Sensur av hovedoppgaver
Høgskolen i Buskerud
Avdeling for Teknologi

Prosjektnummer: 2011-14
For studieåret: 2010/2011
Emnekode: SFHO-3200

Prosjektnavn
HIBU Bugsters

Utført i samarbeid med: Høgskolen i Buskerud avd Kongsberg

Ekstern veileder: Dag Samuelsen

Sammendrag:
Styring av robot over webinterface

Stikkord:
- Robot
- Sun SPOT
- Webinterface

Tilgjengelig: ja

Prosjekt deltagere og karakter:

Navn	Karakter
Fredrik Gulbrandsen	
Kim Pollvik	
Sebastian Teie	
Helene Ruud	
Nedim Golo	

Dato: 10. Juni 2011

Torbjørn Strøm Intern Veileder	Olaf Hallan Graven Intern Sensor	Dag Samuelsen Ekstern Sensor

Hovedprosjektrapport
Versjon 1.0

28.05.2011

Gruppe 14

Fredrik Gulbrandsen, Sebastian Teie, Nedim Golo
Kim Pollvik, Helene Ruud

Forord

Denne rapporten omhandler en oppgave gitt som hovedprosjekt ved Høgskolen i Buskerud, skoleåret 2010-11. Oppgaven gikk ut på å videreutvikle en robot slik at den kan programmeres via ett webinterface. Systemet skal brukes først og fremst av elever i grunnskolen og videregående skole. Meningen er å øke interessen for teknologi blant barn og unge.

Det har vært spennende og lærerikt å jobbe med dette prosjektet. Vi har fått erfare hvordan det er å jobbe i team der man er avhengig av at alle gjør sin del av oppgaven for å få framgang. Etter endt år føler vi oss mer forberedt for arbeidslivet en det vi gjorde før vi startet.

Vi ønsker å takke:
Dag Samuelsen for samarbeidet og en interessant oppgave
Torbjørn Strøm for god veiledning.
Rolf Longva for stor hjelpsomhet og flott innsats med lodding av mikrokontroller.
Åge Skaug for stor tålmodighet og behjelpelighet med å lete opp komponenter.
Arne Bjørnar Næss for hans bistand og behjelpelighet, selv utenfor arbeidstid. + hans fantastiske gode humør.
Barbro Gulbrandsen for og alltid stille opp uansett hva det måtte gjelde.

1 SAMMENDRAG

Vi har fått hovedprosjektoppgaven våres fra HIBU. Oppgaven har gått ut på å lage et webinterface som muliggjør programmering av en robot slik at den kan bevege seg fra et punkt til et annet gjennom et ukjent terreng. Systemet skal brukes i undervisning og brukergruppen kommer til være alt fra barneskole elever til studenter. Derfor er det et ønske fra oppdragsgiver at flere programmeringsnivå blir utviklet.

Systemet skal plasseres i en Newton hall i Drammen. Funksjonen til hallen er å skape interesse for teknologi blant elever. Systemet skal være i drift 24 timer i døgnet uten menneskelig innblanding. Det er derfor viktig at systemet er helt selvopererende. Det betyr at robotene må kunne lade seg selv og alle delsystemene må være robuste slik at de ikke feiler ofte.

Prosjektet bygger videre på roboten HiBu Bugster. Dette er en robot som er bygget av et tidligere hovedprosjekt. Vi skal videreutvikle denne og bygge fem nye roboter.

Webinterfacet er laget slik at man kan sette opp en liste med kommandoer som roboten skal utføre. Når brukeren er ferdig med å programmere sender han koden til roboten. Vi bruker en java applett til å overføre koden til serveren. Serveren er plassert i Newton hallen. Den overfører koden til roboten ved hjelp av Sun SPOT enheter. En Sun SPOT er en java prosessor med radio. Koden blir overført trådløst fra Sun SPOT baseenheten som er koblet til serveren til en Sun SPOT enhet i roboten. Denne utfører så koden. Brukeren kan se roboten utføre koden via et webkamera.

Sun SPOT enheten i roboten er koblet til en mikrokontroller som styrer motorer og henter informasjon fra sensorene. Sun SPOT enheten gir beskjed til mikrokontrolleren hva slags oppgaver den skal utføre, mens mikrokontrolleren sender informasjon fra sensorer tilbake til Sun SPOT enheten.

For at roboten skal klare å lade seg må den klare å finne ladestasjonen. Vi har laget et posisjoneringssystem for at roboten skal klare dette. Posisjoneringssystemet bruker ultralyd for å finne avstanden til kjente punkter i rommet og ut ifra dette regner den ut sin egen posisjon.

Ladestasjonen består av to kobberplater som er tre meter lange. Dette er gjort for å gjøre oppgaven med å finne ladestasjonen enklest mulig. Kontaktene på roboten er bøyelige slik at de sikrer kontakt uansett hvilken vinkel roboten treffer ladestasjonen med.

2 INNHOLDSFORTEGNELSE

3 INNLEDNING

Denne rapporten vil detaljert dokumentere vårt prosjekt og redegjøre for de løsningene vi har valgt, og resultatene av de valgene vi har tatt. Vi vil først gi en overordnet beskrivelse av prosjektoppgaven, etterfulgt av alle de forskjellige arbeidsområdene og medfølgende løsninger i en rekkefølge basert ut ifra en brukers tilnærming av systemet. Til slutt vil vi ta for oss gruppens organisering og prosess, samt en oppsummering og avslutning av rapporten.

Oppgaven vi har valgt går ut på å utvikle og konstruere 5 roboter med egenskaper som er bestemt av oppdragsgiver Høgskolen i Buskerud. Dette prosjektet bygger videre på et tidligere prosjekt ved skolen, og jobben vår blir å legge til nye funksjoner på roboten og eventuellt rette opp tidligere løsninger som ikke har fungert som ønsket.

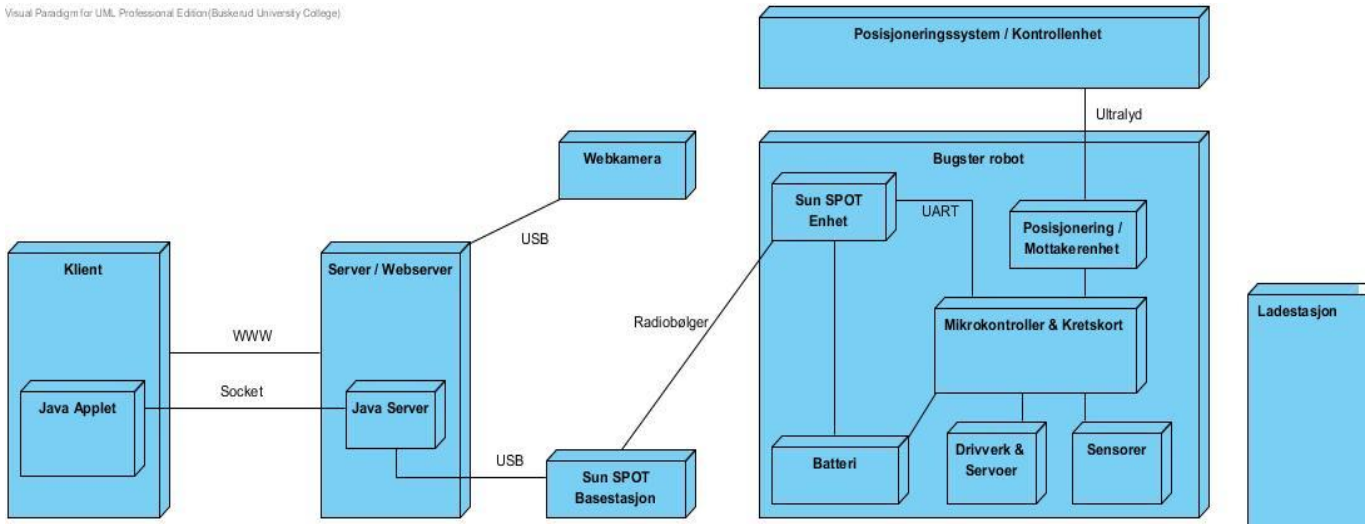
Robotene skal kunne programmeres via et interface på internett til å utføre en rekke kommandoer som er bestemt av brukeren. De skal være tilgjengelige for brukerne 24 timer i døgnet og må derfor være i stand til å finne en ladestasjon og lade seg selv når de går tom for strøm. Robotene skal kunne ta seg gjennom et ukjent terreng med hindringer, så roboten utstyres med sensorer til å oppfatte disse. Brukeren vil ha valgmuligheter for hva roboten skal foreta seg når den møter på en hindring

4 SYSTEMET

Brukeren av systemet kan via prosjektgruppens nettside programmere sin robot. Programmeringsmiljøet vil lastes inn i form av en Java Applet. Her vil brukeren kunne gi inn kommandoer, som blir plukket ut fra et allerede definert kommandoregister og danne en liste over forskjellige gjøremål roboten skal utføre.

Programmeringsmiljøet vil ved benyttelse av Java Sockets kommunisere med en lokal server plassert ved robotene i Newton hallen. Denne lokale serveren vil ha tilkoblet en Sun SPOT basestasjon som trådløst kommuniserer med hver enkelt Sun SPOT enhet lokalisert i robotene.

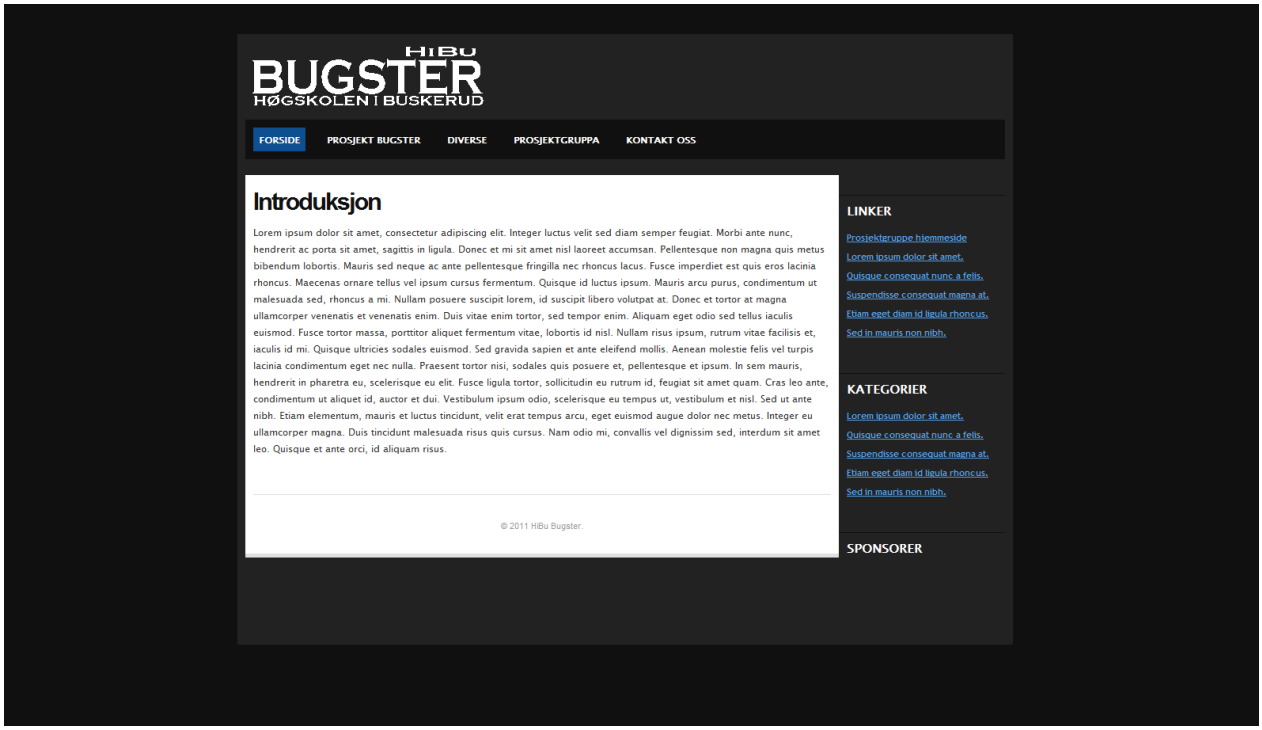
På denne lokale serveren vil det befinne seg en Java applikasjon som lytter og mottar overføringer fra programmeringsmiljøet lokalisert på nettleseren. Kommunikasjonen mellom Java Appleten og lokal server er en string som består av sett med siffere som representerer de forskjellige kommandoene som roboten kan utføre. Stringen lagres lokalt på serveren og sendes til en klar robot. I det en robot har mottatt sin string vil roboten kunne starte å kjøre.



Sun SPOT enheten er koblet til en mikrokontroller som styrer motorene og henter informasjon fra sensorene. Sun SPOT enheten styrer roboten ved å sende kommandoer til mikrokontrolleren. Kommandoene blir tolket og mikrokontrolleren sender signaler til de komponentene som er nødvendige for å utføre kommandoene. Mikrokontrolleren får inn informasjon fra sensorene og sender de videre til Sun SPOT enheten.

5 NETTSIDEN

Webinterfacet vi har laget til dette prosjektet er laget ved benyttelse av Adobe Dreamweaver. Nettsiden består av hovedsakelige fem hovedområder som skal beskrive vårt prosjekt på best mulig måte, samt også inneholde vårt programmeringsinterface til Hibu Bugster roboten. Nettsiden skal også ha en live videooverføring som viser brukeren hva roboten gjør. Nettsiden er hostet på vår stasjonære server ved å benytte en WAMP webserver. En WAMPserver er en ferdig pakket løsning som består av Apache, MySQL og PHP for Windows. Dette vil si at vi selv har valgt å ha en egen server som vert for vår nettside. Nettsidens design er utformet i et eget CSS dokument i rotmappen.



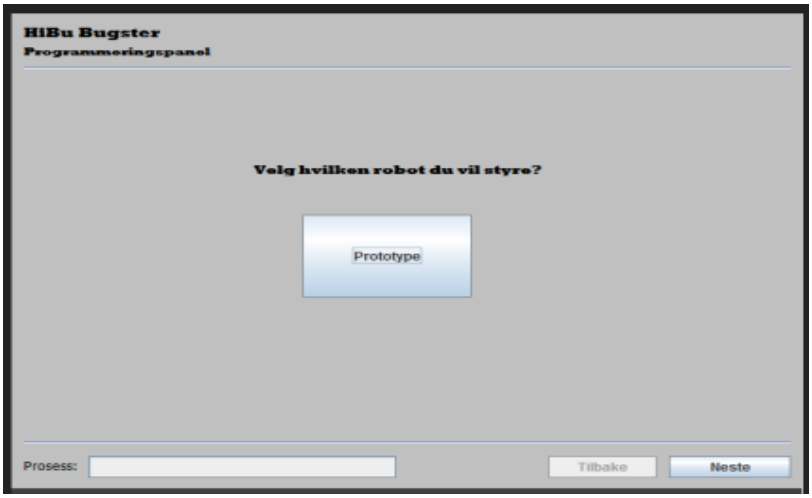
Som vist over i illustrasjonen over kan vi se av vi har valgt følgende få menypunkter:

Forside:	Nettsidens forside, og index dokument. Forøvrig dekkes det her en introduksjon av prosjektet.
Prosjekt Bugster:	Siden som holder Bugsters kontrollpanel, og liveoverføring fra webkamera. For å få tilgang til Bugsters kontrollpanel, vil man være nødt til logge inn med et brukernavn og passord.
Video:	Viser live videooverføring av det roboten gjør.
Prosjektgruppa:	Litt mer om prosjektgruppa, og medlemmene som står bak HiBu Bugster 2011
Kontakt oss:	Noe kontaktinformasjon.

6 JAVA APPLET & SERVER

Når en bruker benytter seg av Hibu Bugster, må dette gjøres via en nettleser. Nettleseren som befinner seg på brukerens datamaskin kommuniserer med roboten som befinner seg i andre enden av systemet. Kommunikasjonsstrømmen foregår da derfor over internett. Løsningen vi har valgt som starten på denne kommunikasjonsstrømmen er en Java Applikasjon. Denne Java Applikasjonen fungerer som en klient som knytter seg til en server som er tett plassert opp mot roboten.

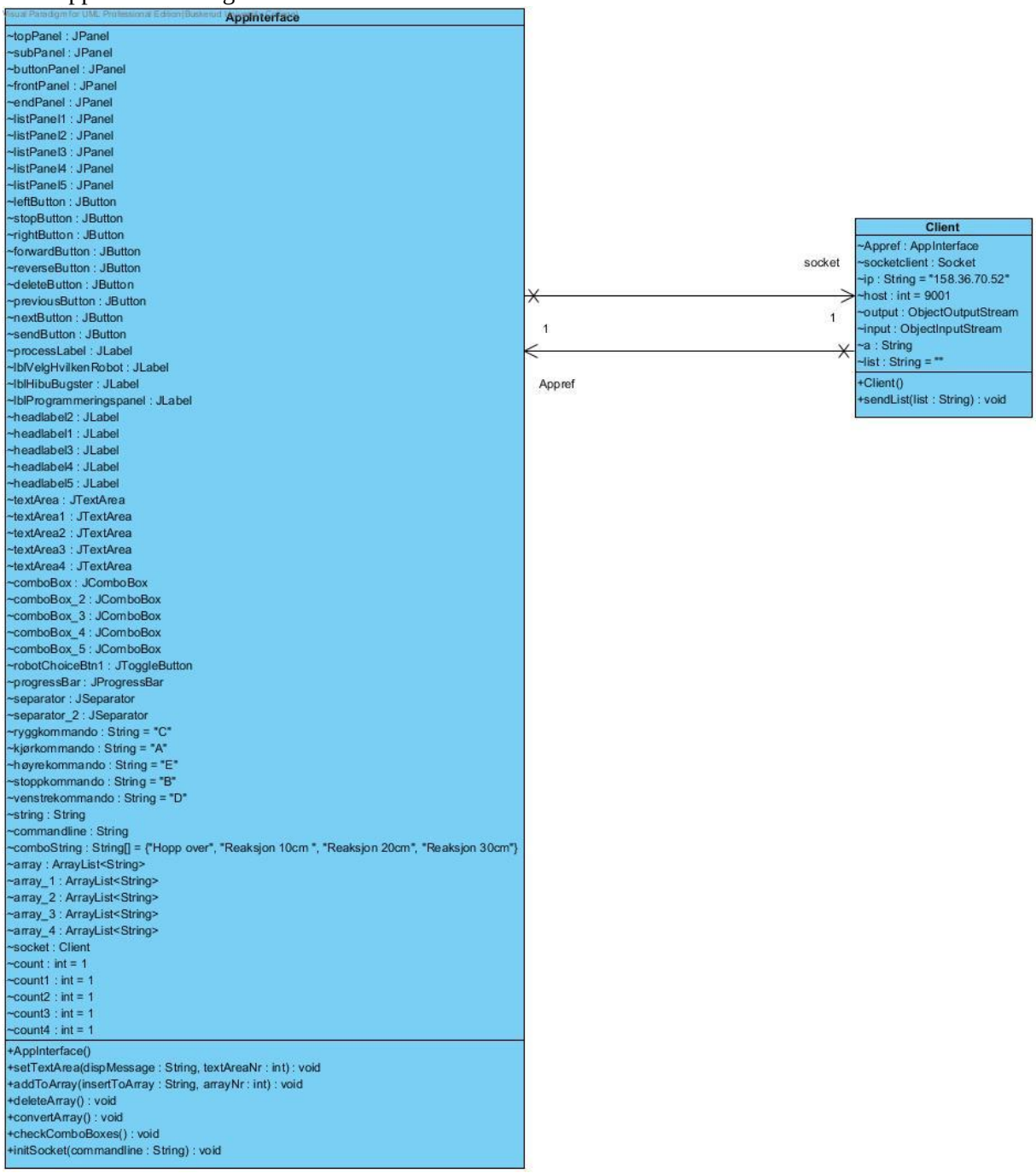
Vi skal nå forklare og redegjøre for hva denne Java Applikasjonen er for noe, hva kommunikasjonen består av, og ikke minst hva som foregår der i andre enden, hos serveren. Vi skal også igjen gi en beskrivelse av hva som foregår når først informasjonen har endt opp hos serveren.



6.1 JAVA KLIENT

Klienten som er laget er en usignert Java Applet, utviklet i utviklingsmiljøet Eclipse. Appleten består av en brukergrensesnittdel, og en socket klient-del. Disse delene er representert som hver sin klasse og objekt.

Java Applet Klassediagram:



Appletens oppgave er å ta i bruk en brukers kommandoer, og så videreformidle disse til serveren. Brukerens kommandoer blir bevart ved å lagre dem i såkalte ArrayLists. Kommandoene tilgjengelig er begrenset til kun enkle forflytningskommandoer og er:

Kjør frem:
Denne kommandoen gir roboten beskjed om utføre fremdrift på robotens bakre aksel. Fremdriftens varighet er på ca 2 sekunder. Kjør frem er representert av bokstaven A.

Rygg bakover:
Denne kommandoen fungerer som det helt motsatte av 'Kjør frem' funksjonen, hvor den bakre akselen reverserer i stedet for fremdrift. Ryggingen varighet er på ca 2 sekunder. Rygg er representert av bokstaven C.

Sving venstre:
Denne kommandoen gir roboten beskjed om å vri på forhjulet, plassert under i front av roboten. Kommandoen retter derimot ikke opp hjulenes posisjon, noe som betyr at hjulene blir å holde den gitte posisjon til den får ny beskjed om å endre hjulstillingen. Hjulenes forflytning i posisjon er lik 45 grader. Sving venstre er representert av bokstaven D.

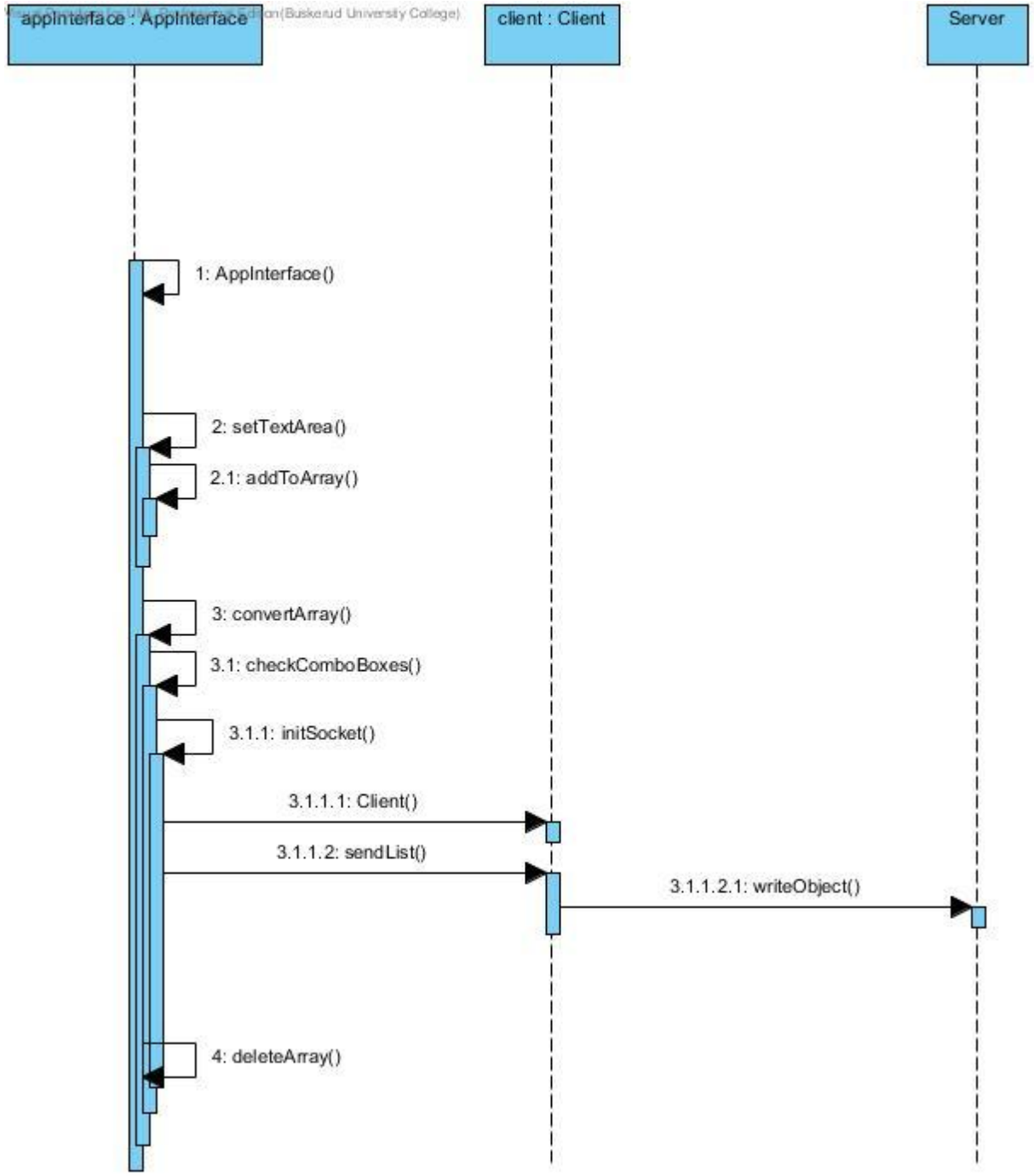
Sving høyre:
Slik som rygg er motsatt til 'Kjør frem', er sving høyre, logisk nok, motsetningen til sving venstre. Likt som med 'Sving venstre' retter heller ikke denne funksjonen opp hjulene i opprinnelig posisjon. Hjulenes forflytning i posisjon er lik 45 grader. Sving høyre er representert av bokstaven E.

Stopp:
Denne kommandoen representerer ingen direkte forflytning eller endring da den kun gir roboten beskjed om å stå stille i et øyeblikks tid. Ved påkalling av denne kommandoen vil roboten da kun stoppe opp en kort periode på ca. 2 sekunder. Stopp er representert av bokstaven B.

Rett opp hjul:
Denne funksjonen er den siste, og denne vil rette opp de fremre hjulene slik at roboten kan ha en normal fremover vendt fremdrift. Dersom denne kommandoene blir kalt, og hjulene allerede befinner seg i fremover vendt posisjon, vil det ikke skje noen som helst merkbar endring for brukeren. Rett opp hjul er representert av bokstaven R.

Appletens brukergrensesnitt er utviklet ved benyttelse av Eclipse sin innebygde GUI Editor. Brukergrensesnittet består av syv trinn; et hvor man gjør et valg av hvilken robot man skal styre, så et trinn hvor man bestemmer kommandoene roboten skal utføre, deretter fire like steg hvor man bestemmer hva roboten skal gjøre i gitte situasjoner, så til slutt en trinn hvor man sender all informasjon til serveren. Trinnene fungerer hovedsakelig ved at paneler byttes ut og at brukergrensesnittets elementer veksles mellom status som synlige og ikke synlige.
Blant de brukte elementene i brukergrensesnittet har vi JtextField, JtextArea, Jlabel, Jpanel, JprogressBar, Jbutton, Jseperator, JtoggleButton og JcomboBox.

Java Applet Sekvensdiagram:

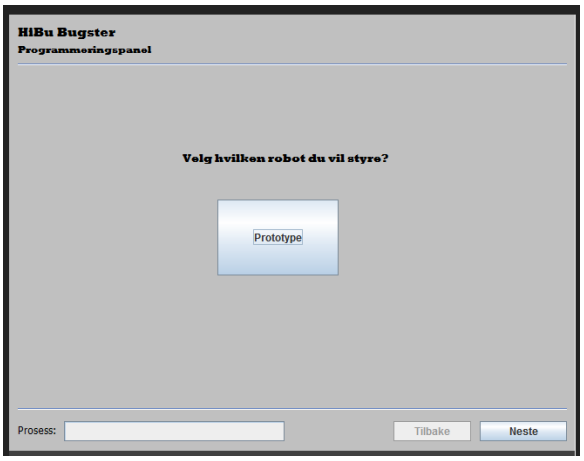


En Java Applet startes opp ved å oppføre en applet-tag i et html eller php-dokument. Denne applet tagen ser slik ut:

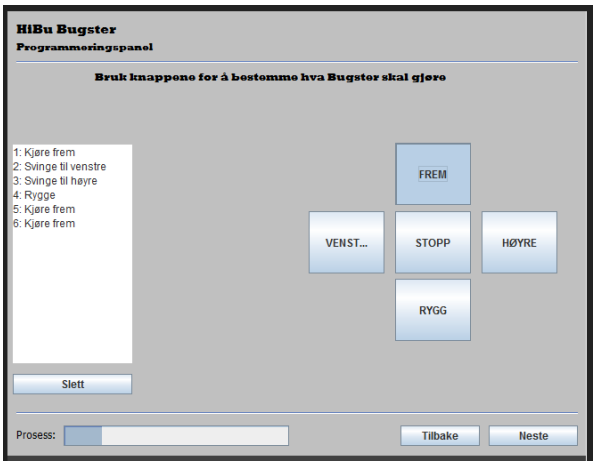
```
<applet code="AppInterface.class"
        width="600"
        height="500">

</applet>
```

Classfilen som skrives opp skal kun være den filen som inneholder startmetoden init() i seg, da denne for en Java Applet erstatter det som ellers i en vanlig Java Applikasjon ville være en main() metode. Selv om dette er grunnregelen for oppstart av en Java Applet, har vi kunnet unngå det i vårt tilfelle. Vår Java Applet applikasjon har ingen init() metode. Vi har likevel klart å få vår applikasjon til å fungere rett, selv om benyttet dokumentasjon og researchmatriell for Java Appleter bekrefter at en init-metode er nødvendig. I vårt tilfelle startes kun opp den overnevnte klassens konstruktørmetode. Denne metoden, AppInterface, har som oppgave å sette opp alle de forskjellige benyttede Swing-elementene som er brukt, samt deres tilknyttede lyttere.

Vi har da fått opp et brukergrensesnitt som venter på brukerens respons i form av trykking av knapper. Første trinn i prosessen er som nevnt tidligere at en bruker først avgjør hvilken robot man ønsker å benytte. Denne bestemmelsen gjøres ved at man velger robotens navn som er tilegnet en JtoggleButton. Vi har i vår applet kun hittil en tilgjengelig robot, og derfor kun en knapp som er tilgjengelig.



Etter å ha valgt 'Neste'-knappen kommer man til området hvor man bestemmer hvilke kommandoer som skal benyttes og hvilken rekkefølge disse skal i. Det er blitt tilegnet ActionEvents til hver kommandoknapp, hvor et knappetrykk på en av disse vil lagre kommandoen i en ArrayList, samt liste opp brukerens valg i et tekstfelt til venstre i brukergrensesnittet. Kommandoen vil i ArrayListen bli oppført som en bokstav fra mellom A - E. Hver bokstav representerer en valgt kommando. Et eksempel på dette er at *en* 'Kjør frem'-kommando vil være representert ved *en* bokstav av typen A. Etterhvert som brukeren trykker inn sin rekkefølge vil flere og flere kommandoer føres opp i dette tekstfeltet. Disse er også blitt nummerert slik at brukeren lett kan holde følge med hvor mange kommandotrinn man lager.



