NTNU
Det skapende universitet

The NTNU Cyborg 1.0

Robottekniske aspekter

Jonas Fyrileiv Nævra

Master i kybernetikk og robotikk
Innlevert: juli 2015
Hovedveileder: Øyvind Stavdahl, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Sammendrag

The NTNU Cyborg skal etter hvert bli en ekte kyborg, altså en kybernetisk organisme, men først skal det utvikles en ren robot. The NTNU Cyborg 1.0 er en robot bestående av en mobil robotbase som skal bære en kropp med en nakkerobot og en Microsoft Kinect som hode/sensor. Kinecten skal brukes til å registrere mennesker slik at roboten kan oppsøke dem. Denne skal avdukes i nær framtid, så derfor har det vært viktig i denne oppgaven å danne et godt grunnlag for denne roboten.

Det har blitt kjøpt inn en mobil robotbase som har en rekke innebygde funksjoner som er nødvendige for NTNU Cyborg, en Pioneer LX fra Adept MobileRobots. Etter minimalt med konfigurering kunne denne roboten brukes til å skanne et område for å lage kart, roboten kunne deretter kjøre fritt innenfor dette området uten å krasje i verken bevegelige eller stillestående hindringer, og den kunne kjøre automatisk til ladestasjonen sin. Det har blitt gitt anbefalinger for videre arbeid med den før avdukingen av The NTNU Cyborg 1.0.

I forbindelse med utvikling/anskaffelse av en nakkemodul til NTNU Cyborg har det blitt satt funksjonalitetskrav som kan forventes av en robotnakke, basert på menneskelig atferd. Det inkluderer bl.a. å nikke, å riste på hodet og å tilte hodet til siden. Undersøkelser har blitt gjort av eksisterende produkter og prosjekter som kan være relevante. Det viste seg at ingen av disse prosjektene tilfredsstilte ønskene for nakkemodulen, så derfor har det blitt vurdert som mest hensiktsmessig å utvikle en egen robot som kan brukes som en humanoid nakkerobot. Resultatet ble nakkeroboten, GrusBot, som har blitt en enkel og allsidig robot med tre frihetsgrader og tre kraftige aktuatorer som hver har sin innebygde regulator som gir god presisjon og konstant hastighet. GrusBot har fått et datablad, en brukermanual og en API dokumentasjon for SDK programvaren som er utviklet. Et eksempelprogram ble laget for å vise hvordan denne programvaren kan brukes i praksis.

Systemintegrasjon har hele tiden vært viktig med tanke på innkjøp av komponenter og selvproduksjon. Installasjon av en kraftig PC på roboten vil kunne avlaste den innebygde datamaskinen ombord i robotbasen som heller kan fokusere på navigasjonsoppgaver. Det er også nødvendig for å kunne kjøre den krevende programvaren til Kinectsensoren. I løpet av denne oppgavens tid har det blitt laget en ramme med en påmontert IRIS- mekanisme av en gruppe studenter. Med noen små modifiseringer kan denne brukes som kroppen til NTNU Cyborg som binder robotbasen sammen med nakkemodulen og Kinectsensoren.

Det neste som bør prioriteres med NTNU Cyborg vil være å fullføre systemintegrasjonen som det ikke var mulighet til å gå videre med i dette prosjektet. Nakkemodulen og dens programvare bør få noen små oppgraderinger før den kan integreres med NTNU Cyborg. En stor del av den videre utviklingen vil antakeligvis være å integrere programvaren til BabyX av Mark Sagar, som er kjent for sin animasjonsteknologi bak bl.a. filmen Avatar.

Summary

The NTNU Cyborg will eventually become a true cyborg, ie a cybernetic organism, but it will first be developed as only a robot. The NTNU Cyborg 1.0 is a robot consisting of a mobile robot base, a body with a humanoid neck robot and a Microsoft Kinect as head/sensor. The Kinect sensor will be used to find people and use the robot base to move towards them. This robot will be unveiled in the near future, so therefore the aim of this task as been to create a good foundation for this robot.

A Pioneer LX from Adept MobileRobots has been purchased, which is a mobile robot base that includes many built-in functions that are necessary for NTNU Cyborg. Afteronly a small effort, this robot could scan an area for making maps, the robot could then run freely in this area without crashing into neither moving or stationary obstacles, and it could run automatically to its charging station.

In conjunction with the development/acquisition of a neck module to NTNU Cyborg, there has been made some functional requirements that should be expected from a robot neck, based on human behavior. It includes nodding, shaking the head and tilting the head to the sides. Studies have been done of existing products and projects that may be relevant. It turned out that none of these projects meet the requirements of the neck module, so therefore it's been considered more appropriate to develop a robot that can be used as a humanoid robot neck. The end result became the neck robot, GrusBot, which is a simple and versatile robot with three degrees of freedom and three powerful actuators each having its embedded controller that provides good accuracy and constant speed. GrusBot has a data sheet, a user manual and API documentation for the SDK software that has been developed too. A sample example program was designed to show how this software can be used in practice.

System integration has always been important in terms of procurement of components and production. Installation of a powerful PC of the robot will be able to relieve the Pioneer LXs embedded computer that rather should focus on navigational tasks. It is also necessary to run the demanding Kinect software on a more powerful computer. During this project there has been made a frame with a mounted IRIS mechanism, done by a group of other students. With some minor modifications, this can be used as the thorax of NTNU Cyborg that binds the robot base with neck module and the Kinect sensor.

The next thing that should be prioritized with NTNU Cyborg will be to complete the system integration that was not possible to finish with this project. The head module and its software should get some small upgrades before it can be integrated with NTNU Cyborg. A large part of the further development will probably be to integrate the software for BabyX by Mark Sagar, who is known for his animation technology behind movies such as the Avatar.

Forord

Jeg er veldig fornøyd med oppgaven jeg fikk som masteroppgave. Arbeidsoppgavene har vært veldig variert, inkludert produktevaulering, bestillingsprosess av en eksklusiv robot og konfigurering av/leiking med denne, og utvikling av en robot som innebar produktspesifikasjon, design og utvikling, produksjon, montering og utvikling av programvare. Med tanke på læringsutbyttet, erfaringen og spennende arbeidsoppgaver så har jeg antakeligvis fått mye mer ut av denne oppgaven enn dersom jeg skulle jobbet mer spesifikt med mer teoretiske oppgaver slik mange av oppgavealternativene var. Alt jeg har lært mens jeg har jobbet med masteroppgaven kommer til å være veldig nyttig for meg i arbeidslivet, og jeg vil uten tvil få god nytte av det i forbindelse med personlige prosjekter.

Øyvind Stavdahl fortjener en stor takk for den gode veiledningen jeg har fått. Faste møtetidspunkter hver uke har hjulpet både på fremgangen og med inspirasjonen. Til tross for stor entusiasme over NTNU Cyborg prosjektet selv, så lot han meg jobbe selvstendig og gjorde at jeg fikk en følelse av et lite eierskap til prosjektet, noe som bare har virket positivt for motivasjonen.

Jeg vil også rette en stor takk til mekanikere på mekanisk verksted som har utført en utmerket jobb med komponentene jeg har designet. Jeg takker også for lånet av verktøy og utstyr slik at jeg kunne gjøre litt mekanisk arbeid selv.

Mine foreldre fortjener også en stor takk all støtten, husly og hjelp til rettlesning. Selv om jeg har skrevet store deler av rapporten i deres hjem, så har jeg til fått den arbeidsfreden jeg trengte. Uten dette ville det vært vanskelig å fullføre oppgaven på like god måte som det nå er gjort.

Jeg er veldig fornøyd med resultatene mine og det jeg har gjort, og gleder meg til å se hvordan dette brukes videre i prosjektet med NTNU Cyborg. Det skal også bli veldig spennende å se hvordan etterkommere av The NTNU Cyborg 1.0 blir og om det en gang blir en ekte kyborg.

Innhold

Tabelliste

Figurliste

ORDLISTE OG FORKORTELSER

Noen ord og forkortelser kan ha flere betydninger. Forklaringene her baserer seg på hvordan de brukes i denne rapporten.

Ordliste

Aksekrysspunkt	-	Krysningspunktet mellom aksene i en sfærisk robot
Anterior	-	Brukes innen medisin for å beskrive at en kroppsdel er plassert nærmere ventralflaten (bukflaten) i forhold til en annen
Baud rate	-	Overføringshastighet av data
Compliance	-	Evnen til å opprettholde, f.eks. en posisjon. Kan sammenlignes med elastisitet
Dødgang	-	Bevegelsesfriheten mellom to objekter uten at objektene settes i bevegelse som følge av objektenes motkrefter
Ekstensjon	-	Fra *extension* på engelsk som betyr utstrekking eller forlengelse
End effector	-	Generelt uttrykk for mekanismen eller apparatet på enden av roboten som samhandler med omgivelsene
Fleksjon	-	Fra *flexion* på engelsk som betyr bøyning
Frihetsgrad	-	Antall frihetsgrader beskriver hvor mange uavhengige bevegelser som kan utføres
Kyborg	-	Kybernetisk organisme, en hybrid av maskin og organisme
Lateral	-	Brukes innen medisin for å beskrive at en kroppsdel er plassert nærmere en lateralflate (sideflate, høyre eller venstre) i forhold til en annen
Posterior	-	Brukes innen medisin for å beskrive at en kroppsdel er plassert nærmere dorsalflaten (ryggflaten) i forhold til en annen
Sfærisk robot	-	Robot med tre rotasjonsakser hvor alle aksene krysser i et felles senterpunkt. End effector opererer da på et kuleformet/sfærisk plan

Forkortelser

API	-	Application Programming Interface, [57]
EEPROM	-	Electrically Erasable Programmable Read-Only Memory, [58]
CM	-	Center of Mass
CCW	-	Counterclockwise
CW	-	Clockwise
DH	-	Denavit-Hartenberg
DXL	-	Dynamixel
ESD	-	Electrostatic discharge
GrusBot	-	Gir, rull, stamp robot
IDE	-	Integrated Development Environment, [59]
PWM	-	Pulse-Width Modulation. En vanlig teknikk for å styre servomotorer fra en digital regulator
RC	-	Radio Control. Fjernstyring via radiosignaler
R/W	-	Read / Write
SDK	-	Software Development Kit
SLAM	-	Simultaneous Localization And Mapping,[62]
TCP	-	Tool Center Point. Tilsvarer midtpunktet til end effector

Kapittel 1

INNLEDNING

Prosjektet The NTNU Cyborg og versjon 1.0 skal snart avdukes og det er viktig å gjøre gode valg for rask fremgang. Produkter, komponenter og utvikling av robotteknisk utstyr koster også mye, så dårlige valg kan få store konsekvenser både for tidsskjemaet og budsjettet. Det finnes begrenset med informasjon om utvikling av en kyborg og humanoid robot ettersom teknologien foreløpig er hovedsaklig på forskningsstadiet, og det er derfor ikke noe konkret å ta utgangspunkt i ved utviklingen av NTNU Cyborg. Utvikling av en humanoid robot innebærer også undersøkelser av menneskets motorikk og atferd. Kroppsspråket er viktig for en sosial robot som denne, og effekten av et godt implementert kroppsspråk og personlighet kan ikke undervurderes. Det langsiktige målet er å utvikle en kyborg. Som definert i en tidligere rapport [25] i forbindelse med dette prosjektet kan en kyborg beskrives slik:

> *En kyborg (kybernetisk organisme) er et kybernetisk system der biologiske nerveceller kan behandle signaler fra elektroniske komponenter, med minst én tilbakekobling direkte mellom elektronikken og nervecellene.*

Foreløpig er dette snakk om noen få år fram i tid, og det vil fram til da være feil å kalle NTNU Cyborg for en kyborg. Uttrykket brukes likevel for å være konsis og for å bruke et navn som kan skape ekstra interesse for prosjektet.

Målgruppen til denne rapporten er studenter og ansatte ved NTNU som skal jobbe med prosjektet, *The NTNU Cyborg*, og andre som jobber med et lignende prosjekt. Selv om enkelte seksjoner av rapporten er noe teknisk, skal det også være mulig for interesserte uten relevant bakgrunn å få glede av rapporten. Rapporten er skrevet på norsk fordi dette er et norsk prosjekt, og for å introdusere språket til denne teknologien som tilnærmet kun eksisterer på engelsk.

1.1 Bakgrunn og hensikt

Det henvises til en tidligere rapport i forbindelse med NTNU Cyborg [25] for beskrivelse av bakgrunnen for prosjektet *The NTNU Cyborg*. Denne oppgavens hensikt er å komme videre med dette prosjektet og å lage et godt grunnlag for veien videre. Det innebærer å jobbe med de robottekniske aspektene som må være på plass før funksjonalitet kan implementeres.

NTNU Cyborg eksisterer kun på papiret etter et grunnleggende forarbeid og undersøkelser om aktuelle robotsystemer og funksjonalitet. Det har blitt undersøkt et utvalg mobile robotplattformer som kan være aktuelle som selve basen til NTNU Cyborg, noen hodesystemer, og forslag til hvordan dette kan settes sammen til første versjon av NTNU Cyborg, [25]. En førsteutgave av funksjonsspesifikasjonen er allerede etablert som danner et grunnlag for hva NTNU Cyborg skal kunne gjøre, [16]. Det er på tide å gi prosjektet et mer stabilt fotfeste ved å gi NTNU Cyborg en fysisk kropp. Ettersom prosjektets mål er å kontinuerlig videreutvikles er det ikke noe konkret arbeid som gjenstår annet enn å videreutvikle roboten til en kyborg og å gjøre den så spektakulær som mulig.

Hensikten med rapporten er ikke bare for å dokumentere et arbeid, men for å gjøre overgangen til etterkommere på prosjektet naturlig og uten misforståelser. Viktig informasjon om prosjektet beskrives og er helt nødvendig for andre som skal ta over arbeidet seinere.

1.2 Konkretisering av oppgaven

Denne oppgaven går ut på å se på robottekniske aspekter som vil bringe NTNU Cyborg nærmest mulig en komplett førsteutgave, *The NTNU Cyborg 1.0*. Utgangspunktet for denne førsteutgaven er hentet fra [25], hvor den beskrives som en mobil robot med SLAM og mulighet for automatisk lading (Pioneer LX fra Adept MobileRobots), en nakkemodul og Microsoft Kinect V2 som skal brukes til å finne personer som roboten kan oppsøke. Selv om det er ønskelig å ferdigstille en førsteutgave raskt, er dette litt for ambisiøst, så konkrete deloppgaver er nødvendig. Oppgaven deles derfor inn i tre deler[1]:

Del 1: *Kjøp inn en relevant mobil robotbase i samråd med veileder.*
En konkret oppgave som skal gjøres på bakgrunn av resultatene fra [25].

Del 2: *Konfigurer/videreutvikle robotbasen (hvis nødvendig) til å kunne navigere innendørs og til å kunne kople seg til ladestasjon autonomt ved behov.*
Hvor omfattende denne oppgaven er avhenger av om robotbasen må videreutvikles eller kun installeres. Navigering innendørs kan begrenses til at roboten autonomt skal kunne kjøre en tilfeldig eller forhåndsbestemt bane på et avgrenset område uten å krasje i gjenstander som både er i ro og i bevegelse.

Del 3: *Identifiser og realiser en egnet "overkropp" til roboten som kan bære relevante sensorer (kamera, kinect etc.), display e.l., og styre disse i ulike retninger. "Overkroppen"*

[1] I samarbeid med veileder [54]

kan kjøpes eller bygges, avhengig av tilgjengelige komponenter og de funksjonelle kravene som er satt.
Dette anses som den største delen av hele oppgaven. De funksjonelle kravene blir satt, og overkroppen vil foreløpig kun være en nakkemodul. Microsoft Kinect V2 velges å brukes som et utgangspunkt for hode, ettersom andre sider av dette prosjektet allerede har begynt å arbeide med denne. En enkel midlertidig overkropp kan evt. skaffes/produseres for å binde robotbasen sammen med nakkemodulen og dens sensorer.

Del 4: *Beskriv hvordan de innkjøpte/produserte modulene kan kobles sammen til et fungerende system, og utfør denne systemintegrasjonen i den grad tiden tillater det.*
Modulene som kjøpes og produseres vurderes underveis i forhold til modularisering og mulighetene for systemintegrasjon, og det foreslås noen konkrete trinn for hvordan alt kan settes sammen. Det anses som en liten del av oppgaven, men likevel en viktig del spesielt med tanke på innkjøp og produsering av kompatible produkter og komponenter.

1.3 Organisering av rapporten

Denne rapporten består av 9 kapitler og 4 vedlegg, der vedleggene er en minst like viktig del av rapporten som selve innholdet. Inndeling av kapitlene og seksjonene er gjort på en naturlig og intuitiv måte med beskrivende titler som skal gi leser best mulig oversikt over rapporten. Oppgavene beskrevet over besvares i følgende kapitler: Del 1 og 2 besvares kun i Kapittel 2, Del 3 besvares i Kapittel 3, 4, 5 og 6 i tillegg til Vedlegg A, B og C, og Del 4 besvares hovedsaklig i Kapittel 7, men er også en naturlig del av utviklingsprosessen hvor innkjøp og produksjon gjøres med tanke på kompabilitet og modulariserbarhet. Litt mer detaljert er strukturen i resten av rapporten som følgende.

Kapittel 2 reserveres helt til besvarelse av Del 1 og Del 2 fra seksjonen over, og er derfor en konkret del av oppgaven. Det blir her gjort et endelig valg av innkjøp av robotbase, og det fortelles om bestillingsprosessen. Til sist beskrives installasjonen og konfigureringen av robotbasen, og anbefalt arbeid videre med roboten.

Kapittel 3 beskriver grunnleggende funksjonalitet som ønskes i nakkemodulen til NTNU Cyborg, og det settes noen krav til hva den må tilfredsstille. Dette gir grunnlaget for besvarelsen av Del 3 fra seksjonen over.

Kapittel 4 består av en presentasjon av noen av de mest relevante robotene som er funnet tilgjengelige for innkjøp, og det vurderes om noen av disse er egnet som nakkemodul til NTNU Cyborg eller om nakkemodulen skal helt eller delvis selvproduseres.

Kapittel 5 beskriver utviklingsprosessen til nakkemodulen. Relevante eksisterende produkter eller løsninger som ikke egner seg for innkjøp kan evt. brukes som et utgangspunkt for selvproduksjon. Kapittelet begynner med å presentere noen slike produkter og løsninger, og det vurderes om noen av disse egner seg å lage. Videre blir det klart at én av disse løsningene kan brukes til inspirasjon for utvikling av en ny robot. Mekanikken til både førsteutgave og andreutgave av nakkeroboten beskrives. Programvare for 3D-modellene

presenteres, og det inkluderes notater om hvordan 3D-modellene av komponentene er utviklet.

Kapittel 6 presenterer kort den ferdige roboten som blir førsteutgaven av nakkemodulen til NTNU Cyborg, og det henvises til databladet, brukermanualen og API dokumentasjonen til roboten. Videre presenteres innkjøpslisten til nødvendige komponenter og produksjonsprosessen for en eventuell reproduksjon av roboten. Det utledes også robotens kinematikk. Til sist blir det foreslått noen forbedringer av roboten.

Kapittel 7 kommer med forslag til hvordan systemintegrasjonen av de ulike modulene til NTNU Cyborg kan utføres i praksis.

Kapittel 8 har et par kommentarer rundt The NTNU Cyborg 1.0 og viser til et forslag til en ekstra funksjon som kan implementeres. Det kommenteres også noe rundt videre arbeid som er naturlig mot slutten av rapporten.

Kapittel 9 beskriver kort hva som er oppnådd med denne oppgaven, og gir er en oppsummering av de viktigste resultatene.

Vedlegg A er nakkemodulrobotens datablad som beskriver egenskaper, spesifikasjoner og kinematikkresultatene.

Vedlegg B er brukermanualen til nakkemodulroboten som beskriver installasjon, tilkoblinger, anbefalt programvare, standardinnstillinger og noen viktige bemerkninger. Det anbefales sterkt å lese gjennom hele denne manualen før roboten tas i bruk.

Vedlegg C er API dokumentasjonen til nakkerobotens SDK bibliotek, og gir informasjon om hvordan denne kan brukes.

Vedlegg D er et e-postsvar fra teknisk støtte hos Adept MobileRobots i forbindelse med et problem som beskrives i Kapittel 2.3.

Kapittel 2

ROBOTBASE: PIONEER LX

I dette kapittelet blir det gjort et endelig valg av robotbase til NTNU Cyborg. Bestillingsprosessen beskrives i korte detaljer med en oppsummering av bestillingsordren. Arbeidet som er gjort med robotbasen har hovedsaklig vært testing for å bli kjent med roboten og for å få en formening om hva som bør jobbes med seinere. Siden noe av de samme prosedyrene også bør utføres av etterkommere i dette prosjektet, så kommer det en oppsummering basert på arbeidsloggen som ble skrevet under testingen. Til sist anbefales det noe som bør undersøkes nærmere før avdukingen av The NTNU Cyborg 1.0.

2.1 Valg av robotbase

Undersøkelsene fra [25] sier at Pioneer LX fra Adept MobileRobots er den beste robotbasen for NTNU Cyborg av de som ble vurdert. Det har likevel blitt gjort noen undersøkelser og litteratursøk i ettertid i forbindelse med denne rapporten for å se om resultatene fra [25] fortsatt er relevante og gyldige, og for å være sikker på at rett robotbase blir valgt. Under er det en kort oppsummering av sammenligningen mellom de tre mest aktuelle robotbasene, Pioneer LX og PatrolBot fra Adept MobileRobots, og Summit XL HL fra Robotnik[1].

Pioneer LX:
Denne har en enkel og rask installasjon for brukeren, og alle funksjoner er ferdig installert og forhåndstestet før levering. Den har bl.a. innebygd PC med WiFi, navigasjonssystem og avstandssensorer, en rekke medfølgende programvare og dockingstasjon for automatisk lading. Selv om dette er den dyreste av disse tre robotbasene, er det er likevel den rimeligste roboten av alle som er undersøkt hvor disse grunnleggende og nødvendige funksjonene er inkludert, og som er nok robust og allsidig til å være plattformen til NTNU Cyborg. Andre lignende alternativer ligger på rundt 25% høyere pris + mva og frakt.

[1] Basert på resultatene fra [25] som også er bekreftet at fortsatt gjelder på nåværende tidspunkt.

PatrolBot:
Dette er en eldre utgave enn Pioneer LX, litt mindre robust, og betydelig dårligere batteri. Prisen er ca. den halve av Pioneer LX uten noe utstyr, men ettersom det meste av dette utstyret er helt nødvendig i dette prosjektet er det i tilfelle mye som måtte blitt selvimplementert, også bare for at roboten trygt skal kunne kjøre aleine blant folk. Prisen er tilsvarende Pioneer LX dersom sammenlignbare funksjoner og utstyr skal være implementert ferdig fra produsenten.

Summit XL HL:
Et robotkjøretøy som denne er helt klart imponerende med omnidirectional hjul. Sammenlignet med robotene over er den noe mer robust og er beregnet for både innendørs og utendørs bruk. Det er mye som ikke medfølger, men mye kan også kjøpes i tillegg. Den er programmerbar og programvare er åpent tilgjengelig. Prisen er nesten 30% av Pioneer LX, men den krever mye ekstra arbeid og/eller ekstrautstyr for å kunne bruke den til NTNU Cyborg.

I løpet av det nye litteratursøket har det fortsatt ikke blitt funnet noen robotbaser som vurderes til å overgå Pioneer LX fra Adept, så den står fortsatt som favoritten. Det har imidlertid skjedd forandringer i økonomien som har ført til en sterk svekkelse av norske kroner [24]. Det betyr at import av produkter fra land hvor valutakursen ikke har fått samme nedgang nå har blitt mye dyrere. Når en robotbase som denne koster så mye som den gjør vil en slik kronesvekkelse på $10 - 15\%$ føre til en total prisøkning på titusener av norske kroner.

Sammen med veileder [54] ble den samme robotbasen, Pioneer LX fra Adept, fortsatt vurdert til å være best egnet til NTNU Cyborg. Både på egenskaper og pris til tross for at budsjettet for robotbasen måtte økes etter nedgangen av kronekursen. Siden det allerede er skrevet om robotbasen i [25] tas ikke med de detaljerte spesifikasjonene til den her. Figur 2.1 viser et bilde av roboten.

Figur 2.1: Pioneer LX fra Adept MobileRobots, [21]

2.2 Bestillingsprosessen

Etter å ha fått bekreftet hvilken robotbase som skal bestilles, kunne forslaget og et budsjett sendes til prosjektleder, Stig Omholt. På grunn av en del forsinkelser og venting ble det også forespurt om å benytte ekstra rask produksjonstid fra Adept MobileRobots, for å få nok tid til å jobbe med roboten før denne oppgaven og delprosjektet må avsluttes. Prisen for dette var 5% av robotverdien. Forslaget ble godkjent og det ble til slutt bekreftet at bestillingen kunne gjennomføres [54]. Et nytt og oppdatert regnestykke for bestilling kan ses i Tabell 2.1.

Bestillingsordre av Pioneer LX fra Adept MobileRobots	
Utsalgspris:	$ 31 995 USD
Rush Order Service, 5 % av utsalgspris:	5 % av $ 31 995 USD = $ 1 600 USD
Wire Transfer Fee[2]	$ 40 USD
Fraktkostnader:	$ 1 750 USD
MVA, 25 % av delsum:	25 % av $ 35 345 USD = $ 8 836.25 USD
Total pris:	$ 44 221.25 USD **= kr. 348 905.70 NOK**[3]
Leveringstid fra bestillingsdato	4 - 6 uker

Tabell 2.1: Totalpris for Pioneer LX fra Adept MobileRobots [15].

Ettersom prosjektet med NTNU Cyborg styres fra Det Medisinske Fakultet (DMF) skal bestillingen også skje derfra. Det ble derfor opprettet kontakt med riktig avdeling i dette fakultetet for å starte bestillingsprosessen. I den forbindelse har det blitt fylt ut en *anskaffelsesprotokoll* og formidlet all nødvendig informasjon for at bestillingsordren kan sendes. Det har hele veien vært oppfølging og god kontakt med både DMF og kontaktpersonen fra Adept MobileRobots for å sikre at alt var i orden med bestillingen.

Robotbasen ble med positiv overraskelse mottatt nesten to uker tidligere enn forventet, og installasjonen begynte umiddelbart.

2.3 Installering og konfigurasjon

Når det i denne seksjonen henvises til *Brukermanualen*, vil det si brukermanualen til Pioneer LX [21].

Pioneer LX er en robotbase som er klar til å brukes nesten rett ut av boksen, så det er i utgangspunktet lite som må gjøres for å få roboten opp å kjøre. Arbeidet med roboten har vært å pakke ut, installere og teste roboten med dens inkluderte programvare, som

[2] Betalingsgebyr

[3] Benyttet valutakurs: 1 USD = 7.89 NOK. Valutakursene er hentet fra DNB sine nettsider og er basert på betaling via "Overføring ved kjøp". Kursene er kun en indikasjon og kan avvike fra faktiske beløp, [26].

beskrives nærmere under. På grunn av for liten tid og prioritering av andre oppgaver som ble vurdert høyere ble det ikke utført noe konkret systemintegrasjon eller programmering med robotbasen. Forfatterne av [52] fikk imidlertid testet IRIS-mekanismen (se Kapittel 7) med robotens batteri og en ekstern datamaskin med vellykket resultat.

2.3.1 Utpakking

Prosedyren med utpakkingen av robotbasen ble fulgt nøye etter instruksjonene i Brukermanualen, som på forhånd ble sendt via e-post fra kundeservice. Det anbefales sterkt av nye brukere å lese denne Brukermanualen selv, for informasjonen om utpakking derfra gjentas ikke her.

2.3.2 Oppstart og testing

Pioneer LX kommer med ferdigoppladet batteri, men ladestasjonen ble installert med en gang likevel. Det ble bare brukt gulvplaten som følger med til å feste ladestasjonen på, noe som anbefales kun som en midlertidig løsning, ettersom den ikke står skikkelig fast. Dette er en av tre måter å installere ladestasjonen på, og for langsiktig bruk bør en av de to andre metodene benyttes. Videre ble det koblet til en PC skjerm, tastatur og mus til Pioneer LX og påloggingen skjer på samme måte som ved en annen Ubuntumaskin med oppgitt påloggingsinformasjon. MERK at default påloggingsinformasjon er endret for å øke sikkerheten, og denne informasjonen er gitt til veileder [54] som har ansvaret for roboten. En personlig arbeidslogg ble skrevet i forbindelse med arbeidet med robotbasen, og i listen under presenteres det viktigste fra denne loggen, noe som kan være til hjelp i tillegg til Brukermanualen for å komme i gang.

