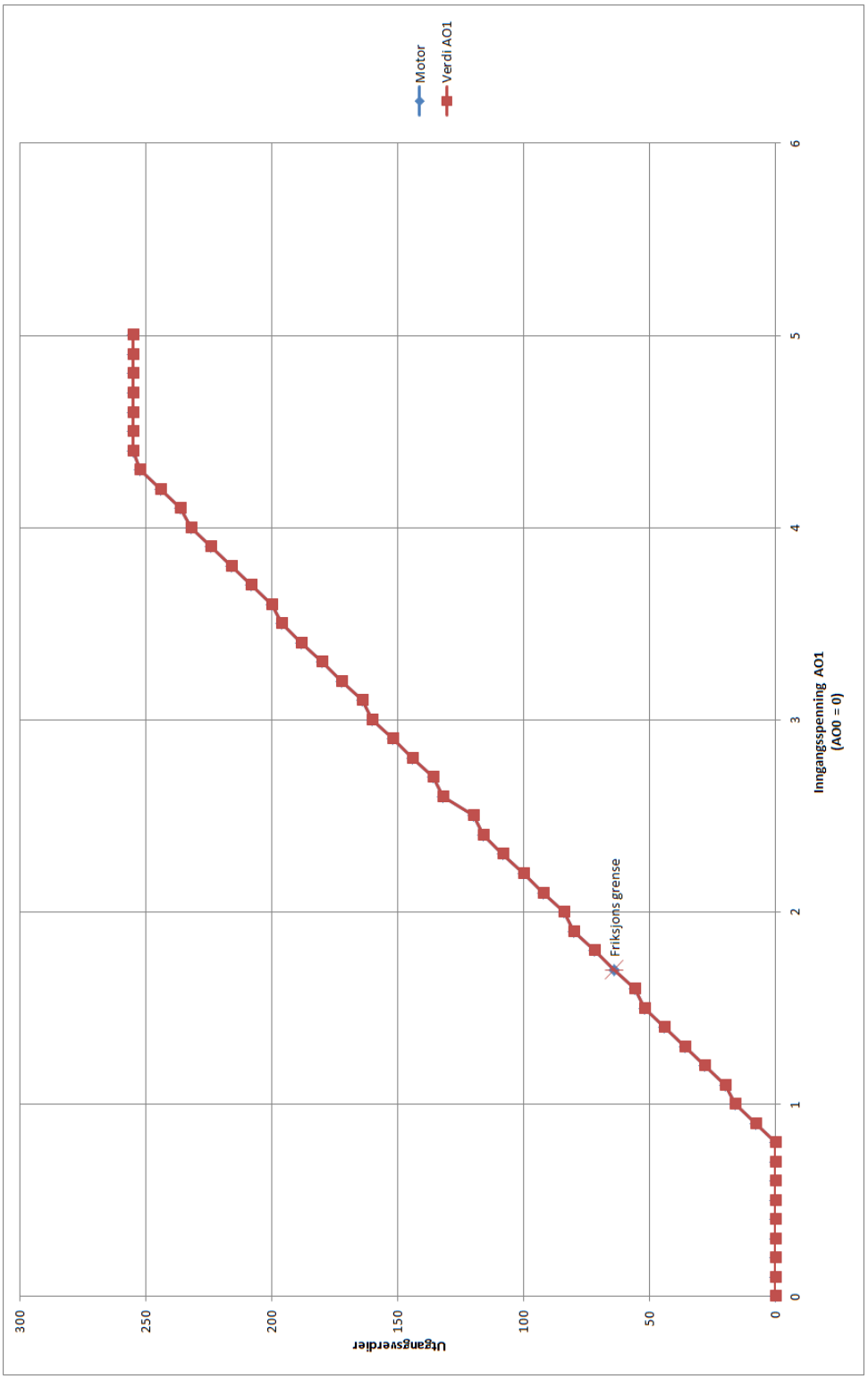
For implementeringen av mulighet for kontroll av protesen i ITKs bibliotek ble det satt opp tre alternativer, se seksjon 3.4. Metoden brukt i denne oppgaven er å erstatte Triad-elektrodene med koblinger til en DAQ. Det er enkelt å implementere bruk av DAQer i LabVIEW siden leverandøren av DAQene er den samme som har utviklet LabVIEW. Til denne oppgaven ble det kjøpt inn et NI 9264 *Measurement System*. Dette systemet består av et chassis og en NI 9264-enhet. Enheten består av 16 utganger og koblingskjema er vist i figur 5.14 [19]. Alle utgangene på enheten deler et felles koblingspunkt og vises som COM i figuren. For å koble DAQen til protesen ble ledningene til Triad-elektrodene brukt. Disse ledningene har et koblingspunkt som kobles til COM på NI9264-enheten. Ledningene som brukes for å styre protesen settes på hvert sitt AO-punkt. For å kunne kontrollere protesen gjennom DAQen ble det laget et program som setter en verdi på utgangene for å teste responsen til protesen. Utgangene AO0–AO3 blir brukt til å styre henholdsvis lukking, åpning, pronasjon og supinasjon av protesen. Figurene 5.15 og 5.16 viser karakteristikken til åpning av ProHand med ulik *gain* og *threshold*. Verdiene for motorpådrag og registrert AO1 er lest av ved bruk av Motion Control sitt program. Verdien for AO1 viser hvordan protesen tolker inngangssignalet fra AO1. Friksjonsgrensen i figur 5.15 og 5.16 er punktet hvor større pådrag medfører bevegelse av motoren. Denne grensen er viktig å kjenne for å få rask respons, altså at all dødperiode fram mot friksjonsgrensen er borte. Figurene viser også hva forskjellige *threshold* gjør med pådraget på motoren. *Gain* brukes til å justere hva maksverdien for motorpådraget kan være.



Figur 5.14: En NI9264-enhet



Figur 5.15: ProHand respons for åpning med, $gain = 65$ og $threshold = 51$



Figur 5.16: ProHand respons for åpning med, $gain = 65$ og $threshold = 0$

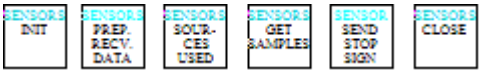
6 Resultat

Tidligere i denne rapporten har endringene som har blitt gjort i ITKs bibliotek blitt beskrevet. Dette kapittelet omhandler resultatene av disse endringene. Hver seksjon beskriver resultatene for hver enkelt del i den modifiserte Losiers modell.