Brukeren har også tilgjengelig en slett-funksjon, deleteArray(), som kalles på ved et trykk på slettknappen under tekstfeltet. deleteArray()-metoden vet selv, da det finnes flere ArrayLists som dannes i systemet, hvilken ArrayList som skal slettes.

Disse trinnene gjøres for totalt 5 nivåer, hvor den første som på bildet over, er robotens grunnleggende kjørerute. De neste 4 trinnene dekker da som tidligere nevnt de forskjellige rutene som velges dersom spesielle betingelser inntreffer. I disse 4 trinnene har man i tillegg til de vist i illustrasjonen over, et ekstra felt for bestemmelse av sensitiviteten til robotens sensorer. Dette feltet er i form av en JcomboBox som lar brukeren velge mellom følgende 4 alternativer:

Hopp over: Velger brukeren denne funksjonen vil man kunne slippe å måtte bestemme en rekkefølge for denne hendelsen. Appleten vil da selv velge en rute som skal benyttes. Reaksjonsavstanden settes til 20cm.

Reaksjon 10cm: Ved valg av denne muligheten vil sensorene iverksette den gitte kommandorekkefølgen dersom noe befinner seg bare 10 cm eller mindre fra roboten.

Reaksjon 20cm: Ved dette valget vil sensorene slik som ovenfor reagere ved en hindring opp til 20 cm unna roboten.

Reaksjon 30cm: Her vil reaksjonsavstanden settes opp til 30 cm unna.

De forskjellige reaksjonsalternativene er representert med hver sin bokstav. En tabell for de forskjellige tilegnede bokstavverdiene ser slik ut:

	Reaksjonsavstand 10cm	Reaksjonsavstand 20cm	Reaksjonsavstand 30cm
Hindring bak	F	G	H
Hindring foran	I	J	K
Hindring til venstre	L	M	N
Hindring til høyre	O	P	Q

Brukerens valg av reaksjonsavstand vil senere bli flettet inn i de forskjellige ArrayListene som er laget for hver kjørerute.
Etter at brukeren har gjort sine valg i alle de hittil nevnte trinnene, vil brukeren komme til et siste panel, hvor brukeren må trykke en knapp markert 'Send' for å bekrefte alle de kjørerutene som er blitt valgt.

Send-knappen iverksetter da metoden convertArray(). Denne konverterer alle valgte kjøreruter om til et String-format. Dette fordi alle de kjørerutene som brukeren har valgt seg ut har vært lagret i sine egne ArrayLister. Denne konverteringen gjøres sammen med at alle de valgte JcomboBoxene sjekkes. Mellom de forskjellige bokstavrekkefølgene, som et resultat av kjørerutene brukeren har valgt, legges det inn i mellom sifferene 2, 3, 4, 5, 6. Dette for å skille mellom de forskjellige rutene, siffer 6 markerer slutten av stringen.

Etter å ha sjekket JcomboBoxene, settes alle de forskjellige stringene sammen til en hel stor string. Her har vi et eksempel på en ferdig string:

AAAEARC2FCDAARB3KAA4AEAR5DARAA6

Dersom vi raskt skal analysere denne stringen, så vil vi kunne dele den opp slik:

AAAEARC + 2 + FCDAARB + 3 + KAA + 4 +LAER + 5 + ODARAA + 6

Kjørerute:	Hindring foran:	Hindring bak:	Hindring venstre:	Hindring høyre:
AAAEARC	FCDAARB	KAA	LAER	ODARAA

Når stringene er samlet til en hel string kaller convertArray()-metoden på initSocket(), som oppretter et socketobjekt av klassen Client.

Client-konstruktøren oppretter en sockettilkobling til vår server, og så sender den den ferdige kodestringen, via metoden sendList(), til JavaServeren som allerede i andre enden av sockettilkoblingen nå står klar og lytter etter data.

Ved dette punktet er Java Appleten ferdig med sin oppgave, og brukeren kan klikke på nettsiden på linken for webkamera for å se roboten i aksjon. Når roboten er ferdig med sin oppgave. Kan man laste inn Appleten på nytt for å lage en ny serie med kommandoer.

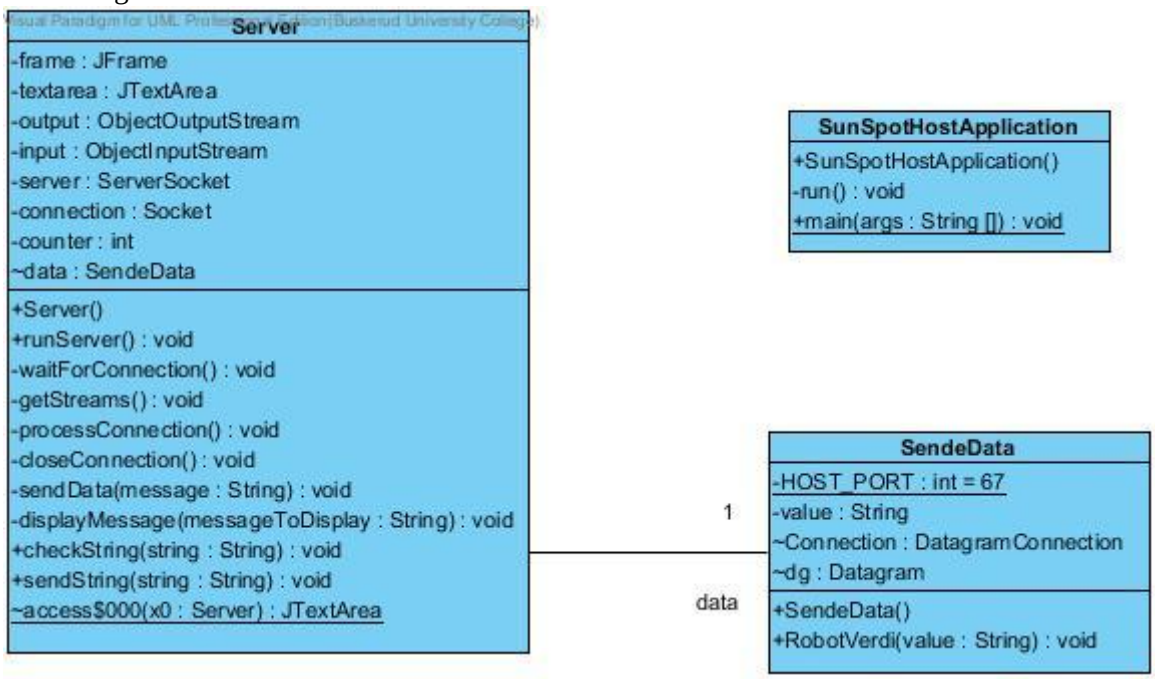
Appleten og nettsiden den er lokalisert på er hostet av vår egen server. Dette av den grunn av at Java Appleten ikke er signert og sertifisert av eksterne kontrollører som f.eks VeriSign, da dette kreves for at en Java Applet skal kunne tilkoble seg til en tredjeparts IP-adresse. Da Java Appleten er usignert vil den kun kunne tilkoble seg fra maskinen som kjører den i nettleseren sin, til den maskinen som hoster Java Appleten.

Strukturen for sockettilkoblingen er laget med grunnlag fra et eksempel fra tidligere støttelitteratur, og er en videreutviklet utgave fra en tidligere Java Applet Prototype.
For kildekoden til Java Applet Applikasjonen, se vedlegg 4.

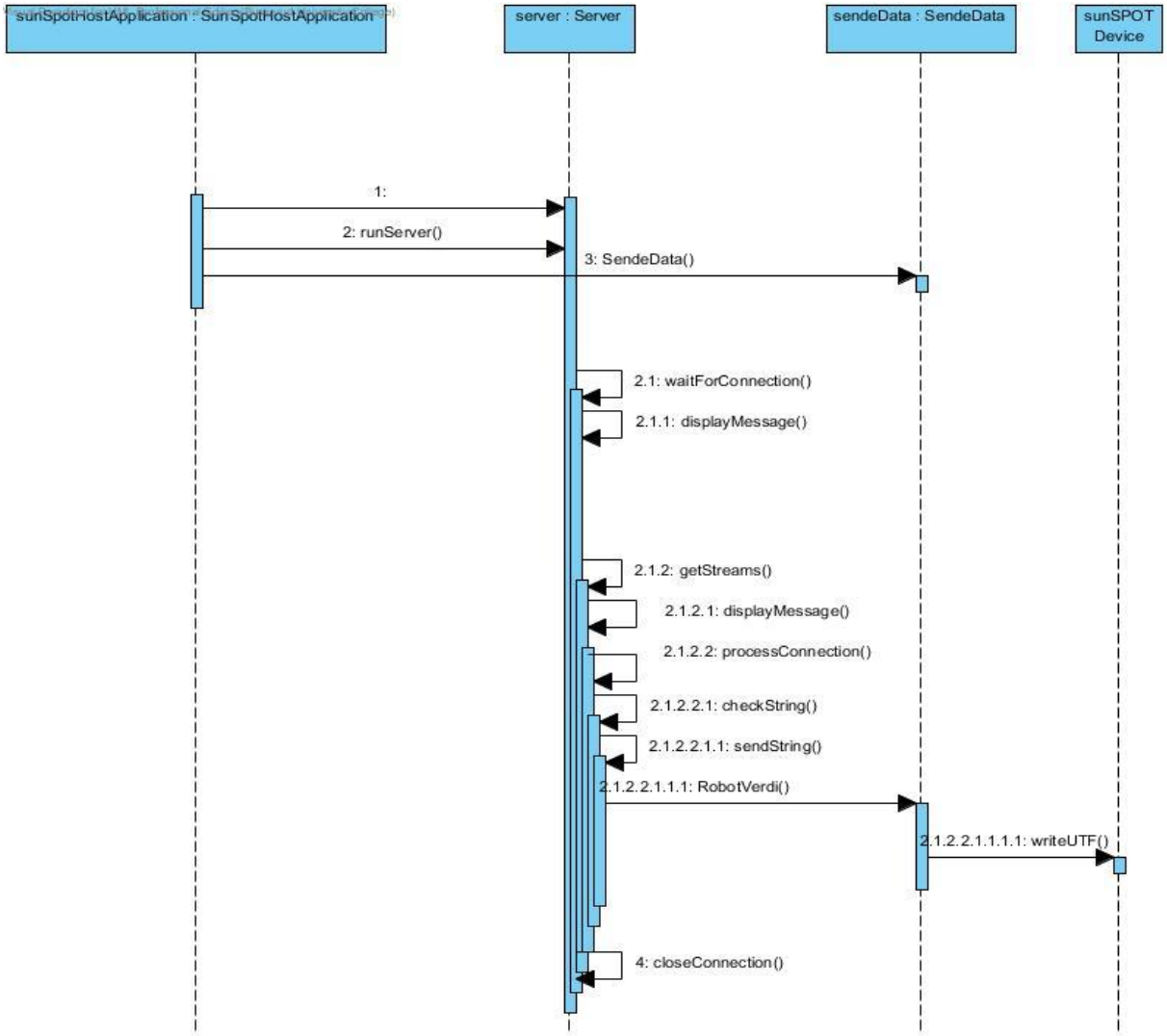
6.2 JAVA SERVER

I andre enden av sockettilkoblingen har vi en Java Applikasjon som kjører på vår stasjonære server. Applikasjonen fungerer som en socketserver, som kun har i oppgave å stå standby og lytte etter tilkoblinger gjort fra vår Java Applet. Denne serveren har ingen spesielle funksjoner, og er fullt og helt selvopererende og har ingen behov for menneskelig innblanding da den først er blitt startet opp. Serveren består av tre klasser, hvor to av disse står for kommunikasjon i hver sin ende. Den ene tar for seg innkommende kommunikasjon mottatt fra Java Appleten, og den andre videreformidler mottatt informasjon til sunSPOT basestasjonen og der igjen trådløst videre til sunSPOT-enheten lokalisert på roboten.

Klassediagram av Java Serveren:



Sekvensdiagram av Java Serveren:



Serveren vil i det man starter opp applikasjonen starte en sunSPOT base-funksjon kalt OTACommandServer. OTA står for OnTheAir, noe som setter sunSPOT basestasjonen i en tilstand hvor den gjør seg klar til å sende informasjonspakker via sitt radionett. Deretter vil oppstartklassen SunSPOTHostApplication, lage et objekt av klassene Server og sunSPOTserver. Serverklassen vil deretter starte og lytte etter tilkoblinger fra Java Appleten.

Det blir også av serverklassen opprettet et applikasjonsvindu kun bestående av et tekstområde, som kun av testårsaker viser frem hvilke verdier som mottas. Dette gjøres av metoden displayMessage. I det serveren mottar en tilkobling på port 9001, vil den opprette et socket leseobjekt. I det stringen fra klienten er blitt mottatt, startes det en checkString og en sendString metode. Disse skal blant annet filtrere bort unødvendige tegn og tomrom som befinner seg i stringen fra Java Appleten, og så sendes stringen via sunSPOT basestasjonen.

Til slutt lukkes tilkoblingen etter at stringen er overført til sunSPOT-enheten. Java Serveren er da automatisk klar for nye tilkoblinger.

Strukturen for sockettilkoblingen er laget med grunnlag fra et eksempel fra tidligere støttelitteratur, og er en videreutviklet utgave fra en tidligere Java Socket Prototype.
For kildekoden til Java Server Applikasjonen, se vedlegg 5.

7 Sun SPOT

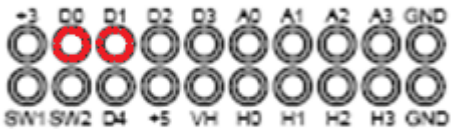
- Sun Small Programmable Object Technology

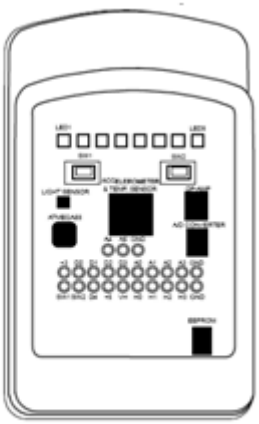
Sun SPOT er en liten trådløs enhet utviklet av Sun Microsystems. Den benytter seg av Squawk Java Virtual Machine(VM) som kjører på prosessoren uten ett underliggende operativsystem.
Enheten består av to kretskort (eSPOT Main Board og eDemo Board) og ett batteri.

For å kommunisere med en Sun SPOT enheten og en PC, benyttes en basestasjon som kobles til PC via USB. Disse kan da kommunisere trådløst ved å benytte seg av ett portnummer, eller benytte seg av IEEE adressen til den aktuelle Sun SPOT enheten. Flere eksterne Sun SPOT enheter kan også kommunisere seg imellom. Alt fra en enkel verdi til hele applikasjoner kan sendes trådløst mellom to enheter.

For å kommunisere med mikrokontrolleren har vi valgt og tatt i bruk UART (universal asynchronous receiver/transmitter).

eDemoBoard inneholder fem GPIO (General Purpose digital I/O) D0-4. D0 og D1 kan i tillegg til å benyttes som I/O porter, også benyttes som UART kanaler. D0 vil da fungere som mottaker (Rx) og D1 som sender (Tx).





Ved å implementere klassen EDemoBoard fra Sun SPOT API kan vi kontrollere alle funksjonalitetene som finnes på dette kretskortet. Denne lar oss lese og skrive til mikrokontrolleren ved å benytte metodene *readUART* og *writeUART*.

8 PROGRAMMERING AV SUN SPOT

8.1 BUGSTERAPP

Denne applikasjonen skal lastes inn på hver Sun SPOT enhet. Det er den som er ansvarlig for at brukerens kjøreprogram blir utført. Den skal tolke brukerens kode og sende kommandoer til mikrokontrolleren som fullfører oppgaven, samt ta i mot og tolke informasjon fra mikrokontrolleren.

8.1.1 KLASSENE

8.1.1.1 CONTROL KLASSEN

Det er denne klassen som oppretter en radioforbindelse med en basestasjonen for å sende over et datagram som inneholder en string verdi. Den oppretter også et objekt av klassene *Buffer* og *DrivingRoute*.Control sin oppgave er og hele tiden kontrollere alle verdiene til sensorene samt hvilken retning roboten kjører og om den står stille eller ei. IR sensorene skal hele tiden sjekkes opp mot de verdiene brukeren har satt for hver enkelt IR sensor. Alle verdiene foruten de brukeren har satt, henter Control fra Bufferklassen. Control benytter seg av de ulike get-metodene til Buffer for å få tak i de verdiene den trenger. Klassen har også ansvaret for og alltid gi kommandoer til mikrokontrolleren for å fortelle hvordan den skal styre roboten.

<u>Består av metodene :</u>

- startApp
- readUart,
- controlString
- handling
- pauseApp
- destroyApp.

.

8.1.1.1.1 STARTAPP()

Det første som blir gjort er å åpne en port for å kommunisere med en basestasjon koblet til server for å ta i mot et datagram med en string verdi. Så blir det gjort et kall på metoden controlString samt metoden drivingPattern, som ligger i klassen DrivingRoute.

Metoden vil nå gå inn i en while løkke som går helt til applikasjonen har utført kjøreprogrammet til bruker.
Det første programmet gjør etter å ha gått inn i whileløkka er å sjekke brukerens ønsket kjørestatus ved å kalle på metoden getbDrive, for så å sjekke om denne verdien er 5. Er verdien 5, kalles metoden notifyDestroyed, og applikasjonen stopper opp å kjøre. Er den ikke 5, går programmet videre med å kalle på metodene getDrive, getReverse, getTurn, getSSensorene, getIrSensorene og lagrer motorens kjørestatus (drive), svingmodus(turn), infrarødsensorene (IrSensorer) og tilslutt støtsensorene.

Programmet vil sjekke om noen av støtsensorene er slått inn. Er en eller flere registrert, vil hver og en sensor bli sjekket for å registrere hvilke sensorer dette er. Grunnen til at det er blitt lagt opp på denne måten er for å minske antall metodekall. Selv om vi i denne applikasjonen ikke har hatt behov for å skille mellom de ulike støtsensorene, har vi like vel valgt å legge opp til muligheten til å gjøre dette ved en senere anledning. Hvis ingen av støtsensorene har blitt registrert, blir hver støtsensor satt til false.

IR sensorene er programmert på samme måte. Et metodekall til getIrSensorene() vil gi svar på om noen av IR sensorene har registrert en hindring, og vil deretter sjekke hver og en sensor. Forskjellen fra støtsensorene er at vi her har behov for å vite hvilke av infrarødsensorene som er slått inn. Alle IR enorene vil bli satt til max verdi om ingen av dem har registrert en hindring.

En if setning sjekker hvilke retning bruker ønsker å kjøre, og gir beskjed til mikrokontroller om å utføre denne handlingen.

If (bDrive == x)
Denne løkken sjekker om brukerens kjøremønster i forhold til om den er: 0,1 eller 2 for deretter å gi beskjed til mikrokontroller om brukerens ønske.

If(bDrive ==1)
Brukeren har satt at roboten skal kjøre framover. Her vil programmet helle tiden passe på at roboten skal kjøre framover så lenge ingen av sensorene har varslet om en hindring. IrBack og NBack er ikke tatt med i betraktning. Så om IrBack eller NBack varsler om en hindring, vil dette bli ignorert.

IRsensorene IrFront, IrLeft og IrRight blir kontrollert hver for seg mot eventuelle minimums verdier som bruker har satt. Er en av IR sensorene lavere enn brukerens ønske, samtidig som motoren er i fremdrift(drive = 1), skal metoden Handling med integer variabelen kalles. En for-løkke med variabelen *count* går igjennom hele rekka med handlinger brukeren har valgt roboten skal utføre, for den IR sensoren som har registrert en hindring. For hver gjennomgang vil den kalle *handling* som vil returnere en byte verdi. Denne verdien blir sendt til mikrokontrolleren.

Viser det seg at verdien som *handling* returnerer er 0x11 eller 0x13, som står for framdrift og revers verdi , skal hoved tråden sove i to sekunder for deretter å fortsette videre. Grunnen til dette er at programmet ellers ville gitt en ny beskjed rett etter og da ville ikke roboten ha rygget eller kjørt fremover i det hele tatt.
For IrFront sensoren lar vi med vilje programmet hoppe over verdien 0x11. Mer detaljert beskrivelse om dette kan leses i avsnitt 9.3.

Om en av sensorene registrerer en hindring og motoren har stoppa (drive = 0) skal ikke programmet her gjøre noe som helst.

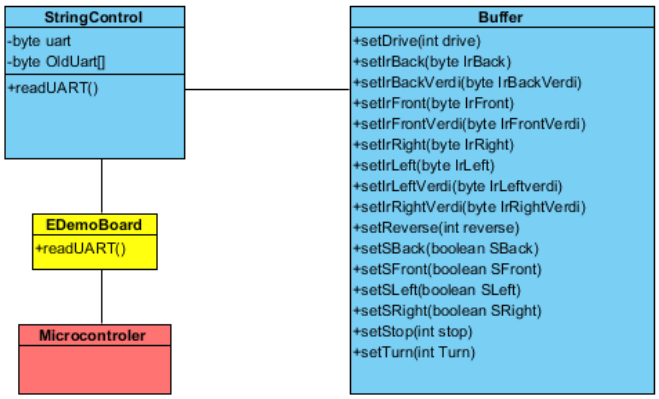
Alle støtsensorene utenom SBack blir kontrollert ”samtidig”, siden alle utløser lik handling, - sende stopp verdien 0x12 til mikrokontrolleren.

Nest siste kontrollpunkt i denne if-setningen er å sjekke alle sensorene er ”Fri” samtidig som motoren har stoppet. Da skal det sendes en startverdi til mikrokontrolleren.

If(bDrive == 0)
Brukeren ønsker at roboten skal stoppe. Programmet sendere en stoppverdi til mikrokontrolleren.

If(bDrive == 2)
Denne koden er programmert likt som under if(bDrive ==1) men byttet om på verdiene drive og reverse.

8.1.1.1.2 READUART();



Oppretter en ny tråd når den blir kalt, og vil gå parallelt med main tråden.
Metoden har oppgave å lese av alle innkommende signaler fra mikrokontrolleren. Disse blir kontrollert og sendt over til Buffer, som lagrer verdiene.

En if-setning går systematisk gjennom hver kommando og oppdaterer deretter setmetoden for denne verdien.

For hver gang kommandoene start, stopp eller rygg mottas, kalles de tre metodene getDrive(),setStop() og setReverse()for oppdatering.

For å minske antallet kall på setmetodene, lar vi ikke samme setmetode bli kalt på nytt med mindre den får en ny verdi.

8.1.1.1.3 CONTROLSTRING();

Denne metoden plukker først og fremst ut brukerens kjøremønster fra stringen *userString* som er mottatt fra server og delt opp i hovedsak to deler. En for kjøreruten og en for hvordan roboten skal oppføre seg i ulike situasjoner. Kjøreruten blir sendt over til DrivingRoute.
Videre vil denne metoden sjekke hvilke IR sensorer brukeren velger å ta i bruk, samt hvor nærme en gjenstand skal være før roboten skal reagere.

Hver enkelt char i *userString* vil bli sjekket og lagret i arrayet UserArray ved å benytte seg av metoden charAt() med *i* som parameter, fra klassen StringBuffer i Javas API. Når UserArray [i] = 2 skal UserArray kopieres over til arrayet *DriveArray*. Alle char verdiene som blir kopiert over til DriveArray er da selve kjøreruta til roboten.

Resten av char verdiene som ligger i userString er de verdiene som forteller hvordan bruker ønsker at robot skal oppføre seg i ulike situasjoner.

A2 får tilegna plasseringen som char verdien 2 ligger i og StartFunction får tilegna plasseringen til den første char verdien som kommer etter 2.

Vi kunne selvsagt droppet StartFunction og benyttet oss kun av A2 verdien, men vi syntes det var mer oversiktelig å ha en egen variabel som skiller kjøreruta til roboten fra funksjonene roboten skal gjøre. Løkka vil deretter gå igjennom resten av userString og sette plassering for variablene A3-A6.

En for-løkke går igjennom hele UserArrayet fra og med StartFunctions plassering til og med siste plassering. Dette for å kontrollere om brukeren har valgt å ta i bruk noen av IR-sensorene og hvilken avstand disse skal registrere en hindring på.

8.1.1.1.4 HANDLING();

Når denne metoden blir kalt blir parameteren representerer som en plassering i UserArray. Det blir sjekket hvilken verdi denne plasseringen inneholder og returnert en byte verdi etter som hva innholdet er.

8.1.1.2 BUFFER

Denne klassen har ansvaret å lagre alle verdiene til sensorene og motorene. Buffer har en rekke med set- og get-metoder som både StringControl klassen og Driving Route klassen benytter seg av. Nye verdier blir overskrevet av gamle uansett om de er lest eller ikke.

Hver metode er synkronisert slik at kun én tråd får tilgang til samme metode om gangen. Alle metodene avslutter med å kalle notifyAll for å varsle alle tråder om at nå er denne metoden ledig.

8.1.1.2.1 GET-METODER

Bufferklassen inneholder en rekke get-metoder. Deres oppgave (med unntak av to metoder) er å returnere en integer-, byte- eller boolean-verdi av en variabel med samme navn som metoden som returnerer. Metodene getIrSensorene og getSSensorene er litt annerledes en de andre da de skal sjekke flere variabler for deretter å sende en booleanverdi ut i fra disse resultatene.

getIrSensorene();
Denne metoden sjekker om alle variablene IrRight, IrLeft, IrFront og IrBack er større en brukerens minimumsverdi for IR sensorene. Oppfylles disse kravene, returnerer metoden boolean verdien false. Er det en eller flere av disse kravene som ikke oppfylles, returnerer metoden true.

getSSensorene();
Denne metoden sjekker først om alle variablene SRight, SLeft, SBack og SFront er satt til false. Oppfylles disse kravet, returnerer metoden boolean verdien false.
Er det en eller flere av disse kravene som ikke oppfylles, returnerer metoden boolean verdien true.

Resterende getmetoder:

getDrive();
getStop();
getReverse();
getTurn();
getIrBack();
getIrFront();
getIrLeft();
getIrRight();
getSBack();
getSFront();
getSLeft();
getSRight();
getSSensorene();
getbDrive();

8.1.1.2.2 SET-METODER

setDrive();
Denne metoden lagrer robotens fremdriftstatus til integer variabelen drive.

setStop();
Denne metoden lagrer robotens fremdriftstatus til integer variabelen stop.

setReverse();
Denne metoden lagrer robotens fremdriftstatus til integer variabelen reverse.

setTurn();
Denne metoden lagrer hvilken posisjon hjulene har, i integer variabelen turn.

setIrBack();
Denne metoden lagrer verdien til IR sensoren plassert bak på roboten i byte variabelen IrBack.

setIrFront();
Denne metoden lagrer verdien til IR sensoren plassert i front på roboten i byte variabelen IrFront.

setIrLeft();
Denne metoden lagrer verdien til IR sensoren plassert på venstre side av roboten i byte variabelen IrLeft.

setIrRight();
Denne metoden lagrer verdien til IR sensoren plassert på høyre side av roboten i byte variabelen IrRight.

setSBack();
Denne metoden lager verdien til støtsensoren plassert bak på roboten i boolean variabelen SBack.

setSFront();
Denne metoden lager verdien til støtsensoren plassert foran på roboten i boolean variabelen SFront.

setSLeft();
Denne metoden lager verdien til støtsensoren plassert på venstre side av roboten i boolean variabelen SLeft.

setSRight();
Denne metoden lager verdien til støtsensoren plassert på venstre side av roboten i boolean variabelen SRight.

setbDrive();
Denne metoden lagrer brukerens verdi satt for motoren.

8.1.1.3 DRIVINGROUTE KLASSEN

Denne klassen har ansvar for å utføre den kjøreruten som bruker har valgt. Den sjekker en og en kommando i kjøreruten som blir utført før den går videre til neste på listen. Det er lagt inn en timer som teller i to sekunder for hver gang roboten skal kjøre fremover eller rygge. Denne teller kun når robot er i bevegelse. Den vil ikke utføre neste kommando på listen før roboten har beveget seg i disse to sekundene. Dette er for å garantere at roboten faktisk kjører når den har fått beskjed om å kjøre.

Klassen består av to metoder:

- DrivingPattern,
- UserDrive.

8.1.1.3.1 DRIVINGPATTERN()

Denne metoden inneholder en for-løkke som går igjennom hele arrayet UserString char for char. For hver plassering som inneholder char verdien A eller C skal UserDrive metoden kalles med 1 eller 2 som parameter. Om plasseringen inneholder verdien B, skal metoden setbDrive fra Bufferklassen kalles, og tråden skal sove i to sekunder. Verdiene D, E og R skal få DrivinPattern til å kalle på Bufferklassens metode setTurn med en integer verdi.

Da alle plasseringene i char Arrayet er kontrollert, skal denne metoden gi beskjed til mikrokontrolleren om å stoppe motoren med å kalle på setbDrive med 0 som parameter. Deretter skal metoden ta en pause på ett sekund og så kalle på metoden setbDrive med 5 som parameter som er den verdien som vil be main tråden om å avslutte.

8.1.1.3.2 USERDRIVE()

UserDrive har en integer parameter med navnet DriveStatus samt to interne integer variabler *status* og *i*. Før programmet kan gå videre, skal metoden formidle kjørestatusen. Dette gjøres ved å kalle på metoden getDrive fra bufferklassen. DriveStatus blir satt som parameter.

8.2 MANGLER I KODEN

Denne applikasjonen tar ikke hensyn til at roboten kan støte på en hindring nummer to mens den utfører en handling for hindring nummer en. Den kan kun løse én handling av gangen.
For å løse dette problemet må main tråden opprette en ny tråd som tar for seg hindringen, slik at den kan gå tilbake til oppgaven sin med å kontrollere alle sensorverdiene. Dukker det da opp en ny hindring, samtidig som den ny oppstartet tråden fortsatt utfører handlingen for forrige hindring, skal main tråden gi beskjed om at den ny oppstartet tråden avslutte dette arbeidet for så å utføre handlingsmønsteret som er satt for den nyeste hindringen som har oppstått. Da denne oppgaven er utført, skal den ikke gå tilbake til forrige oppgave, men vente på en ny oppgave å utføre.

Dette er også grunnen til at vi ikke lar roboten få kjøre fremover som en handling for IR sensor i front. Dette vil føre til at robot krasjer i gjenstanden uten å registrere det.

