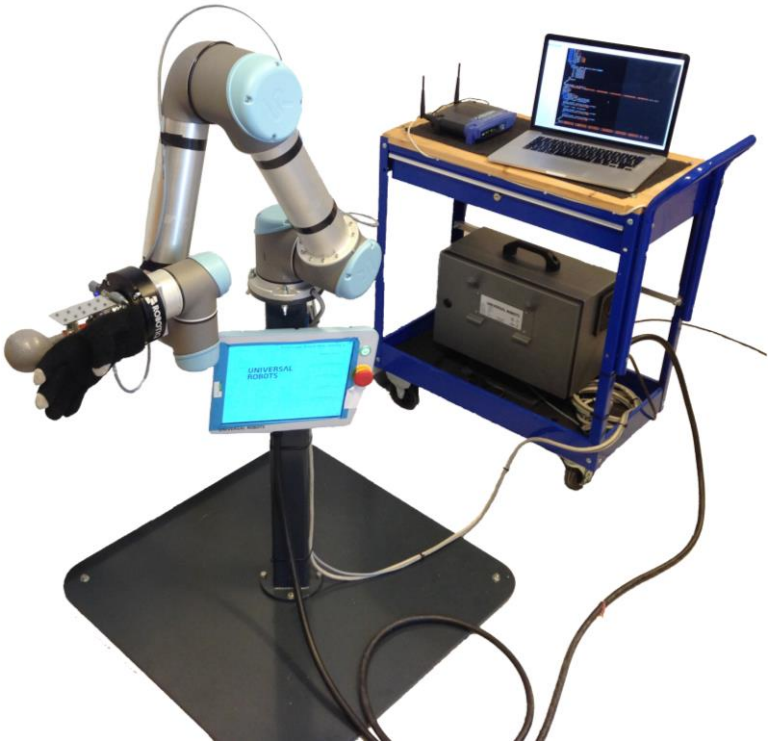
BACHELOROPPGÅVE

Hovedprosjektrapport

Rehabilitering med robotarm

En robotarm i interaksjon med menneske



HO2-300 Bacheloroppgave

Avdeling for ingeniør- og naturfag

23.05.2016

Marius Blom (4)

Børge Godejord (1)

Hans Olav Tuften (19)

HØGSKULEN i SOGN OG FJORDANE

HØGSKULEN i
SOGN OG FJORDANE

Fullmaktsskjema / Avtale

Om innsyn i og elektronisk publisering av bacheloroppgåve eller masteroppgåve.

Namn på forfattar(ar): Marius Blom, Børge Godejord, Hans Olav Tuften

(Mrk: Skriv tydeleg (STORE BOKSTAVAR!.) Kvar student fyller ut eige skjema)

Tittel på oppgåva: Rehabilitering med robotarm

Utgjevingsår: 2016 Kandidatnummer: Marius(4) , Børge (1) , Hans Olav (19)

Namn på studium/Emnekode: Ingeniør elektro - automatiseringsteknikk / HO2-300

Med dette gir ovannemnde student Høgskulen i Sogn og Fjordane (HiSF) retten til å gjere oppgåva tilgjengeleg for ålmenta gjennom å leggje oppgåva i ein database med open access ("Brage") og/eller publisere trykte eksemplar av oppgåva. Brukarane av Brage har retten til fritt å kopiere eller vidareformidle materialet til ikkje-kommersiell bruk.

Studenten beheld opphavsretten til oppgåva si og kan samtidig gjere oppgåva tilgjengeleg på andre måtar.

Studenten garanterer at han er opphavsperson til innlevert materiale og har fullstendig råderett over dette, evt. at han har innhenta samtykke frå dei opphavspersonane han har nytta i oppgåva. Studenten garanterer at han ikkje har kjennskap til eller mistanke om at oppgåva inneheld materiale som kan stride mot gjeldande norsk rett eller innehalde lenkjer eller andre koplingar til slikt materiale.

Dersom HiSF skulle bli gjort erstatningsansvarleg overfor tredjepart fordi studenten ikkje oppfyller garantiar etter denne avtala, er studenten plikta til å halde HiSF fullt ut skadesløs.

Stad Dato

Førde 19.05.2016

Signatur Børge Godejord

Hans Olav Tuften

03.10.12

I Referanseside studentrapport

Boks 523 , 6803 FØRDE. Tlf: 57722500, Faks: 57722501 www.hisf.no

HØGSKULEN i SOGN OG FJORDANE
Avdeling for ingeniør- og naturfag

TITTEL	RAPPORTNR.	DATO
HO2-300 Bacheloroppgave	1	23.05.2016
PROSJEKTTITTEL	**TILGJENGELIG**	**ANTALL SIDER**
Rehabilitering med robotarm	Åpen	34 + 9 vedlegg
FORFATTERE	**ANSVARLIG RETTLEDER**	
Marius Blom, Børge Godejord og Hans Olav Tuften	Joar Sande	
OPPDRAGSGIVER		
Høgskulen i Sogn og Fjordane		

SAMMENDRAG

I Norge blir mange rammet av hjerneslag hvert år, og antallet tilfeller er forventet å øke de neste årene. Disse menneskene trenger opptrening raskt etter slaget og trenger oppfølging i tiden etterpå. En robot kan bistå helsepersonell med rehabiliteringen av pasienter.
På bakgrunn av dette har vi satt mål om å demonstrere at roboten er trygg og nyttig i rehabiliteringen av slagpasienter. Gjennom dialog med oppdragsgiver og helsepersonell har vi funnet oppgaver roboten kan gjøre.
Vi har lagd programmer for å håndhilse med roboten og etterlignet noen vanlige øvelser for slagpasienter. Modellen er ikke et ferdig produkt og av den grunn har vi hatt ekstra fokus på å lage en oversiktlig og strukturert kode, slik at den enklest mulig kan videreutvikles. Resultatet er en modell som kan vise potensialet en robotarm har i rehabilitering av slagpasienter.

SUMMARY

In Norway a lot of people suffer stroke every year, and the amount of cases is expected to rise during the next years. These people need immediate training and monitoring after the incident. A robot can aid health personnel with the rehabilitation of stroke patients.
On this basis we have set the goal to demonstrate that the robot is safe and useful in the rehabilitation process of stroke patients. Through dialog with the assigner and health personnel we have found some tasks that the robot can do.
We have coded programs to shake hands with the robot and imitated some common exercises for stroke patients. The model is not a finished product and for this reason we have focused on structuring the program in such a way that is as easy as possible for continued work. The result is a model that can show the potential of a robotic arm in the rehabilitation of stroke patients.

EMNEORD

HO2-300, bacheloroppgave, HISF, AIN, UR5-robot, rehabilitering, robot, automasjon.

II Forord

Vi er tre studenter som går ingeniørlinjen automatiseringsteknikk ved Høgskulen i Sogn og Fjordane. Dette er hovedprosjektrapport for *Bacheloroppgave HO2-300* som utgjør 20 av 180 studiepoeng på sjette og siste semester av studiet. Prosjektet skal være praktisk og/eller teoretisk. Det skal dokumenteres med prosjektbeskrivelse, forprosjektrapport, pressemelding, plakat, muntlig presentasjon og sluttrapport. Tidlig i prosjektet skal det også opprettes en nettside som skal oppdateres underveis. Den finnes her; http://hisf.heime.nu

Bacheloroppgaven har sammenheng med et prosjekt fra femte semester gjennom at vi brukte samme type robotarm. Slik var vi litt kjent med denne robotarmen fra før. I denne oppgaven har vi jobbet innen velferdsteknologi, som er et aktuelt tema for tiden. Det har vært et svært spennende tema å jobbe med. Størsteparten av prosjektet har bestått av programmering og vi ønsker å rette en stor takk til Jørn Sandvik Nilsson v/Rocketfarm AS og Marcin Fojcik for god hjelp til programmeringsarbeidet. Marcin ga oss også muligheten til å presentere prosjektet vårt på *International Conference on Computer Systems in Medicine and Health* i Polen, noe vi er takknemelige for.

Ønsker å takke Erik Kyrkjebø ved Høgskulen i Sogn og Fjordane for spennende oppgave og engasjerende samarbeid. Vi retter også en takk til Høgskulen i Sogn og Fjordane for lån av robot samt støtte til innkjøp av nødvendig utstyr. Avslutningsvis retter vi en takk til Joar Sande for god hjelp og rettledning gjennom hele prosjektet.

Førde den 23.05.2016

Marius Blom	Børge Godejord	Hans Olav Tuften

III Innholdsfortegnelse

I Referanseside studentrapport I
II Forord II
III Innholdsfortegnelse III
IV Tabeller IV
V Figurer IV
VI Sammendrag VI
1.0 Innledning 1
1.1 Definisjoner 1
1.2 Bakgrunn 3
1.3 Mål 4
2.0 UR5-robot til rehabilitering 5
2.1 Bakgrunnskunnskap og klargjøring av problemstilling 5
2.2 Mulige løsninger og teori 6
2.2.1 Potensielle rehabiliteringsøvelser 6
2.2.2 Teori om modellkomponenter 7
2.2.3 Teori om programmeringsspråk 8
2.3 Valg av løsning 8
2.3.1 Modell 8
2.3.2 Programmering 10
2.4 Diskusjon og konklusjon 13
2.4.1 Resultat 13
2.4.2 Utfordringer 14
2.4.3 Forslag til videreføring av prosjektet 16
2.4.4 Konklusjon 17
3.0 Prosjektadministrasjon 18
3.1 Personer involvert i prosjektet 18
3.1.1 Organisering 18
3.1.2 Ressurspersoner 19
3.2 Gjennomføring i forhold til plan 19

3.3 Ressursforbruk 20
3.3.1 Tid 20
3.3.2 Kostnader 21
3.4 Programvare og dokumentbehandling 22
3.4.1 Programvare 22
3.4.2 Dokumentbehandling 24
3.5 Generell prosjektevaluering 24
Litteraturliste 25
Bildereferanser 26
Tabellreferanser 28
Vedlegg 28

IV Tabeller

Tabell 1: Utklipp av teknisk data for FT 150. 8
Tabell 2: Tidsfrister for bacheloroppgaver 2016. 20
Tabell 3: Viser de planlagte dagene, tom rute er disponibel til prosjektarbeid. 21
Tabell 4: Budsjett og resultat. 22

V Figurer

Figur 1: Tegning som framstiller robotarmen og viser aksene. 1
Figur 2: Hvor blodproppen eller blødningen er gir forskjellig skadeutfall. 2
Figur 3: Skadeutfallet ved slag. 4
Figur 4: Pidestallen før robotarmen ble festet. 9
Figur 5: Verktøyet vi lagde og brukte i øvelsene. 10
Figur 6: Utklipp av koden som oppretter kobling mellom Java og robotarmen. 11
Figur 7: Biblotek-klassen inneholder URScript. Script er lagret som String-verdier. 11
Figur 8: Oversikt over kommunikasjon mellom klassene i Java, roboten og sensoren. 12
Figur 9: Det grafiske grensesnittet laget i JavaFX. På bildet ligger musepekeren over BallRehab. 14
Figur 10: Metoden som sender koden til roboten. 15
Figur 11: Det fungerende programmet. 15

Figur 12: Sammenligning av den leste datastrømmen fra roboten. 16
Figur 13: Utklipp av koden i threaden i URScript. 16
Figur 14: Organisering av bachelorprosjektet. 18
Figur 15: Logo, Ubuntu. 22
Figur 16: Logo, Qt. 23
Figur 17: Logo, NetBeans. 23
Figur 18: Logo, Wordpress. 23
Figur 19: Logo, AutoCAD. 23
Figur 20: Logo, Adobe Photoshop. 23
Figur 21: Logo, iMovie. 23
Figur 22: Logo, Dropbox. 24
Figur 23: Logo, Google dokumenter. 24

VI Sammendrag

I Norge blir omtrent 15 000 mennesker rammet av hjerneslag årlig, dette tallet er forventet å øke med 50% i løpet av de neste 20 årene. Disse menneskene trenger opptrening raskt og oppfølging i tiden etter slaget. En robotarm vil kunne bistå faglærte innen helsetjenesten med rehabilitering av pasienter.

På bakgrunn av dette har vi satt mål om å demonstrere at robotarmen er trygg, og vise at den kan være nyttig i rehabiliteringen av slagpasienter. I denne oppgaven har vi fokusert på rehabiliteringsøvelser for opptrening av en arm. Gjennom samtaler med oppdragsgiver og helsepersonell har vi funnet hva som må gjøres for å nå disse målene.

Vi har lagd et program som håndhilser for å la en person ta på roboten og føle hvordan den beveger seg. I tillegg har vi etterlignet noen vanlige øvelser for å vise at roboten har en nytteverdi i rehabiliteringsprosessen. Modellen er ikke et ferdig produkt og av den grunn har vi hatt ekstra fokus på å lage en oversiktlig og strukturert kode, slik at den enklest mulig kan videreutvikles.

Resultatet vi står igjen med er en robot som kan vise potensialet for å bruke en robot i rehabiliteringen av slagpasienter. Arbeidsgruppen ser nytten av denne roboten i rehabiliteringen, og mener dette er noe som burde videreutvikles.

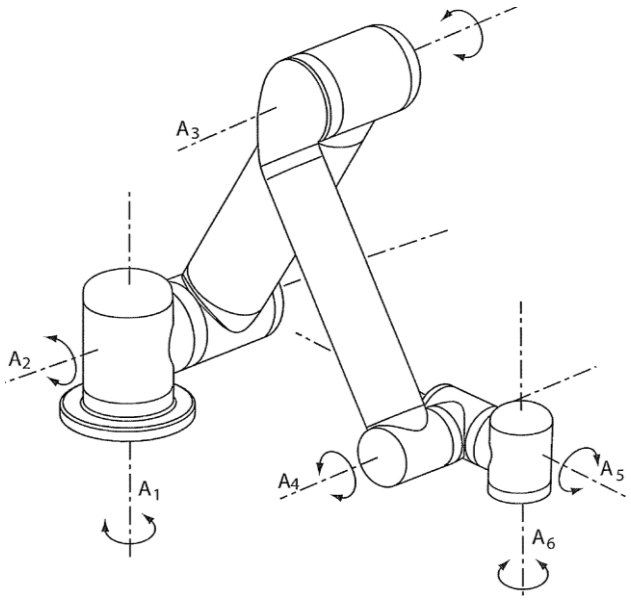
1.0 Innledning

Bacheloroppgaven har opphav i et planlagt forskningsprosjekt ved Erik Kyrkjebø, som skal omhandle bruk av en robotarm i rehabilitering av hjerneslagpasienter. Ved hjelp av en robotarm skal vi i denne bacheloroppgaven demonstrere hvordan en robot kan arbeide i kontakt med et menneske. I tillegg vil vi se på potensialet en robotarm har for å bidra i rehabilitering av pasienter som er rammet av hjerneslag.

Denne prosjektrapporten kan deles opp i to deler. Første del av rapporten omhandler det tekniske rundt oppgaven. Her presenteres det relevante automasjon- og helsefaglige stoffet. Den andre delen omhandler prosjektadministrasjon, arbeidsprosessen og teori rundt prosjektgjennomføring, dette finnes fra og med kapittel tre og utover.

1.1 Definisjoner

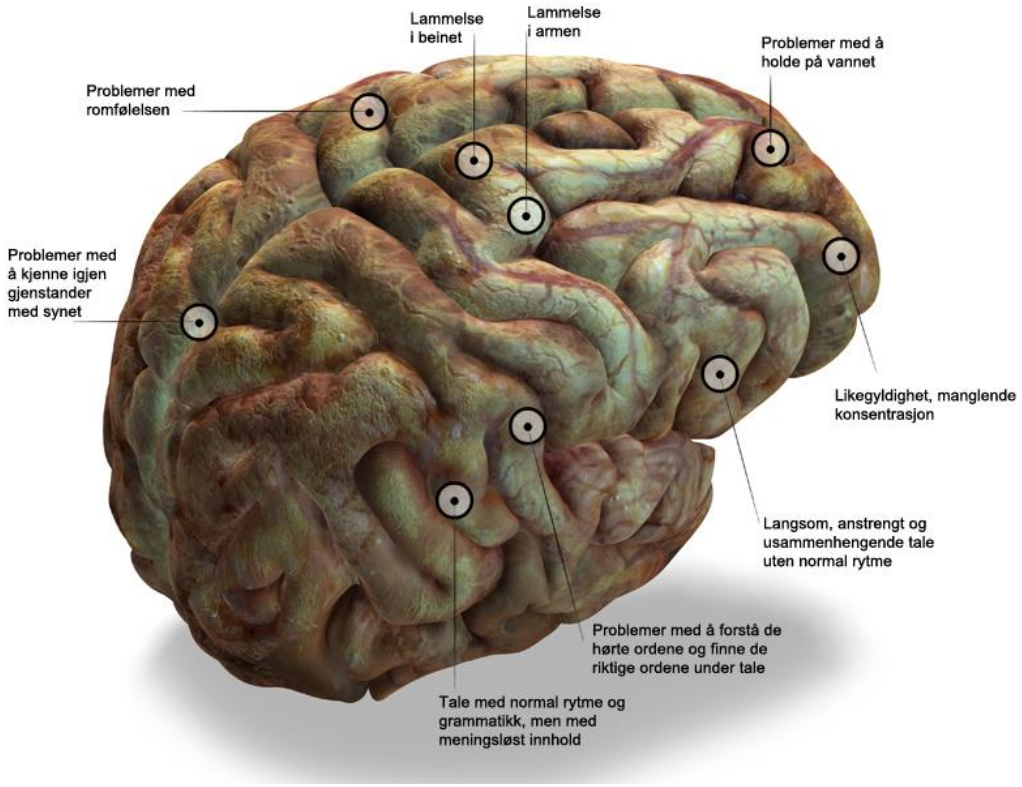
De to mest sentrale uttrykkene for dette prosjektet er robotarm og hjerneslag. En robotarm er en robot som styrer en mekanisk arm. Store Norske Leksikon definerer en robot på følgene måte: *«Robot, en datastyrt enhet som ved hjelp av sensorer kan motta data fra omgivelsene, bearbeide disse og reagere ved å iverksette handlinger i henhold til forhåndsprogrammerte regler* [1]*.»* Robotarmen vi har brukt er illustrert i figur 1, den har seks bevegelige akser.



Figur 1: Tegning som framstiller robotarmen og viser aksene.

Store Medisinske Leksikon definerer hjerneslag slik: *«Hjerneslag, fellesbetegnelse på sykdomstilstander som skyldes en plutselig forstyrrelse av blodsirkulasjonen i hjernen. Dette medfører vanligvis akutte symptomer i form av f.eks. lammelser eller andre uttrykk for en endret funksjon i større eller mindre deler av hjernen. De symptomene pasienten har, kan fortelle om hvor i hjernen skaden har oppstått* [2].*»*

Hjerneslag forekommer som oftest på to forskjellig måter. I 85-90% av tilfellene skyldes hjerneslaget en blodpropp i hjernen. Den andre og mer sjeldne årsaken er hjerneblødning, som oppstår når en blodåre i hjernen brister og en blodansamling oppstår [3]. På figur 2 ser vi eksempler på skadeutfall i forhold til hvor i hjernen blodsirkulasjonen får forstyrrelser.



Figur 2: Hvor blodproppen eller blødningen er gir forskjellig skadeutfall.

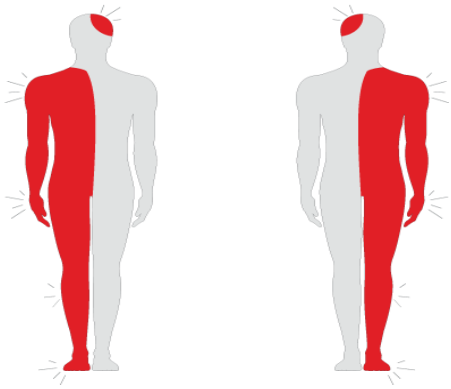
Av skadeutfall etter et hjerneslag er lammelser av forskjellig grad og type det mest utbredte. Vi går mer inn på hjerneslag og skadeutfall i delkapittel 1.2.

