NTNU Trondheim
Norges teknisk-naturvitenskapelige universitet
Eksperter i Team

Instrumentering og styring over Internett med fokus på håndtering av kritiske situasjoner

Styring av robot over internett

Gruppe 2

Prosjektrapport

Gruppemedlemmer:
Andreas Carlsen
Pål Fornebo
Eivind André Taftø
Erik Vattekar
Eirik Hexeberg Henriksen

2. mai 2013

Forord

Dette er en rapport skrevet i som besvarelse i Eksperter i team av gruppe 2 i landsbyen "Instrumentering og styring over Internett med fokus på håndtering av kritiske situasjoner". I løpet av arbeidet med prosjektet har gruppen laget et mudulbasert system for å styre servobaserte roboter over en internettlink.

Gjennom arbeidet med prosjektet har gruppen lært mye både teknisk og sosiologisk. Dette hadde ikke vært mulig uten landsby-teamet med landsbyleder Sven Fjeldaas i spissen. De har hjulpet oss både med tekniske spørsmål og med å reflektere over vår egen fremtoning i en gruppe. Takk for all hjelp.

Gruppe 2:

Pål Fornebo
Andreas Carlsen
Eivind André Taftø
Erik Vattekar
Eirik Hexeberg Henriksen

Sammendrag

Gruppen har i arbeidet med dette prosjektet laget et system der man kan styre en robot satt sammen av forskjellige servo-moduler. Styringen foregår over internett. Kontrollen av roboten kan gjøres ved hjelp av forskjellig input-utstyr, som vinkelsensorer, piltaster eller brytere. Roboten kan også styres autonomt.

All informasjon presenteres til brukeren i et grafisk brukergrensesnitt på serversiden. Dette brukergrensesnittet benyttes også i styringen av roboten. Systemet er laget modulært i alle ledd for å være enkelt å endre og legge til funksjonalitet. Målet har vært å lage et system som gjør prototyping av roboter styrt over internett enkelt.

Roboter som er styrt via en internet-link eller autonomt har et enormt potensiale til å øke menneskers sikkerhet og effektivitet. Ved å lage et system for enkel prototyping av slike systemer håper vi å legge til rette for videre utvikling innenfor dette feltet.

Innhold

Figurer

1 Innledning

Denne rapporten beskriver arbeidet med og resultatene fra arbeidet med prosjektoppgaven i faget Eksperter i team - Instrumentering og styring over Internett med fokus på håndtering av kritiske situasjoner. Prosjektet har dreid seg om å lage et system som muliggjør enkel prototyping av roboter som styres over internett.

Rapporten tar først for seg oppgaven gjennom forarbeid, utarbeidelsen av problemstillingen og samfunnsnytten i prosjektet. Deretter kommer en overordnet løsningsskisse av systemet vi har utviklet og hva dette er istand til å gjøre. De neste seksjonene går mer i dybden og beskriver det tekniske aspektet; hvordan og hvorfor vi har gjort det vi har gjort. Til slutt skisserer vi videre arbeid som kan gjøres med prosjektet før det kommer en avslutningsseksjon.

2 Oppgave

I prosjektdelen av dette faget er det meningen at gruppa skal utarbeide sin egen oppgave med utgangspunkt i Landsbyens tema. Landsbyens tema er: *Instrumentering og styring over Internett med fokus på håndtering av kritiske situasjoner.* Dette setter rammen for oppgaven. Første landsbydag ga landsbyleder en introduksjon til hva han så for seg at ville være gode oppgaver og hva som hadde blitt gjort før. Gruppa hadde så idemyldring og diskuterte internt hva den så på som aktuelle oppgaver.

Gruppas medlemmer bestod av to som studerer kybernetikk, en som studerer datateknikk, en IKT og en som studerer marin kybernetikk. Etter litt diskusjon der vi blant annet kartla gruppas faglige egenskaper og interresser ble gruppa enige om en et førsteutkast til oppgave. Vi ville bruke ett sett med robot-servoer som landsbyleder hadde presentert, samt sensorer for å måle bevegelse. Vi ble enige om at vi ønsket å styre en robot vha. bevegelsessensorene via en internet-link.

Ingen hadde noen klar ide om dette var mulig, eller om hvor mye arbeid som krevdes. Vi mente imidlertid å se en hel rekke samfunnsnyttige formål dette kunne brukes til. Vi var også enige i at den kompetansen gruppa satt på ville passe godt til en slik oppgave. Vi satt derfor i gang et forarbeid.

2.1 Forarbeid

Formålet med forarbeidsfasen var å finne ut hva som er gjort før og om dette er relavant for oss. Vi ønsket også å finne ut om det fantes noe kode som vi kunne benytte oss av for å komme raskere igang, samt å kartlegge om vi trengte noe mer utstyr enn det som landsbyleder kunne tilby. Forarbeidsfasen skulle ende opp i en problemstilling som definerte oppgave og avgrenset den slik at det var klart hva vi skulle jobbe med det kommende semesteret.

I løpet av forarbeidsfasen viste det seg at lignende ting var blitt gjort før. Noen av gruppemedlemmene dykket ned i gamle prosjekter fra samme landsby for å finne ut hva disse hadde gjort, og hvordan de hadde gjort det. Det ble etter kort tid klart at det ville være minst like mye jobb å bruke noe av den gamle koden fra disse prosjektene og tilpasse den til vår problemstilling som å starte fra bunn.

Et annet funn som ble gjort i forarbeidsfasen var at det fantes et programvarebibliotek skrevet i python som var skrevet for de samme servomotorene som vi hadde tenkt til å bruke. Dette inneholdt en del funksjoner som kunne være nyttige for oss. Vi bestemte oss tidlig for at vi skulle bruke dette, og dermed gikk resten av forarbeidsfasen til å finne ut hvordan et brukergrensesnitt kan lages i python, samt hvordan man kan hente ut målinger fra sensorer på en enkel måte. Under forarbeidet ble gruppa også enige om å prøve å gjøre roboten trådløs. For å klare dette ble vi enige om å kjøpe inn en batteripakke og en rasberry-pi mini-pc. Dette

skulle kobles til roboten så den kunne settes hvor som helst der raspberry pi hadde internetforbindelse. Dette kan oppnåes enten ved hjelp av kabel, wifi-modem eller gjennom et modem med kontakt med mobilnettet.

2.2 Problemstilling

Forarbeidsfasen la grunnlaget for å kunne formulere en problemstilling. Problemstillingen skulle brukes til å definere og styre arbeidet med prosjektet. Det var derfor viktig at den avgrenset temaet samtidig som den beskrev det vi ønsket å gjøre. Etter en lengre diskusjon kom vi frem til følgende problemstilling:

Vi skal lage et system for å ta i mot signaler fra bevegelsessensorer, prosessere disse og sende de over internet for å bruke dette til å bevege en robotarm. Systemet skal være lett å utvide og endre, slik at det kan nyttiggjøres til enkel og rask prototyping av roboter, og deres styringssystemer.

2.3 Bruksområder og samfunnsnytte

Roboter som kan styres over internett kan brukes til å en rekke formål der det er ønskelig at det skal utføres en oppgave uten å ha en person på den plassen oppgaven skal utføres. Eksempler på dette kan være:

- Mange personer reiser verden rundt fordi de er eksperter til å utføre en bestemt jobb. Dersom de kunne brukt en robot til å gjøre jobben via en internetlink kunne de vært tilgjengelig øyeblikkelig.
- Personell som jobber på farlige plasser som brannmenn, redningsmannskaper, soldater og bombeteam kunne unngått å sette seg i en farlig situasjon dersom de kunne styre en robot via internett.
- Å ha mannskap ombord på oljeplattformer er dyrt og ikke minst farlig. Dersom man kunne laget roboter som ble styrt fra land kunne man spart mye penger i mannskapsutgifter og man slipper å sette personene i fare. I en kritisk situasjon vil roboter kunne være igjen på plattformen og drive skadebegrensning.
- Leger eller kirurger som kan operere uten å faktisk være tilstede, for eksempel i krigs- eller krisesituasjoner.

Dette er eksempler på situasjoner der man ved å sette sammen kjent teknologi på nye måter kan skape nye løsninger der man kan spare menneskeliv, og penger. For å gjøre dette benytter man seg av *Instrumentering* for å samle informasjon om robotens omgivelser, samt der hvor menneskenene er for å lage et styringssignal til roboten. *Styringen kan skje over internett*, men også via andre kommunikasjonskanaler. Ved hjelp av disse systemene kan man *håndtere kritiske situasjoner* uten å sette menneskeliv i fare.

Vår tilnærming til oppgaven er å skape et system der man raskt og enkelt kan lage prototyper på løsninger som kan løse oppgavene over. Systemet skal være enkelt å utvide eller rekonfigurere. Det skal være mulig å bruke forskjellige sensorer og input-enheter til å styre roboten. Målet med dette systemet er at man å legge til rette for at andre enkelt kan utvikle løsninger på eksemplene over. Grunnen til denne avgrensningen er den begrensede tidsperioden dette prosjektet spenner over.

3 Tidligere Arbeid

Det er gjort en del arbeid på tidligere landsbyer som har likheter med vårt prosjekt. Vi skal her se kort på noen av de tidligere arbeidene og sammenligne dem med vårt prosjekt.

Gruppe 12 i 2008 hadde en oppgave med følgende problemstilling [1]:

> *Lage et system for tolkning av sensorer koblet til menneske eller maskin, samt transport av informasjonen over internett for visualisering og styring av robot.*

Gruppen ser ut til å ha bundet seg mye til en gitt struktur på robotarmen og har ikke veldig stort fokus på fleksibilitet. Programmeringsspråket de har valgt er c++, et språk som krever en del kunnskap fra programmererne og evt de som skal modifisere systemet i etterkant.

Gruppe 2 i 2009 skriver følgende om oppgaven sin [3]:

> *Vi har laget et system som lar operatører styre fartøy fra et klient-grensesnitt. Operatøren kobler seg opp til en server og får en oversikt over alle fartøy som er tilkoblet.... Operatørene får hele tiden oppdatert informasjon om alle de tilkoblede fartøyenes posisjon og kameravinkler. Denne informasjonen blir kontinuerlig oppdatert i både et 2D-kart og et 3D-miljø.*

Igjen ser vi at gruppen har låst seg til en spesifikk appliksjon. Denne gruppen har også valgt c++ som programmeringsspråk.

Gruppe 4 i 2010 skriver følgende om oppgaven sin [2]:

> *Ved en kritisk situasjon på en oljeplattform eller lignende så er det veldig viktig å raskt kunne innhente informasjon om hendelsen.... Vi vil utvikle et system med en operatørdel som gjør det mulig å styre en eller flere roboter over Internett og få tilbake data fra kamera og ulike sensorer som roboten kan være utstyrt med. Det grafiske brukergrensesnittet må presentere data fra roboten til en operatør i tilnærmet sanntid.*

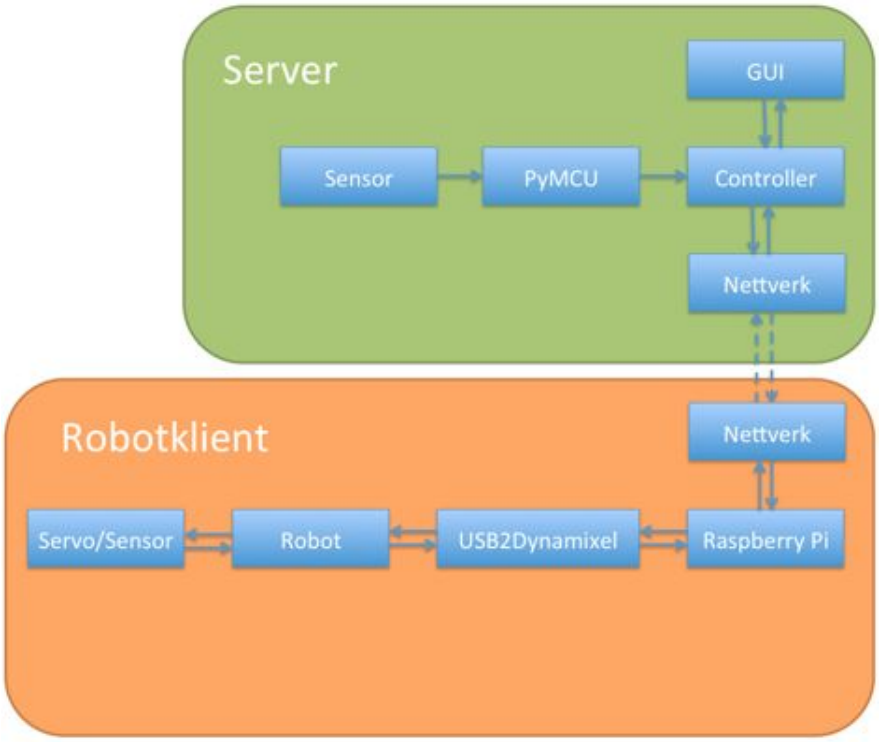
Som i de tidligere prosjektene vi har sett på har også denne gruppen låst seg til en spesifikk applikasjon og også de bruker c++ som programmeringsspråk.

Den store forskjellen mellom vårt prosjekt og tidligere prosjekter er fokuset på skalerbarhet og anvendbarhet. I motsetning til de tidligere prosjektene lager vi et system som skal kunne brukes til prototyping av alle mulige servoapplikasjoner. Utgangspunktet vårt er at man lett skal kunne anvende systemet slik det står for prototyping av forskjellige applikasjoner og at det skal være lett å endre koden dersom det skulle være nødvendig. Systemet er derfor også designet slik at det ved oppstart oppdager alle tilkoblede servoer og sensorer og man kan koble fra og til så mange servoer man vil (begrenset til 256). Man kan også konfigurere hver servo slik man ønsker. Man kan velge om en servo skal settes til gitte vinkler eller om man skal sette hastighet avhengig av om en servo er et ledd eller et hjul. Man kan også gruppere servoer slik at dersom man sender et signal til en gruppe så gjør alle servoene i gruppen det samme. I motsetning til de andre gruppene har vi brukt python som programmeringsspråk. Dette er et mye lettere språk å forstå og gjør det lettere for folk med mer begrenset programmeringskunnskap å endre kode i ettertid.

I tillegg til tidligere prosjekter finnes det også noen kommersielle produkter som kan anses som konkurenter til vårt produkt. Den første og mest åpenbare konkurenten er Bioloid, som selger robot-kits med de samme servoene som vi bruker levert med en mikrokontroller-modul og et GUI for å programmere roboten. Den andre store konkurenten vil være Lego Mindstorm, som også selger robot-kits med eget GUI. Den største forskjellen mellom disse produktene og vårt prosjekt er at vi leverer produktet med åpen kildekode, noe som gjør det enkelt å modifisere koden for å tilpasse applikasjonen til sitt bruk. Med Lego Mindstorm og Bioloid er man låst til det leverte programmeringsverktøyet og dets begrensninger. I tillegg er vårt system satt opp til å kjøre kommunikasjon over internett uten å bruke noe ekstra utstyr enn det som nevnes i denne rapporten. Det finnes noen moduler man kan kjøpe til Lego Mindstorm (f.eks WifiBlock for Lego Mindstorms NXT robot), men disse er veldig dyre og følger ikke med et standard kit. Bioloid har ingen mulighet for kommunikasjon over internett. Vi vil derfor si at dersom du skal ha et prototypeverktøy for robotapplikasjoner som kommuniserer over internett vil vår applikasjon være bedre egnet enn konkurentene.

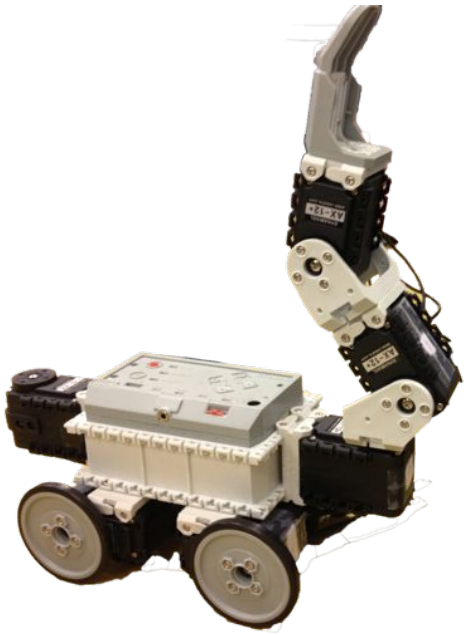
4 Løsningsskisse av Systemet

I dette prosjektet har vi laget et software-system som er istand til å styre en robot over internett. Systemet består av en serverdel og en robot/klientdel. All software er laget modulært slik at det skal være enkelt å utvide med mer funksjonalitet. Det er også laget dokumentasjon til koden slik at denne enkelt skal kunne brukes av andre til å lage prototyper av roboter som blir styrt over internett. En overiktsskisse over de forskjellige modulene i systemet kan finnes i figur 1



Figur 1: Oversikt over systemet

Som et "proof of concept" har vi også satt sammen en robot som styres over internett. Denne roboten er satt sammen av 7 servomotorer og en sensormodul knyttet sammen med forskjellige typer braketter. Til sammen utgjør de en farkost med 4 hjul og en arm. Et bilde av denne roboten kan finnes i figur 2



Figur 2: Robot bestående av 7 servomotorer og en senormodul.

4.1 Server

Serverdelen av systemet vårt er der roboten faktisk styres fra. Denne delen er laget for å kjøres på en hvilken som helst vanlig datamaskin, uavhengig av operativsystem. Serveren har funksjonalitet for å ta imot instrumenteringsdata lokalt og eksternt fra roboten. I vårt testoppsett tar vi inn instrumenteringsdata fra sensorer som måler vinkelutslag i albueleddet til en menneskearm lokalt, mens vi får data om tilstanden til hver enkelt servomotor eksternt fra roboten. All sensordata vises i et grafisk brukergrensesnitt. Brukergrensesnittet brukes også til å styre roboten.

Roboten kan styres på flere måter vha brukergrensesnittet. Sensordata kan "linkes" til en servomotor slik at denne blir styrt direkte fra sensoren. I vårt oppsett bruker vi vinkelsensorer til å sette vinkler på robotarmen. På denne måten kopierer robotarmen bevegelsen til menneskearmen. Vi kan også sette vinkler manuelt, eller sette hvilken hastighet hjulene skal gå med. Brukergrenssnittet støtter også å styre hjulene vha piltaster på tastaturet. Systemet er laget slik at det skal være enkelt å utvikle nye moduler som kan brukes til styring av roboter med andre enheter enn de som er beskrevet her.

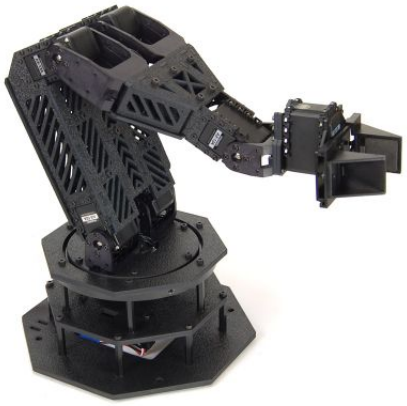
4.2 Robotklient

Robotklienten er hovedsaklig software som er istand til å ta imot data fra serveren og bruke dette til den faktiske styringen av roboten. Vi har også lagt inn mulighet for roboten til å operere autonomt. Klienten støtter forskjellige måter å sette sammen den faktiske roboten slik at man ikke er låst til ett oppsett. Klienten er laget for å kunne kjøre både på en vanlig PC, eller på en mini-pc som kjører en form av operativsystemet Linux. I vårt testoppsett har vi brukt både en vanlig pc og en mini-pc av typen Raspberry pi til styring av roboten.

I vår utvikling av robotklienten har fokus vært på å lage noe som er lett å utvide. På denne måten kan

programmene vi har utviklet brukes til hurtig prototyping av roboter. Robotene vi kan styre ved hjelp av robotklienten må være bygget opp av servoer. Det finnes en rekke måter å sette disse sammen på, og blant annet ved hjelp av braketter slik som vi har gjort. (se figur 2). Disse brakettene gjør det mulig å lage sette sammen servoer på mange måter. Det finnes også flere ferdiglagde verktøy og armer der det vil være enkelt å bruke samme type servoer til å drive. Eksempler på slike kan sees i figur 3 og 4. Ved å bruke en kombinasjon av braketter og ferdiglagde verktøy har man alle muligheter til å sette sammen prototyper på roboter som kan utføre de fleste oppgaver.

Figur 3: Gripeklo som kan styres av servomotorer [8]



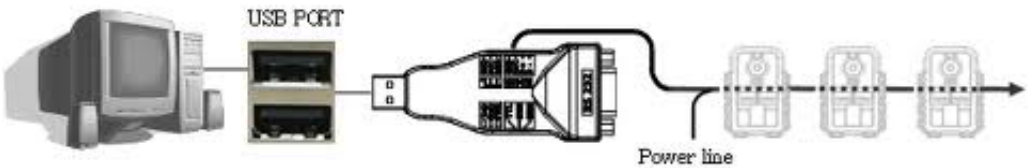
Figur 4: Robotarm som kan styres av servomotorer [8]

5 Robot

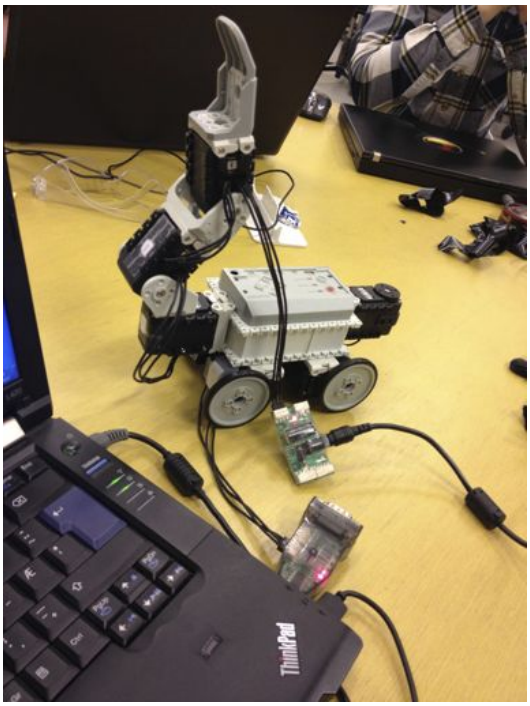
5.1 Servoer og sensorer

Systemet er satt opp til å kjøre på servoer av typen Robotis Dynamixel AX-12, heretter kalt AX-12, og sensorer av typen Robotis Dynamixel AX-S1, heretter kalt AX-S1. Det er også enkelt å modifisere systemet til å kjøre andre servoer og sensorer fra Robotis. Både servoer og sensorer har en bestemt ID. I vårt system har vi definert at ID'en til servoer må være i intervallet 0-30 og ID'en til sensorer må være i intervallet 99-101 (Dette kan endres i koden). Dersom det er usikkert hva som er ID'en til servoene/sensorene som skal brukes kan disse leses og endres i programmet Dynamixel Wizard. Det er viktig at man ikke endrer modellnummeret da dette brukes av systemet for å avgjøre om en tilkoblet enhet er en servo eller en sensor.

Servoer og sensorer kobles i serie som vist i figur 5. Legg merke til at det er koblet inn en ekstra strømforsyning for å dekke strømbehovet til servoene (merket som power line).



Figur 5: Oppkobling av servoer og sensorer



Figur 6: Oppkoblingen som ble brukt under utviklingen av systemet

5.1.1 AX-12

AX-12-servoen kan bli satt til å være et hjul eller et ledd. Dersom man setter minste og største vinkel til å være lik 0 vil servoen oppføre seg som et hjul og man kan sette en hastighet. Dersom minste og største vinkel er definert kan man sette ønsket vinkel på servoen og hvilken hastighet man vil at servoen skal bevege seg til den vinkelen med, men man kan ikke sette en kontinuerlig hastighet.

Man kan også lese av diverse informasjon fra servoen ved å lese fra addressene beskrevet på side 12 i Appendix K. Informasjonen man kan lese inkluderer last, vinkel, volt, temperatur, minimum posisjon og maksimum posisjon.

Figur 7: AX-12 servo

5.1.2 AX-S1

AX-S1-sensoren har IR- og lys-sensorer i 3 retninger. Den har også lydsensorer for å oppdage lyd samt mulighet til å spille av lyder i egendefinerte frekvenser eller preprogrammerte lydsnutter. Man leser denne

informasjonen ved å lese fra addressene beskrevet på side 13 i Appendix L.

Figur 8: AX-S1 sensor

5.2 Hjelpeutstyr

For å få koblet servoene sammen med en pc og få levert strøm trengte vi en del ekstra småutstyr. De forskjellige enhetene vi brukte er beskrevet nedenfor.

5.2.1 SMPS2Dynamixel

SMPS2Dynamixel (figur 9) er en spenningsregulator designet for å drive Dynamixel-servoene. Den konverterer spenningen fra strømforsyningen vår til den spenningen servoene skal ha. Strømforsyningen er 230/12v AC/DC converter som kobles i en vanlig stikkontakt. Spenningsregulatoren kobles inn i serie med servoene, som siste ledd i serien. Et bilde av denne kan sees i figur 9

Figur 9: SMPS2Dynamixe spenningsregulator

5.2.2 Power Bank

For å kunne drive roboten uten å være avhengig av strøm fra veggen kjøpte vi batteripakken i figur 10. Denne skulle levere strøm til servoene samt Raspberry pi'en.

Figur 10: Power bank batteripakke

Batteripakken lades via mini-USB og har to USB-utganger. Ettersom ingen av pluggene vi hadde passet til spenningsregulatoren i figur 9 måtte vi kutte en usb ledning og lage en litt provisorisk løsning (Se figur 11). Ved videre bruk bør denne forbedres.

Figur 11: Selvlaget plugg

5.2.3 USB2Dynamixel

For å kunne koble en datamaskin sammen med servoene trenger vi en overgang fra en USB port til servoene. Dette gjøres ved en enhet spesiallaget for dette formålet. Denne heter USB2Dynamixel. Det denne enheten gjør er å "konvertere" en USB port til en seriell port som man kan skrive bitstrømmer til. Dette gjøres ved hjelp av hjelp av en "FTDI-chip". USB2Dynamixel tilbyr både en RS-232 og TTL seriell interface. Vi bruker TTL delen for å koble PCen sammen med servoene. Denne chipen trenger et sett med drivere for å fungere. Disse driverene har vi ikke kildekoden til, men de finnes i versoner til de fleste plattformer. USB2Dynamixel enheten kan sees i figur 12 En fullstendig spesifikasjon av denne enheten samt brukermanual kan finnes i vedlegg M.

Figur 12: USB2Dynamixel enheten

5.3 Raspberry pi

Raspberry PI er en komplett pc på et lite kretskort. Den har 512mb minne, en 700 MHz ARM prosessor og et stort utvalg av innganger og utganger for alt fra HDMI til lav nivå general purpose pins. Operativsystemet kjører fra et sd-minnekort og vi valgte å bruke Raspbian "wheezy". Wheezy er siste versjon av Raspbian, som er en Debian-variant optimalisert for Raspberry PI. Vi valgte Raspbian på grunn av at det er enkelt, særlig siden det kom med Python ferdig satt opp. I følge den opprinnelige planen skulle PI'en ha et usb wifi adapter, men dette falt bort på grunn av problemer med strømleveransen fra de to USB-portene. Etter en del testing ble det klart at USB-portene til PI'en leverte for lite strøm til å styre servoene hver for seg. Løsningen ble å droppe trådløsnettverket og bruke begge portene til å styre servoene ved hjelp av en y-kabel. Ved å bruke en USB-hub med strømtilførsel kunne vi fått begge deler, men problemet ble desverre oppdaget for sent.

5.4 Programvarebibliotek for robot

For å få python programmer skrevet i python til å kommunisere med servoene har vi benyttet oss av et programvarebibliotek som er laget på Georgia Institute of Technology. Dette biblioteket er har åpen kildekode (open-source) og er skrevet i python. Biblioteket inneholder funksjoner for å lage meldinger og sende disse til servoene, og motta responsen fra servoene. Hvordan en slik melding er bygget opp kan man se i figur 13. Biblioteket inneholder kode for å initialisere og oppdage servoer som er koblet til en PC via en USB2Dynamixel enhet. For å få den funksjonaliteten vi ønsker har vi utvidet biblioteket.

0xFF	0xFF	ID	Lengde	Instruksjon	Parameter 1	...	Parameter n	Sjekksum
0xFF	0xFF	0x00	0x05	0x03	0x0C	0x64	0xAA	0xFD

Figur 13: Struktur på en melding som sendes til en servo

I vår utvidelse av biblioteket har vi valgt å bruke de grunnleggende meldingsfunksjonene fra Georgia Tech sitt bibliotek. På denne måten kan vi forholde oss til minneadresser på hver enkelt servo, og hva som skal skrives til denne isteden for å konstruere de meldingene som skal sendes til servoene. Målet med utvidelsen er å kunne tilby et enkelt grensesnitt for å bruke Robotis-enheter. Det har blitt laget en programklasse

som tar seg av den grunnleggende kommunikasjonen med hver enhet, den som er lik for alle enheter fra Robotis. Denne klassen heter Robotis. Her ligger den grunnleggende funksjonaliteten som lå i biblioteket fra før.

Det er ønskelig at brukeren skal slippe å forholde seg til minneadresser og meldinger som skal sendes til servoen. Derfor har vi utvidet biblioteket med klasser for de Robotis-enhetene vi har tilgjengelige. Dette er AX-12 servoer og en AX-S1 sensor enhet. Disse klassene inneholder program-funksjoner for å bruke den funksjonaliteten som ligger i hver enkelt enhet. Denne funksjonaliteten er forskjellig for hver type enhet. En oversikt over de forskjellige funksjonene kan finnes i Appendix E.

6 Sensorer

Menneskekroppen har hundrevis av ledd og hvis man skal instrumentere et menneske vil det være behov for et tilsvarende antall sensorer for å snappe opp alle bevegelser. I vår prototype instrumenterer vi deler av en menneskearm ved hjelp av to fleksible sensorer og en rotasjonssensor.

6.1 Flexsensor

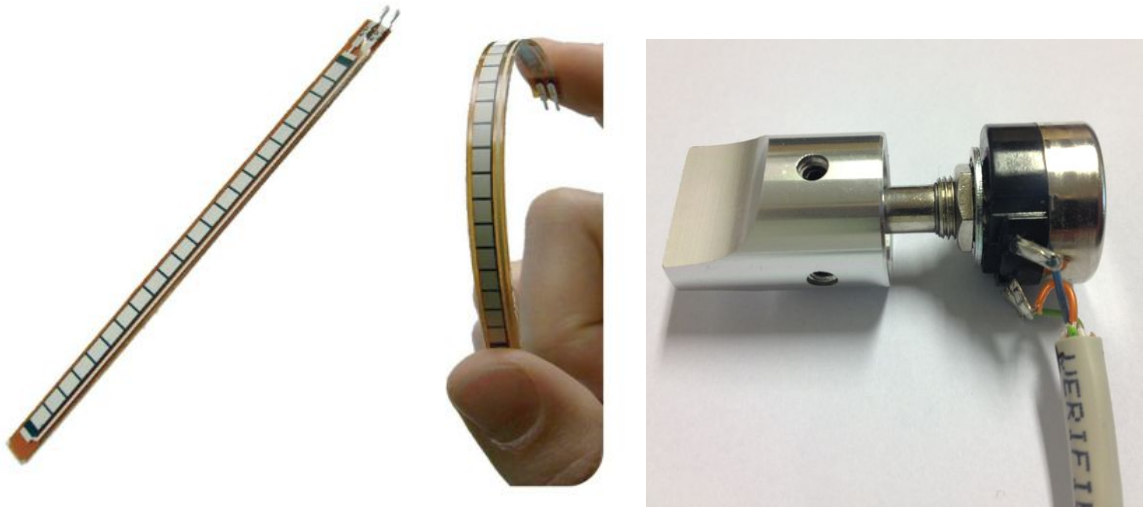
For å instrumentere menneskearmen trenger man sensorer for å måle bøyning av hengselledd som eksempelvis fingre, håndledd, albueledd osv. Til dette kan man bruke fleksible motstander som endrer resistans når de blir bøyd, og på den måten måle endringer av vinkelen til hengselleddet. Når en slik flexsensor er ubøyd er resistansen ca. 10kΩ, men når man bøyer den vil resistansen øke og være omkring 22kΩ ved en vinkel på 90 grader. Flexsensorene har en stykkpris på 75 kr, og er sånn sett nokså kostbare. De må også behandles forsiktig ettersom spesielt området rundt koblingspunktene er svært sårbart og bør forsterkes slik at denne delen ikke blir bøyd for mye. Det er også verdt å merke seg at å bøye en slik sensor for mye i ett punkt kan skade sensoren permanent. Derfor er det smart å bøye sensoren rundt en krumningsradius og forhindre at et punkt på sensoren bøyes mer enn 90 grader. Nøyaktigheten er veldig avhengig av sensorenes plassering og nøyaktigheten til ADC-en som blir brukt. Vår erfaring er at de i praksis har en presisjon på omlag 2 til 3 grader.

I vår prototype har vi festet to slike fleksible motstander til en arm, én på utsiden av albueleddet og én på en finger.

6.2 Rotasjonssensor

Til å måle endring i rotasjon kan man bruke et potensiometer. Disse er billige og enkle å bruke med en stykkpris på omkring 10 kr. Dette er riktignok ikke den mest ideelle sensoren for måling av rotasjon i leddene på en menneskekropp og egner seg kanskje bedre til bruk på maskiner. Et potensiometer har svært god nøyaktighet med en presisjon på omkring en halv grad.

I prototypen vår brukes denne rotasjonssensoren manuelt og er ikke tilknyttet noe ledd på armen vi instrumenterer.



Figur 14: Flex sensor og potensiometer

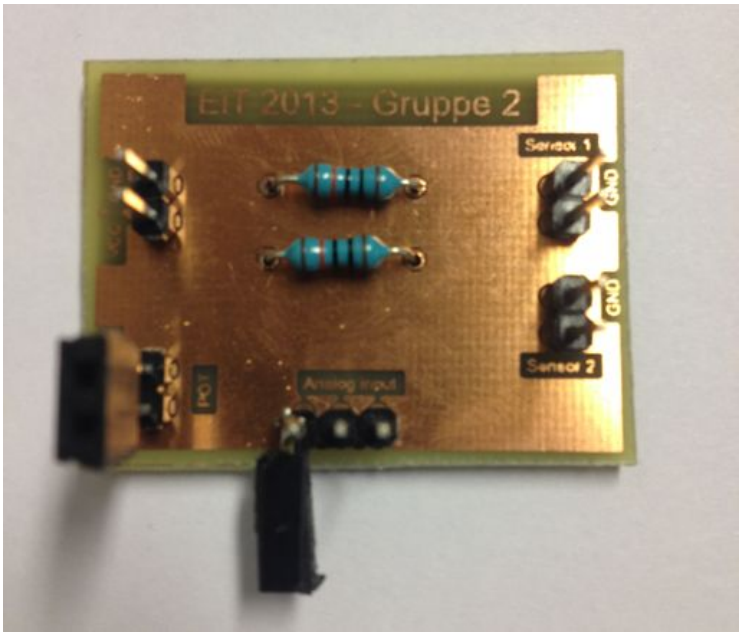
6.3 Kretskort

Som nevnt tidligere endres resistansen i en flexmotstand når den bøyes. Men for å plukke opp en slik endring på en av mikrokontrollerens analoge inputpinner, må vi ha en spenningsdeling mellom en konstant motstand og den fleksible som vi kan sammenligne med. Derfor er det nødvendig å sette opp en krets med denne funksjonen som fungerer som et bindeledd mellom mikrokontrolleren og sensorene. Verdien på den konstante motstanden ble valgt til 10k, slik at forholdet mellom de to når flexmotstanden ikke er bøyd er delt likt.

Innledningsvis ble det brukt et "breadboard" til å sette opp denne kretsen, men til den endelige prototypen valgte vi å lage et kretskort. Nedenfor følger en beskrivelse av hvordan vi fremstilte kretskortet ved hjelp av fotolitografi og etsning. Bilder av kretsskjema og utlegg finnes i vedlegg D.

Utlegget til kretskortet ble laget med programmet "Eagle PCB" som er en gratis programvare man kan laste ned fra internett. Med dette programmet lagde vi et PCB design av den kretsen vi ønsket med "pad"-er til komponenter, ledningsbaner og jordplan. Dette designet ble så printet ut på en transparent med en laser printer som vi la over kretskortet og belyste i to minutter for å løse opp fotoresisten på kretskortet. Videre la vi kretskortet i kaustisk soda i et halvt minutt slik at fotoresisten ble fjernet der hvor det ikke var ledningsbaner eller jordplan. Kretskortet ble så lagt i et etsebad i 20 minutter som fjernet kobberlaget der hvor det ikke var fotoresist. Etter dette vasket vi kretskortet i rødsprit for å fjerne rester etter etsningen og fotoresisten over ledningsbaner og jordplan.

Til slutt ble det borret hull i henhold til fotavtrykkene på kretskortet og komponentene ble loddet på. Vi verifiserte også at kretskortet fungerte, dvs. at endepunkter på alle ledningsbaner hadde kontakt, ved hjelp av et multimeter.

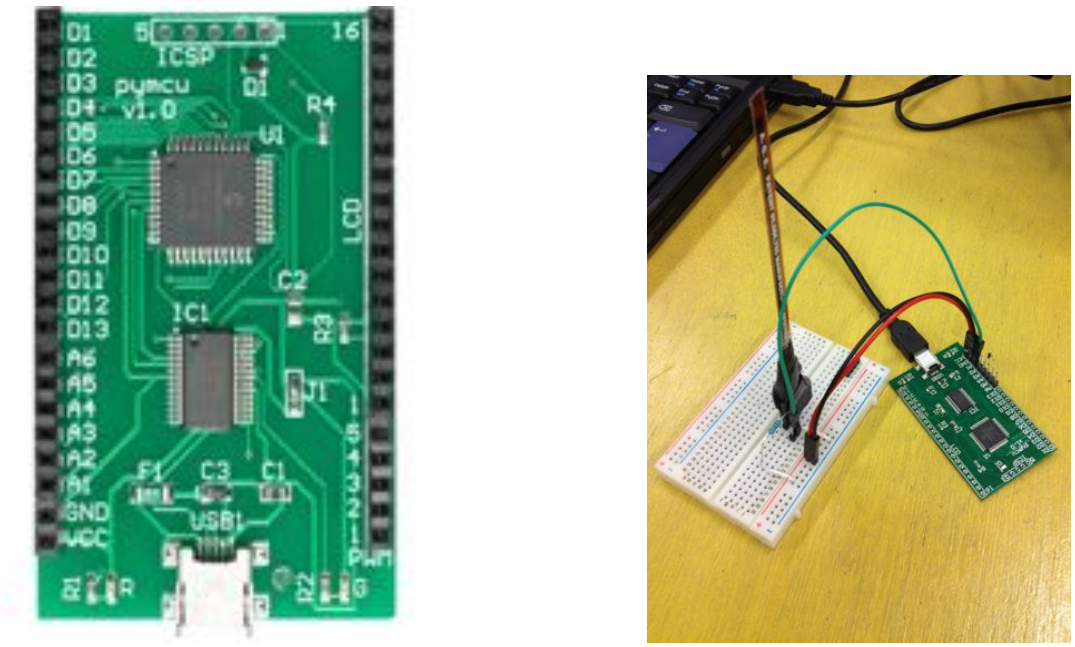


Figur 15: Bindeleddet mellom pyMCU og sensorer

6.4 pyMCU

Til interfacing mellom PC og sensorene har vi brukt en pyMCU v1.0. Dette er en plattform for grensesnitt mellom PC og den fysiske verden ved bruk av programmeringsspråket Python. Plattformen består av en mikrokontroller og en "USB to UART"-konverteringschip. Den fungerer også som et "breakout board", dvs. det er lagt opp ledningsbaner fra alle pinnene på mikrokontrolleren til egne terminalpinner på plattformen. Dette gjør at man enkelt har tilgang til hver eneste pinne og kan koble ledninger direkte til eksterne komponenter, som i vårt tilfelle de fleksible motstandene og potensiometeret. Ved å benytte "pymcu python"-modulen på datamaskinen kan du oppnå ønsket funksjonalitet i sanntid ved å programmere i python, enten direkte i den interaktive kommandolinjen eller som en del av et større program.

Enhetsprisen på en pymcu er ca. 140 kr. Til sammenligning kunne man eventuelt kjøpt mikrokontroller og en "USB to UART" konverter for seg, og laget et eget kretskort med tilsvarende funksjonalitet. "USB to UART"-konverteren som er brukt på denne plattformen koster drøyt 30 kr per stykk, mens PIC-mikrokontrolleren koster omkring 15 kroner kjøpt enkeltvis.



Figur 16: PyMCU og et testoppsett av interfacing med flex-sensor

6.4.1 Microchip PIC16F1939

PIC16F1939 er en 8-bits mikrokontroller med blant annet seks analoge I/0-pinner, hvor vi bruker tre av disse til interfacing med sensorene. Mikrokontrolleren har en intern 10-bit analog til digital konverter (ADC). De analoge inputsignalene fra sensorene multiplekses inn i en "sample and hold"-krets, og utgangen fra denne kobles videre til konverteren. ADC-en genererer et 10-bit binært resultat ved tilnærming (successive approximation) og lagrer den digitale verdien i mikrokontrolleren. Når mikrokontrolleren kjøres på 5 V vil ADC-verdien ligge i området 0 til 1023. I prototypen er det kun benyttet tre sensorer, men ettersom mikrokontrolleren har totalt seks analoge I/O-pinner er det enkelt å implementere ytterligere tre.

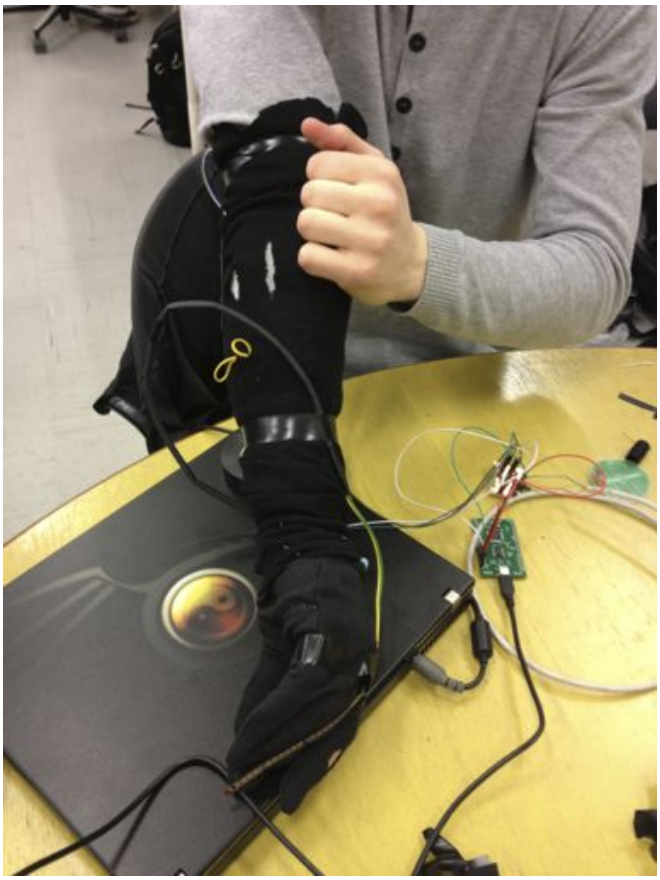
6.4.2 FTDI USB to UART FT232R

FT232R er en "Integrert krets" (IC) som fungerer som et grensesnitt for konvertering fra USB til asynkron, seriell data. På den måten kan man overføre signaler fra UART-interfacet på mikrokontrolleren til en USB-port på datamaskinen og vice versa. Hele USB-protokollen håndteres i chip-en og dermed kreves det ingen USB-spesifikk firmware programmering ved bruk av denne IC-en.

6.4.3 Pymcu, Python modul

Med pymcu plattformen følger det i tillegg til drivere med en "pymcu python"-modul. Ved å opprette en "mcuModule Class object" i python vil du kunne kommunisere direkte med mikrokontrolleren, og dermed i sanntid benytte deg av klassens modulfunksjoner for sette pinner høye og lave, lese verdien fra analog input og en rekke andre funksjoner som er vanlig ved programmering av mikrokontrollere.

I vårt system brukes "pymcu"-en til å ta inn analoge signaler fra de to fleksible motstandene og potensiometeret. Videre foretas det en analog til digital konvertering av signalet og til slutt sendes de målte verdiene videre til en datamaskin. Oppkoblingen mellom pymcu og sensorer finnes i vedleg C. Hele systemet kan sees i figur 17



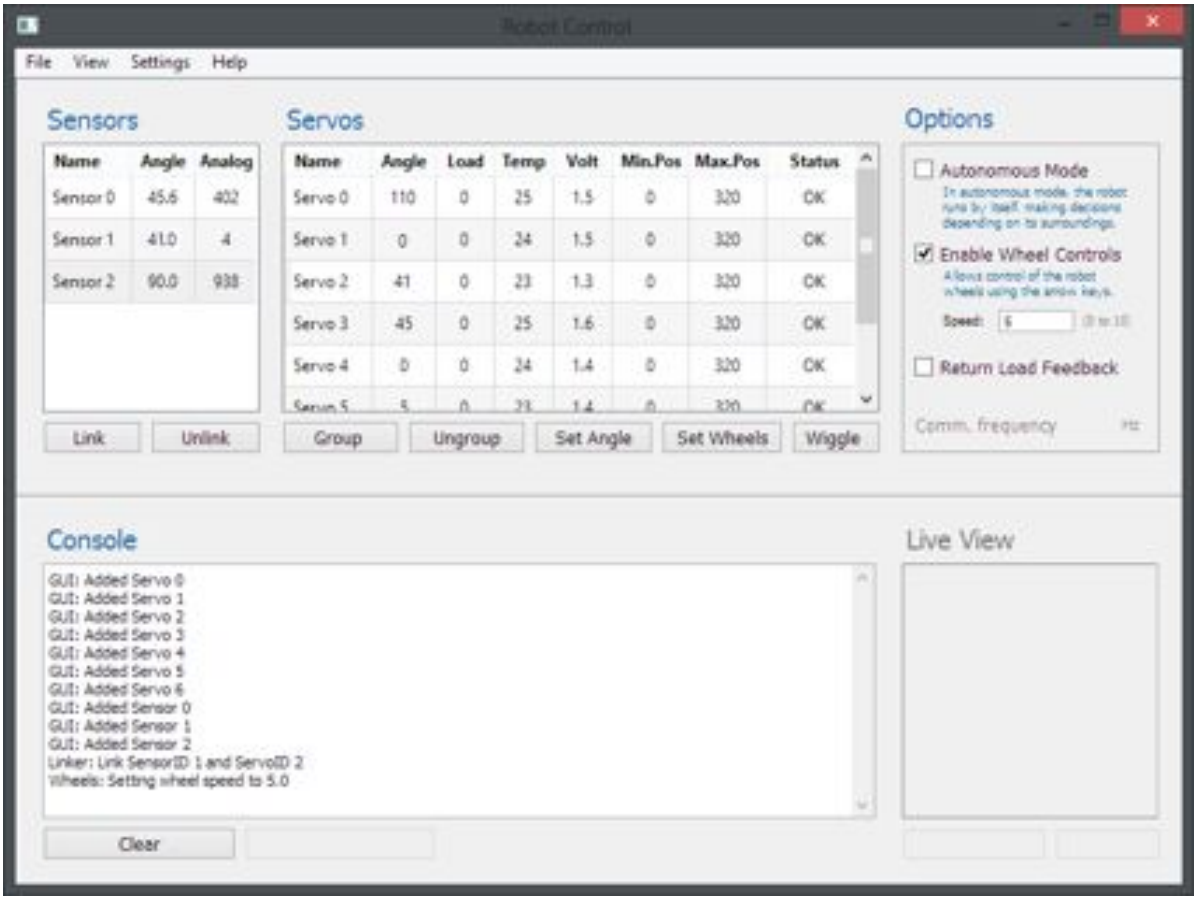
Figur 17: Oppsett med pyMCU, Kretskort og instrumentert hanske

7 Grafisk Brukergrensesnitt - GUI

For å få en enkel oversikt over all instrumenterings info, samt og kunne styre roboten har vi utviklet et grafisk brukergrensesnitt. Dette er det brukeren forholder seg til under enkel prototyping av roboter. Det er lagt opp til at brukergrensesnittet skal kunne utvides dersom man ønsker mer funksjonalitet.

7.1 Hovedvinduet

Hovedvinduet kan deles inn i fire hoveddeler. Øverst til venstre er lister med sensorer og servoer, samt knapper med relevante funksjoner. Til høyre for listene er et panel med diverse opsjoner som gjelder for robotsystemet som en helhet. Nederst finner man en konsoll som viser informasjon eller feilmeldinger fra systemet og roboten. Nederst til høyre er det holdt av plass til et panel som kan vise en grafisk representasjon av sensorenes og/eller robotens tilstand. Dette kan sees i 18



Figur 18: Hovedvinduet i det grafiske brukergrensesnittet

7.1.1 Sensor- og Servolister

Øverst til venstre i grensesnittet finner man lister med sensorer og servoer. Her vises diverse informasjon som kontinuerlig leses inn. Under listene er en rekke knapper som brukes for linking, gruppering, styring og testing av sensorer og servoer. For mer informasjon om funksjonalitet, se brukermanualen for grensesnittet.

7.1.2 Options

I panelet til venstre for listene kan man sette diverse globale settings som angår roboten og systemet som en helhet. De viktigste implementerte funksjonene her er autonom modus av/på, og styring av hjulene av/på, inkludert mulighet til å sette hastighet for kjøring. Når hjulkontroll er aktivert kan brukeren bruke tastaturet til å kjøre roboten manuelt.

7.1.3 Console

Konsollen nederst i vinduet kommer med informasjon og feilmeldinger om roboten og hele systemet. I en kodemodul kan man bestemme hvilke typer meldinger som skal skrives til konsollen. Ved videre utvikling kan disse innstillingene være tilgjengelige direkte fra grensesnittet nær konsollen.

8 Programutvikling

8.1 Python

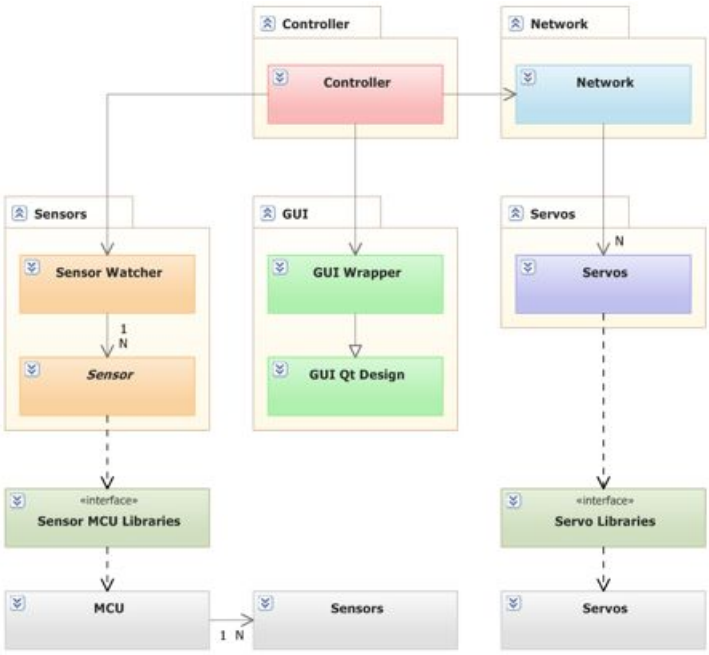
Python er et gratis, open source, høynivå programmeringsspråk. Det støtter programmering innenfor flere paradigmer inkludert funksjonell og objektorientert, og er svært utbredt som skript-språk. Mye av pythons popularitet skyldes den ekspressive syntaksen som gjør det mulig å utføre mye med få linjer kode sammenlignet med lavere nivå språk som C. Dette fører til kode som er lesbar og er veldig produktivt på typiske skripteoppgaver med små krav til optimalisering og ytelse. Vi valgte å bruke Python av følgene grunner:

- **Portabilitet:** Vi kunne skrive og teste koden på våre egne pc'er, for så å porte den direkte til Raspberry PI, nesten uten noen endringer. I tillegg fantes det både Apple- og windowsbrukere i teamet slik at utvikling i et språk som var bundet til et operativsystem var uaktuelt.
- **Eksisterende bibliotek:** Vi oppdaget lib robotis biblioteket tidlig i forstudien.
- **Gjennbruk:** Python er svært utbredt, dessuten oppfattet vi den ekspressive syntaksen og støtten for objektorientert programmering som perfekt for å lage kode som kunne gjenbrukes av andre
- **Interesse:** Python er som nevnt et utbredt programmeringspråk og det var et ønske i gruppen å lære seg å kode i det.

8.2 Pakke- og Klassediagram for hele systemet

For å få en oversikt over systemet som en helhet har vi satt opp et forenklet klassediagram som inneholder de mest essensielle klassene og hvilke pakker de tilhører. I tillegg viser diagrammet hvordan systemet er koblet til fysiske objekter gjennom biblioteker. Pilene i diagrammet viser tilhørighet, arv og avhengighet. For å gjøre diagrammet enkelt og oversiktlig er noen klasser slått sammen og/eller gitt andre navn. Diagrammet kan sees i figur 19.

For å forklare sammenhengen mellom alle klassene og objektene er det naturlig å begynne nede til venstre, på sensorsiden av systemet. En pyMCU er koblet til opptil flere fysiske sensorer. Ved hjelp av Sensor MCU Libraries kan sensorklassen lese informasjon om sensorer via MCUen. Sensor Watcher har en liste med alle sensorer som er koblet til systemet, og gir beskjed så snart en sensor har en ny måling. Controller vet hvilken sensor målingen kom fra, og avgjør hvor den skal videre. Informasjonen sendes til GUI for visning, og via Network til Servos. Herfra kan de fysiske servoene styres ved hjelp av Servo Libraries. På samme måte sendes informasjon fra servoer i retur via Network og Controller slik at det kan vises i GUI.



Figur 19: Klassediagram for hele systemet

8.3 Brukergrensesnitt -GUI

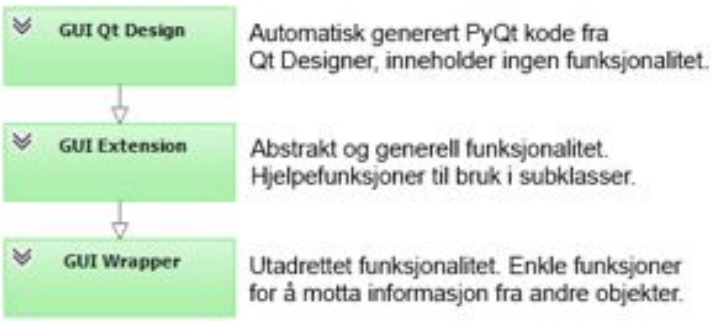
Det grafiske brukergrensesnittet er designet i Qt [5]), som i utgangspunktet er knyttet til C++. All underliggende funksjonalitet er skrevet i Python, som resten av prosjektet. For å knytte Qt til Python har vi benyttet biblioteker fra prosjektet PyQt [4]. Vi har valgt å gå for PyQt4, som gir oss tilgang til all nødvendig funksjonalitet fra Qt 4.

8.3.1 Design

Design og layout av hovedvinduet i brukergrensesnittet er gjort i Qt Designer som følger med Qt fra qt-project. Qt Designer er et grafisk utviklingsmiljø som gjør det enkelt å plassere ut objekter i et vindu og angi forskjellige egenskaper til kontrollene. I tillegg kan man her opprette grunnleggende signals og slots, som er Qts svar på events. Resultatet blir et tomt skall av et grensesnitt, uten noe avansert funksjonalitet. Grensesnittet lagres til en .ui-fil med XML-kode, som så kan konverteres til en Python-fil med PyQt4. Når grensesnittet er overført til Python kan vi arve fra den resulterende klassen og ha direkte tilgang til alle elementene som ble plassert ut i design.

8.3.2 Abstraksjon

Hovedvinduet er delt opp i tre lag for å skille abstraksjonsnivå og typer funksjonalitet. På øverste nivå er den automatisk genererte koden fra Qt Designer. På dette nivået er det ingen funksjonalitet, bare et tomt skall. Andre nivå utvider skallet med generelle nyttefunksjoner og abstrakte metoder. Dette laget er ment å gjøre det lettere å gjøre mer overordnede funksjoner på neste nivå igjen. Det tredje og siste nivået fungerer som en interface mot de objektene som skal sende informasjon til GUI, for eksempel kontrollerklassen. På denne måten forenkler vi de utadrettede funksjonene samtidig som vi skjuler mer abstrakte og generelle hjelpefunksjoner fra omverdenen. En grafisk fremstilling av dette kan sees i figur 20.



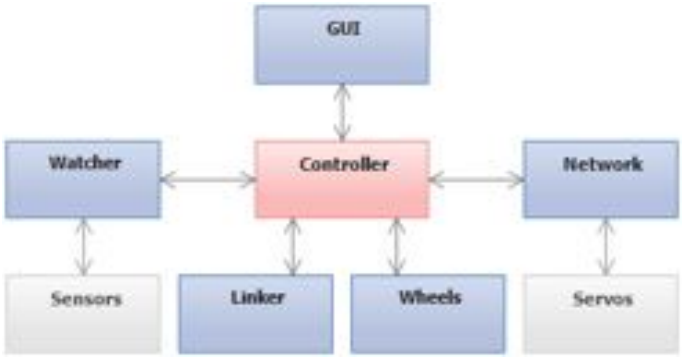
Figur 20: Arkitektur for GUI-design

8.4 Controller

Kontrolleren tar seg av styring og kobling mellom sensorer på den ene siden og servoer på den andre siden via nettverket. I tillegg har kontrolleren en referanse til GUI slik at den kan oppdatere grensesnittet med informasjon, og motta kommandoer fra brukeren som skal passeres videre.

8.4.1 Relaterte klasser

Figur 21 nedenfor viser hvilke objekter kontrolleren inneholder og kommuniserer med. Man kan følge kommunikasjonen fra sensorer til venstre, gjennom Watcher, Controller og Network, før man til slutt ender opp på servoene, altså robotsiden. Linker og Wheels er hjelpeklasser for håndtering av kobling og oppsett av sensorer og servoer. GUI står på siden både som observatør med oppdatert informasjon, og med mulighet for å endre innstillinger manuelt eller styre roboten via grensesnittet.



Figur 21: Klassediagram for Controller

GUI
Grensesnittet mottar oppdatert informasjon om sensorer og servoer kontinuerlig via kontrolleren. Fra hovedvinduet kan brukeren justere diverse globale innstillinger, koble opp sensorer mot servoer, gruppere servoer, angi hjul for kjøring, samt styre roboten manuelt.

Network
Kontrolleren har en referanse til nettverksobjektet slik at den kan sende kommandoer til roboten og servomotorer via nettverket. Når nettverksobjektet opprettes legger kontrolleren også inn en referanse til seg selv, slik at nettverksobjektet kan kalle på metoder i kontrolleren.

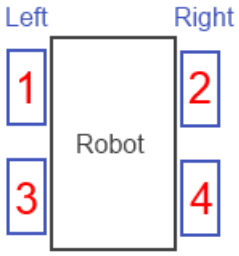
Sensor Watcher

Sensor Watcher har en liste over alle sensorer som er tilkoblet systemet. Den kjører i en egen tråd som kontinuerlig sjekker alle sensorene for nye målinger, og sender dem videre til kontrolleren. Kontrolleren vil avgjøre hva som skal gjøres med nye målinger avhengig av hva som er registrert i Linker. Verdiene sendes også videre til GUI for visning.

Linker
Dette objektet tar for seg koblinger (link) mellom sensorer og servoer, samt gruppering av servoer. Med begrepet link mener vi en kobling fra sensor til servo som tillater fjernstyring av roboten ved hjelp av for eksempel bøyesensorer. Med gruppering (group) mener vi sammenslåing av to eller flere servoer slik at de beveger seg i takt. Dette kan for eksempel gjøres når man vil at flere ledd skal bøye seg på kommando fra én sensor, eller når flere hjul skal kjøre samtidig. Linking og gruppering er samkjørt, slik at hvis én sensor er linket til én servo, og denne servoen samtidig er i en gruppe, vil alle disse servoene inkluderes. På denne måten kan kontrolleren enkelt utføre eller videreføre kommandoer som har med relasjoner mellom sensorer og servoer å gjøre.