- For å kunne overvåke Pioneer LX uten å ha den koblet direkte til skjerm, tastatur og mus, ble det brukt en bærbar PC (Windows 8.1) til å overvåke og kommunisere med roboten mens den kjørte. Roboten måtte da kobles til det trådløse nettet.
- Robotens innebygde datamaskin kobles til trådløst nettverk som beskrevet under ”Ubuntu_Linux_Network_Configuration \GUI-based configuration” på wiki-sidene til MobileRobots [22].
- Installering av Putty på laptop for å etablere kommunikasjon. I Putty hukes av ssh og IP-adressa til roboten skrives inn, som finnes under Wi-Fi \Connection information, og deretter trykkes ”Open”. Samme brukernavn og passord som på den innebygde maskinen på roboten skrives inn, og dersom dette er gjort riktig kan kommandovinduet nå brukes på samme måte som kommandovinduet på maskinen i roboten.
- Demoprogrammet med ARIA ble først testet som beskrevet i Kapittel 3 i Brukermanualen.
- For å teste MobileEyes endres directory til: cd /usr/local/Arnl/examples og deretter kan demoprogrammet: ./arnlServer kjøres. MobileEyes kan nå åpnes uten brukernavn og passord (blank tekst) med IP-adressen til roboten, slik det er beskrevet i

Kapittel 3 i Brukermanualen. Uten å ha laget et kart over arbeidsområdet vil roboten være i en ”Lost” tilstand og det er ikke så mye som kan gjøres annet enn manuell kjøring. For manuell kjøring av roboten i en Lost tilstand må ”Safe Drive Mode” deaktiveres, men da vil avstandssensorer være avslått og roboten kan krasje dersom brukeren ikke er forsiktig. Roboten kan kjøres med piltastene (etter å ha aktivert ”Drive” i menylinjen) eller med konsollen som medfølger.

- Simulatoren MobileSim ble testet sammen med et eksempelkart som beskrevet i Brukermanualen.
- Videre ble det fysiske arbeidsområdet skannet med robotens lasersensor og modifisert med hhv. MobileEyes og Mapper3, slik det beskrives i detalj i den medfølgende pdf-filen, ”ARNL Introduction” og i Mapper3 tekstfilen ”README”. Det kan kommenteres at kartet med rådata fra scanningen lagres som en .2d fil i */usr/local/arnl/examples* på roboten dersom ikke annet spesifiseres. På en ekstern PC kan kartet åpnes fra roboten dersom tilkoblingen er OK, så det er ikke nødvendig å lagre noe som helst på en ekstern PC. Rådatakartet kan i Mapper3 klargjøres ved å viske ut unødvendige hindringer, legge til forbudte sone og tegne hindringer manuelt. Hvordan kartet skal brukes i MobileEyes forklares i detalj i Kapittel 3 i Brukermanualen.
- Etter å ha fått laget et kart og åpnet det i MobileEyes kunne roboten kjøre fritt og autonomt innenfor dette kartet. Det ble opprettet noen målpunkter som bestemmer både posisjon og orientering, avstandssensorene ble testet, og dockingfunksjonen ble brukt til å sende roboten til ladestasjonen.
- Det viste seg at det bør være noe ekstra plass på sidene av ladestasjonen, ikke bare foran som beskrevet i ARNL Introduction, for sonarsensorene kan forstyrre dockingprosessen dersom hindringer er i nærheten av ladestasjonen. Eventuelt kan sonarsensorene deaktiveres i programvaren under dockingprosessen.
- Det ble også gjort noen justeringer av sonarparametrene i ARNL programvaren for å redusere problemet med feildetekteringer av hindringer. Problemet ble ikke helt løst og tiden strakk dessverre ikke til for å undersøke dette nærmere. Se neste seksjon for mer informasjon.

Det har blitt testet en del med roboten for å bli litt kjent med den, for å avdekke eventuelle feil eller mangler, og for å finne ut om den behøver noen modifiseringer før avdukingen av The NTNU Cyborg 1.0. Mye tid har gått med på å lese brukermanualen og ulike manualer fra wiki-sidene til MobileRobots slik at alt blir gjort riktig for å unngå ødeleggelser og unødvendig tid på feilsøking av eventuelle problemer. Det ble avdekket et par ting som antakeligvis trenger noe justering, som beskrives i neste seksjon.

2.4 Anbefalt videre arbeid med Pioneer LX

Oppsettet av denne robotbasen har vært enklere enn forventet, til tross for at mye tid ble brukt på studering av manualer. Veldig mange funksjoner medfølger uten noe ekstra arbeid for brukeren som gjør at roboten kan brukes til en rekke oppgaver allerede rett ut av

boksen. Som sagt tidligere ble det ikke tid til å arbeide så mye med denne roboten på grunn av andre oppgaver med høyere prioritet. Det som gjenstår av arbeid med robotbasen før avduking av The NTNU Cyborg 1.0 er hovedsaklig å integrere de andre komponentene/modulene sammen med den, se Kapittel 7. I tillegg er det noen egenskaper som anbefales å undersøke nærmere:

2.4.1 Feildetektering av hindringer

Pioneer LX er utstyrt med fire ultralyd sonarsensorer både foran og bak. De som er foran er plassert ganske nær gulvet, så dersom gulvet har en høy akustisk refleksjon kan de emitterte signalene fra sensorene reflekteres tilbake og roboten stopper for å unngå en hindring den tror er der. Det vil resultere i at roboten kjører veldig ujevnt og at den må beregne ny kjørerute veldig ofte som et forsøk på å komme videre uten å kjøre i hindringen den ser. Ved bruk av MobileEyes kan sonardetekteringene ses som blå trekanter, slik at dette kan overvåkes ved testing. En løsning på dette kan være å justere parametrene for sonaravlesningene i ARNL programvaren, som beskrevet av teknisk støtte hos MobileRobots etter en forespørsel på e-post. Se Vedlegg D for fullt svar fra MobileRobots. Teknisk støtte er gratis for kundene deres, så det er ikke dumt å benytte seg av det.

Noe av det samme problemet kan også oppstå ved kjøring i trange omgivelser eller gjennom en dør/luke eller lignende, hvor veggene rundt døråpningen vil reflektere ultralyden som igjen oppfattes som en hindring av roboten. Dette er ikke testet, men det er et kjent problem med roboter som bruker sonarsensorer, [56]. En mulig løsning på dette kan være å ignorere sonardeteksjonene akkurat når roboten kjører gjennom en døråpning, noe som er fullt mulig siden roboten alltid vil vite hvor disse er på kartet over området. I tilfelle vil det være tryggest å kjøre forover siden det er installert en støtfanger foran som vil registrere kontakt dersom sonarsensorene ikke gjør det.

2.4.2 Høy deakselerasjon

Når roboten kjører autonomt er ikke kjøringen så jevn som antakeligvis ønskes for NTNU Cyborg på grunn av noe høy deakselerasjon når den stopper opp. Hvis roboten er i en situasjon hvor den kun skal ta små skritt av gangen blir det mye belastning på utstyr dersom den skal bremse så raskt. Det ses ikke på som et kritisk problem, men det bør undersøkes nærmere.

Kapittel 3

NAKKEMODUL: KRAV OG FUNKSJONALITET

Selve idéen og konseptet med en nakkemodul ble introdusert i [25]. Her beskrives funksjonalitet som kan implementeres i nakkemodulen og krav som må eller anbefales å implementere.

3.1 Hodets funksjonalitet

NTNU Cyborg må kunne uttrykke et bevisst kroppsspråk for best mulig interaksjon med mennesker. Dette delkapittelet er en introduksjon til menneskehodets kroppsspråk og forslag til gesturer som kan og bør kunne implementeres i kyborgen.

Mennesker har en utrolig evne til å oppfatte og å forstå ikke-verbal kommunikasjon, altså kroppsspråk. Kroppsspråket er ofte ubevisst og kan derfor oppfattes som mer ærlig og bety mer enn kun ord. En person som ikke har et tydelig kroppsspråk kan oppfattes som usympatisk, uinteressert og negativ, noe som kan gi denne personen vanskeligheter i sosiale sammenhenger. Dessuten vil andre kunne misforstå og ha problemer med å forstå om personen er ærlig eller ikke. Lite bruk av kroppsspråk får personen til å virke statisk, faktisk minne om en robot, noe som til en viss grad ikke ønskes for NTNU Cyborg.

3.1.1 Kroppsspråket til hodebevegelser

Hode- og ansiktsbevegelser er en stor del av kroppsspråket til mennesker, og kan derfor ved riktig bruk være med på å gjøre en robot mer ”levende” og gi den en personlighet. For NTNU Cyborg 1.0 fokuseres det på å realisere hodebevegelser ved bruk av en nakkemodul. Under vises noen karakteristiske hodegesturer som kan tenkes å implementeres i NTNU

Cyborg, med tilhørende beskrivelse av hvordan gesturene kan oppfattes av mennesker[1]:

- **Nikke:** En nokså entydig gestur for å uttrykke samtykke, enighet og bekreftelse. Et enkelt nikk oppover, gjerne med hode litt på skrå, brukes spesielt mye av herrer og gutter i den yngre generasjonen for å hilse på andre. Et enkelt nikk nedover er også brukt på samme måte, men gjerne i en form for å bukke i en mer formell og høytidelig sammenheng.

  Å nikke, riste på hodet og peke, se neste punkter, er de første hodegesturene vi lærer oss som barn, og bør også være de viktigste gesturene som implementeres i NTNU Cyborg.

- **Riste på hodet:** Høyre/venstre-rotasjon. Som det motsatte av å nikke vil dette i de fleste tilfeller oppfattes som et nei. Å være oppgitt kan uttrykkes ved å bl.a. riste på hodet med synet på skrått nedover, en enkel gestur, men krever en relativ kompleks sosial intelligens for å brukes riktig.

- **Peke:** Å rette synet mot et objekt for å uttrykke interesse for det, er antakeligvis en av de gesturene som oppfattes likt for mennesker fra hele verden. Dette bør være den viktigste funksjonen til NTNU Cyborg sin nakkemodul, som også bør holde øyekontakt med mennesker til den grad det er mulig.

- **Lateral tilting:** Ved tilting av hodet sideveis uttrykker personer ofte nysgjerrighet og interesse, som er spesielt relevant for NTNU Cyborg. Det er en typisk positur som brukes ubevisst, men oppfattes nokså tydelig av andre mennesker. Større tilting kan tyde på større nysgjerrighet og interesse.

- **Svaie lateralt:** Repeterende tilting av hodet til begge sidene. Dette kan oppfattes som en litt leken og barnslig oppførsel som kan være underholdende for andre, men også irriterende dersom det gjøres for ofte eller f.eks. under en samtale, ettersom det kan være et tegn på uinteresse. I riktige sammenhenger og på rett tid kan slike bevegelser oppfattes positivt for NTNU Cyborg.

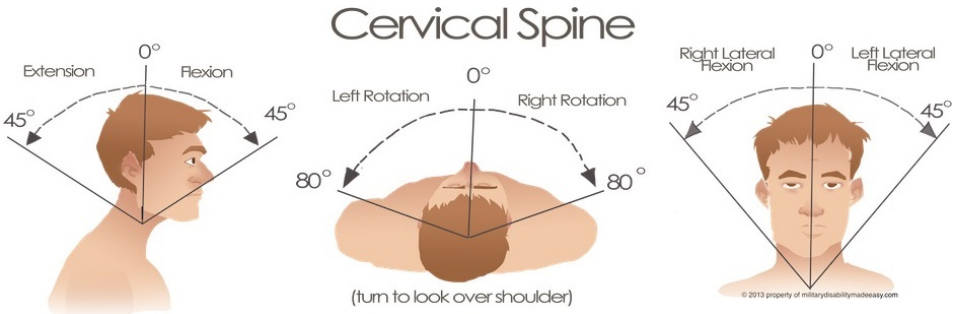
- **Fleksjon:** Å senke hodet. Posituren kan uttrykke spesielt tre ting, at man er sjenert/redd, er trist eller underdanig/submissiv. Noen senker også hodet når det bes om unnskyldning eller tillatelse. For NTNU Cyborg kan det være lurt å være forsiktig med denne hodeposisjonen, ettersom det kan tolkes på flere måter, selv om det kan gi en stor effekt av å brukes riktig. Ofte er det i kombinasjon med mimikk og øyebevegelser som kan avgjøre hva kroppsspråket sier her, men for NTNU Cyborg er dette foreløpig ikke relevant, og derfor er det ekstra viktig å utføre denne bevegelsen i riktig sammenheng for å kunne gi riktig uttrykk.

- **Ekstensjon:** Å heve hodet uttrykker dominanse og selvtillitt, og ingen frykt for at noen kan skade den sårbare halsen som nå er lett tilgjengelig. Dersom NTNU Cyborg skal ha en litt kjepphøy personlighet må den kunne holde denne posisjonen i flere situasjoner. Hevet hode er også vanlig å gjøre ved kjedsomhet, og da gjerne i kombinasjon med synet rettet oppover.

[1] Basert på egen uprofesjonell tolkning av generelle hodebevegelser og fra [55]. Det bør merkes at disse tolkningene baseres på vestlig kultur, og kan ha andre betydninger i andre kulturer som f.eks. i ulike asiatiske og middelhavsland.

3.1.2 Bevegelsesområdet til nakken

Det er naturlig å vise til menneskets bevegelser og bevegelsesområdet ved utvikling av humanoide[2] roboter. NTNU Cyborg skal være type humanoid robot, og det ønskes at bevegelsene dens skal være minst like gode som hos et normalt menneske. Her presenteres bevegelsesområdet til de tre viktigste nakkebevegelsene for en normal og frisk person, og vil bli brukt som et utgangspunkt til spesifikasjonene nakkeroboten skal ha [8].



Figur 3.1: Normalt bevegelsesområdet for fleksjon/ekstensjon, rotasjon og lateral tilting.

3.1.3 Andre hodebevegelser

I tillegg til fleksjon/ekstensjon, rotasjon og lateral tilting er anterior ekstensjon og posterior fleksjon bevegelser de fleste personer klarer, altså å bevege hodet hhv. fremover og bakover. Enkelte personer har også evnen til lateral bevegelse uten å tilte på hodet. Disse bevegelsene antas å ikke være spesielt viktige for NTNU Cyborg, og fokuseres derfor å ikke på her.

3.2 Krav til nakkemodulen

På dette stadiet er det ikke satt noen spesifikke og strenge krav til funksjonaliteten, men det er likevel noe som forventes og ønskes, og enkelte punkter er viktige å ta hensyn til ved anskaffelse og/eller utvikling av nakkemodulen.

3.2.1 Sikkerhet

Sikkerheten må til en hver tid være på plass for at en robot i det hele tatt skal ha lov til å slippes fri blant andre mennesker. Det innebærer bl.a. at fingre, hår og andre kroppsdeler skal ha liten risiko for å kunne hekte seg fast i mekanikken. Bevegelsene kan være relativt raske, men ikke så aggressive at personer ikke får muligheten til å flytte hender og lignende før et eventuelt sammenstøt. Det vil i utgangspunktet ikke være et problem for et hode med en sylinder/kuleform, men dersom hodet vil ha noen "utvekster" kan et sammenstøt gjøre skade på både person og robot, og er derfor viktig å kunne unngå. Det skal ikke være noen

[2]Menneskelignende. Fra *humanoid* på engelsk.

som helst risiko for at en person skal få elektrisk støt ved kontakt med NTNU Cyborg[3], og det gjelder også nakkeroboten. Hele roboten bør uansett være ESD-kompatibel for å unngå ødeleggelse av elektronikken.

Nakkemodulen bør ha sikkerhetsmekanismer som fysisk kan hindre at feil eller andre problemer i programvaren og styrekontrolleren kan gjøre skade. Dersom nakkens aktuatorer ikke har interne sperringer bør selve konstruksjonen kunne hindre aktuatorene til å gi utslag utenfor det akseptable bevegelsesområdet. Uten ytre påvirkninger skal det ikke være mulig for ledninger eller andre deler å komme i klem i den bevegelige mekanikken. Det vil redusere bevegeligheten, gjøre skade og slitasje, og ledninger kan potensielt miste isoleringen og stor skade på elektronikken kan oppstå.

3.2.2 Modularisering

Nakken skal være en egen modul, slik at nakken og resten av NTNU Cyborg i størst mulig grad også kan fungere uavhengig av hverandre. Siden nakken og hodet bør være to tilnærmet uavhengige moduler, vil det være en fordel om monteringsmulighetene til nakkemodulen også kan brukes av hodemodulen slik at hodet kan bli igjen på kyborgen dersom nakken må fjernes for undersøkelse/oppgradering eller lignende. Selve nakkemodulen bør bygges modulært for å enkelt kunne oppgradere og å bytte ødelagte deler. Dersom det skal gjøres oppgraderinger som å implementere en eller flere ekstra frihetsgrader eller annen mekanisk modifisering, forventes det ikke at nakkemodulens software kan brukes uten noen modifiseringer. Det er naturlig at koden må oppgraderes med ekstra funksjoner og annen modifisering for å kunne håndtere den nye mekanikken, men det krever at den opprinnelige koden må være oversiktlig og lettleselig. Strøm og datakommunikasjon skal ha standard tilkoblinger for å enkelt kunne bruke nakkemodulen i andre systemer.

Slik modularisering er spesielt viktig i et prosjekt som dette hvor det vil foregå en kontinuerlig utvikling med antakeligvis mange oppgraderinger. Det vil også gjøre det enklere med vedlikehold, testing, feilsøking og reparering, hvor kun den enkelte delmodulen trenger å bli tatt ut av drift istedenfor hele NTNU Cyborg. Modularisering vil gjøre det mulig å for flere å jobbe med NTNU Cyborg samtidig, ettersom alle kan jobbe med hver sin modul.

3.2.3 Kraft og moment

Den foreløpige planen med NTNU Cyborg 1.0 er at den skal bruke Kinect V2 som "hode". Minimumskravet til kreftene som aktuatorene til nakkemodulen skal kunne yte blir derfor estimert på bakgrunn av Kinectens vekt og størrelse. Kinect V2 veier ca. $1\mathrm{kg}$ pluss kabler[4]. Massesenteret er ikke gitt, men ble estimert til å være $36\mathrm{mm}$ opp fra bunnen. Avstanden fra rotasjonsaksen og ut til end effectoren som Kinectsensoren monteres på er l_e. Det betyr at dreiemomentet fra aktuatoren minst må kunne bevege $m = 1\mathrm{kg}$ med en armlengde på $l = l_\mathrm{kinect} + l_\mathrm{e}$ i hele bevegelsesområdet med ønsket akselerasjon for massesenteret, a_CM.

[3] Det gjelder ikke små ufarlige elektriske utladninger som følge av at personen selv er elektrisk ladd.
[4] Vekten er målt selv.

Dreiemomentet kan beregnes ved å utføre kryssproduktet av kraftvektor og armvektor, som vist med ligning 3.1.

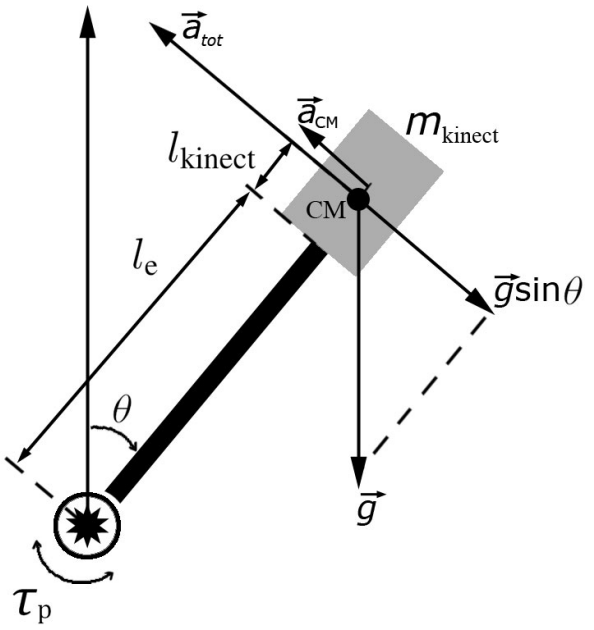
$$\boldsymbol{\tau_p} = \text{kraft} \times \text{arm} \tag{3.1}$$

Dersom armen står vinkelrett på kraftvektoren kan momentet beregnes ved å multiplisere kraftverdien med armlengden. Se Figur 3.2. Minimumskravet for dreiemomentet kan da beregnes med utrykket i ligning 3.2.

$$\tau_p = F \cdot l = ma_{\text{tot}} \cdot (l_{\text{e}} + l_{\text{kinect}})$$

$$\tau_{\text{p, max}} = m[g\sin(\theta_{\text{max}}) + a_{\text{CM}}](l_{\text{e}} + l_{\text{kinect}}) \tag{3.2}$$

der θ_{max} er vinkelen hvor horisontal armlengde er størst, som i Figur 3.2 vil være $\pm 90°$. Dersom det er implementert vinkelbegrensninger innenfor $\pm 90°$, vil θ_{max} være den maksimale vinkelen i begge retninger.



Figur 3.2: Prinsipiell skisse for momentberegning

3.2.4 Konstruksjon

Konstruksjonen må først og fremst kunne tilfredsstille sikkerheten beskrevet over. Det kreves også en solid og robust konstruksjon for at nakken skal kunne tåle å bære et hode, og å kunne bevege det med den akselerasjonen og hastigheten som aktuatorene kan klare. Mye slingring og dødgang er uønsket, og posisjonsavvik pga. elastisitet og svikt i komponenter bør unngås. Fysiske begrensninger for bevegelsesområdet er viktige for at ingen ting blir ødelagt dersom nakken beveger seg utenfor dette tillatte området.

3.2.5 Vedlikehold

Vedlikehold er et viktig punkt for at dette kyborgprosjektet skal kunne leve og utvikles i mange år framover. Prosjektet vil sannsynligvis ikke prioriteres av andre enn studenter som midlertidig vil jobbe med det i forbindelse med studentprosjekter, deltidsjobber, fordypnings- og hovedoppgaver, og derfor vil vedlikeholdsfrie og robuste moduler gjøre at fokuset kan ligge på videreutvikling istedenfor å bruke ressurser på å reparere/oppgradere ødelagte/dårlige moduler. Nakkemodulen vil ha flere bevegelige deler som kan føre til slitasje, og som fort kan ødelegges dersom delene ikke er slitesterke og mekanikken er robust.

3.2.6 Bevegelser og frihetsgrader

I seksjon 3.1.1 ble det beskrevet noen av de vanligste bevegelsene mennesker gjør med hodet. Det er ikke unaturlig at det ønskes at NTNU Cyborg skal kunne gjøre flest mulig av disse, og med minst like stort bevegelsesområde som et normalt menneske. Et realistisk mål for hva nakken skal klare er viktig for å kunne fullføre arbeidet. For hver ekstra frihetsgrad blir hele nakkemodulen ekstra komplisert. Noe avhengig av hvilken mekanikk som benyttes, så vil flere frihetsgrader bety en dyrere nakkemodul, flere ting som kan gå i stykker, mer strømforbruk, mer programmering, noe større krav til prosessorkraft og spesielt større krav til robotens intelligens[5] til hvordan nakken skal brukes i hvilke situasjoner.

Basert på bevegelsene i seksjon 3.1.1 er det *minst* tre nakkebevegelser som det antas forventet av en humanoid robot som NTNU Cyborg: *fleksjon*, *ekstensjon* og *rotasjon*. Disse bevegelsene krever kun to frihetsgrader, *yaw* og *pitch*, og det vil ikke kreve en spesielt komplisert nakkemodul. Dersom kyborgen er veldig flink til å bruke denne nakken i riktige situasjoner vil det være imponerende. Likevel er det enda én bevegelse som skiller de fleste undersøkte humanoide roboter fra mennesker, og det er *lateral tilting*. Ambisjonen vil derfor være å skaffe eller produsere en nakkemodul med minst tre frihetsgrader, *yaw*, *pitch* og *roll*, og da vil nakken kunne gjøre alle bevegelsene beskrevet i 3.1.1. Hvis det viser seg at dette blir for komplisert kan det aksepteres en midlertidig nakkemodul med kun to frihetsgrader.

[5] Dette krever også at utviklerne har en god sosial intelligens som veit hva som må implementeres.

Kapittel 4

NAKKEMODUL: VURDERE KOMMERSIELLE PRODUKTER

I utgangspunktet var ønsket å selvprodusere nakkemodulen, både fordi den da kan lages ut ifra de ønskede spesifikasjonene, og at det vil gi en god erfaring med mye læring. I dette kapittelet presenteres likevel noen nakkemekanismer som er kommersielt tilgjengelige for innkjøp. Dersom det skulle vise seg at et veldig godt produkt allerede eksisterer vil det være unødvendig å produsere et tilsvarende produkt selv. Uansett er det naturlig å undersøke hva som allerede eksisterer, og funnene kan også bli et utgangspunkt og til inspirasjon ved en selvproduksjon av nakkemodulen.

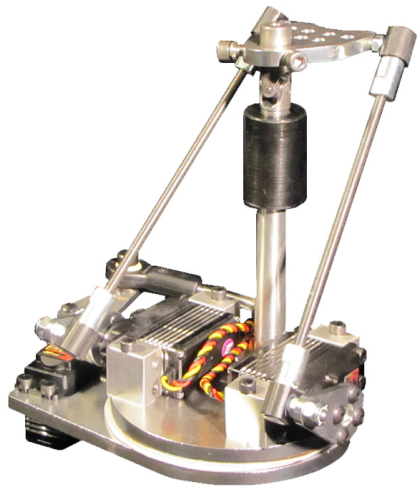
Om det eksisterer en kommersiell robotnakke som er tilnærmet perfekt for NTNU Cyborg basert på ønskede egenskaper fra Kapittel 3 kan ikke bekreftes, ettersom det foreløpig ikke har blitt funnet mens arbeidet med prosjektet har pågått. Et omfattende søk etter kommersielle nakkemoduler har resultert i at følgende produkter vurderes som de mest relevante for dette prosjektet.

4.1 Animatronisk Nakkemodul fra CES

Informasjon og bilder er hentet fra [10].
Custom Entertainment Solutions (CES) har laget en høykvalitets robotnakke i realistisk størrelse som kan utføre menneskelignende hodebevegelser. For NTNU Cyborg kunne denne antakeligvis gjort en god jobb. Prisen er derimot så høy at alternative produkter eller evt. selvproduksjon blir mer aktuelt. Derfor har det også foreløpig blitt sett på som unødvendig å ta kontakt med produsent for mer informasjon om produktet. Under presenteres hovedspesifikasjonene, og Figur 4.1 viser et bilde av roboten.

- Servoer med magnetiske enkodere har lav slitasje og kan operere i årevis.
- Nakken har fem bevegelser som styres av de tre servoene. Dvs. kun tre kontrollinnganger er nødvendig.
- Mekanikken gjør at fem bevegelser kan utføres selv om det kun er tre servoer.
- Styring kan gjøres av hvilken som helst PWM servoregulator, mikrokontroller eller et enkelt RC system.

Pris: $ 2,800.00 USD + frakt og import



Figur 4.1: Industriell Animatronisk Nakkemekanisme fra Custom Entertainment Solutions

4.2 Robot Turrets fra Trossen Robotics

Informasjon og bilder er hentet fra [35] og [36] dersom annet ikke er spesifisert.
Selv om disse robotene kun har to frihetsgrader, altså én mindre enn det som er ønskelig (se Kapittel 3.2), er dem av så god kvalitet at de bør vurderes i tilfelle det blir for ambisiøst med tre frihetsgrader.

Med tanke på pris og andre spesifikasjoner som ble beskrevet i Kapittel 3, er det spesielt to roboter som kan være aktuelle, WidowX MX-28 Pan Tilt Kit og WidowX MX-28 Robot Turret Kit (se mer informasjon under). Begge disse består av to kraftige Dynamixel MX-28T servoer fra ROBOTIS som er laget spesielt for bruk i roboter. Et moment på opp til $3.1\mathrm{Nm}$, dvs. $31.6\mathrm{kg} \cdot \mathrm{cm}$[1], gjør at servoene fungerer fint med Kinect V2 som last, selv om

[1] Avhengig av inngangsspenning

anbefalt grense for dreiemoment er $20\% - 40\%$ av maksimum [37]. For mer informasjon om disse servoene, se delkapittel 5.3.2.