1.2 Bakgrunn

Bakgrunnen for denne oppgaven var at alle tre i arbeidsgruppen ønsket å jobbe med robotarmen vi har brukt i oppgaven. Femte semester jobbet vi med samme robotarm i en prosjektoppgave, og fikk her med oss mye nyttig erfaring. Gjennom faglærer Joar Sande ble vi satt i kontakt med Erik og fikk utformet en oppgave til bachelorprosjektet.

Bacheloroppgaven kan sees på som et forprosjekt til det planlagte forskningsprosjektet. I oppgaven ser vi på rehabiliteringen av hjerneslagpasienter og bruk av robotarmer i direkte kontakt med mennesker. Hjerneslag er et økende problem, og årlig rammes om lag 15 000 personer i Norge [4]. I følge flere studier vil dette antallet ha økt med 50% innen de neste 20 årene [5] [6]. I et intervju utført av NRK belyses det at hjerneslag er en økonomisk kostbar sykdom for samfunnet. *«Hjerneslag er en av de aller dyreste sykdommene vi har, og koster samfunnet enorme summer. Foruten å være den vanligste årsak til invaliditet og uførhet, er det også den nest hyppigste årsak til innleggelse på sykehjem, sier overlege og leder av Norsk nevrologisk forening, Anne Hege Aamodt* [4].*»* Av dette forstår man at kapasitetsbehovet også gjør seg gjeldene. Denne kapasiteten og kostnaden kan trolig forbedres om roboter kan bidra i rehabiliteringsarbeidet av hjerneslagpasienter. En robotarm kan ha et potensiale som et hjelpemiddel for en rehabiliteringsavdeling ved et sykehus eller en annen rehabiliterings- eller treningsinstitusjon. Vi er vant til at roboter blir brukt i en industrisammenheng, der de fysisk avskjermes for mennesker. Av denne grunn vil det komme inn et aspekt omkring det å stole på en robotarm som et hjelpemiddel i direkte kontakt med personer. Helsepersonell og pasient må ha full tillit til roboten, noe som igjen gjør sikkerhet svært viktig. Robotarmen vi har brukt er av typen UR5 fra produsenten Universal Robots som er tilgjengelig for ingeniørstudentene ved Høgskulen i Sogn og Fjordane avdeling Førde. Denne roboten er godt egnet siden den har innebygde sikkerhetsfunksjoner, noe vi kommer tilbake til i delkapittel 2.2.2.

Av de som rammes av et hjerneslag forekommer lammelser i over 80% av tilfellene, og er dermed et av de hyppigste funksjonsutfall ved hjerneslag. Lammelser ved hjerneslag er ofte halvsidige, som vil si at lammelsen skjer på den ene siden av kroppen. Hjernen er todelt og organisert slik at høyre side av hjernen styrer venstre side av kroppen, og motsatt [7]. Får man for eksempel en blodpropp i høyre hjernehalvdel så vil det kunne resultere i lammelser på venstre side av kroppen slik figur 3 viser.



Figur 3: Skadeutfallet ved slag.

De mest fremtredende motoriske funksjonsnedsettelsene er redusert kraft, nedsatt tempo, svekket motorisk kontroll, økt trettbarhet og ukoordinerte bevegelser. I opptreningen etter et slag er følgende faktorer viktige for motorisk læring: Oppgaverelatert trening, høy intensitet og tilstrekkelig mengde [8]. Dette er faktorer en robotarm har svært stor kapasitet på. I denne oppgaven fokuserer vi på øvelser for opptrening av en hånd. Mellom 55% til 75% av slagrammede har vedvarende lammelser i en arm, og mange studier viser at oppgaverelatert trening har god effekt på den rammede armen [8].

Bacheloroppgaven hadde ikke som hensikt å lage et sluttprodukt eller noe som ville vært klart for bruk til rehabilitering av hjerneslagpasienter. Vi har lagt ned mye arbeid i oppbygging av program som kobler til og styrer roboten, slik at det er godt tilrettelagt for videre arbeid og utvidelser. Det er denne programmeringen som utgjør størsteparten av bacheloroppgaven. I neste delkapittel presenteres de konkrete målsetningene vi hadde for prosjektet.

1.3 Mål

Effektmålet for prosjektet er å demonstrere for personer som skal bruke roboten, enten som pasient eller helsepersonell, at UR5 er trygg. Vi skal også demonstrere at UR5 kan være nyttig i rehabiliteringen av slagpasienter. Resultatmålet er å fremstille en modell som kan brukes for å demonstrere at roboten er trygg i interaksjon med mennesker. Denne demonstrasjonen skal kunne gi et innblikk i mulighetene for bruk av roboter i velferdssektoren.

For å nå disse målene har vi satt opp en liste med delmål, samt hva som kreves av prosjektet, i Gantt-skjemaet (vedlegg 1) og arbeidsplan (vedlegg 2). Hovedpunktene er her planlegging, innhenting av informasjon, programmering og montering av modell, samt formidling i form

av rapporter og presentasjon. Sentrale og viktige delmål er anskaffelse av ny pidestall til roboten, og at grunnelementene i programmet som styrer roboten skal skrives slik at det lett kan utvides senere.

Underveis i prosjektet ble det etterhvert et klart mål å legge til rette for videre bruk av roboten. Tanken er å legge til rette for undervisning eller videre arbeid i form av prosjekt, laboppgaver eller lignende. Målsetningen her er å formidle all programkode vi har utarbeidet slik at den kan bygges videre på. Vi skal lage en oppskrift som kan følges for å komme i gang med programmering og bruk av UR5-roboten.

2.0 UR5-robot til rehabilitering

Denne delen av rapporten omhandler mål for oppgaven, mulige løsninger og hvilke løsninger vi har valgt. På slutten diskuterer vi resultatet av bacheloroppgaven og foreslår videre arbeid.

2.1 Bakgrunnskunnskap og klargjøring av problemstilling

Som nevnt innledningsvis hadde vi erfaring i bruk av robotarmen. Under prosjektet femte semester programmerte vi roboten på en enkel måte ved hjelp av en berøringsskjerm. På denne måten fikk vi forståelse for hvordan roboten fungerer.

På bakgrunn av det vi har sagt hittil har problemstillingen for bacheloroppgaven vært:

1. Hvordan vil en UR5-robot kunne bidra med rehabilitering av slagpasienter?
2. Hvilke faktorer vil ha betydning for å få til denne rehabiliteringen med UR5-roboten?

En viktig faktor var å finne ut om en UR5-robot kunne arbeide i kontakt med et menneske på en trygg måte. Hvis en fysioterapeut skal bruke robotarmen i rehabilitering må fysioterapeuten ha full tillit til robotarmen. En annen faktor er om roboten har potensiale til å utføre konkrete øvelser som er relevante i rehabiliteringen av slagpasienter.

Som tidligere nevnt så er lammelser ofte halvsidige, slik at en lammelse vil nesten i alle tilfeller ramme kun en hånd om gangen. Vi fokuserte på øvelser som skulle trene kun en hånd eller arm om gangen. Dette tok vi utgangspunkt i under prosjektet og da vi skulle demonstrere rehabiliterings øvelser.

2.2 Mulige løsninger og teori

Underveis i oppgaven har vi diskutert mye rundt hvilke øvelser vi skal programmere roboten til å kjøre. Dette utdypes her etterfulgt av innledende teori omkring programmeringsspråk og modellen.

2.2.1 Potensielle rehabiliteringsøvelser

For å demonstrere at man kan bruke roboter i rehabilitering skulle vi vise at den klarer å respondere på fysisk berøring. Dette ville vi demonstrere ved å lage et program som håndhilser når robotarmen merker at noen tar i den.

Utvidelser av prosjektet ville i hovedsak bestå av å lage flere programmer som armen kan kjøre, eller utvide programmene med flere funksjoner. Da tenkte vi å etterligne noen vanlige øvelser som blir brukt i rehabilitering av slagpasienter. En øvelse som viste seg å passe roboten bra var en øye til hånd koordinasjonsøvelse, der helsepersonell holder hånden opp og venter på at pasienten tar på hånden. Helsepersonellet flytter på hånden og øvelsen fortsetter.

Noen andre øvelser som vi mente kunne fungere for en robotarm, var å føre ringer ned på pinner i ulik høyde, føre en gjenstand i en bane i rommet og holde på en ball som blir forsiktig ført vekk fra pasienten slik at han mister taket. Øvelsen som fører pasienten langs en bane tenkte vi at vi kunne utvide til å involvere en tavle og en tusj. Dette programmet vil at pasienten skal følge et mønster og avhengig av hvor frisk pasienten er kan han få hjelp eller motstand fra robotarmen til å tegne over mønsteret.

En fordel vi ser ved å bruke en robotarm i stedet for andre fysiske hjelpemiddel er at man kan lagre hvilke programmer som ble kjørt, og dermed lagre progresjonen til pasienten. Dette kan bidra til en mestringsfølelse for pasienten og gi motivasjon i treningsarbeidet.

Siden vi skulle lage en modell som skulle komme i kontakt med mennesker var vi nødt til å ha tilstrekkelig med sikkerhetstiltak. Sikkerhetstiltakene vi så muligheten for var å programmere en kraftsensor til å detektere når den møter en uventet motstand. For de dataprogrammene som er lagd for å føre en pasient langs en bane vurderte vi å feste pasienten til robotarmen på en måte som «ryker» når den møter motstand, for eksempel ved å bruke en borrelås. En annen mulighet var å la pasienten holde seg fast i en anretning festet til robotarmen. Dette ville

begrense hvor svak pasienten kunne være for å fortsatt kunne bruke roboten. Vi tenkte at den tryggeste løsningen var å la pasienten holde seg fast, slik at han kan slippe taket, og at en uventet feil kan stoppes av en sensor.

2.2.2 Teori om modellkomponenter

Modellen består av robotarm med fundament, kraftsensor og verktøy montert på robotarmen. Teori omkring disse komponentene presenteres her for å gi et nødvendig teoretisk grunnlag for denne oppgaven.

UR5-robot fra Universal Robots er en masseprodusert robotarm med seks rotasjonsledd beregnet på industri, som oppdragsgiver ønsket at vi skulle bruke i prosjektet. Robotarmen er kraftbegrenset, som i hovedsak betyr at robotarmen registrerer om den føler en unormal opplevelse av kraft via banen den kjører. Strømmen i aktuatoren blir brukt til å overvåke kraften som blir registrert i de forskjellige leddene på robotarmen [9]. Det medfører at om robotarmen kommer borti noe med en kraft på 150 Newton, som tilsvarer omtrent 15kg, vil den stoppe opp automatisk. Dette gjør at den er tryggere enn en tradisjonell robotarm, som bare vil kjøre videre om den treffer noe. I praksis betyr dette at omtrent 80% av bedriftene som bruker roboter fra Universal Robots i sin produksjon ikke trenger å operere med fysiske sikkerhetssperringer [10]. UR5, som vi bruker, vil ikke kunne registrere at den støter på noe om kraften den treffer med er under 50 Newton, da denne er beregnet på å kunne løfte 5kg. I hovedsak betyr dette at sikkerhetsverdien for hvor mye kraft den må treffe med før den stopper automatisk ligger mellom 50 Newton og 150 Newton [11]. I tillegg har denne allerede en nødstopp koblet inn, som vi vil ha nytte av i vårt prosjekt.

Oppdragsgiver fant en kraftsensor han ville vi skulle bruke. Denne kraftsensoren er av type FT150 fra Robotiq og er godt egnet til å brukes sammen med Universal Robots sine roboter. Kraftsensoren måler kraft i seks forskjellige akser og er immun mot utvendig elektrisk støy [12]. Måleområdet er vist i tabell 1.

Tabell 1: Utklipp av teknisk data for FT 150.

MEASURING RANGE: FX, FY, FZ MX, MY, MZ	±150 N ±15 Nm
OUTSIDE DIAMETER	120 mm
DATA OUTPUT RATE	100 Hz
SIGNAL NOISE FX, FY, FZ (COMBINED)	0.5 N
SIGNAL NOISE FX, FY	
SIGNAL NOISE FZ	
SIGNAL NOISE MX, MY, MZ (COMBINED)	0.03 N·m
SIGNAL NOISE MX, MY	
SIGNAL NOISE MZ	
EXTERNAL NOISE SENSITIVITY	All axes immune

2.2.3 Teori om programmeringsspråk

Vi hadde flere muligheter til å programmere roboten. Programmeringen kunne gjøres gjennom Polyscope, C++ og Python eller Java og nettverkssocket. Polyscope gir en enkel måte å programmere roboten på, men det ville ikke gi oss gode muligheter for utvidelser av programmene [13]. C++ og Python ville gi oss en bedre mulighet for å utvide programmet ved et senere tidspunkt, siden vi kan lage kjernen i programmet i C++, for så å utvide det senere. Det siste alternativet for oss var å programmere det via nettverkssocket og Java. Fordelen med Java er at det fungerer på alle plattformer. Videre måtte vi se på begrensningene ved å programmere den via Java og nettverkssocket.

2.3 Valg av løsning

Etterhvert som vi fikk bedre forståelse for både programmering av roboten og rehabiliteringsaspektet begynte vi å se på hvilke øvelser som vi ønsket å gå videre med. Vi valgte løsninger for modell og programmeringsspråk som la til rette for å utarbeide rehabiliteringsøvelsene.

2.3.1 Modell

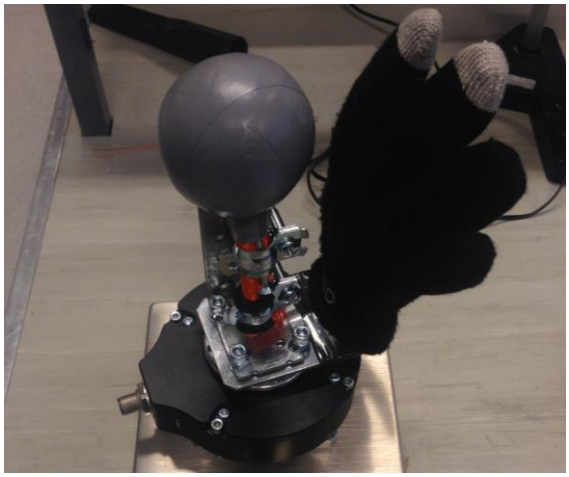
Installasjon av kraftsensoren som oppdragsgiver ønsket å bruke skjer ved å installere programvare fra en minnepenn. Vi fikk problemer med denne installasjonen og det viste seg at software på roboten var utdatert og derfor ikke var kompatibel med sensoren. I følge

brukermanualen (vedlegg 5) var software-versjonen som kraftsensoren var testet på versjon 1.8.14035. Vi valgte å oppdatere roboten til denne versjonen for å unngå framtidige konflikter.

Et problem med robotarmen ved Høgskulen i Sogn og Fjordane var at den stod på et ustødig bord. Om roboten skulle utføre presise bevegelser eller gjøre presise kraftmålinger, måtte vi skaffe den et solid fundament å stå på. Alternativene vi hadde var å konstruere et fundament selv og få det produsert i nærområdet, kjøpe komplett, eller kombinere innkjøp av ferdige og egenkonstruerte deler. Under denne prosessen lagde vi tegninger med AutoCAD for å konstruere en pidestall til roboten. Vi innhentet tilbud på kjøp av ferdig pidestall, og for produksjon ut av våre tegninger. Her ble konklusjonen at det beste var å kjøpe komplett pidestall fra en profesjonell aktør. Dette på det grunnlag at det ikke var så stor differanse i pris, samtidig viste datasimulerte strekktester at vår egenkonstruerte pidestall ikke var god nok. Pidestallen ble levert 30. mars og den oppfylte alle forventninger. Den er veldig stødig, tar lite plass og kan flyttes ved behov.

Figur 4: Pidestallen før robotarmen ble festet.

For å feste ulike verktøy til roboten har vi brukt jernbeslag og skruer. Hånden vi bruker er en hagehanske fylt med akryl. Vi valgte denne løsningen for å få et mykere håndtrykk enn en mannekenghånd i hardplast ville gitt. Ballen er en tennisball som vi har teipet rundt og festa en strikk til, slik at strikken regulerer hvor mye kraft som virker på ballen i drakampen. Strikken er festet gjennom to rørklammer. Rørklammene ble først brukt til å feste en tusj til, så vi kunne tegne med roboten.



Figur 5: Verktøyet vi lagde og brukte i øvelsene.

2.3.2 Programmering

Vi prøvde først å programmere roboten i Polyscope, men fant ut at det ikke ville gi oss gode muligheter for å utvide programmene senere, noe som var en viktig faktor for arbeidsgiver. Siden skolen hadde en pc som kjørte Linux, og som allerede var satt opp med forbindelse til roboten, prøvde vi litt programmering i C++ og Python, men etter noen enkle programmer fant vi ut at det ville bli for tidskrevende å lære nok C++ til å kunne programmere roboten. På grunn av disse begrensingene valgte vi å programmere UR5-roboten gjennom Java, der kommunikasjonen foregår via TCP/IP. Skolen har allerede både trådløst og kablet nettverk, men vi har ikke kontroll selv på hvordan trafikken blir håndtert. Roboten trenger kablet nettverk, mens datamaskinene bruker trådløst nettverk. Det er ikke kobling mellom kablet og trådløst nettverk på skolen. Vi valgte derfor å løse dette ved å opprette et eget privat nettverk, slik at vi kunne ha bedre kontroll på adresser og porter som vi ønsker å bruke. For å komme videre med programmeringen kontaktet vi Rocketfarm i Sogndal, som har erfaring med UR-roboter. De viste seg å være veldig hjelpsomme og hadde god erfaring med å bruke Java til å styre UR-roboter. På dette tidspunktet gikk vi bort fra C++ og Linux.

I Java har vi satt opp kommunikasjon via socket, der vi sender URScript til port 30002 på UR5-roboten. Port 30002 på roboten er dedikert til å motta URScript, i tillegg så sendes det ut data fra roboten, som for eksempel: Leddposisjoner, temperaturer, tilstand og lignende [14]. Det blir sendt data fra roboten kontinuerlig så vi lagde en klasse i Java for å oversette dataene vi mottok. På denne måten fikk vi ut posisjonen til UR som vi lagret som en String-verdi i Java.

```
/*
Method to connect to the UR
 */
public void connectUR() {
    try {
        /*
        Establish socket connection to the UR
         */
        System.out.println("Connecting to " + serverName + " on port " + port);
        client = new Socket(serverName, port);
        client.setSoTimeout(5000);
        System.out.println("Connected to " + client.getRemoteSocketAddress());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

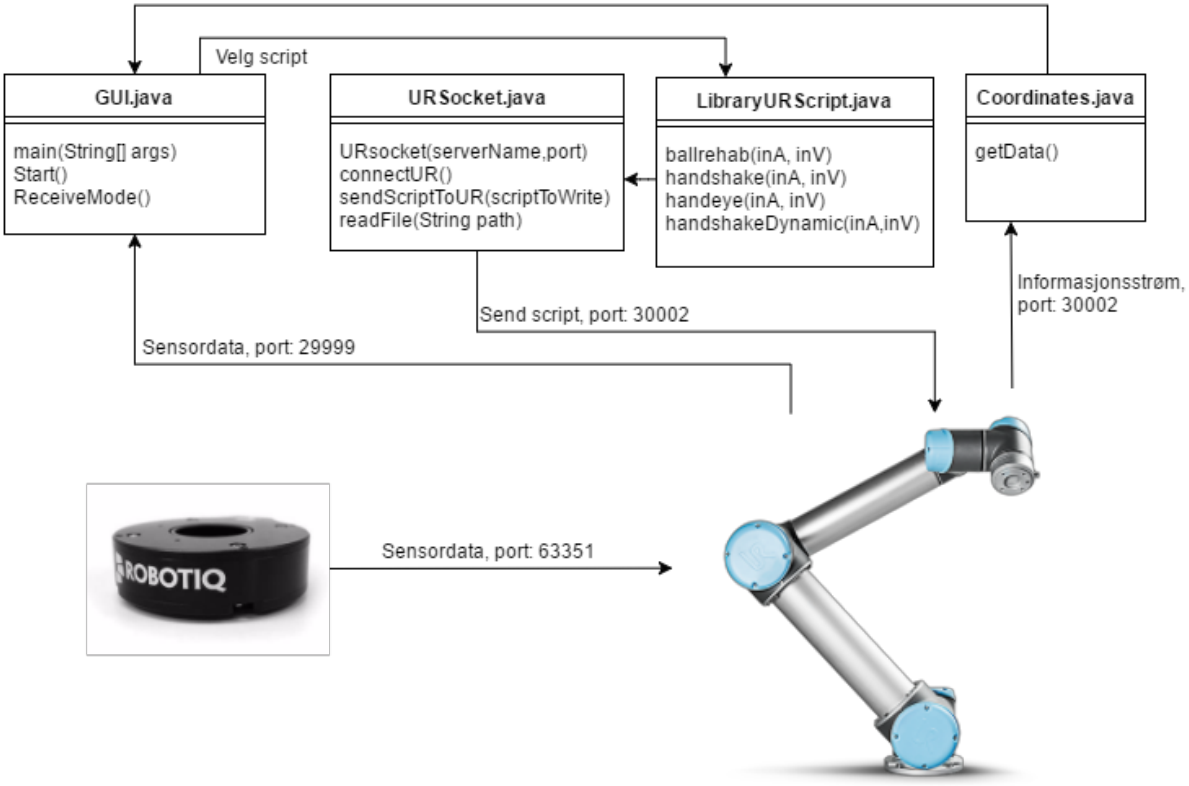
Figur 6: Utklipp av koden som oppretter kobling mellom Java og robotarmen.

Når det gjelder å sende URScript til roboten, har vi flere muligheter for å gjøre dette. Her kan vi skrive og lagre scriptet som en separat fil, for så å lese fra scriptet i Java, som igjen sender det til roboten. Et annet alternativ var å oversette URScript til en verdi som Java forstår, på denne måten kunne vi lagre URScriptet som en String-verdi i Java. Vi valgte å lagre script som String.