Wheels
Her ligger informasjon om robotens tilordning av hjul, samt styring av disse. Når man skal sette opp hjulene til roboten grupperer man først alle høyrehjulene og venstrehjulene i Linker. Deretter angir man et hovedhjul fra hver side og registrerer det her. Når dette er gjort vil klassen håndtere alt av styring basert på enkle inputkommandoer; venstre, høyre, fram, tilbake og stopp. Det er nyttig å pakke inn denne funksjonaliteten her for å forenkle hjulstyringen og gjøre den mer dynamisk med tanke på antall hjul osv. Denne klassen tar for eksempel hensyn til posisjonering og speilvending av servoer (servoer på venstre side må ha negativ vinkelfart i forhold til høyre for å kjøre fremover).

I figur 22 er det et eksempel på hjuloppsett for en robot med fire hjul. Her vil man først gruppere venstrehjulene 1 og 3, og høyrehjulene 2 og 4 i Linker. Videre registrerer man hovedhjulene 1 (venstre) og 2 (høyre) i Wheels klassen. Linker vil sørge for at når hjul 1 eller 2 kjører, så vil også hjul 3 eller 4 kjøre, henholdsvis. Kontrolleren er nå i stand til å ta kjørekommandoer fra grensesnittet, eller roboten kan settes i autonom modus, hvor den vil kjøre av seg selv.



Figur 22: Hjul på roboten

8.5 Nettverk

Målet var å designe et nettverkslag som skulle være modulært og skjule mest mulig av nettverksdetaljene fra resten av systemet. Denne seksjonen beskriver hvordan dette er løst på design-nivået og beskriver de viktigste av teknologiene som er benyttet.

8.5.1 Design

Nettverkslaget er designet med tanke på å skjule mest mulig av nettverksdetaljene fra resten av systemet. Dette er oppnådd ved å organisere nettverket i selvstendige moduler med et enkelt grensesnitt bestående av metoder for å sende og motta meldinger. Kommandoer fra gui-siden sendes gjennom et grensesnitt med

metoder som i størst mulig grad korresponderer direkte med metodene som kalles på robot-siden slik at nettverket blir "gjennomsiktig". Controller-klassen i gui'et initialiserer nettverkslaget og registrer en referanse til seg selv i den innkommende socketen. Data fra nettverket oppdateres i gui'et via callback metoder i controller-klassen. Sekvensdiagram for initialisering og callback finnes i vedlegg H

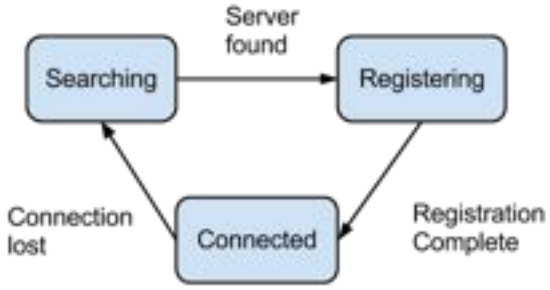
Robotsiden er designet på samme måte, med et enkelt grensesnitt for sending og en referanse til et robotobjekt som inneholder metoder for å utføre de ulike kommandoene fra gui'et. En felles protokollmodul inneholder metoder for å parse meldinger. Protokollen består av enkle strenger formatert etter følgende mønster:

[Kommando, Parameter1, .., ParameterN]

En komplett beskrivelse av protokollen finnes i APPENDIX G

Metodene i nettverkslaget fungerer som en API for vår protokoll, men selve nettverkslaget kan også brukes for å implementere nye protokoller. Dette kan gjøres enkelt ved å arve NetworkLayer klassene på server og klient siden. Et API for sending av meldinger lages ved å skrive et grensesnitt av metoder som bruker send_message metoden til å sende de riktige meldingene. For mottak trenger man bare å override callback metoden rcv_message. Dersom man av en eller annen grunn ønsker å endre meldingsstrukturen kan dette gjøres ved å override metodene i MessageParser-modulen.

Kommunikasjon er to-veis, med en separat innkommende og utgående socket på begge sider. Innkommende socket er en "server socket" som kontinuerlig lytter etter beskjeder i sin egen tråd og spawner separate sockets for hver nye forbindelse. Systemet er uten permanente forbindelser i transportlaget, men systemet har en forbindelsestilstand som avgjør om meldinger kan sendes eller ikke.



Figur 23: Tilstandsdiagram for nettverk

I søketilstanden forsøker roboten jevnlig å opprette en forbindelse med kontrolleren. Registreringen innebærer å sende en spesiell kommando til kontrolleren, samt en liste med id'en til alle servoer og sensorer. Når dette er gjort er forbindelsen klar og toveiskommunikasjon kan begynne.

8.5.2 Teknologi

Python socket modulen gir tilgang til "Berkley socket"-grensesnittet(BSD) som støttes i alle de populære moderne opperativsystemene. Socket modulen er en direkte oversettelse unix-kallene i BSD til pythons objektorienterte stil.

TCP er valgt for enkelhet, ettersom pakkene er garantert å nå frem i samme rekkefølge som de ble sendt.

8.6 main.py

main.py inneholder 5 klasser; Servo, Servoinfo, RobotisSensor, MessageSender og Robot. RobotisServo, Servoinfo og Sensor kjører som tråder og utnytter en utvidet utgave lib_robotis.py-biblioteket (open source bibliotek) og Robot-klassen samler disse klassene til en komplett applikasjon. MessageSender initialiserer internettkommunikasjonen med den eksterne maskinen som kjører GUI slik at man kan kontrollere servoene fra brukergrensesnittet og lese informasjonen fra servoer og sensorer.

I initialiseringen av Robot-klassen initialiseres tre lister; servos, servoinfo og sensors. Listene fylles deretter med tilkoblede servoer og sensorer. Når en servo har blitt funnet legges et objekt av type Servo til i servos-listen og et tilsvarende Servoinfo objekt legges til i servoinfo-listen. Når en sensor har blitt funnet legges et objekt av typen RobotisSensor til i listen. Servo-objektet kjører kontinuerlig og sjekker om den aktuelle servoen er satt til å være kontinuerlig eller ikke. Dersom den er satt til å være kontinuerlig og en ny verdi for farten har blitt satt i objektet settes farten til den fysiske servoen til å være lik denne farten. Dersom den ikke er satt til å være kontinuerlig og en ny verdi for vinkelen har blitt satt i objektet settes vinkelen til den fysiske servoen til å være lik denne vinkelen.

Servoinfo-objektet kjører kontinuerlig og oppdaterer alle sine egne instanser av servoinformasjon. Servoinformasjonen inkluderer last, vinkel, volt, temperatur, minimum posisjon og maksimum posisjon. Sensor-objektet kjører kontinuerlig og oppdaterer alle sine egne instanser av sensorinformasjon. Sensorinformasjonen inkluderer venstre IR-signal, midtre IR-signal, høyre IR-signal, venstre lys-signal, midtre lys-signal, høyre lys-signal, deteksjon av hindringer (en av IR-signalene er aktive), deteksjon av lys (en av lys-signalene er aktive), lyd-signal, maksimum lyd-signal, lydteller (teller antall lyder på rad) og deteksjonstid for lyd. Klassen inneholder også funksjoner for å spille av lyd.

MessageSender-objektet kjører kontinuerlig og sender all servo- og sensorinformasjon over internett til GUI. Den etablerer koblingen til internett ved å initialisere robot_network_service-objektet i RobotNetworkService-biblioteket.

Filen autonomous_controll.py inneholder klassen autonomous_controll. Her er det implementert en enkel autonom oppførsel der roboten i utgansgspunktet kjører rett frem og dersom den møter hindringer så svinger den unna hindringene. Funksjonen set_auto(set_auto) bestemmer om autonom oppførsel er aktiv. Denne klassen kan utvides slik brukeren selv ønsker for å implementere ulike nivåer av autonom oppførsel.

9 Videre Arbeid

Systemet er designet med tanke på at det skal være lett å skalere. Det er derfor lett å legge til ekstra funksjonalitet som skulle være ønskelig når man har gjort seg kjent med funksjonene i de forskjellige bibliotekene (Appendix B.2). Slik systemet står nå har vi laget funksjoner for å hente ut det meste av informasjon fra servoer og sensorer så forbedringsmulighetene ligger stort sett i å bruke denne informasjonen til noe.

Vi har laget et enkelt eksempel på autonomi der roboten svinger til venstre dersom den har en hindring foran og til høyre dersom den har en hindring foran og til venstre. Dersom den har en hindring foran, til venstre og til høyre rygger den. Denne logikken ligger på robotsiden av nettverket og aktiveres med å sende en kommando fra GUI.

Her er det stort forbedringspotensiale. Autonomien kan utvides og gjøres mye mer kompleks enn den er nå. For eksempel kan man ta i bruk lydsensoren eller lage en mer avansert algoritme for hvor roboten skal

kjøre. En annen forbedringsmulighet er å flytte logikken fra robotsiden til GUI-siden. På den måten kan en datamaskin med større prossessorkraft kjøre beregningene som må gjøres, mens rasberri pi'en bare kjører enkle kommandoer.

Et annet forbedringspotensiale vi ikke fikk tid til å implementere er at når du har satt opp et oppsett av servoer i brukergrensesnittet bør man ha mulighet til å lagre dette oppsettet. Da unngår man å måtte sette opp funksjonaliteten til servoene hver gang. Informasjonen som da bør lagres er om en servo er i en gruppe og om den skal settes med vinkel eller fart.

Videre mener vi det hadde vært fornuftig å få inn en 3D modellering av roboten i brukergrensesnittet der man hadde hatt mulighet til å definere hvor servoene står i forhold til hverandre. Grunnen til at vi mener at vi bør kunne definere plasseringen til servoene er at vi da opprettholder fleksibiliteten i systemet.

Hele systemet er designet med tanke på å kunne brukes videre andre. Prosjektet ligger for øyeblikket i versjonskontrollsystemet SVN på google code og vil kunne open sources herfra uten mye arbeid. De fleste modulene er designet etter "åpen for utvidelse - lukket for endring"-prinsippet, men det er rom for noen endringer særlig i nettverksdelen, som kunne gjort systemet enda mer egnet for gjennbruk.

Hvis man har brukt vårt system til prototyping og har kommet frem til et endelig design kan det være ønske om kode om prosjektet til et annet språk for å få raskere og mer stabil kode. Man vil i slike tilfeller ofte velge å kode i c eller c++. Dersom man ønsker dette finnes det et bibliotek skrevet i c som har mye av den samme funksjonaliteten vi har implementert i biblioteket vi har utvidet. Dette er laget av et prosjekt som heter Ikaros project. Linker til dette kan finnes i referansene [6] og [7]

10 Konklusjon

Denne rapporten har beskrevet resultatet av prosjektet til gruppe 2 i Eksperter i team - instrumentering og styring over internett med fokus på håntering av kritiske situasjoner. Vi har implementert et system for styring av roboter over internett, komplett med grafisk brukergrensesnitt og muligheter for input fra ulike sensorer. Systemet tillater roboter å sende sensordata og motta instruksjoner over internett. Vi mener det har et utall potensielle anvendelser i situasjoner hvor kritisk kompetanse er utilgjengelig lokalt, eller forholdene er uegnede for mennesker. Systemet er designet for å kunne brukes av andre og vil bli gjort tilgjengelig som et "open source"-prosjekt på google code.

Referanser

[1] Lohne Christoffer Løkseth Espen og Nøvik Håvard Dybsjord, Kerrin. Robotprosjekt. Technical report, NTNU, 2012.

[2] Barlindhaug Lars Feiring Øygard Per Øyvind og Hansen Jon Anta Buljo Hansen, Thomas Torjuul. Instrumentering og styring over internett med fokus på håndtering av kritiske situasjoner. Technical report, NTNU, 2010.

[3] Monteiro Sølve Næs Eirik Skogestad Sveinsson Hrafn Mar og Tverdal Martin Johnsen, Kenneth K. Fjernstyring av bil over internett. Technical report, NTNU, 2009.

[4] Riverbank Computing ltd. http://www.riverbankcomputing.com/software/pyqt/intro, Mai 2013.

[5] QT Project. http://qt-project.org, Mai 2013.

[6] The Ikaros Project. http://www.ikaros-project.org/, Mai 2013.

[7] The Ikaros Project. http://www.ikaros-project.org/module/dynamixel, Mai 2013.

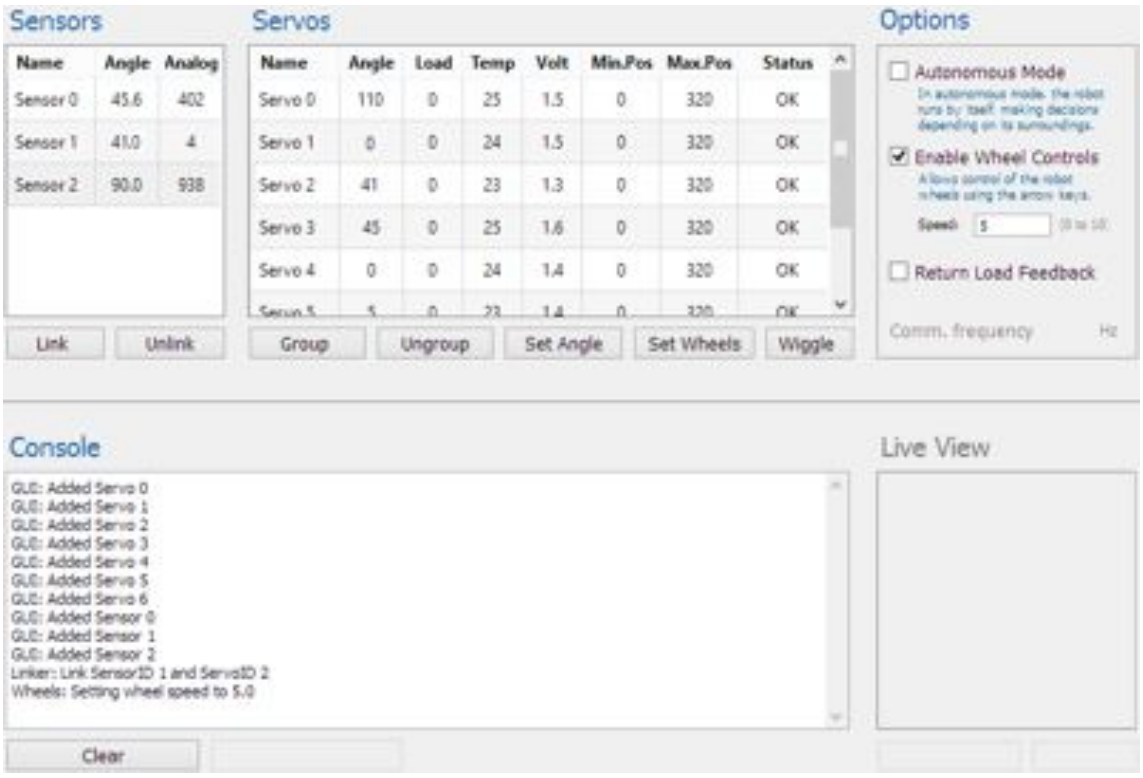
[8] LLC Trossen Robotics. http://www.trossenrobotics.com, April 2013.

A Bruksanvisning - GUI

Bruksanvisning – Grensesnitt

Hovedvinduet

Hovedvinduet kan deles inn i fire hoveddeler. Øverst til venstre er lister med sensorer og servoer, samt knapper med relevante funksjoner. Til høyre for listene er et panel med diverse opsjoner som gjelder for robotsystemet som en helhet. Nederst finner man en konsoll som viser informasjon eller feilmeldinger fra systemet og roboten. Nederst til høyre er det holdt av plass til et panel som kan vise en grafisk representasjon av sensorenes og/eller robotens tilstand. Merk at skjermbildene som vises her er redigert noe til forenkling.



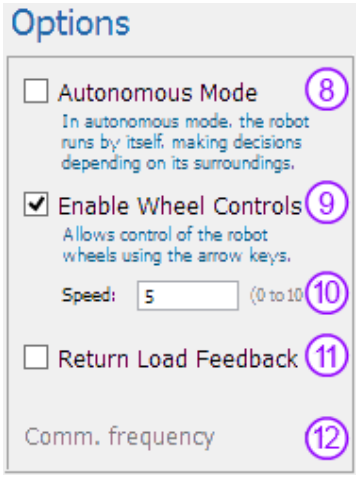
Sensor- og Servolister

Øverst til venstre i grensesnittet finner man lister med sensorer og servoer. Her vises diverse informasjon som kontinuerlig leses inn. Under listene er en rekke knapper som brukes for linking, gruppering, styring og testing av sensorer og servoer.

1. **Link**: Brukes for å koble en sensor (venstre) med en servo (høyre). Velg først både sensor og servo, og trykk deretter på knappen for å opprette koblingen. En melding vises i Console som bekreftelse på at koblingen er opprettet. Endringer i vinkel på sensorsiden vil nå overføres til tilhørende servomotor (på bildet er Sensor 1 linket til Servo 2).
2. **Unlink**: Brukes for å fjerne en kobling mellom sensor og servo. Fremgangsmåten er samme som for Link (1). En melding vises i Console som bekreftelse på at koblingen er brutt. Når koblingen er fjernet vil ikke lenger endringer i vinkel overføres fra sensor til servo.
3. **Group**: Brukes for å gruppere to servoer. Velg første servo og trykk på Group. Velg deretter andre servo og trykk på Group igjen. En melding i Console viser hvilke servoer som ble gruppert. Servoer som er gruppert vil bevege seg i takt dersom en av dem får en endring i vinkel.
4. **Ungroup**: Brukes for å fjerne en gruppering mellom to servoer. Fremgangsmåten er samme som for Group (3). En melding i Console viser hvilke servoer som ble eventuelt ble ugruppert.
5. **Set Angle:** Brukes for å sette en vinkel for en servomotor manuelt. Velg først en servo og trykk på Set Angle. Et lite vindu vil dukke opp for inntasting av vinkel. Når en gyldig vinkel er tastet inn vil servoen stille seg til denne vinkelen. Eventuelle grupperte servoer vil følge.
6. **Set Wheels:** Brukes for å angi hvilke servoer som skal fungere som hjul på roboten. Dersom man trenger flere høyre- og venstrehjul, benytt først Group (3) for å gruppere servoene som hører til hver side. Én servo må stå som hovedhjul i hver gruppering. Velg venstre hovedhjul (servo) og trykk Set Wheels. Velg deretter høyre hovedhjul og trykk Set Wheels igjen. En melding i Console vil vise hvilke hjul som er valgt. Roboten kan nå styres vha. tastaturet når Enable Wheel Controls (9) er aktiv.
7. **Wiggle**: Brukes for å identifisere servoer visuelt. Velg først en servo i listen og trykk på Wiggle. Valgt servo vil nå gi fra seg et lite vink slik at man vet hvilken fysisk servo dette er.

Options

I dette panelet kan man sette diverse globale settings som angår roboten og systemet som en helhet.



8. **Autonomous Mode**: Hvis slått på aktiveres robotens autonome styring. Roboten vil da ferdes rundt på egenhånd og respondere på omgivelsene så godt den kan. Den autonome styringen kan ha varierende oppgaver og kompleksitet avhengig av hvordan den er definert i koden.
9. **Enable Wheel Controls**: Dette aktiverer styring av robotens hjul ved hjelp av tastaturet. Det er viktig å angi hvilke hjul som skal brukes med Group (3) og Set Wheels (6). For best resultat, bruk knappene W (frem), A (venstre), S (bak), D (høyre), og mellomrom (stopp). Merk at for enkelte deler av grensesnittet vil ikke tastetrykk registreres ordentlig; prøv da å flytte fokus til en hvilken som helst vanlig knapp i vinduet.
10. **Speed**: Her settes hastigheten på robothjulene ved å taste inn ønsket verdi. Merk begrensningen i verdiområde (0 til 10). NB! Det kan være lurt å begynne med en lav hastighet dersom det er første gang man skal kjøre (for eksempel 1 til 3).
11. **Return Load Feedback** [ikke implementert]**:** Når denne er aktiv skal servoene returnere hvilken belastning eller motstand de opplever når de blir satt til en gitt vinkel. Høy motstand kan bety at en arm har støtt på en solid gjenstand eller løfter noe tungt. Denne informasjonen vil kunne brukes til å gi brukeren fysisk tilbakemelding i for eksempel en hanske med sensorer og feedbacksystem.
12. **Comm. Frequency** [ikke implementert]**:** Her kan man sette hvor hyppig informasjon fra sensorer skal oppdateres og sendes for styring av servoer. Dette vil ha stor innvirkning på hvor jevnt servoene beveger seg om de styres med for eksempel en hanske. NB! En for høy frekvens kan gjøre at hardware for sensormålinger overbelastes og begynner å henge seg opp. Dersom dette skjer kan man prøve seg frem til man finner en stabil frekvens.

Console

Konsollen nederst i vinduet kommer med informasjon og feilmeldinger om roboten og hele systemet. I en kodemodul kan man bestemme hvilke typer meldinger som skal skrives til konsollen. Ved videre utvikling kan disse innstillingene være tilgjengelige direkte fra grensesnittet nær konsollen. For å tømme konsollen kan man trykke på knappen Clear like under.

B Bruksanvisning - Robotklient

Softwaredokumentasjon av robot

Initialisering

Initialiseringen av robotsiden gjøres ved å kjøre funksjonen Robot(). Dette starter initialiseringen av Robot-klassen som igjen starter initialiseringen av de videre klassene. Først initialiseres USB2Dynamixelen (dyn) med at funksjonen lib_robotis.USB2Dynamixe('port', baudrate). Baudrate skal settes til 1000000 og port skal være den COM-porten USB2Dynamixelen er tilkoblet (f.eks. 'COM3' (windows) eller '/dev/ttyUSB0' (linux)).

Deretter fylles en liste med servo-id'er og en liste med sensor-id'er med å kjøre respektivt find_servos() og find_sensors(). Det kjøres så en løkke som går gjennom hele listen med servo-id'er og for hver av dem initialiserer et servo-objekt og legger dette til i listen kalt servos[]. Disse initialiseres ved å først kjøre servo=lib_robotis.Robotis_servo(dyn, _servo_ids[i]), for deretter å sende dette objektet inn i initialiseringen av selve servo-objekter ved å kjøre servos.append(Servo(servo)). Servo(servo) initialiserer alle aktuelle verdier for servo-objektet. Deretter gjøres det samme for sensorer med å kjøre respektivt sensor=lib_robotis.Robotis_sensor(dyn, _sensor_ids[i]) og sensors.append(RobotisSensor(sensor)).

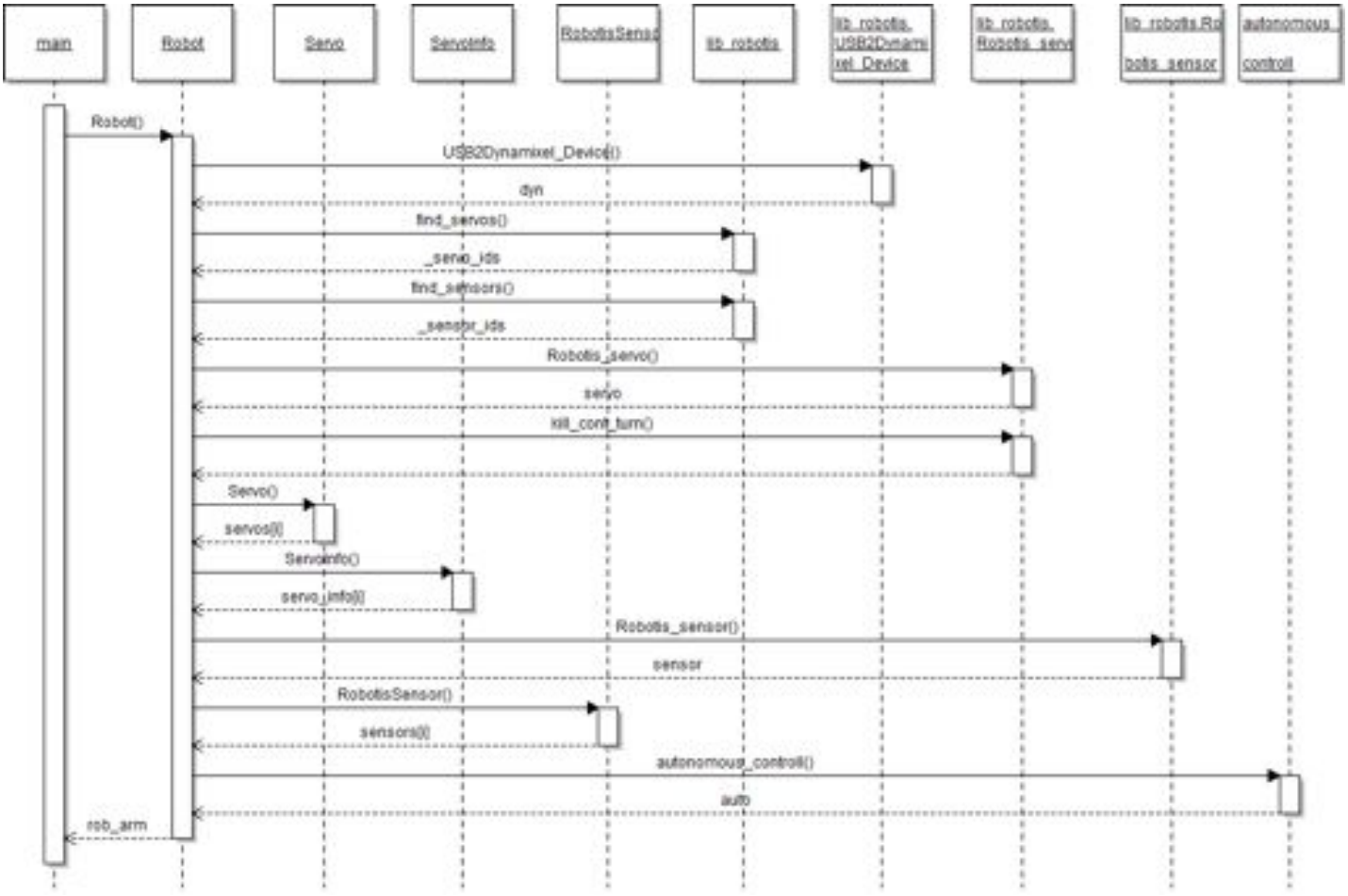
Til slutt initialiseres objektet auto med å kjøre autonomous_controll.autonomous_controll(self.servos[3], self.servos[6], self.servos[4], self.servos[5], self.sensors[0]). Dette gjør at man kan sette roboten i autonom-modus med å kjøre set_auto(1) og ta roboten ut av autonom modus ved å kjøre set_auto(0).

Bruk

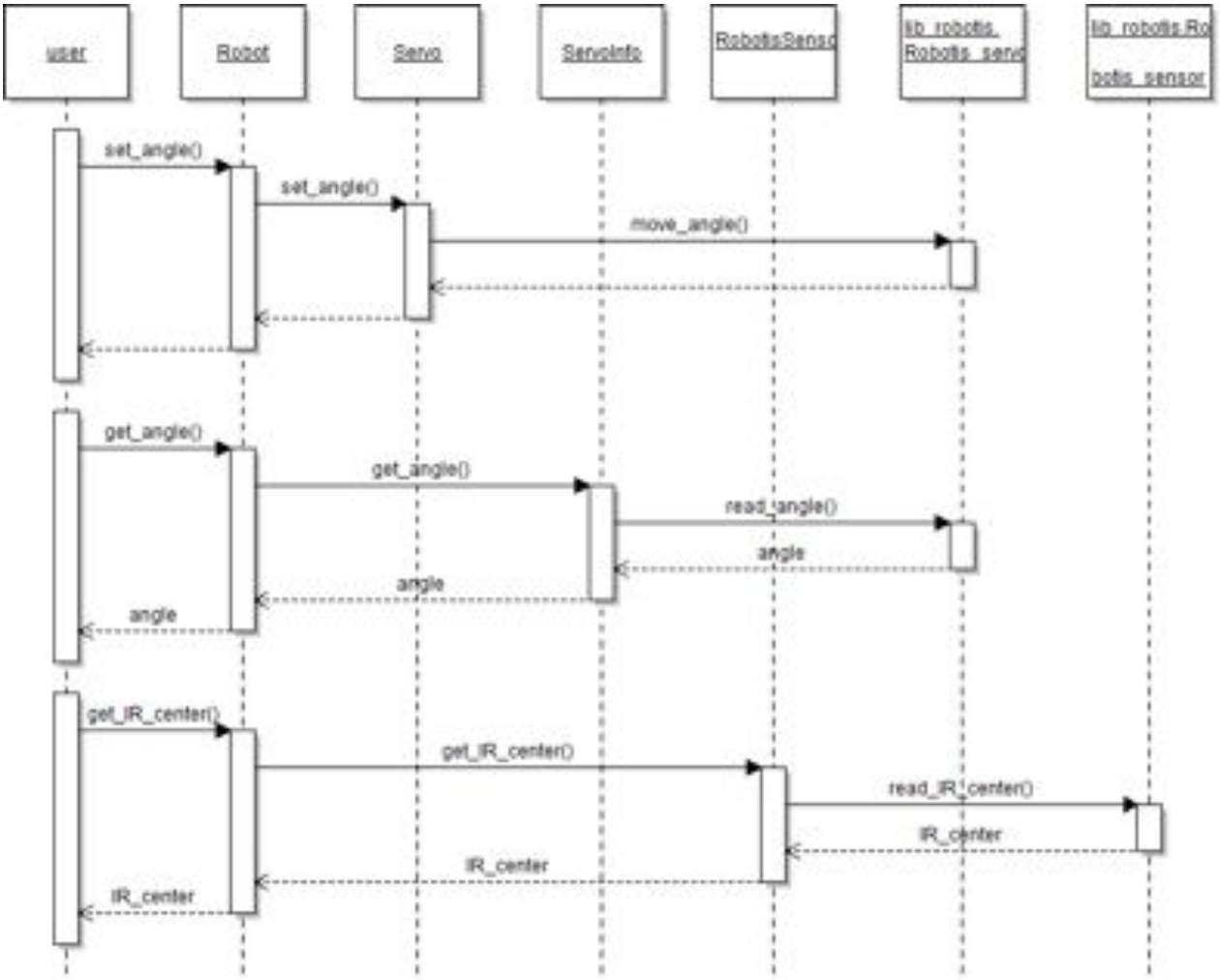
Applikasjonen er bygd slik at man bare skal trenge å interface Robot-klassen. I denne klassen vil en typisk funksjon se slik ut: set_angle(self, angle, _id). Variablen som heter self kan oversees og trenger ikke å settes når man bruker funksjonen. Variablen angle vil representere en vinkel (i grader) og _id vil representere plassen i servolisten der du finner servoen du ønsker å styre. F.eks. vil set_angle(90,0) sette servoen på plass nr 0 i servolisten (servos[0]) til 90 grader.

For en komplett oversikt over alle funksjonene i Robot-klassen, se koden i Vedlegg Dokumentasjon av Robotsoftware.

B.1 Sekvensdiagram for initialisering av Robot

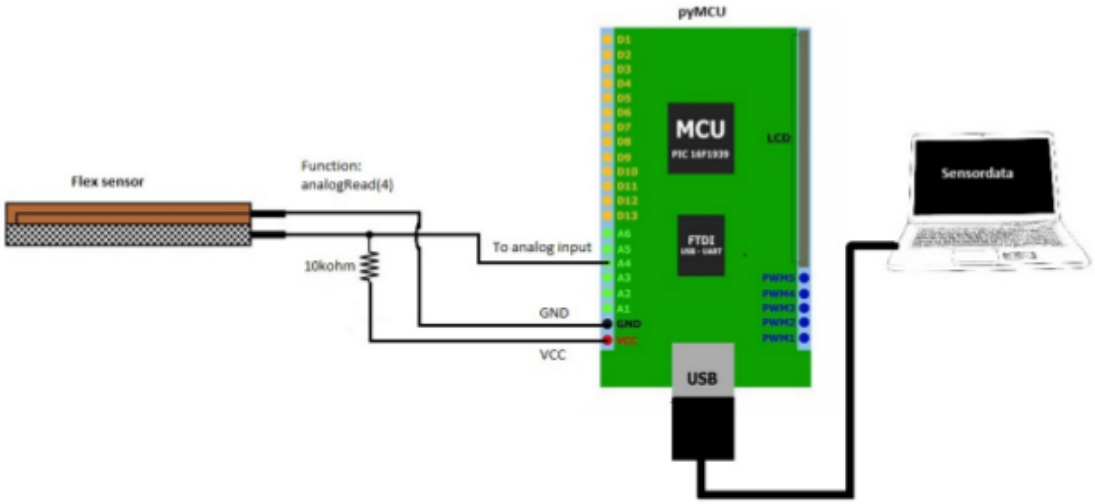


B.2 Sekvensdiagram - Eksempel på bruk av Robot

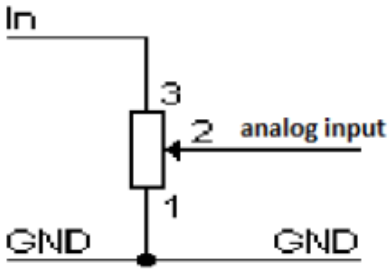


C Hvordan koble opp pyMCU med Sensorer

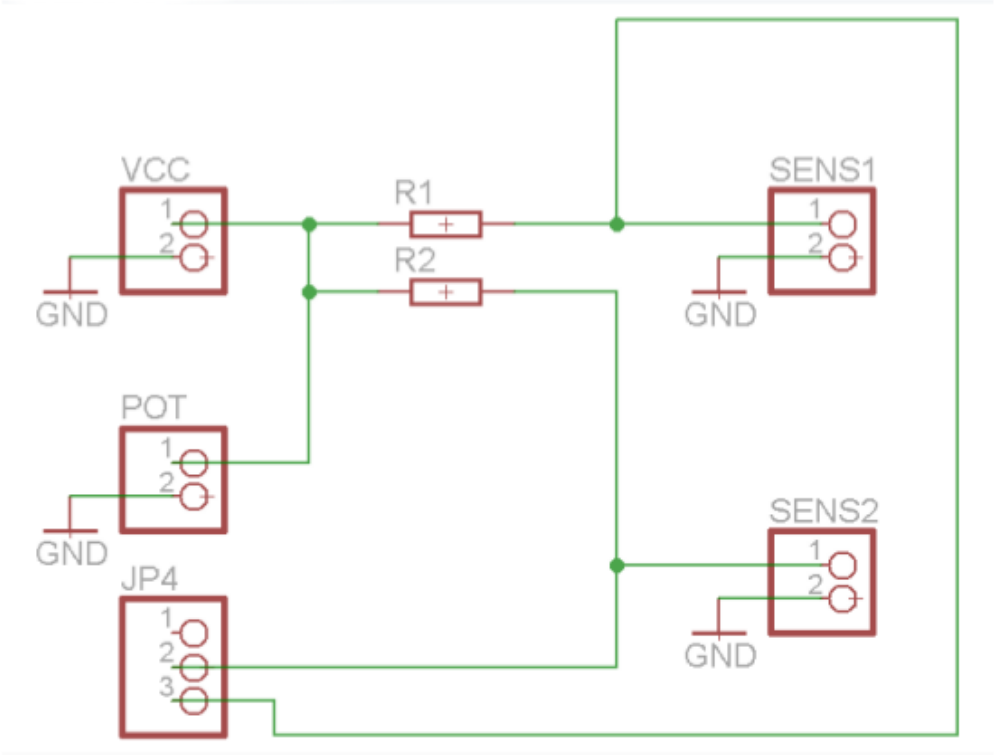
Flexsensor:



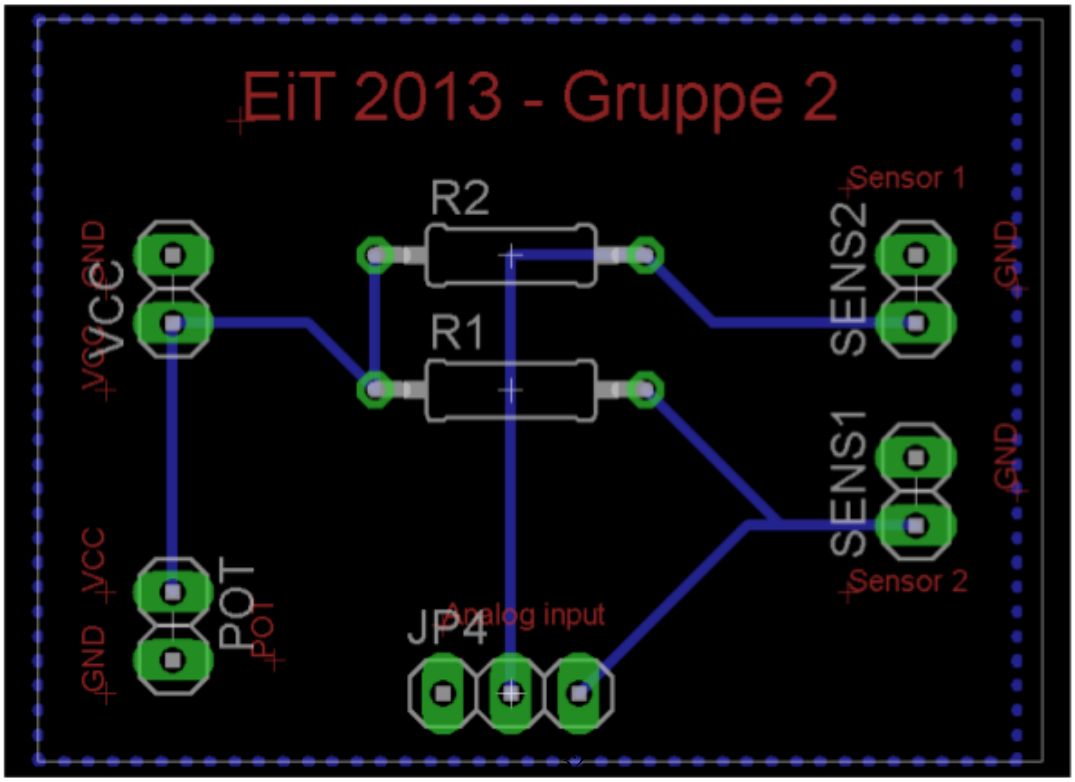
Potensiometer:



D Kretssjema for oppkoblingskrets mellom pyMCU med Sensorer



Utlegg:



E Dokumentasjon Servobibliotek (lib_robotis.py)

lib_robotis.py

lib_robotis.py contains software for controlling Robotis Dynamixel servos and sensors from python using a USB2Dynamixel Adapter.

The library uses the native python modules: threading, time, sys, optparse and math.

The library uses the external module serial. This is a module developed by the pySerial project. The module is Open Source. More information could be found at pyserial.sourceforge.net.

Classes

USB2Dynamixel_Device

This is a class that manages the serial port connection between servos on the same bus. Contains methods for thread safe operation.

Methods

acq_mutex

locks the opbject for operation from other threads

rel_mutex

Releases the thread lock.

send_serial(msg)

Sends the message *msg to* the serial port

read_serial

Recieves a message sent to the serial port

_open_serial(baudrate)

Internal method used in the initialization of the object to open a serial connection. Raises a runtime error exception if no serial port is found.

Robotis

This class is a superclass for Robotis Dynamixel units. It contains the besic methods for communication with the units, and to determine what kind of unit it is. Need a initialized **USB2Dynamixel_Device** object to be able to communicate with servos.

Methods

read_model_number

Reads the model number from the unit. Is used for identifying units

write_id

Changes the id of the robotis unit. Be carful to not let more than one unit on the same bus have the same id. This will cause unecpected behavior.

read_id
reads the bus id from the servo.

read_address
reads the the bytes on a memory adress on a servo. This is the main entrypoint for getting info from the servos and sensors.

write_address
writes data to the memory adress specified. This is the main entrypoint for controlling the behavior of the servos and sensors.

send_instruction
A method that compiles a message that should be sent to the serial port and tries to send it, and recieve the reply. Returns the reply from the serial port.

process_err
Method for raising a RuntimeErrror Exception.

recieve_reply
Method used by the *send_instruction* method. Recieves the reply from the serial port, and extracts and returns the useful info.

send_serial
Method that sends the actual method in ASCII to the serial port.

print_all_adresses
Method for debugging. Prints all the memory adresses on the Robotis unit.

__calc_checksum
calculates the checksum for a message. The checksum is included int the message written to the serial port.

Robotis_Sensor
This class is a subclass of robotis that includes methods for extracting info and controlling robotis servos. This is done by a series of read/write methods. More info can be obtained from robotis servo datasheets.

Robotis_Servo
This class is a subclass of robotis that includes methods for extracting info and controlling robotis sensors. This is done by a series of read/write methods. More info can be obtained from robotis sensor datasheets.

RobotisError
A subclass of exception. Is used for passing non-critical error messages upwards in the controller hierarchy.

Functions

find_servos

Function that finds all servos on the bus, by by trying to connect to all ids and check if there is a unit, and check the model number of this unit at the specified id. Can be limited to search on a lower range of ids to make the scan og faster.

find_sensors

Function that finds all sensors on the bus, by by trying to connect to all ids and check if there is a unit, and check the model number of this unit at the specified id. Can be limited to search on a lower range of ids to make the scan og faster.

find_all_units

Function that returns all ids on the bus with a unit connected.

recover_servo

Recovers a bricked servo by booting into diagnostic bootloader and resetting it.

F Dokumentasjon Robotsoftware (main.py)

main.py

main.py contains software to create a complete servo and sensor application. It is based on the lib_robotis library (created by the Georgia Tech Research Corporation) and also takes use of the RobotNetworkService and autonomous_controll library created by us.

Main.py uses the native python modules: threading, math and time.

Classes

Servo

This class represents a single servo. It runs as a thread and continuously checks if the user/application has set a new value for either velocity (in continuous mode) or angle (not in continuous mode) and if there is a new value it sets the value with its set method which in turn calls the corresponding method in the lib_robtis library. The other methods in this class are described below

Methods

set_mode(continuous)

If continuous is set to 1 the function calls the init_cont_turn() function in the lib_robotis library. For any other value the kill_cont_turn() function from the same library is called.

w*iggle_servo*

Drives the servo back and forth by +/-10 degrees. This function is used to easily identify servo, as a call to this function will wiggle the servo in the physical application.

Servoinfo

This class represents the information that can be extracted from a single servo. It runs as a thread and continuously updates its variables by calling the corresponding read functions in the lib_robotis library. It contains a series of get functions to read these variables.

RobotisSensor

This class represents a single sensor. It runs as a thread and continuously updates its variables by calling the corresponding read functions in the lib_robotis library. It contains a series of get functions to read these variables.

Robot

This class represents a collection of servos and sensors and is the class that should be used by the user/other applications to interface the robot application. It contains a series of set and get functions that corresponds to the set and get functions in the Servo and Sensor class. The _id input of these functions represent the servo/sensors place in the list initialized in this classes init function.

MessageSender

This class runs as a thread and sends information about the servos and sensors of the application to the network every 0.1 seconds.

G Nettverksprotokoll

Complete protocol in extended backus naur form

<Message> ::= “[” , <Command>, “]”
<Command> ::= “SRV_CMD”, <Cmd id>, <Servo id>, VALUE |
“SNS_CMD”, <Cmd id>, <Servo id>, VALUE |
“SRV_GROUP” , <Servo list> |
“SRV_UNGROUP, <Servo id> |
“SNS_GET”, <Value id> , <Sensor id> |
“SRV_GET”, <Value id>, <Servo id>, |
“REG_SNS”, <ID_LIST> |
“REG_SRV”, <ID_LIST> |
“SRV_VALUE <Value id>, <Servo id>, <Value> |
“SNS_VALUE <Value id>, <Sensor id>, <Value> |

<Cmd id> ::= “SET”, <Value id> | “GET”, <Value id> | “WGL” | “SRV_GET” | “SNS_GET” | AUTO
<Value id> ::= “ANGL” | “AVEL” | “VOLT” | “TEMP” | “MAXP” | “MINP” | “MODE” |
“TEMP” | “IRL” | “IRR” | “IRC” | “LL” | “LR” | “LC” | “SOUND”
<Id list> ::= {<Servo id>}| <Servo id>
<Servo id> ::= 1,..., 255
<Sensor id> ::= 1,..., 255
VALUE ::= string

SRV_CMD: Commands for the servos
CMD_ID: The command types
SET: Sets a servo value at some given value
AUTO: sets autonmous mode on/off
GET: tells the robot to send the latest reading for some value
WGL: Tells a given servo to wiggle
SNS_GET: Tells the robot to send all sensor ids
SRV_GET: Tells the robot send all servo ids
SNS_VALUE / SRV_VALUE: A message containing the value for a given parameter for a given sensor or servo

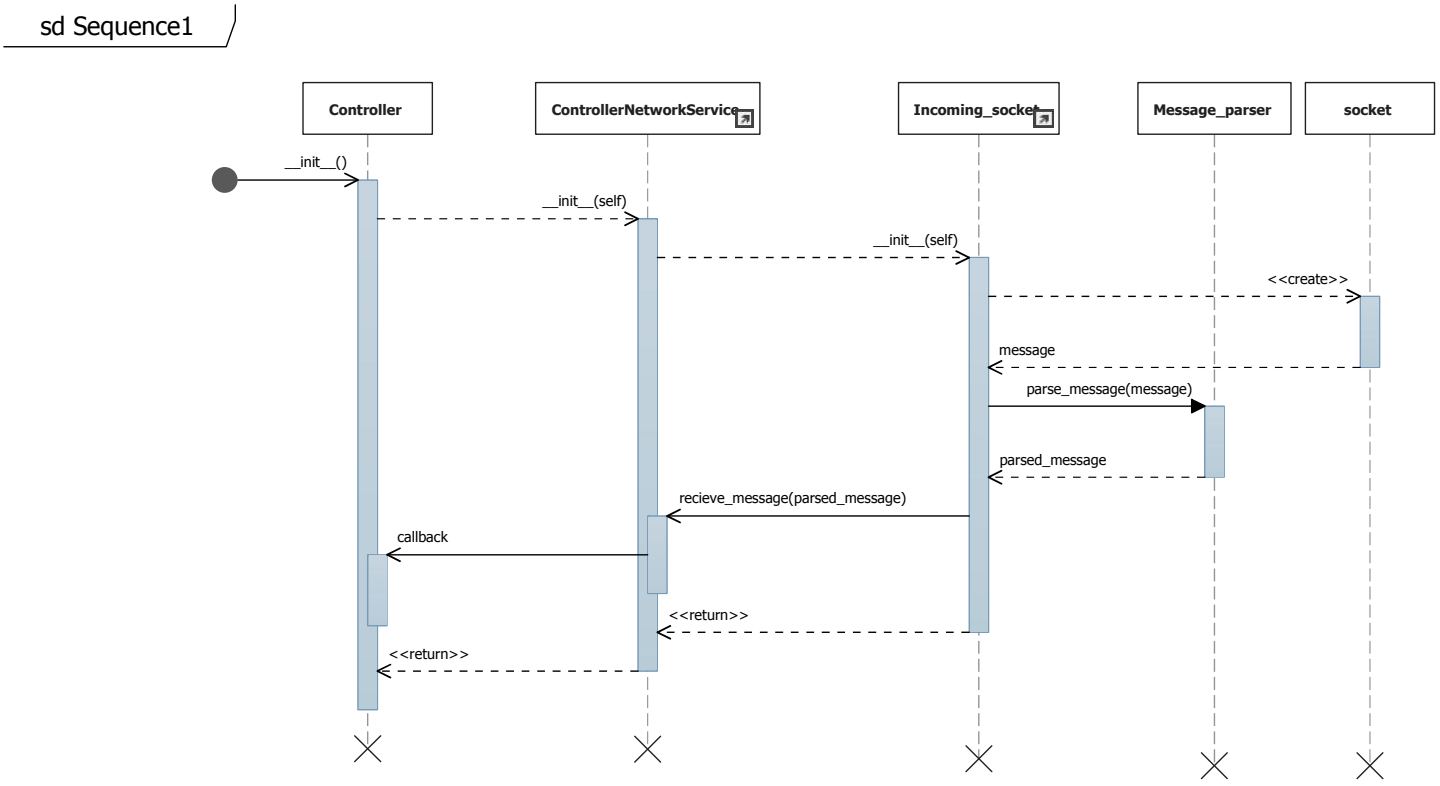
SERVO_ID: The servos id number
SNS_CMD: Sensor commands

Value id: Ids for the different value types:

- ANGL: What angle a servo is set to
- AVEL: Angle velocity for servo
- VOLT: Voltage for servo
- TEMP: Temprature for servo
- MAXP: Maximum angle for servo
- MINP: Minimum angle for servo
- MODE: Continous mode on or off
- IRL/IRR/IRC: Current ir sensor reading for left, right and center light sensor
- LL/LR/LC: Current light sensor reading for left, right and center light sensor
- SOUND: Current reading for sound sensor

SRV_GROUP: Tells the robot to group a given set of servos
SRV_UNGROUP: Tells the robot to ungroup a given set of servos

H Sekvensdiagram for initialisering og callback



I Kildekode

I.1 main.py

```
import lib_robotis
import threading
import math
import time
import RobotNetworkService
import autonomous_controll

class MessageSender(threading.Thread):
    def __init__(self, rob_arm, net):
        super(MessageSender, self).__init__()
        self.net = net
        self.rob_arm = rob_arm
        self.start()

    def run(self):
        '''continuously sends information about servos and sensors'''
        while(1):
            for i in range(self.rob_arm.get_servo_len()):
                self.net.send_load(i, self.rob_arm.get_load(i))
                self.net.send_voltage(i, self.rob_arm.get_voltage(i))
                self.net.send_temp(i, self.rob_arm.get_temp(i))
                self.net.send_angle(i, self.rob_arm.get_angle(i))
                self.net.send_min_pos(i, rob_arm.get_min_pos(i))
                self.net.send_max_pos(i, rob_arm.get_max_pos(i))
                time.sleep(0.1)
            for i in range(self.rob_arm.get_sensor_len()):
                self.net.send_IR_left(i, rob_arm.get_IR_left(i))
                self.net.send_IR_center(i, rob_arm.get_IR_center(i))
                self.net.send_IR_right(i, rob_arm.get_IR_right(i))
                self.net.send_light_left(i, rob_arm.get_light_left(i))
                self.net.send_light_center(i, rob_arm.get_light_center(i))
                self.net.send_light_Right(i, rob_arm.get_light_right(i))


class RobotisSensor(threading.Thread):
    def __init__(self, sensor):
        super(RobotisSensor, self).__init__()
        self.sensor = sensor
        self.IR_left = 0.0
        self.IR_center = 0.0
        self.IR_right = 0.0
        self.light_left = 0.0
        self.light_center = 0.0
        self.light_right = 0.0
        self.obstacle_det = 0
        self.light_det = 0
        self.sound = 0.0
        self.max_sound = 0.0
        self.sound_count = 0.0
        self.sound_detected_time = 0.0
        self.start()

```

```
53      def run(self):
54          '''continuously updates its own variables'''
55          while(1):
56              try:
57                  self.IR_left = self.sensor.read_IR_left()
58                  self.IR_center = self.sensor.read_IR_center()
59                  self.IR_right = self.sensor.read_IR_right()
60                  self.light_left = self.sensor.read_light_left()
61                  self.light_center = self.sensor.read_light_center()
62                  self.light_right = self.sensor.read_light_right()
63                  self.obstacle_det = self.sensor.obstacle_detected()
64                  self.light_det = self.sensor.light_detected()
65                  self.sound = self.sensor.read_sound()
66                  self.max_sound = self.sensor.read_max_sound()
67                  self.sound_count = self.sensor.read_sound_count()
68                  self.sound_detected_time = self.sensor.read_sound_detected_time()
69              except lib_robotis.RobotisError as e:
70                  net.send_error(e.error)
71
72      def get_IR_left(self):
73          return self.IR_left
74
75      def get_IR_center(self):
76          return self.IR_center
77
78      def get_IR_right(self):
79          return self.IR_right
80
81      def get_light_left(self):
82          return self.light_left
83
84      def get_light_center(self):
85          return self.light_center
86
87      def get_light_right(self):
88          return self.light_right
89
90      def obstacle_detected(self):
91          return self.obstacle_det
92
93      def light_detected(self):
94          return self.light_det
95
96      def get_sound(self):
97          return self.sound
98
99      def get_max_sound(self):
100         return self.max_sound
101
102     def get_sound_count(self):
103         return self.sound_count
104
105     def reset_sound_count(self):
106         self.sensor.reset_sound_detected_count()
107
108     def get_sound_detected_time(self):
```

```
109         return self.sound_detected_time
110
111     def buzzer_play_melody(self, melody):
112         self.sensor.buzzer_play_melody(melody)
113
114
115 class Servo(threading.Thread):
116     def __init__(self, servo):
117         super(Servo, self).__init__()
118         self.servo = servo
119         self.angle = math.degrees(servo.read_angle())
120         self.last_angle = self.angle
121         self.velocity = 0.0
122         self.last_velocity = 0.0
123         self.continuous = False
124         self.angle_lock = threading.Lock()
125         self.vel_lock = threading.Lock()
126         self.start()
127
128     def run(self):
129         '''if the servo is continous and has a new velocity value it sets the
                velocity to
130         the new value. if the servo is not continous and has a new anlge value it
                sets the
131         angle to the new value.'''
132         while(1):
133             if(self.continuous):
134                 self.vel_lock.acquire()
135                 if(self.velocity != self.last_velocity):
136                     try:
137                         self.servo.set_angvel(self.velocity)
138                     except lib_robotis.RobotisError as e:
139                         net.send_error(e.error)
140                     self.last_velocity = self.velocity
141                 self.vel_lock.release()
142             else:
143                 self.angle_lock.acquire()
144                 if(self.angle != self.last_angle):
145                     try:
146                         self.servo.move_angle(math.radians(self.angle), 5, False)
147                     except lib_robotis.RobotisError as e:
148                         net.send_error(e.error)
149                     self.last_angle = self.angle
150                 self.angle_lock.release()
151
152     def set_angle(self, angle):
153         self.angle_lock.acquire()
154         self.angle = angle
155         self.angle_lock.release()
156
157     def set_velocity(self, velocity):
158         self.vel_lock.acquire()
159         self.velocity = velocity
160         self.vel_lock.release()
161
162     def set_mode(self, continuous):
```

```
163         '''If continuous=1 this function initializes continuous turn.
164         If continuous!=1 this function kills continuous turn.'''
165         if(continuous):
166             try:
167                 self.servo.init_cont_turn()
168             except lib_robotis.RobotisError as e:
169                 net.send_error(e.error)
170         else:
171             try:
172                 self.servo.kill_cont_turn()
173             except lib_robotis.RobotisError as e:
174                 net.send_error(e.error)
175         self.continuous = continuous
176 
177     def get_servo(self):
178         return self.servo
179 
180     def wiggle_servo(self):
181         '''Use this function to identify the servo. When called the servo will
182         wiggle back and forth.'''
183         self.angle_lock.acquire()
184         try:
185             if(self.angle > 170):
186                 self.servo.move_angle(math.radians(self.angle-10), 10, True)
187                 self.servo.move_angle(math.radians(self.angle), 10, True)
188                 self.servo.move_angle(math.radians(self.angle-10), 10, True)
189                 self.servo.move_angle(math.radians(self.angle), 10, True)
190             else:
191                 self.servo.move_angle(math.radians(self.angle+10), 10, True)
192                 self.servo.move_angle(math.radians(self.angle), 10, True)
193                 self.servo.move_angle(math.radians(self.angle+10), 10, True)
194                 self.servo.move_angle(math.radians(self.angle), 10, True)
195         except lib_robotis.RobotisError as e:
196             net.send_error(e.error)
197         self.angle_lock.release()
198 
199 
200 class Servoinfo(threading.Thread):
201     def __init__(self, servo):
202         super(Servoinfo, self).__init__()
203         self.load = 0.0
204         self.angle = 0.0
205         self.voltage = 0.0
206         self.temp = 0.0
207         self.min_pos = 0.0
208         self.max_pos = 0.0
209         self.pos_lock = threading.Lock()
210         self.servo = servo
211         self.start()
212 
213     def run(self):
214         '''continuously updates its own variables'''
215         while(1):
216             try:
217                 self.load = self.servo.read_load()
218                 self.angle = math.degrees(self.servo.read_angle())
```

```
                self.voltage = self.servo.read_voltage()
                self.temp = self.servo.read_temperature()
                self.pos_lock.acquire()
                self.min_pos = math.degrees(self.servo.read_min_pos())
                self.max_pos = math.degrees(self.servo.read_max_pos())
                self.pos_lock.release()
            except lib_robotis.RobotisError as e:
                net.send_error(e.error)

    def get_load(self):
        return self.load

    def get_angle(self):
        return self.angle

    def get_voltage(self):
        return self.voltage

    def get_temp(self):
        return self.temp

    def get_min_pos(self):
        return self.min_pos

    def get_max_pos(self):
        return self.max_pos

    def set_min_pos(self, angle):
        self.pos_lock.acquire()
        try:
            self.servo.set_min_pos(math.radians(angle))
        except lib_robotis.RobotisError as e:
                net.send_error(e.error)
        self.pos_lock.release()

    def set_max_pos(self, angle):
        self.pos_lock.acquire()
        try:
            self.servo.set_max_pos(math.radians(angle))
        except lib_robotis.RobotisError as e:
                net.send_error(e.error)
        self.pos_lock.release()


class Robot:
    def __init__(self):
        try:
            self.dyn = lib_robotis.USB2Dynamixel_Device('COM3', 1000000)
            _servo_ids = lib_robotis.find_servos(self.dyn, 30)
            _sensor_ids = lib_robotis.find_sensors(self.dyn, 99, 101)
        except lib_robotis.RobotisError as e:
                net.send_error(e.error)
        self.servo_len = _servo_ids.__len__()
        self.sensor_len = _sensor_ids.__len__()
        self.servos = []
        self.servo_info = []
```

```
275         self.sensors = []
276 
277         for i in range(self.servo_len):
278             try:
279                 servo = lib_robotis.Robotis_Servo( self.dyn, _servo_ids[i])
280                 servo.kill_cont_turn()
281             except lib_robotis.RobotisError as e:
282                 net.send_error(e.error)
283             self.servos.append(Servo(servo))
284             self.servo_info.append(Servoinfo(servo))
285 
286         for i in range(self.sensor_len):
287             try:
288                 sensor = lib_robotis.Robotis_Sensor( self.dyn, _sensor_ids[i])
289             except lib_robotis.RobotisError as e:
290                 net.send_error(e.error)
291             self.sensors.append(RobotisSensor(sensor))
292         self.auto = autonomous_controll.autonomous_controll(self.servos[3], self.
                servos[6], self.servos[4], self.servos[5], self.sensors[0])
293 
294     #START OF SERVO FUNCTIONS
295     def set_angle(self, angle, _id):
296         self.servos[_id].set_angle(angle)
297 
298     def set_velocity(self, velocity, _id):
299         self.servos[_id].set_velocity(velocity)
300 
301     def set_mode(self, continous, _id):
302         self.servos[_id].set_mode(continous)
303 
304     def set_min_pos(self, angle, _id):
305         self.servo_info[_id].set_min_pos(angle)
306 
307     def set_max_pos(self, angle, _id):
308         self.servo_info[_id].set_max_pos(angle)
309 
310     def get_load(self, _id):
311         return self.servo_info[_id].get_load()
312 
313     def get_angle(self, _id):
314         return self.servo_info[_id].get_angle()
315 
316     def get_voltage(self, _id):
317         return self.servo_info[_id].get_voltage()
318 
319     def get_temp(self, _id):
320         return self.servo_info[_id].get_temp()
321 
322     def get_min_pos(self, _id):
323         return self.servo_info[_id].get_min_pos()
324 
325     def get_max_pos(self, _id):
326         return self.servo_info[_id].get_max_pos()
327 
328     def wiggle_servo(self, _id):
329         '''Use this function to identify the servo with id = _id. When
```

```
330             called the servo will wiggle back and forth.'''
331             self.servos[_id].wiggle_servo()
332
333         def get_servos(self):
334             return self.servos
335
336         def get_servo_len(self):
337             return self.servo_len
338
339         def set_auto(self, set_auto):
340             '''If set_auto=1 the the robot will be set in autonomous mode.
341             If set_auto!=1 the robot will be set to normal mode'''
342             self.auto.set_auto(set_auto)
343
344         def set_auto_vel(self, vel):
345             self.auto.set_velocity(vel)
346
347         #START OF SENSOR FUNCTIONS
348         def buzzer_play_melody(self, _id, melody):
349             self.sensors[_id].buzzer_play_melody(melody)
350
351         def get_IR_left(self, _id):
352             return self.sensors[_id].get_IR_left()
353
354         def get_IR_center(self, _id):
355             return self.sensors[_id].get_IR_center()
356
357         def get_IR_right(self, _id):
358             return self.sensors[_id].get_IR_right()
359
360         def get_light_left(self, _id):
361             return self.sensors[_id].get_light_left()
362
363         def get_light_center(self, _id):
364             return self.sensors[_id].get_light_center()
365
366         def get_light_right(self, _id):
367             return self.sensors[_id].get_light_right()
368
369         def obstacle_detected(self, _id):
370             return self.sensors[_id].obstacle_detected()
371
372         def light_detected(self, _id):
373             return self.sensors[_id].light_detected()
374
375         def get_sound(self, _id):
376             return self.sensors[_id].get_sound()
377
378         def get_max_sound(self, _id):
379             return self.sensors[_id].get_max_sound()
380
381         def get_sound_count(self, _id):
382             return self.sensors[_id].get_sound_count()
383
384         def get_sound_detected_time(self, _id):
385             return self.sensors[_id].get_sound_detected_time()
```

```
386
387     def get_sensor_len(self):
388         return self.sensor_len
389
390     def get_sensor(self, _id):
391         return self.sensors[_id]
392
393     def wave_if_3_claps(self, _id):
394         '''This function only works if servo[1] and servo[2] are part of the
                robotarm'''
395         if(self.sensors[_id].get_sound_count() == 3):
396             self.buzzer_play_melody(_id, 3)
397             self.servos[0].get_servo().move_angle(math.radians(0), 10, False)
398             time.sleep(0.2)
399             self.servos[1].get_servo().move_angle(math.radians(-20), 10, False)
400             time.sleep(0.2)
401             self.servos[2].get_servo().move_angle(math.radians(0), 10, False)
402             time.sleep(0.2)
403             self.servos[2].get_servo().move_angle(math.radians(70), 10, False)
404             time.sleep(0.2)
405             self.servos[2].get_servo().move_angle(math.radians(0), 10, False)
406             time.sleep(0.2)
407             self.servos[2].get_servo().move_angle(math.radians(70), 10, False)
408             time.sleep(1)
409             self.sensors[_id].reset_sound_count()
410             time.sleep(2)
411             return True
412         else:
413             return False
414
415
416
417 ####### MAIN #######
418
419 rob_arm = Robot()
420 net = RobotNetworkService.robot_network_service(rob_arm)
421
422 sender = MessageSender(rob_arm, net)
423
424 rob_arm.buzzer_play_melody(0, 0)
425
426 has_waved = False
427 while(not has_waved):
428     has_waved = rob_arm.wave_if_3_claps(0)
```