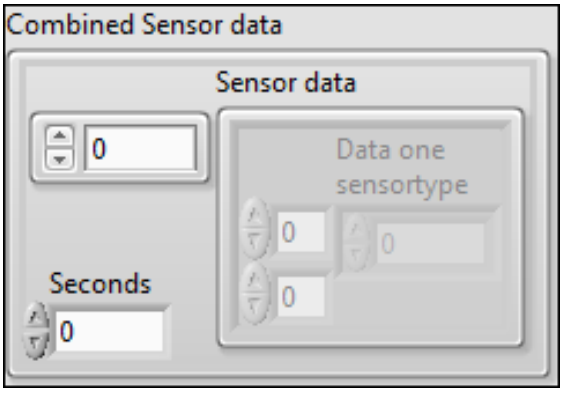
6.1 Sensorer

Figur 6.1 viser overordnede sensormoduler. Dette gjør at eventuelle nye sensortyper vil få sine egne metoder innen hver av disse VIene. Alle VIene deler en felles input som er et cluster av refnumer til hver sensortype. Sensormodulene ligger i mappen /Computer/VI/Sensors i prosjekthierarkiet og gjør som følger:

- **Init:** Initialiserer kommunikasjon med alle sensortyper
- **Prep Recv. data:** Klargjør for mottak av data fra sensorene
- **Sources Used:** Setter hvilke av sensorene som skal brukes, denne har en inngang som sier hvilke indekser som skal brukes for hver sensortype
- **Get samples:** Henter ut data fra sensorene og gir ut data på formen vist i figur 6.2
- **Stop:** Sier at sensorene skal stoppe å sende data
- **Close:** Bryter kommunikasjon med alle sensortyper



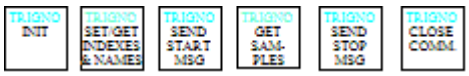
Figur 6.1: Sensormoduler



Figur 6.2: Cluster for sensordata

6.1.1 Trigno

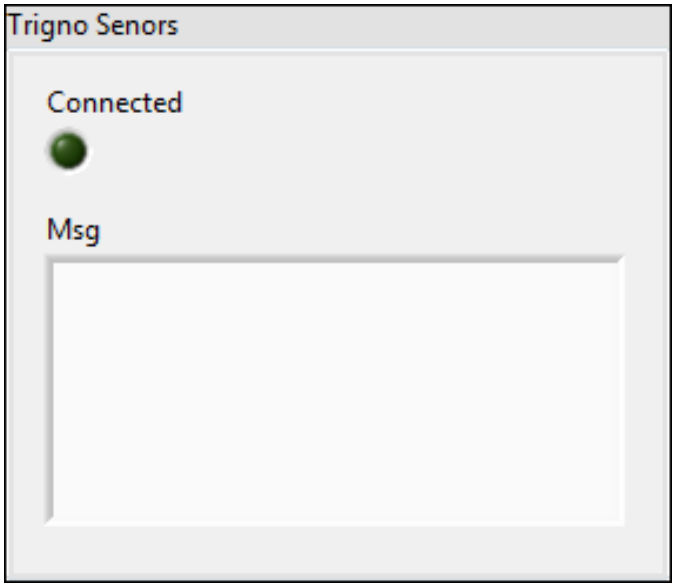
For å implementere at data kan hentes ut over USB ble VIene i figur 6.3 utviklet.



Figur 6.3: Trigno-moduler

De eneste sensortypene som er støttet av programmet er EMG- og ACC-sensorene fra Delsys. Disse sensortypene deler clusteret vist i figur 6.4. Dette clusteret består av datatyper som brukes i GUIet og for lagring av informasjon.

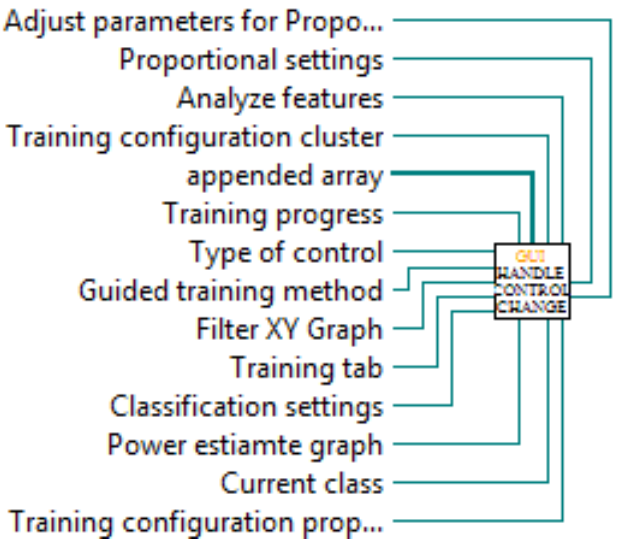
Objektene *Connected* og *Msg* gir operatøren av programmet informasjon om Trigno-sensorene og er de eneste som er en del av GUIet. De resterende objektene er skjult. *COMMAND CONNECTION ID*, *EMG CONNECTION ID* og *ACC CONNECTION ID* er TCP/IP koblinger til Delsys programvare. *Sample freq EMG* og *Sample freq ACC* er samplingsraten for EMG (2000 Hz) og ACC (148,1 Hz). *EMG indexes* og *ACC indexes* blir brukt for å velge hvilke data som skal sendes videre til egenskapsberegning. *Sample Offset EMG* og *Sample Offset ACC* lagrer offset fra forrige uthentingen av sensordata. *Started* brukes for å sikre at data kan hentes ut over USB etter at det er sendt startsignal til Trignobasen.