4.2.1 WidowX MX-28 Pan Tilt Kit

Dette settet består av servoer, kabler, braketter og rammer som er nødvendig for å bygge en Pan Tilt robot. For å kunne styre den kreves en mikrokontroller og/eller en USB2Dynamixel adapter for styring direkte fra PC.
Pris: $ 469.95 USD + frakt og import

Figur 4.2: WidowX MX-28 Pan Tilt Kit fra Trossen Robotics.

4.2.2 WidowX MX-28 Robot Turret Kit

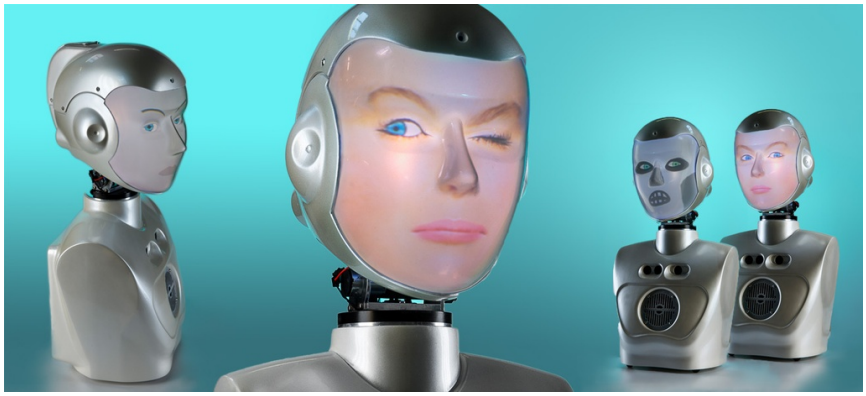
Dette er et komplett sett av en Pan Tilt robot, inkludert kontroller og strømforsyning i motsetning til settet over.
Pris: $ 599.95 USD + frakt og import

Figur 4.3: WidowX MX-28 Robot Turret Kit fra Trossen Robotics.

4.3 SociBot Mini fra Engineered Arts

Denne roboten er ikke direkte relevant i dette stadiet av prosjektet, mye pga. prisen og et lite modulariserbart produkt. Det som gjør den relevant her er teknologien med å prosjektere et 3D ansikt innenfra, som kan være interessant å implementere på NTNU Cyborg i fremtiden. Teknikken gjør at halve hode får et menneskelig preg, mens resten får et maskinelt design, og etter egen oppfatning er det nettopp slik kyborger ofte fremstilles, altså halvt menneske og halvt robot. Innkjøp av hele SociBot Mini er ikke nødvendigvis helt uaktuelt i fremtiden heller, men i dette stadiet av prosjektet er det foreløpig bestemt at Kinect V2 skal brukes som hode. Denne roboten er allerede beskrevet tidligere i forbindelse med NTNU Cyborg prosjektet i [25], men for å gjøre dette kapittelet mest mulig komplett er det valgt å presentere den her også, men all følgende informasjon og figur er nesten direkte kopiert fra denne tidligere rapporten.

> Socibot er en overkropp med hode og uten armer, og en sosial robot som er designet for interaksjon med mennesker. Enkle konversasjoner i et lite støyfullt miljø er mulig, ansiktsgjenkjenning, og den kan holde øye med og følge opp til 12 personer samtidig, selv om det er enda flere mennesker i området. Gestikulering som håndbevegelser og kroppsposisjoner kan også detekteres. Det prosjekterte ansiktet gjør det mulig å velge akkurat det ansiktet som ønskes i et animasjonssystem designet for utvikling. [...] Halsen er fullt bevegelig, som gjør den veldig levende. Den har flere I/O porter, inkludert USB, HDMI, lyd og ethernet. Prisen ligger på £9 500 GBP, men for kun hodet er prisen £3 500 GBP. Se SociBot i Figur 4.4.



Figur 4.4: SociBot Mini fra Engineered Arts

Kapittel 5

NAKKEMODUL: UTVIKLING AV HARDWARE

I Kapittel 4 ble det klart at ingen av de kommersielle produketene som har blitt vurdert er så gode alternativer at en selvproduksjon ville vært unødvendig. Det blir derfor gjort et forsøk på å utvikle en nakkemodul som tilfredsstiller de kravene og spesifikasjonene som beskrives i Kapittel 3.

5.1 Alternativer for mekaniske løsninger

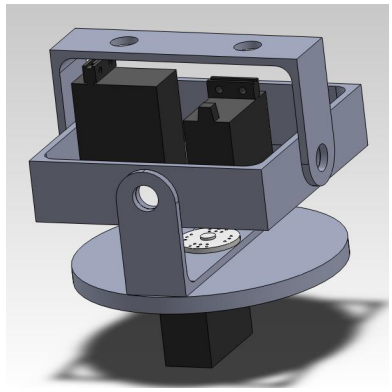
Selv om det er få kommersielle roboter som er veldig aktuelle for nakken til NTNU Cyborg så eksisterer det en del prosjekter og mekaniske løsninger tilgjengelige på nett, og disse kan brukes til inspirasjon ved utvikling av nakkemodulen. I denne seksjonen presenteres det et utvalg av slike prosjekter og mekanismer.

5.1.1 Enkel Animatronisk Nakkemekanisme

I artikkel [11] beskrives designprosessen av et animatronisk hode som bruker en nakkemekanisme som er aktuell for NTNU Cyborg. Selv om den er enkel så kan den tilfredsstille alle punktene i Kapittel 3 dersom den konstrueres riktig. Det er en sfærisk robot som bruker tre vanlige hobbyservoer som aktuatorer som vist i Figur 5.1. Konstruksjonen bør i tilfelle modifiseres noe, men det vil uansett være få og ukompliserte komponenter som ville gjort produseringen rimelig.

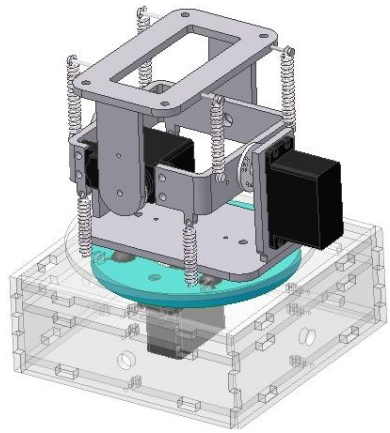
5.1.2 Nakkemekanisme med kompensasjon for tyngekraften

Denne teknikken er hentet fra artikkel [3] hvor en enkel sfærisk nakkemodul fikk montert springfjærer for roll- og pitch-bevegelsene. Disse fjærene har nullpunkt for utslag i oppreist stilling, og på grunn av gravitasjonskraften blir dette en ustabil likevektsposisjon.



Figur 5.1: Senebasert nakkemodul, [11]

End effectoren vil derfor i alle andre posisjoner bli påvirket av kraften fra fjærene som virker mot likevektspunktet. Servoene vil altså få hjelp til å motvirke gravitasjonskraften. Se Figur 5.2 som er hentet fra denne artikkelen.

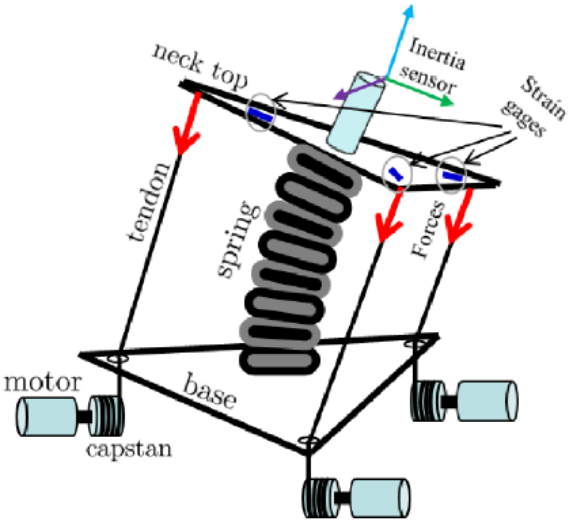


Figur 5.2: Nakkemodul med gravitasjonskompensasjon

5.1.3 Senebasert nakkemodul

Følgende prosjekt er beskrevet i artikkel [14]. Denne artikkelen hander mest om *styring* av nakkemekanismen, men kan likevel være en inspirasjon til NTNU Cyborg. Nakken består av en springfjær som er selve "ryggraden" og tre aktuatorer som drar i hver sin wire for å bevege på hodet, se Figur 5.3. Det som er aktuelt med denne mekanismen er likheten med hvordan en menneskenakke fungerer, og at den kan gjøre alle bevegelsene som beskrives i Kapittel 3. Å få til et stort bevegelsesområde kan derimot være vanskelig å få til. Lastvekten vil ikke være et problem med denne mekanismen med selv middels

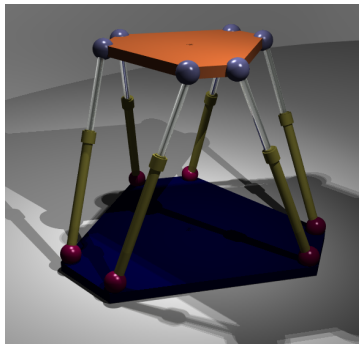
kraftige aktuatorer, ettersom wirene er festet en god avstand fra likevektspunktet slik at dreiemomentet her blir større. Kinematikken til denne roboten kan også bli komplisert, og springfjæra som brukes vil gjøre det vanskelig å bestemme posisjonen til end effectoren nøyaktig.



Figur 5.3: Senebasert nakkemodul, [14]

5.1.4 Parallellmanipulator

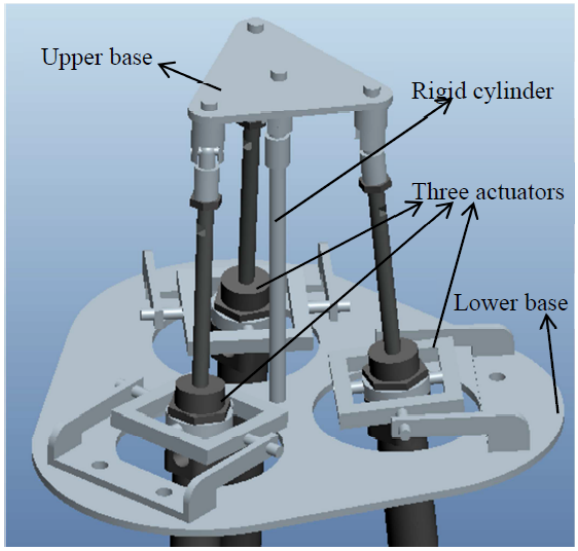
En parallellmanipulator er en mekanisme hvor flere manipulatorer eller lineære aktuatorer bærer samme plattform, som gjerne er end effectoren. En kjent type er Stewartplattformen vist i Figur 5.4, som er en hexapod, altså den bruker 6 aktuatorer, [60]. Dette er en interessant og imponerende mekanisme som kunne ha blitt laget slik at den tilfredsstiller egenskapene beskrevet i Kapittel 3. Det må uansett vurderes om andre enklere mekanismer også kunne gjort samme jobben.



Figur 5.4: Stewart Platform, [60]

Et eksempel på en lignende manipulator er beskrevet i artikkel [23]. Artikkelen beskriver også noen ulike nakkemekanismer, slik som her, før en dem velges. Den resulterende mekanismen ble en parallellmanipulator basert på en Stewart Platform med tre lineæraktuatorer, se Figur 5.5. Det ble i tillegg brukt en stiv søyle som ryggrad, på samme måte som springfjæra i prosjektet over, noe som gjør kinematikken enklere. Aktuatorene er lineære pneumatiske aktuatorer, altså aktuatorer basert på lufttrykk [61]. Disse krever en kompressor i kroppen til roboten, men vil redusere vekt og plass i nakken i forhold til elektriske motorer. Nøyaktigheten til elektriske aktuatorer vil være

vanskelig å oppnå med pneumatisk mekanikk, men dersom en lignende nakkemekanisme skal bygges er det uansett ikke noe problem å bruke elektriske aktuatorer i stedet.



Figur 5.5: Parallellmanipulator med pneumatiske lineæraktuatorer, [23]

5.2 Budsjett

Det har ikke vært et spesielt strengt budsjett for nakkemodulen, men det er selvfølgelig ønskelig å redusere utgiftene mest mulig. Veileder [54] anslo en pris på ca. 1000 kr per aktuator, i tillegg til eventuelle andre komponenter og utstyr. Selv om det ikke er et konkret budsjett så ble det et utgangspunkt for hvilken prisklasse det dreide seg om, og en innkjøpspris på minst 1000 kr per frihetsgrad skulle i hvert fall ikke være noe problem. Nakkemodulen skulle utvikles for å faktisk brukes på NTNU Cyborg, slik at en nakkemodul av dårlig kvalitet kunne få konsekvenser for resten av prosjektet. Det har derfor blitt forsøkt å balansere kvalitet og pris best mulig. Høyere pris for bedre kvalitet har blitt sett på som unødvendig dersom det ikke har vært behov for det.

5.3 Valg av mekanisme

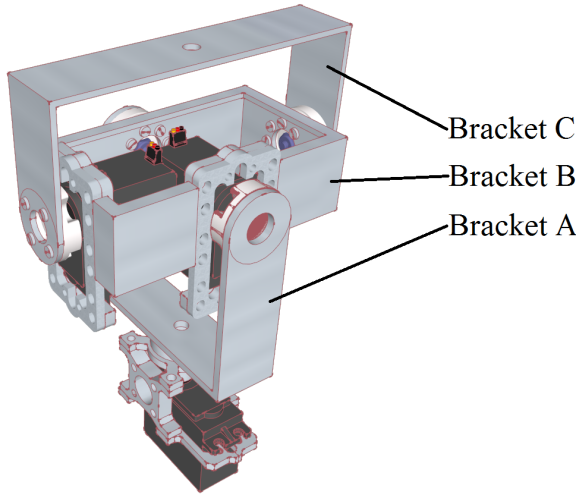
En god løsning til nakkemekanismen vurderes til å være en enkel 3 frihetsgraders (DOF: degrees of freedom) sfærisk håndleddsmanipulator, med nakkeroboten i seksjon 5.1.1 som utgangspunkt. Det er den enkleste av mekanismene som er undersøkt, men gjør likevel de samme bevegelsene som de fleste andre alternativene og kan bygges slik at den tilfredsstiller alle spesifikasjonene som er beskrevet i Kapittel 3. Avgjørelsen begrunnes med at det er bedre å utvikle en enkel og god robot som kan fullføres, istedenfor å begynne

på en kompleks robot som kanskje ikke kan fullføres og risikerer å få dårligere kvalitet. Selve mekanismen er heller ikke spesielt imponerende, men det viktige er uansett at bevegelsene er gode. Derfor er det valgt å optimalisere bevegelsene framfor å arbeide med en imponerende og mer krevende mekanisme som uansett ville gjort tilnærmet samme bevegelser.

Digitale servoer blir foreløpig valgt som aktuatorer til roboten. De er små, kompakte og relativt veldig sterke i forhold til størrelsen. Mange servoer har også posisjonsmåling innebygd, slik at dette ikke vil være nødvendig å installere selv. Nakkemodulen blir designet for å være kompatibel med kvalitetsservoer i mest mulig standard størrelse.

5.3.1 Førsteutgave av nakkeroboten

Det har blitt laget flere testdesign for å få mekanismen best mulig, hovedsaklig for å gjøre den så liten som mulig for å redusere dreiemomentene som kreves av servoene. Dette designet er førsteutgaven og ble kun laget som en 3D-modell, ettersom et annet design måtte lages fordi ønskede servoer ikke var kompatible med dette designet. Presentasjonen av denne førsteutgaven inkluderes likevel her fordi modellen kan fungere bra med servoene den er laget for, og andre prosjekter kan dermed ha nytte dette. Servoene som dette designet er kompatible med er servoer i standardstørrelse for RC og hobbyutstyr, noe som gjør den kompatibel med veldig mange servoer. Komponentene er listet opp under, og Figur 5.6 viser hvordan designet ble.



Figur 5.6: Førsteutgave av nakkeroboten

- 3 servoer, standard RC størrelse. 3D-modell fra Sketchup 3D Warehouse, [7].
- 1 Actobotics ServoBlock fra ServoCity. Den nederste servoen, yaw-servoen, er montert til denne servoblokken for å redusere kreftene på servoakslingen og å gjøre roboten mer stabil. 3D-modell fra nettsidene til Servocity, [46].

- 2 Actobotics Standard Hub Horn fra Servocity. 3D-modell fra nettsidene til Servocity, [47].
- 2 Actobotics Swivel Hub fra ServoCity. 3D-modell fra nettsidene til Servocity, [49].
- 2 Actobotics Standard Servo Plate A fra ServoCity. Disse er kun brukt som et utgangspunkt for å lage Bracket B. 3D-modell fra nettsidene til Servocity, [48].
- Bracket A er designet selv og skal selvproduseres.
- Bracket B er delvis designet selv og skal selvproduseres, der de samme festemulighetene som Servo Plate A har er en del av selve braketten. Servo Plate A skal altså ikke kjøpes.
- Bracket C er designet selv og skal selvproduseres.

5.3.2 Andreutgave av nakkeroboten, GrusBot 1.0

Etter en nærmere undersøkelse av Dynamixel robotservoer fra ROBOTIS ble det bestemt at en av disse servoene skulle brukes, men da måtte nakkemodulen designes helt på nytt. Med de mange monteringsmulighetene, kraft, open-source SDK bibliotek for utvikling, og en rekke parametre som kan skrives og leses, blir Dynamixel servoene laget spesifikt for robotikk. Kommunikasjonsprotokollen er enten TTL eller RS-485, hvor det brukes pakker bestående av servoens ID nummer og en rekke parametre for å kommunisere. Det gjør at servoene med sine unike ID numre kan kobles i serie med kun ett sett med kabler mellom hver servo, [42]. De produserer flere ulike familier av servoer til forskjellige formål, så derfor må det vurderes hvilke som egner seg best til nakkeroboten. Under presenters de to typene som vurderes til de mest aktuelle med tanke på egenskaper og pris.

Dynamixel RX-28:

Dette er en høykvalitetsservo og er den aller sprekeste i 28-størrelsen fra Dynamixel. Med et dreiemoment på opp til $3.7\mathrm{Nm}$ vil den kunne takle Kinectsensoren uten problemer. Både posisjon og hastighet reguleres ved hjelp av målinger med potensiometeret, i tillegg til muligheten til å sette *compliance* for å oppnå jevne og fine bevegelser. Den har hele 43 adresser til sammen i EEPROM og RAM som kan hentes ut og vurderes i hovedkontrolleren, der 30 av disse er skrivbare variabler. En innebygd mikrokontroller håndterer en rekke innebygde sensorer og regulering som gjør at hovedkontrolleren kan fokusere på andre oppgaver. Servoen bruker RS-485 kommunikasjon.
Listeprisen er 209.90 USD + frakt og import.
[29].

Dynamixel MX-28:

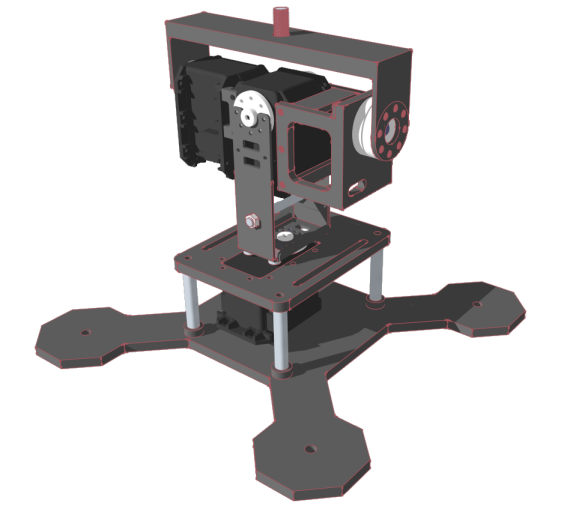
Dynamixel MX-28 er den nyeste generasjonen Dynamixel servoer fra ROBOTIS, med en oppgradering av mikrokontroller, kontaktløs magnetisk enkoder med 4 ganger så høy oppløsning enn hos RX-28, og 3 ganger så høy hastighet opp til 3 Mbps. Servoene har innebygd PID regulator og kan regulere både posisjon og hastighet, som en erstatning av compliance hos RX-28. Et dreiemoment på opp til $3.1\mathrm{Nm}$ bør også være tilstrekkelig til

å bruke Kinectsensoren som last. MX-28 finnes i fire utgaver: MX-28T, MX-28R, MX-28AT og MX-28AR. ”T” står for TTL-kommunikasjon (3 pin), ”R” står for RS-428 kommunikasjon (4 pin) og ”A” står for aluminium der disse typene har et aluminiumsskall og indre gjenger i monteringshullene[1]. Selve funksjonaliteten er den samme for alle. Prisene for ”T”, ”R”, ”AT” og ”AR” ligger på henholdsvis 219.90 USD, 225.90 USD, 239.90 USD og 249.90 USD, pluss frakt og import.
[37].

Både RX-28 og MX-28 vil kunne gjøre omtrent de samme oppgavene, der RX-28 vil kunne løfte noe mer, mens MX-28 vil kunne utføre oppgaven noe bedre på flere måter. Prisforskjellen er minst 10.00 USD. Servoen som ble valgt er Dynamixel RX-28, med følgende argumenter:

- RX-28 har større dreiemoment enn MX-28.
- RX-28 vurderes til å kunne utføre jobben i nakkeroboten godt nok, og mange av MX-28 sine bedre egenskaper vil antakeligvis ikke være nødvendig.
- Listeprisen til RX-28 er 10.00 USD lavere enn den rimeligste av MX-28 typen. Med tre servoer blir det merkbart billigere.
- En leverandør av RX-28 hadde to av disse servoene på tilbud, til 167.92 USD stk. Det resulterte i totalpris på 2×167.92 USD $+1 \times 209.90$ USD $= 545.74$ USD. Det vil si 113.96 USD billigere enn 3 stk MX-28T.

Sluttresultatet ble *gir-rull-stamp*-roboten, *GrusBot 1.0*. Etter mange justeringer, modifiseringer, og vurderinger av hvilke komponenter som skal kjøpes og hvilke som skal selvproduseres, ble roboten slik det vises i Figur 5.7. Detaljert informasjon om GrusBot 1.0 finnes i Kapittel 6 og de tilhørende vedleggende.



Figur 5.7: Andreutgaven av nakkeroboten, GrusBot 1.0

[1] For å slippe å bruke muttere.

5.4 Referanser til robotens 3D-modell

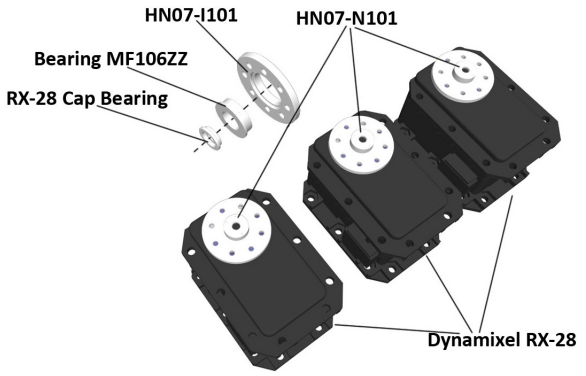
Flere av komponentene i 3D-modellen til GrusBot er hentet fra produsentenes nettsider og andre referanser, og har blitt brukt helt uredigerte eller som utgangspunkt til nye komponenter. Disse komponentene beskrives her og alle andre komponenter er designet helt selv.

5.4.1 Komponenter fra ROBOTIS

Følgende komponenter er hentet fra ROBOTIS sine nettsider, [40]. Ingen av disse er modifisert annet enn å endre farge.

Dynamixel RX-28 og servohjul

HN07-I101 med tilhørende hjullager er festet på baksiden av roll-servoen for å danne en hengsle til braketten. Figur 5.8 viser disse komponentene.



Figur 5.8: 3D-modell av Dynamixel RX-28 og servohjul

FR07-X101K Kryssrammer

Kryssrammene brukes for å montere roll- og pitch-servoene sammen. Se Figur 5.9 for 3D-modell av rammene.

Figur 5.9: 3D-modell av FR07-X101K Kryssrammer

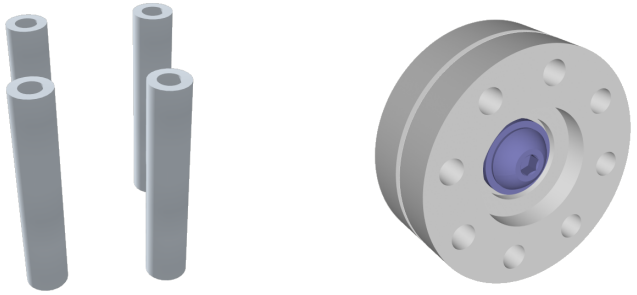
Roll Brakett

Braketten som støtter roll-servoen[2] er en modifisering av *FR07-H101K Hinge Frame* fra ROBOTIS. Modifiseringen vises i Figur 5.10.

Figur 5.10: Til venstre er den originale braketten, mens til høyre er den modifiserte

5.4.2 Komponenter fra ServoCity

Figur 5.11 viser fire *Actobotics 6-32 x 1.32" Tapped Aluminum Standoff* og en *Actobotics Swivel Hub*. Avstandsstykkene er tegnet selv ved å bruke dimensjonene fra leverandøren, [50]. Svivelhjulet er lastet ned som STEP-fil fra produsentens nettside, [49].



Figur 5.11: 3D-modell av Actobotics-komponentene fra Servocity

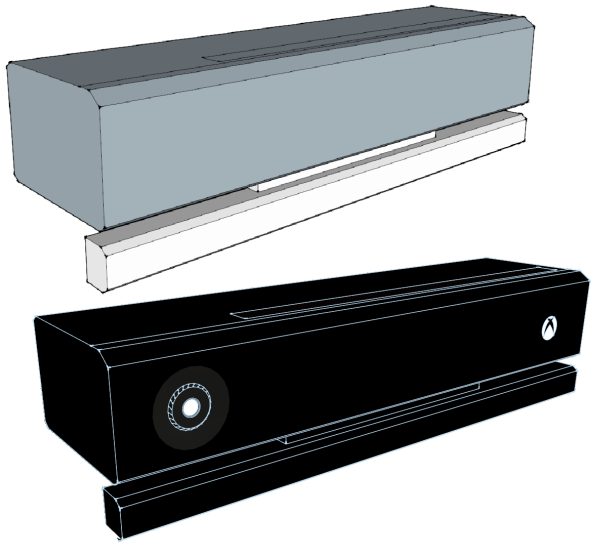
[2] Armen mellom yaw- og roll-servoene, se Figur A.1.

5.4.3 Komponenter fra 3D Warehouse

Følgende produkt er hentet fra Sketchup 3D Warehouse:

Microsoft Kinect V2

Se referanse [6]. Det ble brukt en modell fra 3DWarehouse som er enkel uten farger og design, men størrelsen som er riktig. Modellen ble modifisert til å få et utseende mest mulig likt det virkelige produktet. Se Figur 5.12 for sammenligning.

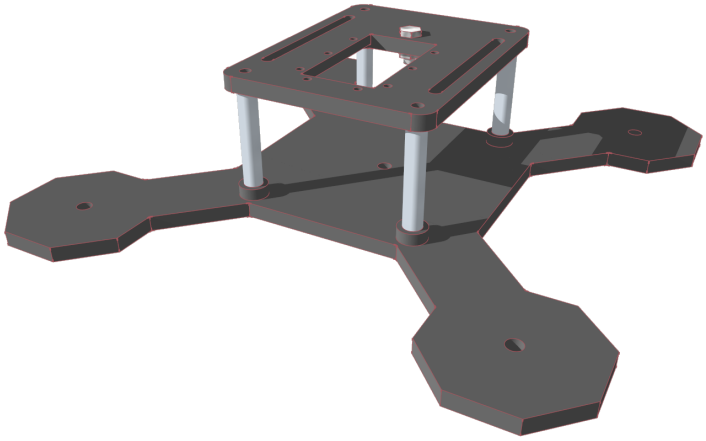


Figur 5.12: Kinect V2 i Sketchup. Øvre figur er hentet fra 3D Warehouse, mens nedre figur er modifisert.

5.4.4 Annet

Basen til roboten

Basen til nakkemodulen har blitt inspirert av RoboTurret fra Trossen Robotics [36], men noen vesentlige endringer er gjort. Se Figur 5.13 og sammenlign med Figur 4.3 fra Kapittel 4.