I.2 lib_robotis.py

```
1 #!/usr/bin/python
2 #
3 # Copyright (c) 2009, Georgia Tech Research Corporation
4 # All rights reserved.
5 #
6 # Redistribution and use in source and binary forms, with or without
7 # modification, are permitted provided that the following conditions are met:
8 #     * Redistributions of source code must retain the above copyright
9 #       notice, this list of conditions and the following disclaimer.
```

```
#     * Redistributions in binary form must reproduce the above copyright
#       notice, this list of conditions and the following disclaimer in the
#       documentation and/or other materials provided with the distribution.
#     * Neither the name of the Georgia Tech Research Corporation nor the
#       names of its contributors may be used to endorse or promote products
#       derived from this software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY GEORGIA TECH RESEARCH CORPORATION ''AS IS'' AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL GEORGIA TECH BE LIABLE FOR ANY DIRECT, INDIRECT,
# INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
# LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA,
# OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
# LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
# OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
# ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
## Controlling Robotis Dynamixel RX-28 & RX-64 servos from python
## using the USB2Dynamixel adaptor.
#
## Authors: Travis Deyle, Advait Jain & Marc Killpack (Healthcare Robotics Lab, Georgia Tech.)
#
# ##########################################################################################

#
# This software is rewritten and extended as a part of the subject TMM4850 - Experts in
# Teamwork - Instrumentation and Control via Internet, Focus on Handling critical Situations
# given at NTNU, Norway. The additions of code follow the same disclaimer and licence as above.
#
## Authors: Andreas Carlsen, P l Fornebo, Eivind Andr  Taft , Erik Vattekar
##          and Eirik Hexeberg Henriksen
#
# ##########################################################################################


import serial
import threading
import time
import sys, optparse
import math


class USB2Dynamixel_Device():
    ''' Class that manages serial port connection between servos on same bus
    '''
    def __init__( self, dev_name = '/dev/ttyUSB0', baudrate = 57600 ):
        try:
            self.dev_name = int(dev_name) # stores the serial port as 0-based
```

```
                integer for Windows
58          except:
59              self.dev_name = dev_name # stores it as a /dev-mapped string for
                    Linux / Mac
60
61          self.lock = threading.Lock()
62          self.servo_dev = None
63
64          self.acq_mutex()
65          self._open_serial( baudrate )
66          self.rel_mutex()
67
68      def acq_mutex(self):
69          self.lock.acquire()
70
71      def rel_mutex(self):
72          self.lock.release()
73
74      def send_serial(self, msg):
75          # It is up to the caller to acquire / release mutex
76          self.servo_dev.write( msg )
77
78      def read_serial(self, nBytes=1):
79          # It is up to the caller to acquire / release mutex
80          rep = self.servo_dev.read( nBytes )
81          return rep
82
83      def _open_serial(self, baudrate):
84          try:
85              self.servo_dev = serial.Serial(self.dev_name, baudrate, timeout=1.0)
86              # Closing the device first seems to prevent "Access Denied" errors on
                    WinXP
87              # (Conversations with Brian Wu @ MIT on 6/23/2010)
88              self.servo_dev.close()
89              self.servo_dev.setParity('N')
90              self.servo_dev.setStopbits(1)
91              self.servo_dev.open()
92
93              self.servo_dev.flushOutput()
94              self.servo_dev.flushInput()
95
96          except (serial.serialutil.SerialException), e:
97              raise RuntimeError('lib_robotis: Serial port not found!\n')
98          if(self.servo_dev == None):
99              raise RuntimeError('lib_robotis: Serial port not found!\n')
100
101
102 class Robotis():
103     ''' Superclass for the use of robotis units. Contains the basic methods
104         for I/O with robotis units
105     '''
106     def __init__(self,USB2Dynamixel,id_):
107
108         if USB2Dynamixel == None:
109             raise RuntimeError('lib_robotis: Robotis Servo requires USB2Dynamixel
                    !\n')
```

```
110         else:
111             self.dyn = USB2Dynamixel
112 
113         # ID exists on bus?
114         self.servo_id = id_
115         try:
116             self.read_address(3)
117         except:
118             raise RuntimeError('lib_robotis: Error encountered.  Could not find
                ID (%d) on bus (%s), or USB2Dynamixel 3-way switch in wrong
                position.\n' %
119                                ( id_, self.dyn.dev_name ))
120 
121         # Set Return Delay time - Used to determine when next status can be
            requested
122         data = self.read_address( 0x05, 1)
123         self.return_delay = data[0] * 2e-6
124 
125 
126     def read_model_number(self):
127         '''Reads the model number from the unit. Is used for identifying units
128         '''
129         return self.read_address(0x00).pop()
130 
131     def write_id(self, id_):
132         ''' changes the id of the robotis unit.
133             BE CARFUL TO NOT LET MORE THAN ONE UNIT ON THE BUS HAVE THE SAME ID
134         '''
135         return self.write_address( 0x03, [id_] )
136 
137     def read_id(self):
138         ''' Reads the bus id from the servo
139         '''
140         return self.read_address(0x03).pop()
141 
142     def __calc_checksum(self, msg):
143         chksum = 0
144         for m in msg:
145             chksum += m
146         chksum = ( ~chksum ) % 256
147         return chksum
148 
149     def read_address(self, address, nBytes=1):
150         ''' reads nBytes from address on the servo.
151             returns [n1,n2 ...] (list of parameters)
152         '''
153         msg = [ 0x02, address, nBytes ]
154         return self.send_instruction( msg, self.servo_id )
155 
156     def write_address(self, address, data):
157         ''' writes data at the address.
158             data = [n1,n2 ...] list of numbers.
159             return [n1,n2 ...] (list of return parameters)
160         '''
161         msg = [ 0x03, address ] + data
162         return self.send_instruction( msg, self.servo_id )
```

```
163
164     def send_instruction(self, instruction, id_):
165         msg = [ id_, len(instruction) + 1 ] + instruction # instruction includes
                the command (1 byte + parameters. length = parameters+2)
166         chksum = self.__calc_checksum( msg )
167         msg = [ 0xff, 0xff ] + msg + [chksum]
168
169         self.dyn.acq_mutex()
170         try:
171             self.send_serial( msg )
172             data, err = self.receive_reply()
173         except:
174             self.dyn.rel_mutex()
175             raise
176         self.dyn.rel_mutex()
177
178         if err != 0:
179             self.process_err( err )
180
181         return data
182
183     def process_err( self, err ):
184         raise RuntimeError('lib_robotis: An error occurred: %d\n' % err)
185
186     def receive_reply(self):
187         start = self.dyn.read_serial( 2 )
188         if start != '\xff\xff':
189             raise RuntimeError('lib_robotis: Failed to receive start bytes\n')
190         servo_id = self.dyn.read_serial( 1 )
191         if ord(servo_id) != self.servo_id:
192             raise RuntimeError('lib_robotis: Incorrect servo ID received: %d\n' %
                    ord(servo_id))
193         data_len = self.dyn.read_serial( 1 )
194         err = self.dyn.read_serial( 1 )
195         data = self.dyn.read_serial( ord(data_len) - 2 )
196         checksum = self.dyn.read_serial( 1 ) # I'm not going to check...
197         return [ord(v) for v in data], ord(err)
198
199
200     def send_serial(self, msg):
201         """ sends the command to the servo
202         """
203         out = ''
204         for m in msg:
205             out += chr(m)
206         self.dyn.send_serial( out )
207
208     def print_all_adresses(self):
209         '''Function used for debugging units. Prints all adresses
210         '''
211         print self.read_address(0x00, 2)
212         print self.read_address(0x02, 1)
213         print self.read_address(0x03, 1)
214         print self.read_address(0x04, 1)
215         print self.read_address(0x05, 1)
216         print self.read_address(0x06, 2)
```

```
217             print self.read_address(0x08, 2)
218             print self.read_address(0x0b, 1)
219             print self.read_address(0x0C, 1)
220             print self.read_address(0x0D, 1)
221             print self.read_address(0x0E, 2)
222             print self.read_address(0x10, 1)
223             print self.read_address(0x11, 1)
224             print self.read_address(0x12, 1)
225             print self.read_address(0x18, 1)
226             print self.read_address(0x19, 1)
227             print self.read_address(0x1A, 1)
228             print self.read_address(0x1B, 1)
229             print self.read_address(0x1C, 1)
230             print self.read_address(0x1D, 1)
231             print self.read_address(0x1E, 2)
232             print self.read_address(0x20, 2)
233             print self.read_address(0x22, 2)
234             print self.read_address(0x24, 2)
235             print self.read_address(0x26, 2)
236             print self.read_address(0x28, 2)
237             print self.read_address(0x2A, 1)
238             print self.read_address(0x2B, 1)
239             print self.read_address(0x2C, 1)
240             print self.read_address(0x2D, 1)
241             print self.read_address(0x2E, 1)
242             print self.read_address(0x2F, 1)
243             print self.read_address(0x30, 2)
244
245 class Robotis_Sensor(Robotis):
246     '''Class to use a robotis AX-S1 Integrated sensor unit
247     '''
248     def __init__(self,USB2Dynamixel,sensor_id):
249         Robotis.__init__(self,USB2Dynamixel, sensor_id)
250
251     def read_IR_left(self):
252         ''' Reads the value from the left IR sensor
253         '''
254         return self.read_address(0x1a).pop()
255
256     def read_IR_center(self):
257         ''' Reads the value from the center IR sensor
258         '''
259         return self.read_address(0x1b).pop()
260
261     def read_IR_right(self):
262         ''' Reads the value from the right IR sensor
263         '''
264         return self.read_address(0x1c).pop()
265
266     def read_light_left(self):
267         ''' Reads the value from the left light sensor
268         '''
269         return self.read_address(0x1d).pop()
270
271     def read_light_center(self):
272         ''' Reads the value from the center light sensor
```

```
273             '''
274             return self.read_address(0x1e).pop()
275
276         def read_light_right(self):
277             ''' Reads the value from the right light sensor
278             '''
279             return self.read_address(0x1f).pop()
280
281         def obstacle_detected(self):
282             ''' Is true if the sensor has detected a obstacle
283             '''
284             return self.read_address(0x20).pop()
285
286         def light_detected(self):
287             ''' Is true if the sensor has detected light
288             '''
289             return self.read_address(0x21).pop()
290
291         def read_sound(self):
292             ''' Reads the sound level detected by the sensor
293             '''
294             return self.read_address(0x23).pop()
295
296         def read_max_sound(self,reset=False):
297             ''' Reads the maximum sound level detected since last reset, set reset to
                    true to reset the level
298             '''
299             data = self.read_address(0x24)
300             if reset:
301                 self.write_address(0x24, [0])
302             return data.pop()
303
304         def read_sound_count(self):
305             ''' Function for detecting claps etc. Please refer to the sensor
                documentation for details.
306             '''
307             return self.read_address(0x25).pop()
308
309         def read_sound_detected_time(self):
310             ''' Function for detecting claps etc. Please refer to the sensor
                documentation for details.
311             '''
312             return self.read_address(0x26).pop()
313
314         def reset_sound_detected_count(self):
315             ''' Function for detecting claps etc. Please refer to the sensor
                documentation for details.
316             '''
317             self.write_address(0x25, [0])
318
319         def read_buzzer_note(self):
320             ''' Read what note the buzzer is playing
321             '''
322             return self.read_address(0x28).pop()
323
324         def write_buzzer_note(self,note):
```

```
325         ''' Method for telling the buzzer which note to play
326         '''
327         if note<0 or note >51:
328             raise RobotisError('The note does not exist')
329         else:
330             self.write_address(0x28, [note])
331 
332     def buzzer_ring(self,time=0.3):
333         ''' Tell the buzzer to ring for a given time. 0.3 s is default
334         '''
335         time = math.ceil(10*time)
336         if time<3:
337             raise RobotisError('You can not enter a time smaller than 0.3 s')
338         elif time>50:
339             raise RobotisError('You can not enter a time larger than 5s')
340         else:
341             self.write_address(0x29, [time])
342 
343     def buzzer_start_ringing(self,note=0):
344         ''' Tell the buzzer to ring countinously with a given note
345         '''
346         if note<0 or note >51:
347             raise RobotisError('The note does not exist')
348         else:
349             self.write_address(0x28, [note])
350             self.write_address(0x29, [254])
351 
352     def buzzer_stop_ringing(self):
353         ''' tell the buzzer to stop ringing
354         '''
355         self.write_address(0x29, [0])
356 
357     def buzzer_play_melody(self,melody=0):
358         ''' Make the buzzer play one of 26 predefined melodies
359         '''
360         if melody<0 or melody >26:
361             raise RobotisError('The melody does not exist')
362         else:
363             self.write_address(0x29, [255])
364             self.write_address(0x28, [melody])
365 
366 
367 
368 class Robotis_Servo(Robotis):
369     ''' Class to use a robotis AX 12, RX-28 or RX-64 servo.
370     '''
371     def __init__(self, USB2Dynamixel, servo_id):
372         Robotis.__init__(self,USB2Dynamixel, servo_id)
373         ''' USB2Dynamixel - USB2Dynamixel_Device object to handle serial port.
374                             Handles threadsafe operation for multiple servos
375             servo_id - servo ids connected to USB2Dynamixel 1,2,3,4 ... (1 to
                253)
376                     [0 is broadcast if memory serves]
377         '''
378 
379         self.settings = {
```

```
380             'home_encoder': 0x200,
381             'max_encoder': 0x3FF,  # Assumes 0 is min.
382             'rad_per_enc': math.radians(300.0) / 1024.0,
383             'max_ang': math.radians(148),
384             'min_ang': math.radians(-148),
385             'flipped': False,
386             'max_speed': 11.9381
387             }
388
389
390     def init_cont_turn(self):
391         '''sets CCW angle limit to zero and allows continuous turning (good for
                wheels).
392         After calling this method, simply use 'set_angvel' to command rotation.
                This
393         rotation is proportional to torque according to Robotis documentation.
394         '''
395         self.write_address(0x08, [0,0])
396
397     def kill_cont_turn(self):
398         '''resets CCW angle limits to allow commands through 'move_angle' again
399         '''
400         self.write_address(0x08, [255, 3])
401
402     def read_led(self):
403         ''' Returns the staus of the led light, 1=on, 0 = off
404         '''
405         return self.read_address(0x19).pop()
406
407     def turn_on_led(self):
408         ''' Turn the led light on
409         '''
410         self.write_address(0x19,[1])
411
412     def turn_off_led(self):
413         ''' Turn the led light off
414         '''
415         self.write_address(0x19,[0])
416
417     def set_min_pos(self,min_ang):
418         ''' sets a minimum (Clockwise) limit for the angle.
419         '''
420         enc_tics = int(round( min_ang / self.settings['rad_per_enc'] ))
421         if enc_tics > 1023 or enc_tics < 0 :
422             raise RobotisError( 'minimum angle out of bounds, lib_robotis  is
                    ignoring command')
423         elif enc_tics > self.read_address(0x08,1).pop():
424             raise RobotisError('minimum angle larger than maximum angle,
                    lib_robotis is ignoring command')
425         else:
426             self.write_address(0x06, [enc_tics]);
427             self.settings['min_ang'] = min_ang
428
429     def read_min_pos(self):
430         ''' Returns the minimum position on encoder tics
431         '''
```

```
432         return self.read_address(0x06,2).pop()
433
434     def set_max_pos(self,max_ang):
435         ''' sets a maximum (Counter-clockwise) limit for the angle.
436         '''
437         enc_tics = int(round( max_ang / self.settings['rad_per_enc'] ))
438         if enc_tics > 1023 or enc_tics < 0 :
439             raise RobotisError('maximum angle out of bounds, lib_robotis  is
                    ignoring command')
440         elif enc_tics < self.read_address(0x06).pop():
441             raise RobotisError('maximim angle is smaller than minimum angle,
                    lib_robotis is ignoring command')
442         else:
443             self.write_address(0x08, [enc_tics]);
444             self.settings['max_ang'] = max_ang
445
446     def read_max_pos(self):
447         ''' Returns the maximum position on encoder tics
448         '''
449         return self.read_address(0x08,2).pop()
450
451     def set_max_torque(self,max_torque):
452         ''' sets the maximum torque of the servo (percent)
453         '''
454         if max_torque > 100 or max_torque <0:
455             raise RobotisError('torque has to be between 0 and 100 percent')
456         else:
457             self.write_address(0x22,[int(1023.0 * (max_torque/100.0))] )
458
459     def read_max_torque(self):
460         return self.read_address(0x22,2).pop()
461
462     def is_moving(self):
463         ''' returns True if servo is moving.
464         '''
465         data = self.read_address( 0x2e, 1 )
466         return data[0] != 0
467
468     def read_voltage(self):
469         ''' returns voltage (Volts)
470         '''
471         data = self.read_address( 0x2a, 1 )
472         return data[0] / 10.
473
474     def read_temperature(self):
475         ''' returns the temperature (Celcius)
476         '''
477         data = self.read_address( 0x2b, 1 )
478         return data[0]
479
480     def read_load(self):
481         ''' number proportional to the torque applied by the servo.
482             sign etc. might vary with how the servo is mounted.
483         '''
484         data = self.read_address( 0x28, 2 )
485         load = data[0] + (data[1] >> 6) * 256
```

```
486             if data[1] >> 2 & 1 == 0:
487                 return -1.0 * load
488             else:
489                 return 1.0 * load
490 
491         def read_encoder(self):
492             ''' returns position in encoder ticks
493             '''
494             data = self.read_address( 0x24, 2 )
495             enc_val = data[0] + data[1] * 256
496             return enc_val
497 
498         def read_angle(self):
499             ''' returns the current servo angle (radians)
500             '''
501             ang = (self.read_encoder() - self.settings['home_encoder']) * self.
                    settings['rad_per_enc']
502             if self.settings['flipped']:
503                 ang = ang * -1.0
504             return ang
505 
506         def move_angle(self, ang, angvel=None, blocking=False):
507             ''' move to angle (radians) with angle velocity (rad/s)
508             '''
509             if angvel == None:
510                 angvel = self.settings['max_speed']
511 
512             if ang > self.settings['max_ang'] or ang < self.settings['min_ang']:
513                 raise RobotisError('lib_robotis.move_angle: angle out of range- %.2f
                        deg' % math.degrees(ang) + ', lib_robotis.ignoring move command.'
                        )
514                 return
515 
516             self.set_angvel(angvel)
517 
518             if self.settings['flipped']:
519                 ang = ang * -1.0
520             enc_tics = int(round( ang / self.settings['rad_per_enc'] ))
521             enc_tics += self.settings['home_encoder']
522             self.move_to_encoder( enc_tics )
523 
524             if blocking == True:
525                 while(self.is_moving()):
526                     continue
527 
528         def move_to_encoder(self, n):
529             ''' move to encoder position n
530             '''
531             # In some border cases, we can end up above/below the encoder limits.
532             #   eg. int(round(math.radians( 180 ) / ( math.radians(360) / 0xFFF ))) +
                     0x7FF => -1
533             n = min( max( n, 0 ), self.settings['max_encoder'] )
534             hi,lo = n / 256, n % 256
535             return self.write_address( 0x1E, [lo,hi] )
536 
537         def enable_torque(self):
```

```
538         ''' enable the torque setting on the servo
539         '''
540         return self.write_address(0x18, [1])
541 
542     def disable_torque(self):
543         ''' disable the torque setting on the servo
544         '''
545         return self.write_address(0x18, [0])
546 
547     def set_angvel(self, angvel):
548         ''' angvel - in rad/sec
549             maximum speed is 114rpm = 11.9381rad/s
550         '''
551         if (angvel < self.settings['max_speed']):
552             rpm = (angvel * 60.0) / (2 * math.pi)
553             angvel_enc = int(round( rpm / 0.111 ))
554             if angvel_enc<0:
555                 hi,lo = abs(angvel_enc) / 256 + 4, abs(angvel_enc) % 256
556             else:
557                 hi,lo = angvel_enc / 256, angvel_enc % 256
558         else:
559             raise RobotisError('The angle velocity can not be set higher than
                11.9381 rad/s, lib_robotis is ignoring command')
560         return self.write_address( 0x20, [lo,hi] )
561 
562 
563 
564 def find_servos(dyn,upper_limit =255,verbose=False ):
565     ''' Finds all servo IDs on the USB2Dynamixel '''
566     if upper_limit >255 or upper_limit <1:
567         upper_limit = 255
568         if verbose:
569             print 'upper search limit was set wrong, setting it to 255\n'
570 
571     if verbose:
572         print 'Scanning for Servos.\n'
573 
574     servos = []
575     dyn.servo_dev.setTimeout( 0.1 ) # To make the scan faster
576     for i in range(upper_limit):
577         try:
578             s = Robotis( dyn, i )
579             if verbose:
580                 print 'Found a unit with ID ' +str(i)
581 
582             if s.read_model_number() == 12:
583                 servos.append( i )
584                 s.turn_on_led()
585                 if verbose:
586                     print 'The unit at ID= ' + str(i) + 'is a AX-12 servo and was
                        added to the list of servo IDs'
587             elif verbose:
588                 print 'The unit at ID= ' + str(i) + 'is not a AX-12 servo'
589         except:
590             pass
591     dyn.servo_dev.setTimeout( 1.0 ) # Restore to original
```

```
592     return servos
593
594 def find_sensors(dyn,lower_limit =0,upper_limit =255,verbose=False ):
595     ''' Finds all AX_s1 sensor-IDs on the USB2Dynamixel '''
596     if upper_limit >255 or upper_limit <1:
597         upper_limit = 255
598         if verbose:
599             print 'upper search limit was set wrong, setting it to 255\n'
600
601     if lower_limit <0 or lower_limit >upper_limit:
602         lower_limit = 0
603         if verbose:
604             print 'lower search limit was set wrong, setting it to 0\n'
605
606     if verbose:
607         print 'Scanning for Sensors.\n'
608
609     sensors = []
610     dyn.servo_dev.setTimeout( 0.1 ) # To make the scan faster
611     for i in range(lower_limit, upper_limit):
612         try:
613             s = Robotis( dyn, i )
614             if verbose:
615                 print 'Found a unit with ID ' + str(i)
616             if s.read_model_number() == 13:
617                 sensors.append( i )
618                 s.turn_on_led()
619                 if verbose:
620                     print 'The unit at ID= ' + str(i) + 'is a AX-S1 sensor and
                        was added to the list of servo IDs'
621             elif verbose:
622                 print 'The unit at ID= ' + str(i) + 'is not a AX-S1 sensor'
623         except:
624             pass
625     dyn.servo_dev.setTimeout( 1.0 ) # Restore to original
626     return sensors
627
628 def find_all_units(dyn,verbose=False ):
629     ''' Finds all present IDs on the USB2Dynamixel '''
630     if verbose:
631         print 'Scanning for Servos.\n'
632
633     units = []
634     dyn.servo_dev.setTimeout( 0.1 ) # To make the scan faster
635     for i in range(255):
636         try:
637             s = Robotis( dyn, i )
638             if verbose:
639                 print 'Found a unit with ID ' + str(i)
640             units.append( i )
641         except:
642             pass
643     dyn.servo_dev.setTimeout( 1.0 ) # Restore to original
644     return units
645
646
```

```
647 def recover_servo(dyn):
648     ''' Recovers a bricked servo by booting into diagnostic bootloader and
        resetting '''
649     input('Make sure only one servo connected to USB2Dynamixel Device [ENTER]')
650     input('Disconnect power from the servo, but leave USB2Dynamixel connected to
        USB. [ENTER]')
651 
652     dyn.servo_dev.setBaudrate( 57600 )
653 
654     print 'Get Ready.  Be ready to reconnect servo power when I say \'GO!\''
655     print 'After a second, the red LED should become permanently lit.'
656     print 'After that happens, Ctrl + C to kill this program.'
657     print
658     print 'Then, you will need to use a serial terminal to issue additional
        commands.',
659     print 'Here is an example using screen as serial terminal:'
660     print
661     print 'Command Line:  screen /dev/robot/servo_left 57600'
662     print 'Type: \'h\''
663     print 'Response: Command : L(oad),G(o),S(ystem),A(pplication),R(eset),D(ump),
        C(lear)'
664     print 'Type: \'C\''
665     print 'Response:  * Clear EEPROM '
666     print 'Type: \'A\''
667     print 'Response: * Application Mode'
668     print 'Type: \'G\''
669     print 'Response:  * Go'
670     print
671     print 'Should now be able to reconnect to the servo using ID 1'
672     print
673     print
674     input('Ready to reconnect power? [ENTER]')
675     print 'GO!'
676 
677     while True:
678         s = ''
679         s.write('#')
680         time.sleep(0.0001)
681 
682 
683 if __name__ == '__main__':
684     p = optparse.OptionParser()
685     p.add_option('-d', action='store', type='string', dest='dev_name',
686                  help='Required: Device string for USB2Dynamixel. [i.e. /dev/
                     ttyUSB0 for Linux, \'0\' (for COM1) on Windows]')
687     p.add_option('--scan', action='store_true', dest='scan', default=False,
688                  help='Scan the device for servo IDs attached.')
689     p.add_option('--recover', action='store_true', dest='recover', default=False,
690                  help='Recover from a bricked servo (restores to factory defaults
                     ).')
691     p.add_option('--ang', action='store', type='float', dest='ang',
692                  help='Angle to move the servo to (degrees).')
693     p.add_option('--ang_vel', action='store', type='float', dest='ang_vel',
694                  help='angular velocity. (degrees/sec) [default = 50]', default
                     =50)
695     p.add_option('--id', action='store', type='int', dest='id',
```

```
                  help='id of servo to connect to, [default = 1]', default=1)
    p.add_option('--baud', action='store', type='int', dest='baud',
                  help='baudrate for USB2Dynamixel connection [default = 57600]',
                      default=57600)

    opt, args = p.parse_args()

    if opt.dev_name == None:
        p.print_help()
        sys.exit(0)

    dyn = USB2Dynamixel_Device(opt.dev_name, opt.baud)

    if opt.scan:
        find_servos( dyn )

    if opt.recover:
        recover_servo( dyn )

    if opt.ang != None:
        servo = Robotis_Servo( dyn, opt.id )
        servo.move_angle( math.radians(opt.ang), math.radians(opt.ang_vel) )

class RobotisError(Exception):
    def __init__(self, error):
        self.error = error
    def __str__(self):
        return repr(self.error)
```

I.3 MessageParser.py

```
'''
Message creation and parsing utilities
'''
START_OF_MESSAGE = '{'
END_OF_MESSAGE = '}'
SEPARATOR = ','

def parse_message(message):
    tokens = tokenize_message(message)
    command = tokens[0]
    parameter_list = tokens[1:]
    msg = parsed_message(command, parameter_list)
    return msg

def tokenize_message(message):
    message = str(message)
    if not (message.startswith(START_OF_MESSAGE) and message.endswith(
        END_OF_MESSAGE)):
        print "Message format error: "+message
        return None

    message = message.lstrip(START_OF_MESSAGE)
    message = message.rstrip(END_OF_MESSAGE)
    tokens = message.split(SEPARATOR)
    return tokens
```

```
25
26 def create_message(msg):
27     command = msg.command
28     parameter_string = parameters_to_string(msg.parameter_list)
29
30     message = START_OF_MESSAGE+command+","+parameter_string+END_OF_MESSAGE
31     return message
32
33 def parameters_to_string(parameter_list):
34     stri = ""
35     for parameter in parameter_list:
36         stri = stri+str(parameter)+SEPARATOR
37
38     stri.lstrip()
39     stri = stri.rstrip(SEPARATOR)
40     return stri
41
42 class parsed_message:
43     def __init__(self, command, parameter_list):
44         self.command = command
45         self.parameter_list = parameter_list
46
47 '''
48 SRV_CMD, CMD_ID, SERVO_ID, VALUE
49 SRV_REG, SERVO_IDs
50 SRV_VALUE, VALUE_ID, VALUE
51
52 SNS_VALUE, VALUE_ID, VALUE
53 SNS_REG, SENSOR_IDs
54 SNS_CMD, CMD_ID, SENSOR_ID, VALUE
55 ERROR, MESSAGE
56 '''
```

I.4 protocol.py

```
1 '''
2 SRV_CMD, CMD_ID, SERVO_ID, VALUE
3 SRV_REG, SERVO_IDs
4 SRV_VALUE, VALUE_ID, VALUE
5 ERROR, MESSAGE
6
7 SNS_VALUE, VALUE_ID, VALUE
8 SNS_REG, SENSOR_IDs
9 SNS_CMD, CMD_ID, SENSOR_ID, VALUE
10
11 '''
12
13
14 def servo_command(self, servo_id, value_id, value):
15     pass
```

I.5 RobotNetworkService.py

```
1 '''
2 Network layer for the robot side
3
4 '''
```

```
5  import socket
6  import time
7  import threading
8  import MessageParser
9  import urllib
10 import string
11 import thread
12 from symbol import parameters
13 from MessageParser import parsed_message
14
15 #CONNECTION STATE:
16 SEARCHING = 1
17 CONNECTED = 2
18
19 #CONNECTION PARAMETERS
20 BUFFER_SIZE = 1024
21 RETRY_TIME = 0.5
22 RETRIES = 15
23
24 class robot_network_service:
25
26     def __init__(self, robot_arm):
27         self.controller_host = socket.gethostbyname("eit.no-ip.biz")
28         self.robot_arm = robot_arm
29         self.connection_state = SEARCHING
30         self.out_port = 10002
31         self.buffer_size = 1024
32         self.incoming = robot_incoming_socket(self)
33         self.outgoing = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
34         self.incoming.start()
35         self.outgoing = robot_outgoing_socket(self)
36
37
38     def run(self):
39         self.incoming.start()
40
41     def recive_message(self, message):
42         # TODO: Change to Command pattern?
43         msg = MessageParser.parse_message(message)
44         print msg.command
45         print msg.parameter_list
46
47         if (msg.command == "SRV_CMD"):
48             self.execute_srv_cmd(msg)
49         elif (msg.command == "SGET"):
50             self.execute_sget(msg)
51         elif (msg.command == "WIGL"):
52             self.robot_arm.wiggleServo(msg.parameter_list[0])
53         elif (msg.command == "SERVO"):
54             self.report_servos()
55         elif (msg.command == "MGSET"):
56             pass
57
58     def execute_srv_cmd(self, msg):
59         cmd_id = msg.parameter_list[0]
60         srv_id = int(float(msg.parameter_list[1]))
```

```
61          value = msg.parameter_list[2]
62          if (cmd_id == "SET_ANGL"):
63              self.robot_arm.set_angle(float(value), int(srv_id))
64          elif (cmd_id == "SET_AVEL"):
65              self.robot_arm.set_velocity(float(value), srv_id)
66  #           self.robot_arm.set_mode(1, srv_id)
67          elif (cmd_id == "SET_MAXP"):
68              self.robot_arm.set_max_pos(float(value), srv_id)
69          elif (cmd_id == "SET_MINP"):
70              self.robot_arm.set_min_pos(float(value), srv_id)
71          elif (cmd_id == "SET_MODE"):
72              self.robot_arm.set_mode(float(value), srv_id)
73          elif (cmd_id == "SET_MODE"):
74              self.robot_arm.set_mode(float(value), srv_id)
75          elif (cmd_id == "WGL"):
76              self.robot_arm.wiggle_servo(srv_id)
77          elif (cmd_id == "AUTO"):
78              self.robot_arm.set_auto(int(value))
79
80      def send_servo_list(self):
81          self.report_servos()
82          servo_list = self.robot_arm.getServos()
83          parameter_list = [str(servo.read_id()) for servo in servo_list]
84          msg = MessageParser.parsed_message("SRV_REG", parameter_list)
85          self.send_message(MessageParser.create_message(msg))
86
87      def send_sensor_list(self):
88          #LAST ID NOT LIST
89          sns_ids = self.robot_arm.get_sensor_len()
90          msg = MessageParser.parsed_message("SNS_REG", [str(sns_ids)])
91          message = MessageParser.create_message(msg)
92          self.outgoing.send_message(message)
93
94      def report_servos(self):
95          last_id = self.robot_arm.get_servo_len()-1
96          msg = MessageParser.parsed_message("REG", [str(last_id)])
97          message = MessageParser.create_message(msg)
98          self.outgoing.send_message(message)
99
100
101     def execute_sset(self, msg):
102         print msg.parameter_list
103         value_id = msg.parameter_list[0]
104         servo_id = int(msg.parameter_list[1])
105
106         if (value_id == "SET_ANGL"):
107             self.robot_arm.set_angle(float(servo_id, msg.parameter_list[2]))
108         elif (value_id == "AVEL"):
109             self.robot_arm.setVelocity(float(msg.parameter_list[2]), servo_id )
110         elif (value_id == "MODE"):
111             continuous = (msg.parameter_list[2] == "True")
112             self.robot_arm.setMode(continuous, servo_id)
113
114     def execute_sget(self, msg):
115         value_id = msg.parameter_list[0]
116         servo_id = int(msg.parameter_list[1])
```

```
117             para_list = [value_id, servo_id]
118             reply = MessageParser.parsed_message("SRV_VALUE", para_list)
119 
120             if (value_id == "ANGL"):
121                 reply.parameter_list.append(str(self.robot_arm.getAngle()))
122             elif (value_id == "LOAD"):
123                 reply.parameter_list.append(str(self.robot_arm.getLoad()))
124             msg = MessageParser.parse_message(reply)
125             self.send_message(msg)
126 
127         def send_load(self, id, load):
128             p_msg = MessageParser.parsed_message("SRV_VALUE", ["LOAD", str(id), str(
                    load)])
129             message = MessageParser.create_message(p_msg)
130             self.outgoing.send_message(message)
131 
132         def send_angle(self, id, angle):
133             p_msg = MessageParser.parsed_message("SRV_VALUE", ["ANGL", str(id), str(
                    angle)])
134             message = MessageParser.create_message(p_msg)
135             self.outgoing.send_message(message)
136 
137         def send_temp(self,id, temp):
138             p_msg = MessageParser.parsed_message("SRV_VALUE", ["TEMP", str(id), str(
                    temp)])
139             message = MessageParser.create_message(p_msg)
140             self.outgoing.send_message(message)
141 
142         def send_voltage(self, id, volt):
143             p_msg = MessageParser.parsed_message("SRV_VALUE", ["VOLT", str(id), str(
                    volt)])
144             message = MessageParser.create_message(p_msg)
145             self.outgoing.send_message(message)
146 
147         def send_min_pos(self, id, min_pos):
148             p_msg = MessageParser.parsed_message("SRV_VALUE", ["MINP", str(id), str(
                    min_pos)])
149             message = MessageParser.create_message(p_msg)
150             self.outgoing.send_message(message)
151 
152         def send_max_pos(self, id, max_pos):
153             p_msg = MessageParser.parsed_message("SRV_VALUE", ["MAXP", str(id), str(
                    max_pos)])
154             message = MessageParser.create_message(p_msg)
155             self.outgoing.send_message(message)
156 
157         def send_error(self, error_message):
158             p_msg = MessageParser.parsed_message("ERROR", [error_message])
159             message = MessageParser.create_message(p_msg)
160             self.outgoing.send_message(message)
161 
162         def send_sensor_data(self, value_id, sensor_id, value):
163             p_msg = MessageParser.parsed_message("SNS_VALUE", [value_id, sensor_id,
                    str(value)])
164             message = MessageParser.create_message(p_msg)
165             self.outgoing.send_message(message)
```

```
166
167     def send_IR_left(self, sensor_id, value):
168         self.send_sensor_data("IRL", sensor_id, value)
169
170     def send_IR_right(self, sensor_id, value):
171         self.send_sensor_data("IRR", sensor_id, value)
172
173     def send_IR_center(self, sensor_id, value):
174         self.send_sensor_data("IRC", sensor_id, value)
175
176     def send_light_left(self, sensor_id, value):
177         self.send_sensor_data("LL", sensor_id, value)
178
179     def send_light_Right(self, sensor_id, value):
180         self.send_sensor_data("LR", sensor_id, value)
181
182     def send_light_center(self, sensor_id, value):
183         self.send_sensor_data("LC", sensor_id, value)
184
185     def play_sound_for_sensor(self, sensor_id, value):
186         self.robot_arm.buzzer_play_melody(self, sensor_id, value)
187
188 #    def send_message(self, message):
189 #        self.outgoing.connect((self.controller_host, self.out_port))
190 #        self.outgoing.sendall(message)
191 #        self.outgoing.close()
192
193
194
195
196 class robot_incoming_socket(threading.Thread):
197     # Encapsulated socket for incoming messages. Runs in separate thread and
        passes
198     # recived messages back to network service trough network_layer.
        recive_message(message)
199     def __init__(self, network_layer):
200         self.network_layer = network_layer # "Callback" reference
201         super(robot_incoming_socket, self).__init__()
202         self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
203         self.host = urllib.urlopen('http://icanhazip.com').read()
204         self.host = self.host.rstrip()
205         self.in_port = 10001
206         self.running = True
207
208     def run(self):
209         self.server_socket.bind((self.host, self.in_port))
210         self.server_socket.listen(5)
211         while self.running:
212             client, address = self.server_socket.accept()
213             self.network_layer.recive_message(self.rcv_message(client))
214
215     def rcv_message(self, client_socket):
216         # Waits for end of message symbol and returns complete message
217         message=[]
218         data=''
219         while True:
```

```
220                 data=client_socket.recv(1024)
221                 if MessageParser.END_OF_MESSAGE in data:
222                     message.append(data[data.find(MessageParser.START_OF_MESSAGE)
                            :data.find(MessageParser.END_OF_MESSAGE)+1])
223                     break
224                 message.append(data)
225         return ''.join(message)
226 
227     def shutdown(self):
228         self.running = False
229         time.sleep(0.5)
230         self.server_socket.close()
231 
232 class robot_outgoing_socket(threading.Thread):
233 
234     def __init__(self, network_layer):
235         self.controller_host = socket.gethostbyname("eit.no-ip.biz")
236         self.net = network_layer
237         self.out_port = 10002
238         self.buffer_size = BUFFER_SIZE
239         self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
240         self.connected = False
241         while(not self.connected):
242             self.connected = True
243             try:
244                 self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
245                 self.socket.connect((self.controller_host, self.out_port))
246                 self.socket.sendall(self.get_reg_msg())
247                 self.socket.close()
248 
249             except socket.error as msg:
250                 self.connected = False
251 #                print msg
252                 time.sleep(1)
253                 print "Connection failed: retry"
254 
255 
256     def connect_to_server(self):
257         print ("Searching for server")
258         while (not self.is_connected()):
259             time.sleep(1.0)
260 
261 
262 
263     def get_reg_msg(self):
264         last_id = self.net.robot_arm.get_servo_len()-1
265         msg = MessageParser.parsed_message("REG", [str(last_id)])
266         message = MessageParser.create_message(msg)
267         return message
268 
269     def send_message(self, message):
270 
271         try:
272             self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
273             self.socket.connect((self.controller_host, self.out_port))
274             self.socket.sendall(message)
```

```
275             self.socket.close()
276             done = True
277         except socket.error as msg:
278             pass
279 
280 #         done = False
281 #         attempts = 0
282 #         while (not done):
283 #             attempts += 1
284 #             if (attempts >= RETRIES):
285 #                 print ("Failed sending message after 15 attempts.")
286 #                 return
287 #             try:
288 #                 self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
289 #                 self.socket.connect((self.controller_host, self.out_port))
290 #                 self.socket.sendall(message)
291 #                 self.socket.close()
292 #                 done = True
293 #             except socket.error as msg:
294 #                 raise
295 
296 
297 #     def connect(self):
298 #         print("Searching for server...")
299 #         connected = False
300 #         while (not connected):
301 #             try:
302 #                 self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
303 #                 self.socket.connect((self.controller_host, self.out_port))
304 #                 self.socket.close()
305 #                 print("Connected")
306 #                 connected = True
307 #             except socket.error as msg:
308 #                 self.net.connection_status = SEARCHING
309 #             time.sleep(1.0)
310 
311     def is_connected(self):
312         self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
313         try:
314             self.socket.connect((self.controller_host, self.out_port))
315             self.socket.close()
316             return True
317         except:
318             return False
319 
320 
321 #     def connect(self):
322 #         socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
323 #         if (self.net.connection_state == CONNECTED):
324 #             try:
325 #                 self.socket.connect((self.controller_host, self.out_port))
326 #                 return socket
327 #             except:
328 #                 self.net.connection_state == SEARCHING
329 #
330 #         while (self.net.connection_state == SEARCHING):
```

```
331 #             try:
332 #                 self.socket.connect((self.controller_host, self.out_port))
333 #                 self.net.connection_state == CONNECTED
334 #                 return socket
335 #             except socket.error as msg:
336 #                 "Failed connection attempt. Retrying in "+str(RETRY_TIME)+"
        seconds"
337 #                 time.sleep(RETRY_TIME)
338 #
339
340
341 # TEST CODE
342 #test = robot_network_service()
343 #test.send_load(1, 129.45)
344 #msg = MessageParser.parsed_message("SET", ["ANGL", "1", "45"])
345 #s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
346 #s.connect((socket.gethostbyname("eit.no-ip.biz"), 10001))
347 #s.sendall("{1,OK,0,0,0}")
348 # END OF TEST CODE
```

I.6 autonomus_control.py

```
1  #### Here you can implement your own autonomous behavior.
2  #### A very simple algorithm is already implemented.
3
4  import threading
5
6  class autonomous_controll(threading.Thread):
7      def __init__(self, lft_frt_wheel, lft_bck_wheel, rgt_frt_wheel, rgt_bck_wheel
           , sensor):
8          super(autonomous_controll, self).__init__()
9          self.wheels = []
10         self.wheels.append(lft_frt_wheel)
11         self.wheels.append(lft_bck_wheel)
12         self.wheels.append(rgt_frt_wheel)
13         self.wheels.append(rgt_bck_wheel)
14         self.sensor = sensor
15         self.is_auto = 0
16         self.vel = 4
17         self.stopped = True
18         self.start()
19
20     def run(self):
21         '''If nothing is blocking the robot will drive forward. If something is
               blocking in
22         the front, but not to the left, the robot turns left. If something is
               blocking in
23         the front and to the left, the robot will turn right. If all sides are
               blocked it
24         stoppes.'''
25         while(1):
26             if(self.is_auto == 1):
27                 self.stopped = False
28                 if(self.sensor.get_IR_center() < 200):
29                     self.forward()
30                 elif(self.sensor.get_IR_left() < 150):
```

```
                    self.left()
                elif(self.sensor.get_IR_right() < 150):
                    self.right()
                else:
                    self.stop()
            else:
                if(not self.stopped):
                    self.stop()
                    self.stopped =True


    def forward(self):
        for i in range(4):
            if(i <2):
                self.wheels[i].set_velocity(self.vel)
            else:
                self.wheels[i].set_velocity(-self.vel)

    def left(self):
        for i in range(4):
            if(i <2):
                self.wheels[i].set_velocity(self.vel)
            else:
                self.wheels[i].set_velocity(0)

    def right(self):
        for i in range(4):
            if(i <2):
                self.wheels[i].set_velocity(0)
            else:
                self.wheels[i].set_velocity(-self.vel)

    def stop(self):
        for i in range(4):
            self.wheels[i].set_velocity(0)

    def set_auto(self, set_auto):
        self.is_auto = set_auto

    def set_velocity(self, vel):
        self.vel = vel
```

I.7 et/angleSensor.py

```
'''
Created on 3. mars 2013
@author: Eivind, Erik, Eirik
'''
from __future__ import division
from pymcu import mcuModule #@UnusedImport
import sensor
from sensor import Sensor
from debug import Debug


```

```
13 # SENSOR bends max 90 degrees (safety)
14 # SERVO goes 300 degrees in angle mode
15 # rawAngle varies from 540 to 790 ~
16 # Sensor ID goes from 1 to 6 (physically on the board)
17 RAW_VALUE_MIN = 540
18 RAW_VALUE_MAX = 790
19 RAW_VALUE_90 = 690
20 RAW_VALUE_SPAN = RAW_VALUE_MAX - RAW_VALUE_MIN # 250
21 RAW_TO_REAL_FACTOR_90 = 90 / (RAW_VALUE_90 - RAW_VALUE_MIN)
22 RAW_TO_REAL_FACTOR = RAW_TO_REAL_FACTOR_90
23
24
25
26 class BendSensor(Sensor):
27
28     # pyMCU Controller
29     __mcu = None
30     __raw_value_min = RAW_VALUE_MIN
31     __raw_to_real_factor = RAW_TO_REAL_FACTOR
32
33
34     # Constructor
35     def __init__(self, sensorID, mcu, \
36                  amplifier = sensor.DEFAULT_AMPLIFIER, \
37                  threshold = sensor.DEFAULT_THRESHOLD):
38         super(BendSensor, self).__init__(sensorID, amplifier, threshold)
39         self.__mcu = mcu
40
41     # Property: raw_value_min
42     def raw_value_min(): #@NoSelf
43         def fget(self): return self.__raw_value_min
44         def fset(self, value): self.__raw_value_min = value
45         return locals()
46     raw_value_min = property(**raw_value_min())
47
48     # Property: raw_to_real_factor
49     def raw_to_real_factor(): #@NoSelf
50         def fget(self): return self.__raw_to_real_factor
51         def fset(self, value): self.__raw_to_real_factor = value
52         return locals()
53     raw_to_real_factor = property(**raw_to_real_factor())
54
55     # Calibration input
56     def calibrate(self,amplifier,threshold,raw_value_min,raw_value_90):
57         self.amplifier = amplifier
58         self.threshold = threshold
59         self.raw_to_real_factor = 90 / (raw_value_90 - raw_value_min)
60
61
62
63     #-------------------------------------------------------------------------
64     #    ABSTRACT METHODS - Overridden from superclass
65     #-------------------------------------------------------------------------
66
67
68     # Determines how raw values are acquired
```

```
69      def _abstractReadRawValue(self):
70          return self.__tryGetRawValue()
71
72
73      # Determines how raw value is converted to real value
74      def _abstractRawToReal(self, rawValue):
75
76          # Finds zero-based raw angle (from 0 to N)
77          # Converts the raw angle to real angle
78          '''
79          rawAngle_0 = max(0, rawValue - self.raw_value_min)
80          '''
81          rawAngle_0 = rawValue - self.raw_value_min
82          return rawAngle_0 * self.raw_to_real_factor
83
84
85
86
87
88      #--------------------------------------------------------------------------
89      #    PRIVATE
90      #--------------------------------------------------------------------------
91
92
93      # Tries to read MCU raw value
94      def __tryGetRawValue(self):
95          if self.__mcu <> None:
96              rawValue = self.__mcu.analogRead(self.ID)
97              if rawValue <> None:
98                  return rawValue
99              else:
100                 Debug.SensorMCUFail(self.ID)
101                 return 0
102         else:
103             Debug.SensorMCUFail(self.ID)
104             return 0
```

I.8 et/controller.py

```
1   '''
2   Created on 27. feb. 2013
3   @author: Eivind, Erik
4   '''
5   from __future__ import division
6   from guiWrapper import GUIWrapper
7   from sensorWatcher import SensorWatcher
8   from debug import * #@UnusedWildImport
9   from linker import Linker
10  from wheels import Wheels
11  from ControllerNetworkLayer import ControllerNetworkService
12  '''
13  from servo import ServoGroup
14  '''
15
16  class Controller(object):
17
```

```
18      '''
19      # ServoGroup
20      __servos = None
21      '''
22
23      # Network
24      __net = None
25      __netEnabled = True
26
27      # Main GUI
28      __gui = None
29
30      # SensorWatcher thread
31      __sensorWatcher = None
32
33      # Sensor-SERVO linker
34      __linker = None
35
36      # SERVO-Wheels Controller
37      __wheels = None
38
39
40      # Constructor
41      def __init__(self, enableNet = True):
42          self.__createServoGroup()
43          self.__createGUI()
44          self.__createSensorWatcher()
45          self.__createLinker()
46          self.__createWheels()
47          self.__createNetwork(enableNet)
48          Debug.setConsole(self)
49
50
51      # Property: Links
52      def getLinker(self): return self.__linker
53      Linker = property(getLinker)
54
55
56      # Property: Wheels
57      def getWheels(self): return self.__wheels
58      Wheels = property(getWheels)
59
60
61      # Start the controller
62      def launch(self):
63          self.__gui.show()
64          self.__sensorWatcher.start()
65
66
67      # Stop the controller
68      def stop(self):
69          self.__sensorWatcher.stop()
70
71
72      # Checks if the controller is running with network
73      def hasNet(self):
```

```
74          return self.__netEnabled and self.__net <> None
75
76
77      # Checks if the controller is running with network
78      # Additional condition: robot must also be found
79      def hasNetAndRobot(self):
80          return self.hasNet() and self.__net.robot_found()
81
82
83      # Adds a new BendSensor to watch
84      def addBendSensor(self, sensorID):
85          self.__sensorWatcher.addBendSensor(sensorID)
86          self.__gui.addSensor(sensorID)
87
88
89      # Adds a new BendSensor to watch (with turn sensor calibration)
90      def addTurnSensor(self, sensorID):
91          self.__sensorWatcher.addTurnSensor(sensorID)
92          self.__gui.addSensor(sensorID)
93
94
95      # Gets the SENSOR with given ID
96      # Note: returns None if ID doesn't exist
97      def tryGetSensor(self, sensorID):
98          return self.__sensorWatcher.tryGetSensor(sensorID)
99
100
101     # EVENT - SENSOR Data Changed
102     # Called from SensorWatcher
103     def sensorDataChanged(self, sensor):
104
105         # Get all SERVOS linked to the sensor
106         servoIDs = self.Linker.getServoLinks(sensor.ID)
107
108         # Network
109         if self.hasNetAndRobot():
110             for servoID in servoIDs:
111                 self.__net.set_angle_for_servo(sensor.realValue, servoID)
112
113         # GUI
114         self.__gui.updateSensor(sensor)
115
116
117     # EVENT - SERVO Data Changed
118     # Called from Network
119     def servoDataChanged(self, servoID, index, value):
120
121         # GUI
122         self.__gui.updateServo(servoID, index, value)
123
124
125
126     # Register SERVO count
127     def registerServos(self, servoCount):
128
129         # GUI (update SERVO list count)
```

```
130         self.__gui.registerServos(servoCount)
131
132
133     def clearServos(self):
134         self.Links.clear()
135         print "TODO: Clear GUI and other lists of SERVOS <Controller.clearServos>
                "
136         #
137         #     TODO: Incoming from Network, clear GUI and any other lists
138         #
139
140
141     # Write to GUI Console
142     def writeToConsole(self, text):
143         self.__gui.writeToConsole(text)
144
145
146     # Set SERVO Wheel speeds over Network
147     def setWheelServoSpeed(self, servoID_1, servoID_2, angleSpeed_1, angleSpeed_2
            ):
148
149         # Network
150         if self.hasNetAndRobot():
151             leftServos = self.__linker.getServoGroup(servoID_1)
152             rightServos = self.__linker.getServoGroup(servoID_2)
153             for servo_id in leftServos: self.__net.set_angle_velocity_for_servo(
                    angleSpeed_1, servo_id)
154             for servo_id in rightServos: self.__net.set_angle_velocity_for_servo(
                    angleSpeed_2, servo_id)
155
156
157
158     # Controls a SERVO angle via the network
159     def _controlServoAngle(self, ID, angle):
160
161         # Network
162         if self.hasNetAndRobot():
163             for servoID in self.__linker.getServoGroup(ID):
164                 self.__net.set_angle_for_servo(angle, servoID)
165
166
167     # Autonomous mode
168     def toggleAutonomous(self, enabled):
169
170         # Network
171         if self.hasNetAndRobot(): self.__net.set_auto(enabled)
172
173
174     # SERVO mode
175     def setServoMode(self, servoID, mode):
176
177         # Network
178         if self.hasNetAndRobot() and servoID > -1: self.__net.set_mode_for_servo(
                servoID, mode)
179
180
```

```
181     # WIGGLE SERVO
182     def wiggleServo(self, servoID):
183 
184         # Network
185         if self.hasNetAndRobot(): self.__net.wiggle_servo(servoID)
186 
187 
188 
189     #--------------------------------------------------------------------------
190     #    PRIVATE
191     #--------------------------------------------------------------------------
192 
193 
194     # Create ServoGroup
195     def __createServoGroup(self):
196         pass
197         '''
198         self.__servos = ServoGroup()
199         '''
200 
201 
202     # Create main GUI window
203     def __createGUI(self):
204         self.__gui = GUIWrapper(self)
205 
206 
207     # Create the SensorWatcher
208     def __createSensorWatcher(self):
209         self.__sensorWatcher = SensorWatcher(self)
210 
211 
212     # Create the Linker
213     def __createLinker(self):
214         self.__linker = Linker()
215 
216 
217     # Create the Network object
218     def __createNetwork(self, enableNet):
219         if enableNet:
220             self.__net = ControllerNetworkService(self)
221 
222     # Create the Wheel Controller
223     def __createWheels(self):
224         self.__wheels = Wheels(self)
```

I.9 et/controllerMain.py

```
1 import sys
2 from PyQt4 import QtGui
3 from controller import Controller
4 
5 NET_ENABLED = False
6 
7 if __name__ == '__main__':
8 
9     # Create QtGUI Application
```

```
    app = QtGui.QApplication(sys.argv)

    # Net enabled check
    if not NET_ENABLED: print "WARNING: Network disabled (see NET_ENABLED in
        controllerMain)"

    # Create controller
    controller = Controller(NET_ENABLED)

    # TEMPORARY: manually assign wheel SERVOS
    #controller.Wheels.registerServos(3, 4)

    # TEMPORARY: manually added sensors (1 is main)
    #print "NOTE: Enable/disable sensors in controllerMain.py!"
    controller.addBendSensor(1)
    controller.addBendSensor(2)
    #controller.addBendSensor(3) # dummy
    controller.registerServos(7)
    #
    #    TODO: terminate network threads from this main?
    #

    # Launch controller and application
    controller.launch()
    result = app.exec_()

    # Exit application
    controller.stop()
    sys.exit(result)
```