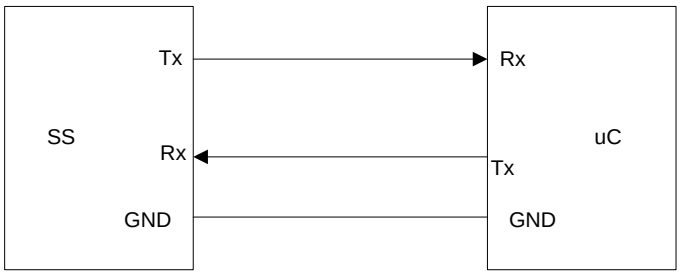
9 KOMMUNIKASJON SUN SPOT – MIKROKONTROLLER

Sun SPOT enheten som skal styre roboten har ikke nok I/O pinner til alle motorene og sensorer. Det er derfor nødvendig å bruke en mikrokontroller til å styre motorene og hente inn informasjon fra sensorer.

Kommunikasjonen mellom Sun SPOT enheten og mikrokontrolleren vil foregå serielt. Vi har valgt å bruke UART kommunikasjon. Vi mener dette passer best i våres system. Den viktigste grunnen til det er at begge enhetene har ferdige funksjoner for UART.

9.1 UART

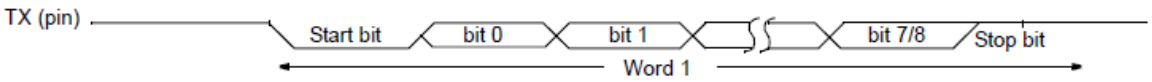
UART (Universel Asynchronus Reciver/Transmitter) er en type seriekommunikasjon. At den er asynkron betyr at enhetene ikke deler noe klokkesignal. Timingen blir da veldig viktig slik at informasjon ikke går tapt. Den er full Duplex, vi kan da sende og motta meldinger samtidig.



Figuren over viser hvordan enhetene er koblet til hverandre.

Som sagt tidligere er det viktig at timingen mellom de to enhetene. Det er baud raten som bestemmer timingen, vi har satt våres baud rate til å være 9600.

Figuren under viser hvordan kommunikasjonen foregår:



Når ingen informasjon sendes over linja er den satt høy, dette kalles idle state. Når en byte skal sendes settes linja lav i en periode, dette er en start bit. Etter det blir byten sendt. Det er her timingen kommer inn, byten må bli sendt med samme hastighet som mottakeren sjekker kanalen med. Etter byten blir linja satt høy, dette er stop biten. Deretter settes linja i idle state.

9.2 KOMMUNIKASJONSPROTOKOLL

Det må være klargjort en protokoll mellom Sun SPOT enheten og mikrokontrolleren slik at de kan tolke meldingen. Størrelsen på meldingen skal være på 8 bit (1 byte). Hver melding deles opp i to deler; en adressedel og en informasjon/kommando del

Adresse: Vær komponent på roboten får sin egen adresse. De fire første bitene er adressen til komponenten.

Informasjon/kommando: De fire siste bitene er informasjon/kommando. Denne innholder enten informasjon fra sensorer eller kommandoer til motorer.

Kommunikasjonsprotokollen ligger med som vedlegg.

10 ROBOT

10.1 ELEKTRONIKK

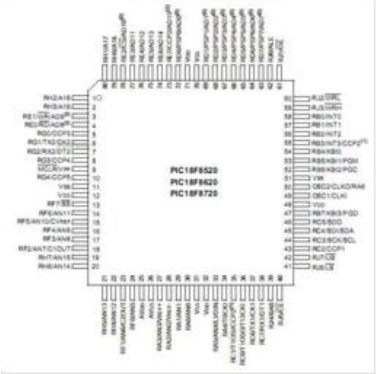
10.1.1 KRETSKORT

Kretskortet vil inneholde de nødvendige komponentene for at roboten skal være i stand til å utføre oppgavene gitt i kravspesifikasjonen. Mikrokontrolleren skal styre motor, servoer og sensorer. Den skal kommunisere med Sun SPOT enheten og tolke informasjonen som blir sendt. I tillegg skal den kunne overvåke batteriet og registrere når det begynner å bli utladet. Ladekretsen skal regulere spenningen og strømmen til batteriet under lading.

10.1.2 KOMPONENTER OG TILKOBLING

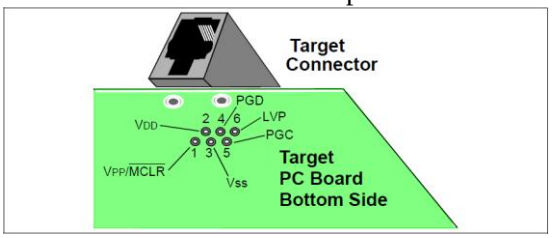
10.1.2.1 PIC18F8720

Mikrokontrolleren er av typen PIC18 fra microchip. Den har 80 pinner og vi bruker 28 av disse. Vi valgte denne da den har det nødvendige antall CCP porter og den var tilgjengelig på skolen. Mikrokontrolleren er programmert til å styre motoren, servoer og sensorer. Den kommuniserer med Sun SPOT enheten via UART porter på kontrolleren og Sun SPOT enheten. I tillegg registrer den når spenningen over batteriet faller til et bestemt nivå.



10.1.2.2 ICD 2

For å opprette kontakt med mikrokontrolleren kobler vi en ICD mellom PC og mikrokontroller. ICDen blir koblet til PCen med en standard USB kontakt og til fem av mikrokontrollerens pinner.



Gjennom denne kan vi laste inn programmer vi skriver på mikrokontrolleren. Vi bruker en pull-up motstand på 10k fra MCLEAR pinnen til 3.3V for å trekke porten høy.

10.1.2.3 REGULATORER

Kretskortet har applikasjoner som trenger 3 forskjellige spenninger.

IR-sensorer, ultralydsensor, støtsensor og Sun SPOT enheten trenger 5V. Vi trenger da en regulator som gir ut denne spenningen hele tiden, uavhengig om spenningen over batteriet synker. Ifølge databladet til regulatoren vil regulatoren trenge minimum 6.8 V på inputen for å kunne garantere 5 V ut. Blir lasten for stor, vil utgangsspenningen kunne variere hvis ikke regulatoren får denne spenningen. Batteriet vårt gir ut en spenning på ca 6V avhengig av tilstanden til batteriet, men gjennom tester vi har gjort, vil ikke utgangen på regulatoren variere. Hvis den skulle synke noe vil ikke det heller være et stort problem, sensorene opererer med spenninger fra 4-6V.

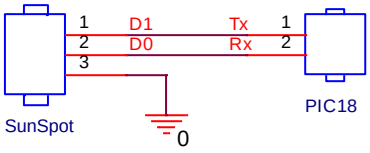
Mikrokontrolleren trenger 3.3 V. Vi bruker da en regulator som forsyner kontrolleren med denne konstante spenningen.

Motorene opererer ifra 4-6V. Vi vil derfor koble spenningen direkte fra batteriet til motorene. I utgangspunktet er denne spenningen 6.4V når batteriet er fulladet, men etter å ha koblet til sensorene vil spenningen falle til ca 6V. Man står derfor ikke i fare for å ødelegge motorene.

10.1.2.4 SUN SPOT

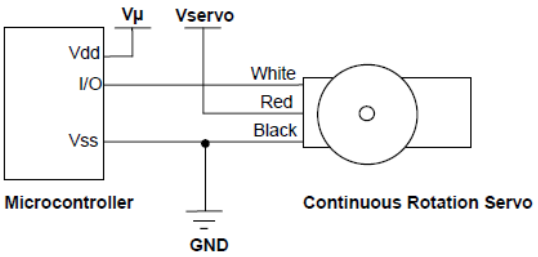
- Sun SPOT enheten blir forsynt med spenning og strøm av blyakkumulatoren. Vi kommer ikke til å overvåke batteriet til Sun SPOT enheten. Så lenge det er strøm på akkumulatoren vil ikke Sun SPOT enheten utlades, og akkumulatoren lades når den begynner å gå tom for strøm.

  Sun SPOT enheten er koblet til Tx1 (transmitter) og Rx1 (receiver) pinnene på mikrokontrolleren. Disse er koblet til D0 (receiver) og D1 (transmitter) pinnene på Sun SPOT enheten.



10.1.2.5 SERVOER

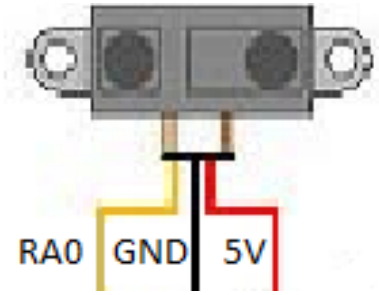
- Motoren er koblet til spenningsforsyningen og styresignalet er koblet til CCP porten RG0 på mikrokontrolleren. Den må være koblet til en CCP port fordi motoren blir styrt av pulsbreddemodulerte signaler.



- Servoene for manøvrering og ultralydsensoren er koblet til henholdsvis CCP portene RG3 og RG4 på mikrokontrolleren. Servoene er i likhet med motoren styrt av pulsbreddemodulerte signaler.

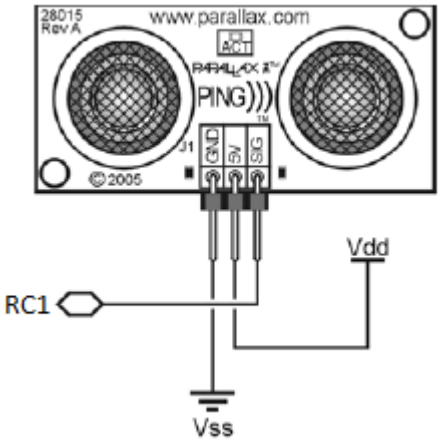
10.1.2.6 IR SENSORER

- IR-sensorene er koblet får spenning ifra 5V regulatoren og det varierende spenningssignalet som representerer avstand er koblet til pinnene RA0, RA1, RA2 og RA3 på mikrokontrolleren.



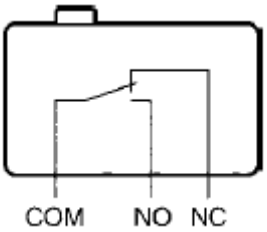
10.1.2.7 ULTRALYD

- Ultralydsensoren er koblet til CCP porten RC1. Dette gjøres for å benytte seg av Capture funksjonen til mikrokontrolleren.



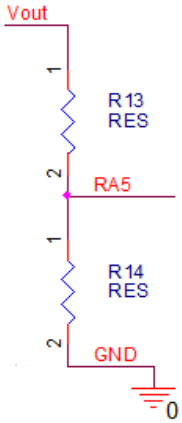
10.1.2.8 STØTSENSORER

- Støtsensorene er koblet pinnene RD1, RD2, RD3 og RD4. Hvis noen av disse pinnene mottar et spenningssignal betyr det at roboten har støtet bort i noe.



10.1.2.9 SPENNINGSFORDELER-DETEKTOR

Vi kommer til å bruke en av AD-konverterne på mikrokontrolleren til overvåke tilstanden til batteriet. Vi kan ikke sette en spenning på 6,4V over mikrokontrolleren. Vi setter derfor opp en spenningsdeler som halverer spenningen inn på mikrokontrolleren. Dette gjør vi ved å koble to like motstander i serie, som vist under



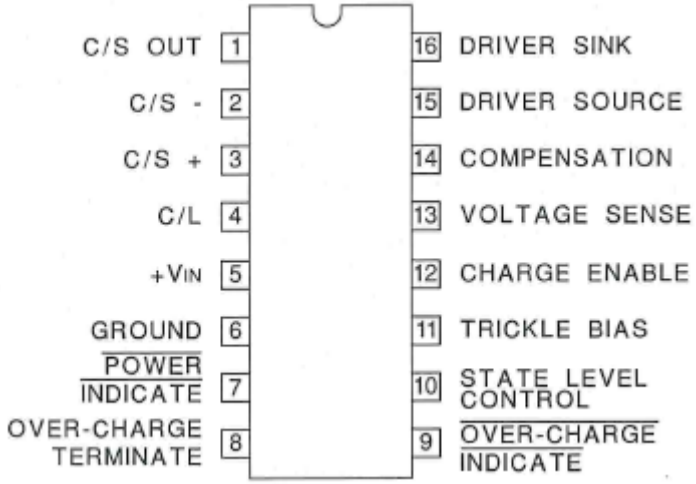
10.1.2.10 TESTING

Vi har utført tester på kretsen underveis i arbeidet med prosjektet. Vi har koblet opp kretsen på laben og testet motor, servoer og sensorer. Vi har opprettet kommunikasjon mellom Sun SPOT og mikrokontroller, og vi får sendt en kommando fra Sun SPOT til mikrokontrolleren.

10.1.3 LADEKRETS

10.1.3.1 BATTERILADER-LADING

Batteriladeren vi har valgt oss er av typen UC2906 fra Texas Instruments. Den overvåker og kontrollerer strømtilførselen og spenningen til batteriene. Avhengig av hvor mye spenning batteriene har vil laderen tilføre strømmen batteriene trenger.



Laderen skiller mellom tre forskjellige lade tilstander:

1. High current bulk-charge state:

   Denne tilstanden begynner da vi får kontakt med spenningsforsyningen. Laderen vil da tilføre batteriene en lav strøm(trickle bias,pin 11) til vi når spenningsterskelen Vt, som er satt av referansespenningen Rf(2.3V, dette er standard for hver enkelt batteri-celle, hvis spenningen øker mer vil man kunne skade batteriene hvis noen er kortsluttet). Når denne spenningen er nådd, vil laderen ”skru av” strømtilførsel fra pin 11 og ”skru på” driveren som tilfører batteriene maksimal strøm(pin 15, emitter i en npn transistor). Driveren blir styrt av separate strøm- og spenningsregulatorer. Disse kontrollerer hvor mye strøm og spenning utgangen av laderen gir.

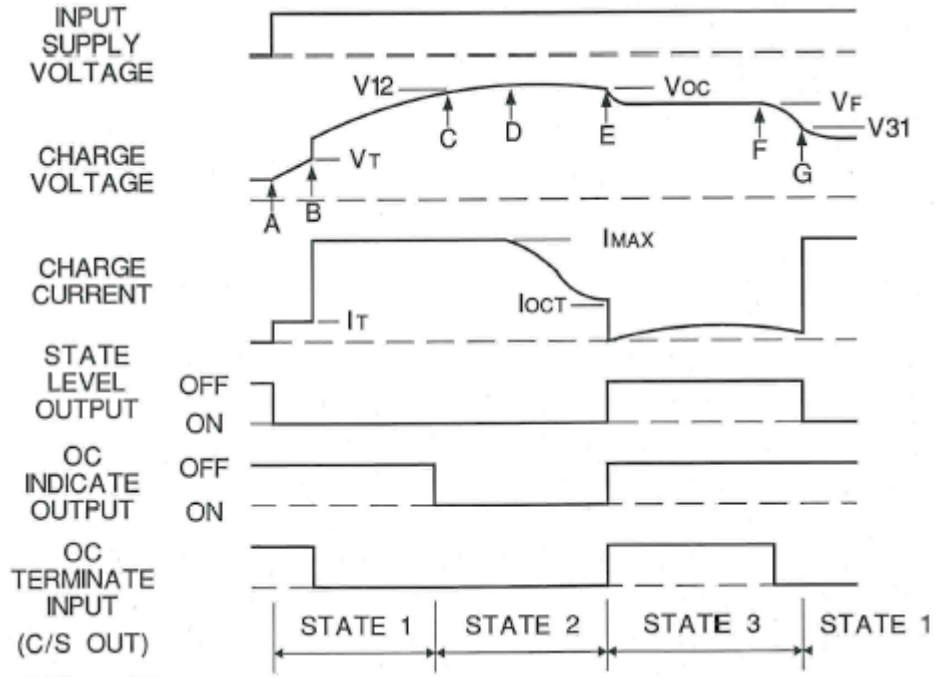
2. Over-charged state:

   Når batteriene når 95% av maksimal kapasitet(Voc) vil laderen gå inn i tilstand 2. Her vil den holde seg inntil Voc er nådd og strømmen begynner å minke. En strømsensor(utgang pin 1) vil sende en høy(1 V) til over-charge terminate inngangen(pin 8). Laderen går inn i float state og holder spenningen konstant.

3. Float-charge:

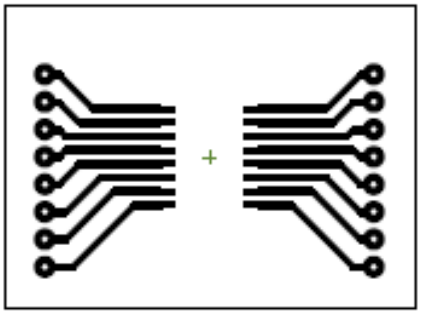
   Spenningen holdes konstant inntil en last begynner å tømme batteriet for strøm.

Vi kan se hendelsesforløpet på følgende figur.



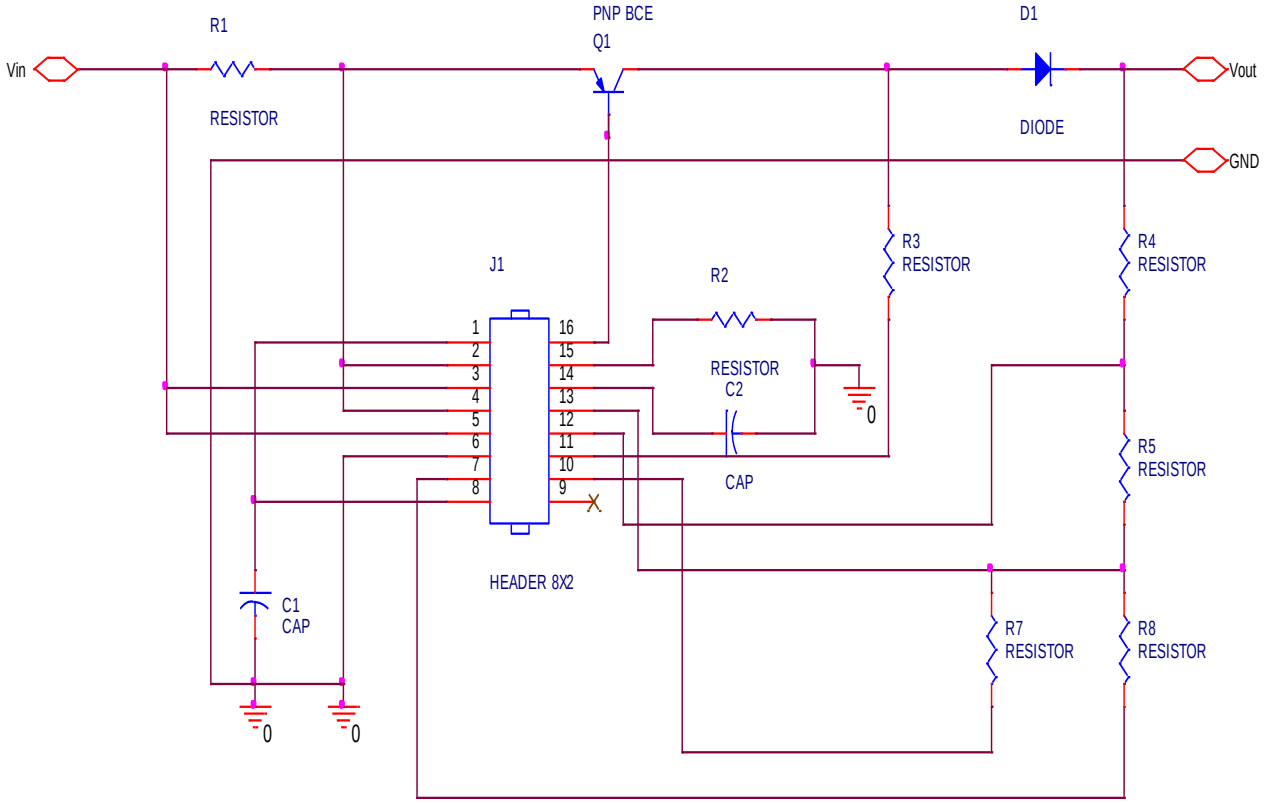
10.1.3.2 ETSING OG LODDING AV BATTERILADER

Laderen vi bestilte er av typen surface mount, og pinnene står for tett til å kunne koble rett opp på et koblingsbrett. Så for å kunne teste laderen måtte vi lage et lite kretskort vi kunne koble til koblingsbrettet på laben. Henviser til vedlegg 3 for testdokument.



10.1.4 LADEKRETS-DESIGN

Ladekretsen ble koblet opp slik, med batterilader, motstander, kondensatorer, pnp transistor og diode.



10.1.4.1 PASSIVE KOMPONENTER

De passive komponentene som inngår i ladekretsen består av

- Motstander, 6 ohm (sensemotsanden), 470 ohm, 220 ohm, 390k, 43k, 18k og 70k.
- Kondensatorer, 2 stk 0.1uF.
- Pnp transister
- Diode

10.1.5 KONSTRUKSJON AV KRETSKORT

Etter å ha tegnet kortet i Orcad PCB editor, skulle vi fysisk lage kortet her på skolen.

10.1.5.1 ETSING OG LODDING

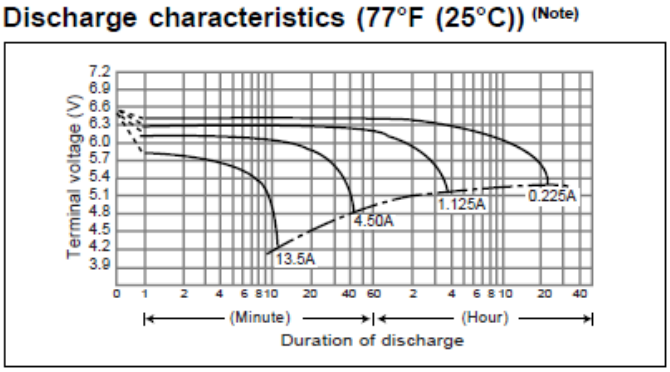
Når vi skal etse kortet, måtte vi først skrive ut en plastikk film med PCB tegningen printet på. Denne tegningen skal overføres til kortet. Dette gjør man ved å legge plastfilmen oppå kortet og belyser med UV-lys. Etter dette må man fremkalle kanalene av kobber som ikke skal etsetes bort. Så bader man kortet i en syre som etser bort kobberet. Kanalene kommer da tydelig frem slik PCB tegningen viser. Etter dette måtte vi borre hull og lodde på komponenter, kontakter og kontaktpinner.

10.1.6 TESTING AV FERDIG KRETSKORT

Etter kortet ble ferdig testet vi dette på laben. Vi målte verdier med digitale multimetre og fikk de samme verdiene som vi fikk når vi koblet opp kretsen på laben tidligere.

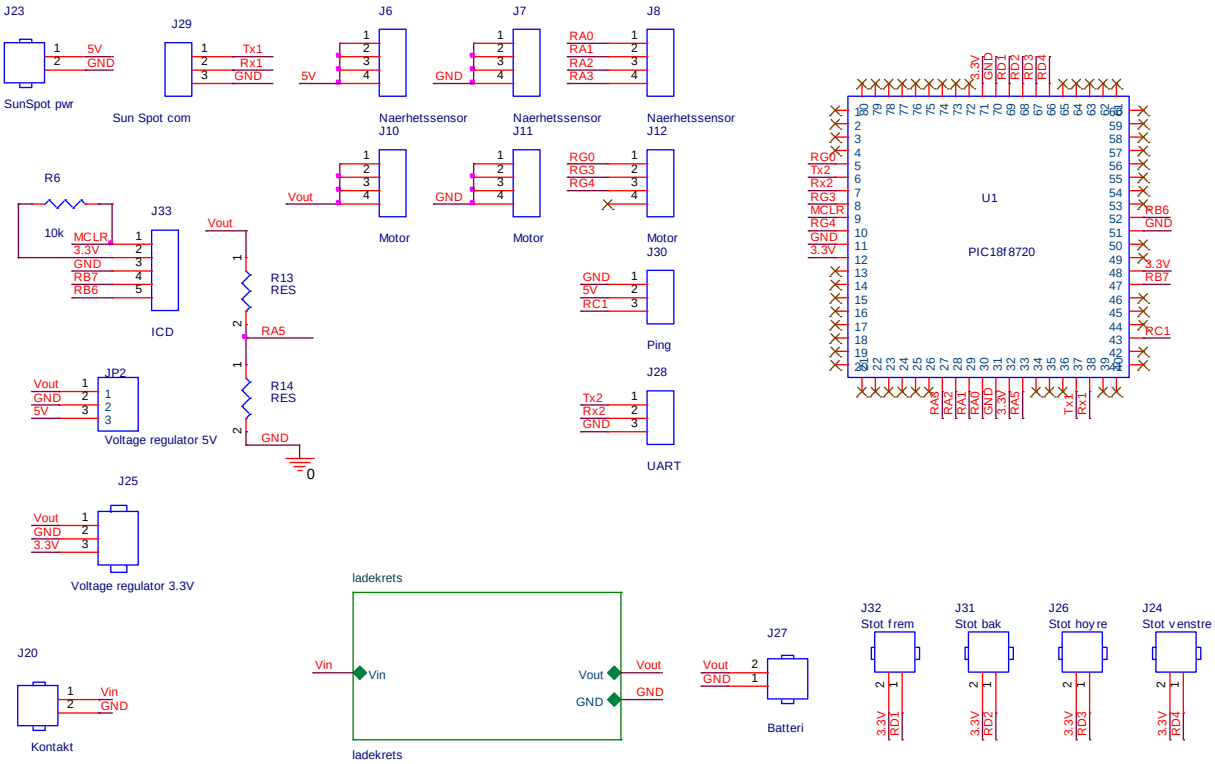
10.1.7 BLYAKKUMULATOR

Vi bruker en Panasonic 6V 4.5Ah blyakkumulator. Dette vil være tilstrekkelig for å kunne bruke roboten, da hver av servoene trekker maksimalt 140mA hver. Den maksimale strømmen batteriet kan lades med før man skader batteriet er 1.8A. Når batteriet er fullt kan se av karakteristikken til batteriet at man har god tid før batteriet er tømt, selv om roboten skulle trekke maksimalt med strøm.

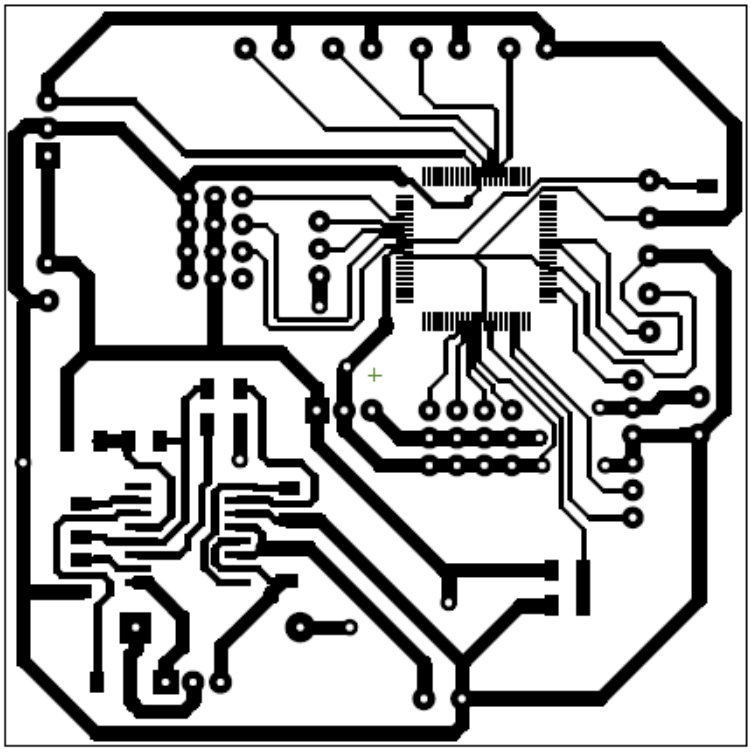


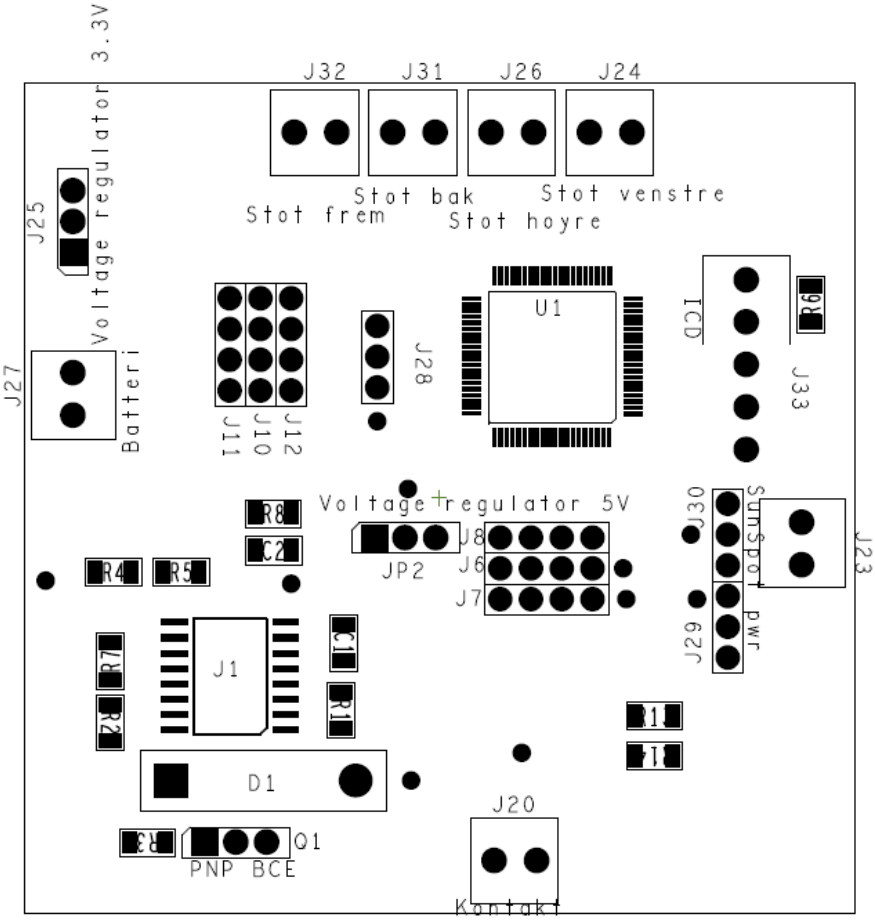
10.1.8 ORCAD DESIGNER OG FERDIG KRETSKORT