```
public class LibraryURScript {

    String payloadUR = "1.985"; // in kilograms ( 1.985kg)
    String serverPC;
    int portPC;

    public LibraryURScript(String serverPC, int portPC) {...4 lines }

    public String threadSensorData() {...58 lines }

    public String initialization() {...24 lines }

    public String ballrehab(String inA, String inV) {...112 lines }

    public String handshake(String inA, String inV) {...80 lines }

    public String handeye(String inA, String inV) {...97 lines }

    public String handshakeDynamic(String inA, String inV) {...95 lines }
}
```

Figur 7: Biblotek-klassen inneholder URScript. Script er lagret som String-verdier.

Java programmet vårt er i hovedsak bygd opp med fire forskjellige hovedklasser. Vi bruker klassen GUI.java for å samle og kjøre de andre klassene, her ligger også det grafiske brukergrensesnittet. URsocket.java oppretter en sockettilkobling til roboten, og via konstruktøren blir det sendt hvilken adresse og port den skal koble seg til for å kommunisere med roboten. Coordinates.java er i hovedsak kun for å oversette posisjonsdata som vi leser av fra port 30002. LibraryURScript.java er et bibliotek vi lagde for å samle alle URScriptene i en klasse. Med denne oppbyggingen kan vi utvide programmet vårt mer fritt, siden vi kan kombinere flere URScript ved hjelp av disse klassene og metodene. Se vedlegg 6 til 9 for koden.



Figur 8: Oversikt over kommunikasjon mellom klassene i Java, roboten og sensoren.

2.4 Diskusjon og konklusjon

I dette kapittelet viser vi de konkrete resultatene av prosjektet, samt foreslår videre arbeid for prosjektet til Kyrkjebø eller bachelorstudenter.

2.4.1 Resultat

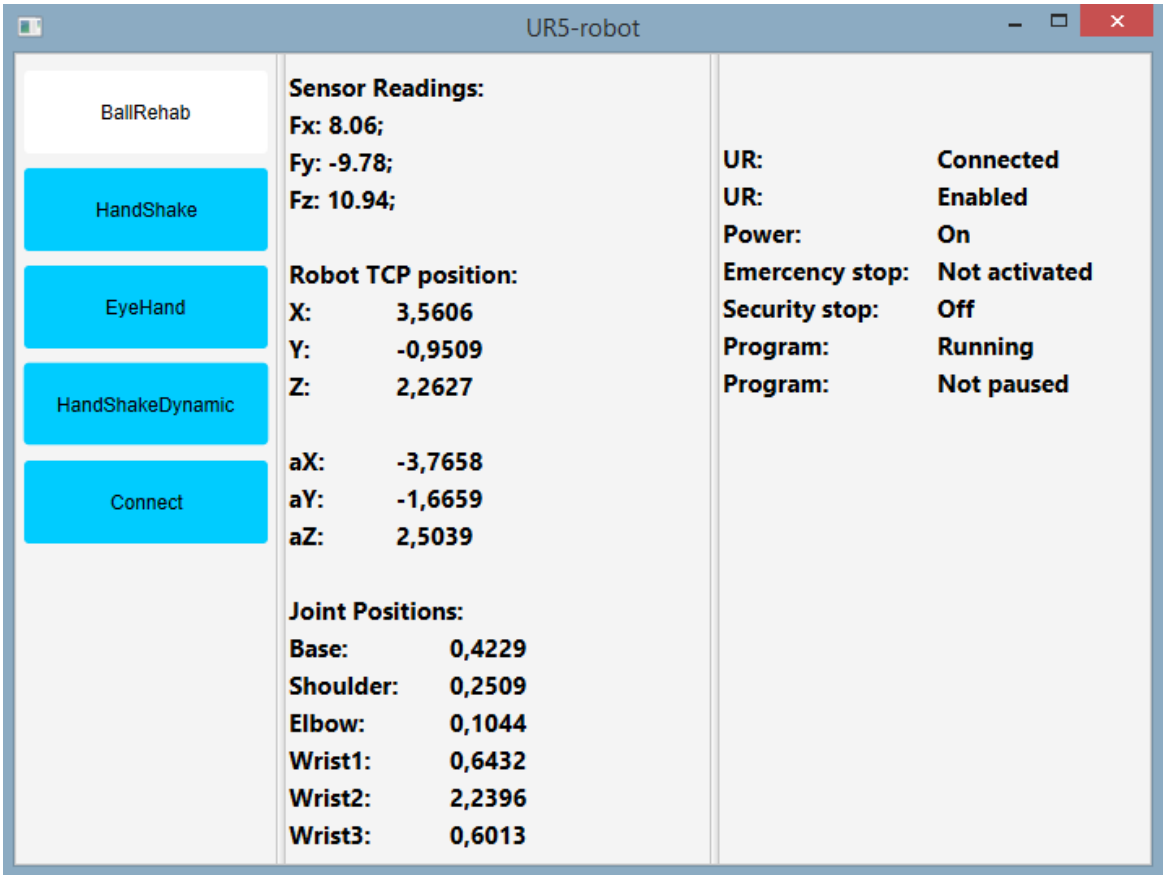
Resultatet av prosjektet er at vi har lagd fire programmer for å demonstrere ulike måter å bruke roboten i rehabilitering. Disse programmene består av to håndhilseprogram, et øye-håndkoordinasjonsprogam og et program for å trene fingerstyrke.

Det første håndhilseprogrammet står i klar-modus og venter på at noen skal starte håndhilsningen. I dette programmet blir sensoren brukt som en startkapp der den må merke en kraft oppover for å starte. Deretter etterligner programmet en vanlig håndhilsning og trekker hånda tilbake etter den er ferdig. Det andre håndehilseprogrammet er mer dynamisk, og her styres bevegelsen til robotarmen via sensoren, der robotarmen følger den retningen den føler kraft. Formålet med programmene var å lære å programmere roboten og sensoren med URScript, men det er også korte og enkle programmer som kan la flere personer føle hvordan roboten beveger seg.

Øye-håndkoordinasjonsprogrammet beveger roboten til en posisjon, og venter på at brukeren skal dytte på sensoren. Dette gjentas til roboten har gått gjennom alle posisjonene. Programmet er lagd med tanke på en bruker som klarer å løfte armen, men må trene øye-håndkoordinasjon. Programmet stiller i tillegg krav til det som kalles *postural-kontroll*, som er sittebalansen. Ved en slik øvelse kan fysioterapeuten stå ved pasienten å observere og veilede pasienten underveis.

Det siste programmet er lagd for å trene arm og fingre. Programmet gjør at roboten kontinuerlig leser data fra sensor, og drar en ball vekk fra pasienten. Roboten vil justere kraften den drar med basert på hvor mye kraft en pasient drar med, og til slutt vil pasienten miste taket på ballen.

For disse programmene har vi lagd et grafisk grensesnitt der man kan velge hvilket program man ønsker å starte, og mens programmet kjører vil datamaskinen vise oppdatert informasjon fra sensoren og roboten.



Figur 9: Det grafiske grensesnittet laget i JavaFX. På bildet ligger musepekeren over BallRehab.

2.4.2 Utfordringer

Programmering har vært den største utfordringen, spesielt med tanke på at teori om roboter ikke har vært en stor del av undervisningen.

En av de første utfordringene vi møtte på var kommunikasjon med roboten. På nettsiden til leverandøren fant vi ut at robotene fra Universal Robots kommuniserer via port 30002. I starten brukte vi NetCat, som er en innebygd funksjon i OS X/Unix til å sende URScript. Dette godt nok for å teste om et script vil fungere, men det er lite brukervennlig og utvidingsmulighetene ble begrenset. Med hjelp fra Rocketfarm fikk vi til å sende URScript gjennom socket ved bruk av Java.

```
public void sendScriptToUR(String scriptToWrite) {
    /*
        Display what we are sending to the server
     */
    System.out.println(scriptToWrite);

    try {
        /*
        Create stream to read from file and send data to UR
         */
        outToServer = client.getOutputStream();
        out = new DataOutputStream(outToServer);
        /*
        Write script to server
         */
        out.write(scriptToWrite.getBytes());
        out.write('\n'); // Always newline after a script
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

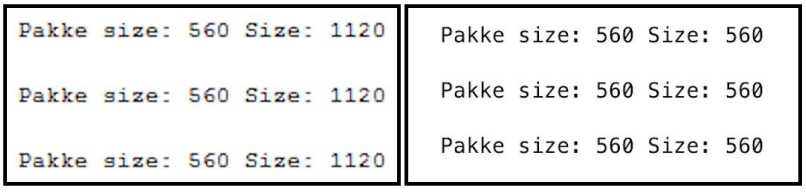
Figur 10: Metoden som sender koden til roboten.

URScript kan minne litt om Python. Et av problemene vi støtte på var at vi måtte avslutte scriptet vårt med linjeskift. Dette var vanskelig å legge merke til, som resulterte i at vi brukte mye tid på å få til et script som bare inneholdt tre linjer.

```
1  # testmove
2  def testmove():
3      movej(p[0, 0.5, 0.5, 3.2415, 0, 0], 0.1, 0.5)
4  end
5
```

Figur 11: Det fungerende programmet.

Som nevnt tidligere i rapporten blir det sendt data fra roboten kontinuerlig, og av den grunn valgte vi å lage en klasse i Java for å oversette dataene vi mottok. Her møtte vi på en utfordring med å tolke data som ble mottatt fra roboten. Siden OS X leser line feed ("\n") som linjeskift, mens Windows leser carriage return og line feed ("\r\n") som linjeskift [15], førte det til at samme programmet vi lagde for å oversette data fra UR gav forskjellig resultat ut i fra hvilken pc det vart kjørt på. Vi lagde en byte-array for sammenligning av data, slik at data ble lest på samme måte i OS X og Windows.



Figur 12: Sammenligning av den leste datastrømmen fra roboten.

Å lese data fra kraftsensoren ved hjelp av Java viste seg å være en utfordring. Det ble vanskelig for Java å skille mellom data fra sensoren og den kontinuerlige datastrømmen siden dette ble sendt samtidig. For å forsikre oss om at det var sensordata vi mottok, så valgte vi å lese av sensordata i en separat tråd fra port 29999 på roboten.

```
while (var_1 ==  False  ):\n"// new
  global var_1 = socket_open(\"" + serverPC + "\"," + portPC + ",\"pc_connect\")\n" // new
end\n"// new
socket_send_string(\"StartOfInput\", \"pc_connect\")\n"
socket_send_string(\";\", \"pc_connect\")\n"
while True:\n" // new
        global sensor_data = socket_read_ascii_float(6,\"stream\")\n"
        varmsg(\"sensor_data\",sensor_data)\n"
        if (sensor_data[0]  >=6): #IfSensorDataArrayIsFilledThenRun\n"
                global Fx = sensor_data[1]#Set input sensor data to appropiate variables\n"
                global Fy = sensor_data[2]\n"
                global Fz = sensor_data[3]\n"
                global Mx = sensor_data[4]\n"
                global My = sensor_data[5]\n"
                global Mz = sensor_data[6]\n"
        else: #IfItIsNotThenSetThemAllToZer0\n"
                global Fx = 0.0 # set the variables to zer0\n"
                global Fy = 0.0\n"
                global Fz = 0.0\n"
                global Mx = 0.0\n"
                global My = 0.0\n"
                global Mz = 0.0\n"
        end #EndIf\n"
```

Figur 13: Utklipp av koden i threaden i URScript.

2.4.3 Forslag til videreføring av prosjektet

En viktig målsetning for prosjektet er at det er tilrettelagt for videreføring av arbeidet vi har gjort. Det er en mulighet at høgskolen lager en valgfri laboratorieøving for studenter på automasjonslinjen som innebærer bruk av roboten. På grunn av dette har vi brukt en del tid på å lage oversiktlige og lett forståelige programkoder i Java og URScript. Vi har også lagd en kort bruksanvisning for å enklere komme i gang med å bruke roboten (vedlegg 4).

Slik vi ser det kan logging av sensordata, leddposisjoner, hvor lenge en pasient har trent og hvilke øvelser pasienten har gjennomført være av interesse. Data kan for eksempel framstilles gjennom grafer og lignende, og burde være oversiktlig og lett å bruke. Fordelen for

helsepersonell er at de kan bruke progresjonen til å tilpasse treningsopplegget for hver enkelt pasient. For pasienten kan det være motiverende å se sin egen progresjon.

Det å lagre data kan utvides til å etterligne et spill slik at man kan motivere til å slå gamle rekorder. For hver øvelse må man finne ut hva som skal gi poenger. For drakampen kan man for eksempel bruke største kraften som virket på sensoren, eller regne ut arbeidet som blir utført gjennom øvelsen. For øye-handkoordinasjonsøvelsen ser vi mulighet til å bruke tiden fra roboten står i ro til ballen blir dyttet på. Selve implementeringen av datalagring kan bli vanskelig på grunn av lover og forskrifter som begrenser hvordan data tilknyttes enkeltpersoner.

En stor fordel med å bruke en robot er at man kan lagre mange øvelser. Det vil alltid være rom for å legge til flere. Man kan for eksempel bruke kamera til å lage et program som leser et mønster fra et whiteboard og fører pasienten lags dette mønsteret etterpå. Her kan man også hjelpe og hindre pasienten.

Roboten har en løftekapasitet på 5 kg men denne løftekapasiteten er redusert siden vi har montert utstyr på tuppen. Dette utstyret veier 1,985 kg som gjør at den reelle løftekapasiteten er omtrent 3 kg. I fremtiden kan denne ekstra vekten påvirke robotens evne til å hjelpe eller hindre pasienten, og av den grunn burde man se på alternative verktøy som veier mindre.

2.4.4 Konklusjon

En UR5-robot vil kunne bidra til rehabilitering på en bedre måte enn andre roboter på grunn av de innebygde sikkerhetsfunksjonene den kommer med. Den er også svært fleksibel og brukervennlig. Vi mener den vil kunne hjelpe å avlaste noe av arbeidsmengden for helsepersonell i forbindelse med rehabilitering. Samtidig tror vi at den har potensiale til å forbedre rehabilitering gjennom blant annet å tilby et stort spekter av øvelser.

For å få til denne rehabiliteringen med en UR5-robot er sikkerhet en viktig faktor. Som nevnt var dette en vesentlig grunn til at vi valgte UR5-roboten. Helsepersonell som skal bruke en slik innretning må være trygg på robotarmen som et rehabiliteringsapparat. Pasienten må også ha denne tryggheten, og ofte så går denne tilliten via helsepersonellet.

Potensialet robotarmen har er et sentralt tema. Andre pasientgrupper utover slagpasienter som har behov for rehabilitering er også en potensiell gruppe som kan benytte seg av en slik innretning. Et annet potensiale ligger i logging av data fra øktene, som kan hjelpe med tilpassing av rehabiliteringsopplegg. Man kan også lage øvelser med et spillpreg. Dette vil kunne øke motivasjon hos en pasient som i mange tilfeller sliter med motivasjon til rehabiliteringstreningen.

Med arbeidet vårt til grunn konkluderer vi med at vi har nådd våre mål. Sluttresultatet vårt demonstrerer potensialet en robotarm har innen rehabilitering. Samtidig har vi greid å introdusere roboten på en sikker og trygg måte for personer som ikke har noe forhold til en robotarm fra før. Vi mener også at det ligger godt til rette for videre utvikling av robotarmen som en innretning for rehabilitering.

3.0 Prosjektadministrasjon

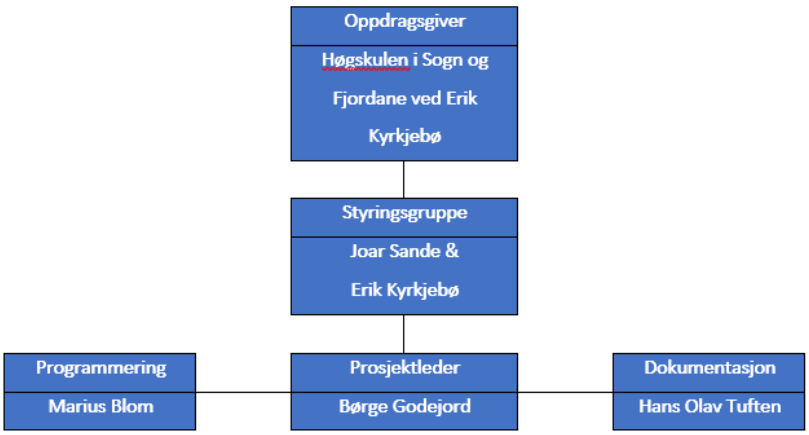
Dette kapittelet omhandler det prosjektadministrative rundt prosjektet. Det går på organisering, ressurspersoner, gjennomføring i forhold til plan, ressursforbruk, verktøy og en generell prosjektevaluering.

3.1 Personer involvert i prosjektet

Det er mange personer som har vært involvert i prosjektet i større og mindre grad.

3.1.1 Organisering

Prosjektet er organisert i tre nivå: Oppdragsgiver, styringsgruppe og arbeidsgruppe.



Figur 14: Organisering av bachelorprosjektet.

Oppdragsgiver er i dette prosjektet Høgskulen i Sogn og Fjordane ved Erik Kyrkjebø. Høgskulen i Sogn og Fjordane er en høyskole med hovedkontor i Sogndal, og undervisningslokaler i Sogndal og Førde. Høyskolen har totalt 330 ansatte og tilsammen 3800 studenter [16].

Styringsgruppen består av Erik Kyrkjebø og Joar Sande. Arbeidsgruppen består av tre avgangsstudenter med ulik bakgrunn og arbeidserfaring. Prosjektleder er Børge Godejord, programmeringsansvarlig er Marius Blom og Hans Olav Tuften er ansvarlig for dokumentering. I arbeidsplanen (vedlegg 2) satte vi tidsfrister og hvem i arbeidsgruppen som var ansvarlig for gjennomføring av de forskjellige delmålene.

3.1.2 Ressurspersoner

Det mest kritiske momentet i prosjektet var programmering av roboten. Her har vi fått god hjelp av Rocketfarm AS, en bedrift med tilholdssted i Sogndal som jobber innen automasjon og IT. Marcin Fojcik er førstelektor ved Høgskulen i Sogn og Fjordane, og har også vært en god ressurs i programmeringsarbeidet.

Arbeidsgruppen består utelukkende av ingeniørstudenter, og har dermed lite forhåndskunnskap om hjerneslag. Etter et møte med Erik Kyrkjebø tok vi kontakt med Eli Natvik. Eli er førsteamanuensis ved Høgskulen i Sogn og Fjordane, og har tidligere blant annet jobbet som fysioterapeut på Førde Sentralsjukehus med slagpasienter. Vi fikk et møte med Eli der vi fikk et innblikk i rehabilitering av hjerneslagpasienter og generelt informasjon om hjerneslag.