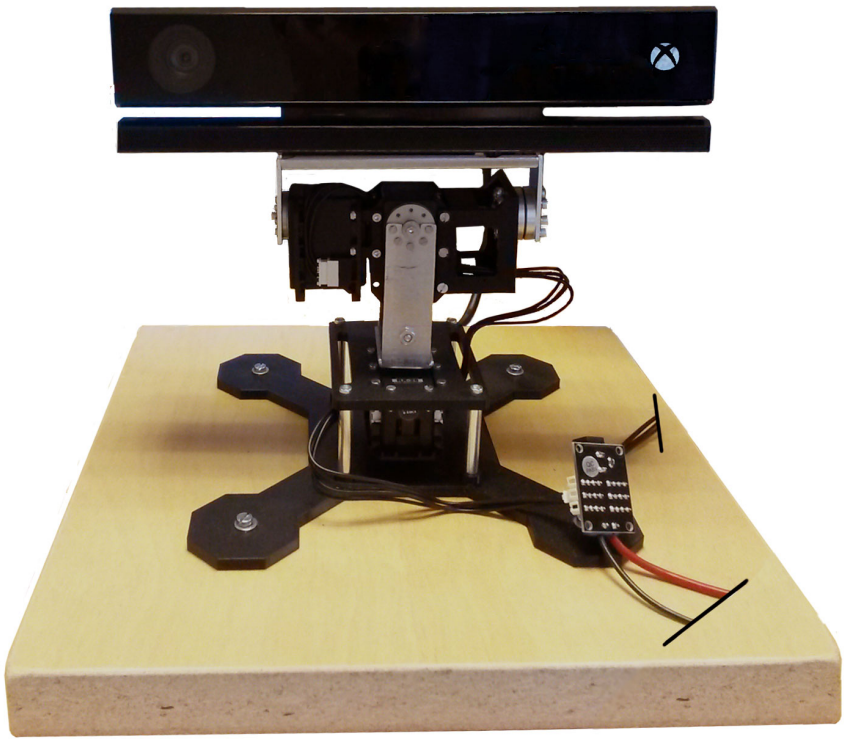
Figur 5.13: Basen som er produsert til nakkemodulen.

5.5 Programvare for 3D-tegning

Google Sketchup Basic versjon 15.3.331. Det har blitt installert prøveversjoner av extensions for importering av STEP-filer og IGES-filer, som er filformatene til komponentene fra produsentenes nettsider.

Kapittel 6

NAKKEMODUL: GRUSBOT 1.0



Figur 6.1: Ferdigmontert nakkemodul, festet på et stødig trebord

6.1 Introduksjon

Roboten som er NTNU Cyborg sin første nakkemodul er *GrusBot 1.0*. De foregående kapitlene beskrev utviklingsprosessen til denne roboten, mens dette kapittelet henviser til dokumentasjon, beskriver innkjøps- og produksjonsprosessen, og forslag og anbefalinger til videreutvikling. Til sammen beskriver dette kapittelet og tilhørende vedlegg alt som er nødvendig for å kunne bruke, reparere, vedlikeholde og videreutvikle denne roboten. Informasjonen skal også være tilstrekkelig nok for en eventuell reproduksjon av GrusBot. Denne rapporten inneholder et eget kapittel for videre arbeid 8, men det er forbeholdt The NTNU Cyborg. Derfor vil videre arbeid og foreslåtte forbedringer av GrusBot beskrives i dette kapittelet.

GrusBot 1.0 er ikke skreddersydd til NTNU Cyborg, men den er konstruert slik at den *kan* brukes av NTNU Cyborg. Den er laget for å være en allsidig robot som kan brukes til flere bruksområder, se Datablad i Vedlegg A. Dersom NTNU Cyborg skal oppgraderes med en annen nakkemodul i fremtiden kan GrusBot anvendes til andre prosjekter og dermed unngå tap av produkt. Ved behov for en robot som denne i andre prosjekter vil det også være mye å spare i utviklingskostnader ved å reprodusere GrusBot istedenfor å utvikle en ny robot.

6.2 Vedlegg tilhørende GrusBot

Følgende dokumenter er vedlegg til denne rapporten som inneholder informasjon om GrusBot.

6.2.1 Datablad

Databladet til GrusBot 1.0 finnes i Vedlegg A. Dette dokumentet beskriver egenskaper, spesifikasjoner og kinematikkresultatene.

6.2.2 Brukermanual

Vedlegg B er brukermanualen til GrusBot 1.0 som beskriver installasjon, tilkoblinger, anbefalt programvare, standardinnstillinger og noen viktige bemerkninger. Det anbefales sterkt å lese gjennom hele denne manualen før roboten tas i bruk.

6.2.3 API Dokumenasjon

Vedlegg C er en API dokumentasjon som gir informasjon om bruk av GrusBot SDK 1.0 biblioteket.

6.3 Innkjøp og kostnader

Tabell 6.1 inneholder alle de innkjøpte komponentene som GrusBot 1.0 består av. Alle priser er oppgitt i amerikanske dollar, USD, ettersom alle leverandørene bruker denne va-

lutaen. Frakt og import tas ikke med her på grunn av variasjoner avhengig av leverandør. Det kan ikke garanteres at alle produktene er best egnet, men det var disse som var tilgjengelige hos leverandøren på kjøpstidspunktet.

Produkter fra ROBOTIS [1]			
Vare	**Antall**	**Enhetspris**	**Sum**
Dynamixel RX-28 N101[1]	3	$209.90	$629.70
USB2Dynamixel Adapter	1	$49.90	$49.90
HN07 - I101 servohjul med kulelager	1	$14.10	$14.10
6 Port RX/EX Power Hub for strømforsyning	1	$7.95	$7.95
FR07-X101K Cross Frame Set	1	$31.90	$31.90
Robot Cable 4-Pin (RS-485) 200mm 10 pack	1	$19.40	$19.40
Subtotal			**$752.95**
Produkter fra ServoCity [2]			
Actobotics Swivel Hub	1	$6.99	$6.99
Actobotics 6-32 × 1.32” Tapped Standoff[2]	4	$0.88	$3.52
5/16” 6-32 Socket Head Screw, 25 pack[3] (for Swivel Hub)	1	$1.79	$1.79
3/8” 6-32 Socket Head Screw, 25 pack (for avstandsstykkene)	1	$1.89	$1.89
Subtotal			**$14.19**
TOTALT			**$767.14**

Tabell 6.1: Innkjøpsliste til GrusBot 1.0

Denne listen er ikke den faktiske innkjøpslisten i forbindelse med NTNU Cyborg prosjektet, men produktene er tilsvarende. Det har blitt utnyttet at en leverandør hadde et midlertidig godt tilbud på RX-28 servoene, men det har også vært nødvendig å bestille fra flere leverandører for å få tak i alle komponentene, noe som resulterte i høyere total fraktkostnad.

Det har i tillegg blitt brukt en del skruer og andre materialer fra mekanisk verksted [5], så innkjøpsprisen på dette kan ikke estimeres her, men det vil antakeligvis ikke utgjøre en betydelig forskjell på totalprisen dersom de kjøpes inn.

[1] N101 er type servohjul som medfølger. Anbefales fremfor den gamle N1 typen.
[2] Det anbefales å heller bruke avstandsstykker med metriske mål istedenfor disse. Gjelder også gjengene.
[3] De 8 skruene som fester braketten til svivelhjulet må slipes ned 2.5mm, eller så må et sett skruer med lengde 3/16” kjøpes i tillegg.

6.3.1 Modifisering av innkjøpte deler

Noen av delene som ble kjøpt inn måtte modifiseres noe i ettertid, som beskrevet under.

Yaw-aktuator:

Akslingen til yaw-aktuatoren holder i utgangspunktet hele GrusBot oppe. For å minimere slark og kreftene som påføres akslingen ble det satt på en tynn skive i tillegg til den medfølgende støtteskiven, slik at servohjulet fikk skikkelig kontakt med overflaten på servoen og kan dermed avlaste akslingen. Siden dette kan skape mer friksjon ble det påført noe fett gjorde friksjonen minimal. Stabiliteten ble uansett vurdert til å være viktigere enn lav friksjon, så derfor ble dette sett på som en enkel og god løsning.

Swivel Hub:

Det var høy friksjon i svivelhjulet og det kunne derfor ikke strammes godt til, noe som resulterte i slark i mekanismen. Svivelhjulet ble derfor åpnet, to tynne nylonskiver ble montert på og litt fett ble lagt på kulelageret. Dette reduserte friksjonen betraktelig og svivelhjulet kunne strammes helt til det omtrent ikke var noe mer slark.
Det ble montert en stabiliseringsskive bak servohjulet til *pitch*-servoen, slik at hele bredden på GrusBot økte med 0.5mm. Det gjorde at *Bracket 2* (se neste seksjon) ble 0.5mm for kort slik at Swivel Hub og servohjulet fikk ujevn belastning. Etter diskusjon med mekaniker [5] var det enkleste å slipe Swivel Hub ned tilsvarende denne økningen. Det er viktig å huske på dette ved en eventuell utskifting av Swivel Hub. Et alternativ er å lage en ny brakett som er 0.5mm lengre enn den originale.

Redusere lengde på skruer:

Ingen av de innkjøpte skruene var korte nok til å feste *Bracket 2* til Swivel Hub. Siden svivelhjulet ikke har metriske gjenger, ble det ikke funnet skruer på verkstedet som passet. Derfor måtte 8 av de innkjøpte *5/16" 6-32 Socket Head*-skruene manuelt slipes ned 2.5mm ved hjelp av verktøy lånt av mekanisk verksted [5].

Justering av kabellengde:

Ett sett med ledninger måtte forlenges og ett måtte forkortes, som beskrevet under.
Ledningene mellom *yaw*- og *roll*-servoene begrenset bevegelsesområdet til GrusBot siden de var for korte. Ledningene som originalt var 200mm lange ble derfor forlenget med 65mm ved å skjøte sammen to ulike Dynamixelkabler. Denne håndteringen av kabler ble vurdert til å gi en veldig liten risiko for at kablene hekter seg fast i noe eller bli klemt av mekanikken.
Ledningene mellom *roll*- og *pitch*-servoene var lengre enn nødvendig og ble derfor forkortet slik at GrusBot så mer ryddig ut. De 4P Molex-kontaktene til Dynamixelledningene var ikke kompatible med noen av kontaktene som ble funnet på verkstedet. På grunn av frakttid og ekstra kostnader ble ikke nye kontakter bestilt, så lengden på de 4 ledningene ble derfor redusert ved å fysisk fjerne hylsene som allerede satt på enden av hver ledning og å sette dem på igjen etter ledningen var kuttet av.

6.4 Produksjon

For tips til montering og installasjon, se brukermanualen i Vedlegg B.

Her beskrives komponentene som ble produsert og skreddersydd til GrusBot, og alt arbeid som ikke refereres til *mekaniker* [5] er utført selv. All produksjon foregikk med en 3D-printer og på et mekanisk verksted hvor det meste av arbeidet ble utført av en mekaniker, [5]. Bilder med dimensjoner og 3D-modeller av disse delene finnes i vedlagte mapper med samme navn som delene, under mappen *"Custom made components"*.

Komponentene som ble 3D-printet ble først laget i Sketchup, men denne programvaren klarte ikke å eksportere en god nok STL-fil, som er formatet 3D-printeren bruker. Dermed måtte detaljerte dimensjoner gis til mekaniker som kunne tegne komponentene på nytt i programvaren Solidworks. STL-filene som ble eksportert av Solidworks var bra, og disse filene sammen med IGS- og Solidworks-filene ligger i komponentmappene.

6.4.1 Frame

- Produksjon: 3D-printing
- Materiale: Svart ABS-plast
- Modifiseringer i ettertid:

    1. Det var nødvendig å lage et lite hull på siden for kabelhåndtering. Utført av mekaniker.
    2. For å bruke forsenkede skruer til å feste rammen med, ble det borret opp en liten forsenkning til hver skrue.

3D-modell av komponenten vises i Tabell B.1 i Brukermanualen i Vedlegg B.

6.4.2 Bracket 1

- Produksjon: Mekanisk Verksted
- Materiale: Aluminium
- Modifiseringer i ettertid:

    1. En sikkerhetsstang på 54mm med M4 gjenger ble montert av mekaniker tvers over armene til braketten som en fysisk sikkerhetsfunksjon for å unngå overrotasjon av roll. Den ble festet 20mm opp fra bunnen av braketten, slik at maksimal vinkelutslag for roll ble $44°$. 4 muttere, en på hver side av hver arm holder den fast og har også en støttefunksjon for braketten. Seinere ble det også festet en silikontube rundt stanga for at stoppemekanismen gir en jevn kraftfordeling og for å unngå harde sammenstøt. Ved en eventuell demontering og montering av sikkerhetsstanga må muttrene strammes til slik at sidene ikke bøyes, men står rett.

2. Dynamixel RX-28 har en dødsone på $60°$ mellom $300°$ og $0/360°$ hvor posisjonsmålingene er ugyldige. Siden servoen fysisk kan rotere kontinuerlig kan den av ytre påvirkninger havne i denne dødsonen. Derfor ble det installert en fysisk begrensing for yaw-rotasjonen. To M3 x 6mm stoppeskruer ble på den ene siden av braketten montert i hvert hjørnet 18mm fra servoens rotasjonsakse. Muttere fester skruene til braketten fra oversiden slik at skruehodene vil stoppe i en hindring som monteres til Top Base. Hodet til skruene ble slipt ned til de var 2mm tykke for å ikke treffe noe annet enn den ønskelige hindringen.

3D-modell av komponenten vises i Tabell B.1 i Brukermanualen i Vedlegg B.

6.4.3 Bracket 2

- Produksjon: Mekanisk Verksted
- Materiale: Aluminium
- Modifiseringer i ettertid: Nei.

3D-modell av komponenten vises i Tabell B.1 i Brukermanualen i Vedlegg B.

6.4.4 Top Base

- Produksjon: 3D-printing
- Materiale: Svart ABS-plast
- Modifiseringer i ettertid:

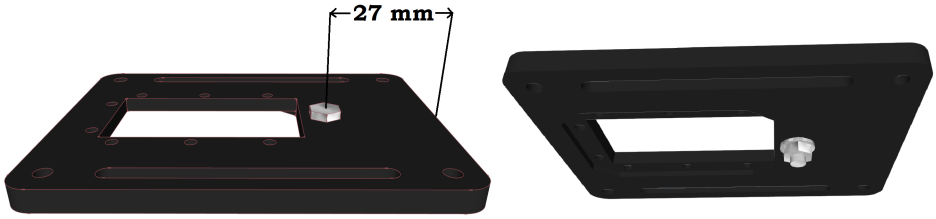
  1. Som beskrevet i oppgraderingene av Bracket 1 ble det installert fysiske begrensninger for yaw-rotasjonen. I denne komponenten ble det montert en skrue 27mm(tilsvarer 18mm fra yaw-aksen) fra enden som vil fungere som en hindring for skruene som er montert i Bracket 1. Skruen er 12mm lang inkludert hode med M5 gjenger, og er festet med en skive og en mutter på undersiden av Top Base. Hodet ble slipt ned til 2mm slik at kun skruene i Bracket 1 treffer den.
  2. For at Bracket 1 ikke skal treffe skruene yaw-servoen er festet med, ble det boret opp forsenkninger for å kunne bruke forsenkede skruer.

3D-modell av komponenten vises i Figur 6.2 der venstre side viser komponenten ovenifra mens høyre side viser den fra undersiden.

6.4.5 Bottom Base

- Produksjon: 3D-printing
- Materiale: Svart ABS-plast
- Modifiseringer i ettertid: Et 5mm hull i sentrum ble laget for å kunne feste Bottom Base med en ekstra skrue til plattformen. Dette reduserte svikt fra denne komponenten.

3D-modell av komponenten vises i Tabell B.1 i Brukermanualen i Vedlegg B.



Figur 6.2: Top Base oven- og underifra.

6.4.6 Plattform

- Produksjon: Trerester som mekaniker kuttet opp til ønsket størrelse.
- Dimensjoner: $30\text{cm} \times 30\text{cm}$
- Modifiseringer i ettertid: Pusset, boret hull til skruer og festet gummiknotter under for stødighet.

Plattformen er avbildet sammen med GrusBot i bildet på forsiden av dette kapittelet 6.

6.5 Kinematikk

6.5.1 Kinematisk kjedemodell

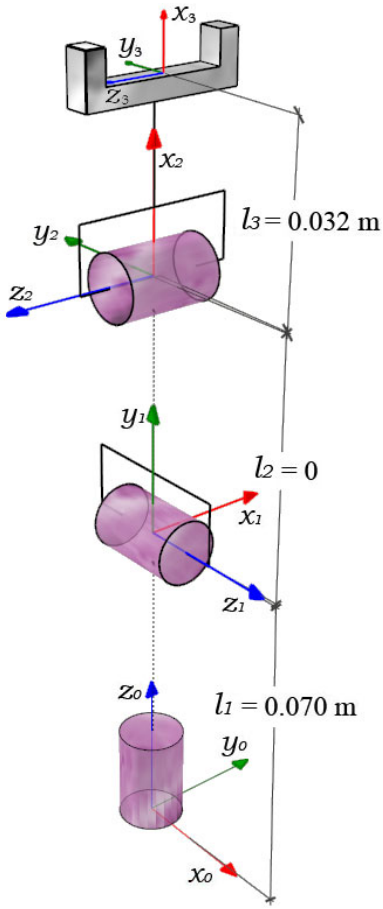
Kinematikken til en robot beskriver dens bevegelse uten å ta hensyn til krefter og dreiemoment som forårsaker bevegelsen. *Denavit-Hartenberg konvensjonen* (DH konvensjonen) er en mye brukt metode for å utlede en robots kinematikk, og skal benyttes her. Det starter med å finne DH parametrene til robotmodellen, som er et sett med parametre som korresponderer til hvert ledd i og hver arm i, slik at hele robotens kinematikk kan beskrives på bakgrunn av disse parametrene [53].

Figur 6.3 er en 3D-modell av GrusBot 1.0 som skal være til hjelp ved utledningen av robotens kinematikk. Koordinatsystemene eller *rammene* i figuren er plassert og orientert i henhold til DH konvensjonen og GrusBot sine egenskaper, og gjør dermed figuren til en grafisk representasjon av DH parametrene som er funnet i Tabell 6.2. Det ville muligens vært mer naturlig å plassere origo til ramme 0, o_0 i samme punkt som o_1, men for lesere uten bakgrunn i robotkinematikk er antas det at plasseringen gjort her er mer intuitiv og forståelig. Parametrene merket med "*" er variabler, de andre er konstanter.

Link	$\boldsymbol{a_i}$	$\boldsymbol{\alpha_i}$	$\boldsymbol{d_i}$	$\boldsymbol{\theta_i}$
1	0	$\pi/2$	d_1	$\pi/2 + \theta_1*$
2	0	$-\pi/2$	0	$\pi/2 + \theta_2*$
3	a_3	0	0	θ_3*

Tabell 6.2: DH parametrene til GrusBot 1.0

der $\{\theta_1, \theta_2, \theta_3\} = \{\psi, \phi, \theta\}$ og $\{d_1, a_3\} = \{l_1, l_3\}$ hendholdsvis for GrusBot. De fire parametrene a_i, α_i, d_i og θ_i kalles for henholdsvis *armlengde*, *armvridning*, *armforsyvning* og *leddvinkel*[4].



Figur 6.3: Kinematisk modell av GrusBot

6.5.2 Foroverkinematikk

Med foroverkinematikk er målet å finne posisjonen og orienteringen til end effector med hensyn på leddvariablene[5]. Foroverkinematikk tar utgangspunkt i den homogene transformasjonsmatrisen A_i for hvert ledd i, som er på formen

$$\boldsymbol{A_i} = \begin{bmatrix} \boldsymbol{R_i^{i-1}} & \boldsymbol{o_i^{i-1}} \\ \mathbf{0} & 1 \end{bmatrix} \tag{6.1}$$

[4]Oversatt fra engelsk: *link length*, *link twist*, *link offset* og *joint angle* [53]

[5]Vinkel for revolute (roterende) ledd eller lengde for prismatisk (lineærbevegelig) ledd.

der $\boldsymbol{R_i^{i-1}}$ er rotasjonmatrisen og $\boldsymbol{o_i^{i-1}}$ er koordinatvektoren.

Transformasjonsmatrisen A_i beskriver både posisjonen og orienteringen av ramme $o_{i-1}x_{i-1}y_{i-1}z_{i-1}$ relativ til ramme $o_ix_iy_iz_i$, med rotasjonsmatrisen som beskriver orienteringen og koordinatvektoren som beskriver posisjonen. Når DH konvensjonen brukes er A_i representert som et produkt av fire grunntransformasjoner:

$$\boldsymbol{A_i} = \text{Rot}_{\boldsymbol{z},\boldsymbol{\theta_i}}\text{Trans}_{\boldsymbol{z},\boldsymbol{d_i}}\text{Trans}_{\boldsymbol{x},\boldsymbol{a_i}}\text{Rot}_{\boldsymbol{x},\boldsymbol{\alpha_i}} \tag{6.2}$$

der $\text{Rot}_{\boldsymbol{i,j}}$ er rotasjon om i-aksen med vinkel j og $\text{Trans}_{\boldsymbol{i,j}}$ er translasjon[6] langs i-aksen med lengde j. Produktresultatet blir den homogene transformasjonsmatrisen

$$\boldsymbol{A_i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

der a_i, α_i, d_i og θ_i er DH parametrene for ledd i fra Tabell 6.2.

De homogene transformasjonsmatrisene beregnes ved å sette inn for disse parametrene. For å spare plass brukes $c_i = \cos(q_i)$ og $s_i = \sin(q_i)$.

$$\boldsymbol{A_1} = \begin{bmatrix} -s_1 & 0 & c_1 & 0 \\ c_1 & 0 & s_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{6.4}$$

$$\boldsymbol{A_2} = \begin{bmatrix} -s_2 & 0 & -c_2 & 0 \\ c_2 & 0 & -s_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{A_3} = \begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.5}$$

Legg merke til at $\sin(\pi/2 + \theta) = \cos(\theta)$ og $\cos(\pi/2 + \theta) = -\sin(\theta)$.

For å beskrive den totale transformasjonen fra grunnrammen, $o_0x_0y_0z_0$ til gjeldende ramme $o_ix_iy_iz_i$ benyttes følgende egenskap ved transformasjonsmatriser:

$$\boldsymbol{T_i^0} = \boldsymbol{T_1^0 T_2^1} \cdots \boldsymbol{T_i^{i-1}}$$

[6] Rettlinjet bevegelse

der $\boldsymbol{T_i^{i-1}} = \boldsymbol{A_i}$ som ble beregnet tidligere.

$$\boldsymbol{T_1^0} = \boldsymbol{A_1}$$

$$\boldsymbol{T_2^0} = \boldsymbol{T_1^0 A_2} = \boldsymbol{T_1^0 T_2^1} = \begin{bmatrix} s_1 s_2 & -c_1 & s_1 c_2 & 0 \\ -c_1 s_2 & -s_1 & -c_1 c_2 & 0 \\ c_2 & 0 & -s_2 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{T_3^0} = \boldsymbol{T_2^0 A_3} = \boldsymbol{T_1^0 T_2^1 T_3^2}$$

$$= \begin{bmatrix} s_1 s_2 c_3 - c_1 s_3 & -s_1 s_2 s_3 - c_1 c_3 & s_1 c_2 & a_3(s_1 s_2 c_3 - c_1 s_3) \\ -c_1 s_2 c_3 - s_1 s_3 & c_1 s_2 s_3 - s_1 c_3 & -c_1 c_2 & -a_3(c_1 s_2 c_3 + s_1 s_3) \\ c_2 c_3 & -c_2 s_3 & -s_2 & d_1 + a_3 c_2 c_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Foroverkinematikken til GrusBot beskrives altså med den homogene transformasjonsmatrisen,

$$\boldsymbol{T_3^0} = \begin{bmatrix} s_\psi s_\phi c_\theta - c_\psi s_\theta & -s_\psi s_\phi s_\theta - c_\psi c_\theta & s_\psi c_\phi & l_3(s_\psi s_\phi c_\theta - c_\psi s_\theta) \\ -c_\psi s_\phi c_\theta - s_\psi s_\theta & c_\psi s_\phi s_\theta - s_\psi c_\theta & -c_\psi c_\phi & -l_3(c_\psi s_\phi c_\theta + s_\psi s_\theta) \\ c_\phi c_\theta & -c_\phi s_3 & -s_\phi & l_1 + a_3 c_\phi c_\theta \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6}$$

der $d_1 = 0.070\text{m}$ og $l_3 = 0.032\text{m}$.

6.5.3 Invers Kinematikk

Invers kinematikk går ut på å finne leddvariablene med hensyn på end effectorens posisjon og orientering, og er veldig nyttig dersom posisjonen til end effector ikke kan måles ved hjelp av sensorer i aktuatorene [53]. Aktuatorene til GrusBot har interne potensiometre som måler posisjonen, men i enkelte situasjoner kan det være nyttig med eksterne posisjonssensorer. Et eksempel er dersom en Kinectsensor brukes som tilbehør, og det ønskes å bruke måleresultatene fra Kinecten til å bestemme posisjonen til GrusBot. Da må aktuatorenes vinkler beregnes utifra denne ønskede posisjonen.

En generell fremgangsmåte for å løse invers kinematikkproblemet er å løse følgende sett med ligninger med hensyn på aktuatorvariablene:

$$\boldsymbol{T_3^0} = \boldsymbol{H} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \\ r_{41} & r_{43} & r_{43} & r_{44} \end{bmatrix}, \tag{6.7}$$

der T_3^0 er den homogene transformasjonsmatrisen for foroverkinematikken, og H er faktiske posisjon og rotasjon til end effector.
Dette gir 16 ligninger, men det er bare 3 ukjente variabler. Ser at ved å kun bruke $r_{31}, r_{32}, r_{33}, r_{23}$ og r_{13} fåes et litt enklere ligningssett ettersom dette er de elementene som korresponderer til de enkleste elementene i T_3^0.

Løsningen kan da skrives som:

$$\psi = A\tan 2(r_{13}, -r_{23}) \tag{6.8}$$

$$\phi = A\tan 2(-r_{33}, \sqrt{r_{13}^2 + r_{23}^2}) \tag{6.9}$$

$$\theta = A\tan 2(-r_{32}, -r_{31}) \tag{6.10}$$

som er hentet fra resultatene for et lignende system i [3].

6.5.4 Singulariteter

GrusBot 1.0 er i singulær konfigurasjon i oppreist posisjon, altså når verken *roll* ϕ eller *pitch* θ har noe vinkelutslag. Roboten kan ha flere singulariteter, men da må maksimale vinkelutslag for *roll* og/eller *pitch* økes i software og evt. i hardware.

6.6 Problemer og anbefalte forbedringer

På grunn av begrensninger i budsjett og av tid, fikk ikke GrusBot alle de kvalitetene som var ønsket. I tillegg har det dukket opp ting underveis som vil være fornuftig å implementere i ettertid. Disse egenskapene og forbedringene beskrives her og det anbefales sterkt å vurdere nærmere om dette er noe som bør implementeres.

6.6.1 Forbedringer i programvaren

Software kan alltid forbedres, men her presenteres de forbedringene som vurderes til å være de viktigste i nærmeste framtid.

Feilmeldinger:

Med GrusBot SDK 1.0 kan det hende at samme feilmelding dukker opp flere ganger, slik at bruker må trykke ekstra mange ganger for å komme videre i programmet. Det påvirker ikke funksjonaliteten til roboten, men er uønsket av bruker. Det sjekkes kommunikasjon for hver enkelt servo, noe som er hensiktsmessig dersom feilen ikke ligger hos alle servoene og feilmeldingen vil fortelle hvilken servo det gjelder. Dermed kan samme feilmelding for de ulike servoene opptre dersom alle servoene har mistet kontakt. Hensikten med dette er tiltenkt, men samme feilmelding kan komme to ganger for hver servo, noe som vil si totalt seks feilmeldinger som brukeren må komme igjennom for å evt. avslutte programmet.

Nå printes statusmeldinger og feilmeldinger direkte til kommandovinduet. Disse bør kunne hentes ut, direktesendes eller lagres i fil for å kunne studere dem ved behov. Det vil også være en fordel å kunne overvåke disse i sanntid fra en annen PC via Internett. For å begrense informasjonen kan det vurderes om hvor mye informasjon skal lagres før det evt. kan overskrives av nye data.