I.10 et/ControllerNetworkLayer.py

```
'''
Created on 20. feb. 2013

'''
import socket
import time
import MessageParser
import threading
from MessageParser import parsed_message
from EServo import EServo

#Connection states
NOT_CONNECTED = 1
SHAKING = 2
CONNECTED = 3


class ControllerNetworkService:

    def __init__(self, controller):
        self.controller = controller
        self.connection_state = NOT_CONNECTED
        self.outgoing = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.incoming = controller_incoming_socket(self)
```

```
25          self.out_port = 10001
26          self.host = '' #socket.gethostbyname("eit.no-ip.biz")
27          self.robot_ip = ""
28          self.incoming.start()
29
30      def run(self):
31          self.incoming.start()
32
33      def reset(self):
34          self.controller.clearServos()
35
36      def robot_found(self):
37          return (self.connection_state == 3)
38
39      def send_message(self, message):
40          if (self.robot_ip == ""):
41              return
42          self.outgoing = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
43          self.outgoing.connect((self.robot_ip, self.out_port))
44          self.outgoing.sendall(message)
45          self.outgoing.close()
46
47      def set_robot_ip(self, ip):
48          self.robot_ip = ip
49
50      def recive_message(self, msg):
51  #         print "Recieved: " +msg
52          message = MessageParser.parse_message(msg)
53          if  (message.command == "SRV_VALUE"):
54              self.rcv_value(message)
55          if (message.command == "REG"):
56              self.connection_state = CONNECTED
57  #             self.reset()
58              count = int(message.parameter_list[0])
59              self.controller.registerServos(count)
60
61      def rcv_value(self, message):
62          type = message.command
63
64          if (type == "SRV_REG"):
65              print ("recived servo list")
66          elif (type == "SNS_REG"):
67              print ("recived sensor list")
68          elif (type == "SRV_VALUE"):
69              value_type = value_type_to_enum(message.parameter_list[0])
70              id = int(message.parameter_list[1])
71              value = message.parameter_list[2]
72              self.controller.servoDataChanged(id, value_type, value)
73          elif (type == "SNS_VALUE"):
74              print ("Recived sensor data")
75          elif (type == "ERROR"):
76              print ("ROBOT ERROR: "+message.parameter_list[0])
77
78  # Private command generation methods
79
80      def servo_command(self, cmd_id, servo_id, value):
```

```
81          command = "SRV_CMD"
82          parameters = [cmd_id, servo_id, value]
83          msg = parsed_message(command, parameters)
84          self.send_message(MessageParser.create_message(msg))
85
86      def sensor_command(self, cmd_id, sensor_id, value):
87          command = "SNS_CMD"
88          parameters = [cmd_id, sensor_id, value]
89          msg = parsed_message(command, parameters)
90          self.send_message(MessageParser.create_message(msg))
91
92  # PUBLIC CONTROLLER NETWORK INTERFACE
93
94  # Servo grouping methods
95
96      def servo_group(self, servo_ids):
97          command = "SRV_GROUP"
98          parameters = servo_ids
99          msg = parsed_message(command, parameters)
100         self.send_message(MessageParser.create_message(msg))
101
102     def servo_ungroup(self, servo_ids):
103         command = "SRV_UNGROUP"
104         parameters = servo_ids
105         msg = parsed_message(command, parameters)
106         self.send_message(MessageParser.create_message(msg))
107
108 # Methods for requesting ids of components
109
110     def get_servos(self):
111         command = "SRV_GET"
112         parameters = []
113         msg = parsed_message(command, parameters)
114         self.send_message(MessageParser.create_message(msg))
115
116     def get_sensors(self):
117         command = "SNS_GET"
118         parameters = []
119         msg = parsed_message(command, parameters)
120         self.send_message(MessageParser.create_message(msg))
121
122 # Servo commands
123
124     def set_angle_for_servo(self, angl, servo_id):
125         self.servo_command("SET_ANGL", str(servo_id), str(angl))
126
127     def set_angle_velocity_for_servo(self, angl_velo, servo_id):
128         self.servo_command("SET_AVEL", servo_id, angl_velo)
129
130     def set_max_pos_for_servo(self, servo_id, max_pos):
131         self.servo_command("SET_MAXP", max_pos)
132
133     def set_min_pos_for_servo(self, servo_id, min_pos):
134         self.servo_command("SET_MINP", min_pos)
135
136     def set_mode_for_servo(self, servo_id, mode):
```

```
137         self.servo_command("SET_MODE", servo_id, mode)
138
139     def set_auto(self, mode):
140         self.servo_command("AUTO", 0, mode)
141
142     def wiggle_servo(self, servo_id):
143         print ("send command")
144         self.servo_command("WGL", servo_id, 0)
145
146     def request_angle_from_servo(self, servo_id):
147         self.servo_command("GET_ANGL", servo_id, 0)
148
149     def request_load_from_servo(self, servo_id):
150         self.servo_command("GET_ANGL", servo_id, 0)
151
152     def request_volt_from_servo(self, servo_id):
153         self.servo_command("GET_VOLT", servo_id, 0)
154
155     def request_move_from_servo(self, servo_id):
156         self.servo_command("GET_MOVE", servo_id, 0)
157
158     def request_max_pos_from_servo(self, servo_id):
159         self.servo_command("GET_MAXP", servo_id, 0)
160
161     def request_min_pos_from_servo(self, servo_id):
162         self.servo_command("GET_MINP", servo_id, 0)
163
164     def request_max_torque_from_servo(self, servo_id):
165         self.servo_command("GET_MAXT", servo_id, 0)
166
167     def request_temp_from_servo(self, servo_id):
168         self.servo_command("GET_TEMP", servo_id, 0)
169
170
171 # Sensor commands
172
173     def request_IR_left_from_sensor(self, sensor_id):
174         self.sensor_command("GET_IRL", sensor_id, 0)
175
176     def request_IR_right_from_sensor(self, sensor_id):
177         self.sensor_command("GET_IRR", sensor_id, 0)
178
179     def request_IR_center_from_sensor(self, sensor_id):
180         self.sensor_command("GET_IRC", sensor_id, 0)
181
182     def request_light_left_from_sensor(self, sensor_id):
183         self.sensor_command("GET_LL", sensor_id, 0)
184
185     def request_light_right_from_sensor(self, sensor_id):
186         self.sensor_command("GET_LR", sensor_id, 0)
187
188     def request_light_center_from_sensor(self, sensor_id):
189         self.sensor_command("GET_LC", sensor_id, 0)
190
191     def play_sound_for_sensor(self, sensor_id, value):
192         self.sensor_command("SET_SOUND", sensor_id, value)
```

```
193
194
195 class controller_incoming_socket(threading.Thread):
196
197     def __init__(self, network_layer):
198         self.network_layer = network_layer
199         super(controller_incoming_socket, self).__init__()
200         self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
201         self.host = '' #socket.gethostbyname("eit.no-ip.biz")
202         self.in_port = 10002
203         self.running = True
204
205     def run(self):
206         print self.host
207         self.server_socket.bind((self.host, self.in_port))
208         self.server_socket.listen(10)
209         while self.running:
210             client, address = self.server_socket.accept()
211             ip, port = client.getpeername()
212             self.network_layer.set_robot_ip(ip)
213             self.network_layer.recive_message(self.rcv_message(client))
214
215     def rcv_message(self, client_socket):
216         message=[]
217         data=''
218         while True:
219                 data=client_socket.recv(1024)
220                 if MessageParser.END_OF_MESSAGE in data:
221                     message.append(data[data.find(MessageParser.START_OF_MESSAGE)
                            :data.find(MessageParser.END_OF_MESSAGE)+1])
222                     break
223                 message.append(data)
224         return ''.join(message)
225
226     def shutdown(self):
227         self.running = False
228         time.sleep(0.5)
229         self.server_socket.close()
230
231
232 def value_type_to_enum(value_type):
233     Img = 0
234     Name = 1
235     Angle = 2
236     Load = 3
237     Temp = 4
238     Volt = 5
239     MinPos = 6
240     MaxPos = 7
241     Status = 8
242     if (value_type == "ANGL"):
243         return Angle
244     if (value_type == "TEMP"):
245         return Temp
246     if (value_type == "LOAD"):
247         return Load
```

```
248     if (value_type == "MINP"):
249         return MinPos
250     if (value_type == "MAXP"):
251         return MaxPos
252     if (value_type == "VOLT"):
253         return Volt
254 
255 ## TEST CODE
256 #msg = MessageParser.parse_message("{S_CMD, SET, ANGL, 0, 0.0}")
257 #print msg.parameter_list
258 #test = ControllerNetworkService()
259 #notDone = True
260 #while (notDone):
261 #    if (test.robot_found()):
262 #        print "found robot"
263 #        while True:
264 #            a = input("Set angle: ")
265 #            test.set_angle_for_servo("0", a)
266 
267 
268 #s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
269 #s.connect((socket.gethostbyname("eit.no-ip.biz"), 10002))
270 #s.sendall("{1,OK,0,0,0}")
271 
272 # END OF TEST CODE
```

I.11 et/debug.py

```
1  '''
2  Created on 5. mars 2013
3  @author: Eivind
4  '''
5  
6  from sys import stderr
7  from EKey import EKey
8  
9  # Specific debugging
10 LINKER = True
11 WHEELS = True
12 SENSOR = False
13 GUI_SENSOR = False
14 GUI_INPUT = False
15 GUI = True
16 
17 # High/Low debugging
18 DEBUG_H = True
19 DEBUG_L = False
20 DEBUG_E = False
21 
22 # See Debug (global object below class definition)
23 class Debugger(object):
24 
25     # Console
26     __controller = None
27 
28 
```

```
29      # Constructor
30      def __init__(self):
31          pass
32
33
34      # Set controller as target console
35      def setConsole(self, controller):
36          self.__controller = controller
37
38
39      # Debug print High level
40      def debugH(self, text):
41          if DEBUG_H: self.__out(text)
42
43
44      # Debug print Low level
45      def debugL(self, text):
46          if DEBUG_L: self.__out(text)
47
48
49      # Debug print Error
50      def debugE(self, text):
51          if DEBUG_E:
52              stderr.write("ERROR: " +  text + "\n")
53              self.__outConsole(text)
54
55
56
57      def __out(self, text):
58          print(text)
59          self.__outConsole(text)
60
61      def __outConsole(self, text):
62          if self.__controller <> None:
63              self.__controller.writeToConsole(text)
64
65      def GUIAddedItem(self, name):
66          if GUI: self.__out("GUI: Added {0}".format(name))
67
68      def GUISensorDataUpdate(self, sensor):
69          if GUI_SENSOR: self.__out("GUI: Sensor data update: ID = {0}, Angle = {1}
                ".format(sensor.ID, sensor.realValue))
70
71      def GUIKeyPressed(self, key):
72          if GUI_INPUT: self.__out("GUI: KeyPress: " + EKey.getName(key))
73
74      def GUIKeyReleased(self, key):
75          if GUI_INPUT: self.__out("GUI: KeyRelease: " + EKey.getName(key))
76
77      def SensorRawToReal(self, rawValue, realValue):
78          if SENSOR: self.__out("Raw to Real:\t{0}\t-->\t{1}".format(rawValue,
                realValue))
79
80      def SensorMCUFail(self, ID):
81          if SENSOR: self.__out("No raw value from sensor {0}!".format(ID))
82
```

```
83      def LinkerGroup(self, servoID_1, servoID_2):
84          if LINKER: self.__out("Controller: Group ServoID {0} and {1}".format(
                servoID_1, servoID_2))
85
86      def LinkerUngroup(self, servoID_1, servoID_2):
87          if LINKER: self.__out("Controller: UN-group ServoID {0} and {1}".format(
                servoID_1, servoID_2))
88
89      def LinkerLink(self, sensorID, servoID):
90          if LINKER: self.__out("Linker: Link SensorID {0} and ServoID {1}".format(
                sensorID, servoID))
91
92      def LinkerUnlink(self, sensorID, servoID):
93          if LINKER: self.__out("Linker: UN-link SensorID {0} and ServoID {1}".
                format(sensorID, servoID))
94
95      def WheelsDrive(self, left, right):
96          if WHEELS: self.__out("Wheels: Left = {0}, Right = {1}".format(left,
                right))
97
98      def WheelsSpeed(self, speed):
99          if WHEELS: self.__out("Wheels: Setting wheel speed to {0}".format(speed))
100
101     def WheelsSet(self, leftID, rightID):
102         if WHEELS: self.__out("Wheels: Setting Left = {0}, Right = {1}".format(
                leftID, rightID))
103
104 # Global debugger object
105 Debug = Debugger()
```

I.12 et/dict.py

```
1  '''
2  Created on 21. mars 2013
3  @author: Eivind
4  '''
5
6  class ArrayDictionary(object):
7
8      # Item:Item-Array Dictionary
9      __items = {}
10
11
12     # Constructor
13     def __init__(self):
14         pass
15
16
17     # Add an item to the array of given key
18     def add(self, key, item):
19
20         if not key in self.__items.keys():
21             self.__items[key] = []
22
23         # Check if key already exists
24         if item in self.__items[key]:
```

```
            return
            #raise Exception("Item of ID {0} already exists in IDDictionary." \
            #                .format(item.ID))

        # Add to dictionary
        self.__items[key].append(item)


    # Adds two keys/items and adds a reverse reference
    def addDuoKeyItems(self, keyItem1, keyItem2):
        self.add(keyItem1, keyItem2)
        self.add(keyItem2, keyItem1)
        self.add(keyItem1, keyItem1)
        self.add(keyItem2, keyItem2)


    # Remove a key-item pair
    def remove(self, key, item):
        if item in self.tryGetItems(key):
            self.__items[key].remove(item)


    # Removes two keys/items with their reverse references
    def removeDuoKeyItems(self, keyItem1, keyItem2):
        self.remove(keyItem1, keyItem2)
        self.remove(keyItem2, keyItem1)
        self.remove(keyItem1, keyItem1)
        self.remove(keyItem2, keyItem2)


    # "Property" - Items
    def Items(self, key): return self.__items[key]


    # Property - AllItems
    def AllItems(): #@NoSelf
        def fget(self):
            result = []
            for items in self.__items.values():
                for item in items:
                    if not item in result: result.append(item)
            return result
        return locals()
    AllItems = property(**AllItems())


    # Property - Count
    def Count(): #@NoSelf
        def fget(self): return len(self.__items)
        return locals()
    Count = property(**Count())


    # Attempt to get items by key
    # Returns default if not found
    def tryGetItems(self, key, default = []):
```

```
81          if key in self.__items:
82              #print "DICT: TryGet Returns {0}".format(self.__items[key])
83              return self.__items[key]
84          else:
85              #print "DICT: TryGet Returns {0}".format(default)
86              return default
87
88
89      # Attempt to get items by key array
90      # Returns default if not found
91      def tryGetAllItems(self, keys, default = []):
92          result = default[:]
93          for key in keys:
94              arr = [self.tryGetItems(key)]
95              for items in arr:
96                  for item in items:
97                      result.append(item)
98          return result
99
100
101
102
103 class IDDictionary(object):
104
105     # ID:Item Dictionary
106     __items = {}
107
108
109     # Constructor
110     def __init__(self):
111         self.__items = {}
112
113
114     # Add an ID-item to the dictionary
115     def add(self, item):
116
117         # Check if ID already exists
118         if item.ID in self.__items:
119             raise Exception("Item of ID {0} already exists in IDDictionary." \
120                             .format(item.ID))
121
122         # Add to dictionary
123         self.__items[item.ID] = item
124
125
126     # "Property" - Item
127     def Item(self, ID): return self.__items[ID]
128
129
130     # Property - Items
131     def Items(): #@NoSelf
132         def fget(self): return self.__items.values()
133         return locals()
134     Items = property(**Items())
135
136
```

```
    # Property - Count
    def Count(): #@NoSelf
        def fget(self): return len(self.__items)
        return locals()
    Count = property(**Count())


    # Attempt to get item by ID
    # Returns None if not found
    def tryGetItem(self, ID):
        if ID in self.__items:
            return self.__items[ID]
        else:
            return None









'''

    # Add an item to the array of given key
    def add(self, key, item):

        if not key in self.__items.keys():
            print "DICT: Creating empty array for key {0}".format(key)
            self.__items[key] = []

        # Check if key already exists
        if item in self.__items[key]:
            print "DICT: Key {0} did not exist".format(key)
            return
            #raise Exception("Item of ID {0} already exists in IDDictionary." \
            #                .format(item.ID))

        # Add to dictionary
        print "DICT: Appending {0} to key {1}".format(item, key)
        print "DICT: Items before append: {0}".format(self.__items[key])
        self.__items[key].append(item)
        print "DICT: Items after append: {0}".format(self.__items[key])




    # Attempt to get items by key array
    # Returns default if not found
    def tryGetAllItems(self, keys, default = []):
        print "DICT: TryGetAll keys {0}".format(keys)
        print "DICT: Default before start {0}".format(default)
        result = default[:]
        print "DICT: Result before start {0}".format(result)
        for key in keys:
            print "DICT: Checking key {0}".format(key)
            arr = [self.tryGetItems(key)]
            for items in arr:
```

```
                print "DICT: TryGetAll looping items {0}".format(items)
                for item in items:
                    print "DICT: TryGetAll appending item {0}".format(item)
                    result.append(item)

        print "DICT: TryGetAll Returning {0}".format(result)
        return result


'''
```

I.13 et/Ekey.py

```
'''
Created on 6. mars 2013

@author: Eivind
'''
from PyQt4.QtCore import Qt


class EKey(object):

    NoKey = 0
    Up = 1
    Down = 2
    Left = 3
    Right = 4
    Stop = 5

    NAMES = ["No Key", "Up", "Down", "Left", "Right", "Stop"]

    @staticmethod
    def fromQtKey(qtKey):
        if    qtKey == Qt.Key_Up     or qtKey == Qt.Key_W:       return EKey.Up
        elif  qtKey == Qt.Key_Down   or qtKey == Qt.Key_S:       return EKey.Down
        elif  qtKey == Qt.Key_Left   or qtKey == Qt.Key_A:       return EKey.Left
        elif  qtKey == Qt.Key_Right  or qtKey == Qt.Key_D:       return EKey.Right
        elif  qtKey == Qt.Key_Escape or qtKey == Qt.Key_Space:   return EKey.Stop
        else: return EKey.NoKey

    @staticmethod
    def getNameQT(qtKey):
        return EKey.NAMES[EKey.fromQtKey(qtKey)]

    @staticmethod
    def getName(key):
        return EKey.NAMES[key]
```

I.14 et/ESensor.py

```
'''
Created on 5. mars 2013

@author: Eivind
'''
from PyQt4.QtCore import QStringList
```

```
7
8  # "ENUM" that handles various indexed information about SENSOR
9  # Used for some GUI table layout and cell indexing
10 class ESensor(object):
11
12     # ENUM Values
13     Img = 0
14     Name = 1
15     Angle = 2
16     Raw = 3
17
18     # Names
19     NAMES = QStringList()
20     NAMES.append("")
21     NAMES.append("Name")
22     NAMES.append("Angle")
23     NAMES.append("Analog")
24
25     # Sizes (column widths)
26     SIZES = [30, 60, 45, 45]
27
28     # ENUM Count
29     COUNT = NAMES.count()
30
31     # ENUM Name
32     ENUM_NAME = "Sensor"
```

I.15 et/EServo.py

```
1  '''
2  Created on 20. feb. 2013
3
4  @author: Eivind
5  '''
6  from PyQt4.QtCore import QStringList
7
8  # "ENUM" that handles various indexed information about SERVO
9  # Used for some GUI table layout and cell indexing
10 class EServo(object):
11
12     # ENUM Values
13     Img = 0
14     Name = 1
15     Angle = 2
16     Load = 3
17     Temp = 4
18     Volt = 5
19     MinPos = 6
20     MaxPos = 7
21     Status = 8
22
23     # Names
24     NAMES = QStringList()
25     NAMES.append("")
26     NAMES.append("Name")
27     NAMES.append("Angle")
```

```
    NAMES.append("Load")
    NAMES.append("Temp")
    NAMES.append("Volt")
    NAMES.append("Min.Pos")
    NAMES.append("Max.Pos")
    NAMES.append("Status")


    # Sizes (column widths)
    SIZES = [30, 60, 45, 42, 42, 42, 55, 55, 70]

    # ENUM Count
    COUNT = NAMES.count()

    # ENUM Name
    ENUM_NAME = "Servo"
```

I.16 et/guiDesign.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'Prototype.ui'
#
# Created: Thu Apr 18 12:55:44 2013
#      by: PyQt4 UI code generator 4.9.6
#
# WARNING! All changes made in this file will be lost!

from PyQt4 import QtCore, QtGui

try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)

class Ui_Main(object):
    def setupUi(self, Main):
        Main.setObjectName(_fromUtf8("Main"))
        Main.resize(1000, 671)
        self.centralwidget = QtGui.QWidget(Main)
        self.centralwidget.setObjectName(_fromUtf8("centralwidget"))
        self.frame = QtGui.QFrame(self.centralwidget)
        self.frame.setGeometry(QtCore.QRect(0, 0, 1001, 71))
        self.frame.setMouseTracking(False)
        self.frame.setAutoFillBackground(False)
        self.frame.setFrameShape(QtGui.QFrame.StyledPanel)
        self.frame.setFrameShadow(QtGui.QFrame.Plain)
```

```
38          self.frame.setLineWidth(1)
39          self.frame.setMidLineWidth(0)
40          self.frame.setObjectName(_fromUtf8("frame"))
41          self.label = QtGui.QLabel(self.frame)
42          self.label.setGeometry(QtCore.QRect(50, 10, 901, 61))
43          font = QtGui.QFont()
44          font.setFamily(_fromUtf8("Narkisim"))
45          font.setPointSize(30)
46          self.label.setFont(font)
47          self.label.setAlignment(QtCore.Qt.AlignCenter)
48          self.label.setObjectName(_fromUtf8("label"))
49          self.frame_2 = QtGui.QFrame(self.centralwidget)
50          self.frame_2.setGeometry(QtCore.QRect(0, 370, 1001, 471))
51          font = QtGui.QFont()
52          font.setPointSize(8)
53          self.frame_2.setFont(font)
54          self.frame_2.setMouseTracking(False)
55          self.frame_2.setAutoFillBackground(False)
56          self.frame_2.setFrameShape(QtGui.QFrame.StyledPanel)
57          self.frame_2.setFrameShadow(QtGui.QFrame.Plain)
58          self.frame_2.setLineWidth(1)
59          self.frame_2.setMidLineWidth(0)
60          self.frame_2.setObjectName(_fromUtf8("frame_2"))
61          self.layoutWidget_3 = QtGui.QWidget(self.frame_2)
62          self.layoutWidget_3.setGeometry(QtCore.QRect(20, 20, 731, 241))
63          self.layoutWidget_3.setObjectName(_fromUtf8("layoutWidget_3"))
64          self.verticalLayout_4 = QtGui.QVBoxLayout(self.layoutWidget_3)
65          self.verticalLayout_4.setMargin(0)
66          self.verticalLayout_4.setObjectName(_fromUtf8("verticalLayout_4"))
67          self.lblTblSensors_2 = QtGui.QLabel(self.layoutWidget_3)
68          palette = QtGui.QPalette()
69          brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
70          brush.setStyle(QtCore.Qt.SolidPattern)
71          palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
72          brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
73          brush.setStyle(QtCore.Qt.SolidPattern)
74          palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
75          brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
76          brush.setStyle(QtCore.Qt.SolidPattern)
77          palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText,
                brush)
78          brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
79          brush.setStyle(QtCore.Qt.SolidPattern)
80          palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
81          brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
82          brush.setStyle(QtCore.Qt.SolidPattern)
83          palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText,
                brush)
84          brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
85          brush.setStyle(QtCore.Qt.SolidPattern)
86          palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
87          self.lblTblSensors_2.setPalette(palette)
88          font = QtGui.QFont()
89          font.setPointSize(14)
90          self.lblTblSensors_2.setFont(font)
91          self.lblTblSensors_2.setIndent(3)
```

```
92          self.lblTblSensors_2.setObjectName(_fromUtf8("lblTblSensors_2"))
93          self.verticalLayout_4.addWidget(self.lblTblSensors_2)
94          self.txtConsole = QtGui.QTextEdit(self.layoutWidget_3)
95          self.txtConsole.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOn)
96          self.txtConsole.setObjectName(_fromUtf8("txtConsole"))
97          self.verticalLayout_4.addWidget(self.txtConsole)
98          self.horizontalLayout_4 = QtGui.QHBoxLayout()
99          self.horizontalLayout_4.setContentsMargins(-1, -1, 450, -1)
100         self.horizontalLayout_4.setObjectName(_fromUtf8("horizontalLayout_4"))
101         self.btnConsoleClear = QtGui.QPushButton(self.layoutWidget_3)
102         self.btnConsoleClear.setEnabled(True)
103         font = QtGui.QFont()
104         font.setPointSize(10)
105         self.btnConsoleClear.setFont(font)
106         self.btnConsoleClear.setObjectName(_fromUtf8("btnConsoleClear"))
107         self.horizontalLayout_4.addWidget(self.btnConsoleClear)
108         self.btnConsole2 = QtGui.QPushButton(self.layoutWidget_3)
109         self.btnConsole2.setEnabled(False)
110         font = QtGui.QFont()
111         font.setPointSize(10)
112         self.btnConsole2.setFont(font)
113         self.btnConsole2.setText(_fromUtf8(""))
114         self.btnConsole2.setObjectName(_fromUtf8("btnConsole2"))
115         self.horizontalLayout_4.addWidget(self.btnConsole2)
116         self.verticalLayout_4.addLayout(self.horizontalLayout_4)
117         self.layoutWidget_2 = QtGui.QWidget(self.frame_2)
118         self.layoutWidget_2.setGeometry(QtCore.QRect(770, 20, 211, 241))
119         self.layoutWidget_2.setObjectName(_fromUtf8("layoutWidget_2"))
120         self.verticalLayout_3 = QtGui.QVBoxLayout(self.layoutWidget_2)
121         self.verticalLayout_3.setMargin(0)
122         self.verticalLayout_3.setObjectName(_fromUtf8("verticalLayout_3"))
123         self.lblLiveView = QtGui.QLabel(self.layoutWidget_2)
124         self.lblLiveView.setEnabled(False)
125         palette = QtGui.QPalette()
126         brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
127         brush.setStyle(QtCore.Qt.SolidPattern)
128         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
129         brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
130         brush.setStyle(QtCore.Qt.SolidPattern)
131         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
132         brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
133         brush.setStyle(QtCore.Qt.SolidPattern)
134         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText,
                brush)
135         brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
136         brush.setStyle(QtCore.Qt.SolidPattern)
137         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
138         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
139         brush.setStyle(QtCore.Qt.SolidPattern)
140         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText,
                brush)
141         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
142         brush.setStyle(QtCore.Qt.SolidPattern)
143         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
144         self.lblLiveView.setPalette(palette)
145         font = QtGui.QFont()
```

```
146         font.setPointSize(14)
147         self.lblLiveView.setFont(font)
148         self.lblLiveView.setIndent(3)
149         self.lblLiveView.setObjectName(_fromUtf8("lblLiveView"))
150         self.verticalLayout_3.addWidget(self.lblLiveView)
151         self.tblLiveView = QtGui.QTableWidget(self.layoutWidget_2)
152         self.tblLiveView.setEnabled(False)
153         self.tblLiveView.setFrameShape(QtGui.QFrame.StyledPanel)
154         self.tblLiveView.setFrameShadow(QtGui.QFrame.Plain)
155         self.tblLiveView.setAlternatingRowColors(True)
156         self.tblLiveView.setSelectionMode(QtGui.QAbstractItemView.SingleSelection
                )
157         self.tblLiveView.setSelectionBehavior(QtGui.QAbstractItemView.SelectRows)
158         self.tblLiveView.setCornerButtonEnabled(False)
159         self.tblLiveView.setObjectName(_fromUtf8("tblLiveView"))
160         self.tblLiveView.setColumnCount(0)
161         self.tblLiveView.setRowCount(0)
162         self.tblLiveView.verticalHeader().setVisible(False)
163         self.verticalLayout_3.addWidget(self.tblLiveView)
164         self.horizontalLayout_3 = QtGui.QHBoxLayout()
165         self.horizontalLayout_3.setObjectName(_fromUtf8("horizontalLayout_3"))
166         self.btnLiveView1 = QtGui.QPushButton(self.layoutWidget_2)
167         self.btnLiveView1.setEnabled(False)
168         self.btnLiveView1.setText(_fromUtf8(""))
169         self.btnLiveView1.setObjectName(_fromUtf8("btnLiveView1"))
170         self.horizontalLayout_3.addWidget(self.btnLiveView1)
171         self.btnLiveView2 = QtGui.QPushButton(self.layoutWidget_2)
172         self.btnLiveView2.setEnabled(False)
173         self.btnLiveView2.setText(_fromUtf8(""))
174         self.btnLiveView2.setObjectName(_fromUtf8("btnLiveView2"))
175         self.horizontalLayout_3.addWidget(self.btnLiveView2)
176         self.verticalLayout_3.addLayout(self.horizontalLayout_3)
177         self.frame_4 = QtGui.QFrame(self.centralwidget)
178         self.frame_4.setGeometry(QtCore.QRect(0, 70, 1001, 301))
179         self.frame_4.setMouseTracking(False)
180         self.frame_4.setAutoFillBackground(False)
181         self.frame_4.setFrameShape(QtGui.QFrame.StyledPanel)
182         self.frame_4.setFrameShadow(QtGui.QFrame.Plain)
183         self.frame_4.setLineWidth(1)
184         self.frame_4.setMidLineWidth(0)
185         self.frame_4.setObjectName(_fromUtf8("frame_4"))
186         self.layoutWidget = QtGui.QWidget(self.frame_4)
187         self.layoutWidget.setGeometry(QtCore.QRect(20, 20, 211, 251))
188         self.layoutWidget.setObjectName(_fromUtf8("layoutWidget"))
189         self.verticalLayout = QtGui.QVBoxLayout(self.layoutWidget)
190         self.verticalLayout.setMargin(0)
191         self.verticalLayout.setObjectName(_fromUtf8("verticalLayout"))
192         self.lblTblSensors = QtGui.QLabel(self.layoutWidget)
193         palette = QtGui.QPalette()
194         brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
195         brush.setStyle(QtCore.Qt.SolidPattern)
196         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
197         brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
198         brush.setStyle(QtCore.Qt.SolidPattern)
199         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
200         brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
```

```
201         brush.setStyle(QtCore.Qt.SolidPattern)
202         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText,
                brush)
203         brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
204         brush.setStyle(QtCore.Qt.SolidPattern)
205         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
206         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
207         brush.setStyle(QtCore.Qt.SolidPattern)
208         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText,
                brush)
209         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
210         brush.setStyle(QtCore.Qt.SolidPattern)
211         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
212         self.lblTblSensors.setPalette(palette)
213         font = QtGui.QFont()
214         font.setPointSize(14)
215         self.lblTblSensors.setFont(font)
216         self.lblTblSensors.setIndent(3)
217         self.lblTblSensors.setObjectName(_fromUtf8("lblTblSensors"))
218         self.verticalLayout.addWidget(self.lblTblSensors)
219         self.tblSensor = QtGui.QTableWidget(self.layoutWidget)
220         self.tblSensor.setFrameShape(QtGui.QFrame.StyledPanel)
221         self.tblSensor.setFrameShadow(QtGui.QFrame.Plain)
222         self.tblSensor.setAlternatingRowColors(True)
223         self.tblSensor.setSelectionMode(QtGui.QAbstractItemView.SingleSelection)
224         self.tblSensor.setSelectionBehavior(QtGui.QAbstractItemView.SelectRows)
225         self.tblSensor.setCornerButtonEnabled(False)
226         self.tblSensor.setObjectName(_fromUtf8("tblSensor"))
227         self.tblSensor.setColumnCount(0)
228         self.tblSensor.setRowCount(0)
229         self.tblSensor.verticalHeader().setVisible(False)
230         self.verticalLayout.addWidget(self.tblSensor)
231         self.horizontalLayout = QtGui.QHBoxLayout()
232         self.horizontalLayout.setObjectName(_fromUtf8("horizontalLayout"))
233         self.btnTblSensorsLink = QtGui.QPushButton(self.layoutWidget)
234         self.btnTblSensorsLink.setEnabled(True)
235         font = QtGui.QFont()
236         font.setPointSize(10)
237         self.btnTblSensorsLink.setFont(font)
238         self.btnTblSensorsLink.setObjectName(_fromUtf8("btnTblSensorsLink"))
239         self.horizontalLayout.addWidget(self.btnTblSensorsLink)
240         self.btnTblSensorsUnlink = QtGui.QPushButton(self.layoutWidget)
241         self.btnTblSensorsUnlink.setEnabled(True)
242         font = QtGui.QFont()
243         font.setPointSize(10)
244         self.btnTblSensorsUnlink.setFont(font)
245         self.btnTblSensorsUnlink.setObjectName(_fromUtf8("btnTblSensorsUnlink"))
246         self.horizontalLayout.addWidget(self.btnTblSensorsUnlink)
247         self.verticalLayout.addLayout(self.horizontalLayout)
248         self.layoutWidget1 = QtGui.QWidget(self.frame_4)
249         self.layoutWidget1.setGeometry(QtCore.QRect(250, 20, 501, 251))
250         self.layoutWidget1.setObjectName(_fromUtf8("layoutWidget1"))
251         self.verticalLayout_2 = QtGui.QVBoxLayout(self.layoutWidget1)
252         self.verticalLayout_2.setMargin(0)
253         self.verticalLayout_2.setObjectName(_fromUtf8("verticalLayout_2"))
254         self.lblTblServos = QtGui.QLabel(self.layoutWidget1)
```

```
255         palette = QtGui.QPalette()
256         brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
257         brush.setStyle(QtCore.Qt.SolidPattern)
258         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
259         brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
260         brush.setStyle(QtCore.Qt.SolidPattern)
261         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
262         brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
263         brush.setStyle(QtCore.Qt.SolidPattern)
264         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText,
                brush)
265         brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
266         brush.setStyle(QtCore.Qt.SolidPattern)
267         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
268         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
269         brush.setStyle(QtCore.Qt.SolidPattern)
270         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText,
                brush)
271         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
272         brush.setStyle(QtCore.Qt.SolidPattern)
273         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
274         self.lblTblServos.setPalette(palette)
275         font = QtGui.QFont()
276         font.setPointSize(14)
277         self.lblTblServos.setFont(font)
278         self.lblTblServos.setIndent(3)
279         self.lblTblServos.setObjectName(_fromUtf8("lblTblServos"))
280         self.verticalLayout_2.addWidget(self.lblTblServos)
281         self.tblServo = QtGui.QTableWidget(self.layoutWidget1)
282         self.tblServo.setFrameShape(QtGui.QFrame.StyledPanel)
283         self.tblServo.setFrameShadow(QtGui.QFrame.Plain)
284         self.tblServo.setEditTriggers(QtGui.QAbstractItemView.NoEditTriggers)
285         self.tblServo.setAlternatingRowColors(True)
286         self.tblServo.setSelectionMode(QtGui.QAbstractItemView.SingleSelection)
287         self.tblServo.setSelectionBehavior(QtGui.QAbstractItemView.SelectRows)
288         self.tblServo.setCornerButtonEnabled(False)
289         self.tblServo.setObjectName(_fromUtf8("tblServo"))
290         self.tblServo.setColumnCount(0)
291         self.tblServo.setRowCount(0)
292         self.tblServo.verticalHeader().setVisible(False)
293         self.verticalLayout_2.addWidget(self.tblServo)
294         self.horizontalLayout_2 = QtGui.QHBoxLayout()
295         self.horizontalLayout_2.setObjectName(_fromUtf8("horizontalLayout_2"))
296         self.btnTblServosGroup = QtGui.QPushButton(self.layoutWidget1)
297         self.btnTblServosGroup.setEnabled(True)
298         font = QtGui.QFont()
299         font.setPointSize(10)
300         self.btnTblServosGroup.setFont(font)
301         self.btnTblServosGroup.setObjectName(_fromUtf8("btnTblServosGroup"))
302         self.horizontalLayout_2.addWidget(self.btnTblServosGroup)
303         self.btnTblServosUngroup = QtGui.QPushButton(self.layoutWidget1)
304         self.btnTblServosUngroup.setEnabled(True)
305         font = QtGui.QFont()
306         font.setPointSize(10)
307         self.btnTblServosUngroup.setFont(font)
308         self.btnTblServosUngroup.setObjectName(_fromUtf8("btnTblServosUngroup"))
```

```
309         self.horizontalLayout_2.addWidget(self.btnTblServosUngroup)
310         self.btnTblServosSetAngle = QtGui.QPushButton(self.layoutWidget1)
311         self.btnTblServosSetAngle.setEnabled(True)
312         font = QtGui.QFont()
313         font.setPointSize(10)
314         self.btnTblServosSetAngle.setFont(font)
315         self.btnTblServosSetAngle.setObjectName(_fromUtf8("btnTblServosSetAngle")
                )
316         self.horizontalLayout_2.addWidget(self.btnTblServosSetAngle)
317         self.btnTblServosSetWheels = QtGui.QPushButton(self.layoutWidget1)
318         self.btnTblServosSetWheels.setEnabled(True)
319         font = QtGui.QFont()
320         font.setPointSize(10)
321         self.btnTblServosSetWheels.setFont(font)
322         self.btnTblServosSetWheels.setObjectName(_fromUtf8("btnTblServosSetWheels
                "))
323         self.horizontalLayout_2.addWidget(self.btnTblServosSetWheels)
324         self.btnTblServosWiggle = QtGui.QPushButton(self.layoutWidget1)
325         font = QtGui.QFont()
326         font.setPointSize(10)
327         self.btnTblServosWiggle.setFont(font)
328         self.btnTblServosWiggle.setObjectName(_fromUtf8("btnTblServosWiggle"))
329         self.horizontalLayout_2.addWidget(self.btnTblServosWiggle)
330         self.verticalLayout_2.addLayout(self.horizontalLayout_2)
331         self.layoutWidget_4 = QtGui.QWidget(self.frame_4)
332         self.layoutWidget_4.setGeometry(QtCore.QRect(770, 18, 211, 251))
333         self.layoutWidget_4.setObjectName(_fromUtf8("layoutWidget_4"))
334         self.verticalLayout_5 = QtGui.QVBoxLayout(self.layoutWidget_4)
335         self.verticalLayout_5.setMargin(0)
336         self.verticalLayout_5.setObjectName(_fromUtf8("verticalLayout_5"))
337         self.lblOptions = QtGui.QLabel(self.layoutWidget_4)
338         self.lblOptions.setEnabled(True)
339         palette = QtGui.QPalette()
340         brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
341         brush.setStyle(QtCore.Qt.SolidPattern)
342         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
343         brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
344         brush.setStyle(QtCore.Qt.SolidPattern)
345         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
346         brush = QtGui.QBrush(QtGui.QColor(32, 102, 168))
347         brush.setStyle(QtCore.Qt.SolidPattern)
348         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText,
                brush)
349         brush = QtGui.QBrush(QtGui.QColor(0, 0, 0))
350         brush.setStyle(QtCore.Qt.SolidPattern)
351         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
352         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
353         brush.setStyle(QtCore.Qt.SolidPattern)
354         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText,
                brush)
355         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
356         brush.setStyle(QtCore.Qt.SolidPattern)
357         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
358         self.lblOptions.setPalette(palette)
359         font = QtGui.QFont()
360         font.setPointSize(14)
```

```
361         self.lblOptions.setFont(font)
362         self.lblOptions.setIndent(3)
363         self.lblOptions.setObjectName(_fromUtf8("lblOptions"))
364         self.verticalLayout_5.addWidget(self.lblOptions)
365         self.frame_3 = QtGui.QFrame(self.layoutWidget_4)
366         sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Preferred, QtGui.
                QSizePolicy.Preferred)
367         sizePolicy.setHorizontalStretch(0)
368         sizePolicy.setVerticalStretch(2)
369         sizePolicy.setHeightForWidth(self.frame_3.sizePolicy().hasHeightForWidth
                ())
370         self.frame_3.setSizePolicy(sizePolicy)
371         self.frame_3.setFrameShape(QtGui.QFrame.Box)
372         self.frame_3.setFrameShadow(QtGui.QFrame.Raised)
373         self.frame_3.setMidLineWidth(0)
374         self.frame_3.setObjectName(_fromUtf8("frame_3"))
375         self.layoutWidget2 = QtGui.QWidget(self.frame_3)
376         self.layoutWidget2.setGeometry(QtCore.QRect(10, 190, 181, 24))
377         self.layoutWidget2.setObjectName(_fromUtf8("layoutWidget2"))
378         self.gridLayout_3 = QtGui.QGridLayout(self.layoutWidget2)
379         self.gridLayout_3.setMargin(0)
380         self.gridLayout_3.setObjectName(_fromUtf8("gridLayout_3"))
381         self.label_4 = QtGui.QLabel(self.layoutWidget2)
382         self.label_4.setEnabled(False)
383         font = QtGui.QFont()
384         font.setPointSize(10)
385         self.label_4.setFont(font)
386         self.label_4.setObjectName(_fromUtf8("label_4"))
387         self.gridLayout_3.addWidget(self.label_4, 0, 0, 1, 1)
388         self.txtCommFreq = QtGui.QLineEdit(self.layoutWidget2)
389         self.txtCommFreq.setEnabled(False)
390         self.txtCommFreq.setObjectName(_fromUtf8("txtCommFreq"))
391         self.gridLayout_3.addWidget(self.txtCommFreq, 0, 1, 1, 1)
392         self.label_5 = QtGui.QLabel(self.layoutWidget2)
393         self.label_5.setEnabled(False)
394         self.label_5.setObjectName(_fromUtf8("label_5"))
395         self.gridLayout_3.addWidget(self.label_5, 0, 2, 1, 1)
396         self.layoutWidget3 = QtGui.QWidget(self.frame_3)
397         self.layoutWidget3.setGeometry(QtCore.QRect(10, 10, 181, 161))
398         self.layoutWidget3.setObjectName(_fromUtf8("layoutWidget3"))
399         self.verticalLayout_6 = QtGui.QVBoxLayout(self.layoutWidget3)
400         self.verticalLayout_6.setMargin(0)
401         self.verticalLayout_6.setObjectName(_fromUtf8("verticalLayout_6"))
402         self.frame_5 = QtGui.QFrame(self.layoutWidget3)
403         self.frame_5.setFrameShape(QtGui.QFrame.StyledPanel)
404         self.frame_5.setFrameShadow(QtGui.QFrame.Raised)
405         self.frame_5.setObjectName(_fromUtf8("frame_5"))
406         self.label_2 = QtGui.QLabel(self.frame_5)
407         self.label_2.setGeometry(QtCore.QRect(20, 20, 141, 41))
408         sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Preferred, QtGui.
                QSizePolicy.Preferred)
409         sizePolicy.setHorizontalStretch(0)
410         sizePolicy.setVerticalStretch(0)
411         sizePolicy.setHeightForWidth(self.label_2.sizePolicy().hasHeightForWidth
                ())
412         self.label_2.setSizePolicy(sizePolicy)
```

```
413         palette = QtGui.QPalette()
414         brush = QtGui.QBrush(QtGui.QColor(0, 85, 127))
415         brush.setStyle(QtCore.Qt.SolidPattern)
416         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
417         brush = QtGui.QBrush(QtGui.QColor(0, 85, 127))
418         brush.setStyle(QtCore.Qt.SolidPattern)
419         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText,
                brush)
420         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
421         brush.setStyle(QtCore.Qt.SolidPattern)
422         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText,
                brush)
423         self.label_2.setPalette(palette)
424         font = QtGui.QFont()
425         font.setPointSize(7)
426         self.label_2.setFont(font)
427         self.label_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|
                QtCore.Qt.AlignTop)
428         self.label_2.setWordWrap(True)
429         self.label_2.setObjectName(_fromUtf8("label_2"))
430         self.chkAutonomous = QtGui.QCheckBox(self.frame_5)
431         self.chkAutonomous.setGeometry(QtCore.QRect(0, 0, 129, 20))
432         sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Minimum, QtGui.
                QSizePolicy.Fixed)
433         sizePolicy.setHorizontalStretch(0)
434         sizePolicy.setVerticalStretch(0)
435         sizePolicy.setHeightForWidth(self.chkAutonomous.sizePolicy().
                hasHeightForWidth())
436         self.chkAutonomous.setSizePolicy(sizePolicy)
437         font = QtGui.QFont()
438         font.setPointSize(10)
439         self.chkAutonomous.setFont(font)
440         self.chkAutonomous.setObjectName(_fromUtf8("chkAutonomous"))
441         self.frame_6 = QtGui.QFrame(self.frame_5)
442         self.frame_6.setGeometry(QtCore.QRect(0, 60, 179, 76))
443         sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Preferred, QtGui.
                QSizePolicy.Preferred)
444         sizePolicy.setHorizontalStretch(0)
445         sizePolicy.setVerticalStretch(0)
446         sizePolicy.setHeightForWidth(self.frame_6.sizePolicy().hasHeightForWidth
                ())
447         self.frame_6.setSizePolicy(sizePolicy)
448         self.frame_6.setFrameShape(QtGui.QFrame.StyledPanel)
449         self.frame_6.setFrameShadow(QtGui.QFrame.Raised)
450         self.frame_6.setObjectName(_fromUtf8("frame_6"))
451         self.label_3 = QtGui.QLabel(self.frame_6)
452         self.label_3.setGeometry(QtCore.QRect(20, 20, 131, 31))
453         sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Preferred, QtGui.
                QSizePolicy.Preferred)
454         sizePolicy.setHorizontalStretch(0)
455         sizePolicy.setVerticalStretch(0)
456         sizePolicy.setHeightForWidth(self.label_3.sizePolicy().hasHeightForWidth
                ())
457         self.label_3.setSizePolicy(sizePolicy)
458         palette = QtGui.QPalette()
459         brush = QtGui.QBrush(QtGui.QColor(0, 85, 127))
```

```
460         brush.setStyle(QtCore.Qt.SolidPattern)
461         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
462         brush = QtGui.QBrush(QtGui.QColor(0, 85, 127))
463         brush.setStyle(QtCore.Qt.SolidPattern)
464         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText,
                brush)
465         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
466         brush.setStyle(QtCore.Qt.SolidPattern)
467         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText,
                brush)
468         self.label_3.setPalette(palette)
469         font = QtGui.QFont()
470         font.setPointSize(7)
471         self.label_3.setFont(font)
472         self.label_3.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|
                QtCore.Qt.AlignTop)
473         self.label_3.setWordWrap(True)
474         self.label_3.setObjectName(_fromUtf8("label_3"))
475         self.chkWheelsEnabled = QtGui.QCheckBox(self.frame_6)
476         self.chkWheelsEnabled.setGeometry(QtCore.QRect(0, 0, 181, 20))
477         sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Minimum, QtGui.
                QSizePolicy.Fixed)
478         sizePolicy.setHorizontalStretch(0)
479         sizePolicy.setVerticalStretch(0)
480         sizePolicy.setHeightForWidth(self.chkWheelsEnabled.sizePolicy().
                hasHeightForWidth())
481         self.chkWheelsEnabled.setSizePolicy(sizePolicy)
482         font = QtGui.QFont()
483         font.setPointSize(10)
484         self.chkWheelsEnabled.setFont(font)
485         self.chkWheelsEnabled.setObjectName(_fromUtf8("chkWheelsEnabled"))
486         self.txtWheelVelocity = QtGui.QLineEdit(self.frame_6)
487         self.txtWheelVelocity.setEnabled(True)
488         self.txtWheelVelocity.setGeometry(QtCore.QRect(60, 50, 54, 16))
489         font = QtGui.QFont()
490         font.setPointSize(7)
491         self.txtWheelVelocity.setFont(font)
492         self.txtWheelVelocity.setObjectName(_fromUtf8("txtWheelVelocity"))
493         self.label_6 = QtGui.QLabel(self.frame_6)
494         self.label_6.setGeometry(QtCore.QRect(20, 50, 41, 16))
495         sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Preferred, QtGui.
                QSizePolicy.Preferred)
496         sizePolicy.setHorizontalStretch(0)
497         sizePolicy.setVerticalStretch(0)
498         sizePolicy.setHeightForWidth(self.label_6.sizePolicy().hasHeightForWidth
                ())
499         self.label_6.setSizePolicy(sizePolicy)
500         font = QtGui.QFont()
501         font.setPointSize(7)
502         self.label_6.setFont(font)
503         self.label_6.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|
                QtCore.Qt.AlignVCenter)
504         self.label_6.setWordWrap(True)
505         self.label_6.setObjectName(_fromUtf8("label_6"))
506         self.label_7 = QtGui.QLabel(self.frame_6)
507         self.label_7.setGeometry(QtCore.QRect(120, 50, 41, 16))
```

```
508         sizePolicy = QtGui.QSizePolicy(QtGui.QSizePolicy.Preferred, QtGui.
                QSizePolicy.Preferred)
509         sizePolicy.setHorizontalStretch(0)
510         sizePolicy.setVerticalStretch(0)
511         sizePolicy.setHeightForWidth(self.label_7.sizePolicy().hasHeightForWidth
                ())
512         self.label_7.setSizePolicy(sizePolicy)
513         palette = QtGui.QPalette()
514         brush = QtGui.QBrush(QtGui.QColor(125, 125, 125))
515         brush.setStyle(QtCore.Qt.SolidPattern)
516         palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
517         brush = QtGui.QBrush(QtGui.QColor(125, 125, 125))
518         brush.setStyle(QtCore.Qt.SolidPattern)
519         palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText,
                brush)
520         brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
521         brush.setStyle(QtCore.Qt.SolidPattern)
522         palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText,
                brush)
523         self.label_7.setPalette(palette)
524         font = QtGui.QFont()
525         font.setPointSize(7)
526         self.label_7.setFont(font)
527         self.label_7.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|
                QtCore.Qt.AlignVCenter)
528         self.label_7.setWordWrap(True)
529         self.label_7.setObjectName(_fromUtf8("label_7"))
530         self.verticalLayout_6.addWidget(self.frame_5)
531         self.chkLoadFeedback = QtGui.QCheckBox(self.layoutWidget3)
532         font = QtGui.QFont()
533         font.setPointSize(10)
534         self.chkLoadFeedback.setFont(font)
535         self.chkLoadFeedback.setObjectName(_fromUtf8("chkLoadFeedback"))
536         self.verticalLayout_6.addWidget(self.chkLoadFeedback)
537         self.verticalLayout_6.setStretch(0, 2)
538         self.verticalLayout_6.setStretch(1, 1)
539         self.verticalLayout_5.addWidget(self.frame_3)
540         Main.setCentralWidget(self.centralwidget)
541         self.menubar = QtGui.QMenuBar(Main)
542         self.menubar.setGeometry(QtCore.QRect(0, 0, 1000, 21))
543         self.menubar.setObjectName(_fromUtf8("menubar"))
544         self.menuFile = QtGui.QMenu(self.menubar)
545         self.menuFile.setObjectName(_fromUtf8("menuFile"))
546         self.menuOptions = QtGui.QMenu(self.menubar)
547         self.menuOptions.setEnabled(True)
548         self.menuOptions.setObjectName(_fromUtf8("menuOptions"))
549         self.menuHelp = QtGui.QMenu(self.menubar)
550         self.menuHelp.setEnabled(True)
551         self.menuHelp.setObjectName(_fromUtf8("menuHelp"))
552         self.menuView = QtGui.QMenu(self.menubar)
553         self.menuView.setObjectName(_fromUtf8("menuView"))
554         Main.setMenuBar(self.menubar)
555         self.menuFileExit = QtGui.QAction(Main)
556         self.menuFileExit.setObjectName(_fromUtf8("menuFileExit"))
557         self.actionAbout = QtGui.QAction(Main)
558         self.actionAbout.setEnabled(False)
```

```
559         self.actionAbout.setObjectName(_fromUtf8("actionAbout"))
560         self.actionHelp = QtGui.QAction(Main)
561         self.actionHelp.setEnabled(False)
562         self.actionHelp.setObjectName(_fromUtf8("actionHelp"))
563         self.actionPreferences = QtGui.QAction(Main)
564         self.actionPreferences.setEnabled(False)
565         self.actionPreferences.setObjectName(_fromUtf8("actionPreferences"))
566         self.actionLive_View = QtGui.QAction(Main)
567         self.actionLive_View.setCheckable(True)
568         self.actionLive_View.setChecked(True)
569         self.actionLive_View.setEnabled(False)
570         self.actionLive_View.setObjectName(_fromUtf8("actionLive_View"))
571         self.menuFile.addAction(self.menuFileExit)
572         self.menuOptions.addAction(self.actionPreferences)
573         self.menuHelp.addAction(self.actionAbout)
574         self.menuHelp.addSeparator()
575         self.menuHelp.addAction(self.actionHelp)
576         self.menuView.addAction(self.actionLive_View)
577         self.menubar.addAction(self.menuFile.menuAction())
578         self.menubar.addAction(self.menuView.menuAction())
579         self.menubar.addAction(self.menuOptions.menuAction())
580         self.menubar.addAction(self.menuHelp.menuAction())
581 
582         self.retranslateUi(Main)
583         QtCore.QObject.connect(self.menuFileExit, QtCore.SIGNAL(_fromUtf8("
                triggered()")), Main.close)
584         QtCore.QObject.connect(self.btnConsoleClear, QtCore.SIGNAL(_fromUtf8("
                clicked()")), self.txtConsole.clear)
585         QtCore.QMetaObject.connectSlotsByName(Main)
586 
587     def retranslateUi(self, Main):
588         Main.setWindowTitle(_translate("Main", "Robot Control", None))
589         self.label.setText(_translate("Main", "Remote Robot Control", None))
590         self.lblTblSensors_2.setText(_translate("Main", "Console", None))
591         self.btnConsoleClear.setText(_translate("Main", "Clear", None))
592         self.lblLiveView.setText(_translate("Main", "Live View", None))
593         self.lblTblSensors.setText(_translate("Main", "Sensors", None))
594         self.btnTblSensorsLink.setText(_translate("Main", "Link", None))
595         self.btnTblSensorsUnlink.setText(_translate("Main", "Unlink", None))
596         self.lblTblServos.setText(_translate("Main", "Servos", None))
597         self.btnTblServosGroup.setText(_translate("Main", "Group", None))
598         self.btnTblServosUngroup.setText(_translate("Main", "Ungroup", None))
599         self.btnTblServosSetAngle.setText(_translate("Main", "Set Angle", None))
600         self.btnTblServosSetWheels.setText(_translate("Main", "Set Wheels", None)
                )
601         self.btnTblServosWiggle.setText(_translate("Main", "Wiggle", None))
602         self.lblOptions.setText(_translate("Main", "Options", None))
603         self.label_4.setText(_translate("Main", "Comm. frequency", None))
604         self.label_5.setText(_translate("Main", "Hz", None))
605         self.label_2.setText(_translate("Main", "In autonomous mode, the robot
                runs by itself, making decisions depending on its surroundings.",
                None))
606         self.chkAutonomous.setText(_translate("Main", "Autonomous Mode", None))
607         self.label_3.setText(_translate("Main", "<html><head/><body><p>Allows
                control of the robot wheels using the arrow keys.</p></body></html>",
                 None))
```

```
608         self.chkWheelsEnabled.setText(_translate("Main", "Enable Wheel Controls",
                None))
609         self.label_6.setText(_translate("Main", "<html><head/><body><p>Speed:</p
                ></body></html>", None))
610         self.label_7.setText(_translate("Main", "<html><head/><body><p>(0 to 10)
                </p></body></html>", None))
611         self.chkLoadFeedback.setText(_translate("Main", "Return Load Feedback",
                None))
612         self.menuFile.setTitle(_translate("Main", "File", None))
613         self.menuOptions.setTitle(_translate("Main", "Settings", None))
614         self.menuHelp.setTitle(_translate("Main", "Help", None))
615         self.menuView.setTitle(_translate("Main", "View", None))
616         self.menuFileExit.setText(_translate("Main", "Exit", None))
617         self.actionAbout.setText(_translate("Main", "About", None))
618         self.actionHelp.setText(_translate("Main", "Help", None))
619         self.actionPreferences.setText(_translate("Main", "Preferences...", None)
                )
620         self.actionLive_View.setText(_translate("Main", "Live View", None))
621
622
623 if __name__ == "__main__":
624     import sys
625     app = QtGui.QApplication(sys.argv)
626     Main = QtGui.QMainWindow()
627     ui = Ui_Main()
628     ui.setupUi(Main)
629     Main.show()
630     sys.exit(app.exec_())
```

I.17 et/guiExtension.py

```
1  '''
2  Created on 20. feb. 2013
3
4  @author: Eivind
5  '''
6  from PyQt4 import QtCore, QtGui #@UnusedImport
7  from PyQt4.QtCore import Qt, QEvent, QObject  #@UnusedImport
8  from PyQt4.QtGui import QTableWidget, QTableWidgetItem #@UnusedImport
9  from guiDesign import Ui_Main
10 from ESensor import ESensor
11 from EServo import EServo
12 from EKey import EKey #@UnusedImport
13 from debug import * #@UnusedWildImport
14
15
16
17 class GUIExtension(QtGui.QMainWindow):
18
19     # ID:Row lookups
20     __sensorRowLookup = {}
21     __servoRowLookup = {}
22
23     # Row:ID lookups
24     __sensorIDLookup = {}
25     __servoIDLookup = {}
```

```
26
27
28      # Constructor
29      def __init__(self, parent=None):
30          QtGui.QWidget.__init__(self, parent)
31          self.ui = Ui_Main()
32          self.ui.setupUi(self)
33          self.__initTables()
34          self.__createSignalSlots()
35
36
37
38
39
40      #
41      #
42      #    TODO: Prevent user edit in tables, set in code or design?
43      #
44      #
45
46
47      #
48      #
49      #    TODO: Use the Enum approach other places where possible (see
            __initTables)
50      #
51      #
52
53
54      # Adds a new SENSOR to the GUI
55      def _addSensor(self, sensorID):
56          self.__addRow(self.ui.tblSensor, ESensor, self.__sensorRowLookup, self.
                __sensorIDLookup, sensorID)
57
58
59      # Adds a new SERVO to the GUI
60      def _addServo(self, servoID):
61          self.__addRow(self.ui.tblServo, EServo, self.__servoRowLookup, self.
                __servoIDLookup, servoID)
62
63
64
65
66
67
68      def _setSensorData(self, sensorID, col, value):
69          if sensorID in self.__sensorRowLookup:
70              self.ui.tblSensor.item(self.__sensorRowLookup[sensorID], col).setText
                    (str(value))
71
72
73      def _setServoData(self, servoID, col, value):
74          if servoID in self.__servoRowLookup:
75              self.ui.tblServo.item(self.__servoRowLookup[servoID], col).setText(
                    str(value))
76
```

```
77
78
79
80      def keyPressEvent(self, event):
81          if not event.isAutoRepeat():
82              key = EKey.fromQtKey(event.key())
83              Debug.GUIKeyPressed(key)
84              if key != EKey.NoKey: self._abstractKeyPressed(key)
85          return QtGui.QMainWindow.keyPressEvent(self, event)
86
87      def keyReleaseEvent(self, event):
88          if not event.isAutoRepeat():
89              key = EKey.fromQtKey(event.key())
90              Debug.GUIKeyReleased(key)
91              if key != EKey.NoKey: self._abstractKeyReleased(key)
92          return QtGui.QMainWindow.keyReleaseEvent(self, event)
93
94
95      # Property - WheelControlsEnabled
96      def WheelControlsEnabled(): #@NoSelf
97          def fget(self): return self.ui.chkWheelsEnabled.isChecked()
98          return locals()
99      WheelControlsEnabled = property(**WheelControlsEnabled())
100
101
102
103     def selectedSensorID(self):
104         return self.__tryGetSelectedLookupID(self.ui.tblSensor, self.
                __sensorIDLookup)
105         #
106         #    TODO: Support for multiple indexes?
107         #
108
109     def selectedServoID(self):
110         return self.__tryGetSelectedLookupID(self.ui.tblServo, self.
                __servoIDLookup)
111         #
112         #    TODO: Support for multiple indexes?
113         #
114
115
116
117     #-------------------------------------------------------------------------
118     #    ABSTRACT METHODS - Override in subclass
119     #-------------------------------------------------------------------------
120
121
122     # Determines what happens on key pressed
123     def _abstractKeyPressed(self, key):
124         raise NotImplementedError("_abstractKeyPressed")
125
126     # Determines what happens on key released
127     def _abstractKeyReleased(self, key):
128         raise NotImplementedError("_abstractKeyReleased")
129
130
```

```
131
132
133
134     #-------------------------------------------------------------------------
135     #    PRIVATE
136     #-------------------------------------------------------------------------
137
138
139     def __tryGetSelectedLookupID(self, table, lookup):
140         indexes = table.selectedIndexes()
141         if len(indexes) > 0:
142             index = indexes[0].row()
143             if index in lookup.keys():
144                 return lookup[index]
145             else:
146                 return -1
147         else:
148             return -1
149
150
151     # Initialize tables
152     def __initTables(self):
153         self.__initTableColumns(self.ui.tblSensor, ESensor)
154         self.__initTableColumns(self.ui.tblServo, EServo)
155
156
157     # Initialize table columns
158     def __initTableColumns(self, tbl, enum):
159         tbl.setColumnCount(enum.COUNT)
160         tbl.setHorizontalHeaderLabels(enum.NAMES)
161         for col in range(0, enum.COUNT):
162             tbl.setColumnWidth(col, enum.SIZES[col])
163         tbl.horizontalHeaderItem(enum.Name).setTextAlignment(Qt.AlignLeft)
164
165
166     # Add a table row with cells
167     def __addRow(self, tbl, enum, rowLookup, IDLookup, ID):
168
169         # Insert new row
170         row = tbl.rowCount()
171         tbl.setRowCount(row + 1)
172         rowLookup[ID] = row
173         IDLookup[row] = ID
174
175         # Create cells
176         for col in range(0, enum.COUNT):
177             cell = QTableWidgetItem()
178             cell.setTextAlignment(Qt.AlignCenter)
179             tbl.setItem(row, col, cell)
180
181         # Name cell
182         name = enum.ENUM_NAME + " " + str(ID)
183         cell = tbl.item(row, enum.Name)
184         cell.setText(name)
185         cell.setTextAlignment(Qt.AlignLeft + Qt.AlignVCenter)
186         Debug.GUIAddedItem(name)
```

```
187
188
189     # Creates all signals and slots for GUI events
190     def __createSignalSlots(self):
191         CLICK = QtCore.SIGNAL("clicked()")
192         CHECK = QtCore.SIGNAL("stateChanged(int)")
193         TEXT_ENTER = QtCore.SIGNAL("returnPressed()")
194
195         # Buttons
196         QObject.connect(self.ui.btnTblSensorsLink, CLICK, self._btnLinkSlot)
197         QObject.connect(self.ui.btnTblSensorsUnlink, CLICK, self._btnUnlinkSlot)
198         QObject.connect(self.ui.btnTblServosGroup, CLICK, self._btnGroupSlot)
199         QObject.connect(self.ui.btnTblServosUngroup, CLICK, self._btnUngroupSlot)
200         QObject.connect(self.ui.btnTblServosSetAngle, CLICK, self.
                _btnSetAngleSlot)
201         QObject.connect(self.ui.btnTblServosSetWheels, CLICK, self.
                _btnSetWheelsSlot)
202         QObject.connect(self.ui.btnTblServosWiggle, CLICK, self._btnWiggleSlot)
203
204         # Check Boxes
205         QObject.connect(self.ui.chkAutonomous, CHECK, self._chkAutonomousSlot)
206         QObject.connect(self.ui.chkLoadFeedback, CHECK, self._chkLoadFeedback)
207
208         # Text Boxes
209         QObject.connect(self.ui.txtWheelVelocity, TEXT_ENTER, self.
                _txtWheelVelocitySlot)
210
211
212     # Button: Link
213     def _btnLinkSlot(self):
214         print "Button not implemented"
215
216     # Button: Unlink
217     def _btnUnlinkSlot(self):
218         print "Button not implemented"
219
220     # Button: Group
221     def _btnGroupSlot(self):
222         print "Button not implemented"
223
224     # Button: UNgroup
225     def _btnUngroupSlot(self):
226         print "Button not implemented"
227
228     # Button: SetAngle
229     def _btnSetAngleSlot(self):
230         print "Button not implemented"
231
232     # Button: SetWheels
233     def _btnSetWheelsSlot(self):
234         print "Button not implemented"
235
236     # Button: WIGGLE
237     def _btnWiggleSlot(self):
238         print "Button not implemented"
239
```