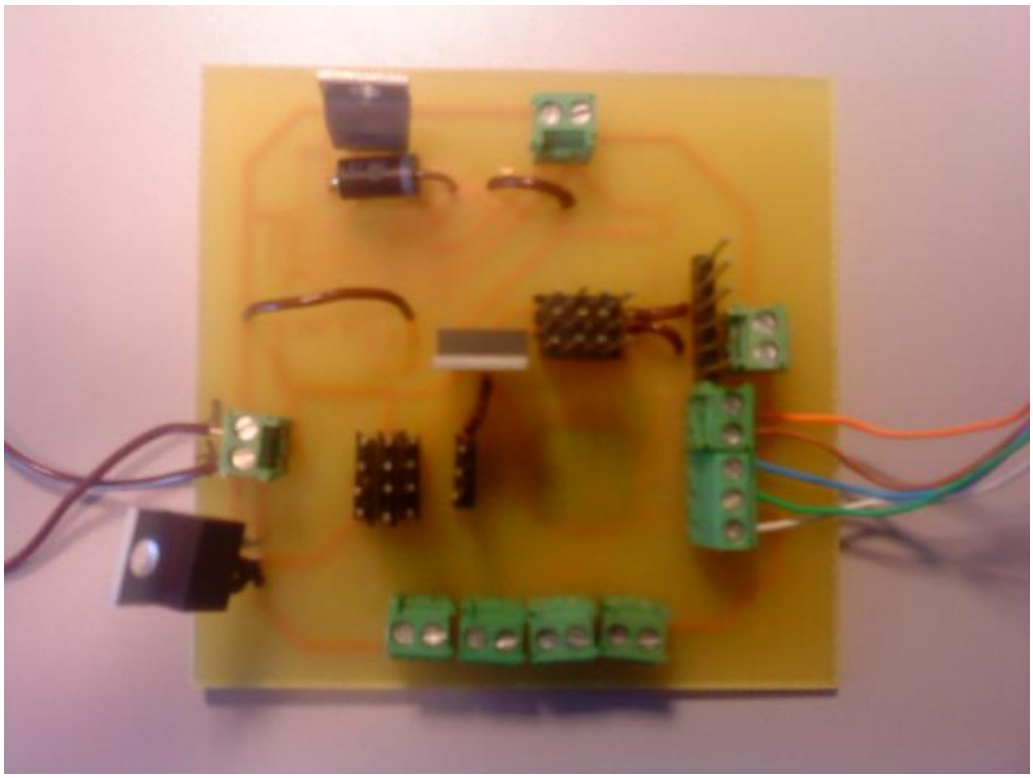
Skjemategning i Orcad:

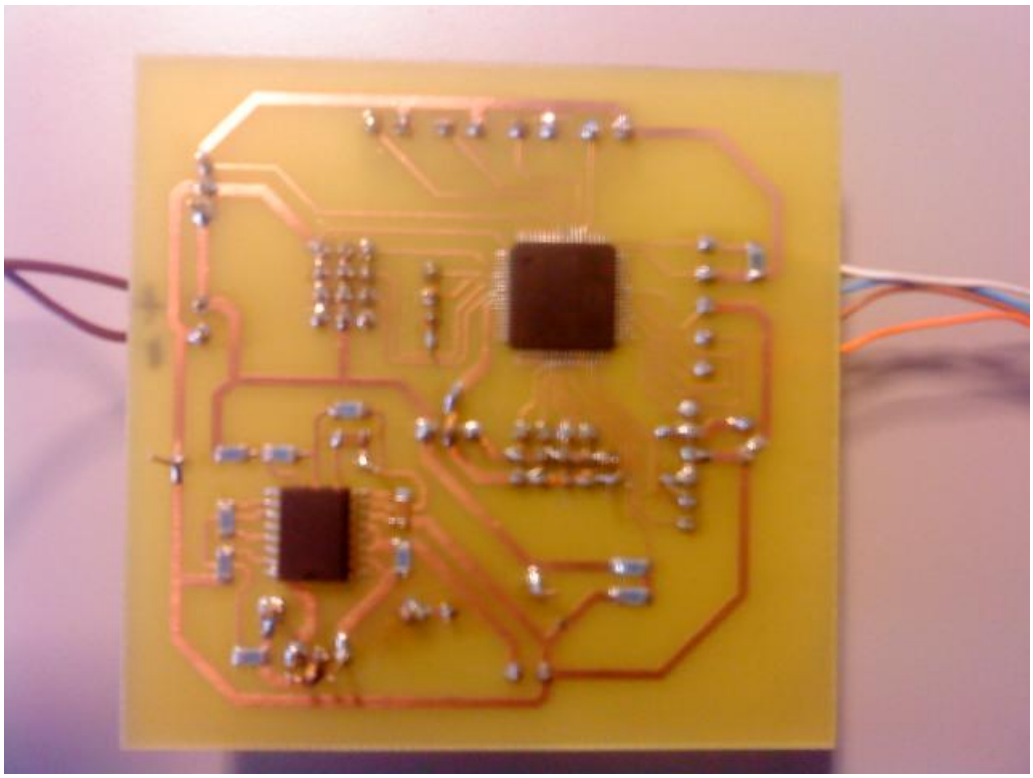


PCB design:




J32
J31
J26
J24
Stot frem
Stot bak
Stot hoyre
Stot venstre
J25
Voltage regulator 3.3V
J27
Batteri
J11
J10
J12
J28
U1
ICD
J33
R6
Voltage regulator 5V
R8
C2
JP2
J8
J6
J7
J30
Sunspot pwr
J29
J23
R4
R5
J1
C1
R1
R7
R2
D1
J20
Kontakt
Q1
PNP BCE





10.2 PROGRAMMERING AV MIKROKONTROLLER

10.2.1 MIKROKONTROLLER

Type: PIC18F8720
Pinner: 80
ADC kanaler: 16
CCP pinner: 5
UART : 2

Denne mikrokontrolleren ble valgt for det var denne som var tilgjengelig på skolen og hadde alle funksjonene vi trenger:

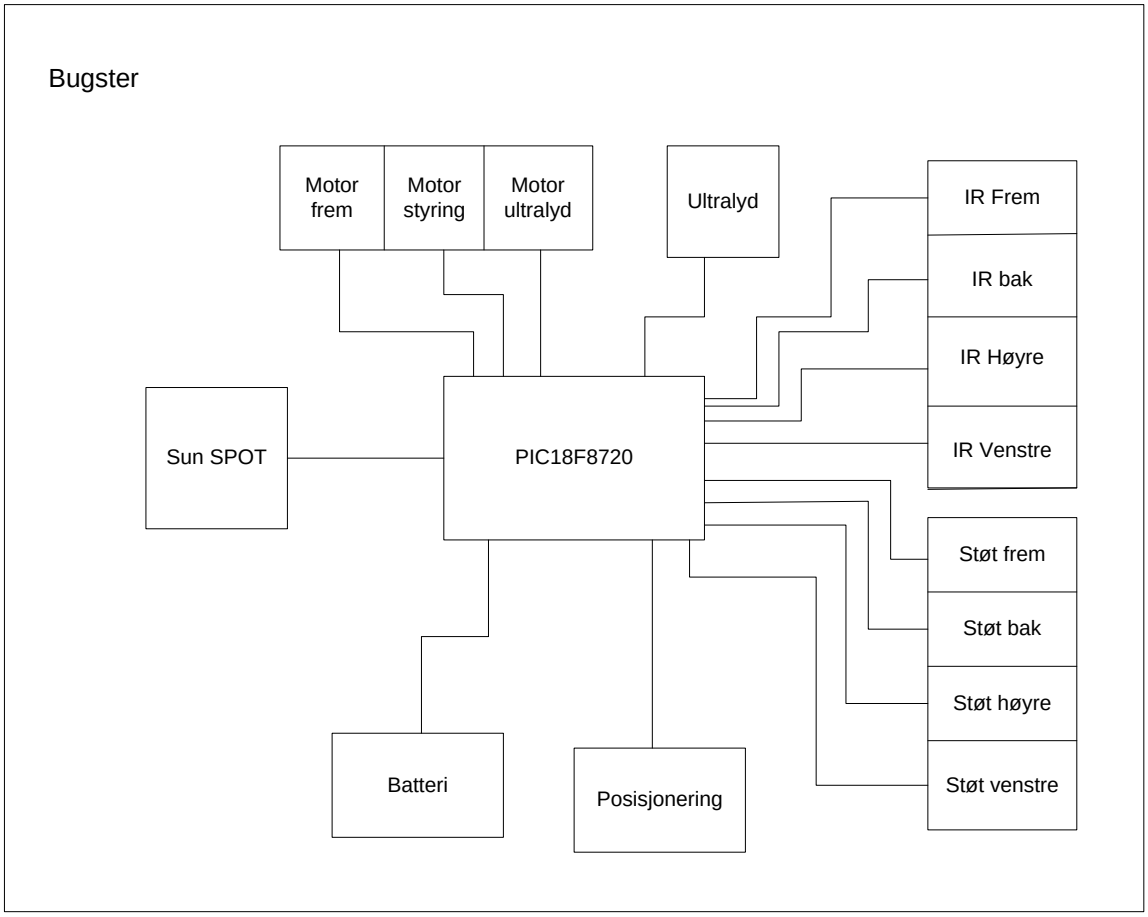
1. Fire CCP (Capture, Compare og PWM) pinner
    a. Tre til styring av motor
    b. En til ultralydsensoren
2. Fem ADC kanaler
    a. Fire til IR sensorene
    b. En til overvåking av batteriets spenningsnivå
3. Funksjoner for seriekommunikasjon
    a. En for kommunikasjon med Sun SPOT
    b. Kommunikasjon mot andre systemer (feks; posisjoneringssystem)
4. Minst fire I/O pinner
    a. For overvåking av støtsensorer

Til dette prosjektet hadde en mikrokontroller med færre pinner vært et bedre valg. Vi bruker kun et fåtall av de tilgjengelige pinnene og en større mikrokontroller med færre pinner ville vært enklere å lodde på kretskortet.

10.2.2 ARBEIDSOPPGAVER

Sun SPOT enheten som skal styre roboten har ikke nok I/O pinner til alle motorer og sensorer. Det er derfor nødvendig med en mikrokontroller til å styre motorene og ta inn informasjon fra sensorene.

Mikrokontrolleren skal utføre kommandoer gitt av Sun SPOT enheten og sende informasjon fra sensorer.



På figuren vises hvilke komponenter mikrokontrolleren skal kobles til. Under er sensorene og motorene som mikrokontrolleren skal styre listet opp

Sensorer:

1. Ultralyd sensor
2. IR sensor
3. Støtsensor
4. Batterisensor

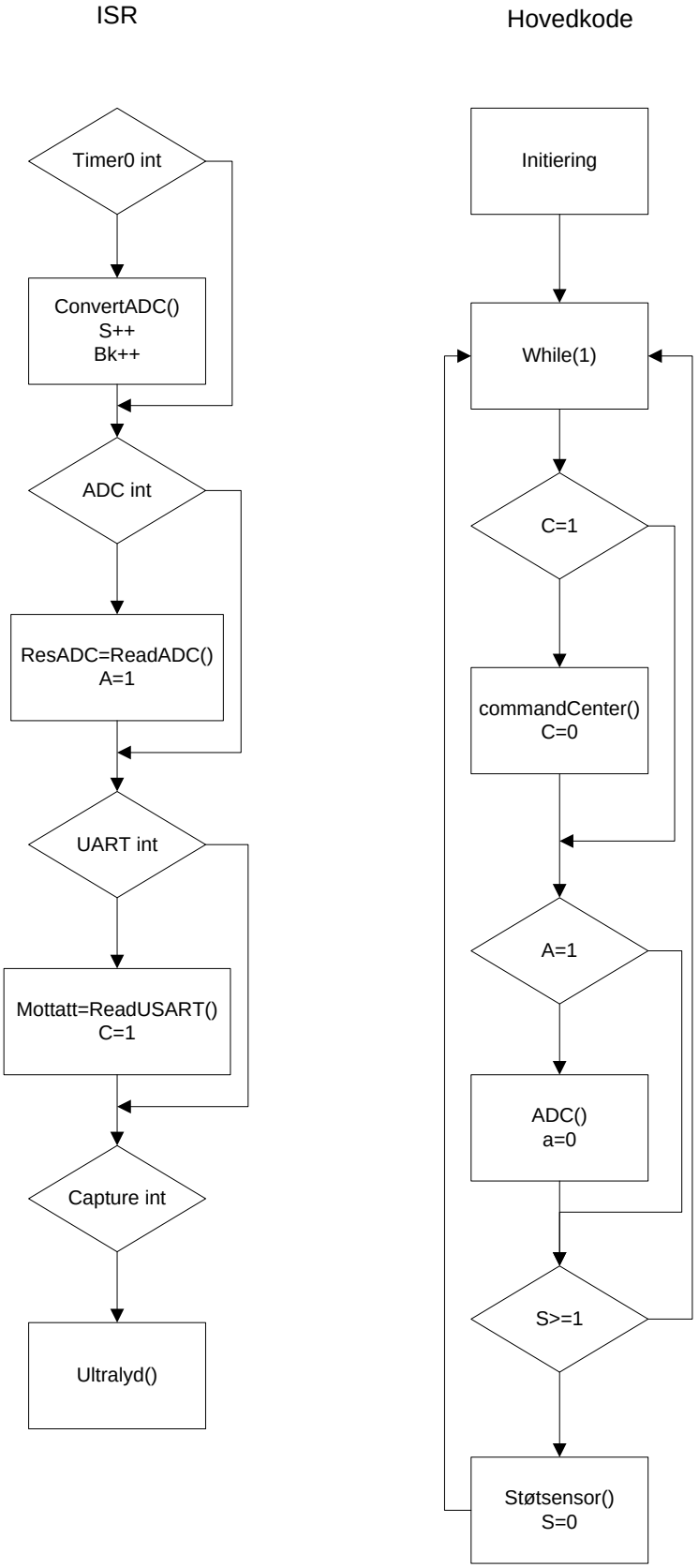
Servoer:

1. Servo for fremdrift
2. Servo for styring
3. Servo for retning på ultralyd

I tillegg er roboten koblet til en Sun SPOT og et posisjoneringssystem. Begge disse skal kommunisere med mikrokontrolleren med UART. Denne kommunikasjonen blir beskrevet nærmere i egne underkapitler.

10.2.3 PROGRAMSTRUKTUR

Under vises et flytdiagram av hovedstrukturen til programmet, detaljerte beskrivelser av hver enkelt del kommer senere i dokumentet.



Koden vil ha to deler; hovedkoden og en interrupt service routine (ISR). Hovedkoden vil gå i en løkke og sjekke hvilke oppgaver den skal utføre. Når en funksjon gir et interrupt vil programmet hoppe fra hovedkoden til ISR metoden. ISR metoden utfører de oppgavene som hører til interruptet, f.eks starter en AD konvertering. Den gir også beskjed til hovedkoden hvilke oppgaver som skal utføres ved å sette tilhørende kontrollvariabel høy.

Kontrollvariabler:

a: styrer når ADC() skal kjøres
c: styrer når commandCenter skal kjøres
s: styrer når støtsensorene skal sjekkes
bk: styrer når spenningsnivå over batteri skal sjekkes

Det er fire funksjoner som gir interrupt:

1. Timer0
    a. Gir interrupt når Timer0 går fra 65535-0
    b. Styrer når AD konverteringen skal starte
    c. Styrer når støtsensorene skal sjekkes
2. AD konverter
    a. Gir interrupt når en AD konvertering er ferdig
    b. Henter resultat fra konvertering
    c. Gir bedskjed til hovedkoden at den kan starte ADC() metoden
3. UART
    a. Gir interrupt når ny byte er mottatt
    b. Henter ut informasjonen
    c. Gir bedskjed til hovedkode om å starte commandCenter()
4. Capture
    a. Gir interrupt ved bedskjed fra ultralydsensor
    b. Starter ultralyd()

10.2.3.1 INITIALISERING

Her opprettes og konfigureres de nødvendige hardware funksjonen som Timer0 og AD konverteren. I tillegg opprettes og settes variabler. Mikrokontrolleren blir konfigurert:

1. Laster inn nødvendige biblioteker
2. Skrur av WDT(Watch Dog Timer) og LVP(Low Voltage Programming)
3. Setter intern oscillator frekvens (Fosc) til å være 4 MHz
4. Setter retningen på I/O portene som skal brukes

10.2.4 MOTORER

Roboten har tre motorer. En skal styre robotens fremdrift, de to andre er servoer som skal styre roboten og retningen til ultralydsensoren.

Fremdrift:

Type:	Parallax continuous rotation servo
Hastighet:	50rpm
Spenning:	4-6 V
Vekt:	42 g
Annet:	Bidirectional

Styring:

Type:	Parallax standard servo
Retning:	0-180
Spenning:	4-6 V
Vekt:	44 g

Disse motorene blir brukt fordi de er enkle å bruke, hastigheten er god nok og de var tilgjengelig på skolen.

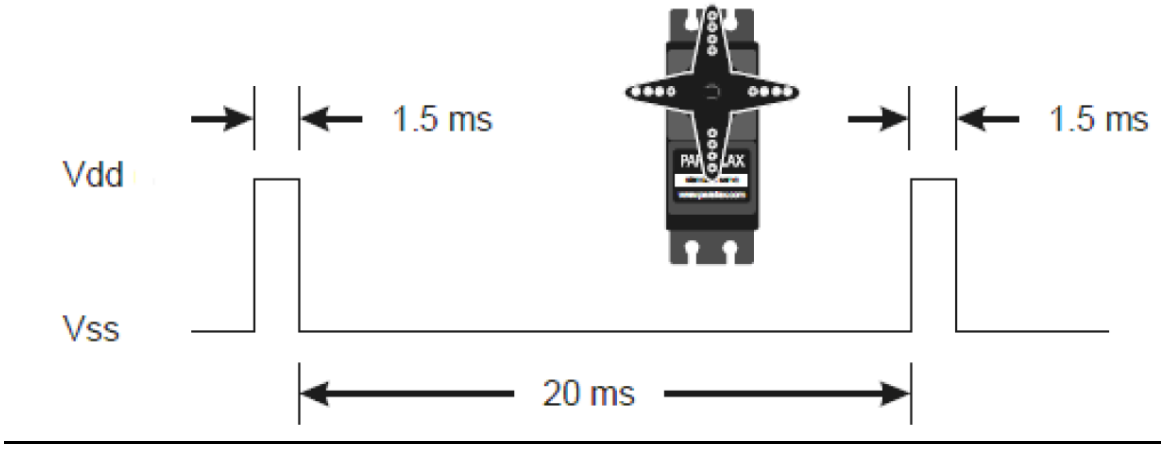
10.2.4.1 VIRKEMÅTE

10.2.4.1.1 CONTINUOUS ROTATION SERVO

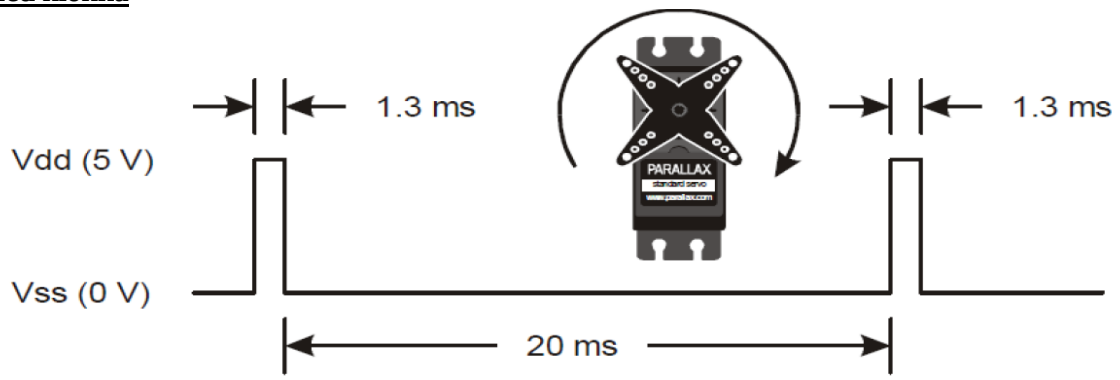
Servoen styres med PWM (pulsbreddemodulering). Den har en lineær respons til PWM, det betyr at hastigheten til motoren forandrer seg proporsjonalt med dutycyclen til PWM pulsen.

Motoren er bidirectional så pulsbredden bestemmer retning i tillegg til fart. For optimal drift skal motoren ha en pause på 20 ms mellom pulser.

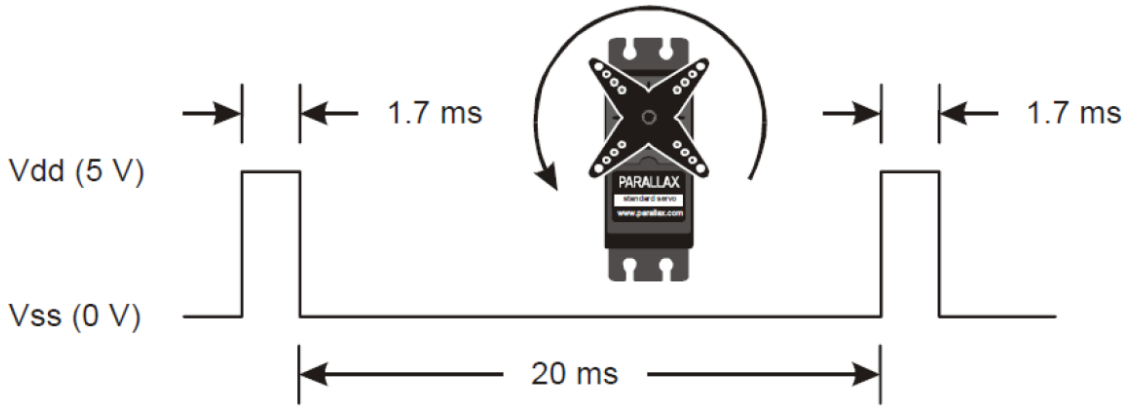
Stopp



Med klokka



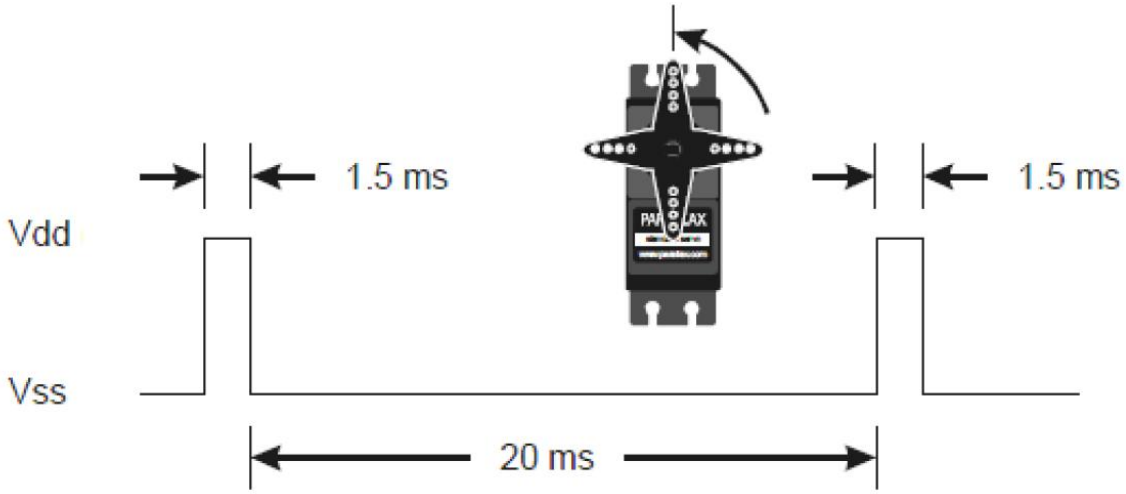
Mot klokka



Pulser på 1,3-1,5 ms fører til rotasjon med klokka. Pulser på 1,5 - 1,7 ms fører til rotasjon mot klokka. En puls på 1,5 ms stopper motoren(hvis den er riktig kalibrert). Motorene kan kalibreres ved å stille på et potensiometer i motoren.

10.2.4.1.2 STANDARD SERVO

Standard servo fungerer på samme som continuous rotation servo. I dette tilfellet styrer lengden på pulsen retningen til servoen. Denne skal også ha en pause mellom pulser på 20ms for optimal drift.



10.2.4.2 TILKOBLING

Servoene kobles til pinner med CCP funksjon slik at de kan styres med PWM:

Servo	Pinne
Fremdrift	RG0/CCP3
Styring	RG3/CCP4
Ultralyd	RG4/CCP5

10.2.4.3 PROGRAMMERING

Mikrokontrolleren har hardware funksjoner for PWM, dette betyr at mikrokontrolleren bruker mye mindre krefter på å kjøre motorene i forhold til om man skulle styrt motorene med software. En ulempe med PWM funksjonen er den klarer ikke å lage en periode på 20 ms. Det er derfor ikke mulig å styre motorene optimalt.

Vi velger å bruke en periode på 4 ms. Dette er den største mulige perioden når frekvensen til den interne oscillatoren (Fosc) er på 4 MHz. Beregner perioden med likningen:

$$T_{PWM} = (PR2 + 1) * 4 * N * T_{OSC}$$

Hvor PR2 er et 8 bits register som styrer perioden, N er prescaleren til Timer2 og Tosc er perioden til den interne oscilatoren.

For å få en periode på 4 ms når Fosc er 4MHz(Tosc=0.25uS) må PR2 settes til 255 og N være 16.

PWM funksjonen på mikrokontrolleren er avhengig av Timer2. Dette er en 8 bits teller (0-255) som inkrementeres med en frekvens på Fosc/4. I tillegg har den en prescaler som gjør at den kan telle tregere. Denne timeren styrer perioden til PWM funksjonen sammen med PR2 registeret. Timer2 startes med denne kommandoen:

OpenTimer2(TIMER_INT_OFF & T2_PS_1_16 & T2_POST_1_1)

Ingen interrupt, prescaler på 16 for oppnå ønsket periode og ingen postscaler.

For å starte PWM funksjonen brukes denne kommandoen:

OpenPWMx(PR2); (x bestemmer hvilken CCP pinne som skal brukes)

PR2 settes til 255 for å oppnå en periode på 4 ms.

Hastighet og retning på motoren bestemmes av Tdtc. Det er tiden signalet til motoren er høy i hver periode. For å sette dutycyclen brukes denne kommandoen:

SetPWMx($CCPxL : CCPxCON\,5{,}4$)

Hvor $CCPxL : CCPxCON\,5{,}4$ (dtc)er et tall mellom 0-1023. For å bestemme dens verdi brukes likningen:

$$T_{dtc} = (CCPxL : CCPxCON\,5{,}4)T_{OSC} * N$$

Ut i fra denne likningen kan vi finne verdien som skal legges inn. Tabellen under viser hvilke verdier vi bruker for motorene.

Motor Fremdrift			Motor styring/ultralyd		
	dtc	**Tdtc** (ms)		**dtc**	**Tdtc** (ms)
Frem	325	1,3	**45**	400	1,6
Rygg	500	2	**90**	300	1,2
Stopp	0	0	**135**	200	0,8

Disse verdiene er funnet ved tester. For å stoppe motoren setter vi dutycyclen til 0. Dette gjøres pga. tester har vist at motoren har stått stille i en liten stund før den begynner å rotere ved pulser på 1,5ms. Motoren vil heller ikke trekke like mye strøm når vi stenger den helt ned.

10.2.5 IR SENSOR

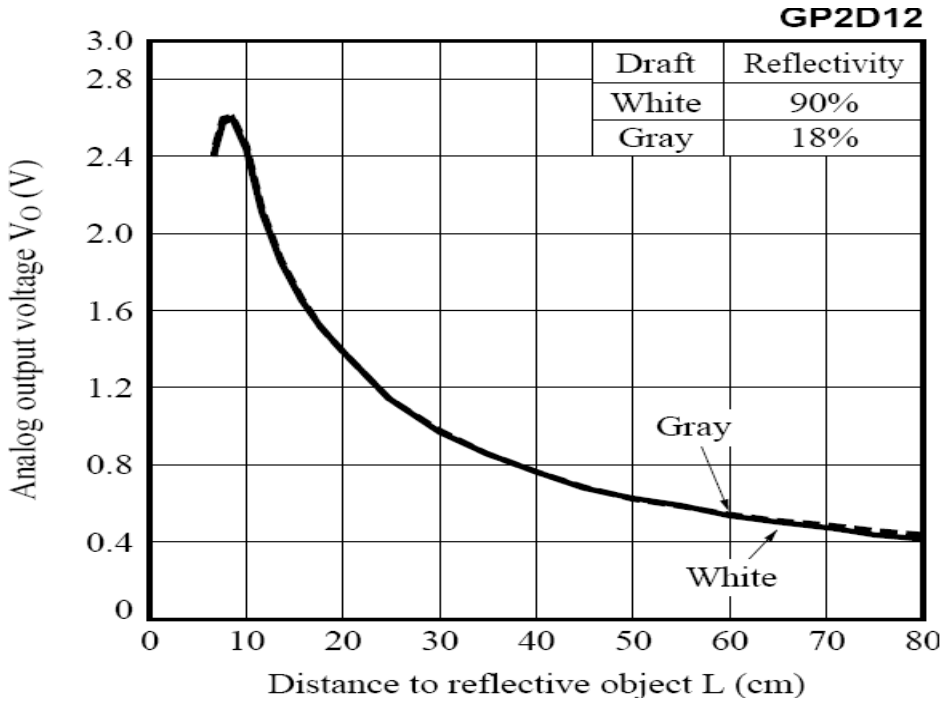
Roboten skal ha fire nærhetssensorer av typen SHARP GP2D12

Type: SHARP G2PD12
Spenning: 5 V
Rekkevidde: 10-80 cm

Disse sensorene plasseres på hver side av roboten, de skal detektere objekter i nærheten av roboten.

10.2.5.1 VIRKEMÅTE

Sensoren gir ut en spenning på signal pinnen. Denne spenningen varierer i forhold til avstanden fra sensor til objekt.



Figuren over viser karakteristikken til sensoren.

10.2.5.2 TILKOBLING

Sensoren kobles til en pinne på mikrokontrolleren som har AD konverter.

Sensor	Pinne
Frem	RA0/AN0
Bak	RA1/AN1
Høyre	RA2/AN2
Venstre	RA3/AN3

10.2.5.3 PROGRAMMERING

AD konverteren på mikrokontrolleren er en 10 bits konverter. Spenningen på inngangen blir gjort om til et tall mellom 0-1023. Med bruk av en referanse spenning på 3,3V får vi en step size på 3.22 mV

$$step\_size = \frac{V_{Ref}}{1024}$$

Det kan kun gjøres en AD konvertering av gangen. Siden vi har fire sensorer bytter vi mellom kanalene.