Kommunikasjonssvikt:

En sjelden gang i tilfeldige situasjoner med GrusBot SDK 1.0 skjer det en kommunikasjonssvikt av ukjent årsak. Ettersom det også kan skje mens programmet står og venter på brukerinput er det mulig at problemet ligger i hardware, men programvaren skal heller ikke utelukkes pga. for lite testing. En mulig årsak kan være korte og midlertidige spenningsfall på inngangen, siden feilmeldingene er de samme som når strømmen slås av under kjøring. I tilfelle kan det prøves å justere toleransen for spenningsnivået i servoenes EEPROM fra Dynamixel Wizard.
Ved kommunikasjonssvikt blir brukeren spurt om å sjekke kabler og annet hardware, og ved et tastetrykk gå videre for å la programmet prøve å kommunisere på nytt, eventuelt "Esc" for å ikke prøve igjen og avslutte. Uansett hvor mange nye forsøk som blir gjort vil ikke kommunikasjonen etableres igjen, og programmet må startes på nytt. Det ble ikke nok tid til å undersøke dette nærmere, men problemet antas å være at i GrusBot SDK 1.0 opprettes det ikke ny kontakt med USB2Dynamixel etter en kommunikasjonssvikt. Løsningen kan dermed være som følger:

- Gå til mappelokasjonen "Software\GrusBot_10-20150702 BETA\GrusBot" og åpne grusbot.cpp
- Finn fram til definisjonen av funksjonen *GrusBot::Connections::communication_result_success(actuator)*, som skal være fra linje 31
- Gå til linje 58 hvor *newTry* settes til *true*
- Opprett en ny linje, til linje 59, og gjør et kall til funksjonen som initialiserer USB2Dynamixel og Dynamixel aktuatorene:
  GrusBot::Connections::initalize_robot(defaultSpeed, baudRate, usbPortNum)
- For å teste om dette fungerer kan følgende gjøres: kjør programmet, slå av strømmen, gå videre i programmet helt til en feilmelding dukker opp, slå på strømmen igjen og prøv å gå videre uten å trykke "Esc". Dersom feilmeldingen fortsatt dukker opp, trykk videre minst 6 ganger. Hvis kommunikasjonen fortsatt ikke er etablert bør det undersøkes nærmere for en annen løsning, evt. akseptere at programmet må restartes for å etablere kommunikasjonen igjen.

Automatisering av programvaren:

GrusBot SDK 1.0 krever nå input fra brukeren for å velge og gå videre i programmet. Det er nyttig ved testing og feilsøking, men når roboten er integrert med NTNU Cyborg bør alt gå automatisk, i hvert fall store deler av programvaren. Det anbefales derfor å modifisere en kopi av programvaren til å være mer egnet til bruk på NTNU Cyborg. Det skal ikke være mye arbeid å endre dette, ettersom brukerens målvinkler og målhastigheter kun brukes som argumenter i funksjonskall, slik at disse argumentene enkelt kan erstattes

med input fra en annen del av programvaren. Dessuten er enkelte av brukerens input unødvendige for funksjonaliteten og kan sløyfes. Det er funksjonene *getch()* og *std::cin* som brukes til å hente brukerinput. Ved å bruke søkefunksjonen i f.eks. Visual Studio kan disse enkelt bli funnet og fjernet, erstattet eller modifisert til ønsket håndtering av den korresponderende situasjonen.

6.6.2 Forbedringer av mekanikk og konstruksjon

Forbedringene som beskrives her er ikke for å løse noen kritiske problemer, men isteden anbefalinger eller forslag.

Fysiske vinkelbegrensninger:

Det er allerede laget mekaniske stopp for å begrense vinklene til yaw- og roll-rotasjonene. Det er også en naturlig fysisk begrensning for pitch-rotasjonen, men denne tillater et litt høyere vinkelutslag enn det som gjøres i programvaren. Derfor kan det være en fordel å lage en mekanisk stopp ved maks vinkelutslag for pitch. I tillegg kan roboten i situasjoner hvor den skal holde posisjonen i maksimalt vinkelutslag la servoene hvile end effectoren på disse stoppmekanismene, for å redusere strømbruk og slitasje på servoene. Det minimerer også risikoen for at servoalarmen går. Stoppemekanismen bør være justerbar, f.eks. en skrue som kan vris for å justere endepunktet til end effectoren.

Installere støtteramme til yaw-aktuator:

Det kan være hensiktsmessig å redusere kreftene som virker på akslingen til yaw-aktuatoren. Dette kan gjøres f.eks. ved å lage en kompatibel ”servoblokk” som ble brukt i designet til førsteutgaven av nakkeroboten i delkapittel 5.3.1.

Reprodusering med sterkere materialer:

Delene *Frame*, *Top Base* og *Bottom Base* er 3D printet med ABS plast. Det er et lett og sterkt materiale, men kreftene i roboten er så store at disse komponentene svikter og gir etter. Det gjelder spesielt *Top Base* som det anbefales å repsodusere i aluminium. *Bottom Base* påvirkes lite og behøver antakeligvis ikke å forsterkes. *Frame* svikter lite, men kjennes såpass spinkel ut at det kan være en fordel å reprodusere denne i aluminium dersom GrusBot kommer til å brukes til sitt maksimale. I prosjekter der det vil få store konsekvenser dersom GrusBot får en skade, anbefales det å bytte alle disse svake 3D printede komponentene på forhånd for å unngå mest mulig nedetid.

Øke dreiemoment:

Den enkleste måten å øke dreiemomentet på er å øke inngangsspenningen, men ettersom anbefalt inngangsspenningen er på 14.8V så bør ikke den maksimale spenningen på 18.5V brukes som en langsiktig løsning uten en nærmere undersøkelse av hvordan dette kan øke risikoen for skade og slitasje.

Pitch er den rotasjonen som antakeligvis vil kreve mest dreiemoment i de aller fleste situasjoner, og det er den rotasjonen som det er enklest å øke dreiemomentet til. For at GrusBot skal kunne utføre pitch raskere og tåle mer last, kan den 3D printede rammen, *Frame* (se Datablad i Vedlegg A for komponentliste) erstattes med en ekstra Dynamixel RX-28 servo. Det som må gjøres da er følgende:

- Anskaffelse av en Dynamixel RX-28 robotservo
- Anskaffelse av et ekstra sett med kryssrammene *FR07-X101K Cross Frame Set*
- Anskaffelse av en ekstra Dynamixel RS485 4-pin ledning og selve servoen
- Lage eller modifisere *Bracket 2* med korrekte hull, ettersom denne bracketen har ulike hull på hver arm
- Modifisere programvaren slik at rotasjonen *pitch* håndterer begge servoene nøyaktig likt, bare at den ene roterer i motsatt retning

Som beskrevet over under *Fysiske vinkelbegrensninger* vil det hjelpe noe å montere fysiske begrensninger for maksimalt utslag av pitchbevegelsen, slik at den aller tyngste posisjonen isteden kan hvile på disse stoppemekanismene.
En annen måte som muligens kan være mulig er å installere en mekanisme for kompensasjon av gravitasjonskraften slik som eksempelet i delkapittel 5.1.2. Med denne mekanikken er det springfjærer eller lignende som vil hjelpe servoene til å motvirke gravitasjonskraften slik at lasten påvirker servoene mindre. En viktig bemerkning med dette er at det kreves ekstra mye av servoene dersom roboten vris/roteres slik at tyngdekraften ikke virker parallelt med retningen til yaw-aksen.

Utskifting av aktuatorer:

Dersom det er nødvendig med nye servoer pga. skade eller lignende, eller dersom bedre egenskaper ønskes kan det være verdt å undersøke en litt mer solid, dyrere og bedre Dynamixel servo, *Dynamixel MX-28 AR*, se delkapittel 5.3.2 for mer informasjon om denne. Servoen har samme størrelse som RX-28, så den kan monteres direkte på GrusBot og bruke nøyaktig samme tilkobling. Dersom alle servoene skal utskiftes kan *Dynamixel MX-28 AT* med TTL kommunikasjon og 3 pins kabler brukes om ønskelig, men det krever en *6 Port AX/MX Power Hub* som støtter dette.

6.6.3 Andre viktige bemerkninger

Vibrering og ulyd

Dersom en relativt tung last som Kinect er montert, vil servoene jobbe konstant med å holde posisjonen til Kinecten. Det gir små vibrasjoner som kan gå utover kvaliteten på kameraet, og en slitsom lyd produseres. Etter søk på Internett viser det seg at det er normal oppførsel, men sammen med tilbehør som Kinect er det ønskelig å minimere denne effekten. Det kan forbedres ved å justere Punch, Compliance Slope og Compliance Margin (addressene 26-29 og 48-49 i RAM), men dette vil også påvirke presisjonen. For mer informasjon om disse parametrene, se i brukermanualen til servoene [38]. Merk at dette gjelder i hovedsak for roll- og pitch-servoene. Det er forsøkt å redusere disse effektene, ved

å justere Compliance Slope som beskrevet under Standardinnstillinger i Brukermanualen, B.7.

Alarm Shutdown

Dersom servoen må yte $20\% - 30\%$ av stall torque[7] over lengre tid vil servoen slå seg selv av, en funksjon som settes med Adresse 18 Alarm Shutdown, og står på som standard-innstilling. Denne funksjonen kan deaktiveres, men det frarådes da det er en sikker måte å unngå skade på servo og konstruksjon, og å holde strømforbruket nede. Dersom en servo får en slik Alarm Shutdown må strømmen slås av og på for å resette servoen.
De største belastningene vil være når lasten akselereres mot tyngdekraften, men det vil sjeldent bli et problem ettersom det tar kort tid å bevege lasten til en lettere posisjon. Det kan derimot være et problem når servoen står stille i en tung posisjon over lengre tid. En løsning på det kan være å forhindre for tunge posisjoner i programvaren, ved å senke vinkelbegrensningene. Hvor lang tid det tar før servoen får en Alarm Shutdown er varierende, ca. 30sec til 60sec, men det anbefales ikke å kun redusere tiden servoene holder de tunge posisjonene, selve posisjonen bør forhindres. Merk at vinkelbegrensninger må gjøres både i EEPROM og i programvaren til GrusBot. Det beste vil antakeligvis være å øke selve dreiemomentet slik at alle posisjoner, og gjerne bevegelser, ikke krever mer enn 20% av maksimalt dreiemoment. Se i seksjonen over for forslag til å øke dreiemomentet.

[7] Servoens maksimale holdemoment. Kan økes ved å bruke høyere spenning.

Kapittel 7

SYSTEMINTEGRASJON

7.1 Arkitektur

Et forslag til den overordnede arkitekturen vises i Figur 7.1, som er et diagram valgt til å baseres på *distribusjonsdiagrammet* fra UML 2 [12]. Diagrammet viser hvordan de ulike modulene er koblet sammen med tanke på dataoverføringer, og det viser hvor forskjellige typer software er tenkt til å kjøres fra. Figuren av IRIS mekanismen er hentet fra [52] og figuren av robotbasen er et 2D bilde av robotens 3D-modell som er hentet fra Adept sine nettsider [4].

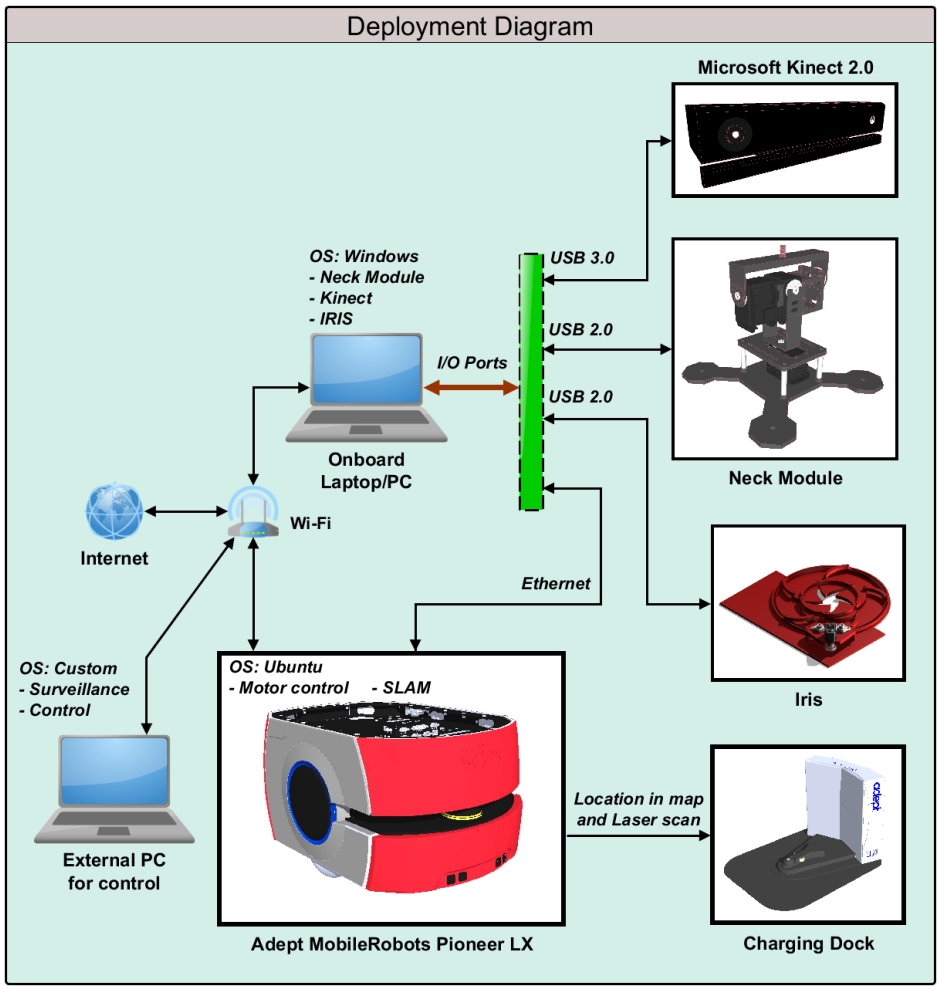
7.2 Strømforsyninger

Batteriet til Pioneer LX vil foreløpig være eneste strømkilde til hele NTNU Cyborg så selve robotbasen og alt tilbehøret må dele på strømmen fra dette batteriet. Det er tre tilgjengelige strømutganger med regulert spenning på 5 VCD, 12 VCD og 20 VDC, og fire utganger med direkte batterispenning 22 - 30 VCD, [21]. For å kunne bruke de uregulerte utgangene med batterispenning er det nødvendig med en spenningsomformer for å få en forutsigbar og ønsket spenning. Det vil da være en fordel om omformeren er kompatibel med den maksimale strømmen for den aktuelle utgangen. Selv om enheten som skal bruke strømmen ikke kan trekke så mye strøm, vil det være bedre om muligheten er der dersom enheten skal skiftes ut til en som faktisk krever denne strømforsyningen.

7.3 Nakkemodul

7.3.1 Strømtilkobling

Som beskrevet tidligere vil GrusBot 1.0 være første utgave av nakkemodulen til NTNU Cyborg. Fra databladet til GrusBot i Vedlegg A står det at maksimal strøm er 3A og anbefalt inngangsspenning er 14.8V, men spenning mellom 12V og 18.5V også aksepteres. I

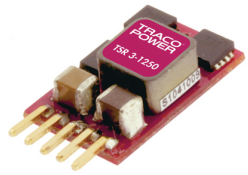


Figur 7.1: Et foreslått distribusjonsdiagram for NTNU Cyborg 1.0

brukermanualen Vedlegg B beskrives disse spesifikasjonene som et minimum for strømkilden for at roboten skal fungere skikkelig, mens det anbefalte er en strømforsyning på 5.7A for å håndtere de ekstreme tilfellene. Det har blitt undersøkt flere DC/DC omformere å kunne brukel robotbasens batteri. Det har da vært vanskelig å finne en som kan levere opp til 5.7A og 18.5V uten å se på store og dyre omformere som vil være en fordel å unngå. Omformeren må også tåle en inngangsspenningen som beskrevet i seksjon 7.2 over. Dersom det kan aksepteres maks utgangsstrøm på 3A og utgangsspenning på 15V, altså en effekt på 45W er mulighetene mye større. Da kan følgende produkt være aktuelt:

DC/DC-Omformer TSR3-24150 fra Traco Power:
Denne kan levere opp til 45W med 3A og 15V. Den kan også ha en inngangsspenning

på opp til 30V som holder akkurat til robotbasens batterispenning på $22 - 30$VDC. Utgangsspenningen må være minst 3V lavere enn inngangsspenningen, men det vil ikke bli noe problem her. Innebygd filter og kortslutningsvern, og effektivitet opp til 95%, [27]. Siden Pioneer LX har to utganger med batterispenning og maks 4A ville denne omformeren passet godt til en av disse utgangene.



Figur 7.2: DC/DC omformer fra Traco Power, [27]

7.3.2 Integrering med Kinect

Ettersom Kinectsensoren ikke har vært tilgjengelig før helt mot slutten av prosjektet så har det ikke vært mulighet for grundig testing av nakkemodulen sammen med Kinectsensoren. Det som må gjøres er å utvikle et program som kan kjøre både nakkeroboten GrusBot 1.0 og Kinect, ved å inkludere bibliotekene fra begge disse slik at de kan samhandle.

Kinectsensoren trenger også strøm. På AC/DC adapteren står det at den gir ut en spenning på 12V og strøm opp til 2.67A. Det er usikkert om sensoren faktisk bruker så mye strøm, så det bør undersøkes om det holder med 1A, for da kan den kobles direkte til den regulerte 12V utgangen på robotbasen som kan levere opp til 1A. Hvis den krever mer strøm så må en av de andre utgangene benyttes. Uansett må det utføres noe hardwaremodifisering av strømtilkoblingen for å kunne koble den til en DC spenning. En mulighet er å kutte kabelen mellom AC/DC adapteren og Y-koblingen hvor strøm og data integreres i samme kabelsamling, og koble denne på 12 VDC utgangen.

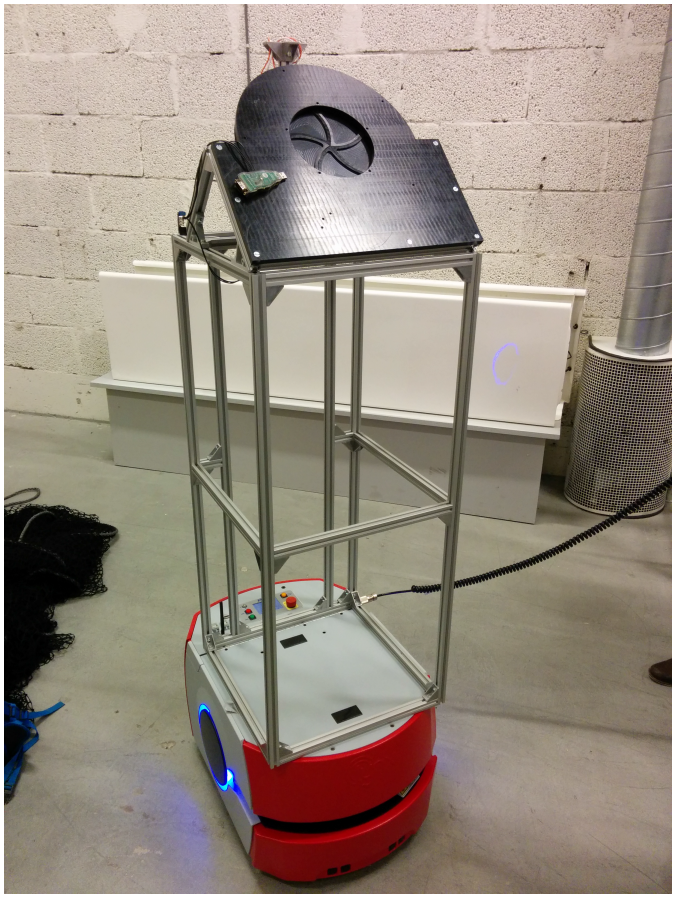
7.4 Ekstra datamaskin

Etter undersøkelser av systemkravene til Kinect V2 [18], så er det tydelig at den krever mer datakraft enn det som er tilgjengelig i den innebygde maskinen i robotbasen. Dette har i utgangspunktet vært kjent, og det har ikke blitt sett på som et problem siden det er fullt mulig å installere ekstra datakraft på NTNU Cyborg. Figur 7.1 tidligere i dette kapittelet viser hvordan denne datamaskinen kan installeres. I første omgang foreslås det en Kinect-kompatibel bærbar PC dersom det allerede er tilgjengelig, på grunn av enkelhet og ekstern PC-skjerm, tastatur og mus vil ikke være nødvendig. En bærbar PC kan også kobles til DC utgangene til robotbasen med en modifisert lader.

7.5 Kropp og IRIS

Det har vært noe samarbeid med en gruppe på 6 studenter i emnet TTK4850 - Eksperter i Team, hvor oppgaven deres skulle bli rettet mot NTNU Cyborg prosjektet. Det har vært noe oppfølging, men ellers har disse studentene gjort arbeidet selv.

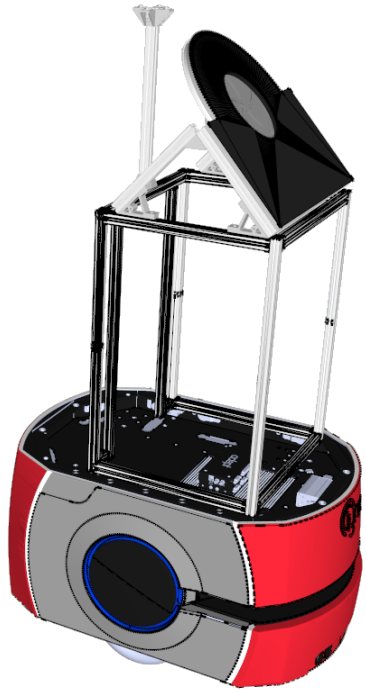
Oppgaven de endte opp med var å lage en kropp og en tilhørende IRIS mekanisme. En IRIS er en åpne/lukkemekanisme for hull bestående av en serie med plater som former seg til en sirkulær flate eller legger seg over hverandre når mekanismen er lukker seg. Det er en interessant mekanisme som er tenkt til å trekke oppmerksomhet fremfor å ha en konkret oppgave. Rammen som er laget skal være kroppen som binder de ulike modulene til NTNU Cyborg sammen og skal gi gode muligheter for montering av sensorer, utstyr og ekstra datakraft. Resultatet ble som vist i Figur 7.3. Det må åpenbart gjøres noen



Figur 7.3: Kropp og IRIS montert på robotbasen. Hentet fra [52].

justeringer av ”foten” til rammen for å kunne montere den på robotbasen, som kan ses på

bildet. I tillegg er kroppen altfor høy. Det anbefales å undersøke design litt nærmere slik at denne rammen i en tidlig fase kan modifiseres til danne et grunnlag for et godt design, men samtidig som at den er kompatibel med utstyret som skal monteres på den. Et forslag til en enkel modifisering av rammen vises i Figur 7.4. I denne figuren er modellen av rammen en modifisert versjon av 3D-modellen som ble laget av EiT-gruppa [52].



Figur 7.4: The NTNU Cyborg 1.0

Kapittel 8

VIDERE ARBEID MED NTNU CYBORG

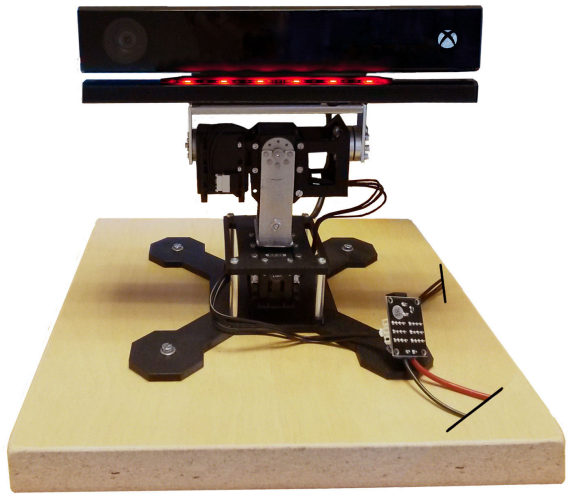
8.1 The NTNU Cyborg 1.0

Det ønskes en avduking av NTNU Cyborg 1.0 i begynnelsen av vårsemesteret 2016 [54]. Gjenstående arbeid av versjon 1.0 er hovedsaklig systemintegrasjon, ettersom de ulike modulene begynner å bli klare. Se Kapittel 7 for hva som konkret må gjøres. Problemer kan fort oppstå når modulene skal settes sammen til ett fungerende system, og dersom avdukingen skal skje innen rimelig tid bør det ikke fokuseres på så mye annet. Dersom det i tilfelle er mulighet for å gjøre noe ekstra foreslås det en enkel implementasjon av en munn, som vist i neste seksjon, og muligens gjøre en modifisering av Kinectsensoren.

8.1.1 Munn

En morsom effekt som kunne blitt implementert allerede på NTNU Cyborg 1.0 er en munn som vist i Figur 8.1. Munnen er bare en LED stripe som er festet på mikrofonrekken til Kinectsensoren, og ved å koble den sammen med f.eks. høyttalerene[1] vil den kunne blinke i takt med lyden som produseres. Dersom det er for tidlig for å implementere en stemme kan vanlig musikk i tilfelle brukes. Selv om dette er en nokså enkel funksjon og heller ikke spesielt imponerende, vil det likevel være med å løfte helhetsinntrykket av roboten og gi litt mer liv i den. En så kort LED stripe vil også være ganske rimelig, spesielt dersom det brukes en fast farge og bestilles fra Asia[2].

[1] Må antakeligvis bruke en adapter for å få riktig spenningsnivå.
[2] Basert på priser fra Ebay [9]



Figur 8.1: NTNU Cyborg Nakkemodul med en animert munn av LED-strips

8.1.2 Modifisering av Kinectsensoren

Kinect har en liten plattform med en mikrofonrekke. Denne er i utgangspunktet unødvendig for NTNU Cyborg, og for å redusere armlengden ut til massesenteret til Kinect kan det undersøkes om denne plattformen kan fjernes. Det ville forbedret motorikken ved at roll og pitch bevegelsene ikke krever så mye dreiemoment fra servoene, og hastighetsbegrensningene kunne blitt økt. Uten denne plattformen må i tilfelle Kinectsensoren festes på en annen måte enn den nåværende festemekanikken. På IFIXIT sine nettsider har dem demontert en Kinect V2 og detaljene rundt dette er presentert [13]. Det kan være et godt sted å begynne for å unngå unødvendig demontering selv. Den 28. juni 2015 kan følgende link brukes direkte:

Xbox One Kinect Teardown:
https://www.ifixit.com/Teardown/Xbox+One+Kinect+Teardown/19725

8.2 The NTNU Cyborg X.0

Det er ikke mye som er blitt bestemt konkret hva som skal gjøres videre annet enn videreutvikling til det spektakulære. Det som nevnes her er derfor hovedsaklig forslag til oppgraderinger og videre utvikling, dersom ikke annet er spesifisert.

8.2.1 Oppgradering av nakkemodulen

Det refereres til ”Problemer og anbefalte forbedringer” i Kapittel 6.6 for videreutvikling av nakkemodulen, både hardware og software.

8.2.2 Øke datakraft

Kyborgen vil etter hvert få mer og mer funksjonalitet som krever mer datakraft. Det kan da være hensiktsmessig å bruke eksterne datamaskiner til å kjøre store deler av programvaren via WiFi. Dette vil hindre vektøkning som følge av ekstra datamaskiner, det vil redusere strømforbruket, gjøre systemet mer modulariserbart, og mindre utstyr og verdi på roboten vil uansett være en fordel. Det kan også være hensiktsmessig å gjøre NTNU Cyborg mer og mer uavhengig av den innebygde datamaskinen i robotbasen, for at basen i størst mulig grad kan skilles ut som en egen uavhengig modul. Robotbasen vil uansett ha god bruk for datakraften den har selv, etter hvert som det antakeligvis blir gjort modifiseringer og implementert tilleggsfunksjoner i SLAM-systemet til basen.

8.2.3 Integrering av BabyX

Mark Sagars Baby X ble presentert i [25] og gjentas derfor ikke her. Det kan likevel nevnes at integrering av denne teknologien med NTNU Cyborg allerede har fått høy prioritet, der det ønskes å bruke denne modellen som ansikt og å være hjernen bak en stor del av interaksjonen med andre mennesker, [54].