```
240     # Check Box: Autonomous
241     def _chkAutonomousSlot(self):
242         print "Check Box not implemented"
243 
244     # Check Box: Load Feedback
245     def _chkLoadFeedback(self):
246         print "Check Box not implemented"
247 
248     # Text Box: WheelVelocity
249     def _txtWheelVelocitySlot(self):
250         print "Button not implemented"
251 
252 
253 
254 
255     '''
256     TBL.selectedIndexes()
257     TBL.selectedItems()
258     TBL.selectedRanges()
259     TBL.sortByColumn(int)
260     '''
```

I.18 et/guiMain.py

```
1  '''
2  Created on 13. mars 2013
3  
4  @author: Eivind
5  '''
6  import sys
7  from PyQt4 import QtGui
8  from guiWrapper import GUIWrapper
9  from pydoc import gui
10 
11 NET_ENABLED = False
12 
13 if __name__ == '__main__':
14 
15     # Create QtGUI Application
16     app = QtGui.QApplication(sys.argv)
17     gui = GUIWrapper(None)
18     gui.show()
19     gui.registerServos(10)
20     gui.registerSensors(3)
21 
22 
23     # Launch controller and application
24     result = app.exec_()
25 
26     # Exit application
27     sys.exit(result)
```

I.19 et/guiWrapper.py

```
1  '''
2  Created on 18. feb. 2013
3  @author: Eivind
```

```
4  '''
5  from PyQt4 import QtCore, QtGui #@UnusedImport
6  from PyQt4.QtGui import QInputDialog
7  from guiExtension import GUIExtension
8  from EServo import EServo #@UnusedImport
9  from ESensor import ESensor
10 from debug import Debug
11 from msilib import Control
12
13
14 class GUIWrapper(GUIExtension):
15
16     # Controller
17     __controller = None
18
19     # Pending Group/UnGroup/Wheel IDs
20     __pendingIDs = [-1, -1, -1]
21
22
23     # Constructor
24     def __init__(self, controller, parent=None):
25         super(GUIWrapper, self).__init__()
26         self.__controller = controller
27
28
29     # Register SERVO count
30     def registerServos(self, servoCount):
31         for i in range(self.ui.tblServo.rowCount(), servoCount + 1):
32             self.addServo(i)
33
34
35     # Register Sensor count
36     def registerSensors(self, sensorCount):
37         for i in range(self.ui.tblSensor.rowCount(), sensorCount + 1):
38             self.addSensor(i)
39
40
41     # Add a Sensor to the GUI
42     def addSensor(self, sensorID):
43         self._addSensor(sensorID)
44
45
46     # Add a SERVO to the GUI
47     def addServo(self, servoID):
48         self._addServo(servoID)
49
50
51     # Update all data for the given sensor
52     def updateSensor(self, sensor):
53         Debug.GUISensorDataUpdate(sensor)
54         self._setSensorData(sensor.ID, ESensor.Angle, sensor.realValue)
55         self._setSensorData(sensor.ID, ESensor.Raw, sensor.rawValue)
56
57
58     # Update a value for the given sensor
59     def updateServo(self, servoID, index, value):
```

```
60          self._setServoData(servoID, index, value)
61
62
63      # Write text to the console
64      def writeToConsole(self, text):
65          self.ui.txtConsole.append(text)
66
67
68
69
70      #--------------------------------------------------------------------------
71      #    ABSTRACT METHODS - Overridden from superclass
72      #--------------------------------------------------------------------------
73
74
75      # Register key press to control wheels and more
76      def _abstractKeyPressed(self, key):
77          if self.__controller != None and self.WheelControlsEnabled:
78              self.__controller.Wheels.keyPressed(key)
79
80
81      # Register key press to control wheels and more
82      def _abstractKeyReleased(self, key):
83          if self.__controller != None and self.WheelControlsEnabled:
84              self.__controller.Wheels.keyReleased(key)
85
86
87
88      #--------------------------------------------------------------------------
89      #    EVENT SLOTS - Overridden from superclass
90      #--------------------------------------------------------------------------
91
92
93      # Button: Link
94      def _btnLinkSlot(self):
95          self.__controller.Linker.link(self.selectedSensorID(),
96                                        self.selectedServoID())
97
98
99      # Button: Unlink
100     def _btnUnlinkSlot(self):
101         self.__controller.Linker.unlink(self.selectedSensorID(),
102                                         self.selectedServoID())
103
104
105     # Button: Group
106     def _btnGroupSlot(self):
107         ok, ID_1, ID_2 = self.__pendID(0, self.selectedServoID())
108         if ok: self.__controller.Linker.group(ID_1, ID_2)
109
110
111     # Button: UNgroup
112     def _btnUngroupSlot(self):
113         ok, ID_1, ID_2 = self.__pendID(1, self.selectedServoID())
114         if ok: self.__controller.Linker.ungroup(ID_1, ID_2)
115
```

```
116
117     # Button: SetAngle
118     def _btnSetAngleSlot(self):
119         selected = self.selectedServoID()
120         if selected > -1:
121             text, ok = QInputDialog.getText(self, "Set Angle", "Aangle in degrees
                    :")
122             value, numeric = text.toFloat()
123             if ok and numeric:
124                 self.__controller._controlServoAngle(selected, value)
125
126     #
127     #
128     #    TODO: Feedback or status indicators to aid with link/group/wheels?
129     #
130     #
131
132
133     # Button: SetWheels
134     def _btnSetWheelsSlot(self):
135         ok, ID_1, ID_2 = self.__pendID(2, self.selectedServoID())
136         if ok: self.__controller.Wheels.registerServos(ID_1, ID_2)
137
138
139     # Button: WIGGLE
140     def _btnWiggleSlot(self):
141         servoID = self.selectedServoID()
142         if servoID > -1: self.__controller.wiggleServo(servoID)
143
144
145     # Check Box: Autonomous
146     def _chkAutonomousSlot(self):
147         if self.ui.chkAutonomous.checkState() == 2:
148             self.__controller.toggleAutonomous(1)
149         else:
150             self.__controller.toggleAutonomous(0)
151
152
153     # Check Box: Load Feedback
154     def _chkLoadFeedback(self):
155         print "Check Box not implemented"
156
157
158     # Text Box: WheelVelocity
159     def _txtWheelVelocitySlot(self):
160         value, numeric = self.ui.txtWheelVelocity.text().toFloat()
161         if numeric:
162             if value >= 0 and value <= 10:
163                 self.__controller.Wheels.Speed = value
164
165
166
167
168
169     #--------------------------------------------------------------------------
170     #    PRIVATE
```

```
171     #-------------------------------------------------------------------------
172
173
174     # Pend an ID to list of pending IDs
175     # Returns True if set of IDs are complete, and returns IDs
176     def __pendID(self, pendIndex, newID):
177         pending = self.__pendingIDs[pendIndex]
178         if newID > -1:
179             if pending > -1 and pending <> newID:
180                 self.__pendingIDs[pendIndex] = -1
181                 return (True, pending, newID)
182             else:
183                 self.__pendingIDs[pendIndex] = newID
184                 return (False, pending, newID)
185         else:
186             return (False, pending, newID)
```

I.20 et/linker.py

```
1  '''
2  Created on 13. mars 2013
3  @author: Eivind
4  '''
5
6  from dict import ArrayDictionary
7  from debug import * #@UnusedWildImport
8
9
10
11 class Linker(object):
12
13     # SensorID:ServoID-Array dictionary
14     __links = ArrayDictionary()
15
16     # ServoID:ServoID-Array dictionary
17     __groups = ArrayDictionary()
18
19
20     # Constructor
21     def __init__(self):
22         pass
23
24
25     # Link a Sensor to SERVO
26     def link(self, sensorID, servoID):
27         if self.__validIDs(sensorID, servoID):
28             Debug.LinkerLink(sensorID, servoID)
29             self.__links.add(sensorID, servoID)
30
31
32     # Unlink a Sensor from SERVO
33     def unlink(self, sensorID, servoID):
34         if self.__validIDs(sensorID, servoID):
35             Debug.LinkerUnlink(sensorID, servoID)
36             self.__links.remove(sensorID, servoID)
37
```

```
38
39      # Group two SERVOS. The first SERVO ID is the group ID
40      def group(self, groupServoID, servoID):
41          if self.__validIDs(groupServoID, servoID):
42              Debug.LinkerGroup(groupServoID, servoID)
43              self.__groups.addDuoKeyItems(groupServoID, servoID)
44
45
46      # UN-group two SERVOS
47      def ungroup(self, groupServoID, servoID):
48          if self.__validIDs(groupServoID, servoID):
49              Debug.LinkerUngroup(groupServoID, servoID)
50              self.__groups.removeDuoKeyItems(groupServoID, servoID)
51
52
53      # Clear all links and groups
54      def clear(self):
55          self.__links.clear()
56          self.__groups.clear()
57
58
59      # Gets an array of SERVOS linked to given Sensor
60      def getServoLinks(self, sensorID, groups = True):
61          linkedServos = self.__links.tryGetItems(sensorID)
62          if groups:
63              return self.getServoGroups(linkedServos)
64          else:
65              return linkedServos
66
67
68      # Gets an array of SERVOS with given group ID
69      def getServoGroup(self, groupID):
70          return self.__groups.tryGetItems(groupID, [groupID])
71
72
73      # Gets an accumulated array of SERVOS with given group IDs
74      def getServoGroups(self, groupIDs):
75          return self.__groups.tryGetAllItems(groupIDs, groupIDs)
76
77
78      # Checks if two IDs are valid
79      def __validIDs(self, ID_1, ID_2):
80          return ID_1 > -1 and ID_2 > -1
```

I.21 et/sensor.py

```
1  '''
2  Created on 3. mars 2013
3  @author: Eivind, Erik
4  '''
5
6  from debug import * #@UnusedWildImport
7
8
9  # High amplifier (>1.0) gives larger real value output
10 DEFAULT_AMPLIFIER = 1.0
```

```
11
12  # Minimum RAW angle change to raise event
13  # Variations in raw sensor measurements +/- 2~
14  DEFAULT_THRESHOLD = 0.0 # 2.0
15
16
17  class Sensor(object):
18
19      # Sensor ID
20      __ID = 0
21
22      # Raw value measurements
23      __rawValue = 0.0
24      __rawValuePrevious = 0.0
25
26      # Amplifier: Real value modifier
27      __amplifier = DEFAULT_AMPLIFIER
28
29      # Threshold: Minimum raw value change
30      __threshold = DEFAULT_THRESHOLD
31
32
33      # Constructor
34      def __init__(self, \
35                   sensorID, \
36                   amplifier = DEFAULT_AMPLIFIER, \
37                   threshold = DEFAULT_THRESHOLD):
38          self.__ID = sensorID
39          self.__amplifier = amplifier
40          self.__threshold = threshold
41
42
43      # Property: ID
44      def ID(): #@NoSelf
45          def fget(self): return self.__ID
46          return locals()
47      ID = property(**ID())
48
49
50      # Property: Raw Value
51      def rawValue(): #@NoSelf
52          def fget(self): return self.__rawValue
53          return locals()
54      rawValue = property(**rawValue())
55
56
57      # Property: Raw Value Previous
58      def rawValuePrevious(): #@NoSelf
59          def fget(self): return self.__rawValuePrevious
60          return locals()
61      rawValuePrevious = property(**rawValuePrevious())
62
63
64      # Property: Real Value
65      def realValue(): #@NoSelf
66          def fget(self): return self.__rawToReal()
```

```
67          return locals()
68      realValue = property(**realValue())
69
70
71      # Property: Amplifier
72      def amplifier(): #@NoSelf
73          def fget(self): return self.__amplifier
74          def fset(self, value): self.__amplifier = value
75          return locals()
76      amplifier = property(**amplifier())
77
78
79      # Property: Threshold
80      def threshold(): #@NoSelf
81          def fget(self): return self.__threshold
82          def fset(self, value): self.__threshold = value
83          return locals()
84      threshold = property(**threshold())
85
86
87      # Reads and returns new raw value
88      def update(self):
89          self.__rawValuePrevious = self.__rawValue
90          self.__rawValue = self._abstractReadRawValue()
91          return self.rawValue
92
93
94      # Gets the difference between current and previous raw value
95      def getRawValueDiff(self):
96          return self.rawValue - self.rawValuePrevious
97
98
99      # Gets the absolute difference between current and previous raw value
100     def getRawValueAbsDiff(self):
101         return abs(self.getRawValueDiff())
102
103
104     # True if raw value change is >= threshold value
105     def hasRawValueChanged(self):
106         return self.getRawValueAbsDiff() > self.threshold
107
108
109
110
111
112
113     #--------------------------------------------------------------------------
114     #    ABSTRACT METHODS - Override in subclass
115     #--------------------------------------------------------------------------
116
117
118     # Determines how raw values are acquired
119     def _abstractReadRawValue(self):
120         raise NotImplementedError("_abstractReadRawValue")
121
122
```

```
123     # Determines how raw value is converted to real value
124     def _abstractRawToReal(self, rawValue):
125         raise NotImplementedError("_abstractRawToReal")
126 
127 
128 
129 
130 
131 
132     #---------------------------------------------------------------------------
133     #    PRIVATE
134     #---------------------------------------------------------------------------
135 
136 
137     # Converts raw value to real value (via abstract and amplifier)
138     def __rawToReal(self):
139         realValue = self._abstractRawToReal(self.rawValue) * self.amplifier
140         Debug.SensorRawToReal(self.rawValue, realValue)
141         return realValue
```

I.22 et/sensorWatcher.py

```
1  '''
2  Created on 27. feb. 2013
3  @author: Eivind, Erik
4  '''
5  import time
6  from threading import Thread
7  from pymcu import mcuModule
8  from angleSensor import BendSensor
9  from debug import * #@UnusedWildImport
10 from dict import IDDictionary
11 
12 # Delay between each sensor check
13 DEFAULT_INTERVAL = 0.05
14 
15 # Baud Rate
16 DEFAULT_BAUD_RATE = 12
17 
18 
19 class SensorWatcher(Thread, IDDictionary):
20 
21     # Main __controller
22     __controller = None
23 
24     # pyMCU Controller
25     __mcu = None
26 
27     # Thread runs while True
28     __enabled = False
29 
30     # The thread sleep __interval
31     __interval = 0
32 
33 
34     # Constructor
```

```
35      def __init__(self, controller, interval = DEFAULT_INTERVAL):
36          super(SensorWatcher, self).__init__()
37          self.__controller = controller
38          self.__interval = interval
39          self.__createMCU()
40
41
42      # Manually stop thread from outside
43      def stop(self):
44          self.__enabled = False
45
46
47      # Thread Run
48      def run(self):
49          self.__enabled = True
50
51          # Continuously check for sensor updates
52          while(self.__enabled):
53              self.__checkForSensorChanges()
54              time.sleep(self.__interval)
55
56
57      # Adds a new BendSensor to watch
58      def addBendSensor(self, sensorID):
59          sensor = BendSensor(sensorID, self.__mcu)
60          self.add(sensor)
61
62
63      # Adds a new BendSensor to watch (with TurnSensor calibration)
64      def addTurnSensor(self, sensorID):
65          sensor = BendSensor(sensorID, self.__mcu)
66          sensor.calibrate(1.0, 0.0, 600, 1000)
67          self.add(sensor)
68
69
70      # Sets the watcher update __interval
71      def setInterval(self, interval):
72          self.__interval = interval
73
74
75      # Sets the amplifier of given sensor
76      def setAmplifier(self, sensorID, amplifier):
77          sensor = self.__tryGetSensor(sensorID)
78          if sensor != None: sensor.amplifier = amplifier
79
80
81      # Sets the threshold of given sensor
82      def setThreshold(self, sensorID, threshold):
83          sensor = self.__tryGetSensor(sensorID)
84          if sensor != None: sensor.threshold = threshold
85
86
87
88
89
90      #--------------------------------------------------------------------------
```

```
    #    PRIVATE
    #--------------------------------------------------------------------------


    # Checks for any sensor changes, triggers changed event
    def __checkForSensorChanges(self):
        for sensor in self.Items: #.values():
            sensor.update()
            if sensor.hasRawValueChanged:
                self.__sensorDataChanged(sensor)


    # EVENT: Sensor Data Changed
    def __sensorDataChanged(self, sensor):

        # Notify Controller
        self.__controller.sensorDataChanged(sensor)


    # Create the MCU object
    def __createMCU(self):
        try:
            self.__mcu = mcuModule()
            self.__mcu.mcuSetBaudRate(DEFAULT_BAUD_RATE)
        except ValueError as e:
            print e
```

I.23 et/servo.py

```
'''
Created on 20. mars 2013
@author: Eivind
'''

from dict import IDDictionary


class Servo(object):

    # Controller
    __controller = None

    # Properties
    __ID = -1
    __Inverted = False


    # Constructor
    def __init__(self, ID, controller = None):
        self.__ID = ID
        self.__controller = controller


    # Property - ID
    def ID(): #@NoSelf
        def fget(self): return self.__ID
```

```
28          return locals()
29      ID = property(**ID())
30
31
32      # Property - Inverted
33      def Inverted(): #@NoSelf
34          def fget(self): return self.__Inverted
35          def fset(self, value): self.__Inverted = value
36          return locals()
37      Inverted = property(**Inverted())
38
39
40      # Sets the angle of this SERVO via the Controller
41      def controlAngle(self, angle):
42          if self.__controller != None:
43              self.__controller._controlServoAngle(self.ID, self.__outputAngle(
                    angle))
44
45
46      def __outputAngle(self, angle):
47          if self.Inverted:
48              return -angle
49          else:
50              return angle
51
52
53
54  class ServoGroup(IDDictionary):
55
56
57      # Constructor
58      def __init__(self):
59          super(ServoGroup, self).__init__()
60
61
62      # Set All Angles
63      def controlAngle(self, angle):
64          for s in self.Items: s.controlAngle(angle)
```

I.24 et/wheels.py

```
1   '''
2   Created on 19. mars 2013
3   @author: Eivind
4   '''
5
6   from EKey import EKey
7   from debug import * #@UnusedWildImport
8
9
10  '''
11  # Turning speed
12  TURN_SPEED = 5
13  '''
14
15  # Drive speed
```

```
DRIVE_SPEED = 5

# Max Drive and Turn speed
MAX_SPEED = 5


class Wheels(object):

    # Controller object
    __controller = None

    # Wheels' SERVO IDs
    __wheels_L = -1
    __wheels_R = -1

    # Current Directions
    __U = 0 # Up
    __D = 0 # Down
    __L = 0 # Left
    __R = 0 # Right

    # Speed
    __Speed = DRIVE_SPEED


    # Constructor
    def __init__(self, controller):
        self.__controller = controller


    # Property - Speed
    def Speed(): #@NoSelf
        def fget(self): return self.__Speed
        def fset(self, value):
            Debug.WheelsSpeed(value)
            self.__Speed = value
            self.__updateWheelSpeeds()
        return locals()
    Speed = property(**Speed())


    # Register a SERVO for left and right wheels
    def registerServos(self, servoID_L, servoID_R):
        c = self.__controller
        Debug.WheelsSet(servoID_L, servoID_R)

        # Disable wheel mode for any existing wheels
        for s in c.Linker.getServoGroup(self.__wheels_L): c.setServoMode(s, 0)
        for s in c.Linker.getServoGroup(self.__wheels_R): c.setServoMode(s, 0)

        # Set new wheel IDs
        self.__wheels_L = servoID_L
        self.__wheels_R = servoID_R

        # Enable wheel mode for new wheels
        for s in c.Linker.getServoGroup(servoID_L): c.setServoMode(s, 1)
```

```
72          for s in c.Linker.getServoGroup(servoID_R): c.setServoMode(s, 1)
73
74
75      # Checks if wheels are registered
76      def hasServos(self):
77          return self.__wheels_L >= 0 and self.__wheels_R >= 0
78
79
80      # EVENT - Key Pressed (EKey)
81      def keyPressed(self, key):
82          self.__setDirectionValue(key, 1)
83          self.__updateWheelSpeeds()
84
85
86      # EVENT - Key Released (EKey)
87      def keyReleased(self, key):
88          self.__setDirectionValue(key, 0)
89          self.__updateWheelSpeeds()
90
91
92      # Stop all wheels
93      def stop(self):
94          self.__setDirectionValue(EKey.Stop, 0)
95          self.__updateWheelSpeeds()
96
97
98      # Start moving forwards
99      def forwards(self):
100         self.move(EKey.Up)
101
102
103     # Start moving backwards
104     def backwards(self):
105         self.move(EKey.Down)
106
107
108     # Start turning left
109     def left(self):
110         self.move(EKey.Left)
111
112
113     # Start turning right
114     def right(self):
115         self.move(EKey.Right)
116
117
118     # Start driving/turning by the given key
119     def move(self, key):
120         self.__setDirectionValue(EKey.Stop, 0)
121         self.__setDirectionValue(key, 1)
122         self.__updateWheelSpeeds()
123
124
125
126
127
```

```
128     #-------------------------------------------------------------------------
129     #    PRIVATE
130     #-------------------------------------------------------------------------
131
132
133     # Sets a direction value depending on key type
134     def __setDirectionValue(self, key, value):
135         if   key == EKey.Up:    self.__U = value
136         elif key == EKey.Down:  self.__D = value
137         elif key == EKey.Left:  self.__L = value
138         elif key == EKey.Right: self.__R = value
139         elif key == EKey.Stop:
140             self.__U = self.__D = self.__L = self.__R = 0
141
142
143     # Updates wheel speeds through the Controller
144     def __updateWheelSpeeds(self):
145         if not self.hasServos(): return
146
147         # Positive = Drive forwards
148         # Negative = Drive backwards
149         drive = (self.__U - self.__D) * self.__Speed #DRIVE_SPEED
150
151         # Positive = Turn left
152         # Negative = Turn right
153         turn = (self.__R - self.__L) * self.__Speed #TURN_SPEED
154
155         # Set drive and turn speeds
156         self.__setControllerSpeed(turn + drive, turn - drive)
157
158
159     # Sets the wheel speeds through the Controller
160     def __setControllerSpeed(self, left, right):
161         Debug.WheelsDrive(left, right)
162         self.__controller.setWheelServoSpeed(
163             self.__wheels_L,
164             self.__wheels_R,
165             self.__limitSpeed(left),
166             self.__limitSpeed(right))
167
168
169     # Reduces a speed to max speed (positive or negative)
170     def __limitSpeed(self, speed):
171         if speed > 0:
172             return min(speed, MAX_SPEED)
173         else:
174             return max(speed, -MAX_SPEED)
```

I.25 sensors/bendSensor.py

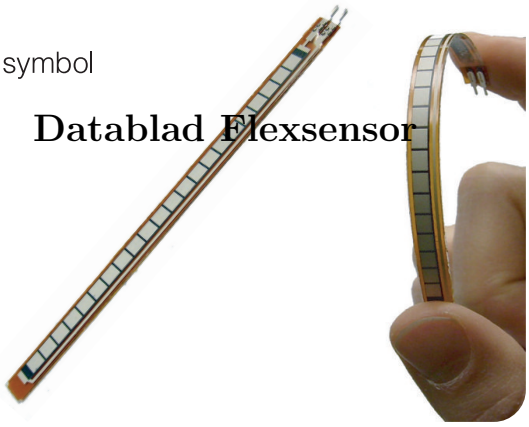
```
1 '''
2 Created on 3. mars 2013
3 @author: Eivind, Erik, Eirik
4 '''
5 from __future__ import division
6 from pymcu import mcuModule #@UnusedImport
```

```
7  from sensors import sensor
8  from sensors.sensor import Sensor
9  from debug.debug import * #@UnusedWildImport
10
11
12
13 # SENSOR bends max 90 degrees (safety)
14 # SERVO goes 300 degrees in angle mode
15 # rawAngle varies from 540 to 790 ~
16 # Sensor ID goes from 1 to 6 (physically on the board)
17 RAW_VALUE_MIN = 540
18 RAW_VALUE_MAX = 790
19 RAW_VALUE_90 = 690
20 RAW_VALUE_SPAN = RAW_VALUE_MAX - RAW_VALUE_MIN # 250
21 RAW_TO_REAL_FACTOR_90 = 90 / (RAW_VALUE_90 - RAW_VALUE_MIN)
22 RAW_TO_REAL_FACTOR = RAW_TO_REAL_FACTOR_90
23
24
25
26 class BendSensor(Sensor):
27
28     # pyMCU Controller
29     __mcu = None
30     __raw_value_min = RAW_VALUE_MIN
31     __raw_to_real_factor = RAW_TO_REAL_FACTOR
32
33
34     # Constructor
35     def __init__(self, sensorID, mcu, \
36                  amplifier = sensor.DEFAULT_AMPLIFIER, \
37                  threshold = sensor.DEFAULT_THRESHOLD):
38         super(BendSensor, self).__init__(sensorID, amplifier, threshold)
39         self.__mcu = mcu
40
41     # Property: raw_value_min
42     def raw_value_min(): #@NoSelf
43         def fget(self): return self.__raw_value_min
44         def fset(self, value): self.__raw_value_min = value
45         return locals()
46     raw_value_min = property(**raw_value_min())
47
48     # Property: raw_to_real_factor
49     def raw_to_real_factor(): #@NoSelf
50         def fget(self): return self.__raw_to_real_factor
51         def fset(self, value): self.__raw_to_real_factor = value
52         return locals()
53     raw_to_real_factor = property(**raw_to_real_factor())
54
55     # Calibration input
56     def calibrate(self,amplifier,threshold,raw_value_min,raw_value_90):
57         self.amplifier = amplifier
58         self.threshold = threshold
59         self.raw_to_real_factor = 90 / (raw_value_90 - raw_value_min)
60
61
62
```

```
63      #-------------------------------------------------------------------------
64      #    ABSTRACT METHODS - Overridden from superclass
65      #-------------------------------------------------------------------------
66
67
68      # Determines how raw values are acquired
69      def _abstractReadRawValue(self):
70          rawValue = self.__mcu.analogRead(self.ID)
71          if rawValue <> None:
72              return rawValue
73          else:
74              debugE("No raw value from sensor {0}!".format(self.ID))
75              return 0
76
77
78      # Determines how raw value is converted to real value
79      def _abstractRawToReal(self, rawValue):
80
81          # Finds zero-based raw angle (from 0 to N)
82          # Converts the raw angle to real angle
83          rawAngle_0 = max(0, rawValue - self.raw_value_min)
84          return rawAngle_0 * self.raw_to_real_factor
```

spectrasymbol

J Datablad Flexsensor



FLEX SENSOR FS

Features

- Angle Displacement Measurement
- Bends and Flexes physically with motion device
- Possible Uses
  - Robotics
  - Gaming (Virtual Motion)
  - Medical Devices
  - Computer Peripherals
  - Musical Instruments
  - Physical Therapy
- Simple Construction
- Low Profile

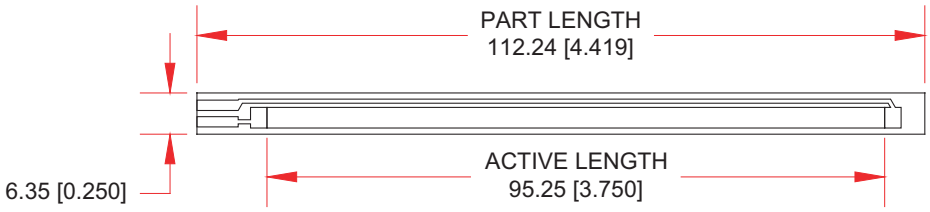
Mechanical Specifications

-Life Cycle: >1 million
-Height: ≤0.43mm (0.017")
-Temperature Range: -35°C to +80°C

Electrical Specifications

-Flat Resistance: 10K Ohms
-Resistance Tolerance: ±30%
-Bend Resistance Range: 60K to 110K Ohms
-Power Rating : 0.50 Watts continuous. 1 Watt Peak

Dimensional Diagram - Stock Flex Sensor

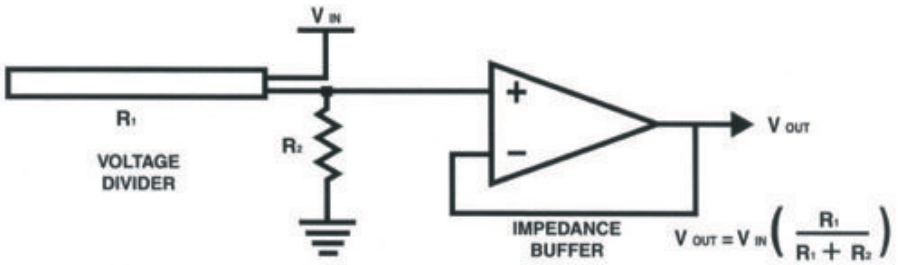


How to Order - Stock Flex Sensor

FS	–	L	–	0095	–	103	–	ST
Series		Model		Active Length		Resistance		Connectors
FS = Flex Sensor		L = Linear		0095 = 95.25mm		103 = 10 KOhms		ST = Solder Tab

How It Works

BASIC FLEX SENSOR CIRCUIT:

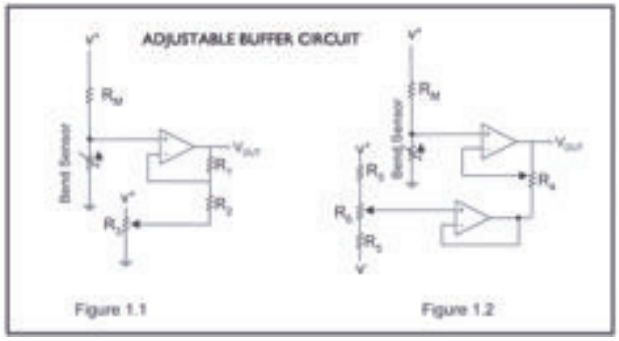


Following are notes from the ITP Flex Sensor Workshop

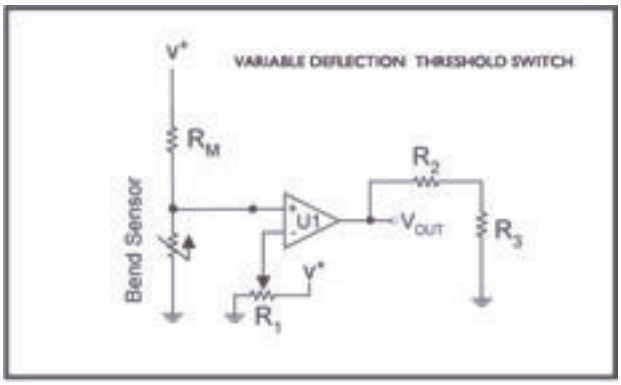
"The impedance buffer in the [Basic Flex Sensor Circuit] (above) is a single sided operational amplifier, used with these sensors because the low bias current of the op amp reduces errer due to source impedance of the flex sensor as voltage divider. Suggested op amps are the LM358 or LM324."

"You can also test your flex sensor using the simplest circut, and skip the op amp."

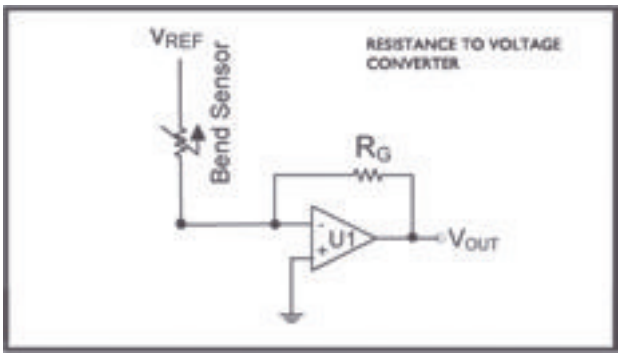
"Adjustable Buffer - a potentiometer can be added to the circuit to adjust the sensitivity range."



"Variable Deflection Threshold Switch - an op amp is used and outputs either high or low depending on the voltage of the inverting input. In this way you can use the flex sensor as a switch without going through a microcontroller."



"Resistance to Voltage Converter - use the sensor as the input of a resistance to voltage converter using a dual sided supply op-amp. A negative reference voltage will give a positive output. Should be used in situations when you want output at a low degree of bending."



K Datablad AX-12

User's Manual 2006-06-14

Contents

1. Dynamixel AX-12

1-1. Overview and Characteristics of AX-12

Dynamixel AX-12 The Dynamixel series robot actuator is a smart, modular actuator that incorporates a gear reducer, a precision DC motor and a control circuitry with networking functionality, all in a single package. Despite its compact size, it can produce high torque and is made with high quality materials to provide the necessary strength and structural resilience to withstand large external forces. It also has the ability to detect and act upon internal conditions such as changes in internal temperature or supply voltage. The Dynamixel series robot actuator has many advantages over similar products.

Precision Control Position and speed can be controlled with a resolution of 1024 steps.

Compliance Driving The degree of compliance can be adjusted and specified in controlling position.

Feedback Feedback for angular position, angular velocity, and load torque are available.

Alarm System The Dynamixel series robot actuator can alert the user when parameters deviate from user defined ranges (e.g. internal temperature, torque, voltage, etc) and can also handle the problem automatically (e.g. torque off)

Communication Wiring is easy with daisy chain connection, and it support communication speeds up to 1M BPS.

Distributed Control Position, velocity, compliance, and torque can be set with a single command packet, thus enabling the main processor to control many Dynamixel units even with very few resources.

Engineering Plastic The main body of the unit is made with high quality engineering plastic which enables it to handle high torque loads.

Axis Bearing A bearing is used at the final axis to ensure no efficiency degradation with high external loads.

Status LED The LED can indicate the error status to the user.

Frames A hinge frame and a side mount frame are included.

1-2. Main Specifications

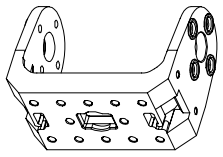
	AX-12	
Weight (g)	**55**	
Gear Reduction Ratio	**1/254**	
Input Voltage (V)	**at 7V**	**at 10V**
Final Max Holding Torque(kgf.cm)	**12**	**16.5**
Sec/60degree	**0.269**	**0.196**

Resolution	0.35°
Operating Angle	300°, Endless Turn
Voltage	7V~10V (Recommended voltage: 9.6V)
Max. Current	900mA
Operate Temperature	-5℃ ~ +85℃
Command Signal	Digital Packet
Protocol Type	Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
Link (Physical)	TTL Level Multi Drop (daisy chain type Connector)
ID	254 ID (0~253)
Communication Speed	7343bps ~ 1 Mbps
Feedback	Position, Temperature, Load, Input Voltage, etc.
Material	Engineering Plastic

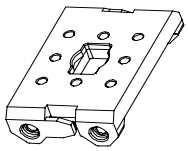
2. Dynamixel Operation

2-1. **Mechanical Assembly**

Frames Provided The two frames provided with AX-12 are shown below.

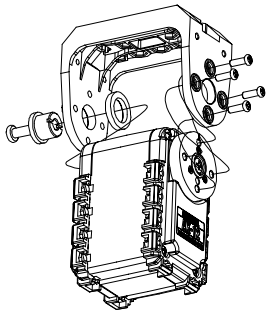


OF-12SH

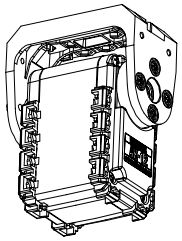


OF-12S

OF-12SH Installation The OF-12SH (hinge frame) can be installed on the AX-12 as the following.



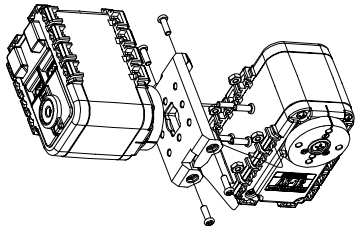
Exploded view



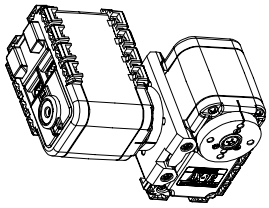
Assembled

OF-12S Installation The OF-12S (side mount frame) can be installed on the AX-12 as the following. The OF-12S can be mounted on any of the three faces (left, right, or under side) of the AX-12 body as needed.

Horn2Body

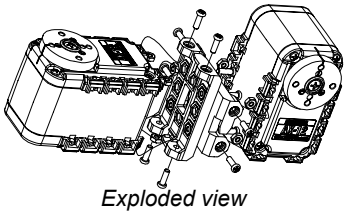


Exploded view

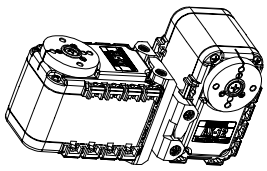


Assembled

Body2Body



Exploded view



Assembled

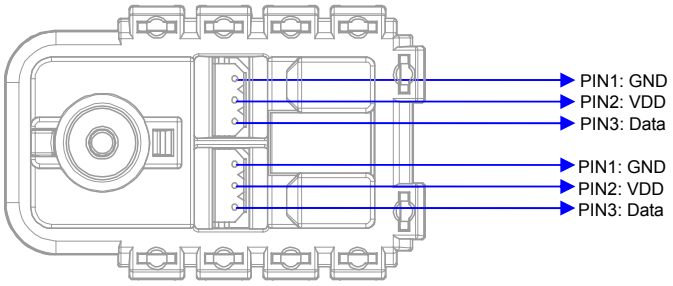
2-2 . Connector Assembly

Assemble the connectors as shown below. Attach the wires to the terminals using the correct crimping tool. If you do not have access to a crimping tool, solder the terminals to the wires to ensure that they do not become loose during operation.

2-3. Dynamixel Wiring

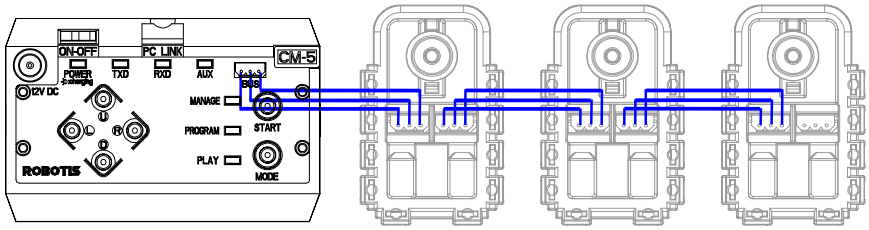
Pin Assignment

The connector pin assignments are as the following. The two connectors on the Dynamixel are connected pin to pin, thus the AX-12 can be operated with only one connector attached.



Wiring

Connect the AX-2 actuators pin to pin as shown below. Many AX-12 actuators can be controlled with a single bus in this manner.



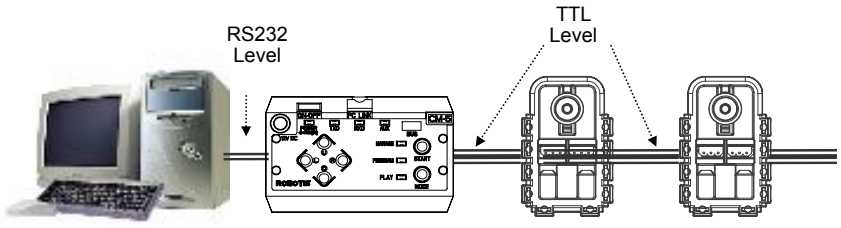
Control Box “CM-5”

Main Controller

To operate the Dynamixel actuators, the main controller must support TTL level half duplex UART. A proprietary controller can be used, but the use of the Dynamixel controller CM-5 is recommended.

PC LINK

A PC can be used to control the Dynamixel via the CM-5 controller.



PC CM-5 Dynamixels

bioloid

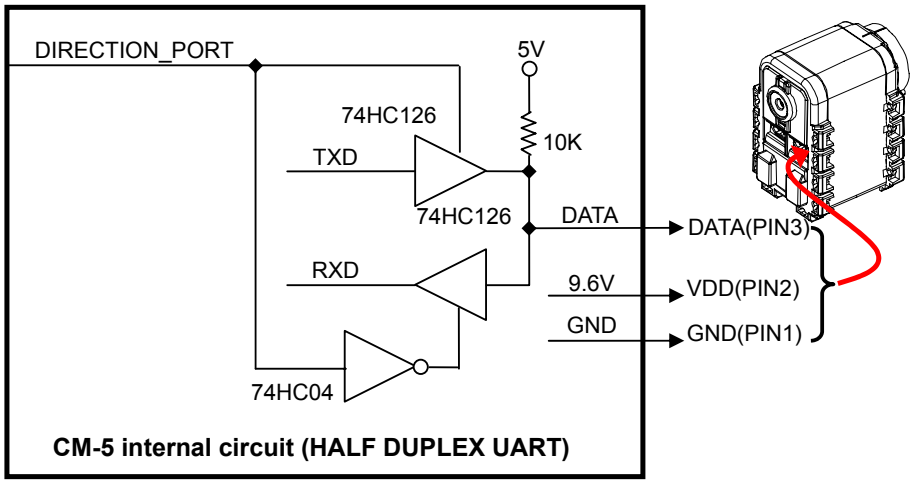
A robot can be built using only the CM-5 controller and a number of AX-12 actuators. An edutainment robotic kit named “Bioloid” is available which is based on the CM-5 controller and the AX-12 actuators.

An example of a robot built with Bioloid

For details, please refer to the Bioloid manual.

Connection to UART

To control the Dynamixel actuators, the main controller needs to convert its UART signals to the half duplex type. The recommended circuit diagram for this is shown below.



CM-5 internal circuit (HALF DUPLEX UART)

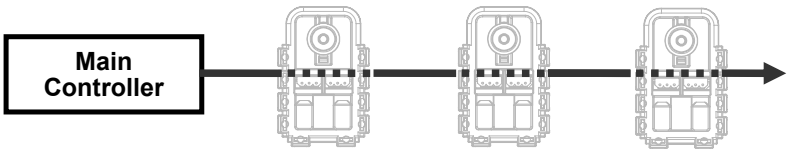
The power is supplied to the Dynamixel actuator from the main controller through Pin 1 and Pin 2 of the Molex3P connector. (The circuit shown above is presented only to explain the use of half duplex UART. The CM-5 controller already has the above circuitry built in, thus the Dynamixel actuators can be directly connected to it)

The direction of data signals on the TTL level TxD and RxD depends on the DIRECTION_PORT level as the following.

• When the DIRECTION_PORT level is High: the signal TxD is output as Data
• When the DIRECTION_PORT level is Low: the signal Data is input as RxD

Half Duplex UART

A multi-drop method of connecting multiple Dynamixel actuators to a single node is possible by using the half duplex UART. Thus a protocol that does not allow multiple transmissions at the same time should be maintained when controlling the Dynamixel actuators.



[Multi Drop Link]

Caution

Please ensure that the pin assignments are correct when connecting the Dynamixel actuators. Check the current consumption when powering on. The current consumption of a single Dynamixel actuator unit in standby mode should be no larger than 50mA

Connection Status Verification

When power is applied to the Dynamixel actuator, the LED blinks twice to confirm its connection.

Inspection

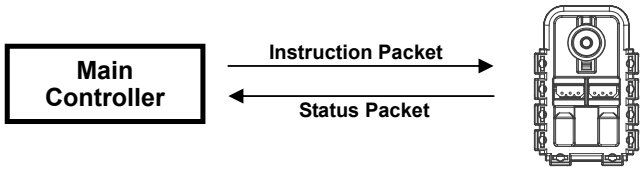
If the above operation was not successful, then check the connector pin assignment and the voltage/current limit of the power supply.

3. Communication Protocol

3-1. Communication Overview

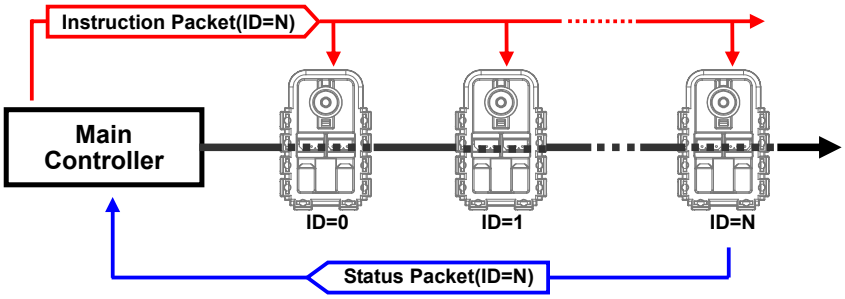
Packet

The main controller communicates with the Dynamixel units by sending and receiving data packets. There are two types of packets; the “Instruction Packet” (sent from the main controller to the Dynamixel actuators) and the “Status Packet” (sent from the Dynamixel actuators to the main controller.)



Communication

For the system connection below, if the main controller sends an instruction packet with the ID set to N, only the Dynamixel unit with this ID value will return its respective status packet and perform the required instruction.



Unique ID

If multiple Dynamixel units have the same ID value, multiple packets sent simultaneously collide, resulting in communication problems. Thus, it is imperative that no Dynamixel units share the same ID in a network node.

Protocol

The Dynamixel actuators communicate through asynchronous serial communication with 8 bit, 1 stop bit and no parity.

3-2. Instruction Packet

The Instruction Packet is the packet sent by the main controller to the Dynamixel units to send commands. The structure of the Instruction Packet is as the following.

Instruction Packet | OXFF | 0XFF | ID | LENGTH | INSTRUCTION | PARAMETER1 | PARAMETER N | CHECK SUM |

The meanings of each packet byte definition are as the following.

0XFF 0XFF The two 0XFF bytes indicate the start of an incoming packet.

ID The unique ID of a Dynamixel unit. There are 254 available ID values, ranging from 0X00 to 0XFD.

Broadcasting ID ID 0XFE is the Broadcasting ID which indicates all of the connected Dynamixel units. Packets sent with this ID apply to all Dynamixel units on the network. Thus packets sent with a broadcasting ID will not return any status packets.

LENGTH The length of the packet where its value is “Number of parameters (N) + 2”

INSTRUCTION The instruction for the Dynamixel actuator to perform.

PARAMETER0 N Used if there is additional information needed to be sent other than the instruction itself.

CHECK SUM The computation method for the ‘Check Sum’ is as the following.

Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)
If the calculated value is larger than 255, the lower byte is defined as the checksum value.
~ represents the NOT logic operation.

3-3. Status Packet(Return Packet)

The Status Packet is the response packet from the Dynamixel units to the Main Controller after receiving an instruction packet. The structure of the status packet is as the following.

| OXFF | 0XFF | ID | LENGTH | ERROR | PARAMETER1 | PARAMETER2 | PARAMETER N | CHECK SUM |

The meanings of each packet byte definition are as the following.

0XFF 0XFF The two 0XFF bytes indicate the start of the packet.

ID The unique ID of the Dynamixel unit returning the packet. The initial value is set to 1.

LENGTH The length of the packet where its value is “Number of parameters (N) + 2”

ERROR The byte representing errors sent from the Dynamixel unit. The meaning of each bit is as the following.

Bit	Name	Details
Bit 7	0	-
Bit 6	Instruction Error	Set to 1 if an undefined instruction is sent or an action instruction is sent without a Reg_Write instruction.
Bit 5	Overload Error	Set to 1 if the specified maximum torque can't control the applied load.
Bit 4	Checksum Error	Set to 1 if the checksum of the instruction packet is incorrect.
Bit 3	Range Error	Set to 1 if the instruction sent is out of the defined range.
Bit 2	Overheating Error	Set to 1 if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.
Bit 1	Angle Limit Error	Set as 1 if the Goal Position is set outside of the range between CW Angle Limit and CCW Angle Limit.
Bit 0	Input Voltage Error	Set to 1 if the voltage is out of the operating voltage range as defined in the control table.

PARAMETER0 N Used if additional information is needed.

CHECK SUM The computation method for the ‘Check Sum’ is as the following.
Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)
If the calculated value is larger than 255, the lower byte is defined as the checksum value. ~ represents the NOT logic operation.

3-4. Control Table

	Address	Item	Access	Initial Value
EEPROM Area	0(0X00)	Model Number(L)	RD	12(0x0C)
	1(0X01)	Model Number(H)	RD	0(0x00)
	2(0X02)	Version of Firmware	RD	?
	3(0X03)	ID	RD,WR	1(0x01)
	4(0X04)	Baud Rate	RD,WR	1(0x01)
	5(0X05)	Return Delay Time	RD,WR	250(0xFA)
	6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
	7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
	8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
	9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
	10(0x0A)	(Reserved)	-	0(0x00)
	11(0X0B)	the Highest Limit Temperature	RD,WR	85(0x55)
	12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
	13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
	14(0X0E)	Max Torque(L)	RD,WR	255(0XFF)
	15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
	16(0X10)	Status Return Level	RD,WR	2(0x02)
	17(0X11)	Alarm LED	RD,WR	4(0x04)
	18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
	19(0X13)	(Reserved)	RD,WR	0(0x00)
	20(0X14)	Down Calibration(L)	RD	?
	21(0X15)	Down Calibration(H)	RD	?
	22(0X16)	Up Calibration(L)	RD	?
	23(0X17)	Up Calibration(H)	RD	?
RAM Area	24(0X18)	Torque Enable	RD,WR	0(0x00)
	25(0X19)	LED	RD,WR	0(0x00)
	26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
	27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
	28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
	29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
	30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
	31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
	32(0X20)	Moving Speed(L)	RD,WR	0
	33(0X21)	Moving Speed(H)	RD,WR	0
	34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
	35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
	36(0X24)	Present Position(L)	RD	?
	37(0X25)	Present Position(H)	RD	?
	38(0X26)	Present Speed(L)	RD	?
	39(0X27)	Present Speed(H)	RD	?
	40(0X28)	Present Load(L)	RD	?
	41(0X29)	Present Load(H)	RD	?
	42(0X2A)	Present Voltage	RD	?
	43(0X2B)	Present Temperature	RD	?
	44(0X2C)	Registered Instruction	RD,WR	0(0x00)
	45(0X2D)	(Reserved)	-	0(0x00)
	46[0x2E)	Moving	RD	0(0x00)
	47[0x2F)	Lock	RD,WR	0(0x00)
	48[0x30)	Punch(L)	RD,WR	32(0x20)
	49[0x31)	Punch(H)	RD,WR	0(0x00)

Control Table The Control Table contains information on the status and operation of the Dynamixel actuator. The Dynamixel actuator is operated by writing values to its control table and its status is checked by reading values off its control table.

RAM and EEPROM The data values for the RAM area will be set to the default initial values whenever the power is turned on. However, the data values for the EEPROM area are non-volatile and will still remain even after the power is turned off.

Initial Value The Initial Value column on the right side of the control table shows the Factory Default Values for the case of EEPROM area data, and shows the initial value when the power is turned on for the case of RAM area data.

The following explains the meaning of data stored in each of the addresses in the control table.

Address 0x00,0x01 **Model Number**. For AX-12, this value is 0X000C (12).

Address 0x02 **Firmware Version**.

Address 0x03 **ID**. The unique ID number assigned to each Dynamixel actuators for identifying them. Different IDs are required for each Dynamixel actuators that are on the same network.

Address 0x04 **Baud Rate.** Determines the communication speed. The computation is done by the following formula.

Speed (BPS) = 2000000 / (Address4 + 1)

Data Value for each Major Baud Rate

Adress4	Hex	Set BPS	Goal BPS	Error
1	0X01	1000000.0	1000000.0	0.000%
3	0X03	500000.0	500000.0	0.000%
4	0X04	400000.0	400000.0	0.000%
7	0X07	250000.0	250000.0	0.000%
9	0X09	200000.0	200000.0	0.000%
16	0X10	117647.1	115200.0	-2.124%
34	0X22	57142.9	57600.0	0.794%
103	0X67	19230.8	19200.0	-0.160%
207	0XCF	9615.4	9600.0	-0.160%

Note A maximum Baud Rate error of 3% is within the tolerance of UART communication.

Caution The initial value of Baudrate is set to 1(1000000bps)

Address 0x05 **Return Delay Time.** The time it takes for the Status Packet to return after the Instruction Packet is sent. The delay time is given by 2uSec * Address5 value.

Address 0x06,0x07,0x08,0x09

Operating Angle Limit. Sets the Dynamixel actuator's operating angle range. The Goal Position needs to be within the range of: CW Angle Limit <= Goal Position <= CCW Angle Limit. An Angle Limit Error will occur if the Goal Position is set outside this range set by the operating angle limits.

Address 0x0B **the Highest Limit Temperature.** The upper limit of the Dynamixel actuator's operating temperature. If the internal temperature of the Dynamixel actuator gets higher than this value, the Over Heating Error Bit (Bit 2 of the Status Packet) will return the value 1, and an alarm will be set by Address 17, 18. The values are in Degrees Celsius.

Address 0x0C,0x0D **the Lowest (Highest) Limit Voltage.** The upper and lower limits of the Dynamixel actuator's operating voltage. If the present voltage (Address 42) is out of the specified range, a Voltage Range Error Bit (Bit 0 of the Status Packet) will return the value 1, and an alarm will be set by Address 17, 18. The values are 10 times the actual voltage value. For example, if the Address 12 value is 80, then the lower voltage limit is set to 8V.

Address 0x0E,0x0F, 0x22,0x23

Max Torque. The maximum torque output for the Dynamixel actuator. When this value is set to 0, the Dynamixel actuator enters the Free Run mode. There are two locations where this maximum torque limit is defined; in the EEPROM (Address 0X0E, 0x0F) and in the RAM (Address 0x22, 0x23). When the power is turned on, the maximum torque limit value defined in the EEPROM is copied to the location in the RAM. The torque of the Dynamixel actuator is limited by the values located in the RAM (Address 0x22, 0x23).

Address 0X10 **Status Return Level.** Determines whether the Dynamixel actuator will return a Status Packet after receiving an Instruction Packet.

Address16	Returning the Status Packet
0	Do not respond to any instructions
1	Respond only to READ_DATA instructions
2	Respond to all instructions

In the case of an instruction which uses the Broadcast ID (0XFE) the Status Packet will not be returned regardless of the Address 0x10 value.

Address 0X11 **Alarm LED.** If the corresponding Bit is set to 1, the LED blinks when an Error occurs.

Bit	Function
Bit 7	0
Bit 6	If set to 1, the LED blinks when an Instruction Error occurs
Bit 5	If set to 1, the LED blinks when an Overload Error occurs
Bit 4	If set to 1, the LED blinks when a Checksum Error occurs
Bit 3	If set to 1, the LED blinks when a Range Error occurs
Bit 2	If set to 1, the LED blinks when an Overheating Error occurs
Bit 1	If set to 1, the LED blinks when an Angle Limit Error occurs
Bit 0	If set to 1, the LED blinks when an Input Voltage Error occurs

This function operates following the “OR” logical operation of all bits. For example, if the value is set to 0X05, the LED will blink when an Input Voltage Error occurs or when an Overheating Error occurs. Upon returning to a normal condition from an error state, the LED stops blinking after 2 seconds.

Address 0X12 **Alarm Shutdown.** If the corresponding Bit is set to a 1, the Dynamixel actuator’s torque will be turned off when an error occurs.

Bit	Function
Bit 7	0
Bit 6	If set to 1, torque off when an Instruction Error occurs
Bit 5	If set to 1, torque off when an Overload Error occurs
Bit 4	If set to 1, torque off when a Checksum Error occurs
Bit 3	If set to 1, torque off when a Range Error occurs
Bit 2	If set to 1, torque off when an Overheating Error occurs
Bit 1	If set to 1, torque off when an Angle Limit Error occurs
Bit 0	If set to 1, torque off when an Input Voltage Error occurs

This function operates following the “OR” logical operation of all bits. However, unlike the Alarm LED, after returning to a normal condition, it maintains the torque off status. To recover, the Torque Enable (Address0X18) needs to be reset to 1.

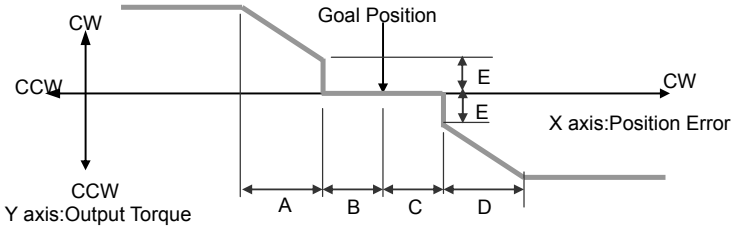
Address 0x14~0x17 **Calibration.** Data used for compensating for the differences between the potentiometers used in the Dynamixel units. The user cannot change this data.

The following (from Address 0x18) is in the RAM area.

Address 0x18 **Torque Enable.** When the power is first turned on, the Dynamixel actuator enters the Torque Free Run condition (zero torque). Setting the value in Address 0x18 to 1 enables the torque.

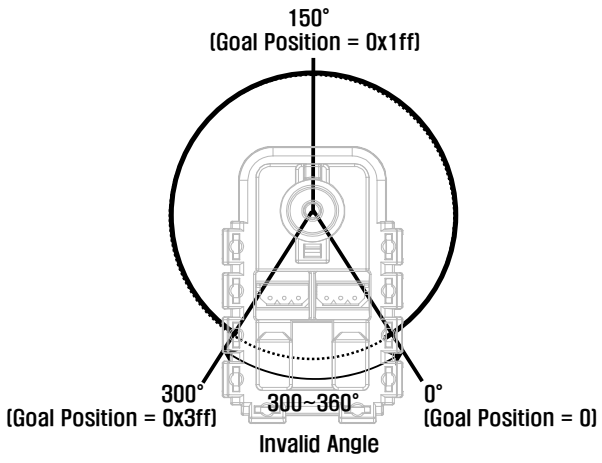
Address 0x19 **LED**. The LED turns on when set to 1 and turns off if set to 0.

Address 0x1A~0x1D **Compliance Margin and Slope.** The compliance of the Dynamixel actuator is defined by setting the compliance Margin and Slope. This feature can be utilized for absorbing shocks at the output shaft. The following graph shows how each compliance value (length of A, B, C & D) is defined by the Position Error and applied torque.



A : CCW Compliance Slope(Address0x1D)
B : CCW Compliance Margin(Address0x1B)
C : CW Compliance Margin(Address0x1A)
D : CW Compliance Slope (Address0x1C)
E : Punch(Address0x30,31)

Address 0X1E,0x1F **Goal Position** Requested angular position for the Dynamixel actuator output to move to. Setting this value to 0x3ff moves the output shaft to the position at 300°.



Address 0x20,0x21 **Moving Speed.** Sets the angular velocity of the output moving to the Goal Position. Setting this value to its maximum value of 0x3ff moves the output with an angular velocity of 114 RPM, provided that there is enough power supplied (The lowest velocity is when this value is set to 1. When set to 0, the velocity is the largest possible for the supplied voltage, e.g. no velocity control is applied.)

Address 0x24,0x25 **Present Position.** Current angular position of the Dynamixel actuator output.

Address 0x26,0x27 **Present Speed.** Current angular velocity of the Dynamixel actuator output.

Address 0x28,0x29 **Present Load.** The magnitude of the load on the operating Dynamixel actuator. Bit 10 is the direction of the load.

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Load Direction	Load Value									

Load Direction = 0 : CCW Load, Load Direction = 1: CW Load

Address 0x2A **Present Voltage.** The voltage currently applied to the Dynamixel actuator. The value is 10 times the actual voltage. For example, 10V is represented as 100 (0x64).

Address 0x2B **Present Temperature.** The internal temperature of the Dynamixel actuator in Degrees Celsius.

Address 0x2C **Registered Instruction.** Set to 1 when an instruction is assigned by the REG_WRITE command. Set to 0 after it completes the assigned instruction by the Action command.

Address 0x2E **Moving.** Set to 1 when the Dynamixel actuator is moving by its own power.

Address 0x2F **Lock.** If set to 1, only Address 0x18 to 0x23 can be written to and other areas cannot. Once locked, it can only be unlocked by turning the power off.

Address 0x30,0x31 **Punch.** The minimum current supplied to the motor during operation. The initial value is set to 0x20 and its maximum value is 0x3ff.

Endless Turn If both values for the CW Angle Limit and the CCW Angle Limit are set to 0, an Endless Turn mode can be implemented by setting the Goal Speed. This feature can be used for implementing a continuously rotating wheel.

Goal Speed Setting

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Turn Direction	Speed Value									

Turn Direction = 0 : CCW Direction Turn, Load Direction = 1: CW Direction Turn

Range

Each data has a valid minimum and maximum values. Write instructions made outside of these valid ranges will return an error. The following table summarizes the data range for each register. 16 bit data registers are indicated with two bytes (L) and (H). Both bytes need to be written at the same time as one instruction packet.

Write Address	Writing Item	Length (bytes)	Min	Max
3(0X03)	ID	1	0	253(0xfd)
4(0X04)	Baud Rate	1	0	254(0xfe)
5(0X05)	Return Delay Time	1	0	254(0xfe)
6(0X06)	CW Angle Limit	2	0	1023(0x3ff)
8(0X08)	CCW Angle Limit	2	0	1023(0x3ff)
11(0X0B)	the Highest Limit Temperature	1	0	150(0x96)
12(0X0C)	the Lowest Limit Voltage	1	50(0x32)	250(0xfa)
13(0X0D)	the Highest Limit Voltage	1	50(0x32)	250(0xfa)
14(0X0E)	Max Torque	2	0	1023(0x3ff)
16(0X10)	Status Return Level	1	0	2
17(0X11)	Alarm LED	1	0	127(0x7f)
18(0X12)	Alarm Shutdown	1	0	127(0x7f)
19(0X13)	(Reserved)	1	0	1
24(0X18)	Torque Enable	1	0	1
25(0X19)	LED	1	0	1
26(0X1A)	CW Compliance Margin	1	0	254(0xfe)
27(0X1B)	CCW Compliance Margin	1	0	254(0xfe)
28(0X1C)	CW Compliance Slope	1	1	254(0xfe)
29(0X1D)	CCW Compliance Slope	1	1	254(0xfe)
30(0X1E)	Goal Position	2	0	1023(0x3ff)
32(0X20)	Moving Speed	2	0	1023(0x3ff)
34(0X22)	Torque Limit	2	0	1023(0x3ff)
44(0X2C)	Registered Instruction	1	0	1
47(0X2F)	Lock	1	1	1
48(0X30)	Punch	2	0	1023(0x3ff)

[Control Table Data Range and Length for Writing]

4. Instruction Set and Examples

The following Instructions are available.

Instruction	Function	Value	Number of Parameter
PING	No action. Used for obtaining a Status Packet	0x01	0
READ DATA	Reading values in the Control Table	0x02	2
WRITE DATA	Writing values to the Control Table	0x03	2 ~
REG WRITE	Similar to WRITE_DATA, but stays in standby mode until the ACION instruction is given	0x04	2 ~
ACTION	Triggers the action registered by the REG_WRITE instruction	0x05	0
RESET	Changes the control table values of the Dynamixel actuator to the Factory Default Value settings	0x06	0
SYNC WRITE	Used for controlling many Dynamixel actuators at the same time	0x83	4~

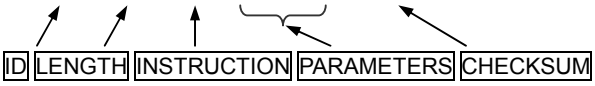
4-1. WRITE_DATA

Function To write data into the control table of the Dynamixel actuator
Length N+3 (N is the number of data to be written)
Instruction 0X03
Parameter1 Starting address of the location where the data is to be written
Parameter2 1st data to be written
Parameter3 2nd data to be written
Parameter N+1 Nth data to be written

Example 1 **Setting the ID of a connected Dynamixel actuator to 1**

Write 1 to address 3 of the control table. The ID is transmitted using the Broadcasting ID (0xFE).

Instruction Packet : 0XFF 0XFF 0XFE 0X04 0X03 0X03 0X01 0XF6`



Because it was transmitted with a Broadcast ID (0XFE), no status packets are returned.

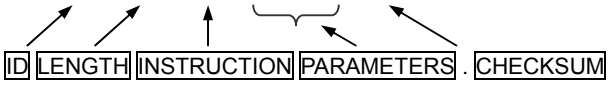
4-2. READ_DATA

Function	Read data from the control table of a Dynamixel actuator
Length	0X04
Instruction	0X02
Parameter1	Starting address of the location where the data is to be read
Parameter2	Length of the data to be read

Example 2 **Reading the internal temperature of the Dynamixel actuator with an ID of 1**

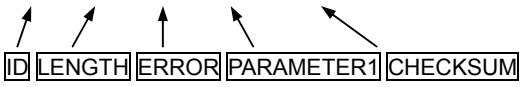
Read 1 byte from address 0x2B of the control table.

Instruction Packet : 0XFF 0XFF 0X01 0X04 0X02 0X2B 0X01 0XCC`



The returned Status Packet will be as the following.

Status Packet : 0XFF 0XFF 0X01 0X03 0X00 0X20 0XDB



The data read is 0x20. Thus the current internal temperature of the Dynamixel actuator is approximately 32°C (0X20).

4-3. REG_WRITE과 ACTION

4-3-1. REG_WRITE

Function The REG_WRITE instruction is similar to the WRITE_DATA instruction, but the

execution timing is different. When the Instruction Packet is received the values are stored in the Buffer and the Write instruction is under a standby status. At this time, the Registered Instruction register (Address 0x2C) is set to 1. After the Action Instruction Packet is received, the registered Write instruction is finally executed.

Length	N+3 (N is the number of data to be written)
Instruction	0X04
Parameter1	Starting address of the location where the data is to be written
Parameter2	1st data to be written
Parameter3	2nd data to be written
Parameter N+1	Nth data to be written

4-3-2. ACTION

Function	Triggers the action registered by the REG_WRITE instruction
Length	0X02
Instruction	0X05
Parameter	NONE

The ACTION instruction is useful when multiple Dynamixel actuators need to move simultaneously. When controlling multiple Dynamixel actuator units, slight time delays can occur between the 1st and last units to receive an instruction. The Dynamixel actuator handles this problem by using the ACTION instruction.

Broadcasting The Broadcast ID (0XFE) is used when sending ACTION instructions to more than two Dynamixel actuators. Note that no packets are returned by this operation.

4-4. PING

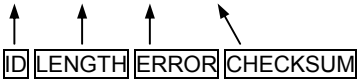
Function	Does not command any operations. Used for requesting a status packet or to check the existence of a Dynamixel actuator with a specific ID.
Length	0X02
Instruction	0X01
Parameter	NONE

Example 3 **Obtaining the status packet of the Dynamixel actuator with an ID of 1**

Instruction Packet : 0XFF 0XFF 0X01 0X02 0X01 0XFB`

The returned Status Packet is as the following

Status Packet : 0XFF 0XFF 0X01 0X02 0X00 0XFC



Regardless of whether the Broadcasting ID is used or the Status Return Level (Address 16) is 0, a Status Packet is always returned by the PING instruction.

4-5. RESET

Function Changes the control table values of the Dynamixel actuator to the Factory Default Value settings

Length 0X02

Instruction 0X06

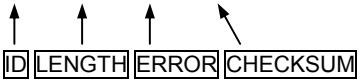
Parameter NONE

Example 4 **Resetting the Dynamixel actuator with an ID of 0**

Instruction Packet : 0XFF 0XFF 0X00 0X02 0X06 0XF7`

The returned Status Packet is as the following

Status Packet : 0XFF 0XFF 0X00 0X02 0X00 0XFD



Note the ID of this Dynamixel actuator is now changed to 1 after the RESET instruction.

4-6. SYNC WRITE

Function Used for controlling many Dynamixel actuators at the same time. The communication time decreases by the Synch Write instruction since many instructions can be transmitted by a single instruction. However, you can use this instruction only when the lengths and addresses of the control table to be written to are the same. Also, the broadcasting ID needs to be used for transmitting.

ID	0XFE
Length	(L + 1) * N + 4 (L: Data length for each Dynamixel actuator, N: The number of Dynamixel actuators)
Instruction	0X83
Parameter1	Starting address of the location where the data is to be written
Parameter2	The length of the data to be written (L)
Parameter3	The ID of the 1st Dynamixel actuator
Parameter4	The 1st data for the 1st Dynamixel actuator
Parameter5	The 2nd data for the 1st Dynamixel actuator
Parameter L+3	The Lth data for the 1st Dynamixel actuator
Parameter L+4	The ID of the 2nd Dynamixel actuator
Parameter L+5	The 1st data for the 2nd Dynamixel actuator
Parameter L+6	The 2nd data for the 2nd Dynamixel actuator
Parameter 2L+4	The Lth data for the 2nd Dynamixel actuator

Data for the 1st Dynamixel actuator

Data for the 2nd Dynamixel actuator

.

Example 5 **Setting the following positions and velocities for 4 Dynamixel actuators**

Dynamixel actuator with an ID of 0: to position 0X010 with a speed of 0X150
Dynamixel actuator with an ID of 1: to position 0X220 with a speed of 0X360
Dynamixel actuator with an ID of 2: to position 0X030 with a speed of 0X170
Dynamixel actuator with an ID of 0: to position 0X220 with a speed of 0X380

Instruction Packet : 0XFF 0XFF 0XFE 0X18 0X83 0X1E 0X04 0X00 0X10 0X00 0X50 0X01 0X01 0X20 0X02 0X60 0X03 0X02 0X30 0X00 0X70 0X01 0X03 0X20 0X02 0X80 0X03 0X12
No status packets are returned since the Broadcasting ID was used.

5. Example

For the following examples, we assume a Dynamixel actuator with an ID of 1 in Reset status and that the Baud rate is 57142 BPS.

Example 6 **Reading the Model Number and Firmware Version of the Dynamixel actuator with an ID of 1**

Instruction Packet Instruction = READ_DATA, Address = 0x00, Length = 0x03

Communication ->[Dynamixel]:FF FF 01 04 02 00 03 F5 (LEN:008)
<-[Dynamixel]:FF FF 01 05 00 74 00 08 7D (LEN:009)

Status Packet Result Model Number = 116 (0x74) (for the case of DX-116) Firmware Version = 0x08

Example 7 **Changing the ID to 0 for a Dynamixel actuator with an ID of 1**

Instruction Packet Instruction = WRITE_DATA, Address = 0x03, DATA = 0x00

Communication ->[Dynamixel]:FF FF 01 04 03 03 00 F4 (LEN:008)
<-[Dynamixel]:FF FF 01 02 00 FC (LEN:006)

Status Packet Result NO ERROR

Example 8 **Changing the Baud Rate of a Dynamixel actuator to 1M bps**

Instruction Packet Instruction = WRITE_DATA, Address = 0x04, DATA = 0x01

Communication ->[Dynamixel]:FF FF 00 04 03 04 01 F3 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 9 **Resetting the Return Delay Time to 4 uSec for a Dynamixel actuator with an ID of 0**

A Return Delay Time Value of 1 corresponds to 2uSec.

Instruction Packet Instruction = WRITE_DATA, Address = 0x05, DATA = 0x02

Communication ->[Dynamixel]:FF FF 00 04 03 05 02 F1 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

It is recommended to set the Return Delay Time to the minimum value allowed by the Main Controller.

Example 10 **Limiting the operating angle range to 0°~150° for a Dynamixel actuator with an ID of 0**

Since the CCW Angle Limit of 0x3ff corresponds to 300°, the angle 150° is represented by the value 0x1ff

Instruction Packet Instruction = WRITE_DATA, Address = 0x08, DATA = 0xff, 0x01

Communication ->[Dynamixel]:FF FF 00 05 03 08 FF 01 EF (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 11 **Resetting the upper limit for the operating temperature to 80°C for a Dynamixel actuator with an ID of 0**

Instruction Packet Instruction = WRITE_DATA, Address = 0x0B, DATA = 0x50

Communication ->[Dynamixel]:FF FF 00 04 03 0B 50 9D (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 12 **Setting the operating voltage to 10V ~ 17V for a Dynamixel actuator with an ID of 0**

10V is represented by 100 (0x64), and 17V by 170 (0xAA).

Instruction Packet Instruction = WRITE_DATA, Address = 0x0C, DATA = 0x64, 0xAA

Communication ->[Dynamixel]:FF FF 00 05 03 0C 64 AA DD (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 13 **Setting the maximum torque to 50% of its maximum possible value for a Dynamixel actuator with an ID of 0**

Set the MAX Torque value located in the ROM area to 0x1ff which is 50% of the maximum value 0x3ff.

Instruction Packet Instruction = WRITE_DATA, Address = 0x0E, DATA = 0xff, 0x01

Communication ->[Dynamixel]:FF FF 00 05 03 0E FF 01 E9 (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR
To verify the effect of the adjusted Max Torque value, the power needs to be turned off and then on.

Example 14 **Set the Dynamixel actuator with an ID of 0 to never return a Status Packet**

Instruction Packet Instruction = WRITE_DATA, Address = 0x10, DATA = 0x00

Communication ->[Dynamixel]:FF FF 00 04 03 10 00 E8 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR
The Status Packet is not returned starting with the following instruction.

Example 15 **Set the Alarm to blink the LED and Shutdown (Torque off) the actuator when the operating temperature goes over the set limit**

Since the Overheating Error is Bit 2, set the Alarm value to 0x04.

Instruction Packet Instruction = WRITE_DATA, Address = 0x11, DATA = 0x04, 0x04

Communication ->[Dynamixel]:FF FF 00 05 03 11 04 04 DE (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 16 **Turn on the LED and Enable Torque for a Dynamixel actuator with an ID of 0**

Instruction Packet Instruction = WRITE_DATA, Address = 0x18, DATA = 0x01, 0x01

Communication

->[Dynamixel]:FF FF 00 05 03 18 01 01 DD (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result

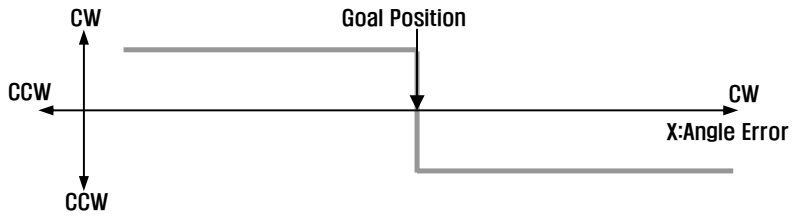
NO ERROR
You can verify the Torque Enabled status by trying to move the output of the actuator by hand.

Example 17

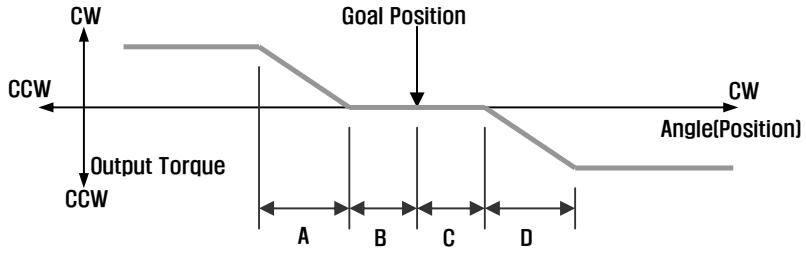
Setting the Compliance Margin to 1 and Compliance Slope to 0x40 for a Dynamixel actuator with an ID of 0

Compliance

The Angle Error and Torque Output can be represented with the following graph.



Even if the position deviates a little from the goal position in the CW direction, a large amount of torque is generated in the CCW direction to compensate for this. However, since inertia must be considered, a realistic implementation differs from this approach. Considering this, the given conditions can be represented by the following graph.



A : CCW Compliance Slope (Address0x1D) = 0x40 (about 18.8°)
B : CCW Compliance Margin (Address0x1B) = 0x01 (about 0.29°)
C : CW Compliance Margin (Address0x01A) = 0x01 (about 0.29°)
D : CW Compliance Slope (Address0x1C) = 0x40 (about 18.8°)

Instruction Packet Instruction = WRITE_DATA, Address = 0x1A, DATA = 0x01, 0x01, 0x40, 0x40

Communication ->[Dynamixel]:FF FF 00 07 03 1A 01 01 40 40 59 (LEN:011)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

The Compliance Slope takes effect with discrete steps of 2^n (n is integer). Thus any Compliance value between 0x11 and 0x20 has identical effects.

Example 18 **Position the output of a Dynamixel actuator with an ID of 0 to 180° with an angular velocity of 057RPM**

Set Address 0x1E (Goal Position) to 0x200 and Address 0x20 (Moving Speed) to 0x200.

Instruction Packet Instruction = WRITE_DATA, Address = 0x1E, DATA = 0x00, 0x02, 0x00, 0x02

Communication ->[Dynamixel]:FF FF 00 07 03 1E 00 02 00 02 D3 (LEN:011)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Example 19 **Position the output of a Dynamixel actuator with an ID of 0 to 0° and Position the output of a Dynamixel actuator with an ID of 1 to 300°, and initiate the movement at the same time.**

If the WRITE_DATA is used, the movement of the two actuators cannot be initiate at the same time, thus the REG_WRITE and ACTION instructions should be used instead.

Instruction Packet ID=0, Instruction = REG_WRITE, Address = 0x1E, DATA = 0x00, 0x00
ID=1, Instruction = REG_WRITE, Address = 0x1E, DATA = 0xff, 0x03
ID=0xfe(Broadcasting ID), Instruction = ACTION,

Communication ->[Dynamixel]:FF FF 00 05 04 1E 00 00 D8 (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)
->[Dynamixel]:FF FF 01 05 04 1E FF 03 D5 (LEN:009)
<-[Dynamixel]:FF FF 01 02 00 FC (LEN:006)
->[Dynamixel]:FF FF FE 02 05 FA (LEN:006)
<-[Dynamixel]: //No return packet against broadcasting ID

Status Packet Result NO ERROR

Example 20 **Lock all addresses except for Address 0x18 ~ Address0x23 for a Dynamixel actuator with an ID of 0**

Set Address 0x2F (Lock) to 1.

Instruction Packet Instruction = WRITE_DATA, Address = 0x2F, DATA = 0x01

Communication
->[Dynamixel]:FF FF 00 04 03 2F 01 C8 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Once locked, the only way to unlock it is to remove the power.

If an attempt is made to access any locked data, an error is returned.

->[Dynamixel]:FF FF 00 05 03 30 40 00 87 (LEN:009)
<-[Dynamixel]:FF FF 00 02 08 F5 (LEN:006)

Example 21 **Set the minimum power (Punch) to 0x40 for a Dynamixel actuator with an ID of 0**

Instruction Packet Instruction = WRITE_DATA, Address = 0x30, DATA = 0x40, 0x00

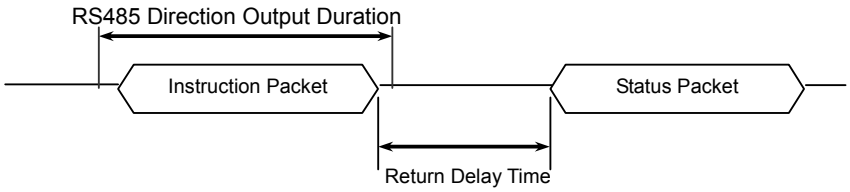
Communication
->[Dynamixel]:FF FF 00 05 03 30 40 00 87 (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Appendix

Half duplex UART

Half duplex UART is a serial communication protocol where both TxD and RxD cannot be used at the same time. This method is generally used when many devices need to be connected to a single bus. Since more than one device are connected to the same bus, all the other devices need to be in input mode while one device is transmitting. The Main Controller that controllers the Dynamixel actuators sets the communication direction to input mode, and only when it is transmitting an Instruction Packet, it changes the direction to output mode.



Return Delay Time

The time it takes for the Dynamixel actuator to return the Status Packet after receiving an Instruction Packet. The Default Value is 160 uSec and can be changed via the Control Table at Address 5. The Main Controller needs to change the Direction Port to input mode during the Return Delay Time after sending an instruction packet.

Tx,Rx Direction

For Half Duplex UART, the transmission ending timing is important to change the direction to receiving mode. The bit definitions within the register that indicates UART_STATUS are as the following

TXD_BUFFER_READY_BIT: Indicates that the transmission DATA can be loaded into the Buffer. Note that this only means that the SERIAL TX BUFFER is empty, and does not necessarily mean that the all the data transmitted before has left the CPU.

TXD_SHIFT_REGISTER_EMPTY_BIT: Set when all the Transmission Data has completed its transmission and left the CPU.

The TXD_BUFFER_READY_BIT is used when one byte is to be transmitted via the serial communication channel, and an example is shown below.