3.2 Gjennomføring i forhold til plan

Bacheloroppgaven hadde tidsfrister som fungerte som milepæler og delmål i prosjektet. Med utgangspunkt i disse lagde vi et Gantt-skjema i regneark (vedlegg 1) og en arbeidsplan (vedlegg 2). Vi har holdt alle tidsfrister og har gjennomført bra i forhold til plan. Anskaffelse av ny pidestall til roboten tok lengere tid enn først forventet, men vi holdt likevel den endelige fristen.

Tabell 2: Tidsfrister for bacheloroppgaver 2016.

Frist	Oppgave
19. februar	Forprosjekt innlevert Nettside opprettet
6. april	Midtvegspresentasjon
11.mai	Pressemelding sendt ut
23. mai	Rapport Plakat klar & hengt opp
27. mai	Presentasjon
3. juli	Nettside ferdigstilt Opprydding ferdig

3.3 Ressursforbruk

Ressurser i prosjektet er i hovedsak tid og penger. Det er satt opp en forventet tidsbruk på 500 timer for hver student, og skolen dekker vanligvis materialer til prosjekter etter avtale.

3.3.1 Tid

I tillegg til bacheloroppgaven har alle tre medlemmer i arbeidsgruppen emnet *Programmering 2*, og to av medlemmene har også tatt *Matematikk 3*. Dette har ført til at vi har hatt mye å gjøre utenfor bacheloroppgaven. Emnet *Programmering 2* tok vi på bakgrunn i at det skulle hjelpe oss med bacheloroppgaven, noe som det i stor grad har gjort. Flere tema som vi har hatt undervisning i *Programmering 2* har vi brukt i programmet for å koble til og styre roboten.

Vi satte opp en plan med full dag mandag og fredag til bacheloroppgave, i tillegg har vi supplert med ekstra tid etter lunsj tirsdag, onsdag og torsdag. Helgene har stort sett blitt brukt til individuelt arbeid med kontakt over nett. Prosjektmøter ble satt opp onsdag hver tredje uke kl.14 som et utgangspunkt. Disse har til tider blitt flyttet eller avlyst ved behov. Rettledende timetall for prosjektet er 500 timer for hver student, eller 1500 totalt for vår gruppe. Til rapporten vart levert hadde vi omtrent 1300, og etter presentasjonenen forventer vi å ligge på rundt 1500.

Tabell 3: Viser de planlagte dagene, tom rute er disponibel til prosjektarbeid.

Tidsrom	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag
0830 - 1010	Bachelor	Matte 3	Java 2	Matte 3	Bachelor		
1020 - 1200	Bachelor	Java 2	Java 2	Matte 3	Bachelor		
1200 - 1230	Lunsj	Lunsj	Lunsj	Lunsj	Lunsj		
1230 - 1410	Bachelor				Bachelor		
1420 - 1600	Bachelor		Møte		Bachelor		

3.3.2 Kostnader

For å gjennomføre prosjektet måtte vi ha tilgjengelig UR5-roboten og kraftsensoren, tidlig i prosjektperioden så vi også behovet for et nytt fundament til roboten. Uten disse komponentene blir deler av prosjektet ugjennomførbart. Roboten lånte vi fra høyskolen, og forsikret oss om at det bare var vår gruppe som skulle bruke den dette semesteret. Erik Kyrkjebø skaffet kraftsensoren, den ble sendt fra Robotiq i Canada før jul. Skolen dekket denne, og den kommer til å være disponibel for andre studenter ved høyskolen etter bachelorprosjektet er gjennomført. En stabil pidestall i stål er nå det nye fundamentet til roboten. Denne ble bestilt fra APS Robotics As, samme firma som kraftsensoren ble bestilt fra. Pidestallen ble også dekket av skolen og gir nå roboten vesentlig bedre stabilitet. Ellers har vi gått til anskaffelse av diverse andre deler fra Biltema til relativt små summer. Disse komponentene har vi i hovedsak brukt i forbindelse med oppgaver roboten skulle utføre.

I tabell 3 kan man se hva vi budsjetterte med opp mot hva vi har fått av utlegg. Her ser man at det ikke er prosentvis stort avvik men at vi har overskredet budsjettet med 1787,70kr. Det er i hovedsak pris på pidestall som utgjør denne forskjellen. Endelig kostnad for pidestallen ble avklart med styringsgruppen før den ble bestilt.

Tabell 4: Budsjett og resultat.

Artikkel	Antall	Budsjettert	Antall	Resultat
FT 150 sensor	1 stk	36 000 kr	1 stk	36 000 kr
Festeplate for sensor	1 stk	2 250 kr	1 stk	2 250 kr
Frakt av sensor, Canada	1 stk	1600 kr	1 stk	1600 kr
Frakt av sensor, Norge	1 stk	369 kr	1 stk	369 kr
Whiteboard	1 stk	89,90 kr	1 stk	89,90 kr
Whiteboard tusj	1 stk	24,90 kr	1 stk	24,90 kr
Maskinskruer (M6)	1 pk	20 kr	1 pk	20 kr
Maskinskruer (M8)	1 pk	20 kr	1 pk	20 kr
Antisklimatte	1 stk	50 kr	1 stk	50 kr
Hagehanske	1 par	14,90 kr	1 par	14,90 kr
Akryltetning 300ml	1 stk	19,90 kr	**2 stk**	**39,80 kr**
Planskive	1 pk	24,90 kr	1 pk	24,90 kr
Rørklammer	2 stk	49,80 kr	2 stk	49,80 kr
Plakat	1 stk	200 kr	1 stk	200 kr
Frakt av router	1 stk	145 kr	1 stk	145 kr
Ny fot til armen	1 stk	8000 kr	1 stk	**9688 kr**
Ball/Tennisball	0	0	**1 pk**	**24,90 kr**
Bagasjestropper	0	0	**1 pk**	**54,90 kr**
Totalt		**48 858,40 kr**		**50 646,10 kr**

3.4 Programvare og dokumentbehandling

Her følger kort informasjon om dataverktøy og elektroniske tjenester vi har benyttet.

3.4.1 Programvare

Figur 15: Logo, Ubuntu.

I starten av prosjektet kjørte vi Ubuntu som operativsystem på datamaskinene vi programmerte roboten fra. Ubuntu er basert på Debian GNU/Linux, og har veldig stort fokus på brukervennlighet. Tanken bak Ubuntu er at den skal benytte fri programvare, som gjør det enklere å tilby et oppdatert og gratis operativsystem. Versjonen vi bruker er Ubuntu 14.04.3

Figur 16: Logo, Qt.

Qt er et utviklingsverktøy med grafisk grensesnitt. Dette brukte vi i starten sammen med Ubuntu når vi så på programmering i språket C++. Vi gikk etterhvert over til å programmere i Java siden det er et språk vi har kjennskap til.

Figur 17: Logo, NetBeans.

For å programmere Java vil vi bruke NetBeans. Dette er det mest brukte utviklingsverktøyet for Java, og er veldig brukervennlig. Java er et objekt-orientert programmeringsspråk som har vært en del av pensum tidligere semester. Versjonen vi bruker er NetBeans 8.1

Figur 18: Logo, Wordpress.

Vi brukte publiseringsverktøyet Wordpress til hjemmesiden vår. Denne baserer seg på PHP og MySQL databaser for lagring av innholdet. Mye av nettsiden vil da være et ferdiglaget oppsett som vi kan editere slik vi ønsker det selv. For å få bedre kontroll på nettstedet vårt, valgte vi å sette opp en database selv, for deretter å kjøre nettstedet selv. Versjonen vi bruker er Wordpress 4.4.2

Figur 19: Logo, AutoCAD.

AutoCAD er en applikasjonsprogramvare for 2D og 3D dataassistert konstruksjon. Vi brukte AutoCAD til å tegne nytt fundament til roboten. Vi var inne på muligheten å tegne pidestall selv og få den konstruert i nærområdet. Det viste seg å være tidkrevende og vanskelig å finne noen som ville produsere pidestallen, slik at vi endte med å kjøpe pidestall fra en profesjonell aktør.

Figur 20: Logo, Adobe Photoshop.

Adobe Photoshop er et avansert dataverktøy for redigering av bilder. Photoshop ble brukt til å lage informasjonsplakaten til prosjektet og redigere bilder til nettside og rapport.

Figur 21: Logo, iMovie.

iMovie videoredigeringsverktøy for OS X. Vi ønsket å demonstrere oppgavene vi utførte underveis i prosjektet, og da ønsket vi å formidle dette via videosnutter. iMovie har et enkelt og oversiktlig brukergrensesnitt ved redigering av videosnutter. Versjonen vi bruker er iMovie 10.1.1.

3.4.2 Dokumentbehandling

Figur 22: Logo, Dropbox.

For å dele dokumenter og andre filer har vi brukt Dropbox. Dropbox er en fildelingstjeneste som fungerer som en ekstern harddisk, som lagres i skyen. På datamaskiner vil en mappe bli kopiert over til skyen automatisk, og oppdateres. Denne mappen kan man dele med andre, og ble i vårt prosjekt brukt som prosjektperm.

Figur 23: Logo, Google dokumenter.

Timeliste, risikovurdering, loggbok og Gantt-skjema ble lagd i Google Docs til tidligere prosjekt, og vi har fortsett å bruke disse dokumentene. Fordelen med Google Docs sin tjeneste er at man kan dele dokumenter med ulike personer, og gi dem ulik grad av tilgang. Dokumentene vil også oppdateres i sanntid, som gjør det mindre sannsynlig at noen lagrer over andre sitt arbeid.

3.5 Generell prosjektevaluering

Arbeidsgruppen er veldig fornøyd med gjennomføringen av bacheloroppgaven og selve bacheloroppgaven vi har fått jobbe med. Vi har tilegnet oss ny kunnskap og nye erfaringer også utover den tekniske og faglige delen av oppgaven. Av den tekniske delen har programmering vært størsteparten av oppgaven og her har læringen vært stor. Vi har i hovedsak brukt URScript og Java til å programmere roboten og det er dermed disse språkene vi har tilegnet oss mest kunnskap innen. I Java har vi brukt threading for å få deler av program til å kjøre samtidig, socket har vi brukt til å få kontakt med roboten og vi har lagd et enkelt grafisk brukergrensesnitt ved hjelp av JavaFX.

I forhold til prosjektgjennomføring har vi forsøkt å bruke teori og tips vi lærte i emnet *Ingeniørfagleg Systememne* og tidligere prosjekter vi har gjennomført. Gantt-skjema og arbeidsplan med klare ansvarsområder er dokumenter som vi har dratt nytte av ved styring og gjennomføring av prosjektet. En kort orientering før dager med prosjektarbeid er noe vi har startet med underveis i prosjektet. Noen minutter her øker arbeidseffektiviteten gjennom at vi får oppsummert og satt klare planer for dagen. På slutten av prosjektperioden ble vi invitert til Polen for å presentere bachelorprosjektet på en konferanse for datasystem i helse og medisin. Dette gav oss en god erfaring innen det å presentere og et innblikk i mange spennende prosjekt. Kort oppsummert så er vi veldig fornøyde med bacheloroppgaven. Vi har nådd alle mål og vi føler vi har nådd de på en god måte.

Litteraturliste

[1] I. M. Liseter, «Robot,» Store Norske Leksikon, 28 06 2012. [Internett]. Available: https://snl.no/robot. [Funnet 01 02 2016].

[2] L. Gjerstad, «Hjerneslag,» Store Medisinske Leksikon, 13 02 2009. [Internett]. Available: https://sml.snl.no/hjerneslag. [Funnet 01 02 2016].

[3] N. f. folkehelse, 18 12 2015. [Internett]. Available: http://nasjonalforeningen.no/hjerte-og-kar/ulike-hjertesykdommer/hjerneslag/.

[4] G. Larsen og H. Heggen, «NRK,» NRK, 08 02 2016. [Internett]. Available: http://www.nrk.no/norge/_-far-du-hjerneslag-har-du-darlig-tid-1.12794068 . [Funnet 11 02 2016].

[5] C. R. Johannesen og J. L. B, «Tidsskrift for den norske legeforening,» Den norske legeforeningen, 07 04 2015. [Internett]. Available: http://tidsskriftet.no/article/3320212. [Funnet 11 02 2016].

[6] E. Borge, «Livsstilssykdomsnorge,» Media Planet, 09 2015. [Internett]. Available: http://www.livsstilssykdommernorge.no/sykdommer/hjerneslag/veien-tilbake-fra-hjerneslag. [Funnet 11 02 2016].

[7] L. S. I. L. WENCHE HEIR. [Internett]. Available: http://ndla.no/nb/node/24132.

[8] B. Indredavik, R. Salvesen, H. Næss og D. Thorsvik, Behandling og rehabilitering ved hjerneslag, Oslo: Helsedirektoratet, 2010.

[9] M. Bélanger-Barrette, «Questions on Universal Robots Answered: Here!,» Robotiq, 14 08 2015. [Internett]. Available: http://blog.robotiq.com/all-your-questions-on-universal-robots-answered-here. [Funnet 17 02 2016].

[10] J. Falconer, «Universal Robots' tablet-controlled factory robots work safely with humans,» Gizmag, 16 12 2012. [Internett]. Available: http://www.gizmag.com/universal-robots-tablet-factory/25452/. [Funnet 17 02 2016].

[11] Universal Robots A/S, «Microsoft Word - universal_robots_zacobria_hints_and_tips_manual_1_4_3,» 01 03 2012. [Internett]. Available: https://www.zacobria.com/pdf/universal_robots_zacobria_hints_and_tips_manual_1_4_3.pdf. [Funnet 17 02 2016].

[12] Robotiq, «Robotiq-Force-Torque-Sensor-FT-150-Specifications.pdf,» 10 09 2014. [Internett]. Available: http://robotiq.com/wp-content/uploads/2014/09/Robotiq-Force-Torque-Sensor-FT-150-Specifications.pdf. [Funnet 22 05 2016].

[13] Universal Robots A/S, «UR robots - Easy programming, fast set up, flexible,» Universal Robots, 01 01 2015. [Internett]. Available: http://www.universal-robots.com/products/ur-robot-benefits/. [Funnet 22 05 2016].

[14] Universal Robots A/S, «Remote Control Via TCP/IP - 16496,» Universal Robots A/S, 01 01 2015. [Internett]. Available: http://www.universal-robots.com/how-tos-and-faqs/how-to/ur-how-tos/remote-control-via-tcpip-16496/. [Funnet 22 05 2016].

[15] Computer Science University of Toronto, «Windows vs. Unix Line Endings,» Computer Science University of Toronto, 04 01 2005. [Internett]. Available: http://www.cs.toronto.edu/~krueger/csc209h/tut/line-endings.html. [Funnet 22 05 2016].

[16] «Om HiSF,» Høgskulen i Sogn og Fjordane, [Internett]. Available: https://www.hisf.no/nn/om-hisf. [Funnet 01 02 2016].

Bildereferanser

Framside «Framside» av Børge Godejord (19 05 2016).

Figur 1 «Fig. 26» (2013) [Figur] Available: https://www.google.com/patents/US8410732 (Funnet 22 05 2016).

Figur 2 «shutterstock_hjerne_tekst-stor» (2012) [Figur] Available: http://www.slag.no/Hjerneslag/Hva-er-hjerneslag (Funnet 22 05 2016).

Figur 3 «v-h hjerne» (2012) [Figur] Available: http://www.slag.no/novus/upload/tab1/Article/Hjerneslag/v-h-hjerne.png (Funnet 17 02 2016).

Figur 4 *«Pidestallen før robotarmen ble festet»* av Børge Godejord (31 03 2016).

Figur 5 «Verktøyet vi lagde og brukte i øvelsene» av Marius Blom (19 05 2016).

Figur 6 «Utklipp av koden som oppretter kobling mellom Java og robotarmen» av Marius Blom (22 05 2016).

Figur 7 «Biblotek-klassen inneholder URScript. Script er lagret som String-verdier» av Marius Blom (22 05 2016).

Figur 8 «Oversikt over kommunikasjon mellom klassene i Java, roboten og sensoren»

av Hans Olav Tuften (21 05 2016).

Figur 9 «Det grafiske grensesnittet laget i JavaFX. På bildet ligger musepekeren over BallRehab» av Hans Olav Tuften (19 05 2016).

Figur 10 «Metoden som sender koden til roboten» av Marius Blom (22 05 2016).

Figur 11 «Det fungerene programmet» av Marius Blom (22 05 2016).

Figur 12 «Sammenligning av den leste datastrømmen fra roboten» av Marius Blom (22 05 2016).

Figur 13 «Utklipp av koden i threaden i URScript» av Marius Blom (22 05 2016).

Figur 14 *«Organisering av bachelorprosjektet»* av Hans Olav Tuften (17 02 2016).

Figur 15 *«Ubuntu Logo»*(2012) [Figur] Available: http://refugeeks.com/wp-content/uploads/2012/08/Ubuntu-Logo.png (Funnet 17 02 2016).

Figur 16 *«qt»* () [Figur] Available: https://bugreports.qt.io/secure/attachment/33234/qt.png (Funnet 17 02 2016).

Figur 17 *«netbeans icon»* [Figur] Available: http://www.file-extensions.org/imgs/app-icon/128/5149/netbeans-icon.png (Funnet 17 02 2016).

Figur 18 *«wordpress logo notext rgb»* [Figur] Available: https://s.w.org/about/images/logos/wordpress-logo-notext-rgb.png (Funnet 17 02 2016).

Figur 19 *«autocad logo vector download»* (2015) [Figur] Available: https://www.seeklogo.net/wp-content/uploads/2015/09/autocad-logo-vector-download.jpg (Funnet 17 02 2016).

Figur 20 *«latest»* [Figur] Available: http://vignette4.wikia.nocookie.net/logopedia/images/5/55/Ps7.png/revision/latest?cb=20141210093120 (Funnet 17 02 2016).

Figur 21 *«latest»* [Figur] Available: http://vignette1.wikia.nocookie.net/logopedia/images/6/6a/IMovie4.png/revision/latest?cb=20150622190412 (Funnet 17 02 2016).

Figur 22 *«MetroUI Apps Dropbox icon»* (2014) [Figur] Available: http://educationcluster.net/wp-content/uploads/2014/11/MetroUI-Apps-Dropbox-icon.png (Funnet 17 02 2016).

Figur 23 *«docs»* (2015) [Figur] Available: http://odyssey.antiochsb.edu/wp-content/uploads/2015/05/docs.png (Funnet 17 02 2016).

Tabellreferanser

Tabell 1 «SPECIFICATIONS» (2016) [Figur] Available: http://robotiq.com/products/robotics-force-torque-sensor/ (Funnet 22 05 2016).

Tabell 2 *«Tidsfrister for bacheloroppgaver 2016»* av Hans Olav Tuften (17 02 2016).