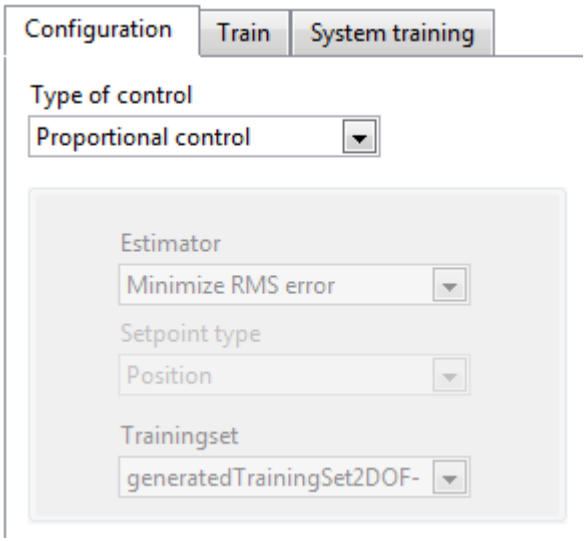
Figur 6.4: Cluster for Trigno-sensorene

6.2 Styringssystemer

For å håndtere valg av de ulike styringssystemene er det laget en VI som oppdaterer GUIet hvis det blir gjort bytte av styringssystem. Ikonet til VIen er vist i figur 6.5 og dens innganger er referanser til objekter i GUIet for alle styringssystemene. Dette viser at det kreves mye arbeid for å legge inn nye styringssystemer. Det kan bli enklere å legge til nye systemer ved å lage clustere for hvert styringssystem. Disse clusterne må inneholde referansene til alle GUI-enhetene hvert av systemene bruker. Figur 6.6 viser hvordan de forskjellige styringssystemene kan velges i *Type of control*.



Figur 6.5: VI for GUI-endringer ved bytte av styringssystem



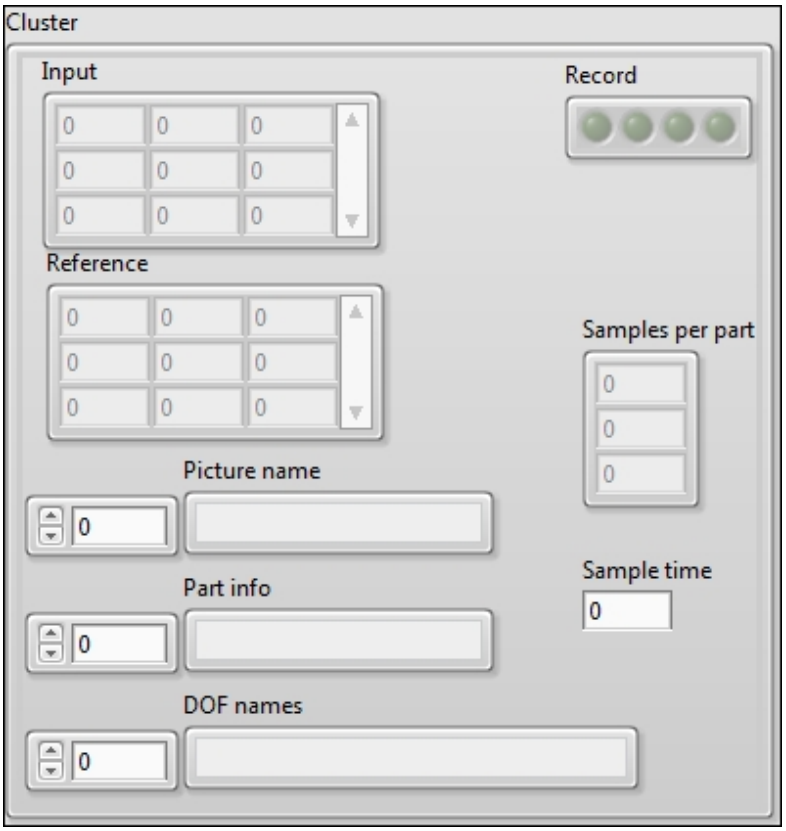
Figur 6.6: Valg av styringssystem som skal brukes under trening

Proporsjonalstyring

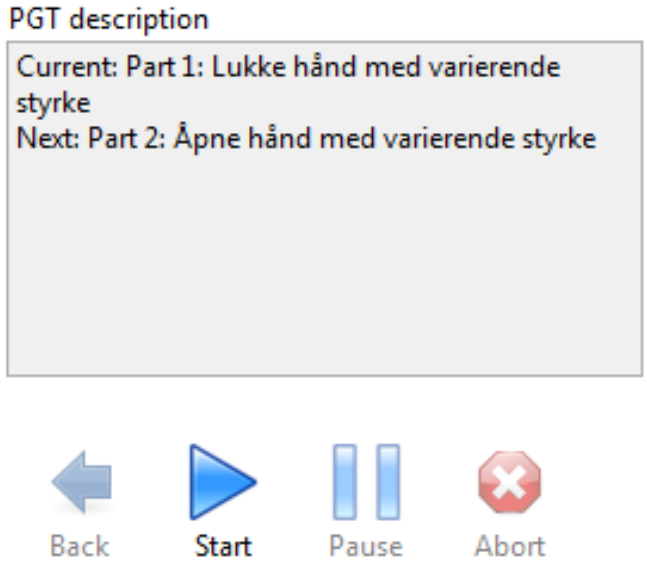
Implementasjonen av proporsjonalstyring begynte med å lage et treningsett som bruker clusteret vist i figur 6.7. Metode for å generere treningsett ligger i mappen /Computer/Trainingset/. Denne metoden lagrer data til en xml-fil ved hjelp av clusteret. Når treningssettet leses inn mottar clusteret data som kan brukes i trening av systemet. Clusteret består av følgende variabler:

- **Input:** Pådrag til aktuatorene.
- **Reference:** Referanseverdiene som skal settes i relasjon til data fra sensorene
- **Record:** Forteller hvilke av referanseverdiene som skal brukes i trening av systemet
- **Part Info:** En beskrivelse av hvilke bevegelser brukeren skal gjøre steg for steg
- **Samples per part:** Beskriver hvilke av samplene i *Input* og *Reference* som hører til de ulike delene av treningssettet
- **Picture name:** Et alternativ for å vise informasjon som bilde istedet for tekst
- **DOF names:** Beskriver hvilken frihetsgrad hver kolonne i *Input* og *Reference* hører til, for eksempel pronasjon/supinasjon
- **Sample time:** Sier noe om hvor mye data som skal hentes ut for hver referanseverdi

Hvordan de ulike treningssettene og treningsmetodene kan velges er vist i figur 6.6. Ved å trykke på fanen *Train* i figuren lastes det valgte treningsettet inn i systemet. I *Train*-fanen vises det med tekst hva brukeren skal gjøre i neste del, se figur 6.8. I tillegg vises knapper som for eksempel kan brukes for å gjøre nye opptak.