```
TxDByte(byte bData)
{
    while(!TXD_BUFFER_READY_BIT);    //wait until data can be loaded.
    SerialTxDBuffer = bData;          //data load to TxD buffer
}
```

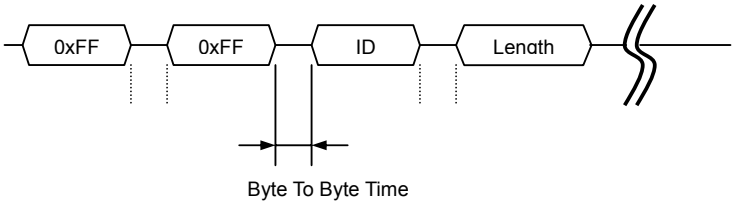
When changing the direction, the TXD_SHIFT_REGISTER_EMPTY_BIT must be checked.

The following is an example program that sends an Instruction Packet.

```
LINE 1    DIRECTION_PORT = TX_DIRECTION;
LINE 2    TxDByte(0xff);
LINE 3    TxDByte(0xff);
LINE 4    TxDByte(bID);
LINE 5    TxDByte(bLength);
LINE 6    TxDByte(bInstruction);
LINE 7    TxDByte(Parameter0);    TxDByte(Parameter1);
LINE 8    DisableInterrupt(); // interrupt should be disable
LINE 9    TxDByte(Checksum);    //last TxD
LINE 10   while(!TXD_SHIFT_REGISTER_EMPTY_BIT); //Wait till last data bit has been sent
LINE 11   DIRECTION_PORT = RX_DIRECTION; //Direction change to RXD
LINE 12   EnableInterrupt(); // enable interrupt again
```

Please note the important lines between LINE 8 and LINE 12. Line 8 is necessary since an interrupt here may cause a delay longer than the return delay time and corruption to the front of the status packet may occur.

Byte to Byte Time The delay time between bytes when sending an instruction packet. If the delay time is over 100ms, then the Dynamixel actuator recognizes this as a communication problem and waits for the next header (0xff 0xff) of a packet again.



The following is the source code of a program (Example.c) that accesses the Dynamixel actuator using the Atmega 128.

C Language Example : Dinamixel access with Atmega128

```
/*
 * The Example of Dynamixel Evaluation with Atmega128
 * Date : 2005.5.11
 * Author : BS KIM
 */

/*
 * included files
 */
#define ENABLE_BIT_DEFINITIONS
//#include <io.h>
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>

#define cbi(REG8,BITNUM) REG8 &= ~(_BV(BITNUM))
#define sbi(REG8,BITNUM) REG8 |= _BV(BITNUM)


typedef unsigned char byte;
typedef unsigned int word;
#define ON 1
#define OFF 0
#define _ON 0
#define _OFF 1


//--- Control Table Address ---
//EEPROM AREA
#define P_MODEL_NUMBER_L      0
#define P_MODOEL_NUMBER_H     1
#define P_VERSION             2
#define P_ID                  3
#define P_BAUD_RATE           4
#define P_RETURN_DELAY_TIME   5
#define P_CW_ANGLE_LIMIT_L    6
#define P_CW_ANGLE_LIMIT_H    7
#define P_CCW_ANGLE_LIMIT_L   8
#define P_CCW_ANGLE_LIMIT_H   9
#define P_SYSTEM_DATA2        10
#define P_LIMIT_TEMPERATURE   11
#define P_DOWN_LIMIT_VOLTAGE  12
#define P_UP_LIMIT_VOLTAGE    13
#define P_MAX_TORQUE_L        14
#define P_MAX_TORQUE_H        15
#define P_RETURN_LEVEL        16
#define P_ALARM_LED           17
#define P_ALARM_SHUTDOWN      18
#define P_OPERATING_MODE      19
#define P_DOWN_CALIBRATION_L  20
#define P_DOWN_CALIBRATION_H  21
#define P_UP_CALIBRATION_L    22
#define P_UP_CALIBRATION_H    23

#define P_TORQUE_ENABLE          (24)
#define P_LED                    (25)
#define P_CW_COMPLIANCE_MARGIN   (26)
#define P_CCW_COMPLIANCE_MARGIN  (27)
#define P_CW_COMPLIANCE_SLOPE    (28)
#define P_CCW_COMPLIANCE_SLOPE   (29)
#define P_GOAL_POSITION_L        (30)
#define P_GOAL_POSITION_H        (31)
#define P_GOAL_SPEED_L           (32)
#define P_GOAL_SPEED_H           (33)
#define P_TORQUE_LIMIT_L         (34)
#define P_TORQUE_LIMIT_H         (35)
#define P_PRESENT_POSITION_L     (36)
#define P_PRESENT_POSITION_H     (37)
#define P_PRESENT_SPEED_L        (38)
#define P_PRESENT_SPEED_H        (39)
#define P_PRESENT_LOAD_L         (40)
#define P_PRESENT_LOAD_H         (41)
#define P_PRESENT_VOLTAGE        (42)
#define P_PRESENT_TEMPERATURE    (43)
#define P_REGISTERED_INSTRUCTION (44)
#define P_PAUSE_TIME             (45)
#define P_MOVING (46)
#define P_LOCK                   (47)
#define P_PUNCH_L                (48)
#define P_PUNCH_H                (49)

//--- Instruction ---
#define INST_PING           0x01
#define INST_READ           0x02
#define INST_WRITE          0x03
#define INST_REG_WRITE      0x04
#define INST_ACTION         0x05
#define INST_RESET          0x06
#define INST_DIGITAL_RESET  0x07
#define INST_SYSTEM_READ    0x0C
#define INST_SYSTEM_WRITE   0x0D
#define INST_SYNC_WRITE     0x83
#define INST_SYNC_REG_WRITE 0x84

#define CLEAR_BUFFER gbRxBufferReadPointer = gbRxBufferWritePointer
#define DEFAULT_RETURN_PACKET_SIZE 6
#define BROADCASTING_ID 0xfe

#define TxD8 TxD81
#define RxD8 RxD81

//Hardware Dependent Item
#define DEFAULT_BAUD_RATE 34   //57600bps at 16MHz

////// For CM-5
#define    RS485_TXD    PORTE    &=    ~_BV(PE3),PORTE    |=    _BV(PE2)
                       //PORT_485_DIRECTION = 1
#define    RS485_RXD    PORTE    &=    ~_BV(PE2),PORTE    |=    _BV(PE3)
                       //PORT_485_DIRECTION = 0
/*
////// For CM-2
#define RS485_TXD PORTE |= _BV(PE2); //_485_DIRECTION = 1
#define RS485_RXD PORTE &= ~_BV(PE2);//PORT_485_DIRECTION = 0
*/
//#define TXD0_FINISH  UCSR0A,6  //This bit is for checking TxD Buffer
                       in CPU is empty or not.
//#define TXD1_FINISH  UCSR1A,6

#define SET_TxD0_FINISH   sbi(UCSR0A,6)
#define RESET_TXD0_FINISH cbi(UCSR0A,6)
#define CHECK_TXD0_FINISH bit_is_set(UCSR0A,6)
#define SET_TxD1_FINISH  sbi(UCSR1A,6)
#define RESET_TXD1_FINISH cbi(UCSR1A,6)
#define CHECK_TXD1_FINISH bit_is_set(UCSR1A,6)

#define RX_INTERRUPT 0x01
#define TX_INTERRUPT 0x02
#define OVERFLOW_INTERRUPT 0x01
#define SERIAL_PORT0 0
#define SERIAL_PORT1 1
#define BIT_RS485_DIRECTION0  0x08  //Port E
#define BIT_RS485_DIRECTION1  0x04  //Port E

#define BIT_ZIGBEE_RESET              PD4  //out : default 1 //PORTD
#define BIT_ENABLE_RXD_LINK_PC        PD5  //out : default 1
#define BIT_ENABLE_RXD_LINK_ZIGBEE    PD6  //out : default 0
#define BIT_LINK_PLUGIN               PD7  //in, no pull up

void TxD81(byte bTxdData);
void TxD80(byte bTxdData);
void TxDString(byte *bData);
void TxD8Hex(byte bSentData);
void TxD32Dec(long lLong);
byte RxD81(void);
void MiliSec(word wDelayTime);
void PortInitialize(void);
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt);
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength);
byte RxPacket(byte bRxLength);
void PrintBuffer(byte *bpPrintBuffer, byte bLength);
```

```
// --- Gloval Variable Number ---
volatile byte gbpRxInterruptBuffer[256];
byte gbpParameter[128];
byte gbRxBufferReadPointer;
byte gbpRxBuffer[128];
byte gbpTxBuffer[128];
volatile byte gbRxBufferWritePointer;

int main(void)
{
  byte bCount,bID, bTxPacketLength,bRxPacketLength;

  PortInitialize(); //Port In/Out Direction Definition
  RS485_RXD; //Set RS485 Direction to Input State.
  SerialInitialize(SERIAL_PORT0,1,RX_INTERRUPT);//RS485
                    Initializing(RxInterrupt)
  SerialInitialize(SERIAL_PORT1,DEFAULT_BAUD_RATE,0);          //RS232
                    Initializing(None Interrupt)

  gbRxBufferReadPointer  =  gbRxBufferWritePointer  =  0;    //RS485
                    RxBuffer Clearing.

  sei();  //Enable Interrupt -- Compiler Function
  TxDString("¥r¥n   [The   Example   of   Dynamixel   Evaluation   with
                    ATmega128,GCC-AVR]");

//Dynamixel Communication Function Execution Step.
// Step 1. Parameter Setting (gbpParameter[]). In case of no parameter
                    instruction(Ex.  INST_PING),  this  step  is  not
                    needed.
//   Step   2.   TxPacket(ID,INSTRUCTION,LengthOfParameter);   --Total
                    TxPacket Length is returned
//  Step  3.  RxPacket(ExpectedReturnPacketLength);  --  Real  RxPacket
                    Length is returned
// Step 4 PrintBuffer(BufferStartPointer,LengthForPrinting);

  bID = 1;
  TxDString("¥r¥n¥n Example 1. Scanning Dynamixels(0~9). -- Any Key to
                    Continue."); RxD8();
  for(bCount = 0; bCount < 0x0A; bCount++)
  {
    bTxPacketLength = TxPacket(bCount,INST_PING,0);
    bRxPacketLength = RxPacket(255);
    TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString(", RxD:");    PrintBuffer(gbpRxBuffer,bRxPacketLength);
    if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE)
    {
      TxDString(" Found!! ID:");TxD8Hex(bCount);
      bID = bCount;
    }
  }

  TxDString("¥r¥n¥n Example 2. Read Firmware Version. -- Any Key to
                    Continue."); RxD8();
  gbpParameter[0] = P_VERSION; //Address of Firmware Version
  gbpParameter[1] = 1; //Read Length
  bTxPacketLength = TxPacket(bID,INST_READ,2);
  bRxPacketLength                                                    =
                    RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpParameter
                    [1]);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);
  if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE+gbpParameter[1])
  {
    TxDString("¥r¥n Return Error      : ");TxD8Hex(gbpRxBuffer[4]);
    TxDString("¥r¥n Firmware Version  : ");TxD8Hex(gbpRxBuffer[5]);
  }

  TxDString("¥r¥n¥n Example 3. LED ON -- Any Key to Continue.");
                    RxD8();
  gbpParameter[0] = P_LED; //Address of LED
  gbpParameter[1] = 1; //Writing Data
  bTxPacketLength = TxPacket(bID,INST_WRITE,2);
  bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

  TxDString("¥r¥n¥n Example 4. LED OFF -- Any Key to Continue.");
                    RxD8();
  gbpParameter[0] = P_LED; //Address of LED
  gbpParameter[1] = 0; //Writing Data
  bTxPacketLength = TxPacket(bID,INST_WRITE,2);
  bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

  TxDString("¥r¥n¥n Example 5. Read Control Table. -- Any Key to
                    Continue."); RxD8();
  gbpParameter[0] = 0; //Reading Address
  gbpParameter[1] = 49; //Read Length
  bTxPacketLength = TxPacket(bID,INST_READ,2);
  bRxPacketLength                                                    =
                    RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpParameter
                    [1]);

  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);
  if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE+gbpParameter[1])
  {
    TxDString("¥r¥n");
    for(bCount = 0; bCount < 49; bCount++)
    {
      TxD8('[');TxD8Hex(bCount);TxDString("]:");
                    TxD8Hex(gbpRxBuffer[bCount+5]);TxD8(' ');
    }
  }

  TxDString("¥r¥n¥n Example 6. Go 0x200 with Speed 0x100 -- Any Key to
                    Continue."); RxD8();
  gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
  gbpParameter[1] = 0x00; //Writing Data P_GOAL_POSITION_L
  gbpParameter[2] = 0x02; //Writing Data P_GOAL_POSITION_H
  gbpParameter[3] = 0x00; //Writing Data P_GOAL_SPEED_L
  gbpParameter[4] = 0x01; //Writing Data P_GOAL_SPEED_H
  bTxPacketLength = TxPacket(bID,INST_WRITE,5);
  bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

  TxDString("¥r¥n¥n Example 7. Go 0x00 with Speed 0x40 -- Any Key to
                    Continue."); RxD8();
  gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
  gbpParameter[1] = 0x00; //Writing Data P_GOAL_POSITION_L
  gbpParameter[2] = 0x00; //Writing Data P_GOAL_POSITION_H
  gbpParameter[3] = 0x40; //Writing Data P_GOAL_SPEED_L
  gbpParameter[4] = 0x00; //Writing Data P_GOAL_SPEED_H
  bTxPacketLength = TxPacket(bID,INST_WRITE,5);
  bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

  TxDString("¥r¥n¥n Example 8. Go 0x3ff with Speed 0x3ff -- Any Key to
                    Continue."); RxD8();
  gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
  gbpParameter[1] = 0xff; //Writing Data P_GOAL_POSITION_L
  gbpParameter[2] = 0x03; //Writing Data P_GOAL_POSITION_H
  gbpParameter[3] = 0xff; //Writing Data P_GOAL_SPEED_L
  gbpParameter[4] = 0x03; //Writing Data P_GOAL_SPEED_H
  bTxPacketLength = TxPacket(bID,INST_WRITE,5);
  bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

  TxDString("¥r¥n¥n Example 9. Torque Off -- Any Key to Continue.");
                    RxD8();
  gbpParameter[0] = P_TORQUE_ENABLE; //Address of LED
  gbpParameter[1] = 0; //Writing Data
  bTxPacketLength = TxPacket(bID,INST_WRITE,2);
  bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
  TxDString("¥r¥n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
  TxDString("¥r¥n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

  TxDString("¥r¥n¥n End. Push reset button for repeat");
```

```
  while(1);
}

void PortInitialize(void)
{
  DDRA = DDRB = DDRC = DDRD = DDRE = DDRF = 0;  //Set all port to
                    input direction first.
  PORTB = PORTC = PORTD = PORTE = PORTF = PORTG = 0x00; //PortData
                    initialize to 0
  cbi(SFIOR,2); //All Port Pull Up ready
  DDRE  |=  (BIT_RS485_DIRECTION0|BIT_RS485_DIRECTION1);  //set  output
                    the bit RS485direction

  DDRD                                                          |=
                    (BIT_ZIGBEE_RESET|BIT_ENABLE_RXD_LINK_PC|BIT_ENA
                    BLE_RXD_LINK_ZIGBEE);

  PORTD &= ~_BV(BIT_LINK_PLUGIN); // no pull up
  PORTD |= _BV(BIT_ZIGBEE_RESET);
  PORTD |= _BV(BIT_ENABLE_RXD_LINK_PC);
  PORTD |= _BV(BIT_ENABLE_RXD_LINK_ZIGBEE);
}

/*
TxPacket() send data to RS485.
TxPacket()  needs  3  parameter;  ID  of  Dynamixel,  Instruction byte,
                    Length of parameters.
TxPacket() return length of Return packet from Dynamixel.
*/
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength)
{
    byte bCount,bCheckSum,bPacketLength;

    gbpTxBuffer[0] = 0xff;
    gbpTxBuffer[1] = 0xff;
    gbpTxBuffer[2] = bID;
    gbpTxBuffer[3]                =                bParameterLength+2;
                    //Length(Paramter,Instruction,Checksum)
    gbpTxBuffer[4] = bInstruction;
    for(bCount = 0; bCount < bParameterLength; bCount++)
    {
        gbpTxBuffer[bCount+5] = gbpParameter[bCount];
    }
    bCheckSum = 0;
    bPacketLength = bParameterLength+4+2;
    for(bCount  =  2;  bCount  <  bPacketLength-1;  bCount++)  //except
                    0xff,checksum
    {
        bCheckSum += gbpTxBuffer[bCount];
    }
    gbpTxBuffer[bCount]  =  ~bCheckSum;  //Writing  Checksum  with  Bit
                    Inversion

    RS485_TXD;
    for(bCount = 0; bCount < bPacketLength; bCount++)
    {
        sbi(UCSR0A,6);//SET_TXD0_FINISH;
        TxD80(gbpTxBuffer[bCount]);
    }
    while(!CHECK_TXD0_FINISH); //Wait until TXD Shift register empty
    RS485_RXD;
    return(bPacketLength);
}

/*
RxPacket() read data from buffer.
RxPacket() need a Parameter; Total length of Return Packet.
RxPacket() return Length of Return Packet.
*/

byte RxPacket(byte bRxPacketLength)
{
#define RX_TIMEOUT_COUNT2   3000L
#define RX_TIMEOUT_COUNT1   (RX_TIMEOUT_COUNT2*10L)
  unsigned long ulCounter;
  byte bCount, bLength, bChecksum;
  byte bTimeout;

  bTimeout = 0;
  for(bCount = 0; bCount < bRxPacketLength; bCount++)
  {
    ulCounter = 0;
    while(gbRxBufferReadPointer == gbRxBufferWritePointer)
    {
      if(ulCounter++ > RX_TIMEOUT_COUNT1)
      {
        bTimeout = 1;
        break;
      }
    }
    if(bTimeout) break;
    gbpRxBuffer[bCount]                                                 =
                    gbpRxInterruptBuffer[gbRxBufferReadPointer++];
  }
  bLength = bCount;
  bChecksum = 0;

  if(gbpTxBuffer[2] != BROADCASTING_ID)
  {
    if(bTimeout && bRxPacketLength != 255)
    {
      TxDString("¥r¥n [Error:RxD Timeout]");
      CLEAR_BUFFER;
    }

    if(bLength > 3) //checking is available.
    {
      if(gbpRxBuffer[0] != 0xff || gbpRxBuffer[1] != 0xff )
      {
        TxDString("¥r¥n [Error:Wrong Header]");
        CLEAR_BUFFER;
        return 0;
      }
      if(gbpRxBuffer[2] != gbpTxBuffer[2] )
      {
        TxDString("¥r¥n [Error:TxID != RxID]");
        CLEAR_BUFFER;
        return 0;
      }
      if(gbpRxBuffer[3] != bLength-4)
      {
        TxDString("¥r¥n [Error:Wrong Length]");
        CLEAR_BUFFER;
        return 0;
      }
      for(bCount  =  2;  bCount  <  bLength;  bCount++)  bChecksum  +=
                    gbpRxBuffer[bCount];
      if(bChecksum != 0xff)
      {
        TxDString("¥r¥n [Error:Wrong CheckSum]");
        CLEAR_BUFFER;
        return 0;
      }
    }
  }
  return bLength;
}


/*
PrintBuffer() print data in Hex code.
PrintBuffer()  needs  two  parameter;  name  of  Pointer(gbpTxBuffer,
                    gbpRxBuffer)
*/
void PrintBuffer(byte *bpPrintBuffer, byte bLength)
{
    byte bCount;
    for(bCount = 0; bCount < bLength; bCount++)
    {
        TxD8Hex(bpPrintBuffer[bCount]);
        TxD8(' ');
    }
    TxDString("(LEN:");TxD8Hex(bLength);TxD8(')');
}
```

```
/*
Print value of Baud Rate.
*/
void PrintBaudrate(void)
{
  TxDString("¥r¥n
                    RS232:");TxD32Dec((16000000L/8L)/((long)UBRR1L+1
                    L) );  TxDString(" BPS,");
  TxDString("    RS485:");TxD32Dec((16000000L/8L)/((long)UBRR0L+1L)  );
                    TxDString(" BPS");
}


/*Hardware Dependent Item*/
#define TXD1_READY                  bit_is_set(UCSR1A,5)
                    //(UCSR1A_Bit5)
#define TXD1_DATA                   (UDR1)
#define RXD1_READY                  bit_is_set(UCSR1A,7)
#define RXD1_DATA                   (UDR1)

#define TXD0_READY                  bit_is_set(UCSR0A,5)
#define TXD0_DATA                   (UDR0)
#define RXD0_READY                  bit_is_set(UCSR0A,7)
#define RXD0_DATA                   (UDR0)

/*
SerialInitialize() set Serial Port to initial state.
Vide Mega128 Data sheet about Setting bit of register.
SerialInitialize() needs port, Baud rate, Interrupt value.

*/
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt)
{
  if(bPort == SERIAL_PORT0)
  {
    UBRR0H = 0; UBRR0L = bBaudrate;
    UCSR0A = 0x02;  UCSR0B = 0x18;
    if(bInterrupt&RX_INTERRUPT) sbi(UCSR0B,7); // RxD interrupt enable
    UCSR0C = 0x06; UDR0 = 0xFF;
    sbi(UCSR0A,6);//SET_TXD0_FINISH; // Note. set 1, then 0 is read
  }
  else if(bPort == SERIAL_PORT1)
  {
    UBRR1H = 0; UBRR1L = bBaudrate;
    UCSR1A = 0x02;  UCSR1B = 0x18;
    if(bInterrupt&RX_INTERRUPT) sbi(UCSR1B,7); // RxD interrupt enable
    UCSR1C = 0x06; UDR1 = 0xFF;
    sbi(UCSR1A,6);//SET_TXD1_FINISH; // Note. set 1, then 0 is read
  }
}

/*
TxD8Hex() print data seperatly.
ex> 0x1a -> '1' 'a'.
*/
void TxD8Hex(byte bSentData)
{
  byte bTmp;

  bTmp =((byte)(bSentData>>4)&0x0f) + (byte)'0';
  if(bTmp > '9') bTmp += 7;
  TxD8(bTmp);
  bTmp =(byte)(bSentData & 0x0f) + (byte)'0';
  if(bTmp > '9') bTmp += 7;
  TxD8(bTmp);
}

/*
TxD80() send data to USART 0.
*/
void TxD80(byte bTxdData)
{
  while(!TXD0_READY);
  TXD0_DATA = bTxdData;
}

/*
TXD81() send data to USART 1.
*/
void TxD81(byte bTxdData)
{
  while(!TXD1_READY);
  TXD1_DATA = bTxdData;
}

/*
TXD32Dex() change data to decimal number system
*/
void TxD32Dec(long lLong)
{
  byte bCount, bPrinted;
  long lTmp,lDigit;
  bPrinted = 0;
  if(lLong < 0)
  {
    lLong = -lLong;
    TxD8('-');
  }
  lDigit = 1000000000L;
  for(bCount = 0; bCount < 9; bCount++)
  {
    lTmp = (byte)(lLong/lDigit);
    if(lTmp)
    {
      TxD8(((byte)lTmp)+'0');
      bPrinted = 1;
    }
    else if(bPrinted) TxD8(((byte)lTmp)+'0');
    lLong -= ((long)lTmp)*lDigit;
    lDigit = lDigit/10;
  }
  lTmp = (byte)(lLong/lDigit);
  /*if(lTmp)*/ TxD8(((byte)lTmp)+'0');
}

/*
TxDString() prints data in ACSII code.
*/
void TxDString(byte *bData)
{
  while(*bData)
  {
    TxD8(*bData++);
  }
}

/*
RxD81() read data from UART1.
RxD81() return Read data.
*/
byte RxD81(void)
{
  while(!RXD1_READY);
  return(RXD1_DATA);
}