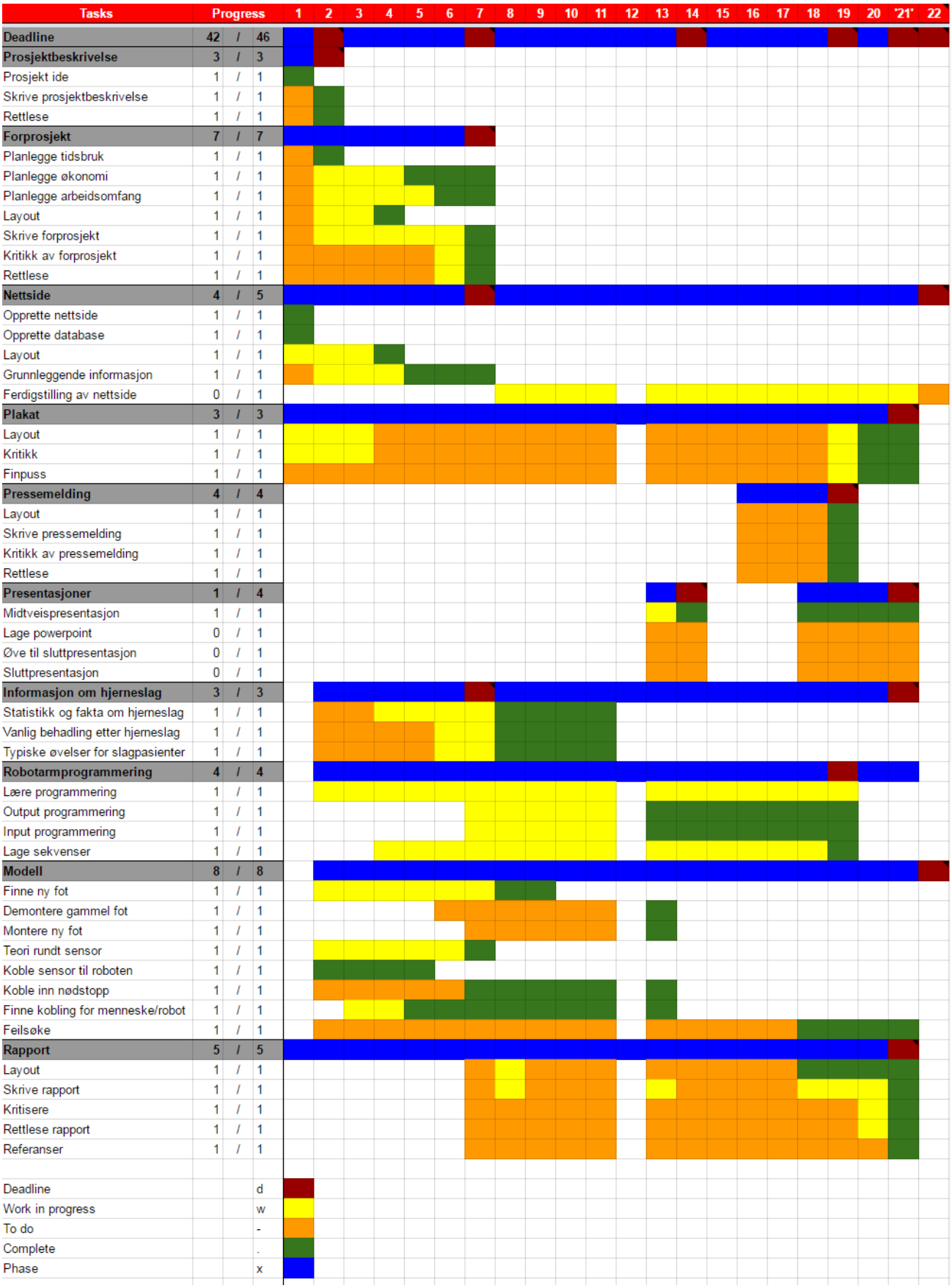
Tabell 3 *«Viser de planlagte dagene, tom rute er disponibel til prosjektarbeid»* av Hans Olav Tuften (17 02 2016).

Tabell 4 *«Budsjett og resultat»* av Børge Godejord (17 02 2016).

Vedlegg

Vedlegg 1 Gantt-skjema
Vedlegg 2 Arbeidsplan
Vedlegg 3 Timeliste
Vedlegg 4 Hurtigstart
Vedlegg 5 Utklipp fra brukermanual FT 150
Vedlegg 6 GUI.java
Vedlegg 7 URsocket.java
Vedlegg 8 LibraryURScript.java
Vedlegg 9 Coordinates.java

Vedlegg 1 Gantt-skjema



Vedlegg 2 Arbeidsplan

MÅL	AKTIVITET	TID	ANSVARLIG	Status
Delmål 1: Levere prosjektbeskrivelse	*Konkrete tiltak:* Prosjektidé, skrive og rettlese.	*Frist:* 22.01.16	Prosjektleder - Børge Godejord	OK
Delmål 2: Opprette nettside	*Konkrete tiltak:* Opprette database, opprette nettside med layoute og grunnleggende informasjon.	*Frist:* 19.02.16	Programmering - Marius Blom	OK
Delmål 3: Levere forprosjektrapport	*Konkrete tiltak:* Planlegge prosjekt, tilegne seg informasjon og kunnskap, skrive rapport og rettlese/rette.	*Frist:* 19.02.16	Dokumentasjon - Hans Olav Tuften	OK
Delmål 4: Anskaffelse av stabil og flyttbar pidestall til UR5-robot.	*Konkrete tiltak:* Innhenting av tilbud med pris og leveringstid, bestilling og montering av pidestall.	Skal være i vår besittelse før det blir kritisk i forhold til å gjøre nøyaktige bevegelser med UR5, *Frist:* 01.04.16	Prosjektleder - Børge Godejord	OK
Delmål 5: Midtveis-presentasjon	*Konkrete tiltak:* Lage PowerPoint, gjennomgang.	*Frist:* 06.04.16	Prosjektleder - Børge Godejord	OK
Delmål 6: Sende ut pressemelding	*Konkrete tiltak:* Skrive, rettlese og rette.	*Frist:* 11.05.16	Prosjektleder - Børge Godejord	OK
Delmål 7: Levere plakat	*Konkrete tiltak:* Layout, skrive tekst, kritikk, finpuss.	*Frist:* 23.05.16	Prosjektleder - Børge Godejord	OK
Delmål 8: Programmering av robot.	Løpende prosess som pågår gjennom hele prosjektperioden. Etterhvert som vi tar nye “steg/mål” går vi videre.	*Frist:* 23.05.16	Programmering - Marius Blom	OK

	Konkrete tiltak: Tilegne seg kunnskap og informasjon. Programmere, teste, feilsøke og rette for så å utvide/tilpasse program. Skrive en solid/robust grunnstruktur som lett kan brukes videre og utvides.			
Delmål 9: Modell	Løpende prosess som pågår parallelt med programmering av roboten. Modellen utvides med verktøy etter behov. *Konkrete tiltak:* Lage verktøy for ulike øvelser/program	*Frist:* 23.05.16	Prosjektleder - Børge Godejord	OK
Delmål 10: Levere hovedrapport	*Konkrete tiltak:* Skrive rapport, rettlese og rette.	*Frist:* 23.05.16	Dokumentasjon - Hans Olav Tuften	OK
Delmål 11: Presentasjon	*Konkrete tiltak:* Lage PowerPoint, gjennomgang.	*Frist:* 27.05.16	Prosjektleder - Børge Godejord	Under arbeid
Delmål 12: Avslutning og opprydning,	*Konkrete tiltak:* Rydde opp etter prosjektet. Ferdigstille nettside. (Nettside skal oppdateres fortløpende gjennom hele prosjektet)	*Frist:* 03.06.16	Prosjektleder - Børge Godejord	Under arbeid

Vedlegg 3 Timeliste

Dato	Antal timar				Notat
	MB	BG	HOT	Totalt	
04.12.2015	1,0	1,0	1,0	3,0	Møte med Erik Kyrkjebø
04.12.2015	1,0	1,0	1,0	3,0	Diskusjon rundt prosjektet [Møte]
06.12.2015	3,0			3,0	Nettside
06.01.2016	1,5	1,5	1,5	4,5	Planleggingsmøte
07.01.2016		1,5		1,5	Rapport
07.01.2016	1,5	1,5	1,5	4,5	Møte med Erik Kykjebø
07.01.2016	1,0	2,0	2,0	5,0	Forprosjektrapport
08.01.2016	1,0	2,0		3,0	Rapport
09.01.2016		1,0		1,0	Rapport
09.01.2016	1,0	3,0	0,5	4,5	Plakat, kritikk til plakat
11.01.2016	9,0	9,5	9,0	27,5	Nettside, Gantt, Prosjektbeskrivelse, Risikovurdering, Prosjektbeskrivelse
12.01.2016	2,0	2,0	2,0	6,0	Java - oop
12.01.2016		3,5		3,5	Risikovurdering, prosjektbeskrivelse, info innsamling kraftsensor, møteinnkalling
12.01.2016	0,5	0,5	0,5	1,5	Diskusjon om oppmøte på relevant temadag
12.01.2016	1,0	1,0	1,0	3,0	Prøvekjøring med kraftsensor påmontert (ikkje kopla)
12.01.2016		4,0		4,0	Lese seg opp på FT150. Og fysisk montering.
13.01.2016	4,0	4,0	4,0	12,0	Temadag om Telemedisin og Velferdsteknologi. Fylle inn notat i logg
13.01.2016	1,0	1,5	1,0	3,5	Software update UR5CB2 fra 1.7.10293 til 1.8.14035, programmering
13.01.2016	1,0	1,0	1,0	3,0	Forprosjektrapport
14.01.2016	3,0	4,0		7,0	Lese seg opp på hjerneslag.
15.01.2016	7,5	8,0	8,0	23,5	Koblet opp sensor til robot, forprosjektrapport
15.01.2016	1,0			1,0	Nettside
16.01.2016	1,0	1,5		2,5	Finne materiell til tusjfeste. Lage handleliste med priser og plassering. Nettside
18.01.2016	1,5	1,5		3,0	forberedelse til møte, nettverk
18.01.2016	9,0	4,0		13,0	Programmering
18.01.2016		3,0	8,5	11,5	Forprosjekt
19.01.2016	2,0	2,0	2,0	6,0	Java - oop
19.01.2016			0,5	0,5	Forprosjekt
19.01.2016	1,5		1,5	3,0	demonstrasjon, Forberedelse til møte
20.01.2016	4,0	4,0	4,0	12,0	Java - oop

20.01.2016	2,0	2,0	2,0	6,0	Møte med preben ang forrige års rapport + diskusjon etterpå
20.01.2016	2,0	2,0	2,0	6,0	Møte med Erik Kyrkjebø
20.01.2016	0,5	0,5	0,5	1,5	Risikovurdering
22.01.2016	1,0		7,5	8,5	Forprosjekt
22.01.2016	7,0		1,5	8,5	Programmering
22.01.2016		8,0		8,0	Innhenting av informasjon, handle, lage tusjfeste og støpe hånd. 3D tegning
22.01.2016	1,0	1,0		2,0	Nettside, videoredigering
23.01.2016	4,0			4,0	Programmering
25.01.2016	1,5	1,5	1,5	4,5	Møte med Marcin Fojcik ang. Programmering av roboter
25.01.2016		0.5	5,0	5,0	Forprosjektrapport
25.01.2016		6,0		6,0	Nettside, innhenting av informasjon. administrerende.
25.01.2016	9,0	1,0	2,0	12,0	Programmering, installering av Ubuntu og Qt
26.01.2016	2,0	2,0	2,0	6,0	Java - oop
27.01.2016		6,0		6,0	STYRA med platformfot te UR5
28.01.2016		4,0		4,0	Tegne platformfot
29.01.2016		6,0		6,0	Tegne platformfot, prat med Johannes. Innenting av tilbud
01.02.2016	2,0	3,0	8,5	13,5	Rapport
01.02.2016	6,0	5,0		11,0	Installasjon av Ubuntu m/ div. software. Admin møte og platformfot
02.02.2016		1,0		1,0	Forprosjektrapport
03.02.2016	1,5	1,5	1,5	4,5	Møte med Eli Natvik
04.02.2016	1,0	2,0	1,0	4,0	Forprosjektrapport
04.02.2016	1,5	1,5	1,5	4,5	Møte med Marcin Fojcik ang. Programmering av roboter
05.02.2016		4,0	7,0	11,0	Forprosjektrapport
05.02.2016	7,5			7,5	Programmering, samtale med Marcin
05.02.2016	1,0	4,5		5,5	Modell
06.02.2016		0,5		0,5	Møteinnkalling
06.02.2016	1,0		4,0	5,0	Forprosjektrapport
07.02.2016			2,5	2,5	Forprosjektrapport
08.02.2016	2,5	2,5	2,5	7,5	Programmering, samtale med Marcin
08.02.2016	3,5	7,5	7,0	18,0	Forprosjektrapport
08.02.2016			1,0	1,0	Diskusjon m mamma?
10.02.2016	1,0	1,0	1,0	3,0	Møte med Erik Kyrkjebø, Marcin
10.02.2016	1,5	1,5	1,5	4,5	Diskusjon etter møte
10.02.2016		0,5		0,5	Møtereferat
11.02.2016	6,5	6,0	9,5	22,0	Forprosjektrapport

11.02.2016	2,0		2,0	4,0	Presentasjon av prosjekt til VGS elektro
12.02.2016		4,0	4,0	8,0	Forprosjektrapport
12.02.2016	1,0		1,5	2,5	Informasjon ang UR5
12.02.2016	8,0	4,0	2,5	14,5	Programmering
13.02.2016		0,5		0,5	Programmering (Mail til nLink)
15.02.2016	2,0			2,0	Programmering, rydding i koden
15.02.0216	5,0	6,0	6,0	17,0	Forprosjektrapport
15.02.0216		1,5		1,5	Modell
15.02.2016	1,5			1,5	Rocketfarm
15.02.2016	1,5			1,5	Nettside, statistikk
16.02.2016	1,5			1,5	Nettside, ikon, galleri
16.02.2016	1,0		4,0	5,0	Forprosjektrapport
17.02.2016	3,0	7,5	7,0	17,5	Forprosjektrapport
17.02.2016	4,0			4,0	Programmering, back to basics
18.02.2016	2,5	5,5	2,5	10,5	Forprosjektrapport
18.02.2016	0,5	0,5		1,0	Modell. Hullfeste "hånd"
19.02.2016	0,5	0,5		1,0	Modell. Pidestall
22.02.2016		6,0	3,0	9,0	Rapport
22.02.2016	7,0		4,5	11,5	Programmering, mail Rocketfarm
22.02.2016	1,0	1,5	0,5	3,0	Nettside
24.02.2016		0,5		0,5	Modell. Pidestall
26.02.2016		1,0		1,0	Møteinnkalling
26.02.2016	8,0	4,0	4,5	16,5	Programmering
26.02.2016		2,5	3,0	5,5	Rapport
29.02.2016	7,0			7,0	Programmering "med" Rocketfarm
29.02.2016		6,0	7,0	13,0	Programmering
29.02.2016		1,0		1,0	Modell. Pidestall
01.03.2016	2,0	2,0	2,0	6,0	Java - Software engineering
02.03.2016	1,0	1,0	1,0	3,0	Møte
03.03.2016		0,5		0,5	Rapport
04.03.2016	7,5	7,0	7,5	22,0	Programmering
04.03.2016		0,5		0,5	Modell. Pidestall
07.03.2016	8,0	7,0	4,5	19,5	Programmering
07.03.2016		0,5		0,5	Nettside
07.03.2016			3,0	3,0	Rapport

07.03.2016	1,0	1,0	1,0	3,0	Diskusjon rundt programmering og lokale
08.03.2016	2,0	2,0	2,0	6,0	Java - threads
09.03.2016	2,0	2,0	2,0	6,0	Java - threads
11.03.2016	8,5	8,5	8,5	25,5	Programmering
11.03.2016	1,0			1,0	Nettside, loggbok
12.03.2016	1,0			1,0	Nettside
14.03.2016	10,5	9,5	9,5	29,5	Programmering
18.03.2016	5,0	4	5,0	14,0	Programmering
18.03.2016	1,5			1,5	Nettside
29.03.2016	0,5	0,5	0,5	1,5	Ta opp tråden igjen etter påske
29.03.2016		2,0		2,0	Modell. Montere ny pidestall
29.03.2016	2,0	0,5		2,5	Nettside
31.03.2016	1,0	2,0		3,0	Midtveispresentasjon
01.04.2016	1,0	1,0	1,0	3,0	Midtveispresentasjon
01.04.2016	1,5	1,5	1,5	4,5	Statusmøte
01.04.2016	5,0	5,0	5,5	15,5	Programmering
01.04.2016	2,0			2,0	Nettside, videoredigering
01.04.2016		1,0		1,0	Modell
03.04.2016	2,0		3,0	5,0	Rapport
03.04.2016			0,5	0,5	Midtvegspresentasjon
04.04.2016	7,5	7,5	7,5	22,5	Rapport
04.04.2016	1,5	1,5	1,5	4,5	Møte
06.04.2016	1,0	1,0	1,0	3,0	Møte med Erik
06.04.2016	0,5	0,5	0,5	1,5	Midtvegspresentasjon
07.04.2016		0,5		0,5	Rapport
08.04.2016	6,0	6,0		12,0	Programmering
08.04.2016	1,0	2,0	3,0	6,0	Statusmøte og møteplanlegging
08.04.2016	2,0			2,0	Nettside
08.04.2016			5,0	5,0	Rapport
08.04.2016		0,5		0,5	Modell
11.04.2016		1,5		1,5	Modell
11.04.2016		0,5		0,5	Møteinnkalling
11.04.2016		0,5		0,5	Administrasjon - refusjonkrav
11.04.2016	2,0	4,5	4,5	11,0	Rapport
11.04.2016	7,0	2,0	4,5	13,5	Programmering

12.04.2016		0,5		0,5	Rapport
12.04.2016	1,0			1,0	Programmering, rocketfarm
13.04.2016		2,0		2,0	Rapport
15.04.2016		6,0	4,0	10,0	Rapport
15.04.2016	8,0	2,0	5,0	15,0	programmering
15.04.2016	1,0			1,0	Nettside
18.04.2016	7,5			7,5	Rapport
18.04.2016	1,0	8,0	9,0	18,0	Programmering
18.04.2016	0,5	0,5	0,5	1,5	Møte
19.04.2016		2,0		2,0	Rapport
20.04.2016	1,0	1,0		2,0	Rapport
20.04.2016		1,0		1,0	Programmering
20.04.2016	1,5	1,5	1,5	4,5	Møte
20.04.2016		1,5		1,5	Programmering
21.04.2016	0,5	4,0	0,5	5,0	Programmering
21.04.2016	1,0			1,0	Administrativt
22.04.2016	7,0	7,0	2,0	16,0	Programmering
22.04.2016			4,0	4,0	Rapport
22.04.2016		0,5		0,5	Modell
25.04.2016		1,0		1,0	Modell
25.04.2016		2,0	4,5	6,5	Rapport
25.04.2016	0,5	2,0		2,5	Pressemelding
25.04.2016	6,5	2,0	1,0	9,5	Programmering
25.04.2016		2,0		2,0	Plakat
29.04.2016	7,0	8,0	11,0	26,0	Programmering
29.04.2016	1,0			1,0	Administrasjon
02.05.2016		0,5		0,5	Pressemelding
02.05.2016	7,0	4,0	9,5	20,5	Programmering
02.05.2016	1,0	2,0		3,0	Rapport
02.05.2016	1,0			1,0	Nettside
03.05.2016		2,0		2,0	Rapport
03.05.2016	2,0	2,0	2,0	6,0	Intervju
04.05.2016	2,5	5,0	5,0	12,5	Polen tur
05.05.2016	1,0		6,5	7,5	Rapport
05.05.2016	7,5	7,5		15,0	Presentasjon, Polen

05.05.2016			2,0	2,0	Programmering
06.05.2016	8,0	9,0	4,5	21,5	Presentasjon, Polen
06.05.2016			5,0	5,0	rapport
07.05.2016	1,0			1,0	Programmering
07.05.2016	4,0	6,0		10,0	Presentasjon, video
07.05.2016	4,0	4,0		8,0	Presentasjon, Polen
08.05.2015	3,0	3,0		6,0	Presentasjon, Polen
08.05.2015	2,0	3,0		5,0	Presentasjon,video
08.05.2015	1,0			1,0	Nettside
09.05.2015	8,0	8,0	8,0	24,0	Konferanse, Polen
11.05.2015	10,0	10,0	10,0	30,0	Konferanse, Polen
12.05.2015	7,0	7,0	7,0	21,0	Konferanse, Polen
13.05.2015	8,0	8,0	8,0	24,0	Konferanse, Polen
15.05.2015	1,0			1,0	Nettside, bilder
15.05.2015			2,0	2,0	Rapport
18.05.2015			3,0	3,0	Publisering
19.05.2015		0,5		0,5	Administrasjon
19.05.2015		3,5		3,5	Plakat
19.05.2015	6,0	4,5	10,5	21,0	Rapport
20.05.2015	8,0	8,0	12,5	28,5	Rapport
20.05.2015	0,5	0,5	0,5	1,5	Intervju
21.05.2015	9,0	11,5	9,0	29,5	Rapport
22.05.2016	7,0	3,0	8,0	18,0	Rapport
				1 304,5	Totalt før presentasjon

Vedlegg 4 Hurtigstart

Universal Robots – Startguide (v. 1.0)

Kjapp innføring for å komme i gang med tekstbasert programmering av UR5

Roboten må motta URScript for å kontrollere aksjonene roboten skal utføre. Den enkleste måten å skrive URScript på er å bruke en tekst editor. URScript ligner veldig på Python, så mange tekst editorer kan gjenkjenne koden som Python. Det er svært viktig å avslutte koden med linjeskift (en tom linje på slutten av koden). Vi brukte Java for å lese data fra roboten, i tillegg til å kontrollere hvilke URScript som skal brukes. Hvilket språk som brukes for å lese data er valgfritt, vår kildekode er skrevet i Java.
Roboten sender en kontinuerlig datastrøm på port 30002, denne porten er også dedikert til å motta URScript.
Real time klient på port 30003 (vi har ikke brukt denne)

Før start

- Koble til nettverkskabel
  - Anbefalt å lage sitt eget nettverk, i stedet for å bruke skolen sitt
- Finn IP addressen til roboten ved hjelp av berøringsskjermen

Programmering

- Socket port 30002 på roboten
  - URsocket.java inneholder koden for å koble til roboten
  - Sett IP til robot og PC i GUI.java
- Bevegelsen til robot skrives i URScript, som ligner Python
  - Bruk våre script som eksempel
  - Library.java
- Sensordata sendes på port 29999 fra UR i egen tråd
  - Kildekoden for sensordata ligger i Library.java,
    - threadSensorData()
  - Leses av i GUI.java
- TCP og joint positions
  - Sendes på port 30002
  - Leses av i Coordinates.java
  - Vises i GUI.java

Den enkleste måten å komme i gang med tekstbasert programmering er å bruke vår kode som utgangspunkt. Spør faglærer om koden.

