3.1 Servomotorer – Dynamixel AX-12+

Figur 3.1: Servo [5].

Figur 3.1: Servo [5].

Servomodulene som er brukt i dette prosjektet er av typen Dynamixel AX-12+ fra Robotis, se figur 3.1. Dette er moduler som inneholder både servo, måleelektronikk og kommunikasjonselektronikk. Innebygde sensorer kan bl.a. registrere vinkel, vinkelhastighet, dreiemoment og temperatur. Datautveksling skjer over en én-leders seriell buss, UART. Over denne bussen sendes datapakker fra mikrodatamaskinen, for å hente ut eller legge inn data i modulens RAM og EEPROM, også kalt kontrolltabellen. Pakkene er nærmere beskrevet i neste underkapittel og i modulenes brukermanual [5].

AX-12+ er bare én av enhetene i Dynamixel-serien. Det finnes mange andre enheter, f.eks. sensormoduler og kraftigere servoer. Felles er kommunikasjonsdelen; alle har samme fysiske plugger og kabler, og datapakkenes oppbygning er lik. Maskinvaredelen av dette prosjektet vil følgelig være kompatibel med hele Dynamixel-serien.

3.1.1 Datapakker

Datapakkene som brukes til kommunikasjon med servomodulene kalles «instruksjonspakker » (data fra Arduino) og «svarpakker» (data fra servo). Både instruksjonspakker og svarpakker er bygd opp på samme måte. Den følgende listen går igjennom oppbygningen, byte for byte. Se også figur 3.2.

- Først kommer 2 startbyte, som alltid er 0xFF 0xFF.
- Servomotorens ID 2 [0x00, 0xFE]. Den høyeste lovlige ID-en 0xFE, er en såkalt «broadcasting ID», som gjør at alle tilkoplede moduler vil motta og lese pakken.
- Pakkens lengde, som er antall byte etter denne, medregnet sjekksum. Det er viktig å skille mellom denne lengden og den totale lengden som også inkluderer de fire første bytene 0xFF, 0xFF, ID og lengdebyten selv.
- Instruksjonstypen ved sending eller «error»-byte ved respons. Alltid én byte.
- Eventuelle parametre tilhørende instruksjonen. Antall parametre er alltid gitt av [verdien av lengdebyten] - 2. Totallet kommer av instruksjonsbyten og sjekksumsbyten, som alltid er med.
- Sjekksumsbyte, definert som
    - Sjekksum =_(ID + lengde + instruksjon + parameter 1 + _ _ _ + parameter n)

- der operatoren ~ betyr invertering av bit. Summeringen gjøres i en byte-variabel uten overflyt, slik at f.eks. 0xFF + 0x01 = 0x00.

- Pakkeoppbygningen er illustrert i figur 3.2 med et tilhørende eksempel. Eksemplet setter drivspenningsområdet til modulen med ID 0 til området 10–17V (0x64_0xAA) ved hjelp av funksjonen WRITEDATA = 0x03 (se neste underkapittel).

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

Figur 3.2: Oppbygning (øverst) og eksempel (nederst) på instruksjonspakke

Videre i denne rapporten blir uttrykket «gyldig servopakke» brukt. En «gyldig» servopakke defineres som følger: Pakken starter med bytene 0xFF 0xFF og det totale antall byte er [element med indeks 3]+4. Elementet med indeks 3 er lengdebyten i pakken, som vist i figur 3.2. Firetallet er for å telle de fire første pakkene (som ikke er inkludert i lengdebyten). Sjekksummen blir ikke kontrollert.

3.1.2 Instruksjoner

Servomodulene støtter flere typer instruksjoner som er vist i tabell 3.1. Verdiene vist i tabellen er verdien til instruksjonsbyten i pakken, ref. figur 3.2.

Tabell 3.1: Instruksjoner for servomodulene [5]