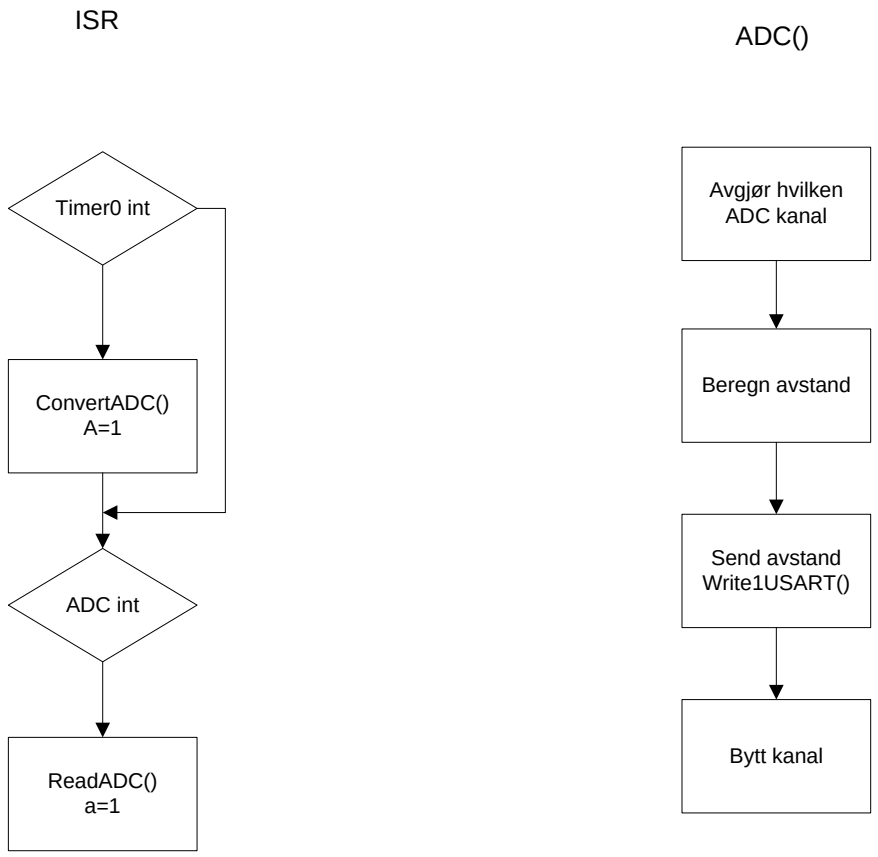
Bruker Timer0 til å styre når konverteringen skal begynne. Timer0 blir satt opp med denne kommandoen:

OpenTimer0(TIMER_INT_ON & T0_16BIT & T0_SOURCE_INT & T0_PS_1_4)

Dette gir en 16 bits teller (0-65535) med en prescaler på 4. Timer0 vil da gi et interrupt hvert 0.26 sekund. Det gir ca. fire AD konverteringer i sekundet.

AD konverteren blir satt opp ved å bruke denne kommandoen:

OpenADC(ADC_FOSC_8 & ADC_RIGHT_JUST & ADC_6ANA & ADC_CH0 & ADC_INT_ON & ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS);



Figuren over viser et flytskjema for AD konverteringen.

Konverteringen starter når Timer0 gir et interrupt. Bruker funksjonen

ConvertADC()

for å starte en AD konvertering

Når AD konverteringen er klar gis det et interrupt, da hentes resultatet:

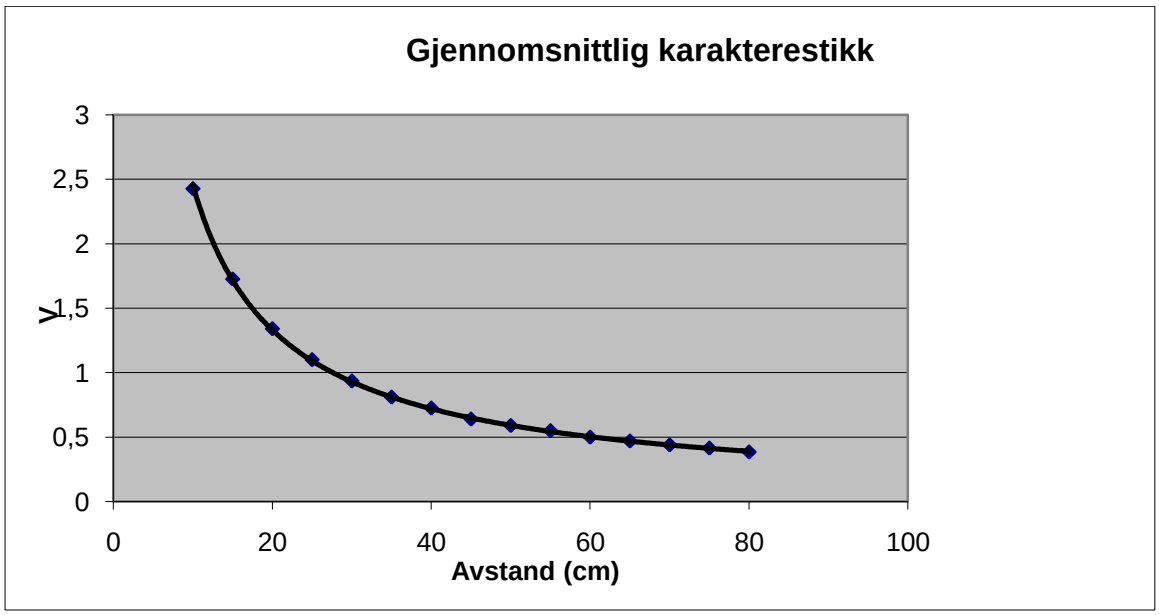
ReadADC()

og ADC kontrollvariabel (a) settes til 1. Når hovedprogrammet merker at a =1 starter den ADC() metoden. I denne metoden beregnes avstanden og sendes til Sun SPOT enheten. Til slutt byttes det til neste ADC kanal med funksjonen:

SetChanADC(hvilken kanal)

10.2.5.4 KONVERTERING; SPENNING-CM

Etter tester har vi funnet karakteristikken til IR sensorene.



Funksjonen som best beskriver denne karakteristikken er:

$$U = 18.88x^{-0,89}$$

Hvor U er spenningen og x er avstanden. AD konverteren tar spenningen U og gjør den om til en verdi mellom 0-1023.

$$B = \frac{1023}{V_{\mathrm{Re}\,f}}U$$

Ut i fra disse to likningene får vi denne funksjonen som gir oss avstanden i cm ut ifra hvilken verdi vi får fra AD konverteren.

$$x = 11265,92B^{-1.13}$$

Bruker funksjonen pow(B, -1,13) for å regne det ut. Denne funksjonen krever endel datakraft og tar endel tid å utføre.

Som beskrevet i avsnittet om kommunikasjonsprotokollen vil vi dele opp avstandene med et intervall på 10 cm (f.eks; 10 cm-20 cm) og gi hvert intervall en egen ID.

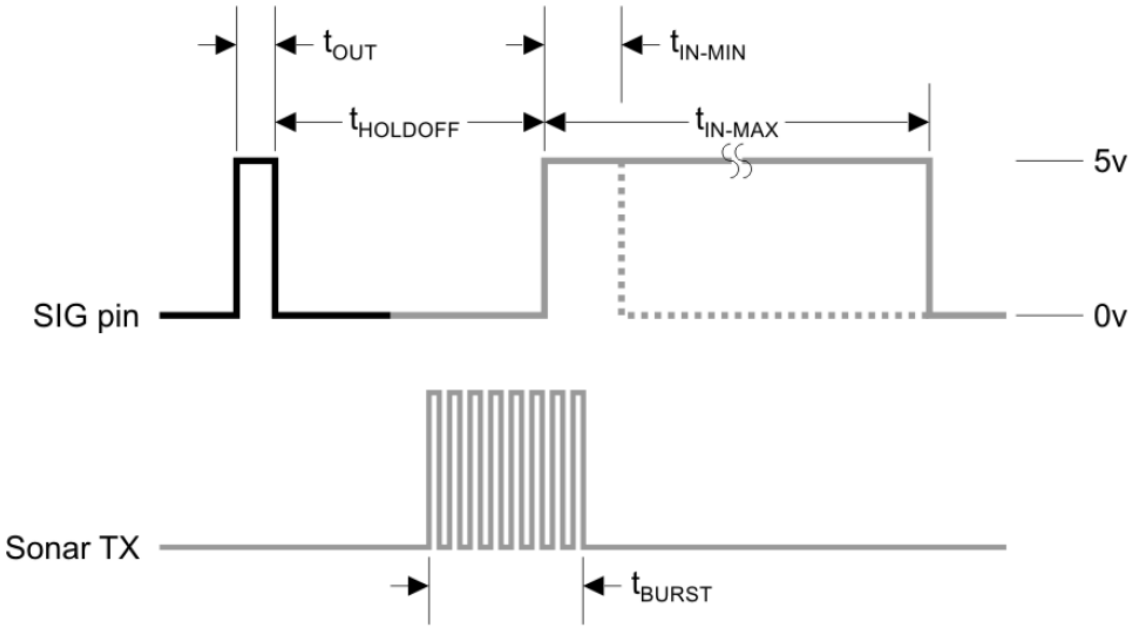
10.2.6 ULTRALYDSENSOR

Roboten skal ha en ultralydsensor montert på en servo for å kunne skanne terrenget i bestemte retninger.

Type: Parallax PING
Spenning: 5 V
Rekkevidde: 2 cm - 3 m

10.2.6.1 VIRKEMÅTE

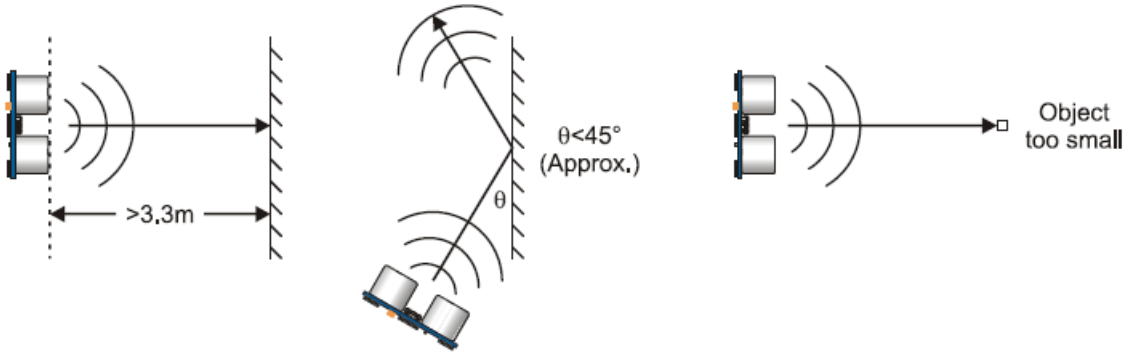
Parallax PING sender ut en høyfrekvent puls for å måle avstand. Sensoren styres fra mikrokontrolleren. For å sende pulsen setter brukeren først signalpinnen høy i 2 - 5 us. Ultralydsensoren sender da ut en høyfrekvent puls. Når pulsen er sendt setter ultralydsensoren signalpinnen høy. Når pulsen har returnert setter sensoren signalpinnen lav.



	Host Device	Input Trigger Pulse	t_{OUT}	2 µs (min), 5 µs typical
	PING))) Sensor	Echo Holdoff	$t_{HOLDOFF}$	750 µs
		Burst Frequency	t_{BURST}	200 µs @ 40 kHz
		Echo Return Pulse Minimum	t_{IN-MIN}	115 µs
		Echo Return Pulse Maximum	t_{IN-MAX}	18.5 ms
		Delay before next measurement		200 µs

For at sensoren skal kunne måle avstanden til et objekt er det noen krav som må være oppfylt.

1. Objektets overflate må kunne reflektere lyd
2. Avstanden til objektet må være innenfor rekkevidden til sensoren
3. Vinkelen til objektet kan ikke være så skarp at pulsen ikke reflekteres tilbake
4. Objektet kan ikke være for lite, da vil ikke pulsen reflekteres.

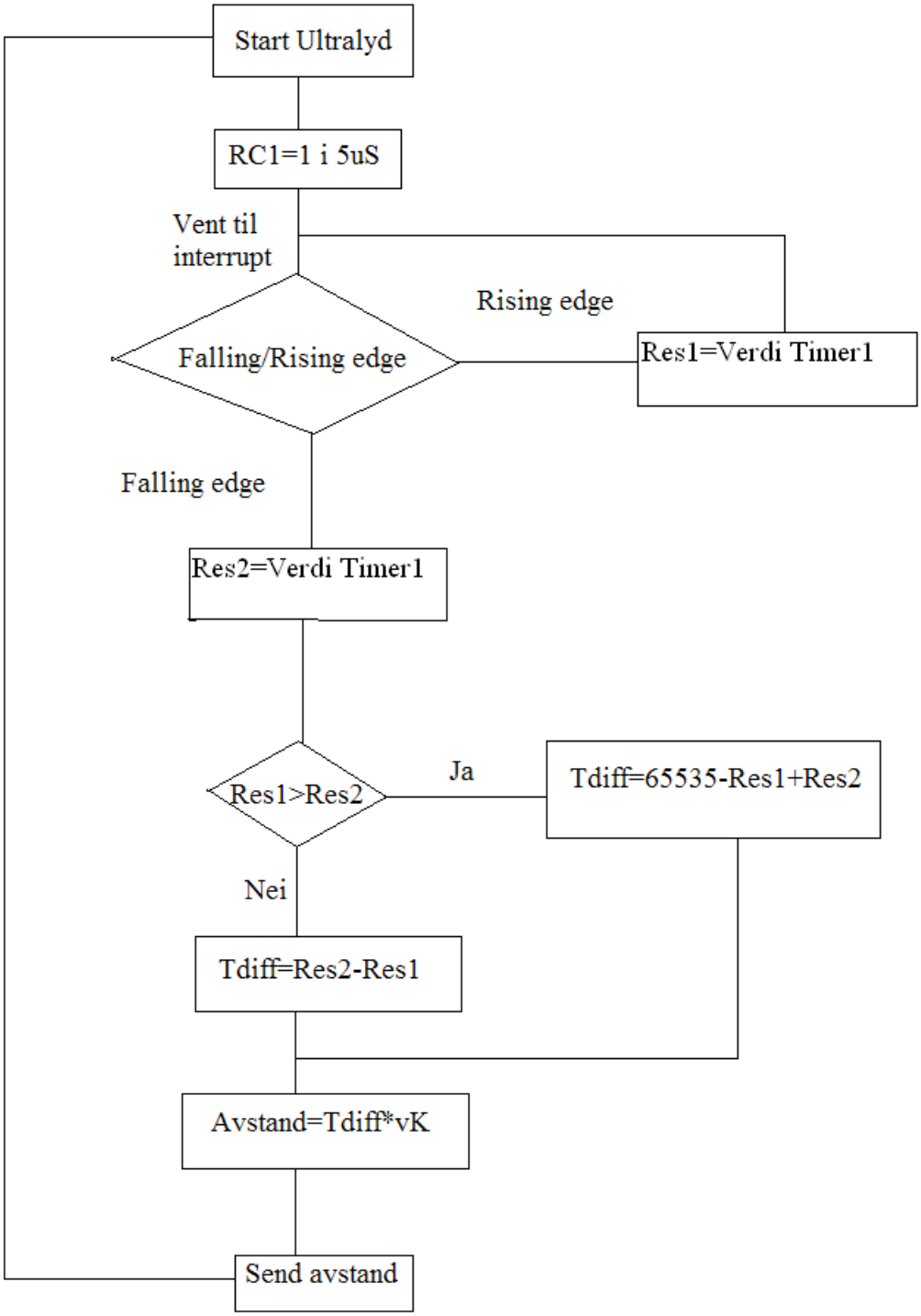


10.2.6.2 TILKOBLING

Må bruke en pinne med CCP (Capture Compare PWM). Vi bruker RC1 (CCP2).

10.2.6.3 PROGRAMMERING

Mikrokontrolleren har tre oppgaver; starte måling, overvåke RC1 og beregne avstanden.



For å starte målingen sendes en puls på ca 5 us fra mikrokontroller så settes RC1 til inngang. Til å overvåke RC1 bruker vi en capture funksjon.

OpenCapture2(CAPTURE_INT_OFF & C2_EVERY_RISE_EDGE);

Capture funksjonen overvåker RC1, hvis den oppdager en forandring i spenning (falling edge eller rising edge) lagrer den verdien til Timer1. Når vi vet hvor fort Timer1 inkrementerer kan vi finne tiden pulsen bruker fra den er sendt ut til den returnerer.

Etter mikrokontrolleren har startet målingen setter den capture funksjonen til å gi interrupt når spenningen på RC1 går fra 0-5V(rising edge). Verdien fra Timer1 blir lagret:

ReadCapture2();

Capture funksjonen settes til å gi interrupt når spenningen på RC1 går fra 5-0V (falling edge). Ved interrupt lagres verdien fra Timer1 og avstanden beregnes.

10.2.6.3.1 BEREGNING AV AVSTAND

Når Fosc=4MHz inkrementeres Timer1 hvert 1us. Differansen mellom Res1 og Res2 vil være antall ganger Timer1 har inkrementert fra pulsen ble sendt til den returnerte. Timer1 teller fra 0-65535, det kan skje at den ruller over fra 65535-0. Da vil Res1 være større Res2 og differansen må beregnes som:

Tdiff=65535-Res1+Res2

Det er ingen fare for at Timer1 skal telle en hel runde mens avstanden måles da den bruker ca 65,5 ms på det og maks tiden til avstandsmålingen er 18,5 ms.

Avstanden beregnes med formelen:

$$s = \frac{c * t}{2}$$

Hvor c er hastigheten til lyden og t er tiden brukt. Da Timer1 inkrementeres hvert 1us vil t=tDiff us. Hastigheten til lyden ved romtemperatur er ca 344m/s. Får da:

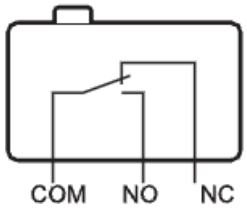
$$s = tDiff * 0.0172_{(\mathrm{cm})}$$

10.2.7 STØTSENSOR

Roboten skal ha 4 støtsensorer, en på hver side av roboten. Vi bruker en mikroswitch som støtsensor

10.2.7.1 VIRKEMÅTE

Kobler COM pinnen og NO pinnen, når bryteren presses inn vil det bli kontakt mellom COM pinnen og NO pinnen.



10.2.7.2 TILKOBLING

Com pinnen kobles til 3,3 V og NO pinnen kobles til en I/O pinne på mikrokontrolleren.

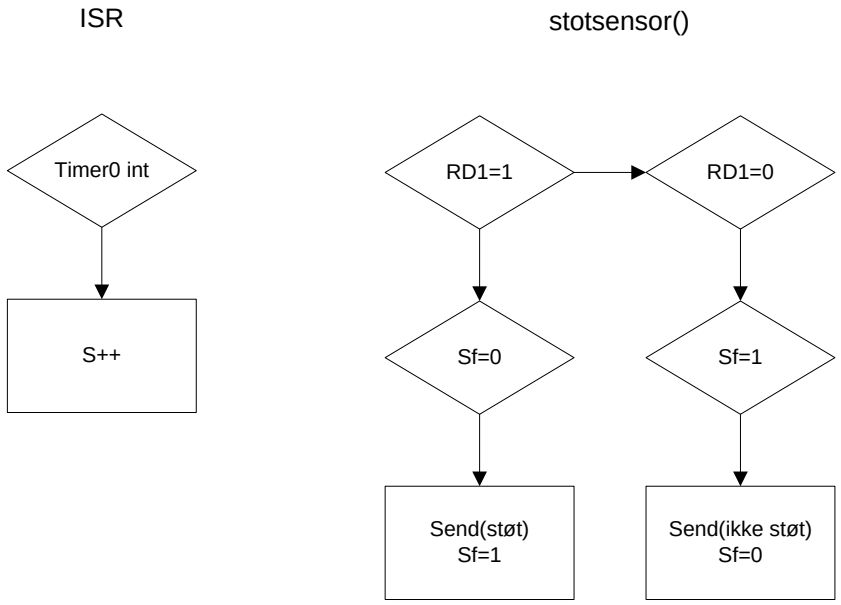
Objekt	I/O
Støt frem	RD1
Støt bak	RD2
Støt høyre	RD3
Støt venstre	RD4

10.2.7.3 PROGRAMMERING

Først må PORTD gjøres til en inngang (TRISD=1). For mikrokontrolleren PIC18F8720 brukes PORTD også som en kommunikasjonsbuss. Denne funksjonen skrus av ved å sette EDIS biten i MEMCON registret til 1.

Timer0 styrer når støtsensorene skal sjekkes, hver sensor blir sjekket ca 4 ganger i sekundet, hvert 0,26 sekund. Dette er sjeldent men vi unngår problemer som mekanisk bounce og falske støt signaler. Roboten beveger seg ikke fort nok til at en treg sjekk rate vil være et problem.

Når støtsensoren trykkes inn vil pinnen som er festet til den sensoren få 3,3V over seg. Programmet leser det som om pinnen er satt høy (RD1=1). Da vil programmet registrere et støt. Hvis ikke støt allerede er registrert vil den sende en beskjed til Sun SPOT enheten.



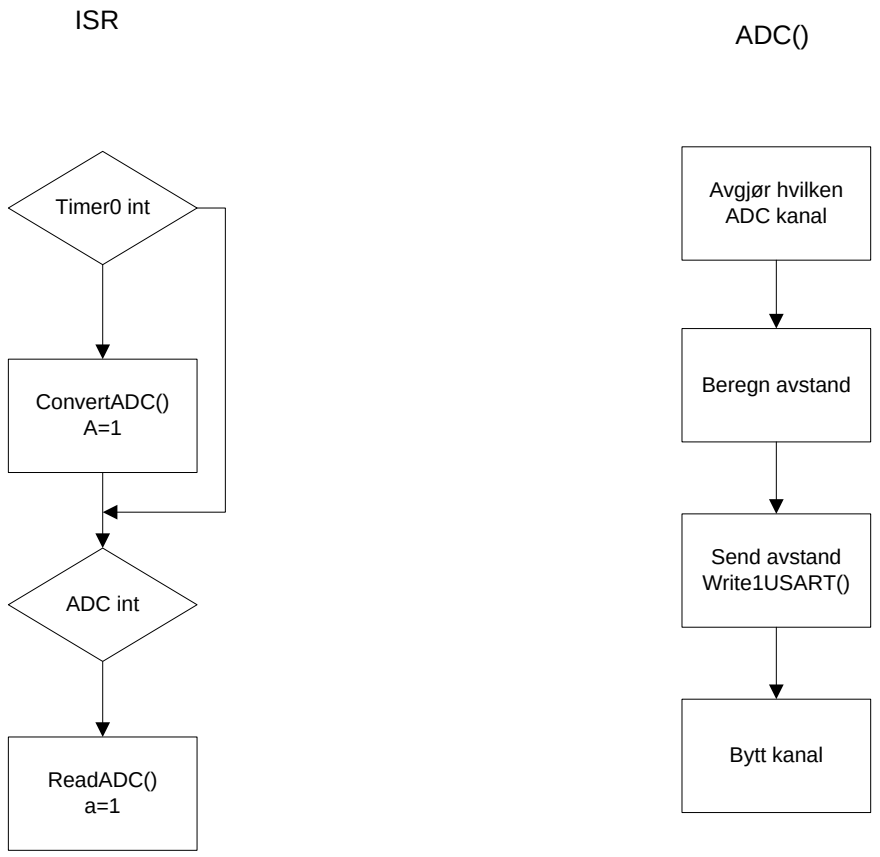
Figuren over viser flytdiagram for sjekk av fremre støtsensor.

Hver støtsensor har en status variabel (sf,sb,sh,sv), disse brukes til å lagre hvilken tilstand støtsensoren er i. De settes høy ved støt og lav ellers. Programmet skal kun sende til Sun SPOT enheten når status variablene endrer seg.

10.2.8 BATTERIOVERVÅKING

Mikrokontrolleren skal brukes til å overvåke spenningsnivået til batteriet. Batteriet kobles til en ADC kanal (AN4) på mikrokontrolleren. Siden batteriet har en spenning på 6.4V er det nødvendig å ha en spenningsdeler mellom batteri og mikrokontrolleren.

Timer0 styrer når spenningsnivået over batteriet skal måles. Spenningen over batteriet sjekkes ca hvert 30 sekund. Hvis nivået faller under 6V og holder seg der en tid vil roboten si ifra til Sun SPOT enheten at batteriet må lades.



Når bk>120 (gått 30 sekunder siden forrige batteri sjekk) settes ADC kanalen til AN4. Når AD konverteringen er fullført hentes resultatet og sjekkes. Hvis resultatet er under 930 (tilsvarer 6V over batteriet) vil det registreres (b++). Hvis b blir større enn 4 skal programmet si ifra at roboten må lades. Det betyr at spenningsnivået må holdt seg under nivået i minst 2 minutter før programmet sier ifra. Gjør dette for å være sikker på at nivået er lavt.

10.2.9 KOMMUNIKASJON

Mikrokontrolleren har hardware funksjon for UART kommunikasjon. Denne startes med kommandoen:

Open1USART(USART_TX_INT_OFF & USART_RX_INT_ON & USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX & USART_BRGH_HIGH, 25);

Denne oppretter en UART kommunikasjon med en baud rate på 9600. Den vil gi et interrupt når den har mottatt en byte.

Når en byte mottas fra Sun SPOT enheten hentes den ut ved kommandoen ReadUSART().
Når mikrokontrolleren skal sende en byte brukes kommandoen WriteUSART().

Sjekker for feil i kommunikasjonen ved å overvåke bitene FERR og OERR i RCSTA registeret. FERR biten settes høy når en framing error inntreffer. For å nullstille FERR biten tømmes mottaks registeret(RCREG).
OERR biten settes høy når en overrun error inntreffer. For å nullstille OERR biten må CREN biten settes til 0.

10.2.9.1 COMMAND CENTER

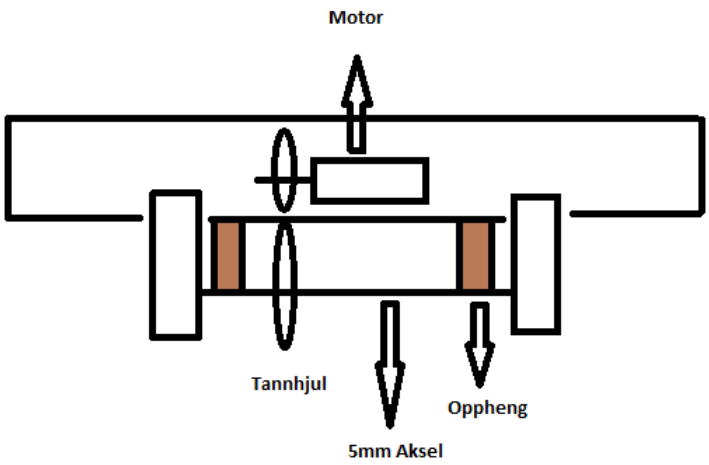
Metoden commandCenter() styrer mikrokontrolleren. Det er i denne metoden mikrokontrolleren tolker beskjeden fra Sun SPOT enheten og utfører oppgaven.

Metoden starter når mikrokontrolleren har mottatt en byte fra Sun SPOT enheten. Den deler den mottatte byten opp i adresse bits og kommandobits. Bruker switch kommandoer til å avgjøre hvilken motor/sensor adressen hører til og hvilken kommando som skal utføres.

10.3 MEKANISK

10.3.1 AKSEL

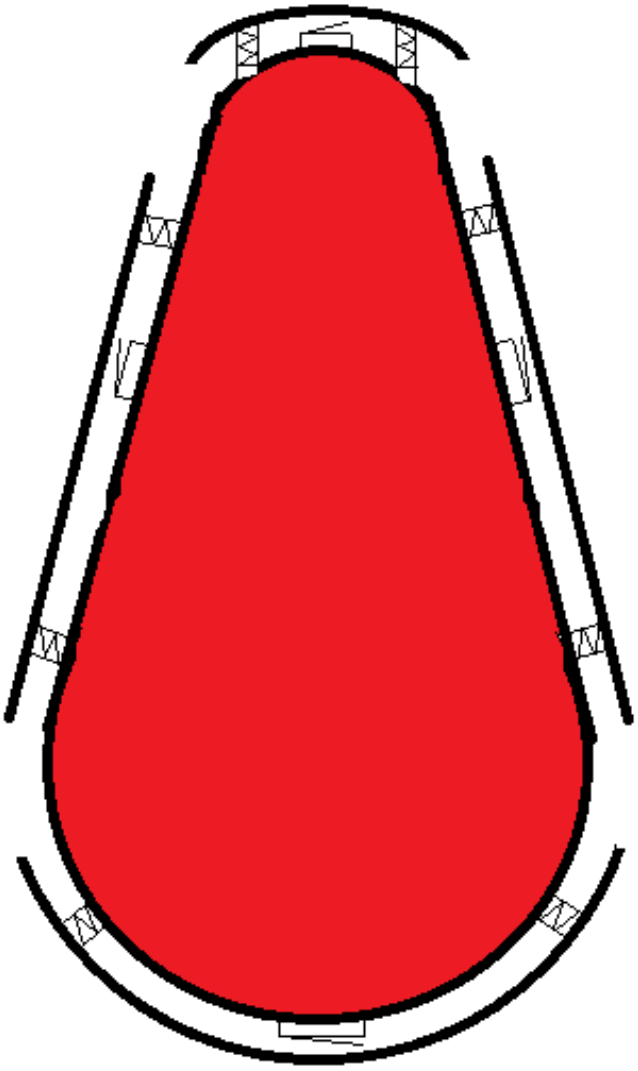
Akslingen er laget av 5 mm gjengestang. Men akselen er rund, og dette fører til at både hjul og spesielt tannhjul står og spinner på akslingen. Så vi har forbedret dette med å feste tannhjulet til akslingen med en settskrue. Dette vil føre til at tannhjulet ikke står og spinner på akslingen, og den vil være enkel å demontere der settskruen kan bare skrues ut. På hjulene har vi satt bolter fra hver side, slik at det blir holdt på plass når roboten kjører.



Fra forrige prosjekt var drivverket holdt på plass av to øyeskruer, dette førte til at det var veldig vanskelig å demontere drivverket bak når vi hadde satt inn settskrue på tannhjulet. Vi har bygd om og satt på to plastikk biter (oppheng) og borret hull i dem slik at man bare kan dytte drivverket opp i opphenget. Dette vil føre til at det blir letter å demontere drivverket, og man slipper å ta av hjulene for hver gang. Tegningen under viser en enkel beskrivelse av ideen.