/*
SIGNAL() UART0 Rx Interrupt - write data to buffer
*/
SIGNAL (SIG_UART0_RECV)
{
  gbpRxInterruptBuffer[(gbRxBufferWritePointer++)] = RXD0_DATA;
}
```

Connector

Company Name : Molex

Pin Number: 4

Model Number

	Molex Part Number	Old Part Number
Male	22-03-5045	5267-03
Female	50-37-5043	5264-03

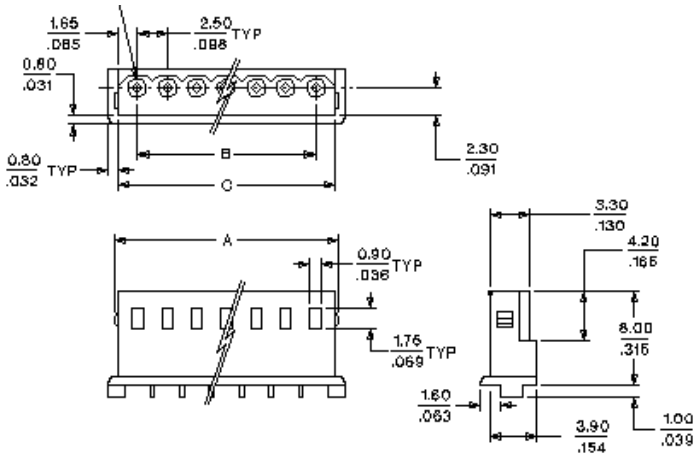
Temperature range : -40°C to +105°C

Contact Insertion Force-max : 14.7N (3.30 lb)

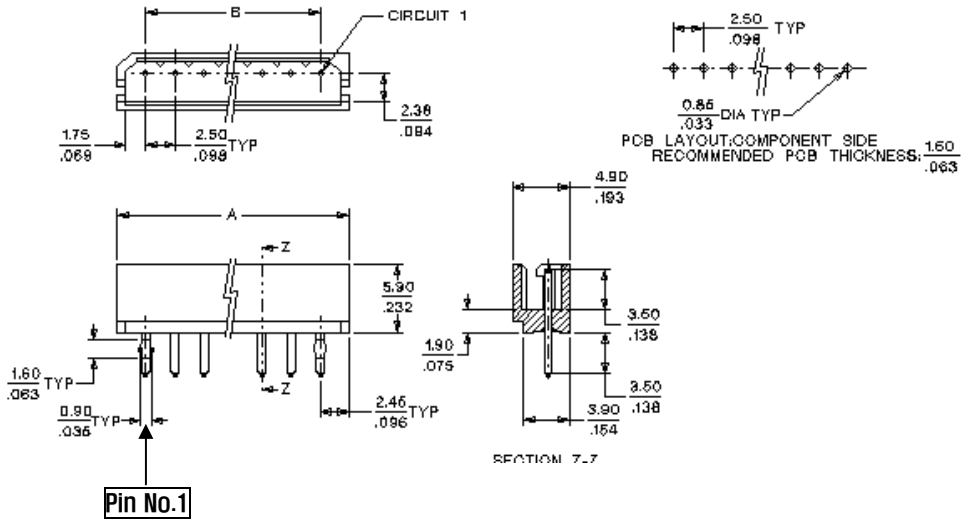
Contact Retention Force-min : 14.7N (3.30 lb)

www.molex.com or www.molex.co.jp for more detail information

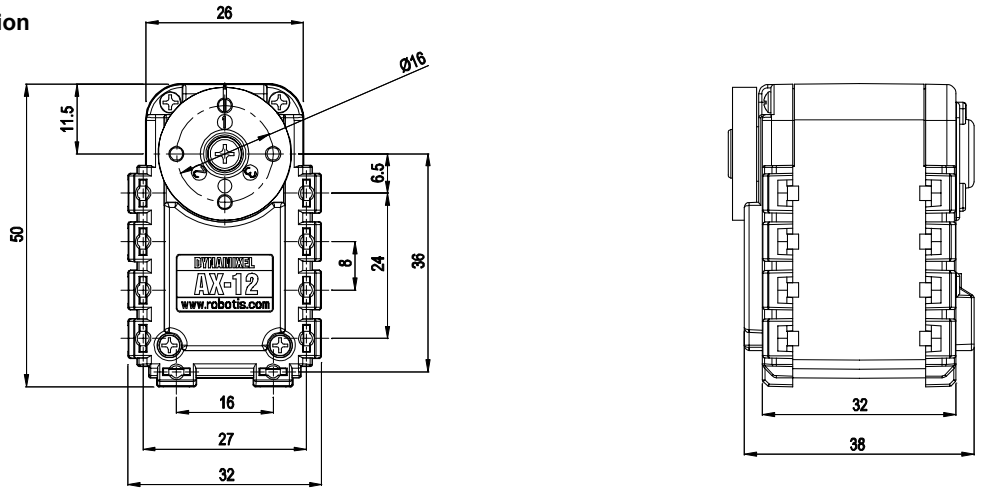
Female Connector



Male Connector



Dimension



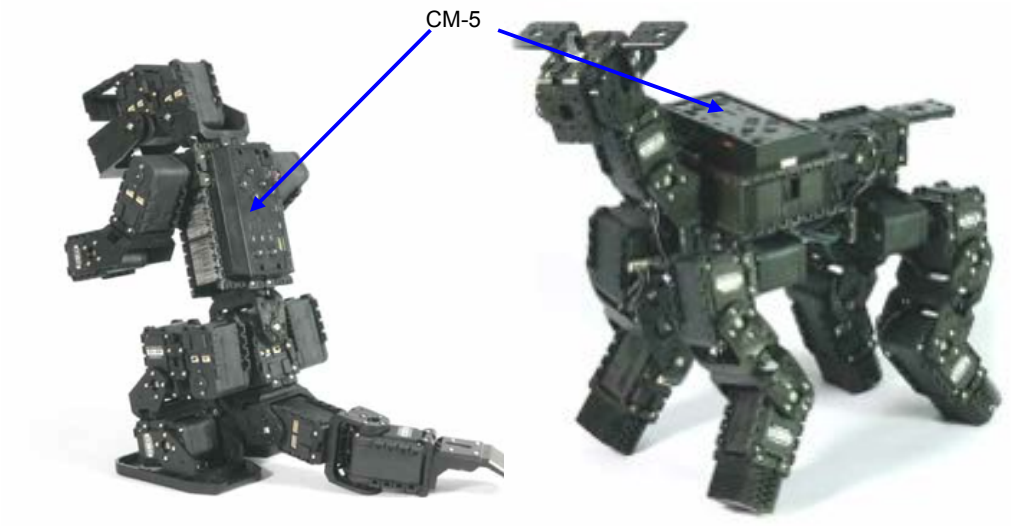
CM-5

Dedicated AX-12 control box. Able to control 30 AX-12 actuators.

6 push buttons (5 for selection, 1 for reset)

Optional installable wireless devices available

Battery compartment (AA x 8) with recharging capability (when connected to an external SMPS)



L Datablad AX-S1

User's Manual 2006-06-14

Contents

1. Dynamixel AX-S1

1-1. Overview and Characteristics of AX-S1

Dynamixel AX-S1 Dynamixel Sensor Module 'AX-S1' is a Smart Sensor Module that integrates the functions of sound sensor, infrared remote control receiver, infrared distance sensor, light sensor, buzzer, as well as the driver, control unit and network. Compact in size, AX-S1 has various functions and it is made up of special materials that can withstand even the extreme external force. In addition, it can readily recognize subtle changes such as internal temperature, service voltage and other internal conditions and has built-in capability to resolve the situations at hand. Followings are the strengths of the Dynamixel Sensor Module AX-S1.

Precision Control Capability to read sensor that has been detected through 1024 steps resolution

Feedback Feedback capabilities for the values of infrared distance sensor, light sensor, sound sensor.

Alarm System Alarm system that detects out of the range values of internal temperature, torque, service voltage were preset by users (Alarming)

Communication Wiring is easy with daisy chain connection, and it support communication speeds up to 1M BPS.

Distributed Control Position, velocity, compliance, and torque can be set with a single command packet, thus enabling the main processor to control many Dynamixel units even with very few resources.

Engineering Plastic The main body of the unit is made with high quality engineering plastic which enables it to handle high torque loads.

Frames Hinge and side mount frame are included as basics. AX-S1 is compatible with AX-12 frames 100%, making it possible to use in various ways. Be cautious as unlike AX-12, Horn part of AX-S1 does not turn, so assemble frame in correct angle with the usage purpose in mind.

Infra-red Sensor It is embedded with three directions infrared sensor, making it possible to detect left/center/right distance angle as well as the light.

Remocon Sensor It has built-in remote control sensor in center, making it possible to transmit and receive infrared data between sensor modules.

Internal Mic It has built-in micro internal microphone, making it possible not only to detect current sound level and maximum loudness but also an ability to count the number of sounds, for instance, the numbers of handclapping

Buzzer Built-in buzzer allows the playback of musical notes and other special note effects.

1-2. Main Specifications

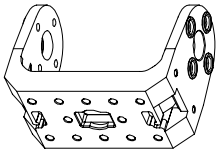
Dynamixel
Networked Sensor Module AX-S1 for Robot Application

Weight	37g
Resolution	10bit (1024)
Voltage	7V~10V (Recommended voltage: 9.6V)
Supply Current	40mA
Operate Temperature	-5℃ ~ +85℃
Command Signal	Digital Packet
Protocol Type	Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity)
Link (Physical)	TTL Level Multi Drop (daisy chain type Connector)
ID	254 ID (0~253)
Communication Speed	7343bps ~ 1 Mbps
Feedback	Infra-red Sensor, Internal Mic, Temperature, Input Voltage, IR Remocon Tx/Rx Data, etc.
Material	Engineering Plastic

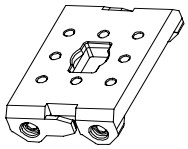
2. Dynamixel Operation

2-1. Mechanical Assembly

Frames Provided The two frames provided with AX-S1 are shown below.

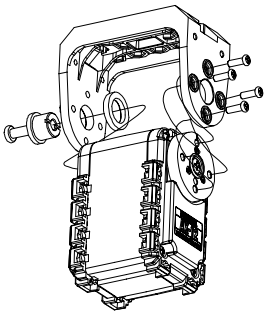


OF-12SH

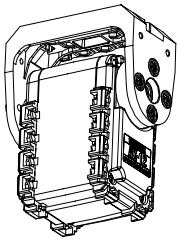


OF-12S

OF-12SH Installation The OF-12SH (hinge frame) can be installed on the AX-12 as the following.



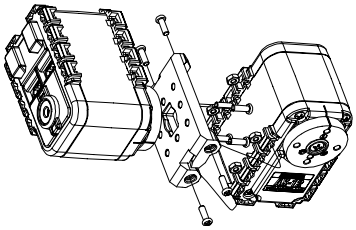
Exploded view



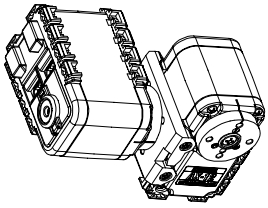
Assembled

OF-12S Installation The OF-12S (side mount frame) can be installed on the AX-12 as the following. The OF-12S can be mounted on any of the three faces (left, right, or under side) of the AX-12 body as needed.

Horn2Body

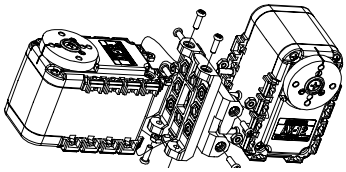


Exploded view

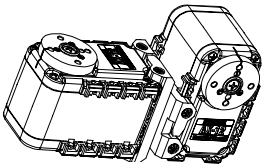


Assembled

Body2Body



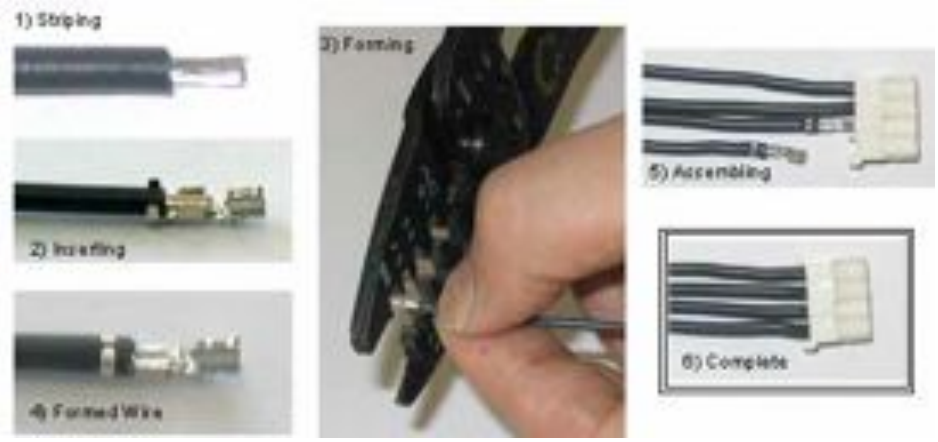
Exploded view



Assembled

2-2. Connector Assembly

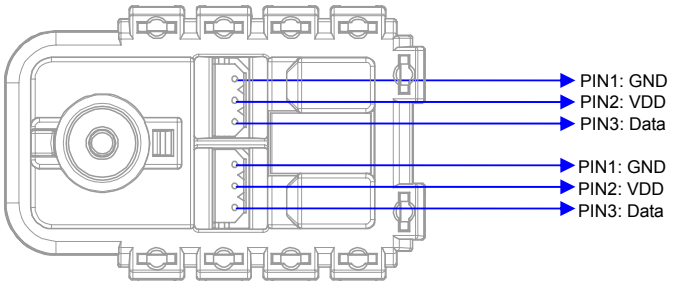
Assemble the connectors as shown below. Attach the wires to the terminals using the correct crimping tool. If you do not have access to a crimping tool, solder the terminals to the wires to ensure that they do not become loose during operation.



2-3. Dynamixel Wiring

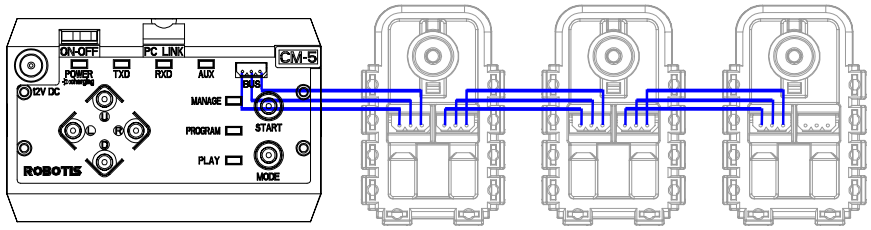
Pin Assignment

The connector pin assignments are as the following. The two connectors on the Dynamixel are connected pin to pin, thus the AX-S1 can be operated with only one connector attached.



Wiring

Connect the AX-2 actuators pin to pin as shown below. Many AX-S1 andAX-12 actuators can be controlled with a single bus in this manner.



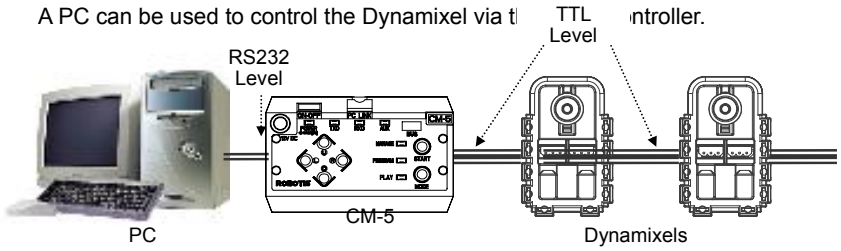
Control Box “CM-5”

Main Controller

To operate the Dynamixel actuators, the main controller must support TTL level half duplex UART. A proprietary controller can be used, but the use of the Dynamixel controller CM-5 is recommended.

PC LINK

A PC can be used to control the Dynamixel via t ntroller.



Bioloid

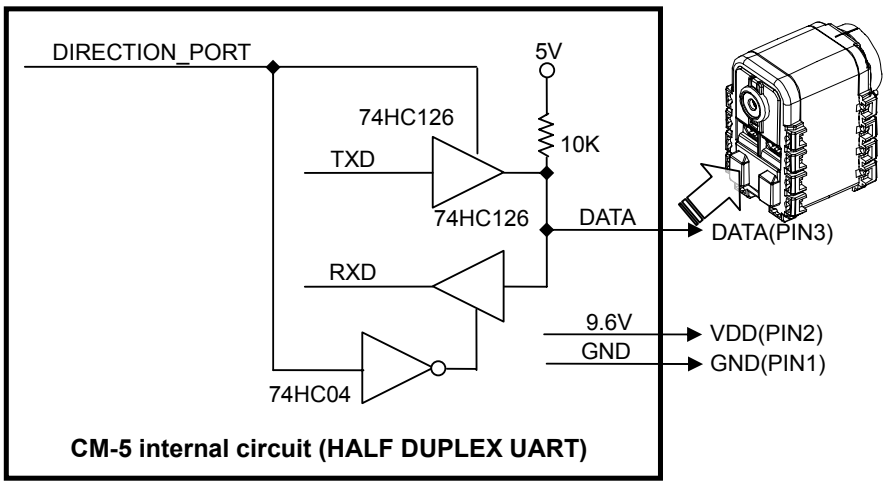
A robot can be built using only the CM-5 controller, a number of AX-12 actuators and AX-S1. An edutainment robotic kit named “Bioloid” is available which is based on the CM-5 controller, the AX-12 actuators and AX-S1

An example of a robot built with Bioloid

For details, please refer to the Bioloid manual.

Connection to UART

To control the Dynamixel actuators, the main controller needs to convert its UART signals to the half duplex type. The recommended circuit diagram for this is shown below.



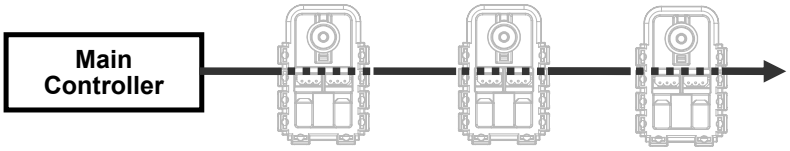
CM-5 internal circuit (HALF DUPLEX UART)

The power is supplied to the Dynamixel actuator from the main controller through Pin 1 and Pin 2 of the Molex3P connector. (The circuit shown above is presented only to explain the use of half duplex UART. The CM-5 controller already has the above circuitry built in, thus the Dynamixel actuators can be directly connected to it)

The direction of data signals on the TTL level TxD and RxD depends on the DIRECTION_PORT level as the following.

• When the DIRECTION_PORT level is High: the signal TxD is output as Data
• When the DIRECTION_PORT level is Low: the signal Data is input as RxD

Half Duplex UART A multi-drop method of connecting multiple Dynamixel actuators to a single node is possible by using the half duplex UART. Thus a protocol that does not allow multiple transmissions at the same time should be maintained when controlling the Dynamixel actuators.



[Multi Drop Link]

Caution Please ensure that the pin assignments are correct when connecting the Dynamixel actuators. Check the current consumption when powering on. The current consumption of a single Dynamixel actuator unit in standby mode should be no larger than 50mA

Connection Status Verification

When power is applied to the Dynamixel actuator, the LED blinks twice to confirm its connection.

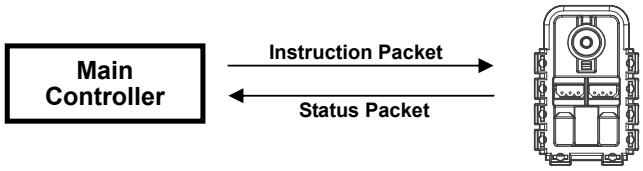
Inspection If the above operation was not successful, then check the connector pin assignment and the voltage/current limit of the power supply.

3. Communication Protocol

3-1. Communication Overview

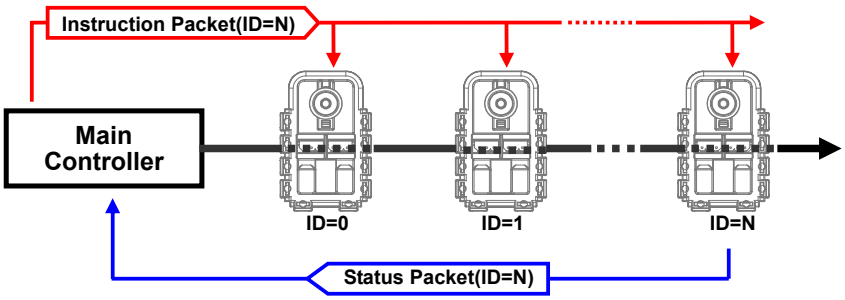
Packet

The main controller communicates with the Dynamixel units by sending and receiving data packets. There are two types of packets; the “Instruction Packet” (sent from the main controller to the Dynamixel actuators) and the “Status Packet” (sent from the Dynamixel actuators to the main controller.)



Communication

For the system connection below, if the main controller sends an instruction packet with the ID set to N, only the Dynamixel unit with this ID value will return its respective status packet and perform the required instruction



Unique ID

If multiple Dynamixel units have the same ID value, multiple packets sent simultaneously collide, resulting in communication problems. Thus, it is imperative that no Dynamixel units share the same ID in a network node.

Protocol

The Dynamixel actuators communicate through asynchronous serial communication with 8 bit, 1 stop bit and no parity.

3-2. Instruction Packet

The Instruction Packet is the packet sent by the main controller to the Dynamixel units to send commands. The structure of the Instruction Packet is as the following.

Instruction Packet | OXFF | 0XFF | ID | LENGTH | INSTRUCTION | PARAMETER1 | PARAMETER N | CHECK SUM |

The meanings of each packet byte definition are as the following.

0XFF 0XFF The two 0XFF bytes indicate the start of an incoming packet.

ID The unique ID of a Dynamixel unit. There are 254 available ID values, ranging from 0X00 to 0XFD.

Broadcasting ID ID 0XFE is the Broadcasting ID which indicates all of the connected Dynamixel units. Packets sent with this ID apply to all Dynamixel units on the network. Thus packets sent with a broadcasting ID will not return any status packets.

LENGTH The length of the packet where its value is “Number of parameters (N) + 2”

INSTRUCTION The instruction for the Dynamixel actuator to perform.

PARAMETER0 N Used if there is additional information needed to be sent other than the instruction itself.

CHECK SUM The computation method for the ‘Check Sum’ is as the following.

Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)
If the calculated value is larger than 255, the lower byte is defined as the checksum value.
~ represents the NOT logic operation.

3-3. Status Packet(Return Packet)

The Status Packet is the response packet from the Dynamixel units to the Main Controller after receiving an instruction packet. The structure of the status packet is as the following.

OXFF 0XFF ID LENGTH ERROR PARAMETER1 PARAMETER2 PARAMETER N CHECK SUM

The meanings of each packet byte definition are as the following.

0XFF 0XFF The two 0XFF bytes indicate the start of the packet.

ID The unique ID of the Dynamixel unit returning the packet.

LENGTH The length of the packet where its value is “Number of parameters (N) + 2”

ERROR The byte representing ERROR sent from the Dynamixel unit. The meaning of each bit is as the following.

Bit	Name	Details
Bit 7	0	-
Bit 6	Instruction Error	Set to 1 if an undefined instruction is sent or an action instruction is sent without a Reg_Write instruction.
Bit 5	0	
Bit 4	Checksum Error	Set to 1 if the checksum of the instruction packet is incorrect
Bit 3	Range Error	Set to 1 if the instruction sent is out of the defined range
Bit 2	Overheating Error	Set to 1 if the internal temperature of the Dynamixel unit is above the operating temperature range as defined in the control table.
Bit 1	Angle Limit Error	Set as 1 if the Goal Position is set outside of the range between CW Angle Limit and CCW Angle Limit.
Bit 0	Input Voltage Error	Set to 1 if the voltage is out of the operating voltage range as defined in the control table.

PARAMETER0 N Used if additional information is needed

CHECK SUM

The computation method for the 'Check Sum' is as the following.

Check Sum = ~ (ID + Length + Instruction + Parameter1 + ... Parameter N)

If the calculated value is larger than 255, the lower byte is defined as the checksum value. ~ represents the NOT logic operation.

3-4. Control Table

	Address	Item	Access	Initial Value
EEPROM Area	0(0X00)	Model Number(L)	RD	13(0x0D)
	1(0X01)	Model Number(H)	RD	0(0x00)
	2(0X02)	Version of Firmware	RD	?
	3(0X03)	ID	RD,WR	100(0x64)
	4(0X04)	Baud Rate	RD,WR	1(0x01)
	5(0X05)	Return Delay Time	RD,WR	250(0xFA)
	6(0X06)	(Reserved)	RD,WR	255(0xFF)
	7(0X07)	(Reserved)	RD,WR	3(0x03)
	8(0X08)	(Reserved)	RD,WR	255(0xFF)
	9(0X09)	(Reserved)	RD,WR	3(0x03)
	10(0x0A)	(Reserved)	-	0(0x00)
	11(0X0B)	the Highest Limit Temperature	RD,WR	100(0x64)
	12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
	13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
	14(0X0E)	(Reserved)	RD,WR	255(0XFF)
	15(0X0F)	(Reserved)	RD,WR	3(0x03)
	16(0X10)	Status Return Level	RD,WR	2(0x02)
	17(0X11)	(Reserved)	RD,WR	4(0x04)
	18(0X12)	(Reserved)	RD,WR	4(0x04)
	19(0X13)	(Reserved)	RD,WR	0(0x00)
	20(0X14)	Obstacle Detected Compare Value	RD,WR	32(0x20)
	21(0X15)	Light Detected Compare Value	RD,WR	32(0x20)
	22(0X16)	(Reserved)	RD,WR	32(0x20)
	23(0X17)	(Reserved)	RD	3(0x03)
RAM Area	24(0X18)	(Reserved)	RD,WR	0(0x00)
	25(0X19)	(Reserved)	RD,WR	0(0x00)
	26(0X1A)	Left IR Sensor Data	RD	?
	27(0X1B)	Center IR Sensor Data	RD	?
	28(0X1C)	Right IR Sensor Data	RD	?
	29(0X1D)	Left Luminosity	RD	?
	30(0X1E)	Center Luminosity	RD	?
	31(0X1F)	Right Luminosity	RD	?
	32(0X20)	Obstacle Detection Flag	RD	?
	33(0X21)	Luminosity Detection Flag	RD	?
	34(0X22)	(Reserved)	RD,WR	0
	35(0X23)	Sound Data	RD,WR	?
	36(0X24)	Sound Data Max Hold	RD,WR	?
	37(0X25)	Sound Detected Count	RD,WR	?
	38(0X26)	Sound Detected Time(L)	RD,WR	?
	39(0X27)	Sound Detected Time(H)	RD,WR	?
	40(0X28)	Buzzer Index	RD,WR	?
	41(0X29)	Buzzer Time	RD,WR	?
	42(0X2A)	Present Voltage	RD	?
	43(0X2B)	Present Temperature	RD	?
	44(0X2C)	Registered Instruction	RD,WR	0(0x00)
	45(0X2D)	(Reserved)	-	0(0x00)
	46(0x2E)	IR Remocon Arrived	RD	0(0x00)
	47(0x2F)	Lock	RD,WR	0(0x00)
	48(0x30)	IR Remocon RX Data 0	RD	?
	49(0x31)	IR Remocon RX Data 1	RD	?
	50(0x32)	IR Remocon TX Data 0	RD,WR	?
	51(0x33)	IR Remocon TX Data 1	RD,WR	?
	52(0x34)	Obstacle Detected Compare	RD,WR	?
	53(0x35)	Light Detected Compare	RD,WR	?

Control Table The Control Table contains information on the status and operation of the Dynamixel actuator. The Dynamixel actuator is operated by writing values to its control table and its status is checked by reading values off its control table.

RAM and EEPROM The data values for the RAM area will be set to the default initial values whenever the power is turned on. However, the data values for the EEPROM area are non-volatile and will still remain even after the power is turned off.

Initial Value The Initial Value column on the right side of the control table shows the Factory Default Values for the case of EEPROM area data, and shows the initial value when the power is turned on for the case of RAM area data.

The following explains the meaning of data stored in each of the addresses in the control table.

Address 0x00,0x01 **Model Number**. For AX-S1, the value is 0X000D(13).

Address 0x02 **Firmware Version**.

Address 0x03 **ID**. The unique ID number assigned to each Dynamixel actuators for identifying them. Different IDs are required for each Dynamixel actuators that are on the same network.

Address 0x04 **Baud Rate.** Determines the communication speed. The computation is done by the following formula.

Speed (BPS) = 2000000 / (Address4 + 1)

Data Value for each Major Baud Rate

Address4	Set BPS	Goal BPS	Error
1	1000000.0	1000000.0	0.000%
3	500000.0	500000.0	0.000%
4	400000.0	400000.0	0.000%
7	250000.0	250000.0	0.000%
9	200000.0	200000.0	0.000%
16	117647.1	115200.0	-2.124%
34	57142.9	57600.0	0.794%
103	19230.8	19200.0	-0.160%
207	9615.4	9600.0	-0.160%

Note A maximum Baud Rate error of 3% is within the tolerance of UART communication.

Address 0x05 **Return Delay Time.** The time it takes for the Status Packet to return after the Instruction Packet is sent. The delay time is given by 2uSec * Address5 value.

Address 0x0B **the Highest Limit Temperature.** The upper limit of the Dynamixel actuator's operating temperature. If the internal temperature of the Dynamixel actuator gets higher than this value, the Over Heating Error Bit (Bit 2 of the Status Packet) will return the value 1, and an alarm will be set by Address 17, 18. The values are in Degrees Celsius

Address 0x0C,0x0D **the Lowest (Highest) Limit Voltage.** The upper and lower limits of the Dynamixel actuator's operating voltage. If the present voltage (Address 42) is out of the specified range, a Voltage Range Error Bit (Bit 0 of the Status Packet) will return the value 1, and an alarm will be set by Address 17, 18. The values are 10 times the actual voltage value. For example, if the Address 12 value is 80, then the lower voltage limit is set to 8V.

Address 0X10 **Status Return Level.** Determines whether the Dynamixel actuator will return a Status Packet after receiving an Instruction Packet.

Address16	Returning the Status Packet
0	Do not respond to any instructions
1	Respond only to READ_DATA instructions
2	Respond to all instructions

In the case of an instruction which uses the Broadcast ID (0XFE) the Status Packet will not be returned regardless of the Address 0x10 value.

Address 0x14 **Obstacle Detected Compare Value** Dynamixel Sensor Module sets the standard value for the object detection that is in the direct line of object sensor parameter. If the infrared sensor value is greater than a standard value, as it indicates an obstacle within the set distance, the bit is set to a value of "1" in corresponding to sensor of IR Obstacle Detected, Address 0x20, and conversely, when the sensor value is lower than a standard value, it is set to "0."

The Obstacle Detected Compare Value is allocated in the ROM (Address 0x14) and RAM (Address 0x34) and when the power switched on, the value of EEPROM is copied to RAM.

Address 0x15 **Light Detected Compare Value** Dynamixel Sensor Module sets the standard value for the light detection that is in the direct line of infrared sensor parameter. If the light sensor value is greater than a standard value, as it indicates a light that is brighter than set light parameter, the bit is set to a value of “1” in corresponding to sensor of Light Detected, and conversely, it is set to “0” when it is lower than a standard value.

The Light Detected Compare Value is allocated in the ROM (Address 0x15) and RAM (Address 0x35) and when the power switched on, the value of EEPROM is copied to RAM.

Subsequent Address 0x18 is in RAM domain.

Address 0x1A~0x1C **Infrared Sensor Data (Left/Center/Right)** It is the infrared sensor value of the Dynamixel Sensor Module for measuring distance. The infrared sensor of AX-S1 emits high frequency Infrared and the emitted ray bounces off an object or wall to return to the IR sensor. The Infrared receiver of AX-S1 measures amount of infrared returned. High value will be acquired when an object or wall is near the sensor. Measured value ranges from 0~255. Only 255 will be acquired until a certain distance. Due to the innate properties of infrared measurement method, value of reflected Infrared ray amount might differ depending on the color of an object or surface texture.

Address 0x1D~0x1F **Luminosity (Left/Center/Right)** It is the light sensor value of the Dynamixel Sensor Module. The technological concept is similar to the infrared sensor. However, this sensor only measures amount of infrared ray emitted from source of illumination. Therefore, light sensor value can be measured from illuminations, such as incandescent bulb, emitting large amount of infrared. Lighter or candle light can be measured from short distance as well. Measured value ranges from 0~255.

Address 0x20

Obstacle Detection Flag When the value of infrared distance sensor becomes larger than the Obstacle Detected Compare Value, the AX-S1 recognizes existence of an object and sets object detection bit to 1. Refer to the below table for bit representation of each sensor.

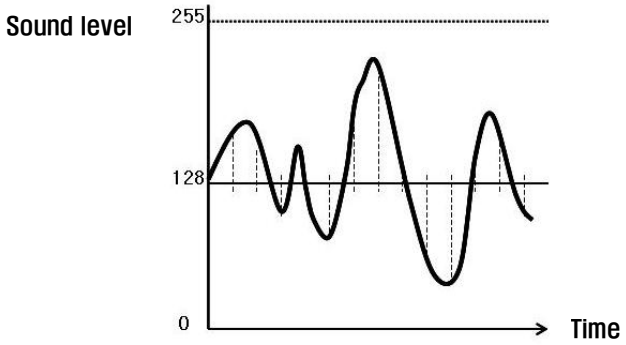
Bit	Representation
Bit 2	An object is detected on the Right Sensor /Light Detected
Bit 1	An object is detected on the Center Sensor /Light Detected
Bit 0	An object is detected on the Left Sensor /Light Detected

Address 0x21

Luminosity Detection Flag When the value of light sensor becomes larger than the light detected compare value, the AX-S1 recognizes existence of source of illumination and sets luminosity detection flag bit to 1. Bit representation of each sensor is the same with bit representation of object detection flag setting. (Refer to Address 0x20)

Address 0x23

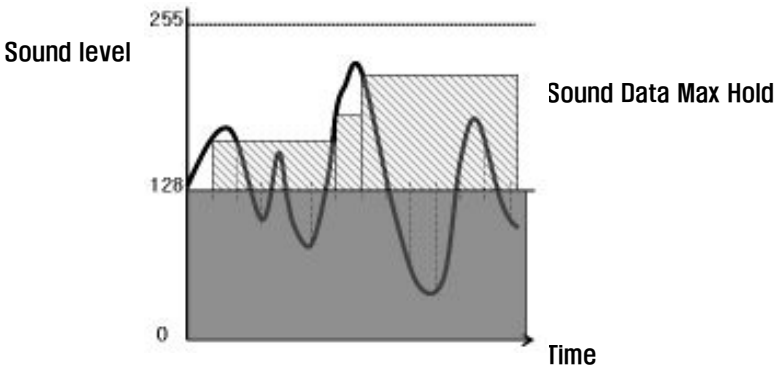
Sound Data It represents intensity of sound waves detected through the microphone of AX-S1. As shown in the illustration below, the magnitude of sound wave fluctuates. Value measured during noiseless state is around 127~128 (0x7F~0x80) and value ranging from 0 to 255 (0xFF) will be measured for noisy state. Sound wave will be measured at the frequency of 3800 input per second.



Address 0x24

Sound Data Max Hold AX-S1 has put aside a value for loudest sound. That is, when the present sound data exceeds the Sound Data Max Hold value, the present sound data will be copied as the Sound Data Max Hold.

Therefore, sound data less than 128 will be ignored and loudest sound intensity will be updated. Below illustration explains the details.



Be cautious as the Sound Data Max Hold is 255 (0xFF) and there is no value that can represent intensity of loudness greater than the optimal loudness, and thus, 255 (0xFF) will be maintained as the Sound Data Max Hold.

Therefore, value of the Sound Data Max Hold should be set at “0” for measuring the value of maximum loudness,

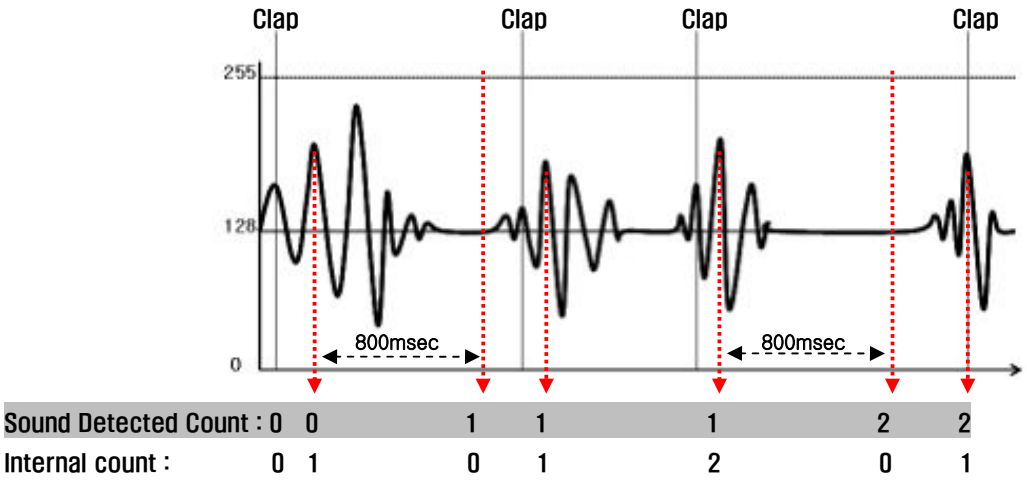
Address 0x25

Sound Detected Count AX-S1 has a counter that counts occurrence of loud sound

exceeding standard level. As an example, number of handclap can be counted by using this.

However, the counter will not count for next 80msec after counting once to prevent a single handclap to be recognized as multiple claps. 800msec after the last count, the value of sound detection frequency counter will be saved.

Timeline of sound detection frequency will be counted internally and then the value of sound detection frequency will be saved after 800msec. After saving, the sound detection frequency value will reset to 0. Below illustration explains the details.

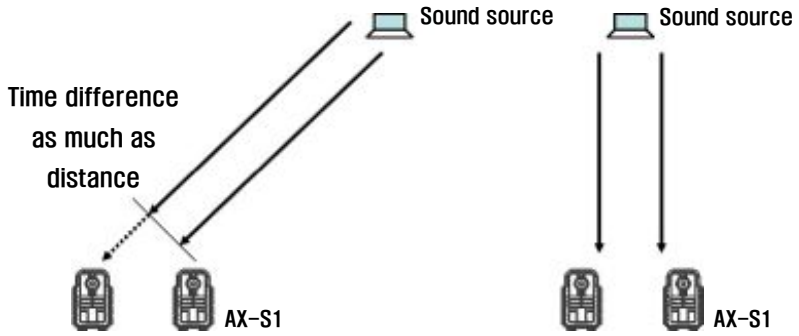


Address 0x26, 0x27 **Sound Detected Time** Anytime Sensor Module AX-S1 counts of sound detection, it saves the time of sound occurrences. This function exists to detect the direction of sound, and thus, it needs at least two AX-S1s; and by using speed of sound (around 343m/sec in 20℃) it uses the time differences of sound arrival in microphone of two AX-S1s.

Sound Detected Time is internally counted (counts 0~65535 repeatedly) and anytime Sound Detected Count is increased, it saves the counted value. Therefore, by placing the two AX-S1s in appropriate distance, and by simultaneously using broadcasting command and initializing to 0 value, the time differenced sound occurs corresponding to sound direction.

If the placement is face to face, the time differences will be almost simultaneous,

however, for the placement that has been set in side angle, the time differences will be influenced by the distances of AX-S1s. With this concept, it can estimate the direction of the sound. Below are detailed illustrations.



(1) Sound from angle (2) Sound from front

It counts completely every 4.096msec and it recounts again from 0. Therefore, in calculating the sound of speed, for every count, sound moves 0.02mm and two AX-S1's distance must be within 70cm.

For example, when two AX-S1s is 10cm apart, by using above method estimation, two AX-S1s' sound detected time difference can be maximum of 5,000. (If it is 5,000, it signifies that sound source is completely from the 90 angle or from the right side.)

Address 0x28

Buzzer Index All AX-S1 has built-in buzzer and thus, can playback the simple notes. Buzzer can play up to 52 notes and as it has whole and semitone in each octave, it can playback various melody sounds. The buzzer index value is assigned as follows.

Buzzer index	Melody notes	Buzzer index	Melody notes	Buzzer index	Melody notes	Buzzer index	Melody notes
0	la	13	la#	26	si	39	do
1	la#	14	si	27	do	40	do#
2	si	15	do	28	do#	41	re
3	do	16	do#	29	re	42	re#
4	do#	17	re	30	re#	43	mi
5	re	18	re#	31	mi	44	fa
6	re#	19	mi	32	fa	45	fa#
7	mi	20	fa	33	fa#	46	sol
8	fa	21	fa#	34	sol	47	sol#
9	fa#	22	sol	35	sol#	48	la

10	sol	23	sol#	36	la	49	la#
11	sol#	24	la	37	la#	50	si
12	la	25	la#	38	si	51	do

Address 0x29 **Buzzer Time** AX-S1 has a capability that controls the time interval of buzzer sound. Controllable within 0.1 second unit, the minimum length of time is 0.3 second and the maximum length of time is 5.0 seconds. That is, if user inputs the value of 0~3, the buzzer goes off in 0.3 second, whereas, if the input value is 50 or above, it goes off in 5 second. When the buzzer sound completes, the value automatically initializes back to 0.

There are two special features of AX-S1 buzzer time.

First is the function that sets the buzzer to sound constantly. If user inputs value of 254 on buzzer time and input the melody note number on buzzer index, the buzzer sounds the note constantly. To stop the buzzer, input 0 on buzzer time.

The second function plays back the special notes. If user inputs value of 255 on buzzer time and value between 0~26 on buzzer index, 27 various melodies is replayed corresponding to each number. When the melody playback is finished, the value automatically initializes back to 0.

Address 0x2A **Present Voltage.** Currently authorized voltage of Dynamixel AX-S1. It reality, it is multiple of 10 of actual voltage. That is, if 10V, it is read as 100(0x64).

Address 0x2B **Present Temperature.** Inner Celsius temperature of Dynamixel AX-S1

Address 0x2C **Registered Instruction.** If it is registered by the command of REG_WRITE, it is set to 1, and if it is registered by Action command, it is changed to 0 after command is completed.

Address 0x2E **IR Remocon Arrived** **AX-**S1 Sensor Module has infrared sensor module built-in in center and thus, it allows infrared remocon communication between AX-S1's. 2 byte transmission is possible.

Be cautious, however, as the infrared emitter is built into left/center/right, it can transmit infrared remocon in all directions, but, as infrared remocon sensor is built in only in center, its remocon data transmission is limited to certain angle.

I When Infrared remocon data is received by sensor, IR Remocon Arrived value changes to 2, signaling 2 byte transmission. If you read IR Remocon RX data, the IR Remocon

and automatically initializes back to 0.

Address 0x2F **Lock.** If the setting is set to 1, it can only write in range from Address 0X18 to Address0x23 and writing to other ranges is forbidden. Once it is locked, it can be unlocked only after power off. (power down)

Address 0x30,0x31 **IR Remocon RX Data** Address where data from infrared remocon sensor is saved. It reads the value and the IR Remocon Arrived value automatically initializes back to 0.

Address 0x32,0x33 **IR Remocon TX Data** Address where remocon data that will be transmitted via infrared emitter is written to. Upon writing of 2 byte value, remocon data is immediately transmitted.

Address 0x34 **Obstacle Detected Compare Value** Control Table RAM Range where obstacle detected compare value of Address 0x14 is saved.

The IR sensors of AX-S1 emit powerful infrared rays to detect an object at a long distance. It is impossible to detect an object in a short distance around 5cm since it always has maximum value in short distance

To prevent this, AX-S1 support low sensitive mode to detect precise value in a short distance. If the Obstacle Detected Compare Value is 0, it converts to low sensitive mode. The low sensitive mode has very weak long-distance sensing capability but it is possible to detect precise and sensitive short-distance detection not to saturate maximum value.

Address 0x35 **Light Detected Compare Value** Control Table RAM Range where light detected compare value of Address 0x15 is saved

Range

Each data has set value where their valid range is defined. Outside of this range, their write command will return Error. Below table indicates the length for writing and its range. 16 bit data is indicated (L) and (H) and as 2 byte. This 2 byte must be written as one in instruction packet.

Write Address	Writing Item	Length (bytes)	Min	Max
3(0X03)	ID	1	0	253(0xfd)
4(0X04)	Baud Rate	1	0	254(0xfe)
5(0X05)	Return Delay Time	1	0	254(0xfe)
11(0X0B)	the Highest Limit Temperature	1	0	150(0x96)
12(0X0C)	the Lowest Limit Voltage	1	50(0x32)	250(0xfa)
13(0X0D)	the Highest Limit Voltage	1	50(0x32)	250(0xfa)
16(0X10)	Status Return Level	1	0	2
17(0X11)	Alarm LED	1	0	127(0x7f)
18(0X12)	Alarm Shutdown	1	0	127(0x7f)
19(0X13)	(Reserved)	1	0	1
20(0X14)	Obstacle Detected Compare	1	0	255(0xff)
21(0X15)	Light Detected Compare	1	0	255(0xff)
36(0X24)	Sound Data Max Hold	1	0	255(0xff)
37(0X25)	Sound Detected Count	1	0	255(0xff)
38(0X26)	Sound Detected Time	2	0	65535(0xffff)
40(0X28)	Buzzer Index	1	0	255(0xff)
41(0X29)	Buzzer Time	1	0	255(0xff)
44(0X2C)	Registered Instruction	1	0	1
47(0X2F)	Lock	1	1	1
50(0X32)	IR Remocon TX Data	2	0	65535(0xffff)

[Control Table Data Range and Length for Writing]

4. Instruction Set and Examples

The following Instructions are available.

Instruction	Function	Value	Number of Parameter
PING	No action. Used for obtaining a Status Packet	0x01	0
READ DATA	Reading values in the Control Table	0x02	2
WRITE DATA	Writing values to the Control Table	0x03	2 ~
REG WRITE	Similar to WRITE_DATA, but stays in standby mode until the ACION instruction is given	0x04	2 ~
ACTION	Triggers the action registered by the REG_WRITE instruction	0x05	0
RESET	Changes the control table values of the Dynamixel actuator to the Factory Default Value settings	0x06	0
SYNC WRITE	Used for controlling many Dynamixel actuators at the same time	0x83	4~

4-1. WRITE_DATA

Function To write data into the control table of the Dynamixel actuator

Length N+3 (N is the number of data to be written)

Instruction 0X03

Parameter1 Starting address of the location where the data is to be written

Parameter2 1st data to be written

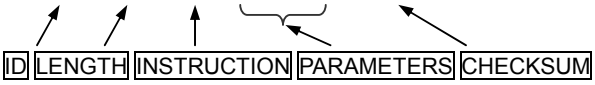
Parameter3 2nd data to be written

Parameter N+1 Nth data to be written

Example 1 **Setting the ID of a connected Dynamixel actuator to 1**

Write 1 to address 3 of the control table. The ID is transmitted using the Broadcasting ID (0xFE).

Instruction Packet : 0XFF 0XFF 0XFE 0X04 0X03 0X03 0X01 0XF6`



Because it was transmitted with a Broadcast ID (0XFE), no status packets are returned.

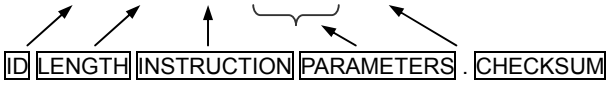
4-2. READ_DATA

Function	Read data from the control table of a Dynamixel actuator
Length	0X04
Instruction	0X02
Parameter1	Starting address of the location where the data is to be read
Parameter2	Length of the data to be read

Example 2 **Reading the internal temperature of the Dynamixel actuator with an ID of 1**

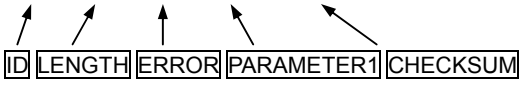
Read 1 byte from address 0x2B of the control table.

Instruction Packet : 0XFF 0XFF 0X01 0X04 0X02 0X2B 0X01 0XCC`



The returned Status Packet will be as the following.

Status Packet : 0XFF 0XFF 0X01 0X03 0X00 0X20 0XDB



The data read is 0x20. Thus the current internal temperature of the Dynamixel actuator is approximately 32°C (0X20).

4-3. REG_WRITE과 ACTION

4-3-1. REG_WRITE

Function The REG_WRITE instruction is similar to the WRITE_DATA instruction, but the execution timing is different. When the Instruction Packet is received the values are stored in the Buffer and the Write instruction is under a standby status. At this time, the Registered Instruction register (Address 0x2C) is set to 1. After the Action Instruction Packet is received, the registered Write instruction is finally executed.

Length N+3 (N is the number of data to be written)
Instruction 0X04
Parameter1 Starting address of the location where the data is to be written
Parameter2 1st data to be written
Parameter3 2nd data to be written
Parameter N+1 Nth data to be written

4-3-2. ACTION

Function Triggers the action registered by the REG_WRITE instruction
Length 0X02
Instruction 0X05
Parameter NONE

The ACTION instruction is useful when multiple Dynamixel actuators need to move simultaneously. When controlling multiple Dynamixel actuator units, slight time delays can occur between the 1st and last units to receive an instruction. The Dynamixel actuator handles this problem by using the ACTION instruction.

Broadcasting The Broadcast ID (0XFE) is used when sending ACTION instructions to more than two Dynamixel actuators. Note that no packets are returned by this operation.

4-4. PING

Function Does not command any operations. Used for requesting a status packet or to check the existence of a Dynamixel actuator with a specific ID.

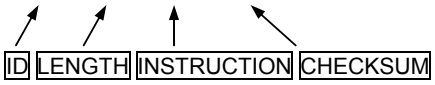
Length 0X02

Instruction 0X01

Parameter NONE

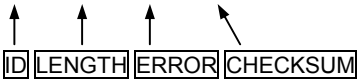
Example 3 **Obtaining the status packet of the Dynamixel actuator with an ID of 1**

Instruction Packet : 0XFF 0XFF 0X01 0X02 0X01 0XFB`



The returned Status Packet is as the following

Status Packet : 0XFF 0XFF 0X01 0X02 0X00 0XFC



Regardless of whether the Broadcasting ID is used or the Status Return Level (Address 16) is 0, a Status Packet is always returned by the PING instruction.

4-5. RESET

Function Changes the control table values of the Dynamixel actuator to the Factory Default Value settings

Length 0X02

Instruction 0X06

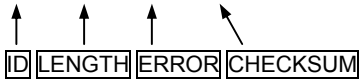
Parameter NONE

Resetting the Dynamixel actuator with an ID of 0

Instruction Packet : 0XFF 0XFF 0X00 0X02 0X06 0XF7`

The returned Status Packet is as the following

Status Packet : 0XFF 0XFF 0X00 0X02 0X00 0XFD



Note the ID of this Dynamixel actuator is now changed to 1 after the RESET instruction

4-6. SYNC WRITE

Function	Used for controlling many Dynamixel actuators at the same time. The communication time decreases by the Synch Write instruction since many instructions can be transmitted by a single instruction. However, you can use this instruction only when the lengths and addresses of the control table to be written to are the same. Also, the broadcasting ID needs to be used for transmitting.
ID	0XFE
Length	(L + 1) * N + 4 (L: Data length for each Dynamixel actuator, N: The number of Dynamixel actuators)
Instruction	0X83
Parameter1	Starting address of the location where the data is to be written
Parameter2	The length of the data to be written (L)
Parameter3	The ID of the 1st Dynamixel actuator
Parameter4	The 1st data for the 1st Dynamixel actuator
Parameter5	The 2nd data for the 1st Dynamixel actuator
Parameter L+3	The Lth data for the 1st Dynamixel actuator
Parameter L+4	The ID of the 2nd Dynamixel actuator
Parameter L+5	The 1st data for the 2nd Dynamixel actuator
Parameter L+6	The 2nd data for the 2nd Dynamixel actuator
Parameter 2L+4	The Lth data for the 2nd Dynamixel actuator

Data for the 1st Dynamixel actuator

Data for the 2nd Dynamixel actuator

Example 5 **Setting the following positions and velocities for 4 Dynamixel actuators**

Dynamixel actuator with an ID of 0: to position 0X010 with a speed of 0X150
Dynamixel actuator with an ID of 1: to position 0X220 with a speed of 0X360
Dynamixel actuator with an ID of 2: to position 0X030 with a speed of 0X170
Dynamixel actuator with an ID of 0: to position 0X220 with a speed of 0X380

Instruction Packet : 0XFF 0XFF 0XFE 0X18 0X83 0X1E 0X04 0X00 0X10 0X00 0X50 0X01 0X01 0X20 0X02 0X60 0X03 0X02 0X30 0X00 0X70 0X01 0X03 0X20 0X02 0X80 0X03 0X12
No status packets are returned since the Broadcasting ID was used.

5. Example

We will give an example of Dynamixel AX-S1 with following setup parameter. Reset state ID=100, Baudrate = 1MBPS

.

Example 6 Dynamixel AX-S1 that has ID 100 reads the Model Number and Firmware Version

Instruction Packet Instruction = READ_DATA, Address = 0x00, Length = 0x03

Communication ->[Dynamixel]:FF FF 64 04 02 00 03 95 (LEN:008)
<-[Dynamixel]:FF FF 64 05 00 0D 00 12 77 (LEN:009)

Status Packet Result Model Number = 13(0x0D)(in case of AX-S1) Firmware Version = 0x12

Example 7 Dynamixel AX-S1 that has ID 100 changes ID to 0.

Instruction Packet Instruction = WRITE_DATA, Address = 0x03, DATA = 0x00

Communication ->[Dynamixel]:FF FF 64 04 03 03 00 91 (LEN:008)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

Example 8 Change the Baud Rate of Dynamixel to 57600 bps.

Instruction Packet Instruction = WRITE_DATA, Address = 0x04, DATA = 0x22

Communication ->[Dynamixel]:FF FF 64 04 03 04 22 6E (LEN:008)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

Example 9 Dynamixel that has ID 100 resets the Return Delay Time to 4uSec

Return Delay Time Value of 1 is applicable to 2uSec.

Instruction Packet Instruction = WRITE_DATA, Address = 0x05, DATA = 0x02

Communication ->[Dynamixel]:FF FF 64 04 03 05 02 8D (LEN:008)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

It is good idea to set the Return Delay Time to minimum value within allowable range in the main controller.

Example 10 Dynamixel that has ID 100 resets the distance sensor standard value to 60.

Instruction Packet Instruction = WRITE_DATA, Address = 0x34, DATA = 0x3C

Communication ->[Dynamixel]:FF FF 64 04 03 34 3C 24 (LEN:008)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

Example 11 Dynamixel that has ID 100 resets the maximum value of temperature to 80°

Instruction Packet Instruction = WRITE_DATA, Address = 0x0B, DATA = 0x50

Communication ->[Dynamixel]:FF FF 64 04 03 0B 50 39 (LEN:008)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

Example 12 Dynamixel that has ID 100 sets the voltage to 10V ~ 17V.

10V is represented by 100 (0x64), and 17V by 170 (0xAA).

Instruction Packet Instruction = WRITE_DATA, Address = 0x0C, DATA = 0x64, 0xAA

Communication ->[Dynamixel]:FF FF 64 05 03 0C 64 AA 79 (LEN:009)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

Example 13 Dynamixel that has ID 100 changes the light sensor standard value to 10..

Instruction Packet Instruction = WRITE_DATA, Address = 0x35, DATA = 0x0A

Communication ->[Dynamixel]:FF FF 64 04 03 35 0A 55 (LEN:08)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

Example 14 Dynamixel that has ID 100 sets the parameter so that status packet is never returned.

Instruction Packet Instruction = WRITE_DATA, Address = 0x10, DATA = 0x00

Communication ->[Dynamixel]:FF FF 64 04 03 10 00 84 (LEN:008)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR
Status packet is not returned from next instruction.

Example 15 Dynamixel AX-S1 that has ID100 reads the right distance sensor value

Instruction Packet Instruction = READ_DATA, Address = 0x1C, DATA = 0x01

Communication ->[Dynamixel]:FF FF 64 04 02 1C 01 78 (LEN:008)
<-[Dynamixel]:FF FF 64 03 00 21 77 (LEN:007)

Status Packet Result NO ERROR
The right distance sensor value is 0x21

Example 16 Dynamixel AX-S1 that has ID 100 reads the center light sensor value

Instruction Packet Instruction = READ_DATA, Address = 0x1E, DATA = 0x01

Communication ->[Dynamixel]:FF FF 64 04 02 1E 01 76 (LEN:008)
<-[Dynamixel]:FF FF 64 03 00 00 98 (LEN:007)

Status Packet Result NO ERROR
The center light sensor value is 0x00

Example 17 Dynamixel AX-S1 that has ID 100 reads the sound loudness

Instruction Packet Instruction = READ_DATA, Address = 0x23, DATA = 0x01

Communication ->[Dynamixel]:FF FF 64 04 02 23 01 71 (LEN:08)
<-[Dynamixel]:FF FF 64 03 00 7E 1A (LEN:007)

Status Packet Result NO ERROR
The sound loudness value is 0x7E (126)

Example 18 Dynamixel AX-S1 that has ID 100 reads the numbers of sound detect frequency

Instruction Packet Instruction = READ_DATA, Address = 0x25, DATA = 0x01

Communication ->[Dynamixel]:FF FF 64 04 02 25 01 6F (LEN:008)
<-[Dynamixel]:FF FF 64 03 00 02 96 (LEN:007)

Status Packet Result NO ERROR
The number of sound detect frequency is 2.

Example 19 Dynamixel AX-S1 that has ID 100 playbacks special melody 5 times through buzzer

.

Case 1. After writing 0xFF(255) on buzzer sound interval, it writes No. 5 on buzzer note melody.

Instruction Packet ID=100, Instruction = WRITE_DATA, Address = 0x29, DATA = 0xFF

ID=100, Instruction = WRITE_DATA, Address = 0x28, DATA = 0x05

Communication

->[Dynamixel]:FF FF 64 04 03 29 FF 6C (LEN:008)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)
->[Dynamixel]:FF FF 64 04 03 28 05 67 (LEN:008)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

Case 2. Writes buzzer note and buzzer sound interval simultaneously

Instruction Packet ID=100, Instruction = WRITE_DATA, Address = 0x28, DATA = 0x05, 0xFF

Communication

->[Dynamixel]:FF FF 64 05 03 28 05 FF 67 (LEN:009)
<-[Dynamixel]:FF FF 64 02 00 99 (LEN:006)

Status Packet Result NO ERROR

Example 20 Dynamixel that has ID 0 sets the parameter so that it cannot write anywhere except in Address0x18 ~ Address0x23

Instruction Packet Instruction = WRITE_DATA, Address = 0x2F, DATA = 0x01

Communication

->[Dynamixel]:FF FF 00 04 03 2F 01 C8 (LEN:008)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Once locked, the only way to unlock it is to remove the power.

If an attempt is made to access any locked data, an error is returned.

->[Dynamixel]:FF FF 00 05 03 30 40 00 87 (LEN:009)
<-[Dynamixel]:FF FF 00 02 08 F5 (LEN:006)

Range Error

Example 21 Dynamixel that has ID 0 sets the minimum output value (punch) to 0x40

Instruction Packet Instruction = WRITE_DATA, Address = 0x30, DATA = 0x40, 0x00

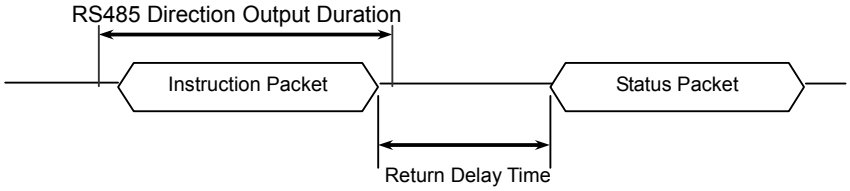
Communication ->[Dynamixel]:FF FF 00 05 03 30 40 00 87 (LEN:009)
<-[Dynamixel]:FF FF 00 02 00 FD (LEN:006)

Status Packet Result NO ERROR

Appendix

Half duplex UART

Half duplex UART is a serial communication protocol where both TxD and RxD cannot be used at the same time. This method is generally used when many devices need to be connected to a single bus. Since more than one device are connected to the same bus, all the other devices need to be in input mode while one device is transmitting. The Main Controller that controllers the Dynamixel actuators sets the communication direction to input mode, and only when it is transmitting an Instruction Packet, it changes the direction to output mode.



Return Delay Time

The time it takes for the Dynamixel actuator to return the Status Packet after receiving an Instruction Packet. The Default Value is 160 uSec and can be changed via the Control Table at Address 5. The Main Controller needs to change the Direction Port to input mode during the Return Delay Time after sending an instruction packet.

Tx,Rx Direction

For Half Duplex UART, the transmission ending timing is important to change the direction to receiving mode. The bit definitions within the register that indicates UART_STATUS are as the following

TXD_BUFFER_READY_BIT: Indicates that the transmission DATA can be loaded into the Buffer. Note that this only means that the SERIAL TX BUFFER is empty, and does not necessarily mean that the all the data transmitted before has left the CPU.

TXD_SHIFT_REGISTER_EMPTY_BIT: Set when all the Transmission Data has completed its transmission and left the CPU.

The TXD_BUFFER_READY_BIT is used when one byte is to be transmitted via the serial communication channel, and an example is shown below.

```
TxDByte(byte bData)
{
    while(!TXD_BUFFER_READY_BIT);    //wait until data can be loaded.
    SerialTxDBuffer = bData;         //data load to TxD buffer
}
```

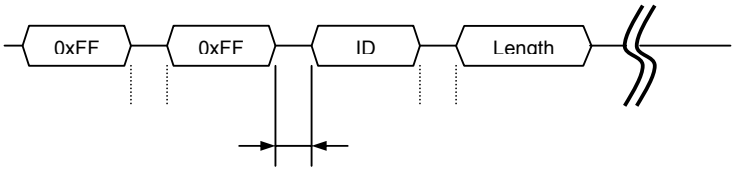
When changing the direction, the TXD_SHIFT_REGISTER_EMPTY_BIT must be checked.

The following is an example program that sends an Instruction Packet.

```
LINE 1    DIRECTION_PORT = TX_DIRECTION;
LINE 2    TxDByte(0xff);
LINE 3    TxDByte(0xff);
LINE 4    TxDByte(bID);
LINE 5    TxDByte(bLength);
LINE 6    TxDByte(bInstruction);
LINE 7    TxDByte(Parameter0);    TxDByte(Parameter1);
LINE 8    DisableInterrupt(); // interrupt should be disable
LINE 9    TxDByte(Checksum);    //last TxD
LINE 10   while(!TXD_SHIFT_REGISTER_EMPTY_BIT); //Wait till last data bit has been sent
LINE 11   DIRECTION_PORT = RX_DIRECTION; //Direction change to RXD
LINE 12   EnableInterrupt(); // enable interrupt again
```

Please note the important lines between LINE 8 and LINE 12. Line 8 is necessary since an interrupt here may cause a delay longer than the return delay time and corruption to the front of the status packet may occur.

Byte to Byte Time The delay time between bytes when sending an instruction packet. If the delay time is over 100ms, then the Dynamixel actuator recognizes this as a communication problem and waits for the next header (0xff 0xff) of a packet again.



Byte To Byte Time

The following is the source code of a program (Example.c) that accesses the Dynamixel actuator using the Atmega 128.

C Language Example : Dinamixel access with Atmega128

```
/*
 * The Example of Dynamixel Evaluation with Atmega128
 * Date : 2005.5.11
 * Author : BS KIM
 */

/*
 * included files
 */
#define ENABLE_BIT_DEFINITIONS
//#include <io.h>
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>

#define cbi(REG8,BITNUM) REG8 &= ~(_BV(BITNUM))
#define sbi(REG8,BITNUM) REG8 |= _BV(BITNUM)


typedef unsigned char byte;
typedef unsigned int word;
#define ON 1
#define OFF 0
#define _ON 0
#define _OFF 1


//--- Control Table Address ---
//EEPROM AREA
#define P_MODEL_NUMBER_L          0
#define P_MODOEL_NUMBER_H         1
#define P_VERSION                 2
#define P_ID                      3
#define P_BAUD_RATE               4
#define P_RETURN_DELAY_TIME       5
#define P_CW_ANGLE_LIMIT_L        6
#define P_CW_ANGLE_LIMIT_H        7
#define P_CCW_ANGLE_LIMIT_L       8
#define P_CCW_ANGLE_LIMIT_H       9
#define P_SYSTEM_DATA2            10
#define P_LIMIT_TEMPERATURE       11
#define P_DOWN_LIMIT_VOLTAGE      12
#define P_UP_LIMIT_VOLTAGE        13
#define P_MAX_TORQUE_L            14
#define P_MAX_TORQUE_H            15
#define P_RETURN_LEVEL            16
#define P_ALARM_LED               17
#define P_ALARM_SHUTDOWN          18
#define P_OPERATING_MODE          19
#define P_DOWN_CALIBRATION_L      20
#define P_DOWN_CALIBRATION_H      21
#define P_UP_CALIBRATION_L        22
#define P_UP_CALIBRATION_H        23

#define P_TORQUE_ENABLE           (24)
#define P_LED                     (25)
#define P_CW_COMPLIANCE_MARGIN    (26)
#define P_CCW_COMPLIANCE_MARGIN   (27)
#define P_CW_COMPLIANCE_SLOPE     (28)
#define P_CCW_COMPLIANCE_SLOPE    (29)
#define P_GOAL_POSITION_L         (30)
#define P_GOAL_POSITION_H         (31)
#define P_GOAL_SPEED_L            (32)
#define P_GOAL_SPEED_H            (33)
#define P_TORQUE_LIMIT_L          (34)
#define P_TORQUE_LIMIT_H          (35)
#define P_PRESENT_POSITION_L      (36)
#define P_PRESENT_POSITION_H      (37)
#define P_PRESENT_SPEED_L         (38)
#define P_PRESENT_SPEED_H         (39)
#define P_PRESENT_LOAD_L          (40)
#define P_PRESENT_LOAD_H          (41)
#define P_PRESENT_VOLTAGE         (42)
#define P_PRESENT_TEMPERATURE     (43)
#define P_REGISTERED_INSTRUCTION  (44)
#define P_PAUSE_TIME              (45)
#define P_MOVING                  (46)
#define P_LOCK                    (47)
#define P_PUNCH_L                 (48)
#define P_PUNCH_H                 (49)

//--- Instruction ---
#define INST_PING                 0x01
#define INST_READ                 0x02
#define INST_WRITE                0x03
#define INST_REG_WRITE            0x04
#define INST_ACTION               0x05
#define INST_RESET                0x06
#define INST_DIGITAL_RESET        0x07
#define INST_SYSTEM_READ          0x0C
#define INST_SYSTEM_WRITE         0x0D
#define INST_SYNC_WRITE           0x83
#define INST_SYNC_REG_WRITE       0x84

#define CLEAR_BUFFER gbRxBufferReadPointer = gbRxBufferWritePointer
#define DEFAULT_RETURN_PACKET_SIZE 6
#define BROADCASTING_ID 0xfe

#define TxD8 TxD81
#define RxD8 RxD81

//Hardware Dependent Item
#define DEFAULT_BAUD_RATE 34      //57600bps at 16MHz

////// For CM-5
#define    RS485_TXD    PORTE    &=    ~_BV(PE3),PORTE    |=    _BV(PE2)
                        //PORT_485_DIRECTION = 1
#define    RS485_RXD    PORTE    &=    ~_BV(PE2),PORTE    |=    _BV(PE3)
                        //PORT_485_DIRECTION = 0
/*
////// For CM-2
#define RS485_TXD PORTE |= _BV(PE2); //_485_DIRECTION = 1
#define RS485_RXD PORTE &= ~_BV(PE2);//PORT_485_DIRECTION = 0
*/
//#define TXD0_FINISH    UCSR0A,6   //This bit is for checking TxD Buffer in
                                    CPU is empty or not.
//#define TXD1_FINISH    UCSR1A,6

#define SET_TxD0_FINISH     sbi(UCSR0A,6)
#define RESET_TXD0_FINISH cbi(UCSR0A,6)
#define CHECK_TXD0_FINISH bit_is_set(UCSR0A,6)
#define SET_TxD1_FINISH    sbi(UCSR1A,6)
#define RESET_TXD1_FINISH cbi(UCSR1A,6)
#define CHECK_TXD1_FINISH bit_is_set(UCSR1A,6)

#define RX_INTERRUPT 0x01
#define TX_INTERRUPT 0x02
#define OVERFLOW_INTERRUPT 0x01
#define SERIAL_PORT0 0
#define SERIAL_PORT1 1
#define BIT_RS485_DIRECTION0    0x08    //Port E
#define BIT_RS485_DIRECTION1    0x04    //Port E

#define BIT_ZIGBEE_RESET                PD4    //out : default 1 //PORTD
#define BIT_ENABLE_RXD_LINK_PC          PD5    //out : default 1
#define BIT_ENABLE_RXD_LINK_ZIGBEE      PD6    //out : default 0
#define BIT_LINK_PLUGIN                 PD7    //in, no pull up

void TxD81(byte bTxdData);
void TxD80(byte bTxdData);
void TxDString(byte *bData);
void TxD8Hex(byte bSentData);
void TxD32Dec(long lLong);
byte RxD81(void);
void MiliSec(word wDelayTime);
void PortInitialize(void);
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt);
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength);
byte RxPacket(byte bRxLength);
void PrintBuffer(byte *bpPrintBuffer, byte bLength);

// --- Gloval Variable Number ---
volatile byte gbpRxInterruptBuffer[256];
byte gbpParameter[128];
byte gbRxBufferReadPointer;
byte gbpRxBuffer[128];
byte gbpTxBuffer[128];
volatile byte gbRxBufferWritePointer;

int main(void)
{
    byte bCount,bID, bTxPacketLength,bRxPacketLength;

    PortInitialize(); //Port In/Out Direction Definition
    RS485_RXD; //Set RS485 Direction to Input State.
    SerialInitialize(SERIAL_PORT0,1,RX_INTERRUPT);//RS485
                                    Initializing(RxInterrupt)
    SerialInitialize(SERIAL_PORT1,DEFAULT_BAUD_RATE,0);          //RS232
```

```
                        Initializing(None Interrupt)

    gbRxBufferReadPointer = gbRxBufferWritePointer = 0;  //RS485 RxBuffer
                        Clearing.

    sei();  //Enable Interrupt -- Compiler Function
    TxDString("\r\n  [The  Example  of  Dynamixel  Evaluation  with
                        ATmega128,GCC-AVR]");

//Dynamixel Communication Function Execution Step.
// Step 1. Parameter Setting (gbpParameter[]). In case of no parameter
                        instruction(Ex. INST_PING), this step is not needed.
// Step 2. TxPacket(ID,INSTRUCTION,LengthOfParameter); --Total TxPacket
                        Length is returned
// Step 3. RxPacket(ExpectedReturnPacketLength); -- Real RxPacket Length is
                        returned
// Step 4 PrintBuffer(BufferStartPointer,LengthForPrinting);

    bID = 1;
    TxDString("\r\n\n  Example  1.  Scanning  Dynamixels(0~9).  --  Any  Key  to
                        Continue."); RxD8();
    for(bCount = 0; bCount < 0x0A; bCount++)
    {
      bTxPacketLength = TxPacket(bCount,INST_PING,0);
      bRxPacketLength = RxPacket(255);
      TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
      TxDString(", RxD:");    PrintBuffer(gbpRxBuffer,bRxPacketLength);
      if(bRxPacketLength == DEFAULT_RETURN_PACKET_SIZE)
      {
        TxDString(" Found!! ID:");TxD8Hex(bCount);
        bID = bCount;
      }
    }

    TxDString("\r\n\n  Example  2.  Read  Firmware  Version.  --  Any  Key  to
                        Continue."); RxD8();
    gbpParameter[0] = P_VERSION; //Address of Firmware Version
    gbpParameter[1] = 1; //Read Length
    bTxPacketLength = TxPacket(bID,INST_READ,2);
    bRxPacketLength                                                    =
                        RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpP
                        arameter[1]);
    TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString("\r\n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);
    if(bRxPacketLength                                                 ==
                        DEFAULT_RETURN_PACKET_SIZE+gbpParameter[
                        1])
    {
      TxDString("\r\n Return Error      : ");TxD8Hex(gbpRxBuffer[4]);
      TxDString("\r\n Firmware Version  : ");TxD8Hex(gbpRxBuffer[5]);
    }

    TxDString("\r\n\n Example 3. LED ON -- Any Key to Continue."); RxD8();
    gbpParameter[0] = P_LED; //Address of LED
    gbpParameter[1] = 1; //Writing Data
    bTxPacketLength = TxPacket(bID,INST_WRITE,2);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString("\r\n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

    TxDString("\r\n\n Example 4. LED OFF -- Any Key to Continue."); RxD8();
    gbpParameter[0] = P_LED; //Address of LED
    gbpParameter[1] = 0; //Writing Data
    bTxPacketLength = TxPacket(bID,INST_WRITE,2);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString("\r\n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

    TxDString("\r\n\n Example 5. Read Control Table. -- Any Key to Continue.");
                        RxD8();
    gbpParameter[0] = 0; //Reading Address
    gbpParameter[1] = 49; //Read Length
    bTxPacketLength = TxPacket(bID,INST_READ,2);
    bRxPacketLength                                                    =
                        RxPacket(DEFAULT_RETURN_PACKET_SIZE+gbpP
                        arameter[1]);

    TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString("\r\n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);
    if(bRxPacketLength                                                 ==
                        DEFAULT_RETURN_PACKET_SIZE+gbpParameter[
                        1])
    {
      TxDString("\r\n");
      for(bCount = 0; bCount < 49; bCount++)
      {
        TxD8('[');TxD8Hex(bCount);TxDString("]:");
                        TxD8Hex(gbpRxBuffer[bCount+5]);TxD8(' ');
      }
    }

    TxDString("\r\n\n Example 6. Go 0x200 with Speed 0x100 -- Any Key to
                        Continue."); RxD8();
    gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
    gbpParameter[1] = 0x00; //Writing Data P_GOAL_POSITION_L
    gbpParameter[2] = 0x02; //Writing Data P_GOAL_POSITION_H
    gbpParameter[3] = 0x00; //Writing Data P_GOAL_SPEED_L
    gbpParameter[4] = 0x01; //Writing Data P_GOAL_SPEED_H
    bTxPacketLength = TxPacket(bID,INST_WRITE,5);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString("\r\n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

    TxDString("\r\n\n Example 7. Go 0x00 with Speed 0x40 -- Any Key to
                        Continue."); RxD8();
    gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
    gbpParameter[1] = 0x00; //Writing Data P_GOAL_POSITION_L
    gbpParameter[2] = 0x00; //Writing Data P_GOAL_POSITION_H
    gbpParameter[3] = 0x40; //Writing Data P_GOAL_SPEED_L
    gbpParameter[4] = 0x00; //Writing Data P_GOAL_SPEED_H
    bTxPacketLength = TxPacket(bID,INST_WRITE,5);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString("\r\n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

    TxDString("\r\n\n Example 8. Go 0x3ff with Speed 0x3ff -- Any Key to
                        Continue."); RxD8();
    gbpParameter[0] = P_GOAL_POSITION_L; //Address of Firmware Version
    gbpParameter[1] = 0xff; //Writing Data P_GOAL_POSITION_L
    gbpParameter[2] = 0x03; //Writing Data P_GOAL_POSITION_H
    gbpParameter[3] = 0xff; //Writing Data P_GOAL_SPEED_L
    gbpParameter[4] = 0x03; //Writing Data P_GOAL_SPEED_H
    bTxPacketLength = TxPacket(bID,INST_WRITE,5);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString("\r\n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

    TxDString("\r\n\n Example 9. Torque Off -- Any Key to Continue."); RxD8();
    gbpParameter[0] = P_TORQUE_ENABLE; //Address of LED
    gbpParameter[1] = 0; //Writing Data
    bTxPacketLength = TxPacket(bID,INST_WRITE,2);
    bRxPacketLength = RxPacket(DEFAULT_RETURN_PACKET_SIZE);
    TxDString("\r\n TxD:"); PrintBuffer(gbpTxBuffer,bTxPacketLength);
    TxDString("\r\n RxD:"); PrintBuffer(gbpRxBuffer,bRxPacketLength);

    TxDString("\r\n\n End. Push reset button for repeat");
    while(1);
}

void PortInitialize(void)
{
    DDRA = DDRB = DDRC = DDRD = DDRE = DDRF = 0;  //Set all port to
                        input direction first.
    PORTB = PORTC = PORTD = PORTE = PORTF = PORTG = 0x00;
                        //PortData initialize to 0
    cbi(SFIOR,2); //All Port Pull Up ready
    DDRE  |=  (BIT_RS485_DIRECTION0|BIT_RS485_DIRECTION1);  //set
                        output the bit RS485direction

    DDRD                                                               |=
                        (BIT_ZIGBEE_RESET|BIT_ENABLE_RXD_LINK_PC|
                        BIT_ENABLE_RXD_LINK_ZIGBEE);

    PORTD &= ~_BV(BIT_LINK_PLUGIN); // no pull up
    PORTD |= _BV(BIT_ZIGBEE_RESET);
    PORTD |= _BV(BIT_ENABLE_RXD_LINK_PC);
    PORTD |= _BV(BIT_ENABLE_RXD_LINK_ZIGBEE);
}

/*
TxPacket() send data to RS485.
TxPacket() needs 3 parameter; ID of Dynamixel, Instruction byte, Length of
                        parameters.
TxPacket() return length of Return packet from Dynamixel.
*/
byte TxPacket(byte bID, byte bInstruction, byte bParameterLength)
{
    byte bCount,bCheckSum,bPacketLength;

    gbpTxBuffer[0] = 0xff;
    gbpTxBuffer[1] = 0xff;
    gbpTxBuffer[2] = bID;
    gbpTxBuffer[3]                  =                  bParameterLength+2;
                        //Length(Paramter,Instruction,Checksum)
```

```
    gbpTxBuffer[4] = bInstruction;
    for(bCount = 0; bCount < bParameterLength; bCount++)
    {
        gbpTxBuffer[bCount+5] = gbpParameter[bCount];
    }
    bCheckSum = 0;
    bPacketLength = bParameterLength+4+2;
    for(bCount = 2; bCount < bPacketLength-1; bCount++) //except 0xff,checksum
    {
        bCheckSum += gbpTxBuffer[bCount];
    }
    gbpTxBuffer[bCount] = ~bCheckSum; //Writing Checksum with Bit Inversion

    RS485_TXD;
    for(bCount = 0; bCount < bPacketLength; bCount++)
    {
        sbi(UCSR0A,6);//SET_TXD0_FINISH;
        TxD80(gbpTxBuffer[bCount]);
    }
    while(!CHECK_TXD0_FINISH); //Wait until TXD Shift register empty
    RS485_RXD;
    return(bPacketLength);
}

/*
RxPacket() read data from buffer.
RxPacket() need a Parameter; Total length of Return Packet.
RxPacket() return Length of Return Packet.
*/

byte RxPacket(byte bRxPacketLength)
{
#define RX_TIMEOUT_COUNT2     3000L
#define RX_TIMEOUT_COUNT1    (RX_TIMEOUT_COUNT2*10L)
  unsigned long ulCounter;
  byte bCount, bLength, bChecksum;
  byte bTimeout;

  bTimeout = 0;
  for(bCount = 0; bCount < bRxPacketLength; bCount++)
  {
    ulCounter = 0;
    while(gbRxBufferReadPointer == gbRxBufferWritePointer)
    {
      if(ulCounter++ > RX_TIMEOUT_COUNT1)
      {
        bTimeout = 1;
        break;
      }
    }
    if(bTimeout) break;
    gbpRxBuffer[bCount] = gbpRxInterruptBuffer[gbRxBufferReadPointer++];
  }
  bLength = bCount;
  bChecksum = 0;

  if(gbpTxBuffer[2] != BROADCASTING_ID)
  {
    if(bTimeout && bRxPacketLength != 255)
    {
      TxDString("\r\n [Error:RxD Timeout]");
      CLEAR_BUFFER;
    }

    if(bLength > 3) //checking is available.
    {
      if(gbpRxBuffer[0] != 0xff || gbpRxBuffer[1] != 0xff )
      {
        TxDString("\r\n [Error:Wrong Header]");
        CLEAR_BUFFER;
        return 0;
      }
      if(gbpRxBuffer[2] != gbpTxBuffer[2] )
      {
        TxDString("\r\n [Error:TxID != RxID]");
        CLEAR_BUFFER;
        return 0;
      }
      if(gbpRxBuffer[3] != bLength-4)
      {
        TxDString("\r\n [Error:Wrong Length]");
        CLEAR_BUFFER;
        return 0;
      }
      for(bCount = 2; bCount < bLength; bCount++) bChecksum += gbpRxBuffer[bCount];
      if(bChecksum != 0xff)
      {
        TxDString("\r\n [Error:Wrong CheckSum]");
        CLEAR_BUFFER;
        return 0;
      }
    }
  }
  return bLength;
}


/*
PrintBuffer() print data in Hex code.
PrintBuffer() needs two parameter; name of Pointer(gbpTxBuffer, gbpRxBuffer)
*/
void PrintBuffer(byte *bpPrintBuffer, byte bLength)
{
    byte bCount;
    for(bCount = 0; bCount < bLength; bCount++)
    {
        TxD8Hex(bpPrintBuffer[bCount]);
        TxD8(' ');
    }
    TxDString("(LEN:");TxD8Hex(bLength);TxD8(')');
}

/*
Print value of Baud Rate.
*/
void PrintBaudrate(void)
{
  TxDString("\r\n  RS232:");TxD32Dec((16000000L/8L)/((long)UBRR1L+1L)  );
                        TxDString(" BPS,");
  TxDString("      RS485:");TxD32Dec((16000000L/8L)/((long)UBRR0L+1L)  );
                        TxDString(" BPS");
}


/*Hardware Dependent Item*/
#define TXD1_READY              bit_is_set(UCSR1A,5)
                        //(UCSR1A_Bit5)
#define TXD1_DATA               (UDR1)
#define RXD1_READY              bit_is_set(UCSR1A,7)
#define RXD1_DATA               (UDR1)

#define TXD0_READY              bit_is_set(UCSR0A,5)
#define TXD0_DATA               (UDR0)
#define RXD0_READY              bit_is_set(UCSR0A,7)
#define RXD0_DATA               (UDR0)

/*
SerialInitialize() set Serial Port to initial state.
Vide Mega128 Data sheet about Setting bit of register.
SerialInitialize() needs port, Baud rate, Interrupt value.