Nettsider som kan være til hjelp
http://www.universal-robots.com/how-tos-and-faqs/how-to/
http://www.zacobria.com/universal-robots-zacobria-forum-hints-tips-how-to/

Vedlegg 5 Utklipp fra brukermanual FT 150

4.2 ROS

Note

Not yet available.

4.3 Universal Robots

The Robotiq Force Torque Sensor FT 150 can be directly and very easily integrated with Universal Robots using the software package described in this section. Once installed, the sensor readings will be available in Polyscope and will be directly usable in the robot program, for example to measure loads. The package contains a software which will run in the background and some program templates to read the data in polyscope.

Software Package Installation

1. Prior to installing the software package, please make sure that the Universal Robot controller software is up-to-date. If required, update the controller software according to the procedure provided by Universal Robots. The package was tested with software version. 1.8.14035. It will not work with versions 1.7 and prior.
2. The Robotiq Force Torque Sensor software package can be downloaded at support.robotiq.com
3. Extract the content of the zip file onto a blank USB flash drive.
4. Plug the flash drive into a USB port of the robot controller (for example the one on the teach pendant). The pendant screen will show some status information while the software package is automatically installing. Do not unplug the flash drive until the operation is completed.
5. When a green "USB" text is shown, unplug the USB drive.

Testing the Software Package

1. Make sure that the sensor is installed on the robot wrist, that the sensor cable is connected to the sensor and that it is routed to the Universal Robot control box. (Remember to practice good cable management and allow enough flexibility in your cable lines.)
2. According to the wiring section of this manual, connect the the power wires of the sensor to a 24V power source. The Universal Robot 24V and GND lines are available for this purpose. The blue led of the sensor should be lit at this point.
3. According to the "Wiring with USB to RS485 converter" section of this manual, connect the sensor wires to the converter. Plug the USB connector of the converter into the available USB port located inside of the Universal Robot control box.
4. Inside Polyscope, load the program name "test_sensor" located in the "sensor" folder and run it. The program will display popups with the following information: sensor firmware version, production year, serial number and the current readings for Fx, Fy, Fz, Mx, My and Mz.

Vedlegg 6 GUI.java

```
1.  package GUI;
2.
3.  import java.io.BufferedReader;
4.  import java.io.InputStreamReader;
5.  import java.net.InetAddress;
6.  import java.net.ServerSocket;
7.  import java.net.Socket;
8.  import java.text.DecimalFormat;
9.  import java.util.concurrent.Executors;
10. import java.util.concurrent.ScheduledExecutorService;
11. import java.util.concurrent.TimeUnit;
12. import javafx.animation.KeyFrame;
13. import javafx.animation.Timeline;
14. import javafx.application.Application;
15. import javafx.event.ActionEvent;
16. import javafx.event.EventHandler;
17. import javafx.geometry.Insets;
18. import javafx.scene.Scene;
19. import javafx.scene.control.Button;
20. import javafx.scene.control.Label;
21. import javafx.scene.control.SplitPane;
22. import javafx.scene.layout.GridPane;
23. import javafx.scene.layout.StackPane;
24. import javafx.stage.Stage;
25. import javafx.util.Duration;
26.
27. public class GUI extends Application {
28.
29.     /*
30.     Here we can set the ip-adress and port to the UR5
31.      */
32.     static final String SERVERUR = "192.168.0.101";
33.     static final int PORTUR = 30002;
34.
35.     /*
36.     Here we can set the ip-address and port for the PC
37.      */
38.     static String serverPC = "192.168.0.105";
39.     static final int PORTPC = 29999;
40.
41.     /*
42.     Here we can choose which file and where to find it
43.     Wont work properly until this is changed to correct folder
44.      */
45.     String filePath = "/Users/mariusbl/Dropbox/Prosjekt/Bachelor/MariusDiv/";
46.     String movePushRehab = "move_push_rehab_v1.script";
47.
48.     /*
49.     Here we can set the acceleration and velocity for the UR
50.     Use values between 0.1 and 1.0
51.      */
52.     String a = "0.5";
53.     String v = "0.5";
54.
55.     /*
56.     Positions for UR
57.     These are tested and it is safe for the UR to move into these positions
58.      */
59.     double x1 = 0.1;
60.     double y1 = 0.4;
61.     double z1 = 0.7;
62.     double rx = 3.1415;
```

```
63.     double ry = 0.0;
64.     double rz = 0.0;
65.
66.     /*
67.     Example of two different ways to move the robot. These are the same waypoints
68.     using two different ways to determine where and how UR will move
69.      */
70.     String scriptMovement
71.             = "def scriptmovement():\n"
72.             + " textmsg(\"Starting script from String\")\n"
73.             + " movej(p[" + x1 + ", " + y1 + "," + z1 + "," + rx + "," + ry + "," +
    rz + "]," + v + "," + a + ")\n"
74.             + "end";
75.     String moveHome
76.             = "= def moveHome():\n"
77.             + " movej([4.19949825862993,-1.5307928265838344,0.7768978331470714,-
    0.816820601535465,-1.570841791732095,-0.512890719912271]," + a + "," + v + ")\n"
78.             + "end";
79.
80.     static GUI gui;
81.     // Create client server connection
82.     URsocket robotClient;
83.     Coordinates coordinatesPointer;
84.     String sensorfx, sensorfy, sensorfz;
85.     Label sensorfxLabel, sensorfyLabel, sensorfzLabel;
86.     Label coordinatesxLabel, coordinatesyLabel, coordinateszLabel,
87.             coordinatesaxLabel, coordinatesayLabel, coordinatesazLabel,
88.             coordinatesBaseLabel, coordinatesShoulderLabel, coordinatesElbowLabel,
89.             coordinatesWrist1Label, coordinatesWrist2Label, coordinatesWrist3Label;

90.     Label connectionLabel, disabledLabel, powerLabel, emergencyStopLabel,
91.             securityStopLabel, programRunningLabel, programPaused, notification;
92.
93.     LibraryURScript library = new LibraryURScript(serverPC, PORTPC);
94.
95.     Timeline timeline;
96.     String programChoice = "";
97.     ServerSocket mySS;
98.     Socket SS_accept;
99.     BufferedReader SS_BF;
100.
101.          public static void main(String[] args) {
102.              try {
103.                  gui = new GUI();
104.                  gui.robotClient = new URsocket(SERVERUR, PORTUR);
105.                  gui.coordinatesPointer = new Coordinates(gui.robotClient);
106.                  gui.mySS = new ServerSocket(PORTPC);
107.
108.                  // Create executor for the thread reading sensor data
109.                  ScheduledExecutorService scheduledThreadPool = Executors.newSched
    uledThreadPool(1);
110.                  scheduledThreadPool.scheduleAtFixedRate(gui.receiveSensor, 2000,
    500, TimeUnit.MILLISECONDS);
111.
112.                  // Create executor for the thread reading sensor data
113.                  ScheduledExecutorService scheduledThreadPool2 = Executors.newSche
    duledThreadPool(1);
114.                  scheduledThreadPool2.scheduleAtFixedRate(gui.receiveCoordinates,
    2000, 500, TimeUnit.MILLISECONDS);
115.
116.                  // Connect to UR
117.                  gui.robotClient.connectUR();
118.
119.                  // Launch the FX window
120.                  launch(args);
121.                  gui.mySS.close();
```

```
122.                } catch (Exception e) {
123.                    System.out.println("Error: " + e);
124.                }
125.            }
126.
127.            @Override
128.            public void start(Stage primaryStage) throws Exception {
129.                int lazyGridCounter = 1;
130.                int minWidth = 150;
131.                int maxWidth = 150;
132.                int minHeight = 50;
133.                int maxHeight = 50;
134.
135.                // set title
136.                primaryStage.setTitle("UR5-robot");
137.
138.                // Create gridpane
139.                GridPane grid = new GridPane();
140.                grid.setHgap(0);
141.                grid.setVgap(10);
142.                grid.setPadding(new Insets(0, 5, 0, 5));
143.
144.                // Create the buttons with set size and add to grid
145.                Button ballRehab = new Button("BallRehab");
146.                ballRehab.setMinSize(minWidth, minHeight);
147.                ballRehab.setMaxSize(maxWidth, maxHeight);
148.                grid.add(ballRehab, 0, lazyGridCounter++);
149.
150.                Button handShake = new Button("HandShake");
151.                handShake.setMinSize(minWidth, minHeight);
152.                handShake.setMaxSize(maxWidth, maxHeight);
153.                grid.add(handShake, 0, lazyGridCounter++);
154.
155.                Button handEye = new Button("EyeHand");
156.                handEye.setMinSize(minWidth, minHeight);
157.                handEye.setMaxSize(maxWidth, maxHeight);
158.                grid.add(handEye, 0, lazyGridCounter++);
159.
160.                Button handShakeDynamic = new Button("HandShakeDynamic");
161.                handShakeDynamic.setMinSize(minWidth, minHeight);
162.                handShakeDynamic.setMaxSize(maxWidth, maxHeight);
163.                grid.add(handShakeDynamic, 0, lazyGridCounter++);
164.
165.                Button connection = new Button("Connect");
166.                connection.setMinSize(minWidth, minHeight);
167.                connection.setMaxSize(maxWidth, maxHeight);
168.                grid.add(connection, 0, lazyGridCounter++);
169.
170.                // SplitPane
171.                SplitPane splitPane = new SplitPane();
172.                StackPane sp1 = new StackPane();
173.                sp1.getChildren().add(grid);
174.
175.                // Create labels to display the information to users
176.                Label sensorDataHeader = new Label("Sensor Readings: ");
177.                sensorfxLabel = new Label("null");
178.                sensorfyLabel = new Label("null");
179.                sensorfzLabel = new Label("null");
180.
181.                Label empty1 = new Label("");
182.                Label toolCenterPointPositionHeader = new Label("Robot TCP position:
    ");
183.                coordinatesxLabel = new Label("null");
184.                coordinatesyLabel = new Label("null");
185.                coordinateszLabel = new Label("null");
186.                Label empty2 = new Label("");
```

```
187.                coordinatesaxLabel = new Label("null");
188.                coordinatesayLabel = new Label("null");
189.                coordinatesazLabel = new Label("null");
190.                Label empty3 = new Label("");
191.                Label jointPositionsHeader = new Label("Joint Positions: ");
192.                coordinatesBaseLabel = new Label("null");
193.                coordinatesShoulderLabel = new Label("null");
194.                coordinatesElbowLabel = new Label("null");
195.                coordinatesWrist1Label = new Label("null");
196.                coordinatesWrist2Label = new Label("null");
197.                coordinatesWrist3Label = new Label("null");
198.
199.                // Create grid for displaying sensor and positions
200.                StackPane sp2 = new StackPane();
201.                GridPane grid2 = new GridPane();
202.                grid2.setHgap(0);
203.                grid2.setVgap(2);
204.                grid2.setPadding(new Insets(10, 2, 10, 2));
205.
206.                // Reset lazyGridCounter
207.                lazyGridCounter = 1;
208.                // Add labels to the grid
209.                grid2.add(sensorDataHeader, 0, lazyGridCounter++);
210.                grid2.add(sensorfxLabel, 0, lazyGridCounter++);
211.                grid2.add(sensorfyLabel, 0, lazyGridCounter++);
212.                grid2.add(sensorfzLabel, 0, lazyGridCounter++);
213.
214.                grid2.add(empty1, 0, lazyGridCounter++);
215.                grid2.add(toolCenterPointPositionHeader, 0, lazyGridCounter++);
216.                grid2.add(coordinatesxLabel, 0, lazyGridCounter++);
217.                grid2.add(coordinatesyLabel, 0, lazyGridCounter++);
218.                grid2.add(coordinateszLabel, 0, lazyGridCounter++);
219.
220.                grid2.add(empty2, 0, lazyGridCounter++);
221.                grid2.add(coordinatesaxLabel, 0, lazyGridCounter++);
222.                grid2.add(coordinatesayLabel, 0, lazyGridCounter++);
223.                grid2.add(coordinatesazLabel, 0, lazyGridCounter++);
224.
225.                grid2.add(empty3, 0, lazyGridCounter++);
226.                grid2.add(jointPositionsHeader, 0, lazyGridCounter++);
227.                grid2.add(coordinatesBaseLabel, 0, lazyGridCounter++);
228.                grid2.add(coordinatesShoulderLabel, 0, lazyGridCounter++);
229.                grid2.add(coordinatesElbowLabel, 0, lazyGridCounter++);
230.                grid2.add(coordinatesWrist1Label, 0, lazyGridCounter++);
231.                grid2.add(coordinatesWrist2Label, 0, lazyGridCounter++);
232.                grid2.add(coordinatesWrist3Label, 0, lazyGridCounter++);
233.
234.                sp2.getChildren().add(grid2);
235.
236.                // Create grid for robot status
237.                StackPane sp3 = new StackPane();
238.                GridPane grid3 = new GridPane();
239.                grid3.setHgap(0);
240.                grid3.setVgap(2);
241.                grid3.setPadding(new Insets(10, 2, 10, 2));
242.
243.                // Create the labels for robot status
244.                connectionLabel = new Label("null");
245.                disabledLabel = new Label("null");
246.                powerLabel = new Label("null");
247.                emergencyStopLabel = new Label("null");
248.                securityStopLabel = new Label("null");
249.                programRunningLabel = new Label("null");
250.                programPaused = new Label("null");
251.                notification = new Label("notifications here");
252.
```

```
253.                // add robot status labels to grid
254.                grid3.add(connectionLabel, 0, lazyGridCounter++);
255.                grid3.add(disabledLabel, 0, lazyGridCounter++);
256.                grid3.add(powerLabel, 0, lazyGridCounter++);
257.                grid3.add(emergencyStopLabel, 0, lazyGridCounter++);
258.                grid3.add(securityStopLabel, 0, lazyGridCounter++);
259.                grid3.add(programRunningLabel, 0, lazyGridCounter++);
260.                grid3.add(programPaused, 0, lazyGridCounter++);
261.                grid3.add(notification, 0, lazyGridCounter++);
262.
263.                sp3.getChildren().add(grid3);
264.
265.                // User has chosen ballRehab
266.                ballRehab.setOnAction(new EventHandler<ActionEvent>() {
267.                    @Override
268.                    public void handle(ActionEvent event) {
269.                        gui.robotClient.sendScriptToUR(gui.library.ballrehab(a, v));

270.                        try {
271.                            gui.SS_accept = gui.mySS.accept();
272.                            gui.SS_BF = new BufferedReader(new InputStreamReader(gui.
    SS_accept.getInputStream()));
273.                            gui.programChoice = "1";
274.                        } catch (Exception ex) {
275.                            System.out.println("EX " + ex);
276.                        }
277.                    }
278.                });
279.                // User has chosen handShake
280.                handShake.setOnAction(new EventHandler<ActionEvent>() {
281.                    @Override
282.                    public void handle(ActionEvent event) {
283.                        gui.robotClient.sendScriptToUR(gui.library.handshake(a, v));

284.                        try {
285.                            gui.SS_accept = gui.mySS.accept();
286.                            gui.SS_BF = new BufferedReader(new InputStreamReader(gui.
    SS_accept.getInputStream()));
287.                            gui.programChoice = "2";
288.                        } catch (Exception ex) {
289.                            System.out.println("EX " + ex);
290.                        }
291.                    }
292.                });
293.                // User has chosen handEyecoordination program
294.                handEye.setOnAction(new EventHandler<ActionEvent>() {
295.                    @Override
296.                    public void handle(ActionEvent event) {
297.                        gui.robotClient.sendScriptToUR(gui.library.handeye(a, v));
298.                        try {
299.                            gui.SS_accept = gui.mySS.accept();
300.                            gui.SS_BF = new BufferedReader(new InputStreamReader(gui.
    SS_accept.getInputStream()));
301.                            gui.programChoice = "3";
302.                        } catch (Exception ex) {
303.                            System.out.println("EX " + ex);
304.                        }
305.                    }
306.                });
307.                // User has chosen the handshake dynamic progam
308.                handShakeDynamic.setOnAction(new EventHandler<ActionEvent>() {
309.                    @Override
310.                    public void handle(ActionEvent event) {
311.                        gui.robotClient.sendScriptToUR(gui.library.handshakeDynamic(a
    , v));
312.                        try {
```

```
313.                            gui.SS_accept = gui.mySS.accept();
314.                            gui.SS_BF = new BufferedReader(new InputStreamReader(gui.
    SS_accept.getInputStream()));
315.                            gui.programChoice = "4";
316.                        } catch (Exception ex) {
317.                            System.out.println("EX " + ex);
318.                        }
319.                    }
320.                });
321.                // user has chosen to connect, without sending a program
322.                connection.setOnAction(new EventHandler<ActionEvent>() {
323.                    @Override
324.                    public void handle(ActionEvent event) {
325.                        try {
326.                            gui.SS_accept = gui.mySS.accept();
327.                            gui.SS_BF = new BufferedReader(new InputStreamReader(gui.
    SS_accept.getInputStream()));
328.                            gui.programChoice = "5";
329.                        } catch (Exception ex) {
330.                            System.out.println("EX " + ex);
331.                        }
332.                    }
333.                });
334.                // timeline for updating the labels
335.                EventHandler<ActionEvent> onFinished = new EventHandler<ActionEvent>(
    ) {
336.                    @Override
337.                    public void handle(ActionEvent t) {
338.                        // Update values for sensor data
339.                        sensorfxLabel.setText(gui.sensorfx);
340.                        sensorfyLabel.setText(gui.sensorfy);
341.                        sensorfzLabel.setText(gui.sensorfz);
342.
343.                        // Update values for coordinates
344.                        coordinatesxLabel.setText("X: \t\t"
345.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getX1norm()));
346.                        coordinatesyLabel.setText("Y: \t\t"
347.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getY1norm()));
348.                        coordinateszLabel.setText("Z: \t\t"
349.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getZ1norm()));
350.                        coordinatesaxLabel.setText("aX: \t\t"
351.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getRxnorm()));
352.                        coordinatesayLabel.setText("aY: \t\t"
353.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getRynorm()));
354.                        coordinatesazLabel.setText("aZ: \t\t"
355.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getRznorm()));
356.                        coordinatesBaseLabel.setText("Base: \t\t"
357.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getX1p()));
358.                        coordinatesShoulderLabel.setText("Shoulder: \t"
359.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getY1p()));
360.                        coordinatesElbowLabel.setText("Elbow: \t\t"
361.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getZ1p()));
362.                        coordinatesWrist1Label.setText("Wrist1: \t\t"
363.                                + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getRxp()));
364.                        coordinatesWrist2Label.setText("Wrist2: \t\t"
```

```
365.                                 + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getRyp()));
366.                         coordinatesWrist3Label.setText("Wrist3: \t\t"
367.                                 + new DecimalFormat("##.####").format(gui.coordinates
    Pointer.getRzp()));
368.
369.                         // Update status
370.                         connectionLabel.setText("UR: \t\t\t\t" + gui.coordinatesPoint
    er.getConnection());
371.                         disabledLabel.setText("UR: \t\t\t\t" + gui.coordinatesPointer
    .getDisabled());
372.                         powerLabel.setText("Power: \t\t\t" + gui.coordinatesPointer.g
    etPower());
373.                         emergencyStopLabel.setText("Emercency stop: \t" + gui.coordin
    atesPointer.getEmergencyStop());
374.                         securityStopLabel.setText("Security stop: \t" + gui.coordinat
    esPointer.getSecurityStop());
375.                         programRunningLabel.setText("Program: \t\t" + gui.coordinates
    Pointer.getProgramRunning());
376.                         programPaused.setText("Program: \t\t" + gui.coordinatesPointe
    r.getProgramPaused());
377.
378.                         // Update Message to user
379.                         notification.setText("");
380.                     }
381.                 };
382.
383.                 // Create Timeline
384.                 timeline = new Timeline();
385.                 timeline.setCycleCount(Timeline.INDEFINITE);
386.                 Duration duration = Duration.seconds(0.5);
387.                 KeyFrame keyFrame = new KeyFrame(duration, onFinished);
388.                 timeline.getKeyFrames().add(keyFrame);
389.                 timeline.play();
390.
391.                 // Set initial positions of splitpane
392.                 splitPane.getItems().addAll(sp1, sp2, sp3);
393.                 splitPane.setDividerPositions(0.2, 0.60, 0.20);
394.                 primaryStage.setScene(new Scene(splitPane, 700, 500));
395.                 // show path to CSS and add to splitpane
396.                 String cssPath = this.getClass().getResource("application.css").toExt
    ernalForm();
397.                 splitPane.getStylesheets().add(cssPath);
398.                 // Show the primary Stage
399.                 primaryStage.show();
400.             }
401.
402.             // Create pointer for the thread which receives sensordata
403.             Runnable receiveSensor = new Runnable() {
404.                 @Override
405.                 public void run() {
406.                     gui.ReceiveMode();
407.                 }
408.             };
409.             // Create pointer for the thread which reads position and status stream
410.             Runnable receiveCoordinates = new Runnable() {
411.                 @Override
412.                 public void run() {
413.                     coordinatesPointer.getData();
414.                 }
415.             };
416.
417.             // The thread for reading and decoding sensordata
418.             void ReceiveMode() {
419.                 if (!((gui.programChoice).equals(""))) {
420.                     try {
```

```
421.                        char in;
422.                        char last = '0';
423.                        String line = "";
424.                        String fx = "";
425.                        String fy = "";
426.                        String fz = "";
427.                        boolean done = false;
428.
429.                        /*
430.                        Sensordata from robot will look like this:
431.                        Fx: number;
432.                        Fy: number;
433.                        Fz: number;
434.                        ;
435.                        we will use ; to save lines and double ; to mark end of input
    .
436.                         */
437.                        do {
438.                            in = (char) (gui.SS_BF).read();
439.                            line = line + in;
440.                            if (in == ';') {
441.                                if (fx.equals("")) {
442.                                    fx = line + "\n";
443.                                } else if (fy.equals("")) {
444.                                    fy = line + "\n";
445.                                } else if (last != ';') {
446.                                    fz = line + "\n";
447.                                }
448.                                line = "";
449.                            }
450.                            // end of sensordata block ';;'
451.                            if (in == ';' && last == ';') {
452.                                done = true;
453.                            }
454.                            last = in; // save last input to determine where end of s
    ensordata block is
455.                        } while (!done);
456.                        // update sensordata to the static variables
457.                        gui.sensorfx = fx;
458.                        gui.sensorfy = fy;
459.                        gui.sensorfz = fz;
460.                    } catch (Exception ex) {
461.                        System.out.println("Executor Error: " + ex);
462.                    }
463.                }
464.            }
465.        }
```