10.3.2 STØTSENSOR

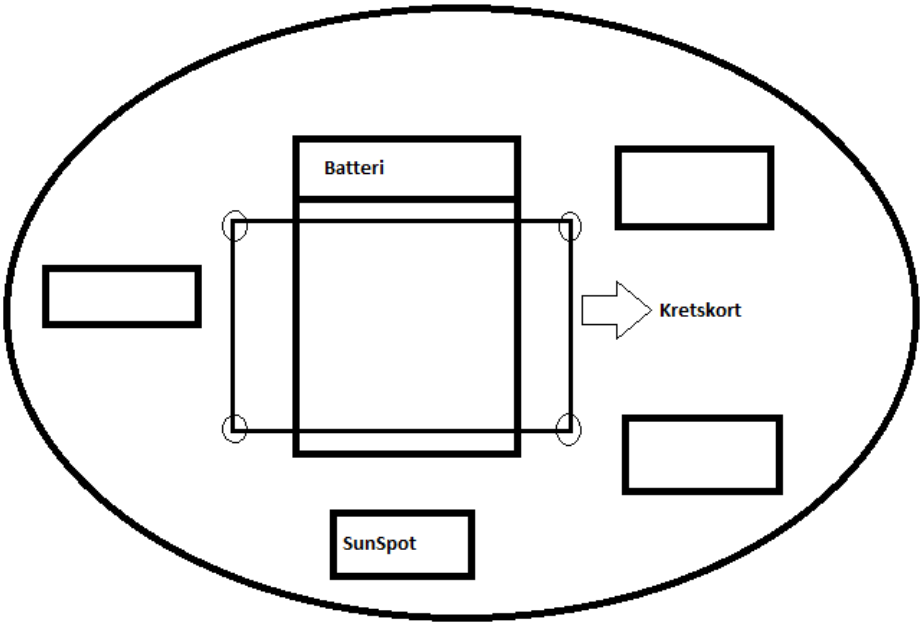
Roboten skal kunne registrere støt. Vi har derfor koblet 4 microswitchere rundt roboten som vil være koblet til mikrokontroller. Når switcheren kommer i kontakt med noe vil det bli sendt et elektrisk signal til mikrokontrolleren. For å minimere antall microswitchere lager vi et støtdempersystem slik at mircoswtichere merker støt fra alle vinkler. Vi har bygd en prototype av den fremste støtdemperen, og tester viser at mircoswitcher blir trygt inn uansett vinkel roboten treffer hindringen i. Den øverste kontakten på ladepunktene vil være festet til den fremste støtdemperen.



10.3.3 PLASSERING AV INNVENDIGE KOMPONENTER

Plasseringen av de innvendige komponentene på roboten er bestemt ut i fra hvor det er plass på bunnplata. Vi har valgt å legge batteriet på langs, mellom hjulet foran og hjulene bak, siden det er såpass stort. For å feste batteriet best mulig bruker vi patentbånd. Ved siden av batteriet får vi akkurat nok plass til å plassere Sun SPOT. Selve kretskortet vil vi plassere over batteriet, der vi kommer til å lage opphøyninger en på hver side av kretskortet og lime det på batteriet.

Motoren til drivverket sitter rett over akselen, og styreservoen er plassert rett over framhjulet. De innvendige komponentene vil bli feste til bunnplaten med skruer, for å få best feste..



11 LADESTASJON

Ideen for ladestasjonen er å legge en kobberplate langs veggen og en kobberplate ned på skrå fra den på veggen. Grunnen for at vi velger 3m plate som er hele veggen på newton rommet er for å treffe ladestasjonen enklest mulig. Ladepunktene på roboten vil være av elastiksk materialet som gir best mulig kontakt med ladestasjonen, siden roboten kan komme inn på skrå må den ha kontaktflate på sidene. Figuren under viser en enkel skisse av ideen.

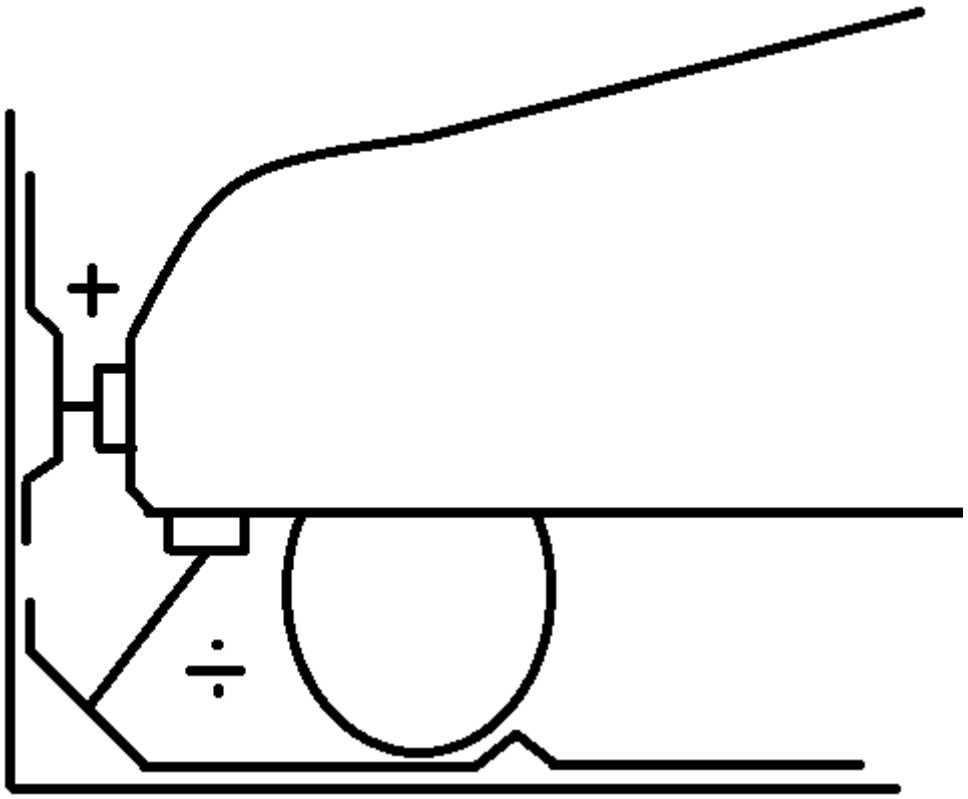
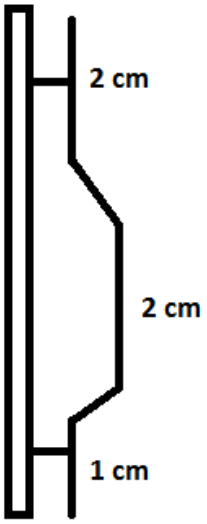


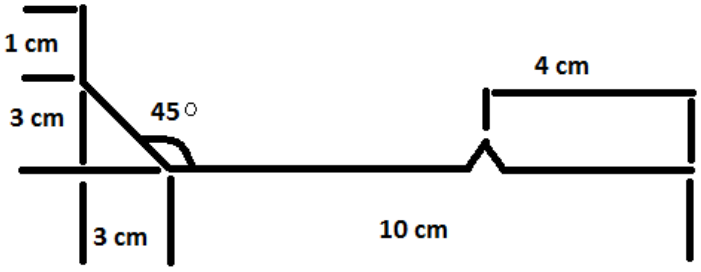
Plate 1



5 cm * 3 m

Kobberplate

Plate 2



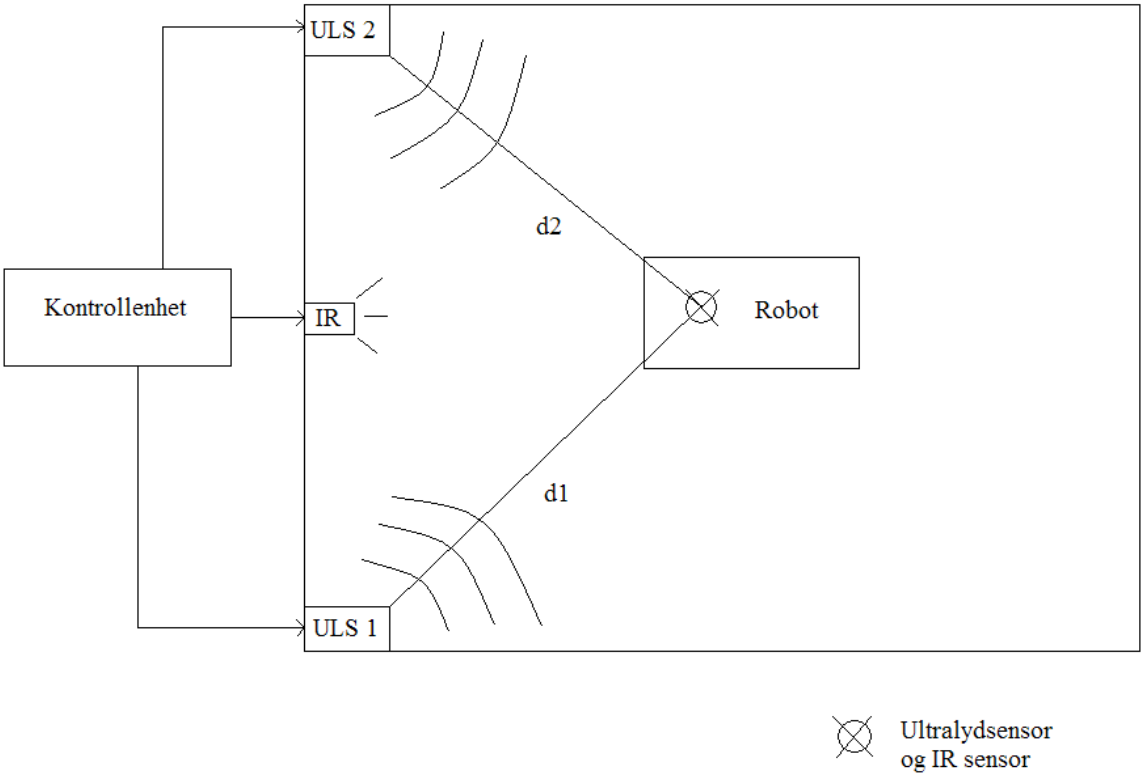
14 cm * 3 m Kobberplate

Av tegning 2 over, kan man se hvordan platene vil bli formet for å få den beste kontakten mellom ladestasjon og robot. For at ikke roboten skal bevege seg mens den lades, vil det bli en liten bøy på plate 2. Platene vil få strøm fra en spenningsregulator som gir ut 12V.

12 POSISJONERINGSSYSTEM

12.1 KONSEPT

Ideen er å lage et posisjoneringssystem som bruker prinsippene om absolutt posisjonering. Det vil si å bruke kjente punkter i rommet til å beregne posisjonen sin. Hvis man finner avstanden til to faste punkter i rommet kan man regne seg frem til hvilken posisjon man selv er i. Vi skal bruke ultralyd til å bestemme avstanden fra punktene til roboten. Figuren under er en enkel beskrivelse av ideen.



For å finne avstanden fra ultralydsender(plassert på faste punkter i rommet) til ultralydsensor(sitter på robot) må vi finne tiden ultralydpulsen bruker fra sender til sensor. For at roboten skal vite når den skal starte klokka vil vi bruke IR LEDs til å fortelle roboten når den skal begynne å ta tiden.

For at roboten skal kunne skille signalet fra de forskjellige ultralydsenderene fra hverandre blir pulsene sendt ut i bestemt rekkefølge som roboten er kjent med. Det er viktig at det er en pause mellom de forskjellige ultralydpulsene slik at refleksjoner fra forrige puls har lagt seg

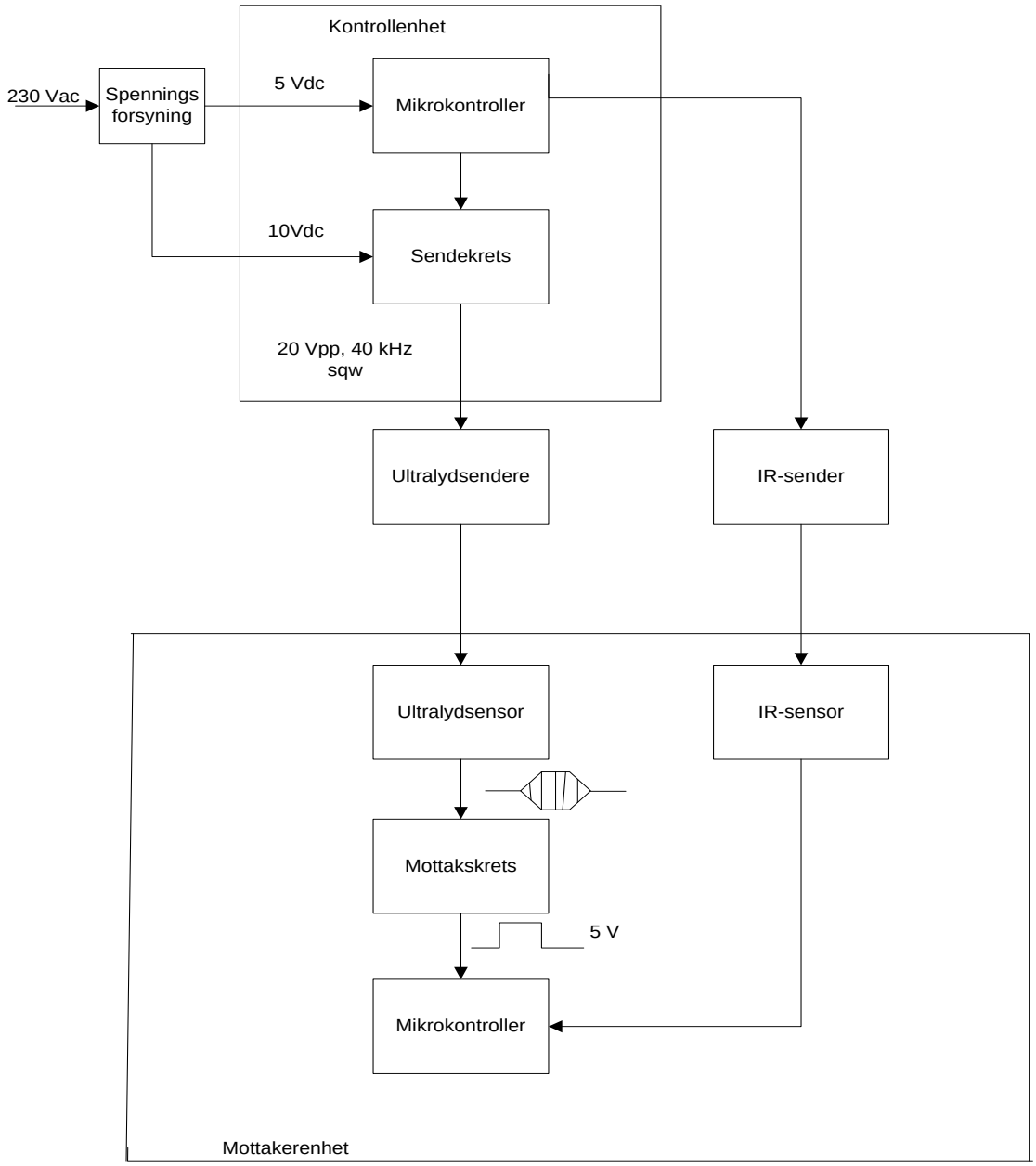
Når roboten kjenner avstanden til de faste punktene beregner den sin egen posisjon som en xy koordinat.

12.2 SYSTEMET

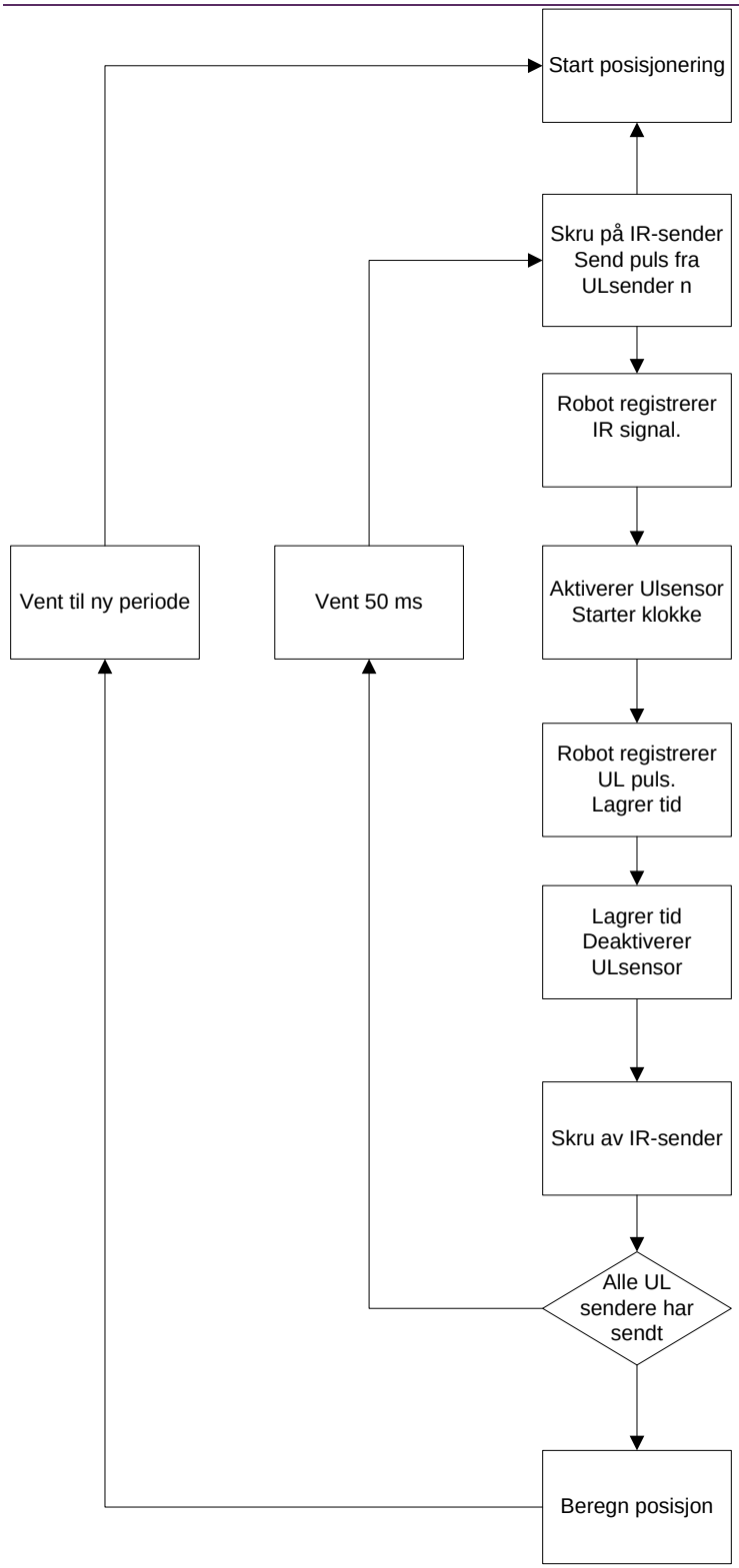
12.2.1 DELSYSTEMER

Systemet vil bestå av to forskjellige delsystemer; kontrollenheten og mottakerenhet. I kontrollenheten sitter det en mikrokontroller som styrer ultralydsenderene og IR-LEDene. Denne styrer hele posisjoneringsprosedyren.

Mottakerenheten sitter på hver robot. Den vil bestå av en ultralydsensor, en IR-sensor og en mikrokontroller. IR-sensoren er koblet direkte opp mot mikrokontrolleren mens ultralydsensoren er koblet til mikrokontrolleren gjennom en mottakerkrets. Mottakerkretsens oppgave er å forsterke opp og omforme signalet fra ultralydsensoren til et signal som mikrokontrolleren kan registrer. Mikrokontrolleren bruker så informasjonen den har fått til å regne ut posisjonen sin.



12.2.2 HENDELSESFORLØP



Figuren over viser en grov oversikt over hendelsesforløpet.

Kontrollenheten starter posisjoneringsprosessen ved å skru på IR-LEDen. Samtidig sender den ut en ultralydpuls på 1 ms fra ultralydsender 1. Mottakerenheten registrerer IR LEDen og starter klokka (tiden lyset bruker på å nå mottakerenheten er så liten at det blir som om mikrokontrolleren starter klokken i samme øyeblikk som ultralydpulsen sendes ut). Etter en stund registrerer mottakerenheten pulsen og finner tiden pulsen har brukt.

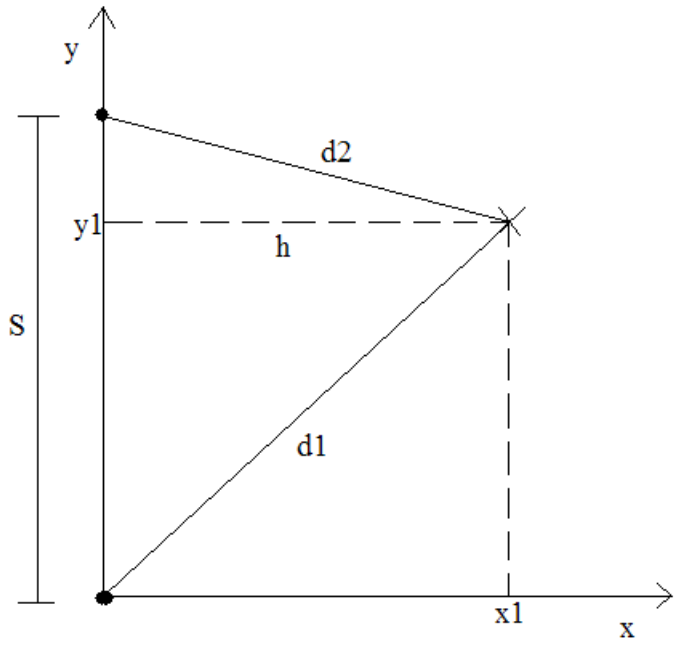
Etter 12ms (tiden det tar for lyd å reise 4m) skrues IR- LEDen av. Det betyr at mottakerenhet skal slutte å vente på pulsen. Hvis den ikke har mottatt noen puls innen det betyr det at roboten er utenfor rekkevidde.

Etter ca 50 ms sendes det ut en puls fra ultralydsender 2. Dette fortsetter til alle ultralydsenderene har sendt ut en puls. Kontrollenheten vil bruke (n-1)*50ms (n er antall ultralydsendere) på en posisjoneringssyklus.

Når mottakerenheten har registrert at IR-LEDen har skrudd seg av n ganger vil den beregne sin egen posisjon (Hvis den har mottatt nok ultralydpulser til å gjøre det). Det er viktig at mottakerenhet alltid har kontakt med IR-LEDen, ellers vil den miste oversikten over hvor den er i posisjoneringssyklusen.

12.3 ALGORITMEN

Det matematiske prinsippet som blir brukt kalles trilateration. Vet du avstanden til to eller fler punkter kan du beregne posisjonen.



Når d1 og d2 er bestemt kan vi begynne å beregne posisjonen. Linjene S, d1 og d2 danner en trekant, vi finner halve omkretsen dens (semi-perimeter).

$$S_P = \frac{S + d1 + d2}{2}$$

Videre bruker vi Herons formel for å beregne arealet av trekanten.

$$A = \sqrt{S_P(S_P - S)(S_P - d1)(S_P - d2)}$$

Når arealet til trekanten er kjent kan høyden (h) beregnes.

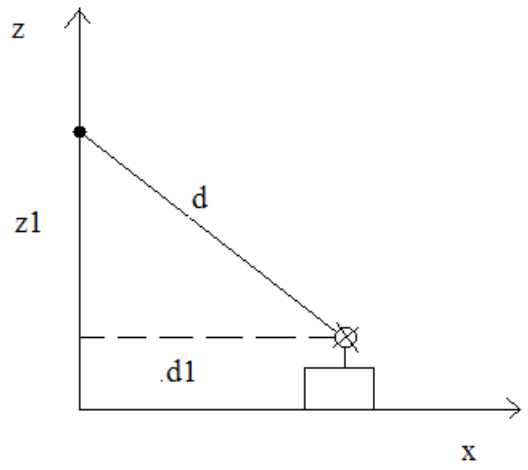
$$h = \frac{2A}{S}$$

Når sensorene er plassert som vist på tegningen er h=x1. Når x1 er kjent kan y1 beregnes.

$$y_1 = \sqrt{d1^2 - x_1^2}$$

Denne algoritmen gir ingen entydig løsning men plasseres senderene som vist på figuren over trenger vi kun å ta hensyn til en løsning. En annen metode for å få en entydig løsning er å innføre en tredje sender.

Denne utregningen gjelder kun hvis senderne og sensorene har samme høyde. Siden det skal være hindringer plassert rundt på området vil det være nødvendig å plassere senderne høyere slik at hindringene ikke blokker for signalet.



Kan finne d1 slik:

$$d1 = \sqrt{d^2 - z1^2}$$

Det samme gjøres for d2. Da kan vi bruke metoden over for å finne posisjonen. Det vil kreve litt mer tid og kraft å beregne posisjonen.

12.4 KOMPONENTER

IR LED

Type:	Everlight IR333
Strålingsvinkel	40’
Bølgelengde	940 nm

IR mottaker

Type:	Everlight IRM8608S
Inntaksvinkel	90’
Bølgelengde	940 nm
Carrier frekvens	36 kHz

Mikrokontroller

Type:	PIC18F4520
Pinner:	40
CCP:	2 stk
Vdd	5 V

OP AMP

Type:	MCP6024
Båndbredde:	10MHz
Vdd	2,5-5,5 V
Slew rate	7 V/us

12.4.1 ULTRALYDSENDER/MOTTAKER

Type:	Murata MA40SR/S
Strålingsvinkel:	80`
Rekkevidde:	0.2-4 m
Frekvens:	40 kHz

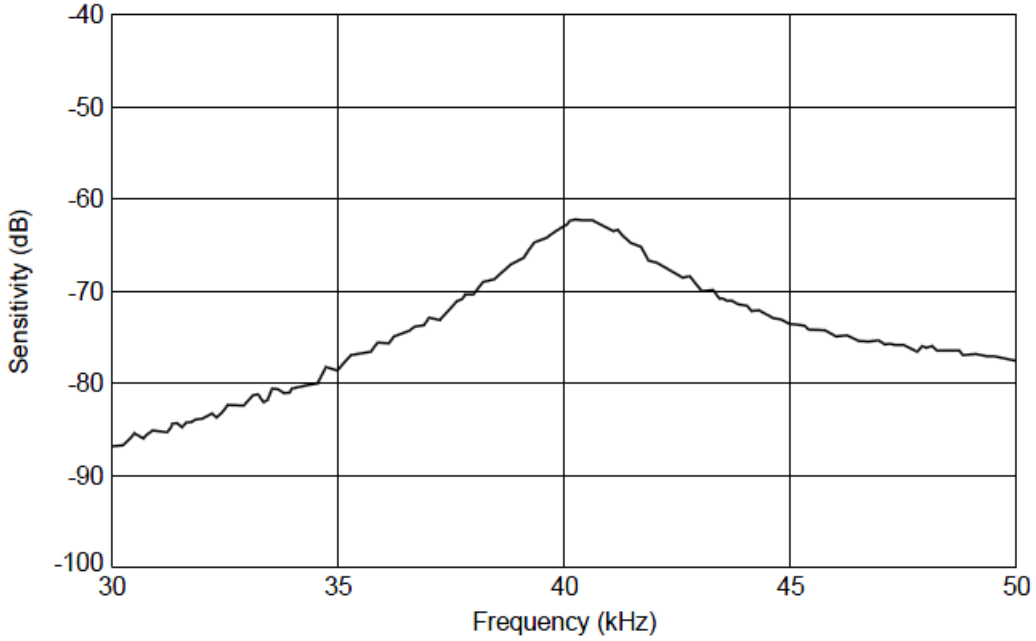
12.4.1.1 VIRKEMÅTE

For at ultralydsenderen skal sende trenger den en firkantpuls på 40kHz. Rekkevidden til senderen er avhengig av styrken til pulsen. For maks rekkevidde trenger den en firkantpuls på 20 Vpp.

Ultralydsensoren omdanner vibrasjonene den får fra ultralysenderen til en spenning. Dette er et svakt signal som inneholder endel støy. Styrken på signalet minker eksponentielt med avstanden mellom sender og sensor. Med en avstand på 4 m får man ut et signal på
20 mVpp.

12.4.1.2 SENSITIVITET

Ultralydsensoren er sensitiv for frekvenser rundt 40 kHz mens sensitiviteten faller og blir raskt lavere for andre frekvenser.



Dette er viktig slik at sensoren ikke blir påvirket av bakgrunnsstøy. Det vil være mye menneske skapt støy i hallen. Stemme båndet til et menneske klarer ikke å produsere frekvenser som vil skape problemer for sensoren. Tidligere prosjekter som har brukt denne sensoren har ikke opplevd problemer med menneskeskapt støy.

Et problem vil være ultralydsensoren som skal brukes til måling av avstand til objekter siden denne bruker samme frekvens. Det vil derfor være nødvendig å deaktivere denne sensoren under posisjoneringsprosedyren.

12.4.1.3 TEMPERATUR

Hastigheten til lyd i luft blir påvirket av temperaturen. Sammenhengen mellom hastighet og temperatur gis av denne funksjonen.

$$c = 331.5 + 0.607t$$

Hvis avstanden mellom sender og sensor er 4 meter vil tiden lyden bruker være 11.84 ms ved 10 grader og 11.64 ms ved 20 grader. Dette gir en feil på 7 cm. Dette er en betydelig feilkilde som sammen med andre feilkilder gjør systemet lite presist.

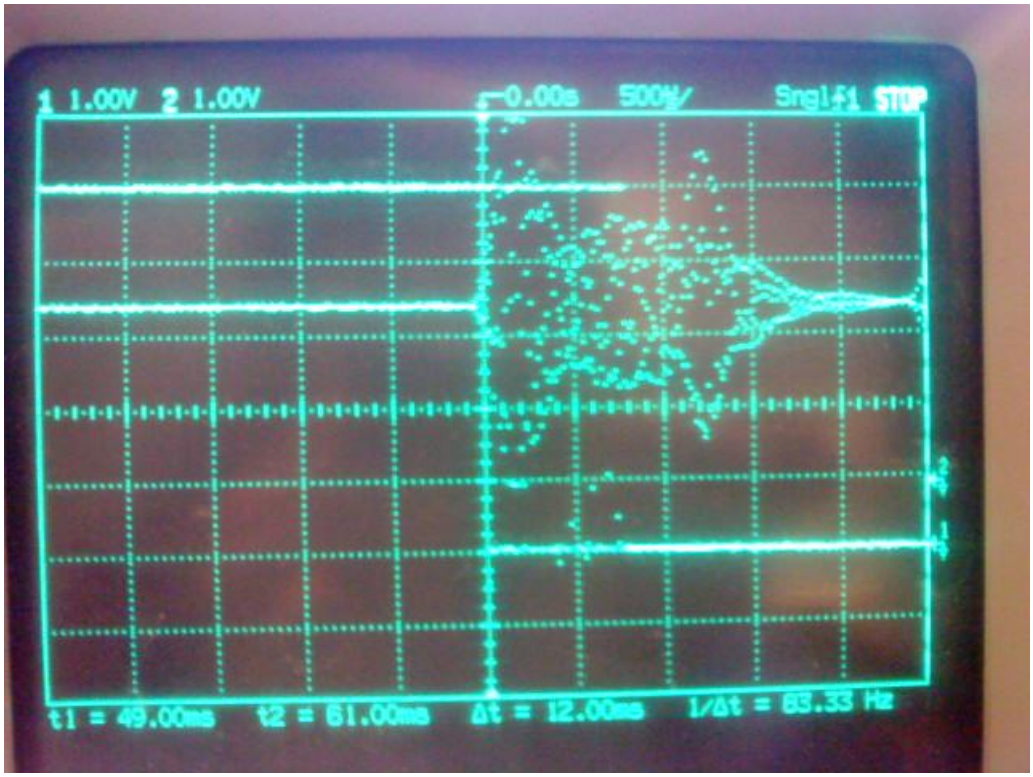
Det er sannsynlig at temperaturen i hallen systemet skal plasseres i vil svinge med minst 10 grader i løpet av året. Det vil derfor være nødvendig å kalibrere systemet. Dette kan gjøres ved å bruke temperatursensoren på sun SPOT enheten og oppdatere hastigheten til lyden.