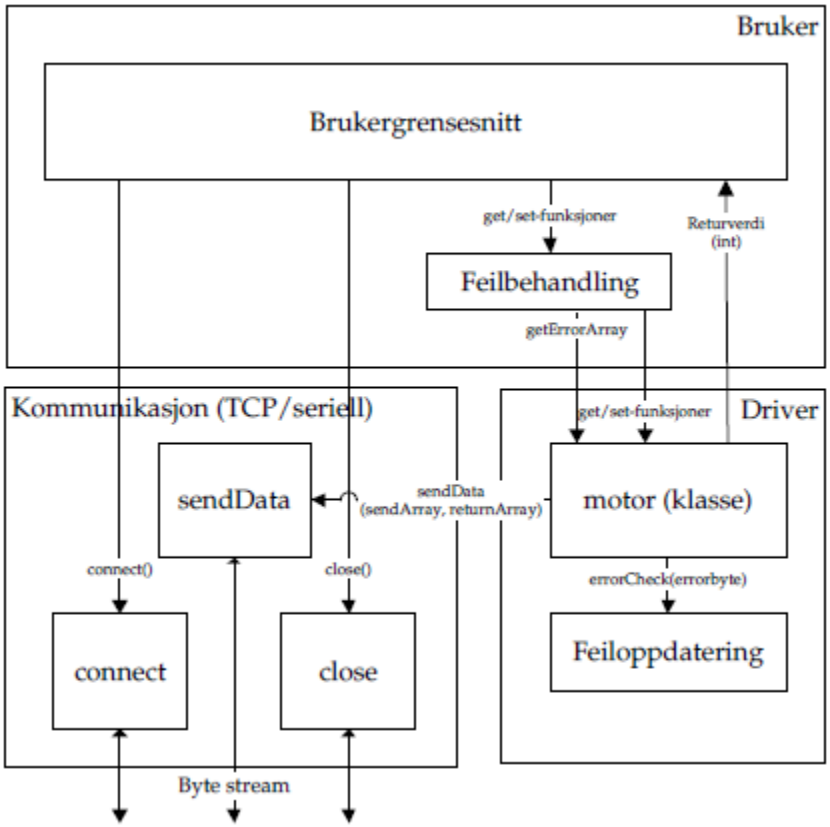
Instruksjon Verdi Forklaring
PING 0x01 Brukes til å få en respons fra servomodulene.
READDATA 0x02 Leser verdier fra servomodulens kontrolltabell.
WRITEDATA 0x03 Endrer verdier i servomodulens kontrolltabell.
REGWRITE 0x04 Samme som WRITEDATA, men instruksjonen blir ikke utført før en ACTION-instruksjon blir sendt.
ACTION 0x05 Trigger REGWRITE-instruksjoner.
RESET 0x06 Endrer verdiene i kontrolltabellen tilbake til fabrikkinnstillingene.
SYNCWRITE 0x83 Brukes for å sende instruksjoner til flere servomoduler samtidig.

3.5 Boost

Boost er en samling med C++-biblioteker som inneholder f.eks. lineær algebra, bildebehandling, multithreading o.l. Boost inneholder også en samling kommunikasjonsbiblioteker, kalt Asio, som har både har nettverks (TCP/IP)- og seriellfunksjonalitet. Boost kan kompileres på mange forskjellige plattformer (Windows, Unix, osv.) og egner seg derfor til dette prosjektet, da det er ønsket en plattform-uavhengig kode. [4]

4.3 Programvare – PC-side

Programvaren på klientsiden, programvaren som styrer servomodulene, er delt opp i tredeler; en motordriver, en kommunikasjonsdel og en bruker av disse. Figur 4.10 gir en oversikt over hvordan gruppa har organisert systemet. Brukeren, som i dette tilfellet er et GUI-program, er delen som kaller motordriveren og kommunikasjonsdelen. I tillegg til at brukeren kaller kommandoer fra kommunikasjonsdelen, typisk ved til- og frakobling, kaller også motordriveren kommunikasjonsdelen når det skal sendes instruksjonspakker til servomodulene.



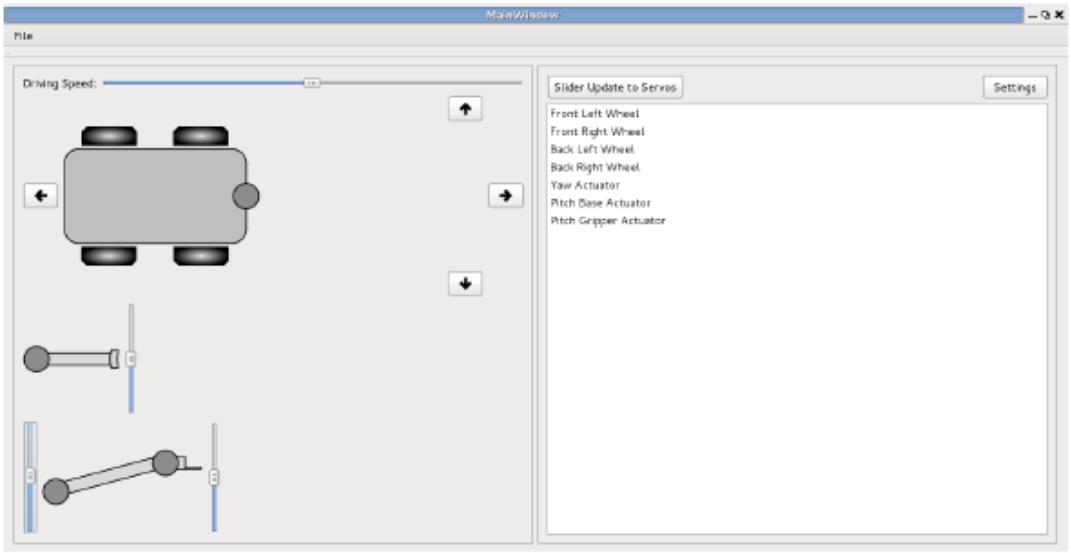
Figur 4.10: Strukturdiagram for PC-programmet/biblioteket.