8.2.4 Armer og manipulatorer

Foreløpig er det ikke tenkt at armer på NTNU Cyborg vil kunne gjøre en viktig oppgave. Likevel bør det på et tidspunkt installeres, ettersom det har noe med utseende og helheten å gjøre. Dersom det ikke blir definert en konkret og viktig oppgave for armene så trenger de ikke å være noe kompliserte eller dyre. Det kan til og med argumenteres for at helt statiske armer uten noen bevegelig funksjon vil være å foretrekke fremfor en ”amputert” robot. På sikt vil armer med en funksjon være det beste, hvor de kan brukes til å f.eks. vinke, peke, håndhilse eller klø seg i hodet med.

8.2.5 Funksjonalitet

Talegjenkjenning og tekst-til-tale funksjoner foreslås at får en høy prioritet ganske tidlig. Gjerne flere språk også. Det er dette som virkelig vil gjøre NTNU Cyborg sosial og spennende, og siden teknologien allerede eksisterer bør det ikke være en altfor ambisiøs oppgave.

En funksjon som vurderes til å være relativt enkel å implementere vil være lydlokalisering, altså å finne ut hvor en lyd kommer fra. Det vil kunne være veldig imponerende og gi en stor effekt dersom NTNU Cyborg kan snu seg, med hele kroppen og/eller hodet, mot den retningen lyden kommer fra. En noe mer avansert implementering vil da være å kunne filtrere lyden slik at kun interessant lyd resulterer i en handling, f.eks. dersom noen roper på NTNU Cyborg eller om en overraskende høy lyd dukker opp.

Et annet forslag er å implementere forskjellige humør i programvaren, der oppførselen varierer med de ulike humørene. Det kan f.eks. brukes vektede desisjonsalgoritmer i hvert humør som angir sannsynlighetene for ulike handlinger. Dersom humøret er ”trist”

kan sannsynligheten for å oppsøke andre mennesker være lavere enn dersom humøret er ”glad”, og å vise en trist oppførsel for å prøve å få andre mennesker til å ta kontakt med den isteden.

Det henvises også til en av de tidligere rapportene [16] som er skrevet i forbindelse med dette prosjektet hvor en rekke funksjonaliteter beskrives.

8.2.6 Estetikk

Utseende og design kan være avgjørende for at cyborger kan aksepteres i samfunnet. En humanoid robot/kyborg som NTNU Cyborg vil kunne gi sterke assosiasjoner og sammenlignes med andre naturlige skapninger. Små designdetaljer kan være forskjellen på assosiasjoner som et farlig rovdyr og en uskyldig valp. Det bør også vurderes om den skal ha et moderne utseende eller ha et hjemmesnekra preg med masse ledninger og mye synlig mekanikk. Et mer moderne og/eller menneskelignende design kan gjøre at forventningene til funksjonaliteten øker betraktelig slik at fallhøyden er stor. Uansett vil det være en fordel å begynne på et komplett design tidlig, slik at det blir lettere å utvikle moduler som passer med dette designet. For ingeniørstudenter og ansatte ved NTNU er antakeligvis funksjonaliteten til NTNU Cyborg det som vurderes til det viktigste, men ettersom dette prosjektet er ønsket å få oppmerksomhet i media kan ikke effekten av utseende undervurderes på noen måte.

Kapittel 9

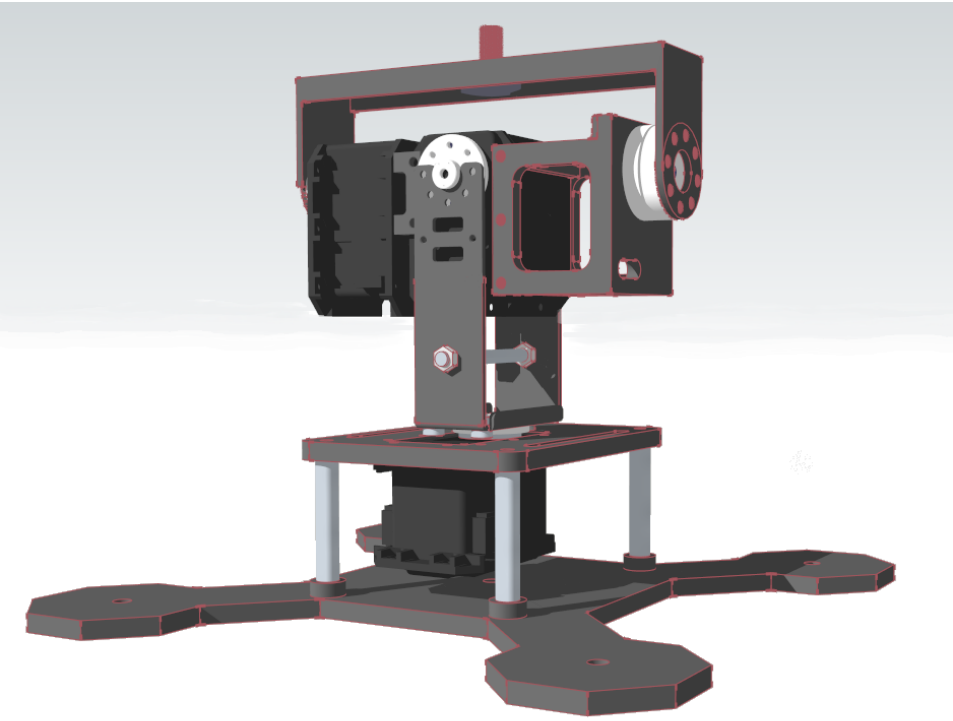
AVSLUTNING

Med dette prosjektet fullført så har NTNU Cyborg fått et solid grunnlag og veien videre til ferdigstillingen av *The NTNU Cyborg 1.0* har blitt betydelig kortere. Med de undersøkelsene som er gjort og det grundige arbeidet som er utført med fokus på kvalitet, har det resultert i minimale tilbakesteg iløpet av prosjektet, noe som sannsynligvis også er en fordel for etterkommere på NTNU Cyborg. Målsettingen for prosjektet om å jobbe med de robottekniske aspektene for å danne et godt fysisk grunnlag for NTNU Cyborg anses dermed som nådd.

En av de viktigste oppnåelsene med prosjektet har vært anskaffelsen av en mobil robotbase, en Pioneer LX fra Adept MobileRobots, som etter kort tid kunne utføre flere viktige oppgaver som er relevante for NTNU Cyborg. Med denne robotbasen har NTNU Cyborg allerede kommet på god vei. Et annet viktig resultat er utviklingen av en allsidig og enkel robot med 3 frihetsgrader, som fungerer ypperlig med Microsoft sin Kinectsensor som påmontert tilbehør. Det har også blitt arbeidet noen med integrering av disse modulene til å fungere sammen, men det var aldri et realistisk mål å fullføre dette. Arbeid med ystemintegrasjon bør få høy prioritet fremover, ettersom dette er vesentlig for ferdigstillingen av The NTNU Cyborg 1.0.

Vedlegg A

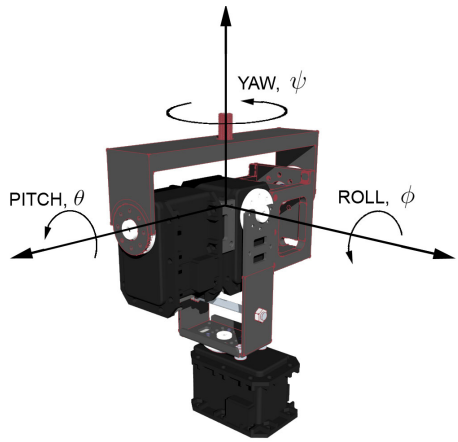
GRUSBOT: DATABLAD



GrusBot er en enkel og allsidig robot med tre kraftige Dynamixel aktuatorer som gjør den meget fleksibel. Innebygd regulator i hver servo gir god presisjon og konstant hastighet, i tillegg til 43 adresser for lesing og skriving av en rekke parametre.

A.1 Introduksjon

Navnet GrusBot stammer fra Eulervinklene gir, roll og stamp (eng. yaw, roll and pitch), som er de tre ulike rotasjonene til roboten når den står i utgangsposisjonen som i Figur A.1. GrusBot er en middels liten sfærisk (eng. spherical) robot med en relativt stor lastkapasitet i forhold til sin størrelse[1]. Konstruksjonen er sikker, lett og solid hvor dødgangen sitter tilnærmet kun i girmekanikken til servoene og i programvaren[2].



Figur A.1: Rotasjonsaksene til GrusBot

Det benyttes smarte Dynamixel RX-28 servoer fra ROBOTIS som alle kan styres fra en PC via en medfølgende USB adapter i opp til 1 Mbps. Servoene har innebygd regulator for både posisjon og vinkelhastighet, og kan i tillegg sende og motta en rekke parametre som posisjon, vinkelhastighet, moment og grenseverdier. Sikkerhetsmekanismer både i software og i hardware gjør at den er sikker i bruk og har minimal risiko for å beveges utenfor det tillatte bevegelsesområdet.

GrusBot er konkurransedyktig både på pris[3], allsidighet og kvalitet, sammenlignet med resultater fra [25] og Kapittel 4, hvor lignende roboter har mindre bevegelsesområde, er mer kompliserte eller har høyere pris. Se Kapittel 6.3 for innkjøpsliste til roboten.

A.1.1 Bruksområder

Her er noen eksempler på GrusBot i bruk. Kun fantasien setter grenser.

- Fjernstyrt eller automatstyrt kamera/Kinect
- Bli integrert som en nakkemodul til en humanoid robot
- Bli integrert som en håndleddsmanipulator
- Fjernstyring eller automatstyring av små lyskastere
- Leketøy for robotentusiaster
- Utvikling og forskning innen robotteknologi og lignende fag

(De fleste av disse bruksområdene krever ekstra programmering for å realiseres.)

[1] Etter sammenligning av flere typer servoer med lik og ulik størrelse.

[2] *Compliance* som er satt i programvaren resulterer i noe dødgang. Se under Standardinnstillinger i Brukermanualen B.7.

[3] Arbeidstimer ikke inkludert.

A.2 Tekniske spesifikasjoner

Alle spesifikasjoner som er merket med @ kan endres i ulik grad i programvaren og/eller i servoenes EEPROM via RoboPlus software B.6, men det må gjøres med forsiktighet i forhold til både hardware og software. Se under Standardinnstillinger i Brukermanualen B.7 for mer informasjon. Spesifikasjoner som påvirkes av inngangsspenningen gjelder for 14.8V dersom ikke annet er spesifisert. Dette er servoprodusentens anbefalte spenning.

GrusBot 1.0 Generelle Spesifikasjoner	
Inngangsspennning	Fra 12V til 18.5V. Anbefales 14.8V
Maks strøm @	3.0A
Standby strøm	150mA
Aksekrysningspunkt til TCP[4]	32mm
Grunnramme til aksekrysningspunkt[5]	70mm
Dreiemoment @	1Nm (14.8V), 1.51Nm (18.5V)
Aktuatorer	3 stk Dynamixel RX-28 fra ROBOTIS
Controller	PC via USB2Dynamixel adapter.
Datatilkobling	USB eller 4-PIN Molex for RS-485
Strømtilkobling	Terminalblokk eller DC barrel jack inngang
Kommunikasjonshastighet @	Opp til 1 Mbps
Materialer	ABS-plast og aluminium

Tabell A.1: GrusBot 1.0 Generelle Spesifikasjoner.

GrusBot 1.0 Kinematiske Begrensninger		
Rotasjonsakse	**Bevegelsesområde [rad] @**	**Vinkelhastighet [rad/s] @**
Yaw	$[-2.28, 2.28]$	2.56
Roll	$[-0.76, 0.76]$	2.09
Pitch	$[-1.23, 1.27]$	1.28

Tabell A.2: GrusBot 1.0 Kinematiske Begrensninger

[4] TCP = Tool Center Point, som tilsvarer midtpunktet til end effector.
[5] Avstand mellom servohjulet til *yaw* aktuator og aksekrysningspunktet.

A.2.1 Aktuatorer

Tre Dynamixel RX-28 robotservoer fra ROBOTIS brukes i GrusBot.

Dynamixel RX-28 Høydepunkter		
Inngangsspennning	**12 V**	**18 V**
Maks dreiemoment *	2.28Nm	3.70Nm
Vinkelhastighet[6]	$59.88\text{rpm} = 6.27\text{rad/s}$	$85\text{rpm} = 8.90\text{rad/s}$
Maks strøm	1.9A	
Standby strøm	50mA	
Oppløsning	$0.29°$	
Vekt	72g	
Dimensjoner	$50.6 \times 35.6 \times 35.5\text{mm}$	
Maks vinkelutslag	$300° = 5\pi/3\text{rad}$, eller kontinuerlig rotasjon	
Feedback	Ja, inkludert posisjon, hastighet og temperatur	
Antall Read-adresser	18 i EEPROM og 25 i RAM	
Antall Write-adresser	15 i EEPROM og 15 i RAM	
Kommunikasjon	Dynamixel Communication 1.0[7]	
Regulator	Compliance	

Tabell A.3: Dynamixel RX-28 Høydepunkter. Hentet fra [29] og [38]

** Maks dreiemoment gjelder for statisk hold. Stabile bevegelser er mulig for last mindre enn $1/5$ av denne verdien.*

Figur A.2: Dynamixel RX-28

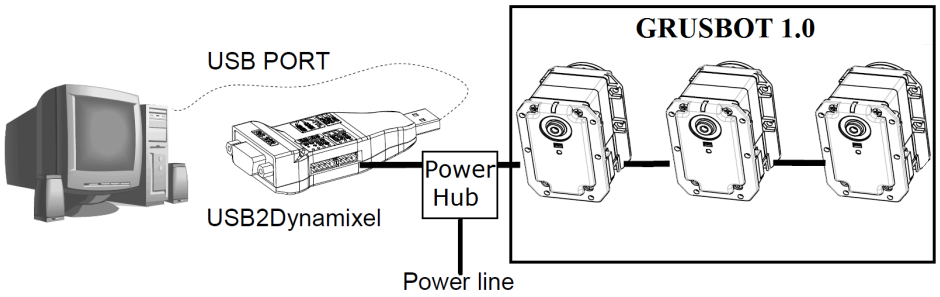
For mer info og brukermanual henvises det til ROBOTIS e-Manual [42]. Den 19. juni 2015 kan følgende link brukes for direkte tilgang.

Dynamixel RX-28 e-Manual v1.25.00:
http://support.robotis.com/en/product/dynamixel/rx_series/rx-28.htm

[6] Uten last.
[7] Asynkron Seriell Kommunikasjon med 8 bit, 1 Stop bit, og None Parity

A.3 Hardwarearkitektur

Figur A.3 av hardwarearkitekturen viser hvor enkelt oppkoblingen av GrusBot kan gjøres. Illustrasjon er en modifisert utgave av figuren hentet fra brukermanualen til USB2Dynamixel, [44].



Figur A.3: Oppkobling av GrusBot

A.4 Kinematikkresultater

Kinematikken til GrusBot baseres på Figur A.4. For mer detaljer, se Kinematikk i Kapittel 6.
Servoenes positive rotasjonsretninger er mot klokka (CCW) og negative rotasjonsretninger er med klokka (CW), se Figur A.1.

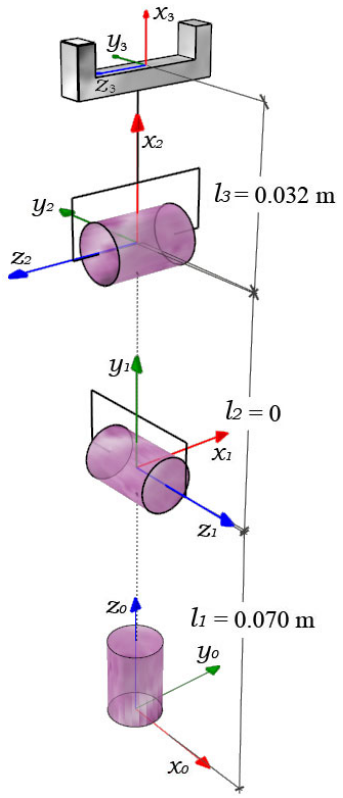
DH-parametrene:

Link	a_i	α_i	d_i	θ_i
1	0	$\pi/2$	d_1	$\pi/2 + \theta_1*$
2	0	$-\pi/2$	0	$\pi/2 + \theta_2*$
3	a_3	0	0	θ_3*

Foroverkinematikk:

$$\boldsymbol{T_3^0} = \begin{bmatrix} s_\psi s_\phi c_\theta - c_\psi s_\theta & -s_\psi s_\phi s_\theta - c_\psi c_\theta & s_\psi c_\phi & l_3(s_\psi s_\phi c_\theta - c_\psi s_\theta) \\ -c_\psi s_\phi c_\theta - s_\psi s_\theta & c_\psi s_\phi s_\theta - s_\psi c_\theta & -c_\psi c_\phi & -l_3(c_\psi s_\phi c_\theta + s_\psi s_\theta) \\ c_\phi c_\theta & -c_\phi s_3 & -s_\phi & l_1 + a_3 c_\phi c_\theta \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.1)}$$

der $d_1 = 0.070\text{m}$ og $l_3 = 0.032\text{m}$.



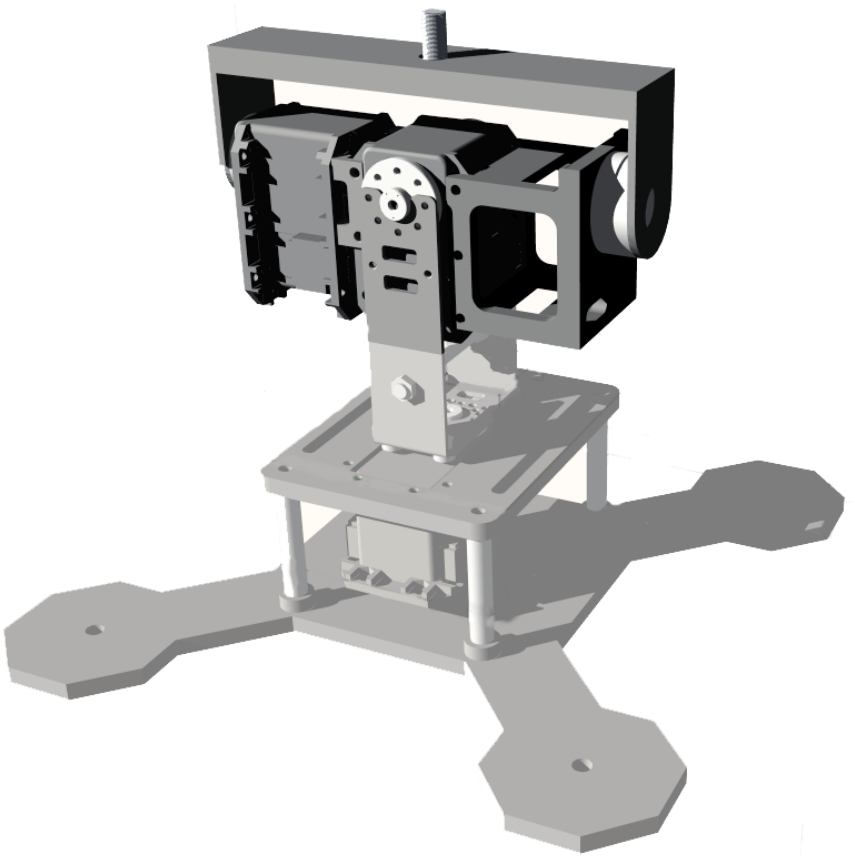
Figur A.4: Kinematisk modell av GrusBot

Singulariteter:

GrusBot 1.0 er i singulær konfigurasjon i oppreist posisjon, altså når verken *roll* ϕ eller *pitch* θ har noe vinkelutslag. Matematisk vil dette også gjelde når ϕ eller θ er lik π, men denne posisjonen er fysisk umulig for GrusBot. Roboten kan ha flere singulariteter, men da må maksimale vinkelutslag for *roll* og/eller *pitch* økes software og evt. i hardware.

Vedlegg B

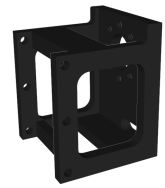
GRUSBOT: BRUKERMANUAL



I denne brukermanualen gis en oversikt over alle komponentene til GrusBot 1.0, og det gis informasjon om lastkapasitet, hva som må gjøres for å komme i gang med roboten og kjøre et eksempelprogram, hvordan et nytt softwareprosjekt kan opprettes, anbefalt programvare for PC, og informasjon om standardinnstillingene og hvordan disse kan endres ved behov. Tekniske spesifikasjoner presenteres i Databladet, Vedlegg A. For informasjon om hvordan GrusBot SDK 1.0 biblioteket skal brukes, se Vedlegg C for API Dokumentasjon.

B.1 Komponenter

Her samles alle komponentene for en god oversikt. Mer informasjon og modifiseringsdetaljer om hver komponent finnes i Kapittel 6.

Selvproduserte komponenter	
Frame	
Bracket 1	
Bracket 2	
Top Base	
Bottom Base	

Tabell B.1: Komponentliste: Selvprodusert

Innkjøpte komponenter		
3 stk. Dynamixel RX-28 N101		[40]
HN07 - I101 Idler Bearing Set		[31]
USB2Dynamixel adapter		[44]
FR07-X101K Cross Frame Set		[28]
6 Port RX/EX Power Hub		[32]
Robot Cable 4-Pin (RS-485) 200mm 10 pack		[45]
Actobotics Swivel Hub (svivelhjul)		[49]
4 stk. Actobotics 6-32×1.32" Tapped Standoff (avstandsstykker)		[50]

Tabell B.2: Komponentliste: Innkjøpt

Skruer[1]	
8 stk. 5/16” 6-32 Socket Head Screw	Montere Swivel Hub til Frame
8 stk. 3/16” 6-32 Socket Head Screw	Montere Swivel Hub til Bracket 2
8 stk. 3/8” 6-32 Socket Head Screw	Montere 4 standoff til Top Base og Bottom Base
18 stk M2 unbrakoskruer, 4mm*	Montering av braketter til servohjul.
10 stk M2.5 forsenkede skruer, 12mm*	Montere Frame til roll-servoen og yaw-servo til Top Base. Må ikke brukes på enden av servoene der kabelen kobles til.
4 stk M2.5 forsenkede skruer, 10mm*	Montere Frame til roll-servoen og yaw-servo til Top Base. Disse skal brukes på enden av servoene der kabelen kobles til.

Tabell B.3: Komponentliste: Skruer

** Dynamixel RX-28 inkluderer flere skruer og muttere, men skruene er for korte.*

B.2 Montere GrusBot

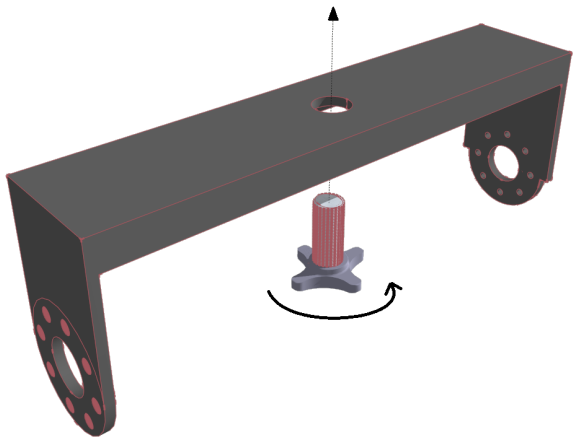
Monteringen av GrusBot er stort sett selvforklarende ved å kun se på figurene av den. Følgende tips kan uansett være til hjelp:

- Servohjul og Swivel Hub monteres *før* rammer, braketter og servoer settes sammen
- Kabler kobles til roll- og pitch-servoene *før* rammer og braketter
- Avstandsstykkene monteres til Bottom Base *før* den eventuelt festes til et bord
- Bottom Base festes til et eventuelt bord *før* Top Base monteres på avstandsstykkene
- Top Base og yaw-servo settes sammen *før* den monteres til avstandsstykkene og Bottom Base
- Stoppeskruene festes til Bracket 1 *før* braketten monteres på yaw-servoen.
- Bracket 1 festes til yaw-servo *før* stoppestanga og roll-servo monteres på denne braketten
- Roll- og pitch-servoene, Frame og Bracket 2 monteres sammen *før* roll-servoen monteres til Bracket 1
- PASS på at orienteringene til servoene er riktig i forhold til Figur A.1 i databladet

[1] Skruer som ikke nevnes her følger de innkjøpte komponentene.

B.2.1 Montere tilbehør

For å gi brukeren stor frihet til å velge hva som kan monteres er det ikke spesiallagd en monteringsmekanisme for én type tilbehør. Det brukes heller en enkel unbrakoskrue som heller ikke er i veien dersom brukeren ønsker å tilpasse monteringen til sitt eget ønske. Den medfølgende skruen kan likevel brukes for mange typer tilbehør, f.eks. et kamera, og monteringen gjøres ved å skru skruen fast til lasten gjennom hullet på 8mm i Bracket 2, slik det vises i Figur B.1. Ettersom det er litt trangt har det blitt modifisert en unbrakonøkkel til å ha kort nok hals til å komme til skruen.



Figur B.1: Montering av tilbehør på GrusBot.

B.3 Vedlikehold

Med jevne mellomrom bør skruene til servohjulene etterstrammes. Swivel Hub har som beskrevet i delkapittel 6.3.1 blitt påsmurt fett for å redusere friksjonen i kulelageret, og det bør derfor med jevne mellomrom undersøkes om kulelageret ikke er tørt og eventuelt smøre på mer fett dersom hjulene ikke ruller lett. Det samme gjelder stabiliseringsskivene under servohjulet til yaw-servoen, hvor det også er påsmurt fett for mindre friksjon.

B.4 Lastkapasitet

Hvor mye last som kan monteres på GrusBot avhenger av

- vekt,
- avstand til massesenter (C.M.),
- treghetsmoment,
- ønsket toppfart,
- ønsket akselerasjon,

- ønsket bevegelsesområde,
- GrusBots orientering
- og ytre krefter.

Dersom C.M. estimeres til å være på yaw-aksen (se Figur A.1 i databladet), dvs. i nullpunktet til pitch-aksen, vil det være aktuatoren for pitch-rotasjonen som får den største påkjenningen. Ettersom dette antas å være tilfelle i de fleste situasjoner, brukes kun pitch-rotasjonen som utgangspunkt for maksimal lastvekt her.

Ved høye krav til enkelte av punktene over vil det gå utover minst et av de andre punktene. F.eks. dersom det ønskes større akselerasjon vil en mulighet være å innsnevre bevegelsesområdet slik at massesenteret holder seg nærmest mulig utgangsposisjonen (oppreist stilling, se Figur A.1 i databladet).

Ved å bruke Ligning 3.1 kan det estimeres en maksimal lastvekt dersom C.M ligger på yaw-aksen og GrusBot er festet over eller under en horisontal flate[2].

$$\tau_{\text{p}} = m[g\sin(\theta) + a_{\text{CM}}](l_e + l_{\text{last}})$$

Ved å gjøre om ligningen med hensyn på masse m kan maksimal lastvekt beregnes:

$$m_{\text{max}} = \frac{\tau_{\text{p, max}}}{[g\sin(\theta_{\text{max}}) + a_{\text{CM}}](l_e + l_{\text{last}})} \tag{B.1}$$

der $\tau_{\text{p, max}}$ er aktuatorens maksimale dreiemoment, $g = 9.81\text{m/s}^2$ er tyngdeakselerasjonen, a_{CM} er ønsket akselerasjon for massesenteret til lasten, $l_e = l_3$ er avstanden fra pitch-rotasjonsaksen (aksekrysspunktet) til TCP, l_{last} er avstanden fra TCP til lastens C.M. i retning yaw-aksen[3], og θ_{max} er vinkelen til pitch som gir størst horisontale armlengde, og dermed størst dreiemoment. $\theta = 0$ i utgangsposisjonen til GrusBot. Merk at dette kun er en tilnærming og ikke nødvendigvis helt korrekt, men det gir en god indikasjon på hvor tung last som kan monteres.

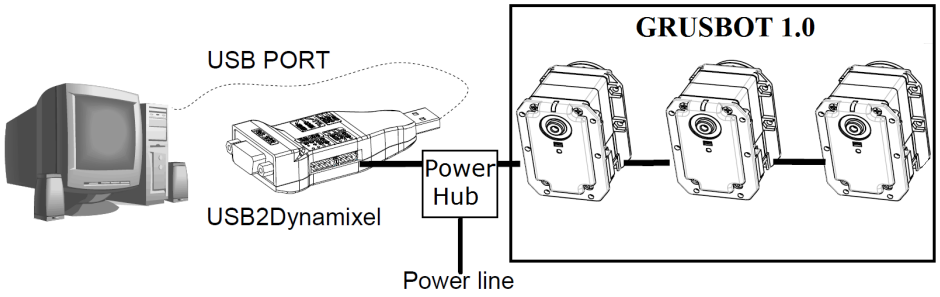
B.5 Oppkobling

GrusBot 1.0 bruker en USB2Dynamixel adapter fra ROBOTIS for styring direkte fra PC. Oppkoblingen gjøres som beskrevet i brukermanualen til USB2Dynamixel [44], men en kort oppsummering og spesielt rettet mot GrusBot presenteres her.