Vedlegg 7 URsocket.java

```java
package GUI;

import java.io.DataOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Paths;

/**
 *
 * @author Marius Blom <mariusbl@stud.hisf.no>
 */
public class URsocket {

    /*
    Declaration of variables
     */
    String serverName;
    int port;
    Socket client;
    DataOutputStream out;
    OutputStream outToServer;

    /*
    Constructor to recieve address and port
     */
    public URsocket(String serverName, int port) {

        this.serverName = serverName;
        this.port = port;
    }

    /*
    Method to connect to the UR
     */
    public void connectUR() {
        try {
            /*
            Establish socket connection to the UR
             */
            System.out.println("Connecting to " + serverName + " on port " + port);

            client = new Socket(serverName, port);
            client.setSoTimeout(5000);
            System.out.println("Connected to " + client.getRemoteSocketAddress());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void sendScriptToUR(String scriptToWrite) {
        /*
            Display what we are sending to the server
         */
        System.out.println(scriptToWrite);

        try {
            /*
            Create stream to read from file and send data to UR
             */
```

```
62.             outToServer = client.getOutputStream();
63.             out = new DataOutputStream(outToServer);
64.             /*
65.             Write script to server
66.              */
67.             out.write(scriptToWrite.getBytes());
68.             out.write('\n'); // Always newline after a script
69.         } catch (Exception e) {
70.             e.printStackTrace();
71.         }
72.     }
73.
74.     /*
75.     Method to read from script file
76.      */
77.     static String readFile(String path) throws IOException {
78.         byte[] encoded = Files.readAllBytes(Paths.get(path));
79.         return new String(encoded, Charset.defaultCharset());
80.     }
81. }
```

Vedlegg 8 LibraryURScript.java

```
1.  package GUI;
2.
3.  /**
4.   * This class contains all the different URScript we use for the UR
5.   *
6.   * Do not change anything here, unless you have some basic knowledge with
7.   * URScript
8.   *
9.   * @author Marius Blom <mariusbl@stud.hisf.no>
10.  */
11. public class LibraryURScript {
12.
13.     String payloadUR = "1.985"; // in kilograms ( 1.985kg)
14.     String serverPC;
15.     int portPC;
16.
17.     public LibraryURScript(String serverPC, int portPC) {
18.         this.serverPC = serverPC;
19.         this.portPC = portPC;
20.     }
21.
22.     public String threadSensorData() {
23.         /*
24.         This method will read data from the torque sensor and send it back
25.         to a socket server in Java. This script need to be combined with other
26.         scripts and run it will then run in a seperate thread.
27.         Add the following line to the script you want to run sensordata to.
28.         + "join threadId_SensorDataThread\n"
29.
30.         The thread will recieve the serveraddress and serverport from URmain.java
31.          */
32.         String threadSensorData
33.                 = " thread SensorDataThread(): \n"
34.                 + "     socket_close(\"stream\") #Close the stream \"stream\"\n"
35.                 + "     sleep(0.1)\n"
36.                 + "     socket_open(\"127.0.0.1\",63351,\"stream\")\n"
37.                 + "     sleep(3)\n"
38.                 + "     socket_close(\"pc_connect\") #Close the stream \"pc_connect\
    "\n"
39.                 + "             global var_1 =   False  \n" // new
40.                 + "             while (var_1 ==  False  ):\n"// new
41.                 + "       global var_1 = socket_open(\"" + serverPC + "\"," + portPC
     + ",\"pc_connect\")\n" // new
42.                 + "             end\n"// new
43.                 + "     socket_send_string(\"StartOfInput\", \"pc_connect\")\n"
44.                 + "     socket_send_string(\";\", \"pc_connect\")\n"
45.                 + "     while True:\n" // new
46.                 + "         global sensor_data = socket_read_ascii_float(6,\"stream\
    ")\n"
47.                 + "         varmsg(\"sensor_data\",sensor_data)\n"
48.                 + "         if (sensor_data[0]  >=6): #IfSensorDataArrayIsFilledThen
    Run\n"
49.                 + "             global Fx = sensor_data[1]#Set input sensor data to
    appropiate variables\n"
50.                 + "             global Fy = sensor_data[2]\n"
51.                 + "             global Fz = sensor_data[3]\n"
52.                 + "             global Mx = sensor_data[4]\n"
53.                 + "             global My = sensor_data[5]\n"
54.                 + "             global Mz = sensor_data[6]\n"
55.                 + "         else: #IfItIsNotThenSetThemAllToZer0\n"
56.                 + "             global Fx = 0.0 # set the variables to zer0\n"
57.                 + "             global Fy = 0.0\n"
```

```
58.                 + "             global Fz = 0.0\n"
59.                 + "             global Mx = 0.0\n"
60.                 + "             global My = 0.0\n"
61.                 + "             global Mz = 0.0\n"
62.                 + "         end #EndIf\n"
63.                 + "         socket_send_string(\";\", \"pc_connect\")#New Line in ja
    va, to seperate the informationblocks\n"
64.                 + "         socket_send_string(\"Fx: \", \"pc_connect\")#SendVariabl
    es\n"
65.                 + "         socket_send_string(Fx, \"pc_connect\")#SendVariables\n"

66.                 + "         socket_send_string(\";\", \"pc_connect\")#NewLineInJava\
    n"
67.                 + "         socket_send_string(\"Fy: \", \"pc_connect\")\n"
68.                 + "         socket_send_string(Fy, \"pc_connect\")\n"
69.                 + "         socket_send_string(\";\", \"pc_connect\")\n"
70.                 + "         socket_send_string(\"Fz: \", \"pc_connect\")\n"
71.                 + "         socket_send_string(Fz, \"pc_connect\")\n"
72.                 + "         socket_send_string(\";\", \"pc_connect\")\n"
73.                 + "         sleep(0.5)\n"
74.                 + "               global var_1 =   False\n"
75.                 + "     end #EndWhile\n"
76.                 + " end #EndThread\n"
77.                 + " threadId_SensorDataThread = run SensorDataThread() #RunThread\n"
    ;
78.         return threadSensorData;
79.     }
80.
81.     public String initialization() {
82.         String initalization
83.                 = "  modbus_add_signal(\"0.0.0.0\", 255, 0, 3, \"Modbus_1\")\n"
84.                 + "  modbus_set_signal_update_frequency(\"Modbus_1\", 10)\n"
85.                 + "  set_analog_inputrange(0, 0)\n"
86.                 + "  set_analog_inputrange(1, 0)\n"
87.                 + "  set_analog_inputrange(2, 0)\n"
88.                 + "  set_analog_inputrange(3, 0)\n"
89.                 + "  set_analog_outputdomain(0, 0)\n"
90.                 + "  set_analog_outputdomain(1, 0)\n"
91.                 + "  set_tool_voltage(0)\n"
92.                 + "  set_runstate_outputs([])\n"
93.                 + "  modbus_set_runstate_dependent_choice(\"Modbus_1\",0)\n"
94.                 + "  set_payload(" + payloadUR + ")\n"
95.                 + "  set_gravity([0.0, 0.0, 9.82])\n"
96.                 + "  Base=p[0.0,0.0,0.0,0.0,0.0,0.0]\n"
97.                 + "  Fx=0.0\n"
98.                 + "  Fy=0.0\n"
99.                 + "  Fz=0.0\n"
100.                    + "  Mx=0.0\n"
101.                    + "  My=0.0\n"
102.                    + "  Mz=0.0\n";
103.             return initalization + threadSensorData();
104.         }
105.
106.         public String ballrehab(String inA, String inV) {
107.             /*
108.         This method will simulate that of a ball being pulled out of the hand.
109.         The UR will try to pull a ball from the user, where the FT-
    150 force sensor
110.         will detect how much force the user is pulling with. The UR will then
111.         increase or decrease the velocity it will try to pull away with. If the
112.         force sensor does not detect any force (nobody is holding the ball) then
    it
113.         will push the ball towards the user until it detect force again.
114.              */
115.             String scriptName = "def ballrehab_v2():\n";
116.             String script
```

```
117.                        = "  def push_z_axis():\n"
118.                        + "    thread Thread_while_47():\n"
119.                        + "      while (True):\n"
120.                        + "        global position_increm = get_forward_kin()\n"
121.                        + "        position_increm[0] = position_increm[0]-
    x_increment\n"
122.                        + "        movel(position_increm,1.2,0.00001)\n"
123.                        + "      end\n"
124.                        + "    end\n"
125.                        + "    if (norm(Fz) <Fz_max):\n"
126.                        + "      global thread_handler_47=run Thread_while_47()\n"
127.                        + "      while (norm(Fz) <Fz_max):\n"
128.                        + "        sync()\n"
129.                        + "      end\n"
130.                        + "      kill thread_handler_47\n"
131.                        + "    end\n"
132.                        + "  end\n"
133.                        + "  def rq_set_zero():\n"
134.                        + "     if(socket_open(\"127.0.0.1\",63350,\"acc\")):\n"
135.                        + "         socket_send_string(\"SET ZRO\",\"acc\")\n"
136.                        + "         sleep(0.1)\n"
137.                        + "         socket_close(\"acc\")\n"
138.                        + "     end\n"
139.                        + "    end\n"
140.                        + "  global Fx = 0.0\n"
141.                        + "  global Fy = 0.0\n"
142.                        + "  global Fz = 0.0\n"
143.                        + "  global Mx = 0.0\n"
144.                        + "  global My = 0.0\n"
145.                        + "  global Mz = 0.0\n"
146.                        + "  global Fx_max = 12\n"
147.                        + "  global Fz_max = 12\n"
148.                        + "  global Fz_tolerance = 5\n"
149.                        + "  global x_increment = 0.002\n"
150.                        + "  global z_increment = 0.004\n"
151.                        + "  global y_increment = 0.004\n"
152.                        + "  thread Thread_1():\n"
153.                        + "    while True:\n"
154.                        + "      while(1):\n"
155.                        + "         listofstuff = socket_read_ascii_float(6,\"stream\
    ")\n"
156.                        + "         if(listofstuff[0] != 0):\n"
157.                        + "             Fx = listofstuff[1]\n"
158.                        + "             Fy = listofstuff[2]\n"
159.                        + "             Fz = listofstuff[3]\n"
160.                        + "             Mx = listofstuff[4]\n"
161.                        + "             My = listofstuff[5]\n"
162.                        + "             Mz = listofstuff[6]\n"
163.                        + "         end\n"
164.                        + "#        textmsg(Fz)\n"
165.                        + "      end\n"
166.                        + "    end\n"
167.                        + "  end\n"
168.                        + "  threadId_Thread_1 = run Thread_1()\n"
169.                        + "    movej([3.2653818126748063, -
    0.5004661537747839, 1.2609107063460065, -3.216423783749357, -
    1.5145955848326498, 2.38019390734757], a=1.0, v=0.5)\n"
170.                        + "  sleep(0.3)\n"
171.                        + "  rq_set_zero()\n"
172.                        + "  push_z_axis()\n"
173.                        + "  sleep(0.5)\n"
174.                        + "  global x_distance = 0\n"
175.                        + "    while (norm(x_distance) < 5):\n"
176.                        + "         position_increm = get_forward_kin()\n"
177.                        + "#        global x_distance = x_distance+x_increment\n"
178.                        + "    if (norm(Fz) > 10) and (norm(Fz) <= 20):\n"
```

```
                        + "        speedl([(-
    0.001),0.0,0.0,0.0,0.0,0.0], 0.5, 0.008)\n"
                        + "        textmsg(\"norm(Fz) > 10) and (norm(Fz) < 20\")\n"

                        + "        textmsg(\"Force after contact Fz = \", Fz)\n"
                        + "        global x_distance = x_distance+0.001\n"
                        + "        textmsg(\"x_distance\", x_distance)\n"
                        + "    elif (norm(Fz) > 20) and (norm(Fz) <= 35):\n"
                        + "        textmsg(\"norm(Fz) > 20) and (norm(Fz) < 35\")\n"

                        + "        textmsg(\"Force after contact Fz = \", Fz)\n"
                        + "        global x_distance = x_distance+0.003\n"
                        + "        textmsg(\"x_distance\", x_distance)\n"
                        + "        speedl([(-
    0.003),0.0,0.0,0.0,0.0,0.0], 0.5, 0.008)\n"
                        + "    elif (norm(Fz) > 35) and (norm(Fz) <= 40):\n"
                        + "        textmsg(\"norm(Fz) > 20) and (norm(Fz) < 40\")\n"

                        + "        textmsg(\"Force after contact Fz = \", Fz)\n"
                        + "        global x_distance = x_distance+0.006\n"
                        + "        textmsg(\"x_distance\", x_distance)\n"
                        + "        speedl([(-
    0.006),0.0,0.0,0.0,0.0,0.0], 0.5, 0.008)\n"
                        + "    elif (norm(Fz) > 40):\n"
                        + "        textmsg(\"norm(Fz) > 40\")\n"
                        + "        textmsg(\"Force after contact Fz = \", Fz)\n"
                        + "        global x_distance = x_distance+0.009\n"
                        + "        textmsg(\"x_distance\", x_distance)\n"
                        + "        speedl([(-
    0.009),0.0,0.0,0.0,0.0,0.0], 0.5, 0.008)\n"
                        + "    else:\n"
                        + "        textmsg(\"DROPPED\")\n"
                        + "        textmsg(\"Force after contact Fz = \", Fz)\n"
                        + "        global x_distance = x_distance-0.002\n"
                        + "        textmsg(\"x_distance\", x_distance)\n"
                        + "        # 'DROPPED'\n"
                        + "        speedl([(0.002),0.0,0.0,0.0,0.0,0.0], 0.5, 0.008)\
    n"
                        + "    end\n"
                        + "        textmsg(\"x_distance FINISHED?\", x_distance)\n"
                        + "    end\n"
                        + "    movej([3.360639076715726, -
    0.750901013913703, 1.8627477276646113, -3.7658198596698145, -
    1.665923337465497, 2.5039267622723], a=0.1, v=0.2)\n"
                        + "    movej([3.560639076715726, -
    0.950901013913703, 2.2627477276646113, -3.7658198596698145, -
    1.665923337465497, 2.5039267622723], a=0.2, v=0.5)\n"
                        + "join threadId_SensorDataThread\n"
                        + "end";
                return scriptName + initialization() + script;
            }

            public String handshake(String inA, String inV) {
                /*
            This method will simulate the feeling of a handshake. The UR will wait
            until it detect force against the FT-
    150 force sensor. Then it will go through
            a couple of predetermined positions to simulate a handshake
                 */
                String scriptName = "def handshake_v2():\n";
                String script
                        = "  def push_z_axis():\n"
                        + "    thread Thread_while_37():\n"
                        + "      while (True):\n"
                        + "        global position_increm = get_forward_kin()\n"
                        + "        position_increm[2] = position_increm[2]-.005\n"
```

```
232.                         + "        movel(position_increm,2.2,0.0001,0,0.0001)\n"
233.                         + "      end\n"
234.                         + "    end\n"
235.                         + "    if (norm(Fx)<Fmax):\n"
236.                         + "      global thread_handler_37=run Thread_while_37()\n"
237.                         + "      while (norm(Fx)<Fmax):\n"
238.                         + "        sync()\n"
239.                         + "      end\n"
240.                         + "      kill thread_handler_37\n"
241.                         + "    end\n"
242.                         + "  end\n"
243.                         + "  \n"
244.                         + "  def rq_set_zero():\n"
245.                         + "     if(socket_open(\"127.0.0.1\",63350,\"acc\")): \n"
246.                         + "         socket_send_string(\"SET ZRO\",\"acc\")\n"
247.                         + "         sleep(0.1)\n"
248.                         + "         socket_close(\"acc\")\n"
249.                         + "     end\n"
250.                         + "    end\n"
251.                         + "\n"
252.                         + "  sleep(1.0)\n"
253.                         + "  rq_set_zero()\n"
254.                         + "  global Fx = 0.0\n"
255.                         + "  global Fy = 0.0\n"
256.                         + "  global Fz = 0.0\n"
257.                         + "  global Mx = 0.0\n"
258.                         + "  global My = 0.0\n"
259.                         + "  global Mz = 0.0\n"
260.                         + "  global Fmax = 10\n" //change this if you want to apply m
    ore force
261.                         + "  thread Thread_1():\n"
262.                         + "    while True:\n"
263.                         + "      while(1):\n"
264.                         + "         listofstuff = socket_read_ascii_float(6,\"stream\
    ")\n"
265.                         + "         if(listofstuff[0] != 0):\n"
266.                         + "             Fx = listofstuff[1]\n"
267.                         + "             Fy = listofstuff[2]\n"
268.                         + "             Fz = listofstuff[3]\n"
269.                         + "             Mx = listofstuff[4]\n"
270.                         + "             My = listofstuff[5]\n"
271.                         + "             Mz = listofstuff[6]\n"
272.                         + "         end\n"
273.                         + "      end\n"
274.                         + "    end\n"
275.                         + "  end\n"
276.                         + "  threadId_Thread_1 = run Thread_1()\n"
277.                         + "  while (True):\n"
278.                         + "    movej([3.200080809962689, -
    1.994541626691599, 2.663925097733759, -2.7613141184321828, -
    1.6226326173531431, 1.6340247019898373], a=1.3962634015954636, v=1.0471975511965976)
    \n"
279.                         + "    sleep(0.5)\n"
280.                         + "    movej([3.1886013797851094, -
    1.059109353239251, 2.0996290640173756, -4.214171099406416, -
    1.6178153424361694, 1.6320974468773706], a=1.3962634015954636, v=1.0471975511965976)
    \n"
281.                         + "    sleep(0.5)\n"
282.                         + "    push_z_axis()\n"
283.                         + "    textmsg(\"Force after contact = \", Fz)\n"
284.                         + "    textmsg(\"Force 1 sec later = \", Fz)\n"
285.                         + "    sleep(0.3)\n"
286.                         + "    movel(pose_trans(Base, p[0.46371057826150597,0.1227582
    4616358655,0.3278104951907625,0.058349744030833324,1.445227432811873,0.0462211698541
    2137]), a=1.2, v=0.25)\n"
```

```
287.                        + "    movel(pose_trans(Base, p[0.5055859434607656,0.12467656
    626114328,0.25855195309048457,0.05609212946852178,1.7284970742543104,0.0285145853729
    03946]), a=1.2, v=0.25)\n"
288.                        + "    movel(pose_trans(Base, p[0.46371057826150597,0.1227582
    4616358655,0.3278104951907625,0.058349744030833324,1.445227432811873,0.0462211698541
    2137]), a=1.2, v=0.25)\n"
289.                        + "    movel(pose_trans(Base, p[0.5055859434607656,0.12467656
    626114328,0.25855195309048457,0.05609212946852178,1.7284970742543104,0.0285145853729
    03946]), a=1.2, v=0.25)\n"
290.                        + "    movel(pose_trans(Base, p[0.46371057826150597,0.1227582
    4616358655,0.3278104951907625,0.058349744030833324,1.445227432811873,0.0462211698541
    2137]), a=1.2, v=0.25)\n"
291.                        + "    movel(pose_trans(Base, p[0.5055859434607656,0.12467656
    626114328,0.25855195309048457,0.05609212946852178,1.7284970742543104,0.0285145853729
    03946]), a=1.2, v=0.25)\n"
292.                        + "    sleep(1.0)\n"
293.                        + "    sleep(0.5)\n"
294.                        + "  end\n"
295.                        + "join threadId_SensorDataThread\n"
296.                        + "end";
297.                return scriptName + initialization() + script;
298.            }
299.
300.            public String handeye(String inA, String inV) {
301.                /*
302.            This method will have the user try to move his hand to the UR, it will th
    en
303.            detect force through the FT-
    150 force sensor when the user touch the tip of
304.            the UR. It will then move to a different predetermied position and the us
    er
305.            will have to touch the tip of the UR once more. This is to train hand eye

306.            coordination of the user.
307.                 */
308.                String scriptName = "def handeye():\n";
309.                String script
310.                        = "  def push_z_axis():\n"
311.                        + "    thread Thread_while_51():\n"
312.                        + "      while (True):\n"
313.                        + "        global position_increm = get_forward_kin()\n"
314.                        + "        varmsg(\"position_increm\",position_increm)\n"
315.                        + "        position_increm[2] = position_increm[2]-.005\n"
316.                        + "        movel(position_increm,2.2,0.0001,0,0.0001)\n"
317.                        + "      end\n"
318.                        + "    end\n"
319.                        + "    if (norm(Fz)<Fmax):\n"
320.                        + "      global thread_handler_51=run Thread_while_51()\n"
321.                        + "      while (norm(Fz)<Fmax):\n"
322.                        + "        sync()\n"
323.                        + "      end\n"
324.                        + "      kill thread_handler_51\n"
325.                        + "    end\n"
326.                        + "  end\n"
327.                        + "  def rq_set_zero():\n"
328.                        + "     if(socket_open(\"127.0.0.1\",63350,\"acc\")): \n"
329.                        + "         socket_send_string(\"SET ZRO\",\"acc\")\n"
330.                        + "         sleep(0.1)\n"
331.                        + "         socket_close(\"acc\")\n"
332.                        + "     end\n"
333.                        + "    end\n"
334.                        + "  global Fx = 0.0\n"
335.                        + "  global Fy = 0.0\n"
336.                        + "  global Fz = 0.0\n"
337.                        + "  global Mx = 0.0\n"
338.                        + "  global My = 0.0\n"
```

```
                        + "  global Mz = 0.0\n"
                        + "  global Fmax = 8\n"
                        + "  thread Thread_1():\n"
                        + "    while True:\n"
                        + "      while(1):\n"
                        + "         listofstuff = socket_read_ascii_float(6,\"stream\
    ")\n"
                        + "         if(listofstuff[0] != 0):\n"
                        + "             Fx = listofstuff[1]\n"
                        + "             Fy = listofstuff[2]\n"
                        + "             Fz = listofstuff[3]\n"
                        + "             Mx = listofstuff[4]\n"
                        + "             My = listofstuff[5]\n"
                        + "             Mz = listofstuff[6]\n"
                        + "         end\n"
                        + "      end\n"
                        + "    end\n"
                        + "  end\n"
                        + "  threadId_Thread_1 = run Thread_1()\n"
                        + "    movej([2.3597812121119945, -
    2.6554155312661014, 2.0940045125362365, -2.2006856657739915, -
    1.6548339300696657, 1.5697192686207784]," + inA + "," + inV + ")\n"
                        + "    sleep(0.5)\n"
                        + "   rq_set_zero()\n"
                        + "  while (True):\n"
                        + "    movej([2.3597812121119945, -
    2.6554155312661014, 2.0940045125362365, -2.2006856657739915, -
    1.6548339300696657, 1.5697192686207784]," + inA + "," + inV + ")\n"
                        + "    sleep(0.5)\n"
                        + "    push_z_axis()\n"
                        + "    sleep(0.5)\n"
                        + "    movej([2.971706895489564, -
    2.075837808367967, 2.2925625958333153, -3.350395146350983, -
    1.9576173017773657, 1.569395215179323]," + inA + "," + inV + ")\n"
                        + "    sleep(0.5)\n"
                        + "    push_z_axis()\n"
                        + "    sleep(0.5)\n"
                        + "    textmsg(\"FORTSOMFAEN 1 \")\n"
                        + "    movel(pose_trans(Base, p[0.08553227176327999,-
    0.3129441826248693,0.3374102646331472,0.6726456817459714,1.3388897101437733,-
    0.8455655847392162])," + inA + "," + inV + ")\n"
                        + "    sleep(0.5)\n"
                        + "    push_z_axis()\n"
                        + "    sleep(0.5)\n"
                        + "    textmsg(\"FORTSOMFAEN 1 \")\n"
                        + "    movel(pose_trans(Base, p[0.12579963594747545,-
    0.23599038250988408,0.7361827434885464,0.6643994012672705,1.4551519891084799,-
    0.4900663886874872])," + inA + "," + inV + ")\n"
                        + "    sleep(0.5)\n"
                        + "    push_z_axis()\n"
                        + "    sleep(0.5)\n"
                        + "    movel(pose_trans(Base, p[0.3663171643410643,0.00855554
    6193374276,0.6897456833543845,0.35771901560902175,1.5716548823160623,-
    0.1967816952526886])," + inA + "," + inV + ")\n"
                        + "    sleep(0.5)\n"
                        + "    push_z_axis()\n"
                        + "    sleep(0.5)\n"
                        + "    movel(pose_trans(Base, p[0.2769653515359639,0.02198804
    0857504797,0.38566288939025917,0.4101252280525745,1.4131672553666632,-
    0.26604284846445503])," + inA + "," + inV + ")\n"
                        + "    sleep(0.5)\n"
                        + "    push_z_axis()\n"
                        + "    sleep(0.5)\n"
                        + "    movel(pose_trans(Base, p[0.196367583520833,0.034782408
    671321555,0.3425156627555773,0.26793253121640853,1.1149430927142523,-
    0.26620128539673843])," + inA + "," + inV + ")\n"
```

```
                        + "    sleep(0.5)\n"
                        + "    push_z_axis()\n"
                        + "    sleep(0.5)\n"
                        + "  end\n"
                        + "    join threadId_SensorDataThread\n"
                        + "end";
                return scriptName + initialization() + script;

            }

            public String handshakeDynamic(String inA, String inV) {
                /*
            This method will simulate the feeling of a handshake. The UR will wait
            until it detect force against the FT-150 force sensor. Then it will follow
                the motion of the user
                 */
                String scriptName = "def handshake_v2():\n";
                String script
                        = "  def push_z_axis():\n"
                        + "    thread Thread_while_47():\n"
                        + "      while (True):\n"
                        + "        global position_increm = get_forward_kin()\n"
                        + "        position_increm[0] = position_increm[0]-x_increment\n"
                        + "        movel(position_increm,1.2,0.00001)\n"
                        + "      end\n"
                        + "    end\n"
                        + "    if (norm(Fy) <Fz_max):\n"
                        + "      global thread_handler_47=run Thread_while_47()\n"
                        + "      while (norm(Fz) <Fz_max):\n"
                        + "        sync()\n"
                        + "      end\n"
                        + "      kill thread_handler_47\n"
                        + "    end\n"
                        + "  end\n"
                        + "  def rq_set_zero():\n"
                        + "     if(socket_open(\"127.0.0.1\",63350,\"acc\")):\n"
                        + "         socket_send_string(\"SET ZRO\",\"acc\")\n"
                        + "         sleep(0.1)\n"
                        + "         socket_close(\"acc\")\n"
                        + "     end\n"
                        + "    end\n"
                        + "  global Fx = 0.0\n"
                        + "  global Fy = 0.0\n"
                        + "  global Fz = 0.0\n"
                        + "  global Mx = 0.0\n"
                        + "  global My = 0.0\n"
                        + "  global Mz = 0.0\n"
                        + "  global Fx_max = 12\n"
                        + "  global Fz_max = 7\n"
                        + "  global Fz_tolerance = 5\n"
                        + "  global x_increment = 0.002\n"
                        + "  global z_increment = 0.004\n"
                        + "  global y_increment = 0.004\n"
                        + "  thread Thread_1():\n"
                        + "    while True:\n"
                        + "      while(1):\n"
                        + "         listofstuff = socket_read_ascii_float(6,\"stream\")\n"
                        + "         if(listofstuff[0] != 0):\n"
                        + "             Fx = listofstuff[1]\n"
                        + "             Fy = listofstuff[2]\n"
                        + "             Fz = listofstuff[3]\n"
                        + "             Mx = listofstuff[4]\n"
                        + "             My = listofstuff[5]\n"
```

```
                        + "             Mz = listofstuff[6]\n"
                        + "             sleep(0.2)\n"
                        + "          end\n"
                        + "#         textmsg(Fz)\n"
                        + "       end\n"
                        + "    end\n"
                        + "  end\n"
                        + "  threadId_Thread_1 = run Thread_1()\n"
                        + "    movej([3.1886013797851094, -
    1.059109353239251, 2.0996290640173756, -4.214171099406416, -
    1.6178153424361694, 1.6320974468773706], a=1.3962634015954636, v=1.0471975511965976)
    \n"
                        + "  sleep(0.3)\n"
                        + "  rq_set_zero()\n"
                        + "  push_z_axis()\n"
                        + "  sleep(0.5)\n"
                        + "  global x_distance = 0\n"
                        + "  global timeout = 0\n"
                        + "    while (norm(x_distance) < 10):\n"
                        + "         position_increm = get_forward_kin()\n"
                        + "#        global x_distance = x_distance+x_increment\n"
                        + "    if (Fy > 5): # press ned = positiv verdi\n"
                        + "        speedj([0,0.1,0.05,0.1,0,0], 1.3, 0.008) # positiv
     beveger seg opp\n"
                        + "        global x_distance = x_distance-0.1\n"
                        + "        global timeout = 0\n"
                        + "    elif (Fy < (-5)): # press opp = negativ verdi\n"
                        + "        global x_distance = x_distance+0.1\n"
                        + "        speedj([0,(-0.1),(-0.05),(-
    0.1),0,0], 1.3, 0.008) # positiv beveger seg opp\n"
                        + "        global timeout = 0\n"
                        + "    else:\n"
                        + "        # 'DROPPED'\n"
                        + "        speedl([(0.001),0.0,0.0,0.0,0.0,0.0], 0.5, 0.008)\
    n"
                        + "        if(timeout >= 100):\n"
                        + "         global x_distance = 10\n"
                        + "        end        \n"
                        + "        global timeout = timeout+1\n"
                        + "    end\n"
                        + "    end\n"
                        + "    sleep(0.5)\n"
                        + "    movej([3.360639076715726, -
    0.750901013913703, 1.8627477276646113, -3.7658198596698145, -
    1.665923337465497, 2.5039267622723], a=0.1, v=0.2)\n"
                        + "    movej([3.560639076715726, -
    0.950901013913703, 2.2627477276646113, -3.7658198596698145, -
    1.665923337465497, 2.5039267622723], a=0.2, v=0.5)\n"
                        + "    join threadId_SensorDataThread\n"
                        + "end";
                return scriptName + initialization() + script;
            }
        }
```