Figur 6.7: Cluster for treningssettene



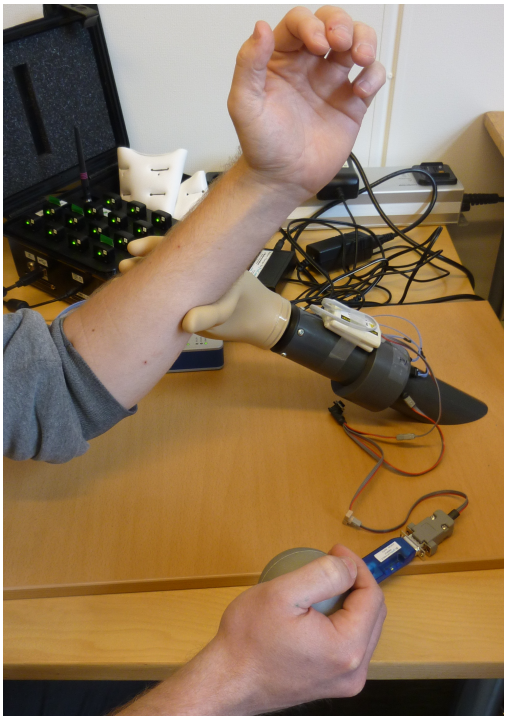
Figur 6.8: Beskrivelse av bevegelsen brukeren skal gjøre

Trening

I trening av systemet er det brukt EMG fra fire Trigno-sensorer som ble plassert på høyre underarm. Anvendt treningsmetode er PGT med kraft og hastighet som motorfunksjoner satt på MC protesen. Lukking av hånden ble trent opp ved å prøve å klemme like hardt rundt en ball som det protesen klemte på venstre underarm, se figur 6.9a. Åpning av hånd ble trent opp ved å holde rundt protesen og prøve å etterligne hvor hardt protesen åpnet seg, se figur 6.9b. For å trene supinasjon og pronasjon ble det brukt tre hastighetsnivåer, definert av treningssettet. Høyeste hastighetsnivå tilsvarer at brukeren skal supinere/pronere maks. Laveste hastighetsnivå tilsvarer at brukeren skal holde hånden i utgangsposisjonen. Figur 6.10 viser resultat av en trening, referanseverdiene er den blå grafen mens estimatet er vist i rød graf. Figuren viser at de estimerte verdiene følger referanseverdiene bra og ses på som gode resultater. Dette viser at ved å kombinere PGT og kroppen, kan åpning og lukking trenes relativt nøyaktig. De hurtige variasjonene rundt referanseverdiene gjør at et filter for å straffe hurtige endringer er ønskelig.

Kjøring av styringssystemet

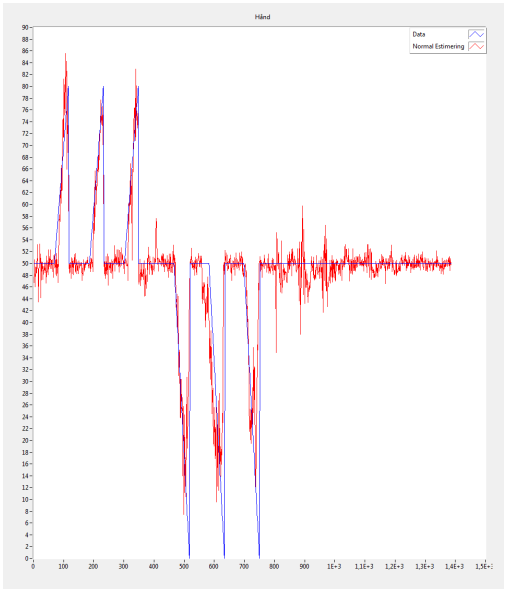
Ved kjøring av systemet vises operatøren dødsonefilter og aktuatorpådrag som punkter, se figur 6.11. Disse punktene er først filtrert med det ulineære filteret for å fjerne hurtige endringer. Styring av filteret for ønsket respons er vist i figur 6.12a. Her er det også en mulighet for å skalere hver av aksene. Hensikten med dette er å kunne nå maksimalverdiene til hver av bevegelsene. Dødsonefilteret kan styres ved bruk av elementene i figur 6.12b. Grafen oppdateres automatisk ved endringer i parameterne for dødsonefilteret. Figur 6.13 viser estimerte verdier og tilhørende filtrerte verdier for pådrag. Rød og mørkeblå graf er estimerte verdier for henholdsvis supinasjon/pronasjon og åpne/lukking. Lyseblå og grønn graf er de filtrerte verdier som settes på aktuatorene for henholdsvis supinasjon/pronasjon og åpne/lukking. Disse filtrene fungerer for flere frihetsgrader enn som vist i figurene, men de er spesiallaget for MC protesen. Dette betyr at NTNU robothånd ikke vil kunne kontrolleres så godt som