Figur B.2 hentet fra databladet til GrusBot viser hardwarearkitekturen for oppkobling. USB2Dynamixel adapteren kobles til USB-utgangen fra datamaskinen, og en 4-PIN Dynamixelkabel kobles til 4-PIN utgangen av adapteren. Pass på at bryteren står på riktig utgang på adapteren. Den andre enden av kabelen kobles til en 4-PIN Power Hub, som igjen er seriekoblet med servoene til GrusBot. Det er akkurat det samme hvilke av de 6 koblingsportene på Power Hub som benyttes. Strømforsyneren gir hele GrusBot strøm

[2] Lastkapasiteten er den samme om GrusBot står oppreist eller er opp ned.
[3] Vekten til end effector kan neglisjeres.

gjennom Power Hub ved å bruke terminalblokken eller DC barrel jack inngangen. Dersom en Power Hub ikke skal brukes kan strømmen kobles til den ledige 4-PIN utgangen til pitch-servoen, men det frarådes for å redusere antall ledninger som er koblet til den bevegelige roboten.



Figur B.2: Oppkobling av GrusBot

B.5.1 VIKTIG USB2Dynamixel Notat

Når USB2Dynamixel adapteren settes inn i en Windows PC vil Windows Update prøve å installere den nyeste FTDI driveren, FTDI 2.12.00. Denne versjonen av driveren har problemer med enkelte FTDI chips, inkludert den i USB2Dynamixel. Adapteren fungerer som den skal første gang, men neste gang den plugges i en PC registreres den ikke fordi FTDI driveren slettet Produkt ID nummeret første gang den ble brukt. Løsningen er å manuelt installere den eldre FTDI 2.10.00, og å stoppe Windows Update med en gang den forsøker å installere FTDI 2.12.00 [34]. Dette skal være nødvendig å gjøres kun én gang per PC. For detaljert beskrivelse av problemet og løsning, se hjelpesidene til Trossen Robotics [30]. Den 19. juni 2015 kan følgende link brukes direkte.

FTDI 2.12.00 notat fra Trossen Robotics:

http://learn.trossenrobotics.com/34-blog/140-ftdi-2-12-00-notice-robotis-usb2dynamixel-cm-530-and-ln-101-october-2014.html

B.5.2 Andre styringsmuligheter

ROBOTIS produserer også en rekke kontrollere til bruk sine aktuatorer, som kan brukes dersom det ønskes å styre roboten uten tilkobling til en PC. For Dynamixel RX-28 er det kontrollerene *CM-700* og *CM-900* som er kompatible. Begge disse gir også mulighet for trådløs kommunikasjon med ZigBee, [39].

B.5.3 Strømforsyning

USB2Dynamixel adapteren får strøm via USB, men dette er ikke nok til å forsyne servoene. GrusBot kan bruke opp til 3A med en inngangsspenning på $12 - 18.5$V der 14.8V er anbefalt, se databladet i Vedlegg A. For å GrusBot skal fungere skikkelig må altså strømforsyneren kunne levere minimum 12V og 3A[4]. Det anbefales likevel en justerbar spenning mellom 12V og 18.5V, og strømforsyning på 5.7A. Dersom det ønskes å presse servoene til det ytterste kan grensen for maksimalt dreiemoment til servoene økes (se neste avsnitt), men det krever enda mer av strømkilden. Databladet viser at hver servo maksimalt kan bruke 1.9A, som til sammen blir 5.7A, og da betyr det at strømkilden må kunne levere 5.7A.

GrusBot 1.0 har en default begrensning på 40% av maksimalt dreiemoment, se seksjon B.7 i denne brukermanualen. Det vil resultere i 40% mindre maksimalt strømforbruk, ettersom dreiemomentet er proporsjonalt med strømmen gjennom DC motoren[5], som vi kan se av dreiemoment/strøm-forholdet til en DC motor i Ligning B.2.

$$\tau_{motor} = K\phi I \tag{B.2}$$

der τ_{motor} er dreiemomentet til motoren, K er en fysisk konstant, ϕ er magnetisk flux som er tilnærmet konstant for motorer med permanente magneter, og I er strømmen gjennom armaturen i motoren, [53].

40% av det maksimale 5.7A er 2.28A. Ettersom dreiemoment/strøm-proporsjonaliteten kan avvike noe er det i Databladet valgt å skrive 3.0A som maks strøm for å ha noe slingringsmonn.

B.6 Programvare og systemkrav

Her presenteres og beskrives programvare som er nødvendig eller nyttig i forbindelse med bruk, feilsøking, vedlikehold og oppgradering av GrusBot. Alle programmene er gratis for alle.

B.6.1 Systemkrav

Hvilke krav som settes til PC eller maskin er avhengig av hvilke programmer som ønskes å benyttes, og det henvises derfor til informasjonen det henvises til for de aktuelle programvarene nedenfor.

[4] Det er usannsynlig at GrusBot klarer å yte maksimalt samtidig på alle servoene i en normal situasjon, men dersom det skjer er det lurt at strømforsyneren klarer å levere så mye strøm.

[5] Ettersom ingen av dokumentene fra ROBOTIS har noe informasjon om moment/strøm-forholdet til Dynamixel RX-28 ble en leverandør kontaktet [33]. Proposjonalitetsforholdet ble bekreftet som nokså godt i de aller fleste situasjoner, og at servoene bruker strømmålinger for å regne ut dreiemomentet til en hver tid.

B.6.2 RoboPlus v1.1.3.0

Gratis nedlastning kan gjøres fra ROBOTIS Software Downloads [41] (kun tilgjengelig på Windows) og installasjon og brukermanual finnes gratis i ROBOTIS e-Manual [42]. Den 19. juni 2015 kan følgende linker brukes for direkte tilgang.

Nedlastning av RoboPlus v1.1.3.0:
http://en.robotis.com/BlueAD/board.php?bbs_id=downloads&mode=view&bbs_no=1132559&page=1&key=&keyword=&sort=&scate=SOFTWARE

e-Manual v1.27.00:
http://support.robotis.com/en/software/roboplus_main.htm

I forbindelse med GrusBot er det hovedsaklig RoboPlus Dynamixel Wizard som er aktuell. Her kan Dynamixel servoer bl.a. få firmware oppdatering, sjekke status og skrive/lese konstante verdier i de 18 adressene i EEPROM som f.eks. servo ID, vinkelbegrensninger og alarmfunksjoner. I tillegg kan alle de 25 R/W adressene i RAM overvåkes og de skrivbare adressene kan bli satt for testing. I brukermanualen til Dynamixel RX-28 i ROBOTIS e-Manual [42] beskrives alle disse adressene. Merk at Dynamixel RX-28 bruker Dynamixel Communication 1.0 og har en baudrate på 1 Mbps (se B.7), og at disse egenskapene må være huket av i RoboPlus for å kunne finne servoene i søket.

B.6.3 USB2Dynamixel SDK v1.02

USB2Dynamixel SDK gir utvikleren tilgang til alle funksjonene som er nødvendig for å kunne bruke Dynaimxel RX-28. Nødvendig informasjon, nedlastning og eksempelprogrammer finnes gratis i ROBOTIS e-Manual. Her finnes også Dynamixel API Referanse. Den 19. juni 2015 kan følgende linker brukes for direkte tilgang.

Nedlastning av USB2Dynamixel SDK v1.02 og e-Manual v1.27.00:
http://support.robotis.com/en/software/dynamixel_sdk/usb2dynamixel.htm

API referanse v1.27.00:
http://support.robotis.com/en/software/dynamixel_sdk/api_reference.htm

USB2Dynamixel SDK finnes i både 32-bit og 64-bit. Bruk det som er egnet til det aktuelle prosjektet.

B.6.4 GrusBot SDK 1.0

GrusBot 1.0 kommer med et SDK bibliotek som er opprettet til et Static Library og er skrevet i C++. Det lille eksempelprogrammet Read/Write for én aktuator fra USB2Dynamixel SDK [**?**] har til en viss grad blitt brukt som utgangspunkt. GrusBot SDK 1.0 er et enkelt bibliotek som er laget for å kunne starte utviklingen med GrusBot raskt og intuitivt, hvor ingen av funksjonene fra USB2Dynamixel SDK er nødvendig for å kunne komme i gang. USB2Dynamixel SDK sine funksjoner kan likevel brukes direkte dersom det ønskes, bl.a.

for å få tilgang til de dynamixeladressene som ikke håndteres i GrusBot SDK. Se API Dokumentasjonen C og eksempelprogrammet i seksjon B.6.5 for mer informasjon om hvordan biblioteket kan brukes. Det anbefales å bruke samme kodestil ved videre utvikling for best mulig lesbarhet. Følgende liste oppsummerer de viktigste punktene:

- Dersom ikke annet er spesifisert under, brukes små bokstaver.
- Funksjoner skrives kun med små bokstaver, der understreker skiller ord.
  dette_er_en_funksjon()
- Variabler skrives med liten forbokstav. Store bokstaver skiller ord.
  detteErEnVariabel
- Konstanter skilles fra variable med en "k" foran navnet som begynner med stor bokstav.
  kDetteErEnKonstant
- Enumerasjoner og klassenavn skrives som variable, bare med stor forbokstav.
  DetteErEnKlasse
- Enumerasjonsverdier skrives med store bokstaver med understrek som skiller ord.
  enum EnumType $\{$*ENUM_VERDI_EN, ENUM_VERDI_TO*$\}$
- Deklarasjoner og definisjoner av konstanter kommer før variabler, som kommer før klasser, som kommer før funksjoner. Inne i selve klassen kommer funksjoner før variabler.
- Etter inndelingen fra punktet over, sorteres deklarasjoner og definisjoner i alfabetisk rekkefølge, hvor type prioriteres, deretter klasse, deretter funksjons- og variabelnavn.

Det er mulig å lage sitt eget SDK eller lage et kjørbart program direkte med USB2Dynamixel SDK dersom det er nødvendig. Se B.6.7 for tips til å opprette nytt prosjekt i Visual Studio.

B.6.5 GrusBot Eksempelprogram

Det er laget et eksempelprogram, *Neck_Module_Example* for å vise hvordan GrusBot SDK kan brukes som Static Library. Både eksempelprogrammet og GrusBot SDK er opprettet som to prosjekter i samme Solution med Visual Studio. Eksempelprogrammet inkluderer Header og Static Library fra GrusBot SDK 1.0, og GrusBot SDK inkluderer Header, Static og Dynamic Library fra USB2Dynamixel SDK v1.02. Det er derfor viktig å følge instruksjonene for bruk av USB2Dynamixel SDK som er gitt ovenfor for at GrusBot SDK skal fungere. GrusBot SDK 1.0 finnes kun i 32-bit, men dersom det er nødvendig med 64-bit kan dette endres i *Configuration Manager* ved å åpne prosjektet i Visual Studio, og å laste ned USB2Dynamixel SDK for 64-bit.

Eksempelprogrammet bruker GrusBot som en nakkemodul hvor et hode tilsvarende Kinect V2 antas å være montert på end effectoren. Det som gjøres i eksempelprogrammet er hovedsaklig å printe ut relevant informasjon til kommandovinduet og la bruker skrive inn ønsket posisjon for hver aktuator til GrusBot. Det anbefales å kjøre, teste og se over koden til eksempelprogrammet for å forstå hvordan og hvorfor det fungerer, dersom det ønskes å videreutvikle programvaren.

B.6.6 Microsoft Visual Studio 2013

Det anbefales å bruke Microsoft Visual Studio 2013 eller seinere, da GrusBot SDK v1.0 kun har blitt testet i dette programmet. IDE verktøyet er brukervennlig, tilpasningsdyktig, har mange nyttige funksjoner og har et meget godt etablert utviklingsmiljø[6]. I tillegg er Microsoft Kinect V2 et anbefalt tilbehør til GrusBot, og Kinect SDK for Windows krever bruk av Visual Studio [20]. Microsoft har flere utgaver av Visual Studio, men dersom det ønskes en gratis utgave anbefales *Microsoft Visual Studio Community 2013*, som også er nøyaktig det samme verktøyet som ble brukt under utviklingen av GrusBot. Denne versjonen har alle de nyttige funksjonene til Visual Studio Professional, og er optimalisert for bl.a. individuelle utviklere, studenter, og små team [19]. Den 19. juni 2015 kan følgende link brukes for å komme til nedlastningssiden til Visual Studio Community 2013.

Visual Studio Community 2013:
https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx

B.6.7 Opprette et nytt GrusBot prosjekt i Visual Studio

Her beskrives hvordan et nytt prosjekt kan opprettes til GrusBot fra starten av. Dersom det ikke ønskes å opprette et helt nytt prosjekt, kan det medfølgende eksempelprogrammet til GrusBot brukes som et utgangspunkt og redigeres slik det ønskes. Lignende veiledninger eksisterer, men enkelte krever noe mer detaljer spesifikt til GrusBot og forklares derfor her.
Merk at disse stegene ikke er eneste måten å opprette et nytt prosjekt på, men det er slik det ble gjort da GrusBot SDK med eksempelprogram ble opprettet og er derfor å anbefale.

Først må installasjonsinstruksene for USB2Dynamixel SDK følges, se delkapittel B.6.3, og Visual Studio 2013 må være installert.

Følgende er skrevet med [17] og [51] som utgangspunkt, men mer rettet mot GrusBot.

Opprett et Static Library for GrusBot:

Dersom det ønskes å bruke den eksisterende GrusBot SDK, kan følgende liste hoppes over og fortsett under *Opprett et kjøreprogram*.

- Åpne Visual Studio 2013.
- Velg **FILE** og **New Project** og sjekk at **.NET Framework 4.5** er rammeverket som brukes i vinduet som dukker opp.
- Til venstre huk av **Installed ->Templates ->Visual C++ ->Win32** og velg **Win32 Console Application**.
- Velg et prosjektnavn **Name**, f.eks. **GrusBot_Lib** og et **Solution Name**, f.eks. **My_Robot_Project**. Finn en egnet mappeplassering for **Solution** og klikk **OK**.

[6] Basert på egne erfaringer.

- **Win32 Application Wizard** dukker nå opp. Trykk **Next** >.
- Huk av **Static Library** og la alt annet være umerket. Klikk **Finish**.
- Det nye prosjektet skal nå automatisk være 32-bit og **Configuration Type** under **Configuration Properties** skal være **Static Library**.
- Opprett ønskede filer, f.eks. grusbot.cpp under **Source** og grusbot.h under **Header**.
- For å inkludere USB2Dynamixel SDK kan det være hensiktsmessig å kopiere hele USB2Dynamixel SDK mappen til **GrusBot_Lib** mappen for så å bruke makroer og relative filplasseringer for å inkludere de nødvendige filene.
- Pass på å bruk 32-bit versjonen av USB2Dynamixel SDK.
- Høyreklikk på prosjektnavnet og velg **Properties**. Gå til **Configuration Properties ->VC++ Directories**. Endre **Library Directories** og legg til filplasseringen $(ProjectDir)\USB2Dynamixel SDK\import for at Visual Studio skal finne **dynamixel.lib**.
- For at kjøreprogrammet også skal ha direkte tilgang til USB2Dynamixel SDK må **dynamixel.h** inkluderes i headerfilen til **GrusBot_Lib**, ellers i sourcefilen. Her kan det også brukes relativ filplassering:
  #include "USB2Dynamixel SDK\import\dynamixel.h"
- Nå skal alt være klart til å programmere et Static Library for GrusBot. For å teste biblioteket uten et tilhørende kjøreprogram kan det lages en main-funksjon i .cpp-filen og **Configuration Type** under **Configuration Properties** kan *midlertidig* settes til **Application**.

Opprett et kjøreprogram:

- Opprett et nytt prosjekt som over, men istedenfor å opprette en ny **Solution**, velg **Add to solution** og velg **My_Robot_Project** som ble opprettet over. Prosjektnavn kan f.eks. være **Robot_Console**.
- **Win32 Application Wizard** dukker nå opp. Trykk **Next** >.
- Huk av **Console Application** og la alt annet være umerket. Klikk **Finish**.
- Noen ferdiglagde filer er nå opprettet i prosjektet. Alle disse kan slettes dersom denne malen ikke ønskes å brukes.
- Høyreklikk på prosjektet **Robot_Console** og velg **Properties ->Common Properties ->References** og klikk **Add New Reference**. Biblioteket **GrusBot_Lib** som ble opprettet over skal nå være tilgjengelig her. Velg biblioteket og trykk **OK**.
- Gå til **Configuration Properties ->VC++ Directories**. Endre **Include Directories** og legg til ny mappeplassering og velg mappeplasseringen til headerfilen til **GrusBot_Lib**. I dette tilfelle er det $(ProjectDir)..\GrusBot_Lib

- Under **VC++ Directories** må også **Library Directories** endres. Legg til mappeplasseringen for .lib-filen som ble generert for **GrusBot_Lib**. Her vil det være $(ProjectDir)..\Debug
- Under **Configuration Properties ->Linker ->Input** må **dynamixel.lib** fra USB2Dynamixel SDK legges til i **Additional Dependencies**. I dette eksempelet vil riktig path være $(ProjectDir)..\GrusBot_Lib\USB2Dynamixel SDK\import\dynamixel.lib
- Sett **Robot_Console** prosjektet som oppstartsprosjekt. Høyreklikk på prosjektet og velg **Set as StartUp Project**
- Inkluder headerfilen til **GrusBot_Lib** i enten .cpp-filen eller .h-filen til **Robot_Console**.
- Nå skal alt være klart til å programmere et kjøreprogram for biblioteket **GrusBot_Lib**.

Merk at filplasseringene beskrevet over ikke nødvendigvis gjelder i alle tilfeller, og at de kun brukes som eksempel i denne veiledningen.

B.6.8 Dependency Walker

Dette programmet er ikke nødvendig, men ved feilsøking kan det være nyttig å ta i bruk. Det som gjøres er å bare åpne den genererte programfilen (.exe) i Dependency Walker. Ved bruk av Visual Studio finnes denne vanligvis i mappen *Debug* i katalogen Solution. For eksempel kan det undersøkes om alle deler av programmet er 32-bit (x86) eller 64-bit (x64) ved å se i kolonnen til *CPU* etter programfilen er lastet inn. Den 19. juni 2015 kan følgende link brukes for å komme direkte til til nedlastningssiden.

Dependency Walker:
http://www.dependencywalker.com/

B.7 Standardinnstillinger i programvaren

GrusBot 1.0 kommer med enkelte standardinnstillinger som er optimalisert for denne, og disse beskrives i denne seksjonen. Det inkluderes også informasjon om hvordan innstillingene kan endres slik at GrusBot kan tilpasses individuelle prosjekter, men det er i tilfelle viktig å gjøre dette med varsomhet. Innstillingene gjelder standardverdier som er satt i hver servo sin EEPROM, og hva som initialiseres når GrusBot SDK 1.0 startes. Dersom en innstilling endres i en servos EEPROM er det viktig å undersøke koden som skal brukes, og modifisere den eller sjekke at den er kompatibel med de nye innstillingene.

Alle verdier som settes i EEPROM er gjort via Dynamixel Wizard i RoboPlus programvaren, og alle tall skrives i titallssystemet og ikke det heksadesimale. Enheten dxl brukes for å beskrive verdiene Dynamixel RX-28 opererer med, som går fra 0 til 1023.

B.7.1 ID nummer

Fra produsenten har alle Dynamixel RX-28 et standard ID nummer på 1. For at flere av disse servoene skal fungere serielt må de ha ulik ID. Derfor ble en og en servo koblet til PC hvor en unik ID ble gitt ved å endre adresse 3 i EEPROM. Ettersom arrays (og vanligvis for-løkker) i C++ begynner med 0 som første element, ble det naturlig å velge ID numrene 0, 1 og 2 for henholdsvis yaw-, roll- og pitch-servoene.

B.7.2 Kalibrering

Servoenes senterpunkt for null vinkelutslag, som er ved 512 dxl ($150.1°$) er ikke nødvendigvis samme senterpunkt for korresponderende rotasjon i GrusBot. Kalibrering er gjort ved å lese av posisjonene i Dynamixel Wizard ved korrekt senterposisjon (utgangsposisjonen) for GrusBot, og deretter implementert disse posisjonene som en offset i SDK programvaren. Det gjør at offset inkluderer både senteravviket og differansen mellom sentervinkelen og 0, slik at senterposisjonen er ved 0 grader istedenfor $150.1°$. Offsetverdiene finnes i Tabell B.4. Merk at Dynamixel Wizard tilbyr en funksjon for kalibrering, men at denne gjelder kun for Dynamixel MX-typen og må ikke utføres på RX-servoene til GrusBot.

Rotasjonsakse	**Yaw**	**Roll**	**Pitch**
Offset [dxl]	530	516	512
Offset [grader]	$155.4°$	$151.3°$	$150.1°$
Offset [rad]	2.713	2.641	2.621

Tabell B.4: Implementert offset for riktig senterposisjon

B.7.3 Baud rate

Fra produsenten har Dynamixel RX-28 en baud rate på 57142 bps, men denne er økt til det maksimale,1 Mbps, som gjøres med adresse 4 i EEPROM. Denne verdien settes også i programvaren ved initialiseringen av GrusBot slik at den kan søke etter Dynamixelenhetene med riktig hastighet. Dersom det ønskes å endre baud raten seinere må den endres både i EEPROM og i programvaren.

B.7.4 Return Delay Time

Adresse 5 i EEPROM er Return Delay Time som er forsinkelsestiden per dataverdi fra transmisjon til retur av pakkesignalene, [38]. Standardverdien for GrusBot 1.0 er $16\mu s$, men kan økes om nødvendig.

B.7.5 Maksimalt dreiemoment

Maksimalt dreiemoment "Max Torque" i adresse 14 og 15 er i EEPROM er satt til 40%, som tilsvarer 410 dxl. Dette er fordi det ikke har blitt gjort rikelig med testing på hvor stor

belastning servoene og mekanikken får dersom dem kjører på full styrke. Produsenten av servoene anbefaler også et dreiemoment på $20 - 40\%$ av det maksimale. Denne verdien kan eventuelt økes seinere hvis det vurderes til å være sikkert. Grensen kan når som helst endres midlertidig ved å å bruke adresse 34 og 35 i RAM, men det frarådes å øke den til mer en 40% før det kan vurderes som sikkert.

B.7.6 Vinkel og vinkelhastighet

Vinkel- og hastighetsbegrensningene er optimalisert for GrusBot 1.0 med Microsoft Kinect V2 som last. Verdiene som oppgis i dxl er nøyaktige verdier som er lagret i servoene.

Tabell B.5 viser vinkelbegrensningene som er bestemt i EEPROM. Minsteverdiene gjelder for adresse 6 og7 "CW Angle Limit" og maksverdiene gjelder for adresse 8 og 9 "CCW Angle Limit". De fysiske vinkelbegrensningene er noe større enn her for å få en til to graders slingringsmonn[7]. Begrensningene presenteres i alle de relevante enhetene. Antall dxl per grad er $1023\text{dxl}/300^\circ = 3.41\text{dxl}/^\circ$. Merk at 0 brukes ikke som minsteverdi, men som maksimal tillatte verdi, slik at området 0 - 1023 består av 1023 dxl og ikke 1024 dxl. Servoene har et bevegelsesområdet på 300°.

Vinkelbegrensninger			
Rotasjonsakse	**Yaw**	**Roll**	**Pitch**
Bevegelsesområde [rad]	$[-2.283, 2.283]$	$[-0.758, 0.758]$	$[-1.228, 1.274]$
Bevegelsesområde [grader]	$[-130.8^\circ, 130.8^\circ]$	$[-43.4^\circ, 43.4^\circ]$	$[-70.4^\circ, 73.0^\circ]$
Bevegelsesområde [dxl]	$[84, 976]$	$[368, 664]$	$[272, 761]$

Tabell B.5: Standard vinkelbegrensninger

Tabell B.6 viser begrensningene for vinkelhastighet som er satt i GrusBot SDK 1.0. Ved initialiseringen i SDK programvaren blir "Moving Speed" i adresse 32 og 33 i RAM endret til en verdi som er så lav at GrusBot klarer å regulere konstant hastighet i hele bevegelsesområdet med en Kinectsensor som last. Disse verdiene er altså ikke lagret permanent i EEPROM og kan derfor endres fortløpende i kjøreprogrammet. Dersom en relativt tung last som Kinect skal brukes anbefales det ikke å øke denne hastigheten, siden tyngdekraften vil hjelpe enkelte bevegelser slik at lasten kan få uønsket høy kinetisk energi. For hver enkelt last kan maksimale hastigheter estimeres ved å sette Moving Speed til maksimum og deretter overvåke Present Speed i Dynamixel Wizard og notere disse verdiene som vises. Maksimalt dreiemoment kan økes for å oppnå høyere hastigheter, men det må gjøres med forsiktighet ettersom det vil være større belastning på servoer og konstruksjon. Begrensningene presenteres i alle de relevante enhetene. Som oppgitt i brukermanualen til Dynamixel RX-28 [38] finnes omdreininger per minutt [rpm] ved å multiplisere hastigheten i dxl med 0.111.

[7] Gjelder ikke for pitch-rotasjonen hvor det er ca. 15° slingringsmonn.

Hastighetsbegrensninger			
Vinkelhastighet [rad/sec]	2.557	2.092	1.279
Vinkelhastighet [grader/sec]	146.5	119.9	73.28
Vinkelhastighet [rpm]	24.42	19.98	12.21
Vinkelhastighet [dxl]	220	180	110

Tabell B.6: Standard hastighetsbegrensninger

Hastighetene er positive for CCW rotasjon og negative for CW rotasjon. Det gjelder derimot ikke i dxl, hvor hastigheten for CW rotasjon kan ha verdiene i området [1024dxl, 2047dxl].

B.7.7 Compliance

I initialiseringen av SDK programvaren settes Compliance Slope til 32 dxl for yaw, og 64 dxl for roll og pitch. Disse verdiene har kun blitt valgt etter testing i Dynamixel Wizard, og det ble valgt de verdiene som resulterte i de mest jevne bevegelsene uten å la det gå utover presisjonen. For detaljert informasjon om compliance, se brukermanualen til servoene [38].

B.7.8 Alarmfunksjon

Dynamixel RX-28 servoene har en LED indikator som lyser eller blinker rødt ved feil, eller dersom programvaren er modifisert til å sette på indikatoren i andre situasjoner[8]. I Dynamixel Wizard kan det velges hvilke av disse feilene som skal trigge indikatoren, som gjøres i adresse 17 i EEPROM. I GrusBot 1.0 er følgende feil valgt som standard til å trigge alarmindikatoren: *Overload, Overheating, Input Angle Limit* og *Input Voltage*.

[8]LED indikatoren vil lyse kort én gang når strømmen skrus på og er ikke en indikasjon på feil.

Vedlegg C

GRUSBOT: API DOKUMENTASJON

GrusBot SDK 1.0 skal gi brukeren en enkel oppstart av roboten GrusBot 1.0, og gi et godt grunnlag for en eventuell videre utvikling. For informasjon om kodestil, oppsett og hvordan biblioteket kan inkluderes i andre prosjekter, se i brukermanualen til GrusBot, Vedlegg B.

Hvordan bruke denne dokumentasjonen

Denne dokumentasjonen består av alt som er tilgjengelig for et program som har inkludert GrusBot SDK biblioteket, bortsett fra funksjonene i USB2Dynamixel SDK biblioteket som finnes i en egen API referanse, [43]. Beskrivelsen her skal gi nok informasjon til at GrusBot SDK skal kunne brukes uten å ha studert selve koden. Det inkluderes ett eller flere eksempler der kun en beskrivelse antakeligvis ikke er nok til full forståelse.

Innholdet som beskrives her er alt som er tilgjengelig i headerfilen til GrusBot SDK 1.0. Klasser, funksjoner, variabler, konstanter og enumerasjoner kommer i samme rekkefølge som deklarasjonene i denne headerfilen, i tillegg til at det er brukt hyperlenker som gjør at brukeren raskt kan finne fram til ønsket beskrivelse.