Vedlegg 9 Coordinates.java

```java
package GUI;

import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.nio.ByteBuffer;

/**
 *
 * @author Marius Blom <mariusbl@stud.hisf.no>
 */
public class Coordinates {

    /*
    Declaration of variables
     */
    int size;
    InputStream fromServer;
    DataInputStream in;
    private double x1norm;
    private double y1norm;
    private double z1norm;
    private double rxnorm;
    private double rynorm;
    private double rznorm;
    private double x1p;
    private double y1p;
    private double z1p;
    private double rxp;
    private double ryp;
    private double rzp;
    byte[] b = new byte[10000];

    URsocket robotClient;

    Coordinates(URsocket robotClient) {
        this.robotClient = robotClient;
    }

    public void getData() {
        final int MAX = 10000;

//        b = new byte[MAX];
        byte buf_net[] = new byte[MAX];
        byte buf_new[] = new byte[MAX];     //  excess?
        int index = 0;

        do {
            /*
            Here we create and read from stream
             */
            try {
                fromServer = robotClient.client.getInputStream();
                size = fromServer.read(b);
            } catch (IOException ex) {
                System.out.println("error" + ex);
            }

            /*
            Here we check if stream is empty
             */
            if (size > 0) {
```

```
63.                 /*
64.                 Here we do something that seems to work
65.                  */
66.                 for (int i = 0; i < size; i++) {
67.                     buf_net[index++] = b[i];
68.                 }
69.                 while (index >= size) {
70.                     for (int i = 0; i < 560; i++) { //  excess?
71.                         buf_new[i] = buf_net[i];    //  excess?
72.                     }
73.                     for (int i = size; i < MAX; i++) {
74.                         buf_net[i - size] = buf_net[i];
75.                     }
76.
77.                     index = index - size;
78.
79.                     /*
80.                     Here we wrap a byte array into a buffer
81.                      */
82.                     ByteBuffer bb = ByteBuffer.wrap(b);
83.                     int sizenew = bb.getInt(0);
84.
85.                     if (sizenew == size) {
86.                         /*
87.                         Get coordinates. Sent from UR as Double
88.                         Print both types of coordinates
89.                         +41 / +8
90.                          */
91.                         x1norm = bb.getDouble(47);
92.                         y1norm = bb.getDouble(88);
93.                         z1norm = bb.getDouble(129);
94.                         rxnorm = bb.getDouble(170);
95.                         rynorm = bb.getDouble(211);
96.                         rznorm = bb.getDouble(252);
97.                         x1p = bb.getDouble(290);
98.                         y1p = bb.getDouble(298);
99.                         z1p = bb.getDouble(306);
100.                                rxp = bb.getDouble(314);
101.                                ryp = bb.getDouble(322);
102.                                rzp = bb.getDouble(330);
103.
104.                        }
105.                    }
106.                }
107.            } while (true);
108.        }
109.
110.        /**
111.         * @return the x1norm
112.         */
113.        public double getX1norm() {
114.            return x1norm;
115.        }
116.
117.        /**
118.         * @return the y1norm
119.         */
120.        public double getY1norm() {
121.            return y1norm;
122.        }
123.
124.        /**
125.         * @return the z1norm
126.         */
127.        public double getZ1norm() {
128.            return z1norm;
```

```
129.            }
130.
131.            /**
132.             * @return the rxnorm
133.             */
134.            public double getRxnorm() {
135.                return rxnorm;
136.            }
137.
138.            /**
139.             * @return the rynorm
140.             */
141.            public double getRynorm() {
142.                return rynorm;
143.            }
144.
145.            /**
146.             * @return the rznorm
147.             */
148.            public double getRznorm() {
149.                return rznorm;
150.            }
151.
152.            /**
153.             * @return the x1p
154.             */
155.            public double getX1p() {
156.                return x1p;
157.            }
158.
159.            /**
160.             * @return the y1p
161.             */
162.            public double getY1p() {
163.                return y1p;
164.            }
165.
166.            /**
167.             * @return the z1p
168.             */
169.            public double getZ1p() {
170.                return z1p;
171.            }
172.
173.            /**
174.             * @return the rxp
175.             */
176.            public double getRxp() {
177.                return rxp;
178.            }
179.
180.            /**
181.             * @return the ryp
182.             */
183.            public double getRyp() {
184.                return ryp;
185.            }
186.
187.            /**
188.             * @return the rzp
189.             */
190.            public double getRzp() {
191.                return rzp;
192.            }
193.
194.            public String getConnection() {
```

```
195.                String ret = "";
196.                switch (b[18]) {
197.                    case 0:
198.                        ret = "Disconnected";
199.                        break;
200.                    case 1:
201.                        ret = "Connected";
202.                        break;
203.                }
204.                return ret;
205.            }
206.
207.            public String getDisabled() {
208.                String ret = "";
209.                switch (b[19]) {
210.                    case 0:
211.                        ret = "Disabled";
212.                        break;
213.                    case 1:
214.                        ret = "Enabled";
215.                        break;
216.                }
217.                return ret;
218.            }
219.
220.            public String getPower() {
221.                String ret = "";
222.                switch (b[20]) {
223.                    case 0:
224.                        ret = "Off";
225.                        break;
226.                    case 1:
227.                        ret = "On";
228.                        break;
229.                }
230.                return ret;
231.            }
232.
233.            public String getEmergencyStop() {
234.                String ret = "";
235.                switch (b[21]) {
236.                    case 0:
237.                        ret = "Not activated";
238.                        break;
239.                    case 1:
240.                        ret = "Activated";
241.                        break;
242.                }
243.                return ret;
244.            }
245.
246.            public String getSecurityStop() {
247.                String ret = "";
248.                switch (b[22]) {
249.                    case 0:
250.                        ret = "Off";
251.                        break;
252.                    case 1:
253.                        ret = "On";
254.                        break;
255.                }
256.                return ret;
257.            }
258.
259.            public String getProgramRunning() {
260.                String ret = "";
```

```
261.                switch (b[23]) {
262.                    case 0:
263.                        ret = "Not Running";
264.                        break;
265.                    case 1:
266.                        ret = "Running";
267.                        break;
268.                }
269.                return ret;
270.            }
271.
272.            public String getProgramPaused() {
273.                String ret = "";
274.                switch (b[24]) {
275.                    case 0:
276.                        ret = "Not paused";
277.                        break;
278.                    case 1:
279.                        ret = "Paused";
280.                        break;
281.                }
282.                return ret;
283.            }
284.        }
```