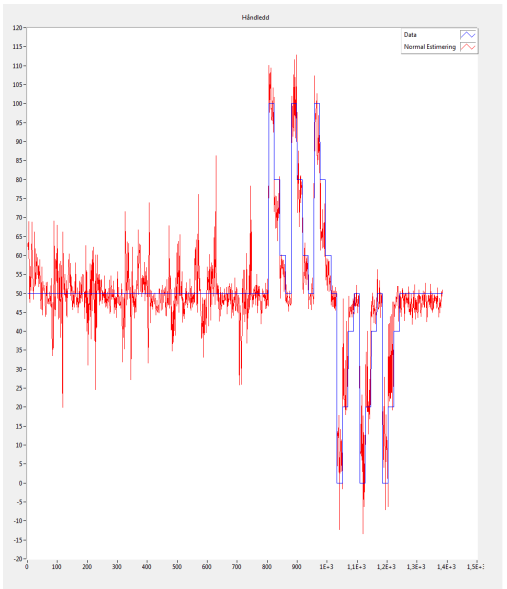
(a) Lukke

(b) Åpne

Figur 6.9: Trening av lukking og åpning



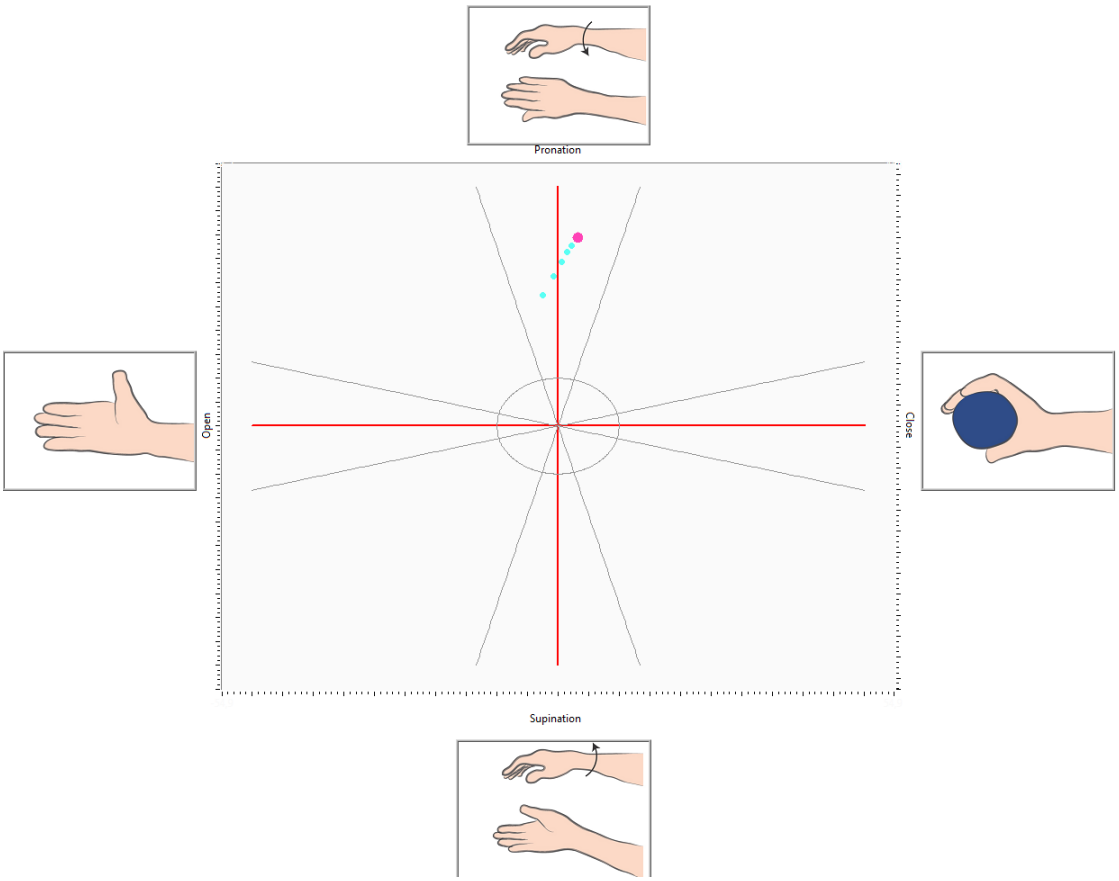
(a) Åpning/Lukking



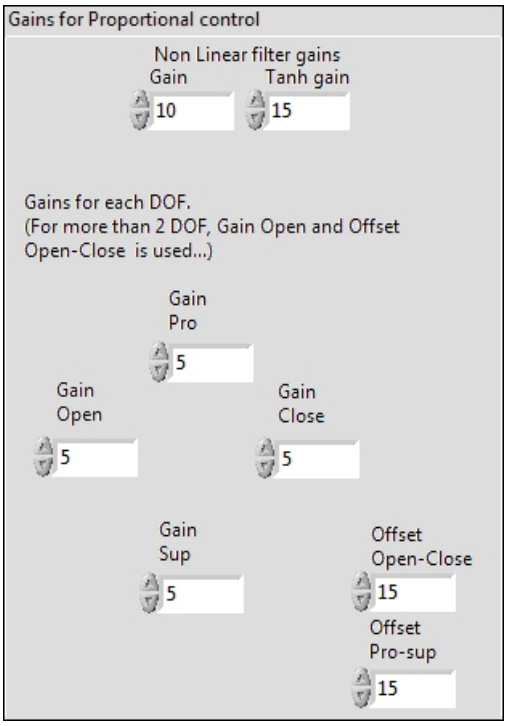
(b) Supinasjon/Pronasjon

Figur 6.10: Estimerte verdier i forhold til referanseverdiene

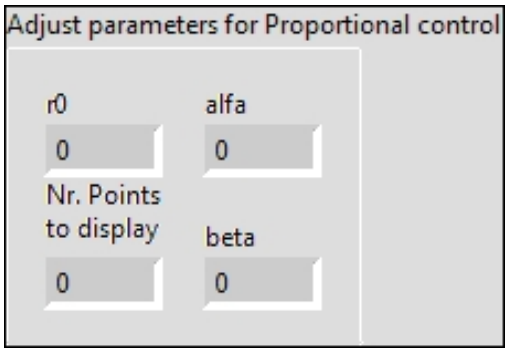
protesen. Robothånden vil kunne brukes i proporsjonalstyring, men det krever at det lages egne treningssett for denne.



Figur 6.11: Visning av hvor et punkt ligger i forhold til dødsonefilteret

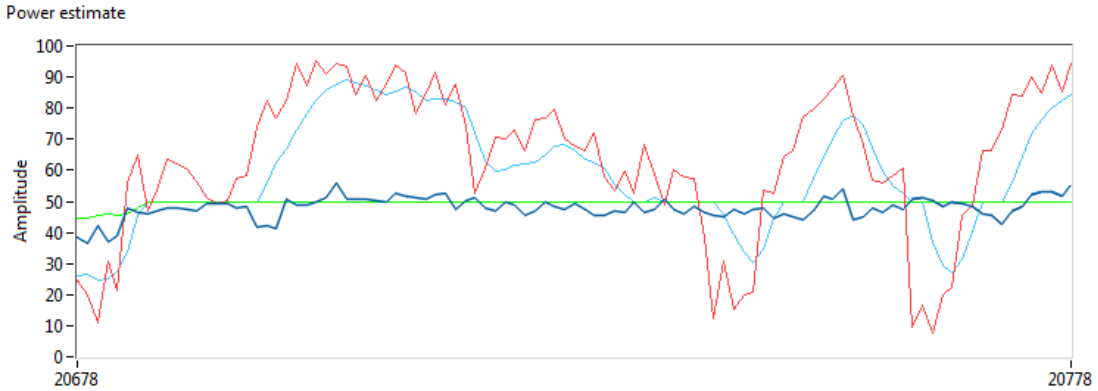


(a) Parametere for demping av hurtige endringer og skalering



(b) Parametere for dødsonefilter

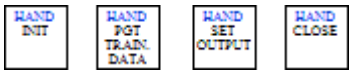
Figur 6.12: Clustere for filter parametere



Figur 6.13: Grafer som viser estimerte verdier og filtrerte verdier

6.3 Utførende enheter

De utførende enhetene som er støttet blir brukt i de overordnede modulene vist i figur 6.14. Den utførende enheten som skal brukes er gitt av en *Ring*. Alle utførende enheter har clustere med samme GUI som figur 6.4, men andre skulte datatyper.