Når det er snakk om verdier så brukes *dxl* som enhet for Dynamixel RX-28. Det er antall punkter av den totale oppløsningen på 1023.

Ved digital fremvisning av denne dokumentasjonen kan hyperlenkene i listen under brukes til å få mer informasjon, eller så kan den alfabetiske rekkefølgen hjelpe til å finne fram til beskrivelsen.

Globale Konstanter

- const int kNumActuators
- const std::string kActuatorName[kNumActuators]
- const double kAngleLimit[kNumActuators][2]
- const double kAngleSpeedLimit[kNumActuators]
- const double kHomePosition
- const double kSleepingPosition[kNumActuators]

class GrusBot

- public:
  - GrusBot()
  - enum ActuatorID
  - ControlTableAddress
  - enum Evaluation
  - enum Limit
  - enum States
  - enum RotationDir
  - class Connections
    - public:
      - Connections()
      - bool communication_result_success(actuator)
      - bool open_usb2dynamixel(baudRate, usbPortNum)
      - void close_device()
      - void initalize_robot(defaultSpeed, baudRate, usbPortNum)
      - void print_communication_status(status)
    - private:
  - class Conversions
    - public:
      - Conversions()
      - double dxl_to_anglevel(actuator, dxlSpeed)
      - double dxl_to_rad(actuator, dxlPosition)
      - int anglevel_to_dxl(angleVel)
      - int rad_to_dxl(actuator, angle)
    - private:
      - static const double kDxlOffset[kNumActuators]
      - static const double kDxlPerRad
      - static const double kRadSecPerRpm
      - static const double kRpmPerDxl
  - bool check_any_movements()

  - bool check_individual_movements(actuator)
  - RotationDir get_moving_direction(actuator)
  - States get_robot_state()
  - void check_actuator_error()
  - void exit_program()
  - void go_to_home()
  - void go_to_resting()
  - void read_present_angles()
  - void read_present_velocities()
  - void send_angle_to_actuator(actuator)
  - void send_anglevel_to_actuator(actuator, angleVel)
  - void set_input_to_goal_angle(actuator, inputAngle)
  - void set_robot_state(state)
  - double presentAngle[kNumActuators]
  - double presentAngleVel[kNumActuators]

- private:
  - Evaluation evaluate_angle(actuator, angle)
  - Evaluation evaluate_anglevel(actuator, angleVel)
  - void manage_exceeded_angle_limits(actuator, evaluationResult)
  - void manage_exceeded_anglevel_limits(actuator, evaluationResult)
  - void set_compliance_slope(actuator, slopeLength)
  - void write_goal_angle(actuator)
  - void write_goal_anglevel(actuator, angleVel)
  - double goalAngle[kNumActuators]

Globale Konstanter

const int kNumActuators

Antall aktuatorer som er koblet til systemet. Endring av denne krever mye modifisering av koden.

const std::string kActuatorName[kNumActuators]

String med unike navn til hver aktuator. Korresponderer til navnene på enumerasjonsverdiene til enum ActuatorID

const double kAngleLimit[kNumActuators][2]

Vinkelbegrensningene i radianer, der indeks 0 er minste vinkel og indeks 1 er største vinkel. Kan bruke enum ActuatorID og enum Limit fra GrusBot klassen for indekseringene:

```
double maxYawAngleLimit = kAngleLimit[YAW][MAX_ANGLE_INDEX]
```

const double kAngleSpeedLimit[kNumActuators]

Maksimalt tillatte fart for hver aktuator, uavhengig av rotasjonsretning. Kan bruke enum ActuatorID fra GrusBot klassen eller korresponderende verdier for den aktuelle aktuatoren for indekseringen.

const double kHomePosition

Utgangsposisjonen/senterposisjonen som er lik for alle aktuatorer.

const double kSleepingPosition[kNumActuators]

Sove- eller hvileposisjonen til GrusBot. Kan bruke enum ActuatorID fra GrusBot klassen eller korresponderende verdier for den aktuelle aktuatoren for indekseringen.

class GrusBot

public:

GrusBot()

Konstruktør som ikke gjør noe.

enum ActuatorID

Består av { `YAW, ROLL, PITCH` } og brukes der det må spesifiseres en enkelt aktuator.

enum ControlTableAddress

Består av:

- `P_CW_COMPLIANCE_SLOPE`
- `P_CCW_COMPLIANCE_SLOPE`
- `P_GOAL_POSITION_L`
- `P_GOAL_POSITION_H`
- `P_MOVING_SPEED_L`
- `P_MOVING_SPEED_H`
- `P_PRESENT_POSITION_L`
- `P_PRESENT_POSITION_H`
- `P_PRESENT_SPEED_L`
- `P_PRESENT_SPEED_H`
- `P_MOVING`

Flere adresser er tilgjengelige, men det er kun disse som brukes i GrusBot SDK 1.0. Se Brukermanualen til Dynamixel RX-28, [38].

enum Evaluation

Består av { `TOO_LOW, VALID, TOO_HIGH` }
Kan brukes i forbindelse med å undersøke om verdier er innenfor et gyldig område.

enum Limit

Består av { `MIN_ANGLE_INDEX, MAX_ANGLE_INDEX` }
Kan brukes ved indeksering. Eksempel:
`double maxYawAngleLimit = kAngleLimit[YAW][MAX_ANGLE_INDEX]`

enum States

Består av:

- `P_CW_COMPLIANCE_SLOPE`
- `NORMAL`
- `SLEEP`
- `ACTUATOR_ERROR`
- `COMMUNICATION_ERROR`
- `QUIT`
- `QUIT_IMMEDIATELY`

Kan brukes i en switch case hvor ulike handlinger må gjøres i de forskjellige tilstandene

enum RotationDir

Består av{ `CW, NO_MOVEMENT, CCW`}
Kan brukes i forbindelse med aktuatorenes rotasjonsretninger.

class Connections

Se class Connections

class Conversions

Se class Conversions

bool check_any_movements()

Returnerer `true` hvis minst en av aktuatorene er i bevegelse, ellers `false`.

bool check_individual_movements(actuator)

Returnerer `true` hvis aktuator med ID ”actuator” er i bevegelse, ellers `false`.

RotationDir get_moving_direction(actuator)

Returnerer en verdi av enumerasjonen enum RotationDir for å beskrive rotasjonsretningen til aktuator med ID "actuator". Dersom aktuatoren ikke er i bevegelse returneres verdien for `NO_MOVEMENT`.

States get_robot_state()

Returnerer tilstanden til GrusBot ved bruk av enumerasjonen enum States

void check_actuator_error()

Sjekker om en av aktuatorene har en feil, og setter i tilfelle robottilstanden til `ACTUATOR_ERROR` og skriver ut en feilmelding til kommandovinduet. Hvilken aktuator som har feil kan ikke bestemmes. Aktuatorene sjekkes for følgende feil:

- Input Voltage Error: Inngangsspenningen er utenfor det definerte området som er satt i EEPROM
- Angle Limit Error: En posisjon utenfor bevegelsesområdet blir sendt til en av aktuatorene. Denne feilen skal i utgangspunktet ikke oppstå, siden andre funksjoner vurderer alle posisjonene før de sendes.
- OverHeating Error: Intern temperatur i en av aktuatorene er høyere enn opereringstemperaturen som er satt i EEPROM
- Range Error: Når en kommando er gitt utenfor rekkevidden til bruksområdet
- CheckSum Error: Når CheckSum for instruksjonspakken er ugyldig. Kan skje dersom pakken blir ødelagt under kommunikasjonen.
- Overload Error: Dreiemomentet som kreves av aktuatoren er for stort
- Instruction Error: Udefinert instruksjon er sendt, eller en aksjonskommando er levert uten informasjon om hvor den skal skrives

Store deler av denne funksjonen er hentet fra eksempelprogrammet Read/Write fra USB2Dynamixel SDK [43].

void exit_program()

Ikke en nødvendig funksjon, men kan brukes dersom det ikke ønskes at programmet skal termineres umiddelbart etter koden er ferdig. Det eneste funksjonen gjør er å vente på et tastetrykk fra brukeren for å fortsette termineringen.

void go_to_home()

Setter målposisjonen for hver aktuator til kHomePosition og sender alle aktuatorene til denne posisjonen.

void go_to_resting()

Setter målposisjonen for hver aktuator til kSleepingPosition[actuator] og sender GrusBot til hvileposisjonen.

void read_present_angles()

Leser av posisjonen til hver aktuator, lagrer posisjonene som vinkler i radianer i presentAngle[actuator], og skriver ut disse vinklene til kommandovinduet.

void read_present_velocities()

Leser av hastigheten til hver aktuator, lagrer hastighetene i rad/sec i presentAngleVel[actuator], og skriver ut disse hastighetene til kommandovinduet.

void send_angle_to_actuator(actuator)

Sender målposisjonen, goalAngle[actuator] til enten alle aktuatorene (actuator = BROADCAST_ID) eller til en spesifikk aktuator. Målposisjon som er utenfor tillatt bevegelsesområde blir håndtert.
BROADCAST_ID og enum ActuatorID er gyldige argumenter.

void send_anglevel_to_actuator(actuator, angleVel)

Sender ønsket vinkelhastighet [rad/sec], angleVel til enten alle aktuatorene (actuator = BROADCAST_ID) eller til en spesifikk aktuator.
BROADCAST_ID og enum ActuatorID er gyldige actuator-argumenter, og angleVel må være en `double`. Vinkelhastigheter som er større enn tillatt blir håndtert.

void set_input_to_goal_angle(actuator, inputAngle)

Kall denne funksjonen med en spesifikk aktuator ID og ønsket målvinkel ”inputAngle” [rad] for å sette goalAngle[actuator] lik ønsket vinkel. Denne funksjonen sender ikke vinkelen til aktuatoren, det gjøres med funksjonen send_angle_to_actuator(actuator).

void set_robot_state(state)

Tar ”state” av typen enum States som argument og setter robottilstanden til denne verdien.

double presentAngle[kNumActuators]

Vinkelen [rad] for hver aktuator sist gang den ble lest.

double presentAngleVel[kNumActuators]

Vinkelhastigheten [rad/sec] for hver aktuator sist gang den ble lest.

private:

Evaluation evaluate_angle(actuator, angle)

Returnerer valideringen av vinkelen ”angle” [rad] for aktuatoren med ID ”actuator”. Valideringen kan ha verdiene til enum Evaluation.
Eksempel:
Vinkelen `angle` for yaw-aktuatoren har grenseverdiene [-1, 1].
`evaluate_angle(YAW, -2)`; returnerer `TOO_LOW`
`evaluate_angle(YAW, 0)`; returnerer `VALID`
`evaluate_angle(YAW, 2)`; returnerer `TOO_HIGH`

Evaluation evaluate_anglevel(actuator, angleVel)

Returnerer valideringen av vinkelhastigheten ”angle” [rad/sec] for aktuatoren med ID ”actuator”. Valideringen kan ha verdiene til enum Evaluation. Merk at det brukes vinkelhastighet og ikke absolutt vinkelfart.
Eksempel:
Vinkelhastigheten `angleVel` for pitch-aktuatoren har grenseverdiene [-1, 1].
`evaluate_angleVel(PITCH, -2)`; returnerer `TOO_LOW`
`evaluate_angleVel(PITCH, 0)`; returnerer `VALID`
`evaluate_angleVel(PITCH, 2)`; returnerer `TOO_HIGH`

void manage_exceeded_angle_limits(actuator, evaluationResult)

Hvis vinkel ikke er gyldig, la brukeren via kommandovinduet bestemme om vinkelen skal settes til den korresponderende aktuatorens vinkelgrense, eller om vinkelen ikke skal oppdateres. Bruker resultatet fra Evaluation evaluate_angle(actuator, angle) som argument.

void manage_exceeded_anglevel_limits(actuator, evaluationResult)

Hvis vinkelhastighet ikke er gyldig, la brukeren via kommandovinduet bestemme om vinkelhastigheten skal settes til den korresponderende aktuatorens grense for vinkelhastighet, eller om vinkelhastigheten ikke skal oppdateres. Bruker resultatet fra Evaluation evaluate_anglevel(actuator, angle) som argument.

void set_compliance_slope(actuator, slopeLength)

Sett stigningstallet for compliance begge retningene til ”slopeLenght, for aktuator med ID ”actuator”. Se i brukermanualen til Dynamixel RX-28 [38] for informasjon om compliance.

void write_goal_angle(actuator)

Bruk heller funksjonen send_angle_to_actuator(actuator) til å sende vinkel goalAngle[actuator] til ønsket aktuator.

Denne funksjonen konverterer målvinkelen fra radianer til dxl, og sender verdien til aktuatoren med ID ”actuator” dersom vinkelen er gyldig.

void write_goal_anglevel(actuator, angleVel)

Bruk heller funksjonen send_angleVel_to_actuator(actuator) til å sende ønsket vinkelhastighet ”angleVel” til ønsket aktuator.
Denne funksjonen konverterer vinkelhastigheten ”angleVel” fra rad/sec til dxl, og sender verdien til aktuatoren med ID ”actuator” dersom vinkelhastigheten er gyldig.

double goalAngle[kNumActuators]

Lagrer ønsket vinkel [rad] for hver aktuator. For å sette goalAngle[actuator] til ønsket vinkel ”inputAngle” [rad] kan funksjonen set_input_to_goal_angle(actuator, inputAngle) benyttes.

class Connections

Klassen tar seg av de aller nødvendigste funksjonene og alt som har med kommunikasjon og oppkobling å gjøre.

public:

Connections()

Konstruktør som ikke gjør noe.

bool communication_result_success(actuator)

Returnerer `false` hvis funksjonen avbrytes på grunn av kommunikasjonssvikt, ellers er kommunikasjonen i orden og det returneres `true`. Ved kommunikasjonssvikt har brukeren mulighet til å prøve å etablere kommunikasjonen på nytt et fritt antall ganger, via kommandovinduet. Dersom kommunikasjonen mislykkes å reetableres kan brukeren trykke ”ESC” for å avslutte løkken. Da oppdateres robottilstanden til ”COMMUNICATION_ERROR”.

bool open_usb2dynamixel(baudRate, usbPortNum)

Returnerer `true` hvis tilgangen til USB2Dynamixel åpnes vellykket, `false` ellers. Inngangsargumentene er overføringshastigheten ”baudRate” [Mbps] og COM porten som USB2Dynamixel er koblet til ”usbPortNum”.

void close_device()

Avslutter kommunikasjonen med GrusBot. For reetablering må funksjonen initalize_robot(defaultSpeed, baudRate, usbPortNum) kalles.

void initalize_robot(defaultSpeed, baudRate, usbPortNum)

Initialiserer roboten ved å etablere kommunikasjon, sette standard vinkelhastighet for alle aktuatorer, sette standard stigningstall for compliance, og å sende roboten til sin utgangsposisjon, kHomePosition.
Inngangsargumentene er ønsket standard vinkelhastighet ”defaultSpeed” [rad/sec], kHomePositionoverføringshastigheten ”baudRate” [Mbps] og COM porten som USB2Dynamixel er koblet til ”usbPortNum”.

void print_communication_status(status)

Funksjonen kalles ved kommunikasjonssvikt og da printer den ut nyttig feilmelding til kommandovinduet.
Store deler av denne funksjonen er hentet fra eksempelprogrammet Read/Write fra USB2Dynamixel SDK [43].

private:

class Conversions

Klassen inneholder funksjoner og konstanter som brukes for konverteringer av vinkler og vinkelhastigheter mellom henholdsvis dxl og radianer, og dxl og radianer per sekund.

public:

Conversions()

Konstruktør som ikke gjør noe.

double dxl_to_anglevel(actuator, dxlSpeed)

For aktuator med ID ”actutator”, konverteres verdien ”dxlSpeed” [dxl] til vinkelhastighet [rad/sec] og funksjonen returnerer denne vinkelhastigheten. Merk at det opereres med vinkelhastighet og ikke absolutt vinkelfart.

double dxl_to_rad(actuator, dxlPosition)

For aktuator med ID ”actutator”, konverteres verdien ”dxlPosition” [dxl] til vinkel [rad] og funksjonen returnerer denne vinkelen.

int anglevel_to_dxl(angleVel)

For aktuator med ID ”actutator”, konverteres vinkelhastigheten ”angleVel” [rad/sec] til tilsvarende verdi ”dxlSpeed” [dxl] og funksjonen returnerer denne dynamixelverdien. Merk at argumentet ”angleVel” er vinkelhastighet og ikke absolutt vinkelfart.

int rad_to_dxl(actuator, angle)

For aktuator med ID ”actutator”, konverteres vinkelen ”angle” [rad] til tilsvarende verdi ”dxlPosition” [dxl] og funksjonen returnerer denne dynamixelverdien.

private:

static const double kDxlOffset[kNumActuators]

Posisjonsavvik for hver aktuator. Se ”Kalibrering” i brukermanualen B.7.2.

static const double kDxlPerRad

Antall dxl per radian: $\frac{1023\text{dxl}}{(5\pi/3)\text{rad}}$.

static const double kRadSecPerRpm

Antall rad/sec per rpm: $\frac{2\pi\text{rad}}{60\text{s}}$.

static const double kRpmPerDxl

Antall rpm per dxl. Oppgitt i brukermanualen til Dynamixel RX-28 [38].

Vedlegg D

ROBOTBASE: TEKNISK STØTTE

Det ble i Kapittel 2.3 beskrevet et problem med sonarsensorene til Pioneer LX. I forbindelse med dette problemet ble det sendt en forespørsel med detaljerte beskrivelser til teknisk støtte hos Adept MobileRobots. På neste side finnes fullt svar som ble motatt etter kort tid.

Hello Jonas,

We have noticed that the sonar sometimes have the problem you describe. It does seem to depend on what kind of surface it drives on. There is a setting you can change to reduce the sensitivity a bit so you don't get the erroneous readings. In the Arnl/params/ directory, open the file "pioneer-lx.p" for editing. Find the section that looks like:

```
Section Sonar parameters
;SectionFlags for Sonar parameters:
SonarNum 4                ; Number of sonars on the robot.
;  SonarUnit <sonarNumber> <x position, mm> <y position, mm>
<heading of disc,
degrees> <MTX sonar board> <MTX sonar board unit position> <MTX
gain> <MTX
detection threshold> <MTX max range> <autonomous driving sensor
flag>
SonarUnit 0 331 61 10 1 1 0 500 400 1
SonarUnit 1 331 -61 -10 1 2 0 500 400 1
SonarUnit 2 -317 90 164 1 3 0 500 500 1
SonarUnit 3 -317 -90 -164 1 4 0 500 500 1

The numbers in red are the relevant parameters.  Try increasing
them to 550 or 600.  You may only need to change it for the
front sensors, units 0 and 1, since the rear ones are further
from the ground.  But if you notice any false readings from the
rear, increase those as well.

I would also recommend making sure that the transducers are
fully snapped into place in their mounting holes as
misalignment can cause the false readings as well.

Let me know if you need further help with this.  Best,
```

Zeb Dahl

Research Robot Support

Adept MobileRobots

Referanser

[1]

[2]

[3] Doroftei I. Crivoi O. Vanderborght B. Adascalitei, F. and D. Lefeber. *A Motorized Gravity Compensation Mechanism Used for the Neck of a Social Robot*. Ukjent utgiver,
2013.

[4] Adept. Nedlasting av cad-filer til robotbase.
http://www.adept.com/products/mobile-robots/mobile-platforms/lynx/downloads.
02. juli 2015.

[5] G. Angell. *Mekanisk Verksted*. Institutt for Teknisk Kybernetikk,
2015.

[6] BartleyNeuro. Kinect V2 fra sketchup 3d warehouse. https://3dwarehouse.sketchup.com/model.html?id=u5614ff69-15ca-4bb1-acda-5ca4f0100278.
16. februar 2015.

[7] dgswaner. Standard Hitec servo fra Sketchup 3D Warehouse. https://3dwarehouse.sketchup.com/model.html?id=6999f0b2a0ddbfb67e20a5c42fa9976a.
11. mars 2015.

[8] Military Disability Made Easy. Bevegelsesområde til hode og nakke.
http://www.militarydisabilitymadeeasy.com/thespine.html.
04. mai 2015.

[9] Ebay. LED lysstriper.
http://www.ebay.com/sch/i.html?_nkw=led+strips&_sacat=0&_from=R40.
23. juni 2015.

[10] Custom Entertainment Solutions. Industrial animatronic neck mechanism.
http://animatronicrobotics.com/product/industrial-animatronic-neck-mechanism/.
18. juni 2015.

[11] R. Fitzpatrick. *Designing and Constructing an Animatronic Head Capable of Human Motion Programmed using Face-tracking Software*. Faculty of the Worchester Polytechnic Institute,
2010.

[12] Martin Fowler. *UML Distilled, 3. Edition*. Pearson Education, Inc.,
2004.

[13] IFIXIT. The free repair guide for everything. https://www.ifixit.com/.
28. juni 2015.

[14] Fumagalli M. Metta G. Natale L. Nori F. og Sandini G. Jamone, L.
Machine-Learning Control of a Human-like Tendon-driven Neck. Ukjent utgiver,
2010.

[15] Adept MobileRobots Kundeservice. Ordre. Mottatt via E-post.
24. juni 2015.

[16] S. Lilleborge. *Funksjonsspesifikasjon For Sosial Robot, NTNU-Cyborg*. Institutt for Teknisk Kybernetikk,
2014.

[17] Microsoft. Creating and Using a Static Library in Visual Studio.
https://msdn.microsoft.com/en-us/library/ms235627.aspx.
19. juni 2015.

[18] Microsoft. Systemkrav til kinect.
http://www.microsoft.com/en-us/kinectforwindows/purchase/sensor_setup.aspx.
02. juli 2015.

[19] Microsoft. Visual Studio Community.
https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx.
19. juni 2015.

[20] Microsoft. Windows Tools for Kinect V2.
https://www.microsoft.com/en-us/kinectforwindows/develop/downloads-docs.aspx.
19. juni 2015.

[21] MobileRobots. *Pioneer LX Brukermanual, Rev. A*. Adept Technology, Inc.,
2013.

[22] Adept MobileRobots. Nettverkskonfigurering for pioneer lx.
http://robots.mobilerobots.com/wiki/Ubuntu_Linux_Network_Configuration.
10. april 2015.

[23] A. Navarrete. *Design of a Humanoid Neck Movements and Eyes-expression Mechanisms*. University of Skovde,
2012.

[24] NRK. Svak norsk kronekurs.
http://www.nrk.no/norge/historisk-kursfall-for-norske-kroner-1.12104866.
24. juni 2015.

[25] J. Nævra. *Robotteknisk Forstudium av Sosial Robot - The NTNU Cyborg*. Institutt for Teknisk Kybernetikk,
2014.

[26] DNB og DNB Markets. Dnbs valutakalkulator.
https://www.dnb.no/bedrift/markets/valuta-renter/kalkulator/valutakalkulator.html.
10. mars 2015.

[27] Traco Power. DC/DC omformer 15V, 3A. https://www.elfaelektronikk.no/elfa3~no_no/elfa/init.do?item=69-534-93&toc=19196.
25. juni 2015.

[28] Trossen Robotics. Cross Frame Set. http://www.trossenrobotics.com/p/Robotis-Dynamixel-FR07-X101K-Cross-Frame-Set.aspx.
30. juni 2015.

[29] Trossen Robotics. Dynamixel RX-28.
http://www.trossenrobotics.com/dynamixel-rx-28-robot-actuator.aspx.
20. juni 2015.

[30] Trossen Robotics. Hjelpesider. http://learn.trossenrobotics.com/.
20. juni 2015.

[31] Trossen Robotics. Power Hub. http://www.trossenrobotics.com/p/Robotis-Dynamixel-HN07-I101-Idler-Bearing-Set.aspx.
30. juni 2015.

[32] Trossen Robotics. Power Hub.
http://www.trossenrobotics.com/6-port-rx-power-hub.
30. juni 2015.

[33] Trossen Robotics. Teknisk support. E-postsamtale.
22. juni 2015.

[34] Trossen Robotics. USB2Dynamixel FTDI 2.12.00 Notat.
http://learn.trossenrobotics.com/34-blog/140-ftdi-2-12-00-notice-robotis-usb2dynamixel-cm-530-and-ln-101-october-2014.html.
19. juni 2015.

[35] Trossen Robotics. WidowX Robot Pan Tilt Kit.
http://www.trossenrobotics.com/widowx-MX-28-pan-tilt.
18. februar 2015.

[36] Trossen Robotics. Widowx Robot Turret Kit.
http://www.trossenrobotics.com/p/WidowX-robot-turret.aspx.
18. februar 2015.

[37] ROBOTIS. Dynamixel MX-28 AR.
http://www.robotis-shop-en.com/?act=shop_en.goods_view&GS=2396&GC=GD080100.
23. juni 2015.

[38] ROBOTIS. Dynamixel RX-28 e-Manual v1.25.00.
http://support.robotis.com/en/product/dynamixel/rx_series/rx-28.htm.
19. juni 2015.

[39] ROBOTIS. Kontrollere.
http://support.robotis.com/en/product/auxdevice/controller_main.htm.
24. juni 2015.

[40] ROBOTIS. Nedlastning av produkttegninger.
http://www.robotis.com/xe/download_en/26324.
23. mars 2015.

[41] ROBOTIS. Nedlastning av software.
http://en.robotis.com/BlueAD/board.php?bbs_id=downloads&scate=SOFTWARE.
19. juni 2015.

[42] ROBOTIS. Support og e-Manual. http://support.robotis.com/.
19. juni 2015.

[43] ROBOTIS. USB2Dynamixel SDK.
http://support.robotis.com/en/software/dynamixel_sdk/usb2dynamixel.htm.
19. juni 2015.

[44] ROBOTIS. USB2Dynamixel User Manual.
http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.htm.
29. juni 2015.

[45] RobotTwinkee. Dynamixel Robot Cable 4PIN. http://www.robottwinkee.com/Robot_Cable_3P_200mm_10pcs_p/903-0078-000.htm.
30. juni 2015.

[46] ServoCity. Actobotics ServoBlocks. https://www.servocity.com/html/standard_hitec_servoblocks.html#.VZAlWUa90ZY. 11. mars 2015.

[47] ServoCity. Actobotics Standard Hub Horn.
https://www.servocity.com/html/standard_hub_horn.html#.VZAmB0a90ZY. 11. mars 2015.

[48] ServoCity. Actobotics Standard Servo Plate A. https://www.servocity.com/html/standard_servo_plate_a__575112.html#.VZArnUa90ZY.
11. mars 2015.

[49] ServoCity. Actobotics Swivel Hub.
https://www.servocity.com/html/swivel_hub__545364_.html#.VY_YHUa90ZY.
11. mars 2015.

[50] ServoCity. Actobotics Tapped Aluminum Standoff. http://www.robotshop.com/en/actobotics-6-32-x-132-tapped-aluminum-standoff.html. 28. mars 2015.

[51] ShaneMcDonald. Microsoft Visual C++ Static and Dynamic Libraries. http://www.codeproject.com/Articles/85391/Microsoft-Visual-C-Static-and-Dynamic-Libraries. 19. juni 2015.

[52] Larsen F. Homb G. Saltvedt J. O. Kraugerud S. og Sollihøgda S. Smith, Ø. *Project report for CYBORG*. TTK4850 Eksperter i Team, 2015.

[53] Hutchinson S. Spong, M. W. and M. Vidyasagar. *Robot Modeling And Control*. John Wiley & Sons, Inc., 2006.

[54] Ø. Stavdahl. *Personlig korrespondanse*. Institutt for Teknisk Kybernetikk, 2015.

[55] D. Straker. Head body language. http://changingminds.org/techniques/body/parts_body_language/head_body_language.htm. 13. februar 2015.

[56] WikiBooks. Ultralydsensorer i robotikk. https://en.wikibooks.org/wiki/Robotics/Real_World_Sensors#Ultrasonic_Distance_sensors. 26. juni 2015.

[57] Wikipedia. Api. https://en.wikipedia.org/wiki/Application_programming_interface. 01. juli 2015.

[58] Wikipedia. Eeprom. https://en.wikipedia.org/wiki/EEPROM. 01. juli 2015.

[59] Wikipedia. Ide. https://en.wikipedia.org/wiki/Integrated_development_environment. 01. juli 2015.

[60] Wikipedia. Parallel Manipulator. https://en.wikipedia.org/wiki/Parallel_manipulator. 30. juni 2015.

[61] Wikipedia. Pneumatikk. https://no.wikipedia.org/wiki/Pneumatikk. 30. juni 2015.

[62] Wikipedia. Slam. https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping. 01. juli 2015.