*/
void SerialInitialize(byte bPort, byte bBaudrate, byte bInterrupt)
{
  if(bPort == SERIAL_PORT0)
  {
    UBRR0H = 0; UBRR0L = bBaudrate;
    UCSR0A = 0x02;   UCSR0B = 0x18;
    if(bInterrupt&RX_INTERRUPT) sbi(UCSR0B,7); // RxD interrupt enable
    UCSR0C = 0x06; UDR0 = 0xFF;
    sbi(UCSR0A,6);//SET_TXD0_FINISH; // Note. set 1, then 0 is read
  }
  else if(bPort == SERIAL_PORT1)
  {
    UBRR1H = 0; UBRR1L = bBaudrate;
    UCSR1A = 0x02;   UCSR1B = 0x18;
    if(bInterrupt&RX_INTERRUPT) sbi(UCSR1B,7); // RxD interrupt enable
    UCSR1C = 0x06; UDR1 = 0xFF;
    sbi(UCSR1A,6);//SET_TXD1_FINISH; // Note. set 1, then 0 is read
  }
}

/*
TxD8Hex() print data seperatly.
ex> 0x1a -> '1' 'a'.
*/
void TxD8Hex(byte bSentData)
{
  byte bTmp;

  bTmp =((byte)(bSentData>>4)&0x0f) + (byte)'0';
```

```
    if(bTmp > '9') bTmp += 7;
    TxD8(bTmp);
    bTmp =(byte)(bSentData & 0x0f) + (byte)'0';
    if(bTmp > '9') bTmp += 7;
    TxD8(bTmp);
}

/*
TxD80() send data to USART 0.
*/
void TxD80(byte bTxdData)
{
    while(!TXD0_READY);
    TXD0_DATA = bTxdData;
}

/*
TXD81() send data to USART 1.
*/
void TxD81(byte bTxdData)
{
    while(!TXD1_READY);
    TXD1_DATA = bTxdData;
}

/*
TXD32Dex() change data to decimal number system
*/
void TxD32Dec(long lLong)
{
    byte bCount, bPrinted;
    long lTmp,lDigit;
    bPrinted = 0;
    if(lLong < 0)
    {
        lLong = -lLong;
        TxD8('-');
    }
    lDigit = 1000000000L;
    for(bCount = 0; bCount < 9; bCount++)
    {
        lTmp = (byte)(lLong/lDigit);
        if(lTmp)
        {
            TxD8(((byte)lTmp)+'0');
            bPrinted = 1;
        }
        else if(bPrinted) TxD8(((byte)lTmp)+'0');
        lLong -= ((long)lTmp)*lDigit;
        lDigit = lDigit/10;
    }
    lTmp = (byte)(lLong/lDigit);
    /*if(lTmp)*/ TxD8(((byte)lTmp)+'0');
}

/*
TxDString() prints data in ACSII code.
*/
void TxDString(byte *bData)
{
    while(*bData)
    {
        TxD8(*bData++);
    }
}

/*
RxD81() read data from UART1.
RxD81() return Read data.
*/
byte RxD81(void)
{
    while(!RXD1_READY);
    return(RXD1_DATA);
}

/*
SIGNAL() UART0 Rx Interrupt - write data to buffer
*/
SIGNAL (SIG_UART0_RECV)
{
    gbpRxInterruptBuffer[(gbRxBufferWritePointer++)] = RXD0_DATA;
}
```

Connector

Company Name : Molex

Pin Number: 3

Model Number

	Molex Part Number	Old Part Number
Male	22-03-5045	5267-03
Female	50-37-5043	5264-03

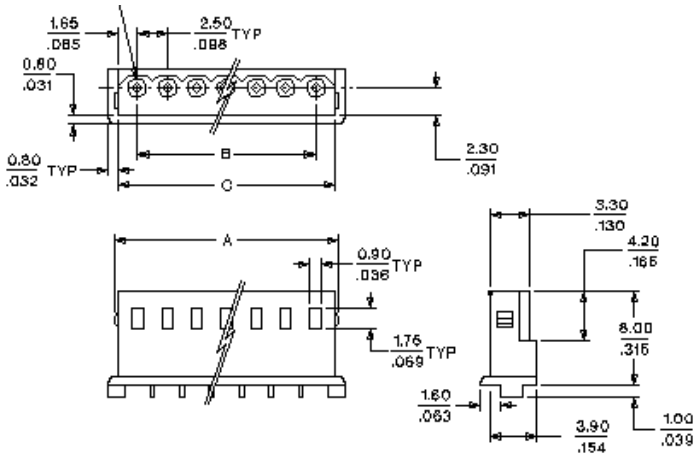
Temperature range : -40°C to +105°C

Contact Insertion Force-max : 14.7N (3.30 lb)

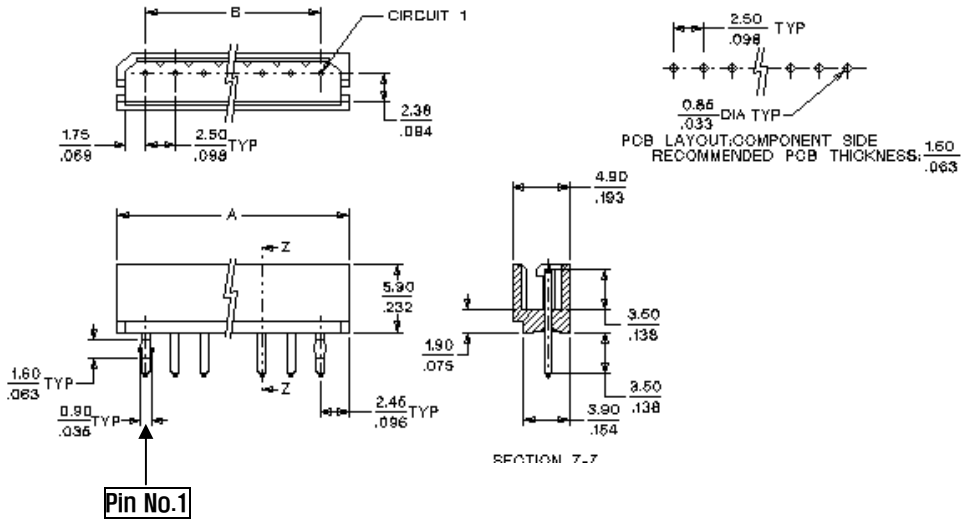
Contact Retention Force-min : 14.7N (3.30 lb)

www.molex.com or www.molex.co.jp for more detail information

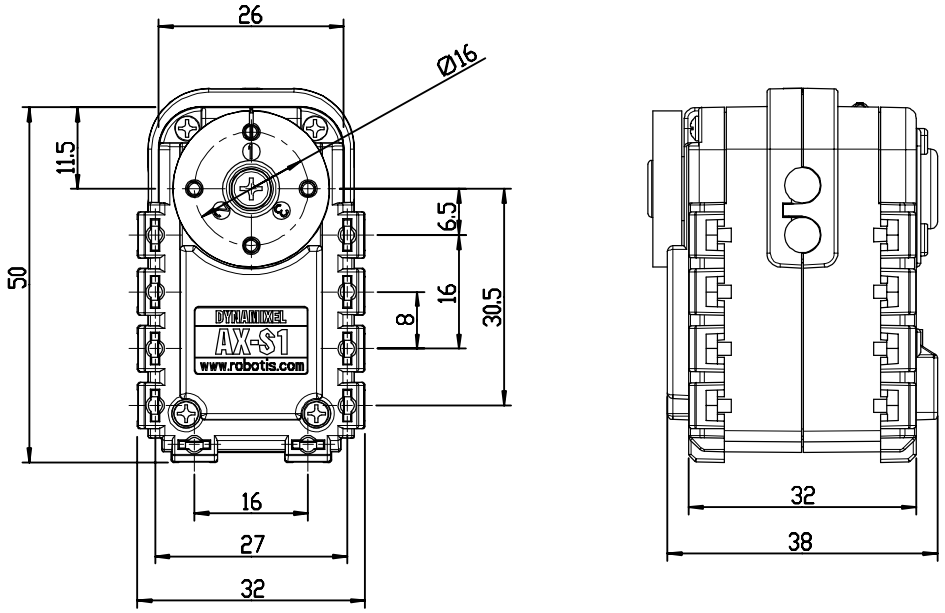
Female Connector



Male Connector

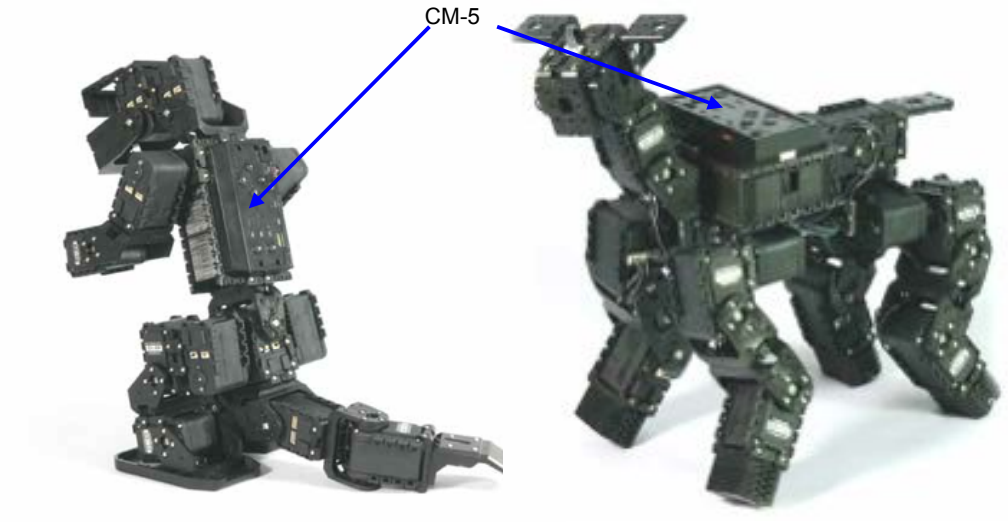


Dimension



CM-5

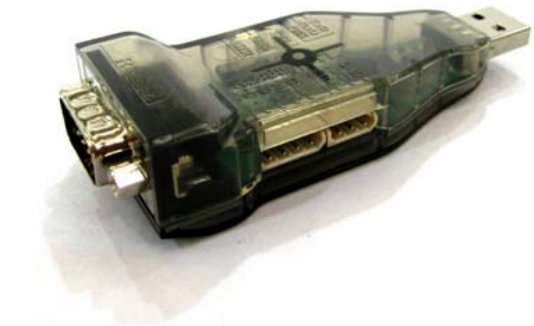
Dedicated AX-12, AX-S1 control box. Able to control 30 AX-12 actuators, 10 AX-S1.
6 push buttons (5 for selection, 1 for reset)
Optional installable wireless devices available
Battery compartment (AA x 8) with recharging capability (when connected to an external SMPS)



M Manual USB2Dynamixel

v1.2

ROBOTIS CO.,LTD. www.robotis.com



contents

1. Introduction

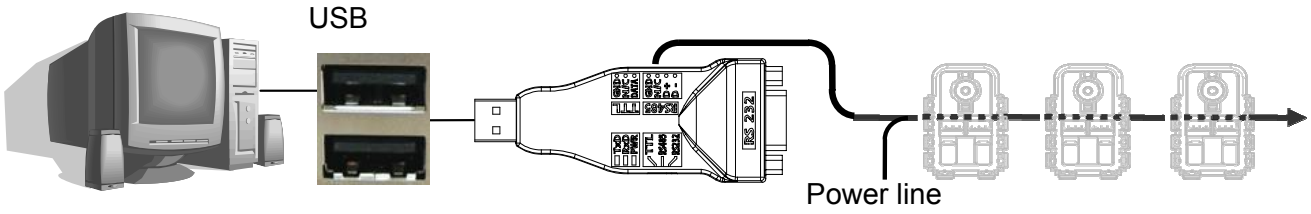
1-1. Functions

The USB2Dynamixel is a device used to directly drive the dynamixel on a PC. The USB2Dynamixel is used by connecting to a USB port of a PC, and is equipped with 3p and 4p connectors to be connected with a variety of dynamixels.

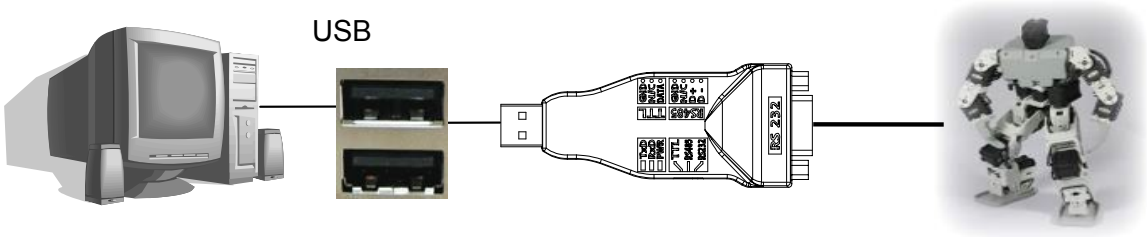
In addition, the USB2Dynamixel can be used to transform a USB port into a serial port for a PC without a serial port such as a lap-top computer. This function can be used effectively when dynamixel exclusive controllers such as the CM-2, CM-2+ and CM-5 are connected to an UBS port or when the ZIG2Serial is connected to an USB port to control a robot by radio.

Following pictures show how to use the USB2Dynamixel:

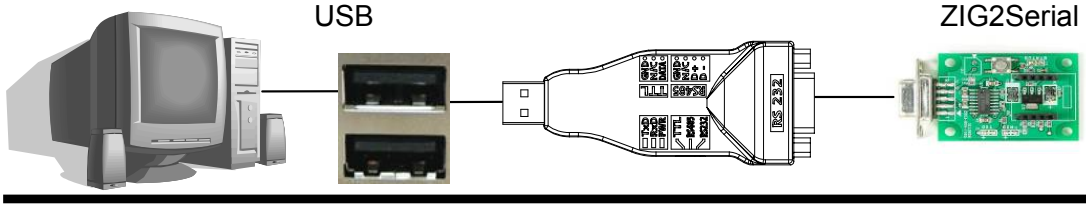
◎ Control of the USB2Dynamixel using a PC



◎ Transformation of a Serial Port



◎ Wireless Network



1-2. Composition

◎ **USB2Dynamixel**

The USB2Dynamixel is used when a dynamixel is directly driven on a PC without a dynamixel exclusive controller such as the CM-2, CM-2+ and CM-5. It is also used to transform an USB port to a serial port.

◎ **Software**

A software that can directly drive a dynamixel on a PC only with the USB2Dynamixel without a dynamixel exclusive controller such as CM-2, CM-2+ and CM-5 is provided. There are two types of software provided:

1. USB2Dynamixel driver
2. Dynamixel Manager

Downloadable from our website (www.robotis.com/usb2dynamixel)

◎ **Manual**

This manual describes how to install and use the USB2Dynamixel / Dynamixel Manager.

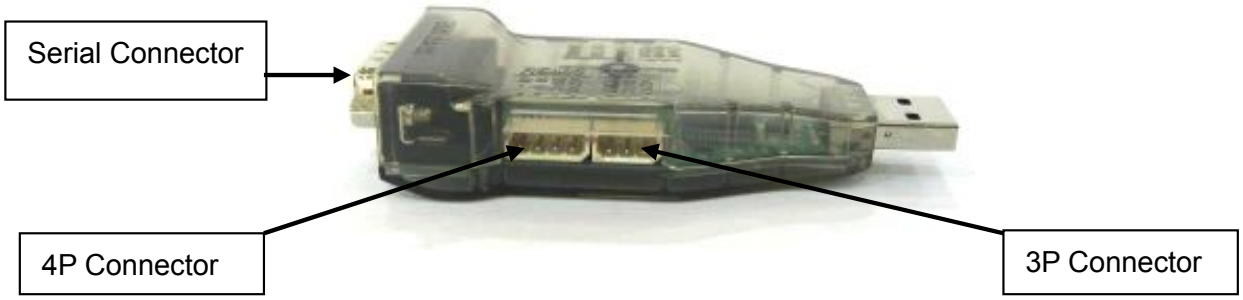
Downloadable from our website (www.robotis.com/usb2dynamixel)

1-3. System Requirements

◎ PC	:	PC compatible with IBM	(required)
		USB 1.1 or higher	(required)
◎ OS	:	Windows 2000 or Windows XP	(required)
◎ CPU	:	Intel Pentium III 1GHz , AMD Athlon XP 1GHz or higher	(recommended)
◎ RAM	:	256MB or higher	(recommended)

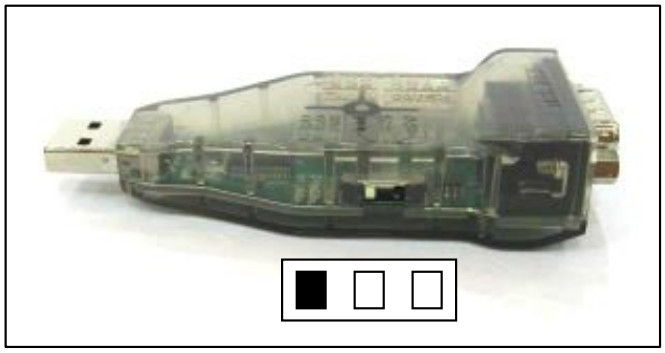
1-4. USB2Dynamixel Connection

1-4-1. USB2Dynamixel Structure



Status Display LED	:	Displays the status of power supply, TxD (writing data), and RxD (reading data)
Function Selection Switch	:	Selects how TTL, RS-485 and RS-232 communicates
3P Connector	:	Connects to AX series dynamixel using TTL network
4P Connector	:	Connects to DX or RX series dynamixel using RS-485 network
Serial Connector	:	Transforms a USB port into a serial port using RS-232 network,

1-4-2. Connection of AX series Dynamixel



Set the function selection switch to TTL mode

Connect the 3P cable to the 3P connector of USB2Dynamixel and the connector of the dynamixel (There are two connectors on the dynamixel, any of which can be connected).

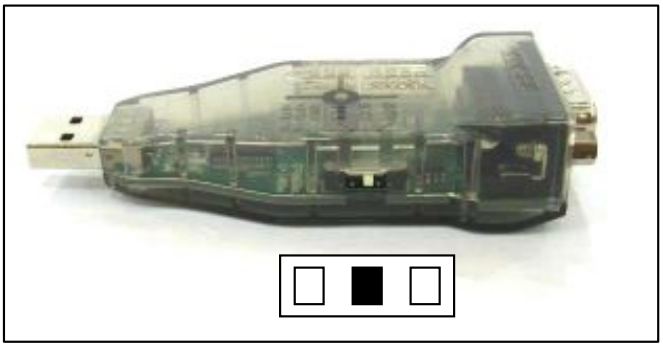
Dynamixels can be connected in series using a 3P cable.
Connect the power line to the last dynamixel.

POWER (DC 7 to 10V)

Caution

The USB2Dynamixel does not supply power to a dynamixel. For further information on how to supply power, please refer to 1-4-6 How to Supply Power.

1-4-3. Connection with DX/RX series Dynamixel



Set the selection function switch to RS 485 mode

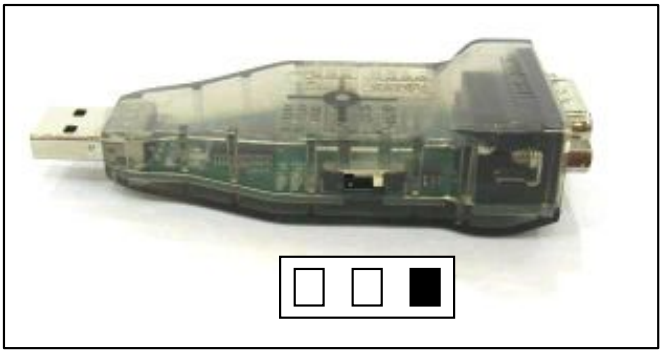
Connect the 4P cable to the 4P connector of USB2Dynamixel and the connector of the dynamixel (There are two connectors on the dynamixel, any of which can be connected).

Dynamixels can be connected in series using a 4P cable.
Connect the power line to the last dynamixel.

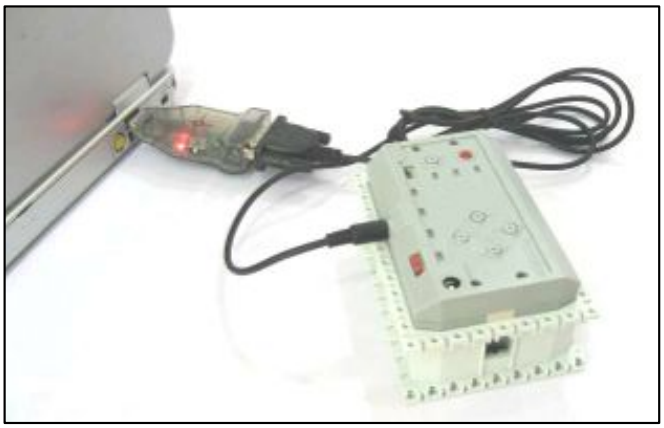
POWER
(RX-10: DC10 to 12V)
(RX-28: DC12 to 16V)
(RX-64: DC15 to 18V)
(DX-117: DC12 to 16V)

1-4-4. Transformation of an USB port into a Serial port

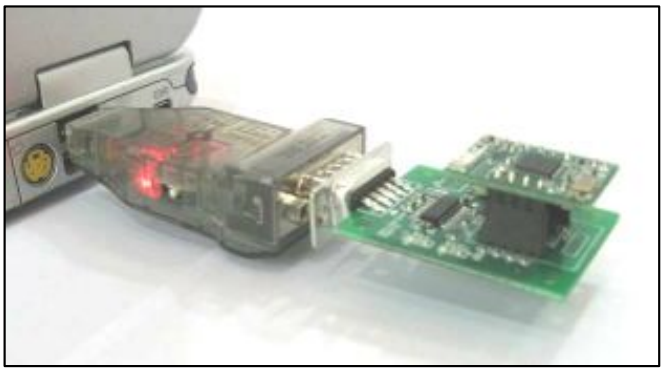
The USB2Dynamixel can be used to transform an USB port into a serial port for a PC without a serial port such as a laptop computer.



Set the function selection switch to RS 232 mode.



The CM-5 communicates using a serial port. When the USB2Dynamixel is used, the network with CM-5 can be made even using an USB port of a PC. For further information on this, please refer to the CM-5 manual



The ZIG2Serial can be used as an USB port using USB2Dynamixel. For this type of operation, the ZIG2Serial can be used without additional power supply. For further information on this, please refer to the ZIG2Serial manual.

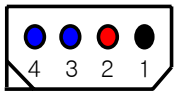
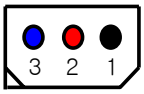
Caution

Do not connect to an equipment that consumes more than 500mA, which may damage the USB board.

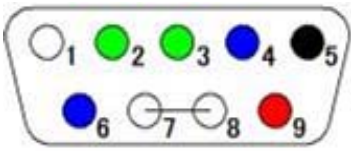
1-4-5. Pin Figure

The following pictures show the usage of the connector pins used in the USB2Dynamixel. To make them suitable for the intended purpose, the user should only use them after being familiarized with the usage of each pin.

Pin Figure of 4P/3P Cable Connector

4 Pin Cable			3 Pin Cable		
Pin No.	Signal	Pin Figure	Pin No.	Signal	Pin Figure
1	GND		1	GND	
2	NOT Connected		2	NOT Connected	
3	DATA + (RS-485)		3	DATA (TTL)	
4	DATA - (RS-485)				

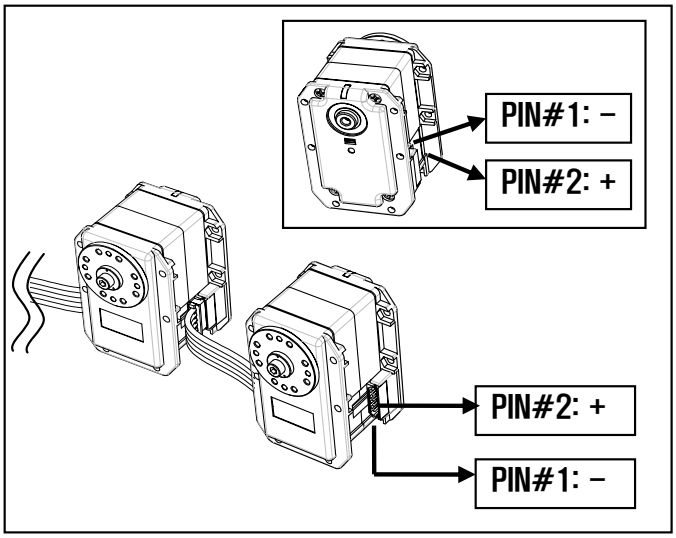
Pin Figure of Serial Connector

Pin No.	Signal	Pin Figure
1	Data (TTL)	
2	RXD (RS-232)	
3	TXD (RS-232)	
4	D+ (RS-485)	
5	GND	
6	D- (RS-485)	
7	Short with No. 8	
8	Short with No. 7	
9	USB power (5V)	

1-4-6. How to Supply Power

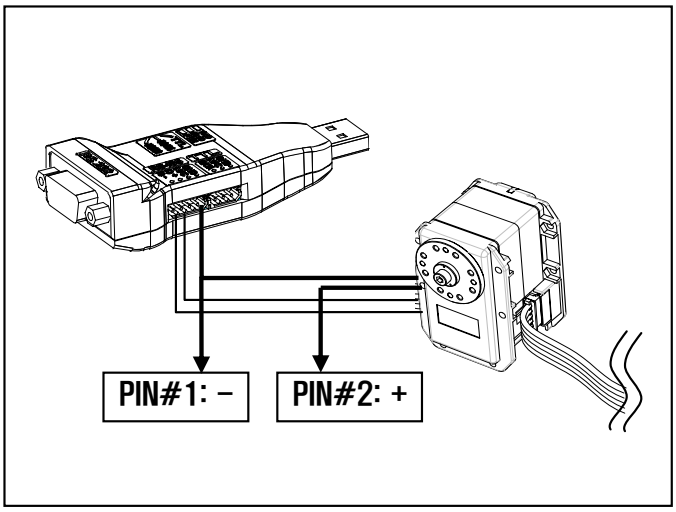
The USB2Dynamixel does not supply power to a dynamixel. Thus, additional power is needed to drive a dynamixel as shown below. For proper voltage to be provided to each dynamixel model, please refer to the dynamixel manual.

◎ Power Connection to the Dynamixel



Based on the "1-4-5 Pin Figure", connect the plus (+) power to the No. 2 pin of the connector, and the minus (-) power to the No. 1 pin (Since two connectors of the dynamixel are identical, the power can be connected to any one of them).

◎ Power Connection between the USB2Dyanmixel and Dynamixel



If the power cannot be connected to the dynamixel as above, connect the power between the USB2Dynamixel and the dynamixel as shown on the left.

Separate the wire connected to the No. 2 pin of the UBS2Dynamixel connector, and connect the plus (+) power to it. Connect additional wire via the Y winding to the wire connected to the No. 1 pin of the USB2Dynamixel connector, and connect the minus (-) power to it.

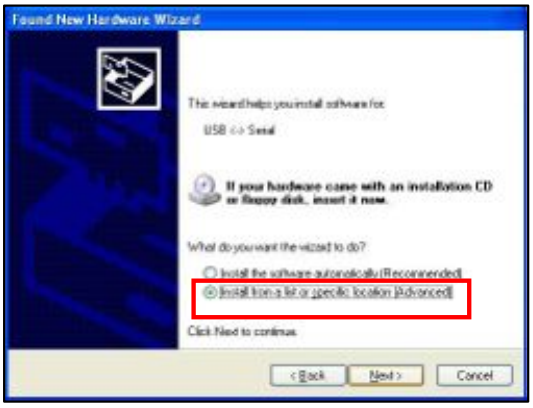
1-5. Installation

1-5-1. Installation of Driver

< Step 1 > Download the Dynamixel Driver file from our website and uncompress it.

(www.robotis.com/usb2dynamixel)

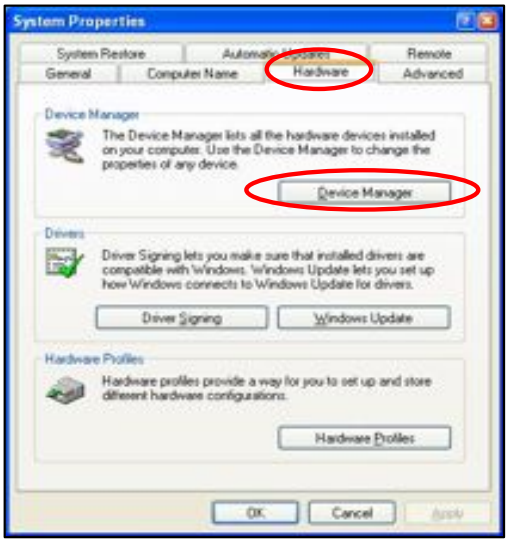
< Step 2 > Connect USB2Dynamixel to a PC, and the New Hardware Search window will apear on the PC. Click on [Install from the list or Specific Location].



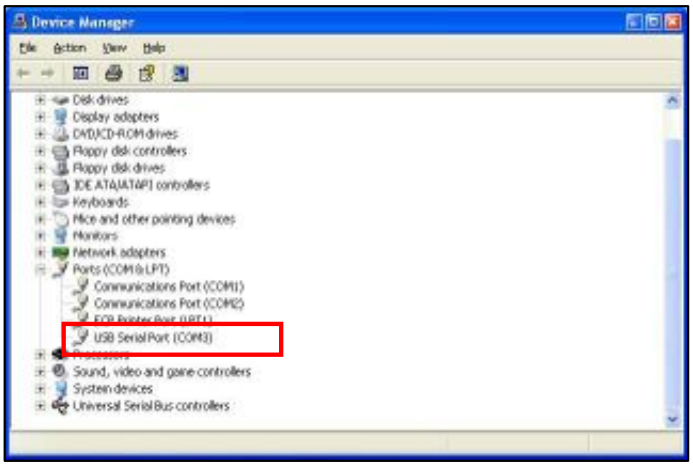
< Step 3 > Move to where USB2Dynamixel driver folder which was uncompressed from step 1.

< Step 4 > If the New Hardware Search window appears again, repeat the Step 2 to 3 one more time.

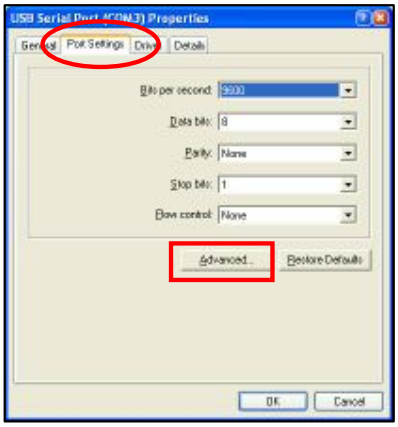
< Step 5 > Select Control Panel>> System>> Hardware>> Management Console.

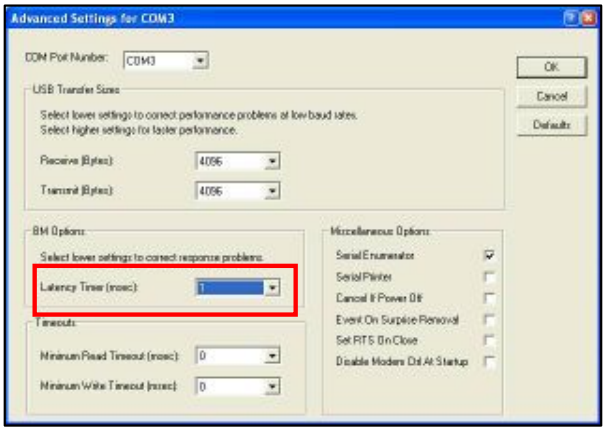


<Step 6 > Check whether "USB Serial Port" is installed at "Port (COM and LPT) of the Management Console. It would be convenient for the user to remember the number of port (COM #) for network connection.



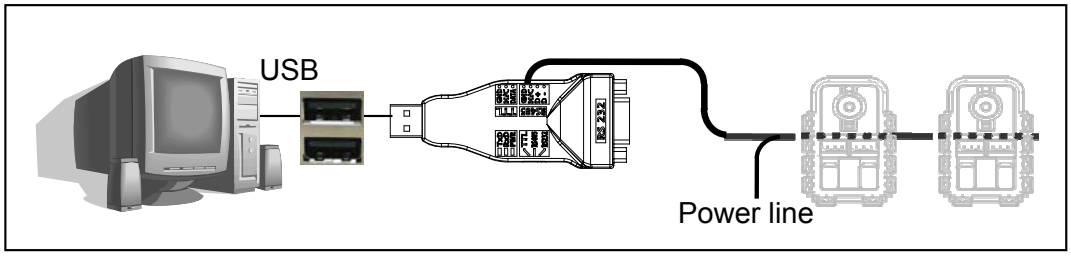
< Step 7 > Double-click on "USB Serial Port(COM#)", and select "Port Setting >> Advanced" to modify the **Latency time to 1ms**. Doing such will increase the network speed.





1-5-2. Installation and Implementation of the Dynamixel Manager

< Step 1 > Make a connection of PC-USB2Dynamixel-Dynamixel as shown below, and supply power to the dynamixel unit(s). For further information on how to supply power, please refer to Section 1-4-6 How to Supply Power.

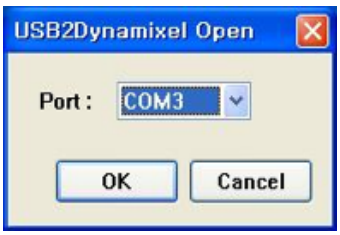


If the USB2Dynamixel driver is not yet installed on the PC and if the USB2Dynamixel is connected to the PC, the following window appears. In this case, install the driver by referring to Section 1-5-1 Installation of Driver.

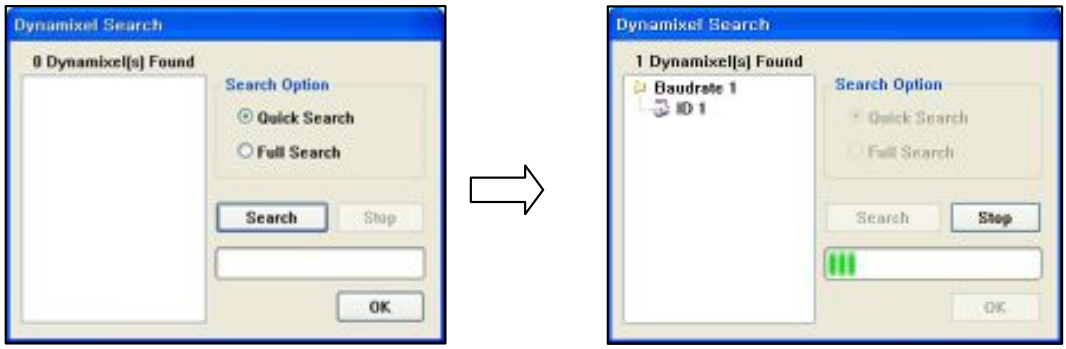
< Step 2 > Download the Dynamixel Manager file, 'DynamixelManager.zip' from our website (www.robotis.com/usb2dynamixel)

< Step 3 > Decompress 'DynamixelManager.zip'. Execute the 'DynamixelManager.exe' file which was decompressed from the zip file.

< Step 4 > A window that sets the port of the PC appears. Select the communication port labeled 'USB Serial Port' which was added at the Management Console during the installation of the driver. The Dynamixel Manager displays all ports, including the USB2Dynamixel , that support USB2Serial. Thus, a port supporting the UBS2Dynamixel should be selected correctly. If the connection of the port is not correct, the dynamixel is searched when the 'Search' function in Step 5 is implemented.



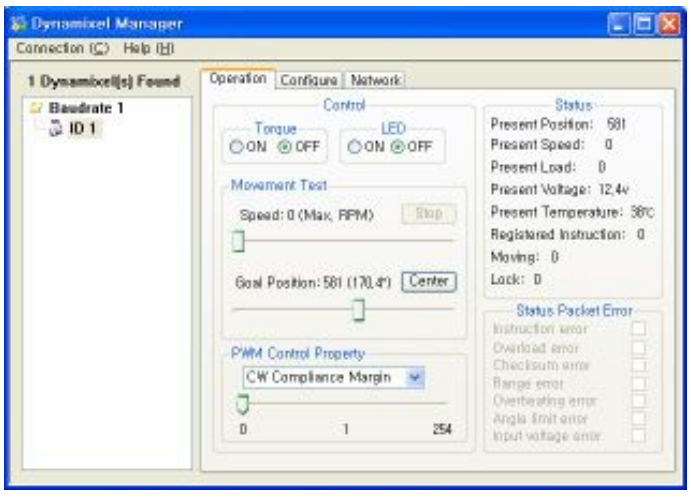
< Step 5 > When the window searching the dynamixel appears, press the 'Search' button.



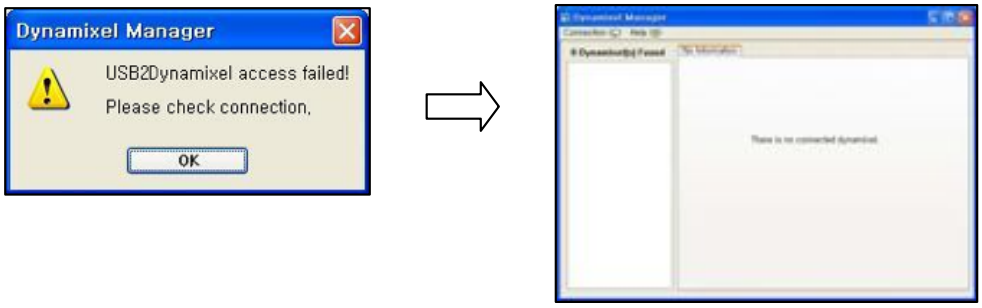
If the 'Search' cannot be implemented, redo 'Search' after cheching the following:

1. Check whether the port is selected correctly.
2. Check whether the function selection switch of the USB2Dynamixel is located correctly.
3. Check the status of the connection between the USB2Dynamixel and the dynamixel and the status of power supply.
4. Quick Search is only possible when the Baud rate of the dynamixel is set to the main Baud rate. When the user has changed the Baud rate of the dynamixel arbitrarily, do not use the Full Search function.

< Step 6 > Dynamixel Manager will be implemented when the 'OK' button is pressed after the dynamixel is searched.



If there is a problem on the connection of the USB2Dynamixel, an error message appears as below. If the OK button is pressed, the Dynamixel Manager is implemented as below. In this case, restart from the Step 4 referring to the Connection menu of the Dynamixel Manager after checking the connection between the USB2Dynamixel and the PC.



If there is a problem on the connection between the USB2Dynamixel and the dynamixel, an error message appears as below. In this case, restart from the Step 5 referring to the Connection menu of Dynamixel Manager after checking the status of the connection between the USB2Dynamixel and the dynamixel and the status of power supply.

2. Dynamixel Manager

The Dynamixel Manager is a utility that helps manipulate a dynamixel from a PC easily. It can also change variables such as the ID or the Baud rate of the dynamixel.

2-1. Connection

The Connection tab is used to set the connection status between the PC and dynamixel.

Open / Close Opens or closes the network port of the USB2Dynamixel.

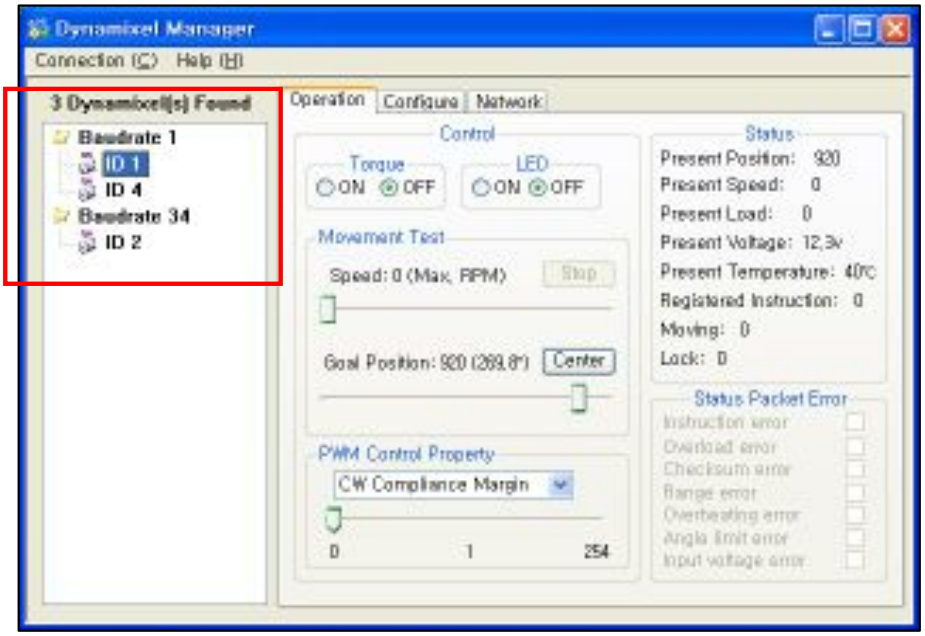
Search Searches a dynamixel connected to the USB2Dynamixel.

Exit Ends the Dynamixel Manager.

2-2. List

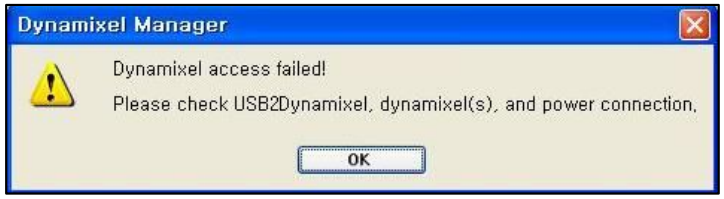
The List tab is used to show which dynamixel is connected to the USB2Dynamixel.

The following picture shows that there are a total of three dynamixels connected. Specifically, two dynamixels with the Baud rate 1 are connected, and their IDs are No. 1 and 4. A dynamixel with the Baud rate 34 is connected, and its ID is No. 2.



Only one dynamixel can be driven. To drive it, click on the corresponding ID.

When a communication error breaks out between the USB2Dynamixel and the dynamixel, click on the dynamixel in question, then the following error message appears. If the OK button is pressed, the dynamixel where the communication error broke out will disappear from the list.



2-3. Operation

The Operation tab is used to drive the dynamixels.

Torque If 'On' is clicked, the output torque is applied to the dynamixel.
If 'Off' is clicked, the output torque of the dynamixel is removed.

LED If 'On' is clicked, the LED of the dynamixel is turned on.
If 'Off' is clicked, the LED of the dynamixel is turned off.

Speed Sets the driving rate of a dynamixel. Tune the rate by moving the scroll bar with the mouse. For the fine tuning, use the left and right direction keys **[** ←, → **]** on the keyboard. The value of Speed can be set 0 to 1023 as data value of rate. The value in the brackets is the value of RPM.

The STOP button is activated when Operation Mode is set to Wheel in Configure, and works as a function stopping the driving of Wheel.

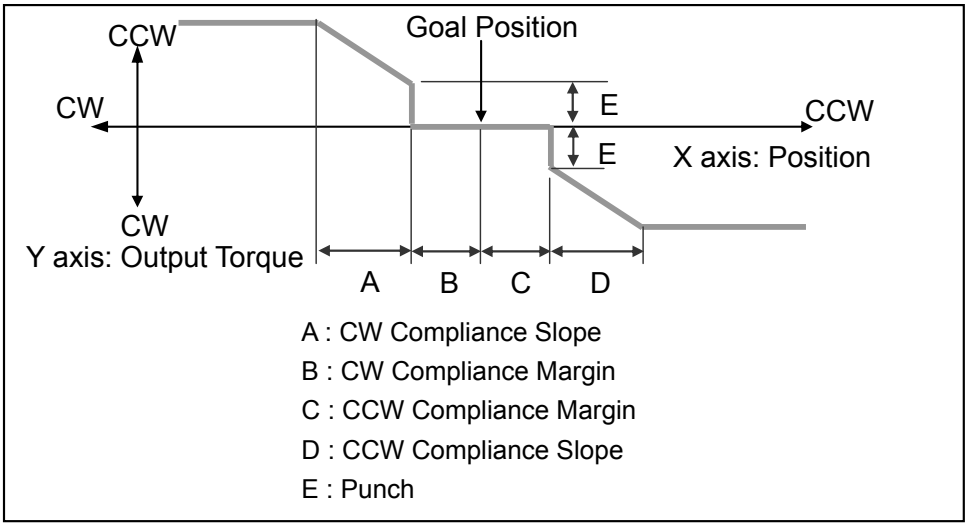
The maximum driving rate of a dynamixel depends on the type of dynamixels and the level of connected voltage. Thus, in most cases, the maximum rate is achieved when the value of Speed is lower than 1023. The driving speed cannot be raised even if the value of Speed is set higher than the value of Speed at which the maximum driving speed reaches. For further information about this, please refer to

the manual of each dynamixels. The Speed value 1 is the lowest speed, and the Speed value 0 the highest.

Goal Position **S**ets the driving angle of a dynamixel. Adjust it by moving the scroll bar with the mouse. For the fine tuning, use the left and right direction keys **[** ←, → **]** on the keyboard. The value of Goal Position can be set 0 to 1023 as data value of position. The value in the brackets is the angle.

The Center button is to move the dynamixel to the center (Position Data: 512).

PWM Control Property A dynamixel can be smoothly driven by setting the value of PWM Control Property.



CW Compliance Margin Sets a margin where torque is removed before arriving at Goal Position while moving it clockwise.

CCW Compliance Margin Sets a margin where torque is removed before arriving at Goal Position while moving it counter-clockwise.

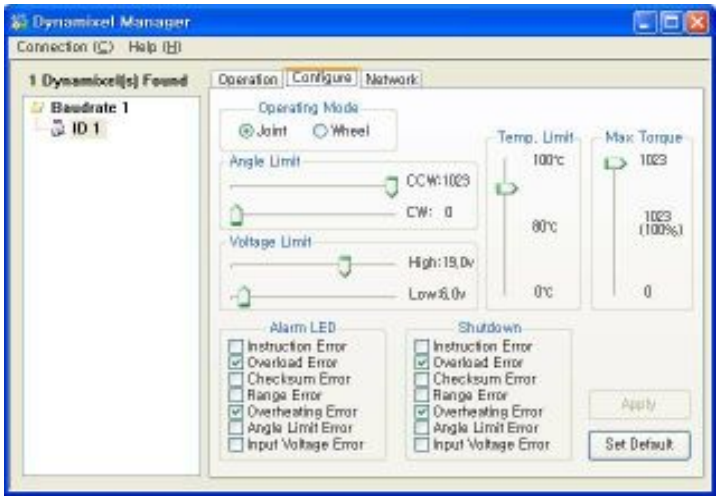
CW Compliance Slope Sets a margin where torque is slowly removed before arriving at Goal Position while moving it clockwise.

CCW Compliance Slope Sets a margin where torque is slowly removed before arriving at Goal Position while moving it counter-clockwise.

Punch	Sets the minimum value of output Torque.
Torque Limit	Sets the maximum value of output Torque.
Status	Displays the present status of a dynamixel.
Status Packet Error	Displays the nature of the error when an error occurs while a dynamixel is driving.

2-4. Configure

The Configure tab is used to set additional restrictive conditions necessary for driving a dynamixel. Save changed information by clicking on the Apply button after changing information.



Operation Mode Selects a driving mode of a dynamixel.
Joint: Used when the position/speed control is required like the joint of a robot.
Wheel: Used when endless turn is required like a wheel.

Angle Limit Sets a driving angle. Angle Limit Error occurs when the system strays out of the set driving angle.

Voltage Limit Sets a level of connected voltage. Input voltage Error occurs when the connected voltage strays out of the set value.

Temp. Limit Sets the maximum value of operating temperature. Overheating Error occurs when the ambient temperature strays out of the operating temperature.

Max Torque Sets the maximum value of output torque.

Alarm LED When a selected error occurs, the LED is turned on

Shutdown When a selected error occurs, this removes output torque.

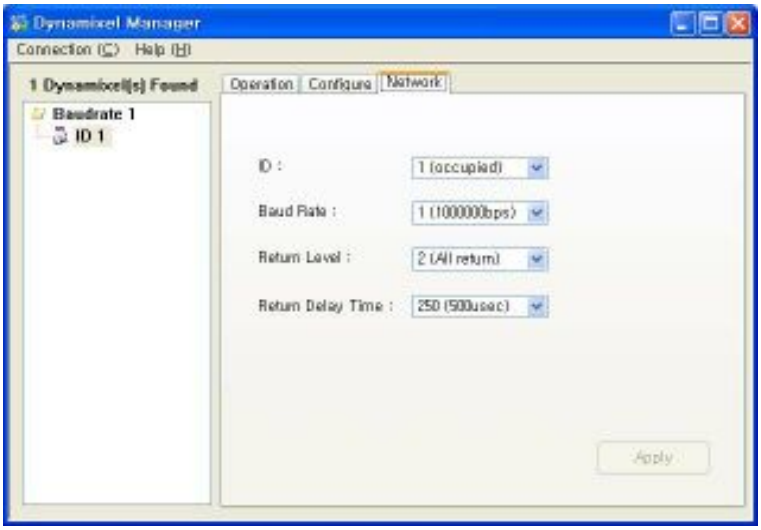
Apply Saves changed information by pressing the Apply button after changing information.

Set Default Restores the default values set in the factory from set information.

Caution If the maximum value of each function is set too low or if the minimum value is set too high, an errors may occur at a dynamixel.

2-5. Network

The Network tab is used to set the functions necessary for the network of the dynamixel and the PC. Save information by pressing the Apply button after changing information.



ID Sets the ID of a dynamixel. The ID connected to the USB2Dynaminxel is marked as [Occupied]. The ID cannot be changed using an Occupied ID.

Baud Rate Sets the Baud rate between the dynamixel and the PC. It can only be changed to the main Baud rate.

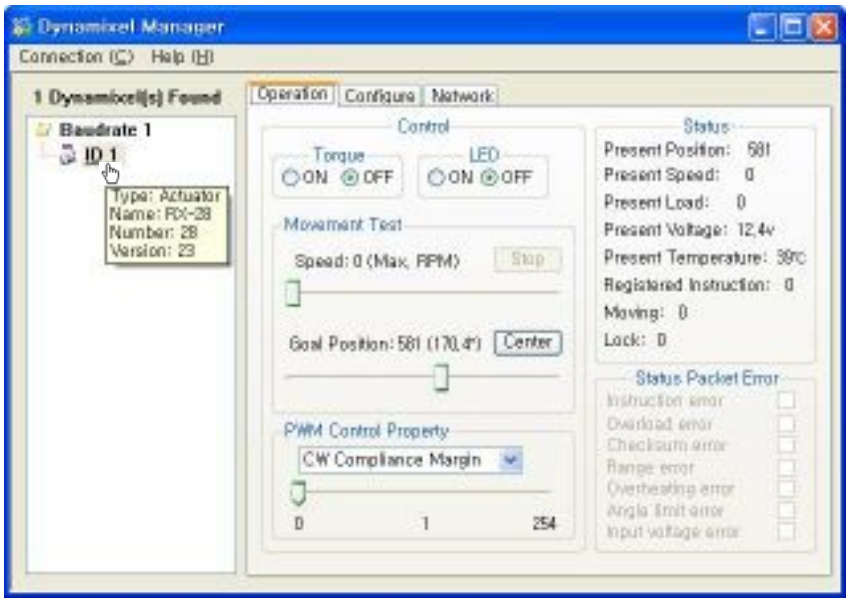
Return Level Sets how a dynamixel returns the achieved output of an order to the PC. If the Return Level is set 0, the [Operation] and [Configure] functions cannot be used.

Return Delay Time There is a time delay between receiving the order of the dynamixel and returning the achieved output of the order to the PC. This menu sets the time delay occurring at this time.

Apply When the Apply button is pressed after information is changed, the changed information is saved.

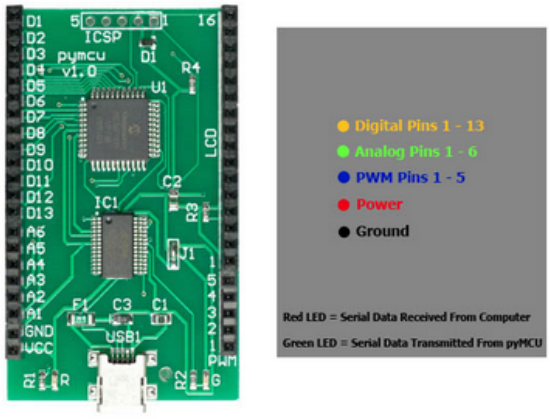
2-6. Information

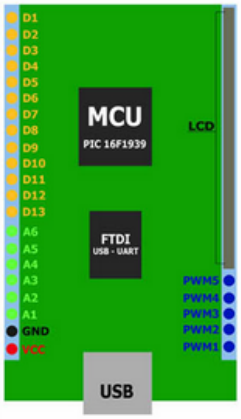
The Information tab displays information about the model and firmware version of the dynamixel connected at the moment.



N Dokumentasjon pyMCU pyMCU v1.0

Overview





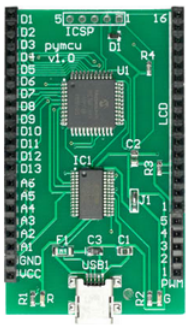
pyMCU was created with the idea of being a simple cost effective platform to interface between the computer and the physical world using python. The heart of pyMCU is a microchip PIC 16F1939 offering 13 Digital IO Pins, 6 Analog IO Pins, 5 10-bit PWM Pins, and a 16 Pin Parallel LCD Interface. The USB Interface uses the industry standard FTDI USB to UART FT232R chip. Drivers and python module are available for Windows, OSX, and Linux.

I wanted to make pyMCU simple to use for the beginner but also have the option to be utilized in a more powerful way for the advanced user. The beginner method for using pyMCU and its initial intent for use is to utilize the pymcu python module, by creating an mcuModule Class Object in python you will have access to various functions common to typical microcontroller programming, setting pins high and low, reading analog pin values, using hardware pwm pins, i2c, 1-wire and SPI protocols and so on. For the more advanced user you can override the pyMCU firmware and program the PIC chip directly and take advantage of all the raw power and functionality the PIC chip has to offer and that way the pyMCU board acts as a nice breakout board platform to build on.

When I began creating pyMCU I looked at several already available boards and thought maybe I would just write the software to interface with them but after evaluating several factors including price and my previous knowledge of PIC microcontrollers I decided I could build a simpler less expensive and more powerful option. I've been working with Microchip PIC microcontrollers for almost 5 years now long before Arduino was as popular as it is now so given that experience as well as time and resources invested in microchip products I figured it was going to be quicker and simpler to create the first pyMCU using PIC. Depending on the interest I get from putting this product out I will consider developing an Arduino version of pyMCU that you would be able to download, modify and compile in the Arduino IDE for any of the available Arduino boards out there.

pyMCU v1.0

Hardware Specs



MCU	Microchip PIC 16F1939
MCU SPEED	32 Mhz
MCU Program Memory	28K Bytes
MCU Ram	1024 Bytes
Analog I/O	6, 10-bit Analog Pins
Digital I/O	13, Digital Pins
PWM	5, 10-bit PWM Pins
LCD	16 Pin Parallel LCD Interface, Supports up to 4 Line LCD
EEPROM	256 Byte EEPROM Storage
i2c	Software Configurable i2c Communication Functions
SPI	Software Configurable SPI Communication Functions
Serial	Software Configurable Serial Functions for MCU to MCU Direct Communications
1-Wire	Software Configurable 1-Wire Communication Functions
Pulse In/Out	Software Configurable Pulse In / Out Functions
Sound	Built-In Functions for Sound Generation: Arbitrary Frequency Generator, Misc. Sound / Tone Generator, DTMF Generator
ICSP	ICSP Interface for firmware updates or override pyMCU firmware with custom code
USB	FTDI FT232R USB to UART Interface
POWER	Power Supplied by USB 25mA Source / Sink Current I/O 500mA Max Draw from USB Solder Jumper J1 Configurable for 3.3V or 5.0V Operation. Battery operatable 1.8V - 5.5V for use when overriding pyMCU firmware with custom code
Measurements	L = 2.45" (62.2mm) W = 1.45" (37.0mm)