Figur 6.14: Overordnede moduler for utførende enheter

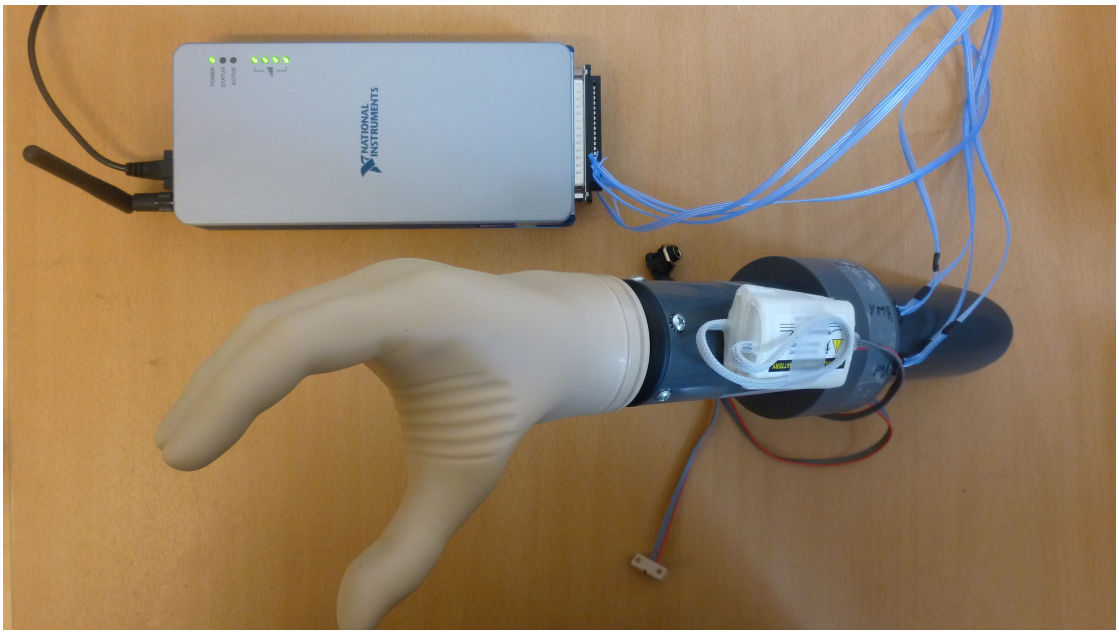
De overordnede modulene er som følger:

- **Init:** Initialisering av de utførende enhetene som er støttet.
- **Set data og set training data:** Setter data som skal bli påtrykt motoren under henholdsvis kjøring og trening. Ved omstrukturering av koden vil disse kunne slås sammen til en enhet.
- **Parse Trainingset:** Tar inn 2D-tabell og hvilke DOF som er definert i treningssettet og gir ut en tabell som er tilpasset den enheten som er valgt. Dette gjøres ved at en liste av hvilke DOF enheten består av sammenlignes med de som er definert i treningssettet. Det er dette som gjør at treningssettene som er laget for MC protesen ikke vil fungere like bra for NTNU robothånd, siden robothånden har flere DOF enn protesen. Dette vil kunne løses ved å innføre at en bevegelse kan bestå av flere DOF. For eksempel består bevegelsen åpne hånd av en DOF for protesen, mens den for robothånden består av tre DOF.
- **Close:** Stenger kommunikasjon til enhetene.

MC protese

Figur 6.15 viser hvordan en trådløs DAQ er koblet til protesen. Denne DAQen kommuniserer med en PC over trådløst nettverk for å kunne kontrollere protesen fra PC. Motorpådraget som settes på protesen er designet til å være

lik friksjonsgrensen ved 0 V på utgangene på DAQen og 5 V ved maks motorpådrag.



Figur 6.15: Oppkobling mellom DAQ og protese

7 Diskusjon

I dette kapittelet vil de implementerte funksjonalitetene bli sammenlignet med de ønskede funksjonene fra tabellen 4.1

7.1 Sensorer

Modularitet

Modularitet kan oppnås ved å gjøre hovedinngangen til alle VIene og utgangen til VIen for sensordata dynamisk. Dette betyr at eventuelle nye sensortyper bare behøver minimalt med kodeendringer for å kunne utnyttes i de overordnede modulene. De endringene som vil måtte gjøres består av å lage et cluster for den nye sensortypen, sørge for at operatøren kan velge hvilke av de nye sensorene det skal hentes ut data fra og valg av egenskaper som skal gjelde for den nye sensortypen. Alle disse endringene finnes det allerede eksempler på i ITKs bibliotek, så det vil bare være å følge disse metodene. Da er det bare nødvendig å lage egne moduler for de nye sensortypene. Dette bidrar til at det er enkelt å legge til nye sensorer. Siden *Typedefs* er brukt i biblioteket vil eventuelle endringer i eksisterende sensortyper håndteres automatisk.

Trigno-sensorene

Endringene i uthenting av data fra Trigno-sensorene har medført at det kan hentes ut data fra 16 sensorer samtidig med en samplingsperiode på minimum 30 ms. Dette er tilstrekkelig for ITKs bibliotek fordi det brukes en samplingsperiode på 100 ms. Protesebrukeren vil ikke merke forskjell på om det hentes ut 30 ms med data eller 100 ms, fordi reaksjontiden til mennesker er høyere dette.

Data fra Trigno-sensorene overføres over TCP/IP for å kunne aksessere data fra

sensorene i LabVIEW. Dette er ikke en optimal løsning, fordi det bare tilfører mer tidsforsinkelse i systemet. Derfor var et av *bør*-kravene(K1.1.3) at det skulle være mulig på hente ut data direkte fra USB. Altså uten bruk av programmet *Trigno Analog Ouput*. Dette er ikke implementert siden det ikke er en høyt prioritert funksjon. På grunn av forsinkelser fra leverandøren av protesen ble det litt tid til å se nærmere på kravet. I hovedsak ble det sett på to metoder for å overføre data rett fra Trigno-basen. For å se dataflyten over USB mellom Trigno-basen og PCen ble det brukt et program kalt USBlyzer. Den første metoden inkluderer bruk av NI-VISA, et program som kan lage generiske drivere til enheter koblet til PCen. Denne generiske driveren er enkel å implementere i LabVIEW, men den må kunne oversette og bruke data sendt fra Trigno-basen. For å få til dette trengs en spesifikasjon fra leverandøren over hvordan kommandoene sendes over USB byte for byte. Den andre metoden er å anvende driveren som Delsys leverer direkte ved å lage støtte for denne gjennom en DLL. Denne driveren heter SiUSBXp og er lett tilgjengelig. Ved å be Delsys om informasjon om hvordan dataflyten er og hvordan data skal tolkes, kan denne driveren benyttes for å hente ut data rett inn i tredjeparts programvare.