12.4.1.4 REFLEKSJON

Et problem med systemet vil være refleksjoner. Bølgene vil reflekteres rundt i hele rommet og i en kort tid vil rommet være fullt av refleksjoner. Dette må tas høyde for i designet av posisjoneringssystemet.

Ultralydpulsene må sendes ut med et intervall slik at vi er sikre på at de tidligere bølgene har dødd ut. Dette intervallet vil være på 50 ms, da vil med sikkerhet alle bølger ha dødd ut.

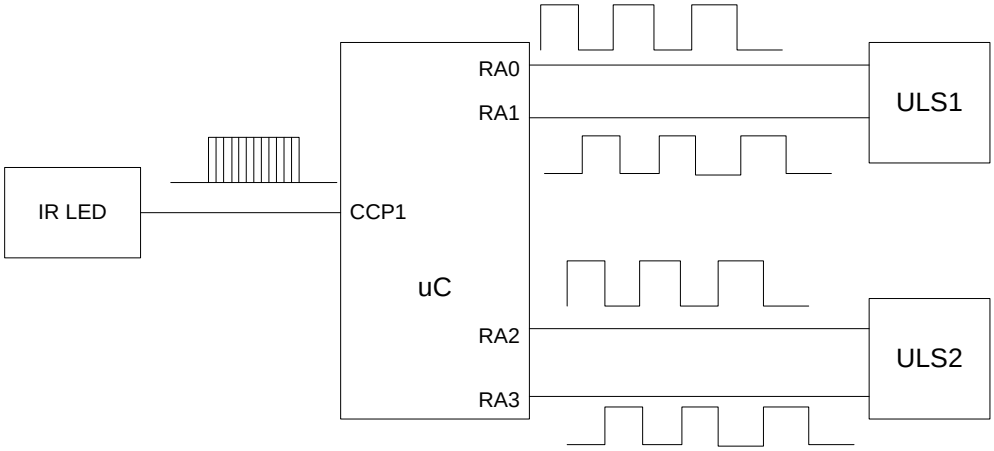
Ultralyd sensoren aktiveres kun rett før den skal motta en puls og deaktiveres rett etter den har mottatt pulsen. Dette gjøres for å minke effekten av refleksjoner og bakgrunnsstøy.



Figuren viser effekten av refleksjoner på utgangen til forsterkerkretsen.

12.5 KONTROLLENHET

Kontrollenhetens oppgave er å sende ut ultralydpulsene og skru av og på IR-LEDen.



Figuren over viser en skisse av kontrollenheten.

Beinene til ultralydsenderen kobles til hver sin pinne på mikrokontrolleren. Mikrokontrolleren sender en firkantpuls på 40 kHz med motsatt fase inn på hvert bein. Dette skaper en firkantpuls med styrke 2*Vdd. Mikrokontrolleren kan da maksimalt skape en firkantpuls på 10 Vpp. For å oppnå senderens maksimale rekkevidde trenger man et signal med styrke på 20 Vpp. Dette kan gjøres ved å koble inn to push-pull MOSFET transistorer.

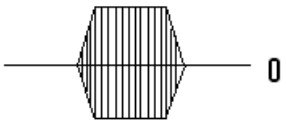
IR LEDen må sende et signal med frekvens på 36 kHz for at IR mottakeren skal oppfatte det. Dette signalet får den fra mikrokontrolleren.

12.6 MOTTAKERENHET

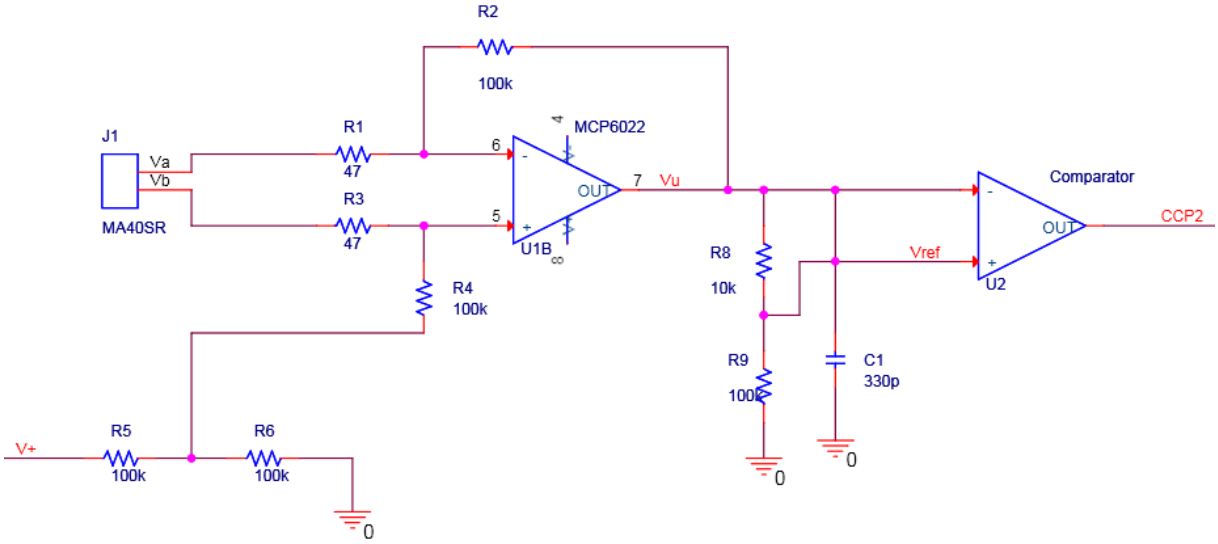
Mottakerenheten består av tre deler

1. Ultralydsensor med mottakerkrets
    a. forsterker signalet fra ultralydsensoren
2. IR mottaker
3. Mikrokontroller

Signalet som kommer fra ultralydsensoren ser slik ut:



Dette er et svakt signal, det kan bli så lavt som 20 mVpp og må derfor forsterkes kraftig. Mottakerkretsen består av to deler; en operasjons forsterker og en komparator.



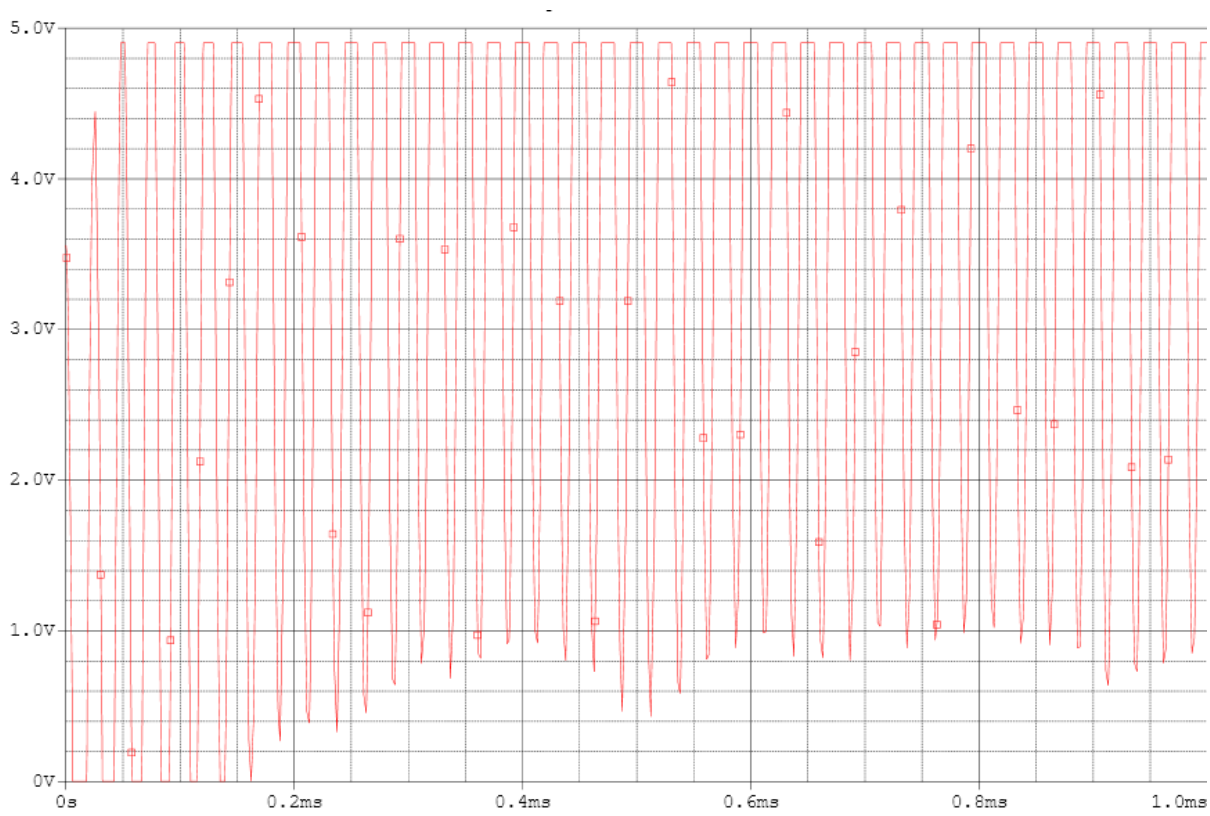
Forsterkerkretsen er en differensiell forsterker. Bena på ultralydsensoren er koblet til hver sin inngang på OP AMPen. Den forsterker forskjellen i spenningen på inngangene. Dette gir en større forsterkning i forhold til om en av inngangene hadde vært koblet til jord.

Forsterkeren som brukes er av typen MCP6024. Den har høy båndbredde noe som er viktig for å få stor forsterkning ved høye frekvenser.

Vu blir gitt av formelen:

$$V_u = \frac{R_2}{R_1}(V_a - V_b) + 2V_{dd}$$

Simulering av forsterkerkretsen gir resultatet som vist i figuren under:



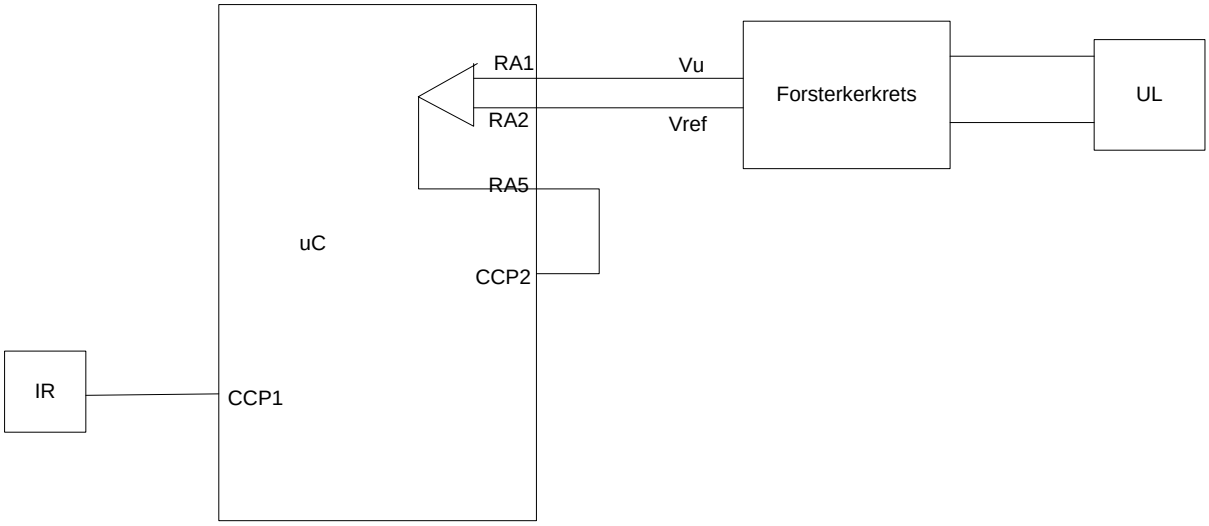
Her er inngangs signalet en sinus kurve på 40 kHz med spenning på 20 mVpp.

Den andre delen av kretsen er en komparator. Denne sammenligner Vu med Vref, hvis Vu>Vref gir den ut 5 V og hvis Vu<Vref gir den ut 0 V.

Vref settes slik at støy fra ultralydsensoren ikke vil påvirke utgangen til komparatoren.

$$V_{ref} = V_u \left(\frac{R_9}{R_9 + R_8} \right)$$

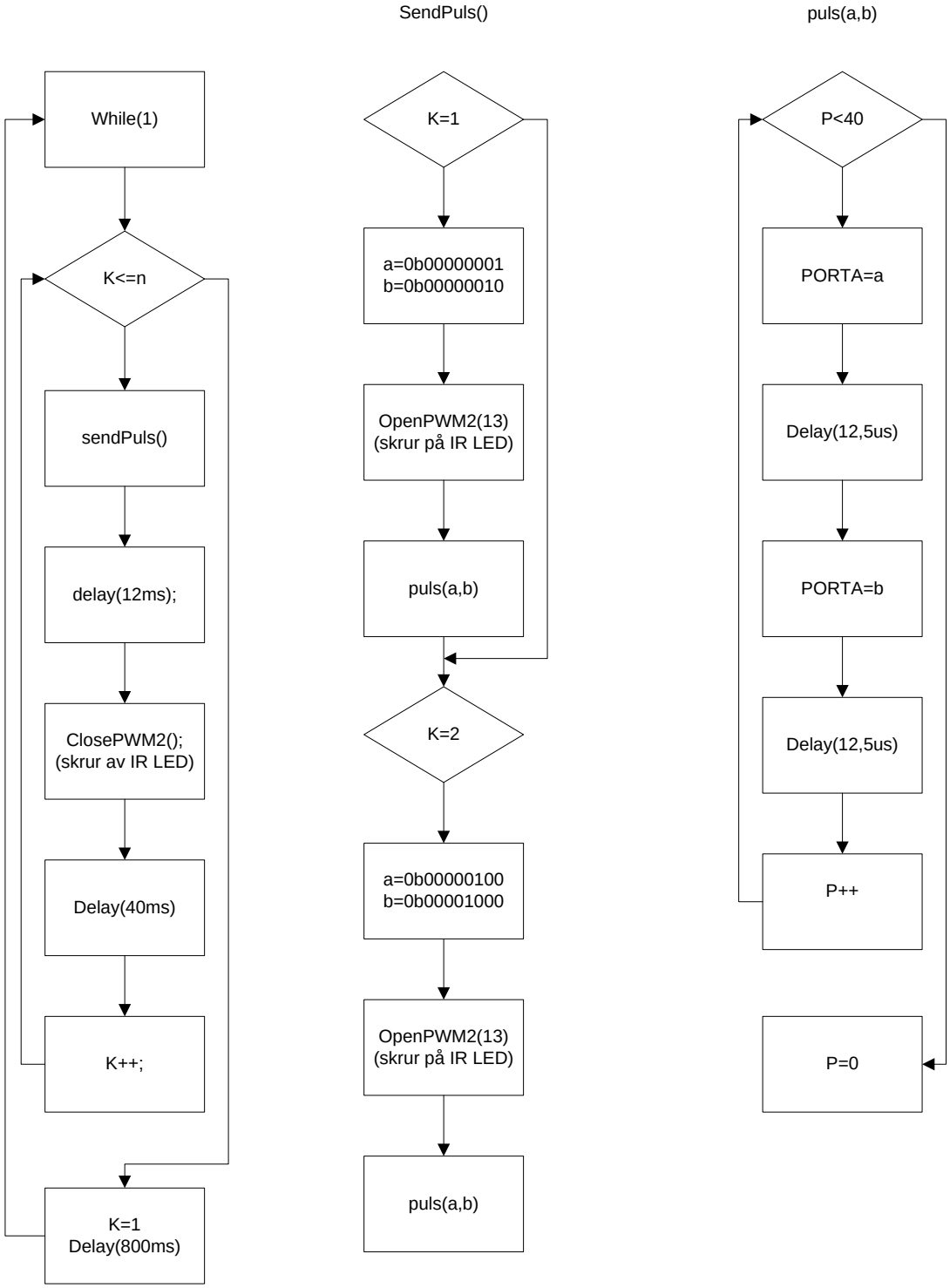
Utgangen til komparatoren vil være høy (5 V) konstant før ultralydsensoren mottar en puls. Når den mottar en puls vil utgangen til komparatoren skifte mellom å være høy og lav med samme frekvens som pulsen.



Figuren viser hvordan mottaksenheten er satt sammen. Mikrokontrolleren har en innebygd komparator som vi velger å bruke.

12.7 PROGRAMMERING

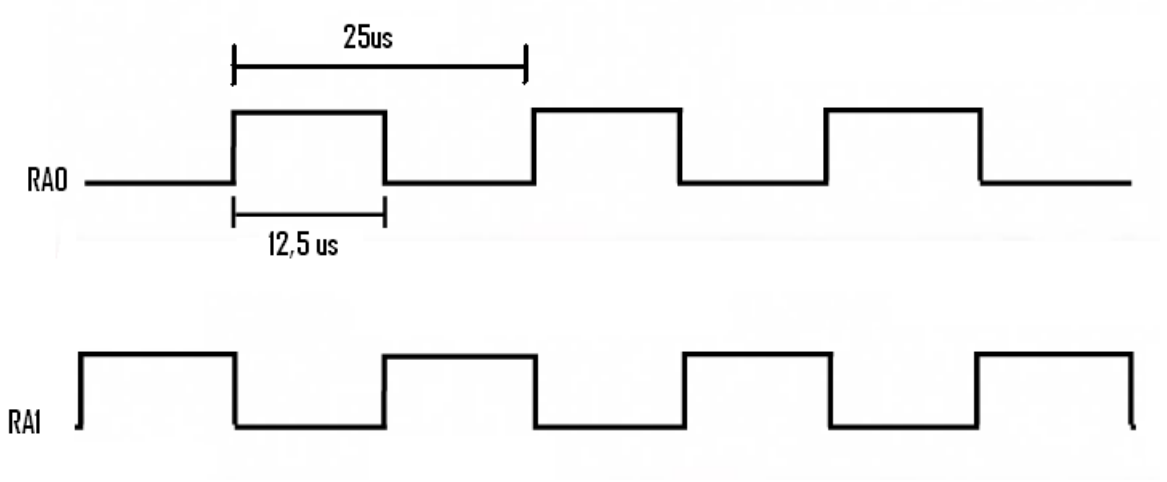
12.7.1 KONTROLLENHET



IR LED
For at IR mottakeren skal registrere IR signalet må det ha en frekvens på 36 kHz. Bruker PWM funksjonen til mikrokontrolleren for å lage denne pulsen.

1. OpenPWM(PR2)
    a. Setter perioden og starter PWM funksjonen
    b. PR2 = 13. Dette gir en periode på ca 28 us.
2. SetDCPWM(dc)
    a. Setter dutycyclen til PWM funksjonen
    b. dc= 28.
3. ClosePWM
    a. Stenger PWM funksjonen

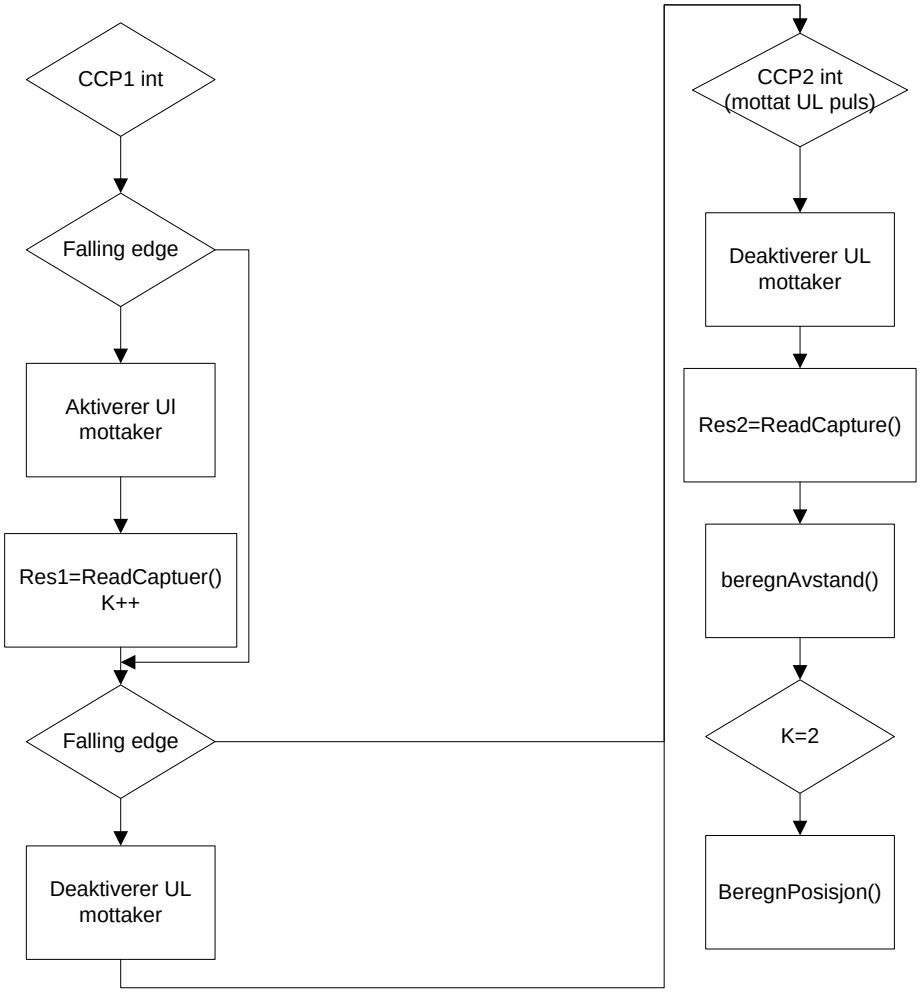
Ultralydsender
Lager firkantpulsen med periode på 25 us. Gjør det ved å sette pinnen høy i 12,5 us og så lav i 12,5 us.



Pulsen sendes i motsatt fase til hvert bein på ultralydsenderen. Tilstanden til beina må endres likt. Dette gjøres ved å skrive til hele porten og ikke bare hver enkelt pinne:

PORTA= 0b00000001 (RA0 høy, RA1 lav)
PORTA= 0b00000010 (RA0 lav, RA1 høy)

12.7.2 MOTTAKERENHET



Når IR mottakeren registrer et signal vil den sette signal pinnen sin lav. Mikrokontrolleren gir et interrupt og henter en verdi fra Timer1. Et nytt interrupt inntreffer når ultralydpulsen blir mottatt. En ny verdi fra Timer1 hentes og avstanden til den første ultralydsenderen beregnes.

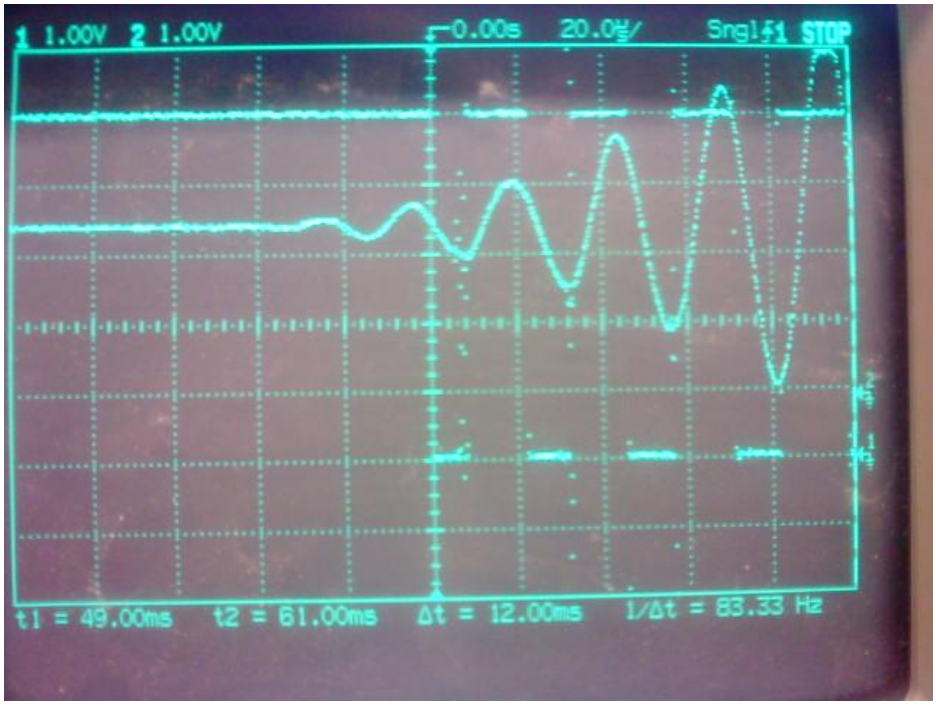
$$s = tDiff * 0.0344$$

Dette gjentar seg til alle ultralydsendere har sendt en puls. Da regner mikrokontrolleren ut posisjonen (se avsnittet om algoritmen).

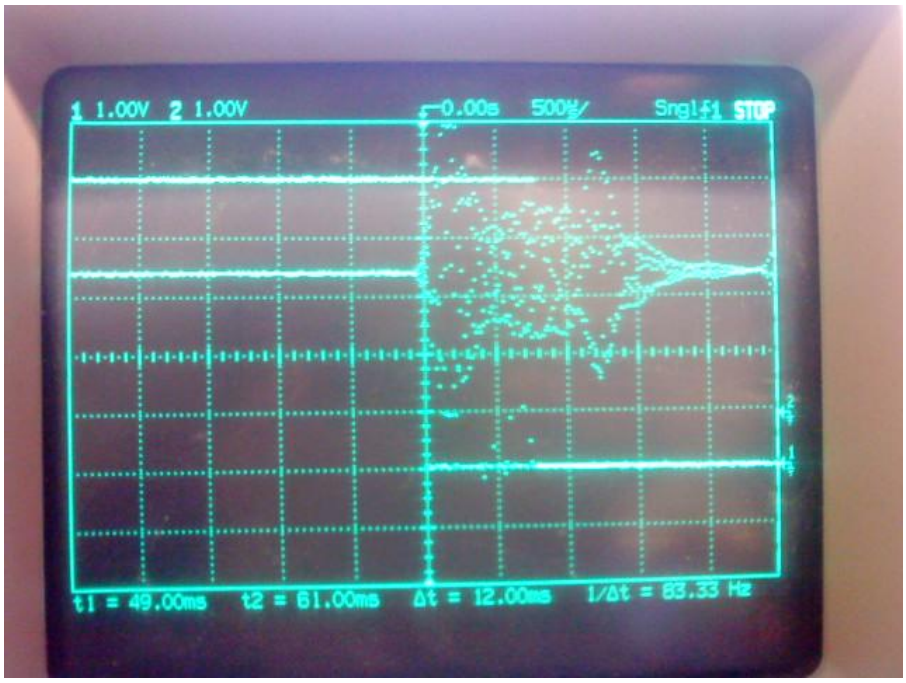
12.8 PROTOTYPE

Vi har bygget en prototype for testing på mindre skala. Prototypen er bygget som beskrevet i avsnittene over.

Vår prototype bruker et signal på 10 Vpp for å drive ultralydsenderene. Tester viser at dette gir en rekkevidde rett i underkant av 2 m. Dette holder til testing på mindre skala men hvis systemet skal fungere i newton rommet trenger vi større rekkevidde.



Bildet over viser signalet på utgangen til forsterkerkretsen og komparatoren i øyeblikket de mottar en puls. Ser at signalet i begynnelsen er så svakt at komparatoren ikke gir noe utslag. Etter omtrent 100 us er signalet blitt sterkt nok til at komparatoren reagerer. Denne forsinkelsen i systemet blir større desto lenger vekk ultralydsenderen er. For å minimere forsinkelsen kan man sette Vref nærmere Vu. Dette fører til en raskere respons fra komparatoren men man må være forsiktig slik at støy ikke vil føre til endringer på utgangen til komparatoren. Kan også ta høyde for forsinkelse i softwaren, vi kalibrerer systemet ved å trekke fra 100 us.



Ser på figuren over at etter pulsen er det betydelige refleksjoner etter pulsen har blitt sendt. Ultralydsensoren har blitt plassert i nærheten av en vegg (her antas refleksjonene å være størst) men vi har ennå ikke opplevd noe problem med refleksjoner under tester. Grunnen til dette mener vi er at ultralydsensoren kun er aktiv i veldig korte perioder, den blir aktivert rett før den skal motta pulsen og deaktivert øyeblikkelig etter den har registrert pulsen. I tillegg har vi en stor pause (50 ms) mellom hver puls slik at alle refleksjoner dør ut.

Initielle tester viser at systemet har en presisjon på rundt 6-7 cm. Vi har ikke fått utført tilstrekkelig med tester ennå til å garantere denne presisjonen over hele området. Systemet er til dags dato ikke kalibrert til maks ytelse. Etter dette er gjort vil vi anta at presisjonen i hvertfall vil ligge innenfor 10 cm når systemet blir satt opp i Newton rommet.

Det vi mangler før systemet kan plasseres ut er:

1. Rekkevidden må økes
    a. Systemet burde minst ha en rekkevidde mellom 3,5-4 m.
    b. Driver signalet til ultralydsenderen må økes til 20 Vpp
2. Systemet må kalibreres
    a. Ønsker en presisjon på under 10 cm
    b. Presisjonen må være tilnærmet den samme for alle punkter i rommet
3. Plassering av ultralydsendere må bestemmes.
    a. Minst to sendere må alltid ha kontakt med ultralydsensoren
4. Softwaren må også utvides for fler ultralydsendere.
5. Videre testing
    a. Vi har kun utført initielle tester, nye problemer kan dukke opp ved mer nøye testing.