4.3.1 Brukergrensesnitt

For å illustrere hvordan man benytter motordriveren og kommunikasjonsdelen, har gruppa laget et eksempel på dette.

Brukeren gruppa har laget er et GUI-program hvor man kan styre en robotbil med gripearm. Programmet kan styre robotens arm, kjøre fremover, bakover og svinge. I tillegg har programmet en dialogboks for hver servomodul som leser forskjellige parametere fra modulen kontrolltabell (temperatur, modellnummer, osv.), og hvor man kan endre forskjellige parametere (ID, grenseverdier for bevegelse, osv.) Et skjermbilde av programmet er vist i figur 4.11.

Det er i prinsippet ikke noen begrensning på hva en bruker kan være, hvis man for eksempel benytter systemet i en industriprosess kan brukeren være programmet som styrer servomodulene til å utføre oppgaver i en gitt rekkefølge.



Figur 4.11: Eksempel på brukergrensesnitt for motordriver/kommunikasjon

For brukeren er det lagt vekt på at kommandoene skulle være så enkle som mulig og alt av lavnivå konstruksjon av instruksjonspakker skulle håndteres av motordriveren.

Til- og frakobling håndterer brukeren direkte med kommunikasjonsdelen, som beskrevet i kapittel 4.3.3. Det er også mulig å sende instruksjonspakker direkte til kommunikasjonsdelen, selv om dette er noe mer kronglete. Servomodulene kan motta en såkalt SYNCWRITE-pakke som kan sende forskjellige verdier til flere servomoduler samtidig. Dette ikke er implementert i motordriveren, pga. dens oppbygging hvor et objekt er én motor. Derfor må slike SYNCWRITE-pakker konstrueres manuelt av brukeren og sendes direkte til kommunikasjonsdelen. Et eksempel på dette er:

```
// konstruksjon av en <<SYNCWRITE>>-pakke, jamfør kapittel 4.6 i
   servomodulenes dokumentasjon
unsigned char syncwritePacket[] = {
    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
    };

// sender denne pakken med kommunikasjonen
communication.sendDataWithoutResponse(&syncwritePacket);

// MERK: <<SYNCWRITE>>-pakker får IKKE svar fra servomodulene
```

4.3.2 Motordriver

Motordriveren er en klasse som representerer én spesifikk servomotor. Når man initialiserer et motorobjekt sender man med identifikasjonen til motoren, samt adressen til kommunikasjonsbiblioteket,
som følger:

```
// initialisering av et motor-objekt ved navn motor1, id = 1 og
   kommunikasjons-objekt = communication
Motor motor1(1, &communication);
```

Motorklassen har funksjoner som kan lese og skrive til motorens kontrolltabell, for eksempel:

```
// spør om temperaturen til motor1
int temp = motor1.getTemp();

// skru på led-lyset til motor1
motor1.setLED(1);
```

En fullstendig liste over funksjoner finnes i servomodulenes dokumentasjon ([5]).

Motordriverens funksjoner konstruerer en pakke med flere byte (unsigned char) som sendes til motoren ved hjelp av kommunikasjonsbiblioteket, se kapittel 4.3.3.

Servomotorene tillater også en såkalt REGWRITE-pakke. Da sendes instruksjonen til servomotoren, men instruksjonen blir ikke utført før man sender en ACTION-instruksjon. Denne ACTION-instruksjonen sendes på broadcasting-ID (0xFE), altså vil alle REGWRITE-instruksjoner på alle servomotorene bli utført samtidig. Motordriveren støtter dette ved å ha et valgfritt argument til alle «set»-funksjoner:

```
// deklarasjon av funksjonen for å skru på LED-lyset på servomotoren
int setLED(int led, bool regWrite = false);

// sender en vanlig instruksjon om å skru av LED-lyset
motor1.setLED(0);

// sender en <<REGWRITE>>-instruksjon om å skru på LED-lyset
motor1.setLED(1, true)

// utfører alle <<REGWRITE>>-instruksjoner på alle servomotorer
motor1.action();
```

Som beskrevet i kapittel 3.1.2, støtter servomodulene instruksjonen SYNCWRITE. Problemet med SYNCWRITE når det gjelder motordriveren er at den er bygd opp rundt en motorklasse som representere én servomotor. Derfor kan ikke motorklassen håndtere SYNCWRITE. Det er beskrevet i kapittel 4.3.1 hvordan man likevel kan sende slike pakker.