Motion Control sensorer

I denne oppgaven er hovedsensoren for uthenting av EMG-signaler Trigno-sensorene. Siden MC har levert protesen ville det vært fint å kunne benytte seg av Triad-elektrodene også. Ved bruk av det trådløse grensesnittet kan data fra elektrodene hentes direkte fra protesen. Men ved å bruke en DAQ for å styre protesen blir denne muligheten borte. Dette betyr at data fra elektrodene må hentes ut på en annen måte, som for eksempel ved hjelp av en DAQ hvor hver sensor har en dedikert inngangskanal. Dette er et *bør*-krav fordi det ikke er nødvendig å bruke disse sensorene siden Delsys sine sensorer er de som er foretrukket her.

7.2 Styringssystem

Operatøren av programmet har valg mellom to styringssystemer; mønstergjenkjenning og proporsjonalstyring. Valg av hvilket av disse som skal benyttes er enkelt. Men det å legge inn nye styringssystemer krever arbeid. Et eksempel er at et nytt styringssytem kan ha mange GUI-objekter som må legges til som innganger til VIen i figur 6.5. Dette resulterer i at det blir vanskelig å holde styr på hva som skal vises og ikke. For å gjøre endringer på dette bør programmet omstruktureres slik at også styringssystemene har overordnede moduler slik som sensorene har. Dette kan gjøres ved at inngangene til disse systemene vil være *refnumer* av clustere som inneholder alle GUI-enheter tilhørende et styringssystem.

Proporsjonalstyring

Styringssystemet er spesiallaget for bruk med protesen, slik at det å kunne kombinere motorfunksjoner er utelukket. Systemet har allikevel mulighet til å kunne styre posisjonen til NTNUs robothånd. Protesen er strømstyrt. Det betyr at både kraftstyring og hastighetsstyring av protesen er mulig, men ikke i kombinasjon med hverandre. Det finnes treningsmetoder hvor PGT kombineres med andre metoder for alle motorfunksjonene. Dette er vist ved trening av protesen hvor åpne- og lukkefunksjonene er trent opp ved å føle hvor mye kraft protesen yter. Supinasjon og pronasjon er trent opp ved å se hvilken hastighet motoren går med, disse kan også trenes opp ved å høre på motorlyden hvor fort den går. Jo høyere lyden er jo fortere går motoren. Resultatene viser at noen av treningsmetodene diskutert i seksjon 4.2 fungerer godt.

Valg av treningssett er enkelt, men treningssettene er laget spesielt for protesen. Dette gjør at de ikke fungerer optimalt for robothånden. Allikevel er fullt mulig å lage treningssett som kan fungere på begge de utførende enhetene. Treningssettene bruker bare en frihetsgrad om gangen. Dette medfører at kontroll av alle frihetsgrader samtidig blir vanskelig. Utvikling av treningssett er derfor viktig å prioritere i videre arbeid.

7.3 Utførende enhet

Både MC sin protese og robothånden kan benyttes som utførende enheter i ITKs bibliotek. Ved eventuelt behov for flere utførende enheter vil modulariteten til programmet sørge for at det er enkelt å legge inn disse utførende enhetene.

MC protesen

Protesen er implementert med en trådløs DAQ. DAQen er en stor boks og er ikke hensiktsmessig å måtte gå rundt med for å kunne kontrollere protesen. Protesen kommer med ferdig trådløs kommunikasjon over Bluetooth og dette grensesnittet er bedre å bruke enn en DAQ. Problemet er at det per dags dato ikke er mulig å sette motorpådrag. Den eneste parameteren som kan styre motorene på protesen er forskjellig strøm. Dette gjør at de i prinsippet er kraft- eller hastighetsstyrt. Å sette posisjon på protesen er ikke mulig.

8 Konklusjon

ITKs bibliotek har blitt utvidet med et nytt styringssystem, proporsjonalstyring, og bruk av protese fra MC som utførende enhet. Biblioteket har blitt enklere å vedlikeholde med tanke på å legge til nye sensortyper og utførende enheter. Når det gjelder styringssystemene er det ikke gjort noen endringer. Men det er viktig at det lages overordnede moduler for disse på lik linje som for sensortypene og de utførende enhetene.

Løsningen for proporsjonalstyring viser at det er mulig å kontrollere flere frihetsgrader samtidig proporsjonalt. Men det kreves bedre treningssett enn det som er laget for å kunne kontrollere kombinasjonen av de forskjellige frihetsgradene mer nøyaktig. Treningen av proporsjonalstyring ved bruk av PGT i kombinasjon med andre elementer, viser seg å være en effektiv og enkel måte å trene opp systemet på. Hvordan bruk av kraft i proporsjonalstyring kan trenes opp, er den viktigste delen. Ved å bruke kroppen til å føle hvor stor kraft protesen yter kan en protesebruker enkelt kunne trene opp igjen systemet hvor som helst. Denne treningsmetoden er bare testet av personer som ikke har måttet foreta amputasjoner. Testpersonene har brukt en klemmeball for å ha motstand under klemming. Dette betyr at vedkommende både kan se og føle hvor hardt han/hun klemmer i forhold til hvor hardt protesen klemmer. Dermed er det uvisst om denne metoden er like intuitiv og enkel for en protesebruker. En protesebruker vil bare føle hvor hardt protesen klemmer og ikke hvor hardt han/hun klemmer med den amputerte hånden. Ved å bruke proporsjonalstyring med MC protesen er det vist at det er mulig å kontrollere en protese bedre enn med et mønstergjenkjenningssystem.

8.1 Forslag til fremtidig arbeid

Sensorer

Trigno

Det er ønskelig å kunne hente ut data rett fra Trigno-sensorene uten bruk av *Trigno Analog Output*. Dette kan gjøres ved å kontakte leverandøren, Delsys, for å få tilgang til den driveren som brukes for å sende data over USB. I tillegg er det viktig å få greie på hvordan data som mottas skal tolkes.

Nye sensortyper

Trond Suleng har laget en prototype på multimodal enhet i hans masteroppgave[20]. Denne enheten har støtte for blant annet Trigno-sensorene og kraftmåling, hvor målingen vil være mulig å bruke i ITKs bibliotek.

Styringssystem

Omstrukturering

For å gjøre det enkelt å kunne vedlikeholde styringssystemene, må strukturen i ITKs bibliotek endres.

Proporsjonalstyring

Filtrering

Testing av dødsonefilteret for å finne eventuelle feil ved høyere DOF enn to.

Autokalibreringsknapp

Ved bruk av proporsjonalstyring kan hver bevegelse skaleres til å tilpasses hva pådraget skal være gitt at brukere for eksempel supinerer maks. Ved å ha en autokalibreringsknapp kan hver av skaleringene og offsetene finnes enkelt ved å gi en rekke kommandoer til bruker.

Treningssett

Lage nye og forbedrede treningssett for å bedre kontroll av flere DOF simultant.

Styringsenhet

MC protese

Kobling til ITKs bibliotek

Endre firmware slik at de trådløse egenskapene til protesen kan brukes istedet for å koble DAQen til protesen. Dette vil medføre at brukeren slipper å ha ledninger til DAQen som vil måtte være i en veske eller lignende.

Referanser

[1] J. Bersvendsen, *Control of a multifunction arm prosthesis model*. NTNU, 2011.

[2] Ådne Solhaug Linnerud, *Styring av multifunksjonell robothånd*. NTNU, 2011.

[3] Y. Losier, "Shoulder complex motion based input strategies for prosthetic limb control," Ph.D. dissertation, UNIVERSITY OF NEW BRUNSWICK, 2012.

[4] A. Fougner, Ø. Stavdahl, P. J. Kyberd, Y. G. Losier, and P. A. Parker, "Control of upper limb prostheses: Terminology and proportional myoelectric control – areview," 2012.

[5] E. Scheme and K. Englehart, "Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use." *Journal of rehabilitation research and development*, vol. 48, no. 6, p. 643, 2011.

[6] R. Duda, P. Hart, and D. Stork, "Pattern classification," *John Willey & Sons*, 2001.

[7] B. Lock, A. Simon, K. Stubblefield, and L. Hargrove, "Prosthesis-guided training for practical use of pattern recognition control of prostheses." Myoelectric Symposium, 2011.

[8] A. Fougner, *Fig02(Painting-8MotionsColor).pdf*, 2011.

[9] A. Simon, B. Lock, K. Stubblefield, and L. Hargrove, "Prosthesis-guided training increases functional wear time and improves tolerance to malfunc-

tioning inputs of pattern recognition–controlled prostheses." Myoelectric Symposium, 2011.

[10] Delsys Inc. (2009, October) Trigno™ Wireless System User's Guide. [Online]. Available: http://www.delsys.com/attachments_pdf/Trigno%20Wireless%20System%20Users%20Guide%20(MAN-012-1-2)-web.pdf

[11] ——. (2012) Trigno™ Digital SDK. [Online]. Available: http://www.delsys.com/Attachments_pdf/Delsys%20SDK%20Brochure.pdf

[12] K. S. Håkonsen, *Utvikling og evaluering av en flerfunksjonell armprotesemodell*. NTNU, 2010.

[13] Motion Control, Inc. (2010) Prosthetist Instructions for the MC Wrist Rotator & ProWrist Controller. [Online]. Available: http://www.utaharm.com/files/?action=login&username=mc&password=mc

[14] ——, "Fitting Procedures Course Workbook," 2011.

[15] ——, "MC Triad Preamp System : The "Swiss Army" Preamp," 2011.

[16] ——, "mcDEV – Wrist DLL – API Use," September 2012.

[17] Klaus Biggers, Motion Control.

[18] A. Fougner, *Proportional Myoelectric Control of a Multifunction Upper-limb Prosthesis*. NTNU, 2007.

[19] National Instruments Corp. (2009, October) OPERATING INSTRUCTIONS AND SPECIFICATIONS NI 9264. [Online]. Available: http://www.ni.com/pdf/manuals/374404e.pdf

[20] T. Suleng, *Design av multimodal input-sensor for armproteser*. NTNU, 2012.

A Notat om utførende enheter

NTNU Robothånd

Ødelagte motorer

Det er to motorer tilhørende robothånden som er defekte.

1. Motor for fleksjon/ekstensjon av indeksfinger
2. Motor for adduksjon/abduksjon av tommel

Indeksfingeren begynte å smelte under testing, dette kommer av at den ikke kunne nå sitt ytterpunkt. Det resulterte i at strømmen til servomotoren økte og motoren ble overopphetet. Dette kommer av at tommelen var i veien for at indeks fingeren kunne nå sitt ytterpunkt. Denne motoren må derfor **ikke** kobles til servokontrolleren. Det er anbefalt å kjøpe inn en ny motor.

Motoren for tommelen er for svak, derfor er det ingen vits i å bytte den ut før en sterkere motor med samme dimensjoner er tilgjengelig.

MC protese

Trådløs tilkobling

DAQen som brukes lager et trådløst nettverk som PCen kan koble seg til på. Det tar litt tid fra oppstart av DAQen til dette nettverket er tilgjengelig. Deretter må chassiet resettes i Measurments og Automation for at NI9264-enheten skal registreres.

B Innhold CD

Dette kapittelet beskriver de elementene som er vedlagt masteroppgaven. Hver paragraf er en mappe på CDen. En elektronisk kopi av masteroppgaven ligger også på CDen.

Artikler

Her ligger de fleste av artiklene som er brukt i oppgaven.

Eksperimentell kommunikasjon

Her ligger elementer for kommunikasjon mellom LabVIEW og DLLen fra MC. Med dette menes ulike versjoner av C#-testprogram og VIer som kan kobles til disse testprogrammene.

Programvare

Her ligger programmet til MC og kildekode for dette programmet. Det ligger også installasjonsfiler for *Trigno Analog Output*.

LabVIEW-prosjekt

Her ligger ITKs utvidede programvarebibliotek, men dette kommer til å bli utvidet av Anders Fougner etter denne masteren. Så det er anbefalt å bruke den versjonen som Anders Fougner lager.