13 ARBEID- OG ANSVARSOMRÅDER:

Her er en oppsummering av de arbeidsområder gruppemedlemmene har hatt å arbeide med.

Fredrik Gulbrandsen:
- Gruppeleder
- Posisjoneringsystem
- Mikrokontroller
- Sensorer

Sebasitan Teie:
- Testing
- Design av kretskort
- Anskaffelse av komponenter

Nedim Golo:
- Produksjon
- Ladesystem

Helene Ruud:
- Dokumentasjon
- Sun SPOT

Kim Pollvik:
- Server
- Webinterface
- Java Applet & Java Server

14 BUDSJETT

Komponenter	Pris per stk	Antall	Totalt	Leverandør
Panasonic blyakkumulator	124,-	5	620,-	Elfa
TI batterilader	59,-	10	590,-	Farnell
Kobberplate 5cm *3m	250,-	1	250,-	Kongsberg Blikk AS
Kobberplate 14cm*3m	250,-	1	250,-	Kongsberg Blikk AS
Microswitcher	ukjent	20	ukjent	Farnell

15 EVALUERING

15.1 PROSESSEN

Gruppen ble i høst satt sammen etter et møte for studenter uten noen prosjektgruppe. Vi var da kommet godt inn i hovedprosjektfaget og lå da allerede et godt stykke etter de andre gruppene. Det var liten tid for å søke på prosjektoppgaver, og valgte da å ta på oss en prosjektoppgave levert fra Høgskolen. Oppgaven gikk ut på å bruke resultatet fra et tidligere prosjekt og videreutvikle det for bruk i læringssammenheng og interesseskaping for elever i ulike skoletrinn fra Drammensområdet. Prosjektet skulle stilles ut og kunne benyttes i et Newton rom, som er lokalisert i Drammen.

Gruppen arbeidet for å hente igjen forsinkelsen og tapt arbeidstid, og klarte bare såvidt å ha nødvendig dokumentasjon klart til innleveringsfristen før juletider. Til tross for en dårlig start hadde gruppen satt opp en stram prosjektplan, og hadde klar tro på å lykkes i å lage det oppdragsgiver ville ha.

Like inn på nyåret hadde gruppen sin første presentasjon, hvor planene for prosjektet skulle presenteres.

Gruppa startet raskt opp med arbeidet rett etter første presentasjon. Diverse nødvendig utstyr som Sun SPOT-sett og server ble lånt fra skolen. Gruppas elektrostudenter startet med å teste løsningene til den forrige gruppen. Datastudentene satt i gang med å konfigurere og installere serveren, lage nettsidene og teste Sun SPOT enhetene. Det er under disse oppgavene at de første bristene i prosjektplanen kommer. Sun SPOT-settene ville ikke fungere som de skulle og robotens evner stemte ikke overens med det som var gitt inntrykk for i forrige gruppes dokumentering. Nye aktiviteter blir lagt til for å løse disse problemene. Allerede ved dette tidspunktet ligger gruppen et ganske godt stykke etter ifølge prosjektplanen.

Gruppa har i denne perioden et møte med oppdragsgiver, hvor det ønskes at deler av systemet skal stå klart og fungerer for demonstrasjon før påske. Kravspesifikasjonen blir også gått igjennom sammen med oppdragsgiver for å oppdatere hvilke deler av prosjektet som kan nedprioriteres. Gruppa gjøres også oppmerksom av veileder at fremskrittene er for små, og at det for dårlig dokumentering av aktiviteter og hva arbeidstimene går til.

Gruppa tar et lite avbrekk fra prosjektet for å lese på eksamener.
Gruppa rakk ikke å levere som ønsket til påske, og rakk også denne gang bare såvidt å skrive nødvendig desingdokumentasjon til fremføringen, etter å ha brukt flere av helligdagene i påsken til dokumentskriving og øving. Presentasjon 2 var på første virkedag etter påske, og gruppa informerte på spørsmål fra sensor at vi ikke ville komme til å bli ferdige. Vi fikk også tilbakemelding om at vi ikke var helt ferdige med designstadiet utifra hva som ble levert av dokumentasjon.

Gruppa har så nok en gang et møte med oppdragsgiver og får beskjed om å bare fokusere på å få noe som fungerer klart til neste uke. Gruppa har her et gjennombrudd i prosjektet og får den etterlengtede kommunkiasjonen i roboten til å fungere som den skulle. Gruppa klarte også like etter å få ønsket kommunikasjon gjennom hele systemet. Gruppa trapper nå opp tempoet for å rekke å få mest mulig ferdig før innleveringsfrist. Kretskort blir omsider laget, server koblet opp på skolen og en prototype av posisjoneringssystem blir laget.

Gruppa rakk ikke å komme noe nevnbart lengre før dokumenteringen måtte gjøres ferdig for prosjektrapport innleveringen.

15.2 STATUS

Vi har ikke blitt ferdig med prosjektet i henhold til kravspesifikasjonen som vi først satte opp med oppdragsgiver. Underveis i gjennomføringen av prosjektet har vi måtte velge krav som skal prioriteres framfor andre. Dette har ført til at det er krav som har blitt valgt bort. Vi skal nå kortfattet gå gjennom hva vi fikk gjort, og hva som gjenstår.

15.2.1 HVA VI HAR GJORT

- Programmert mikrokontrolleren til å kunne styre servoene for framdrift og manøvrering
- Programmert mikrokontrolleren til å kunne motta og tolke verdier ifra sensorene (IR-sensorene, ultralydsensorene og støtsensorene). For eksempel vil spenningen den infrarøde sensoren sender til mikrokontrolleren representere en avstand, og mikrokontrolleren er programmert til å kunne beregne denne avstanden.
- Vi har designet en krets som hvis man kobler til en spenningskilde på mer enn 10V, vil lade batteriet som er montert på roboten. Vi har brukt en såkalt ”smart” lader integrert i en krets som sørger for å optimalisere ladingen av batteriet.
- Gjennom en spenningsfordeler koblet til en AD-konverter på mikrokontrolleren har vi satt en passende grense for når batteriet må lades.
- Designet et kretskort der alle komponenter får strøm og spenning ifra batteriet. Vi bruker spenningsregulatorer til å forsyne komponentene med riktig spenning. Ladekretsen inngår på kretskortet.
- Opprettet UART kommunikasjon mellom SunSPOT og mikrokontroller. SunSPOT sender kommandoer til mikrokontrolleren, og mikrokontrolleren utfører disse (kjøre fram, rygge, svinge)
- Designet et posisjoneringssystem som vi har testet i liten skala. Vi bruker ultralydsensorer til å beregne posisjonen ut ifra to faste punkter i rommet.
- Designet og implementert en Java Applet og en Java server.
- Konfigurert stasjonær server til å hoste en nettside, Applet og Java server.
- Laget nettsiden for prosjektet, som inneholder Java Appleten.
- Oppnådd kommunikasjon hele veien fra Java Applet til SunSPOT-enheten på Bugsteren.
- Programmert en applikasjon som:
    - lar en Sun SPOT enhet kommunisere med en mikrokontroller via UART, og tolke de innkommende kommandoer.
    - Utfører et kjøreprogram bestemt av bruker.
    - Kan Kommunisere med ett webinterface

15.2.2 IKKE FULLFØRTE OPPGAVER

- Implementere et fullstendig posisjoneringssystem.
- Bygge fem komplette roboter.
- Tre programmeringsnivå
- Programmert så robot kan komme seg til ladestasjonen
- Muligheter for flere tilkoblinger til server samtidig
- Mulighet for bevaring av koden
- Lage brukerkonto system
- Sette opp systemet i Newtonhallen

16 KONKLUSJON

Målet med oppgaven var å lage et webinterface som kan gi instrukser til en samling roboter. Robotene skulle ellers helt selv kunne holde oversikt over egen plassering og eget ladebehov. Vi har ved prosjektets slutt fått en robot til å motta instrukser fra et webinterface. Vi har også laget en ladestasjon og en prototype til posisjoneringssystem. Vi har ikke fått roboten til å lade seg selv automatisk, laget flere programmeringmåter eller fått satt opp løsningen i Newtonhallen.

Roboten fungerer ved at den mottar en bokstavstreng, og ut ifra denne vurderer hva brukeren i webinterfacet har bestemt at roboten skal gjøre. Roboten bruker også flere sensorer for å utøve en slags logikk ved evnt hindring i dens kjørebane.

Som noe av årsaken til at prosjektet ikke nådde så langt som først forventet, er fordi gruppa samlet ikke har brukt nok tid. En annen årsak er at gruppa i løpet av prosjektperioden har gjort flere gale valg, og angrepet flere av problemmene på feil måte.

Vi har igjennom prosjektet tilegnet oss gode erfaringer innenfor prosjektarbeid og samarbeid på både godt og vondt. Vi føler at ved senere prosjekt ville ungått svært mange feil som vi desverre har gjort i dette prosjektet. Alle prosjektmedlemmene føler også en kraftig økning i deres faglige kompetanse.

17 REFERANSER

Data:
Deitel – Java How to Program. Seventh Edition
An introduction to network programming with Java
www.sunspotworld.org

Elektro
www.microchip.com
www.batteryuniversity.com
Analog Devices, bok fra faget Analog Elektronikk
PIC18 Programming, bok fra faget Mikrokontroller
Orcad brukermanualer, notater fra faget Kretskortkonstruksjon
www.Elfa.se
www.farnell.no
www.wikipedia.com
Hovedprosjektrapport HIBU BUGSTER 2008
Mechatronics, bok fra faget Mekatronikk

Posisjonering
http://www.tankeogteknikk.no/index.php?option=com_content&view=article&id=2&Itemid=4
http://wenku.baidu.com/view/dbb9f236f111f18583d05a07.html

Vedlegg 1

Brukermanual

26/5/2011

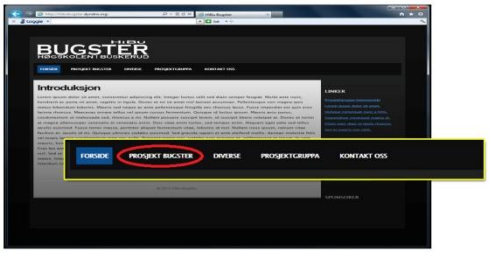
BUSKERUD
FYLKESKOMMUNE

Brukermanual | [Type the author name]

1

1. Åpne din nettleser og skriv inn prosjektets webaddresse: http://hibubugster.dyndns.org

2



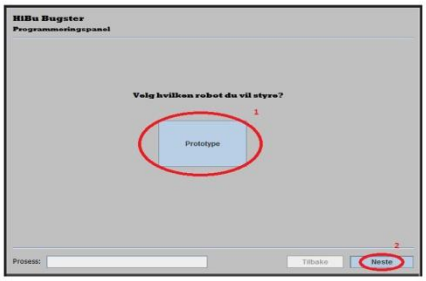
2. Her kan du se info om prosjektet og prosjektgruppa.
Klikk på 'Prosjekt Bugster' for å komme til Bugsters kontrollpanel.

3

3. Det vil da lastes inn en Java applikasjon i nettleseren din. Dersom du skulle få en forespørsel om å gi tillatelse til å kjøre programmet, godta dette.

NB: Du er nødt til å ha installert Java Runtime Environment (JRE) for å starte programmet.

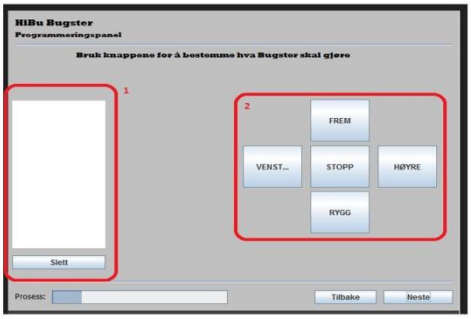
4

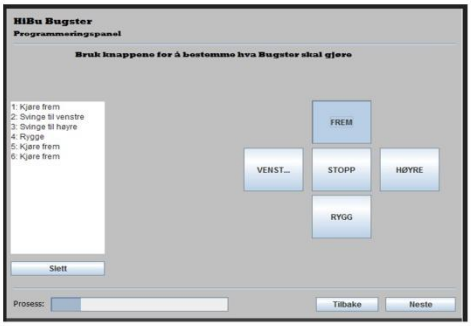


4. Velg her hvilken robot du vil kontrollere, ved å trykke inn robotknappen 'Prototype'(1) Deretter kan du gå videre ved å trykke knappen markert 'Neste'(2)

NB: Merk at etter å ha trykket 'Neste' vil Tilbake-knappen bli tilgjengelig slik at du kan gjøre endringer i dine valg.

5



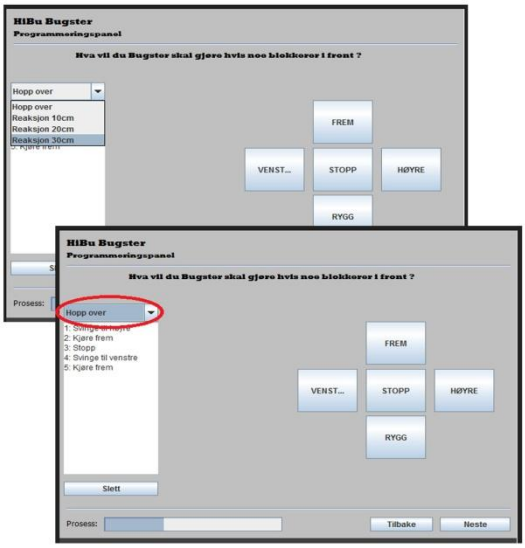


5. Her har du ditt kontrollpanel.
Markert i Ring1 har du kommandolisten, hvor dine valg legger seg i rekkefølge.
I Ring2 har du knappene som du gjør dine valg med.

Dersom du vil endre rekkefølgen du har lagd, kan du trykke på slett-knappen som er markert i Ring1. Denne vil slette rekken, og du kan lage en ny rekke med kommandoer.

Når du er fornøyd med rekkefølgen, kan du velge Neste-knappen for å gå videre.

6



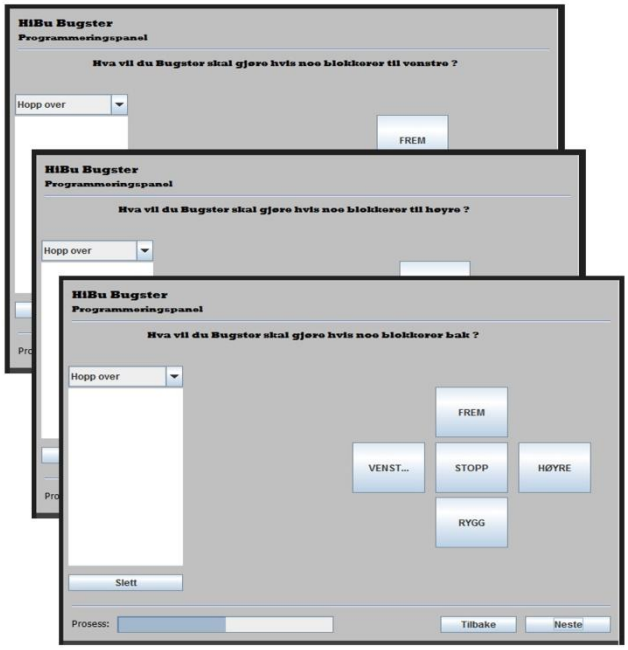
6. Her skal du ta for deg hva du vil roboten skal gjøre dersom noe blokkeren den.
Først skal du avgjøre hva som skal skje dersom dersom noe blokkerer veien foran roboten.

Dersom du ikke vil gjøre noen valg, velger du 'Hopp over' i ruten som er markert med rød ring. Da vil programmet selv ha hånd om dette.

Dersom du derimot vil justere dette selv, må du velge mellom en av de forskjellige reaksjonsavstandene, på 10, 20 og 30 cm.
Dette indikerer hvor nært en hindring kan komme roboten før den begynner å gjøre de valgene du har valgt.

Velg 'Neste' når du er ferdig.

7



7.Her får du mulighet til å programmere hva som skal utføres i tre andre tilfeller slik som tidligere.

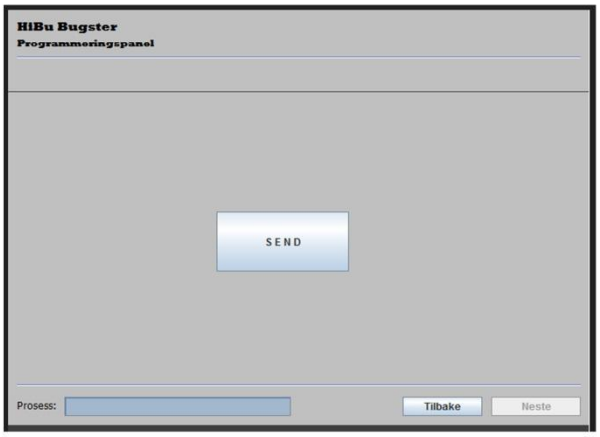
Alle funksjoner og muligheter justeres og bestemmes helt likt som i punkt 6.

Tilfellene du her skal programmere, blir hva roboten skal gjøre hvis det er noe som blokkerer bak, til venstre og sist til høyre.

Du velger 'Neste' og 'Tilbake' for å gå frem og tilbake mellom de forskjellige tilfellene.

Klikk 'Neste' for å gå videre.

8



8.Her er du ferdig med å programmere roboten, og er klar for å overføre dine kommandoer.

Klikk på 'Send'-knappen når du ønsker å sende kommandoene dine til roboten.

NB: Du kan fortsatt endre koden din før du trykker på 'Send'-knappen.
Trykk 'Tilbake'-knappen for gå tilbake.

Vedlegg 2

Kommunikasjonsprotokoll

AIBU BUGSTERS

Komponent	Oppgave	Adresse	Kommando	Info	Heksa
Motor_frem		0001			
	Start		0001		11
	Stopp		0010		12
	Rygg		0011		13
Motor_styring		0010			
nullstilt	90		0001		21
høgre	45		0010		22
venstre	135		0011		23
Motor_ultra		0011			
	90		0001		31
	45		0010		32
	135		0011		33
Nær_Foran		0100			
	10-20			0001	41
	20-30			0010	42
	30-40			0011	43
	40-50			0100	44
	50-60			0101	45
	60-70			0110	46
	70-80			0111	47
Nær_Bak		0101			
	10-20			0001	51
	20-30			0010	52
	30-40			0011	53
	40-50			0100	54
	50-60			0101	55
	60-70			0110	56
	70-80			0111	57
Nær_Høyre		0110			
	10-20			0001	61
	20-30			0010	62
	30-40			0011	63
	40-50			0100	64
	50-60			0101	65
	60-70			0110	66
	70-80			0111	67
Nær_Venstre		0111			
	10-20			0001	71
	20-30			0010	72
	30-40			0011	73
	40-50			0100	74
	50-60			0101	75
	60-70			0110	76
	70-80			0111	77
Ultralyd		1000			
	Start_Ultra		0001		81
	0-30			0010	82
	30-60			0011	83
	60-90			0100	84
	90-120			0101	85

	120-150			0110	86
	150-180			0111	87
	180-210			1000	88
	210-240			1001	89
	240-270			1010	8A
	270-300			1011	8B
Støt_frem		1001			
	Støt			0001	91
	Ikke støt			0010	92
Støt_bak		1010			
	Støt			0001	A1
	Ikke støt			0010	A2
Støt_høyre		1011			
	Støt			0001	B1
	Ikke støt			0010	B2
Støt_venstre		1100			
	Støt			0001	C1
	Ikke støt			0010	C2
Posisjonering		1101			
	Posisjon				
Batterisensor		1110			
	Tom			0001	E1
	Lader			0010	E2
	Ferdig			0011	E3
Div		0000			
	Ikke mottatt			0001	1
	invalid res			0010	2

Sun SPOT	Desimaltall	Hex	Sun SPOT	Desimaltall	Hex
0	0	0x00	-128	128	
1	1	0x01	-127	129	
2	2	0x02	-126	130	0x82
3	3	0x03	-125	131	0x83
4	4	0x04	-124	132	0x84
5	5	0x05	-123	133	0x85
6	6	0x06	-122	134	0x86
7	7	0x07	-121	135	0x87
8	8	0x08	-120	136	0x88
9	9	0x09	-119	137	0x89
10	10	0x0A	-118	138	0x8A
11	11	0x0B	-117	139	0x8B
12	12	0x0C	-116	140	0x8C
13	13	0x0D	-115	141	
14	14	0x0E	-114	142	
15	15	0x0F	-113	143	
16	16		-112	144	
17	17	0x11	-111	145	0x91
18	18	0x12	-110	146	0x92
19	19	0x13	-109	147	0x93
20	20	0x14	-108	148	0x94
21	21	0x15	-107	149	0x95
22	22	0x16	-106	150	0x96
23	23	0x17	-105	151	0x97
24	24	0x18	-104	152	
25	25	0x19	-103	153	
26	26	0x1A	-102	154	
27	27	0x1B	-101	155	
28	28	0x1C	-100	156	
29	29	0x1D	-99	157	
30	30	0x1E	-98	158	
31	31		-97	159	
32	32		-96	160	
33	33		-95	161	0xA1
34	34		-94	162	0xA2
35	35		-93	163	
36	36	0x24	-92	164	
37	37	0x25	-91	165	
38	38	0x26	-90	166	
39	39		-89	167	
40	40		-88	168	
41	41		-87	169	
42	42		-86	170	
43	43		-85	171	
44	44		-84	172	
45	45		-83	173	
46	46		-82	174	
47	47		-81	175	
48	48		-80	176	
49	49		-79	177	0xB1
50	50		-78	178	0xB2

51	51		-77	179	
52	52		-76	180	
53	53		-75	181	
54	54		-74	182	
55	55		-73	183	
56	56		-72	184	
57	57		-71	185	
58	58		-70	186	
59	59		-69	187	
60	60		-68	188	
61	61		-67	189	
62	62		-66	190	
63	63		-65	191	
64	64		-64	192	
65	65	0x41	-63	193	0xC1
66	66	0x42	-62	194	0xC2
67	67	0x43	-61	195	0xC3
68	68	0x44	-60	196	0xC4
69	69	0x45	-59	197	0xC5
70	70	0x46	-58	198	0xC6
71	71	0x47	-57	199	0xC7
72	72	0x48	-56	200	
73	73	0x49	-55	201	
74	74	0x4A	-54	202	
75	75	0x4B	-53	203	
76	76	0x4C	-52	204	
77	77	0x4D	-51	205	
78	78	0x4E	-50	206	
79	79	0x4F	-49	207	
80	80	0x50	-48	208	
81	81		-47	209	
82	82		-46	210	
83	83	0x53	-45	211	
84	84		-44	212	
85	85		-43	213	
86	86		-42	214	
87	87		-41	215	
88	88		-40	216	
89	89		-39	217	
90	90		-38	218	
91	91		-37	219	
92	92		-36	220	
93	93		-35	221	
94	94		-34	222	
95	95		-33	223	
96	96		-32	224	
97	97		-31	225	
98	98		-30	226	
99	99	0x63	-29	227	
100	100		-28	228	
101	101		-27	229	
102	102		-26	230	
103	103		-25	231	
104	104		-24	232	

105	105		-23	233
106	106		-22	234
107	107		-21	235
108	108		-20	236
109	109		-19	237
110	110		-18	238
111	111		-17	239
112	112		-16	240
113	113	0x71	-15	241
114	114	0x72	-14	242
115	115	0x73	-13	243
116	116	0x74	-12	244
117	117	0x75	-11	245
118	118	0x76	-10	246
119	119	0x77	-9	247
120	120		-8	248
121	121		-7	249
122	122		-6	250
123	123		-5	251
124	124		-4	252
125	125		-3	253
126	126		-2	254
127	127		-1	255

Vedlegg 3

Testing

KOMMUNIKASJONSTEST FOR JAVA APPLET OG SERVERAPPLIKASJON.

17.1 HENSIKT

Testen er gjort for å kontrollere og bekrefte kommunikasjon mellom Java Applet - til Java server applikasjonen lokalisert på vår stasjonære server.

17.2 TESTBESKRIVELSE

17.2.1.1 TESTOBJEKT

Java Appleten, Java serveren og kommunikasjonen mellom disse.

17.2.1.2 UTSTYR

- Datamaskin med installert Java Runtime Environment.
- Stasjonær server.
- Internettilkobling for begge maskiner.
- Nettleser.
- Sun SPOT baseenhet

17.2.1.3 TESTOPPSETT

Stasjonær server er startet og Java serverapplikasjonen er startet. Serveren er tilkoblet internett. Sun SPOT basestasjon er tilkoblet stasjonær server da dette kreves for å kjøre Java serverapplikasjonen. Klientmaskin er tilkoblet internett, og har åpnet HiBu Bugster nettsiden via nettlesern.

17.2.1.4 UTFØRSEL

Testen er utført på høgskolen, med stasjonær server tilegnet egen ekstern ip-addresse. Klientmaskin er tilkoblet skolens Eduroam nettverk.

17.3 RESULTAT

- Java serveren mottok en lengre bokstavstring, som etter analysering stemmer overens med forventet utfall.

- Java serveren mottok bokstavstringen fra Java Applet med en forsinkelse på ca 3-4 sekunder, etter at den ble sendt.

- Java Appleten fungerer rett og sender korrekt informasjon ved både bruk av nettleserene Internet Explorer og Google Chrome.

17.4 FEILKILDER

- Ujevn kontakt med internettlinjen.
- Feil eller uventede endringer på server/webserver.

17.5 KONKLUSJON

Testen har vært vellykket og sender forventet informasjon, selv med maskiner tilkoblet ulike nettverk.

TESTDOKUMENT: GP2D12

18 HENSIKT

Hensikten med denne testen er å finne karakterestikken til GP2D12 IR sensoren.

19 TESTBESKRIVELSE

19.1 TESTOBJEKT

Sensoren som skal testes heter GP2D12 og er en nærhetssensor. Den gir ut en spenning som varierer avhengig av avstanden til objektet. Rekkevidde er 10-80 cm.

19.2 UTSTYR

- Sharp GP2D12
- Spenningskilde 5V
- Voltmeter
- Målestokk (1m)
- Div utstyr; ledninger, krokodilleklemmer

19.3 TESTOPPSETT

Kobler opp sensoren til en 5 V spenningskilde og et voltmeter på utgangen til sensoren for å se på spenningen.

Måler opp 80 cm med 5 cm intervaller. Bruker en pappeske med flate kanter som objektet som avstanden skal måles til.

19.4 UTFØRSEL

Begynner første måling på 10 cm, øker deretter lengden med 5 cm for hver måling. Gjør dette til jeg når 80 cm. Gjentar prosedyren for å kontrollere målingene.

I tillegg til målinger i rekkevidden til sensoren(10-80 cm) gjør jeg målinger utenfor for å se hva slags spenningsverdier sensoren gir.

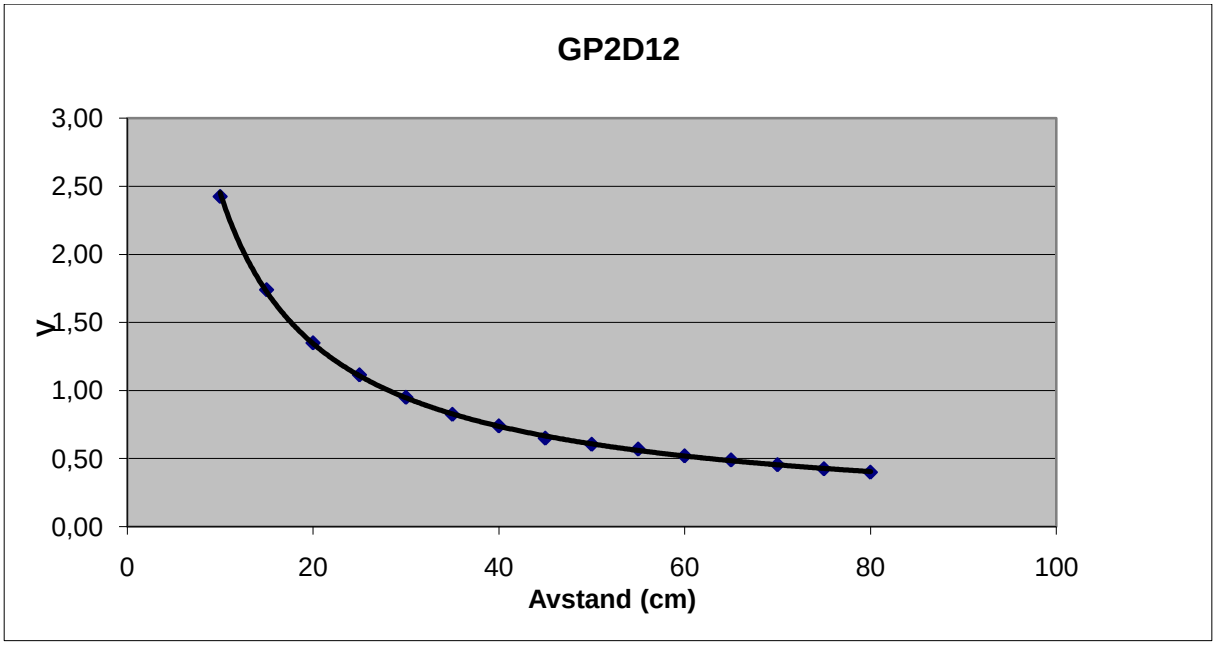
20 RESULTAT

Tabellen under viser resultatene fra de to målingene og gjennomsnittet.

Avstand (cm)	Spenning (V)	2	Gjennomsnitt
10	2,42	2,43	2,43
15	1,74	1,74	1,74
20	1,35	1,35	1,35
25	1,11	1,12	1,12

30	0,95	0,95	0,95
35	0,82	0,83	0,83
40	0,73	0,75	0,74
45	0,63	0,67	0,65
50	0,6	0,61	0,61
55	0,56	0,58	0,57
60	0,52	0,52	0,52
65	0,48	0,5	0,49
70	0,45	0,46	0,46
75	0,42	0,43	0,43
80	0,4	0,4	0,40

Ved noen avstander avviker målingene fra hverandre, det største avviket er på 0,04 V. Dette kan komme av at avstanden sensoren målte ikke var den eksakt samme.



Over er en graf som viser karakterestikken til sensoren. Ser tydelig at sensoren har en ulineær karakterestikk.

Målingene etter 80 cm viser at spenningen går mot 0.02 volt.

Målingene under 10 cm viser en økning opp mot 2.60 V for så å falle under 2 V.

20.1 FEILKILDER

Feilkilder kan være nøyaktigheten til voltmeteret som brukes. Den største feilkilden er plasseriengen av objektet avstanden skal måles til.

21 KONKLUSJON

Karakterestikken til sensoren er ulineær. Det må lages en konverterings funksjon slik at resultatene kan tolkes som cm av mikrokontrolleren.

Det største problemet er karakterestikken til sensoren fra 0-10cm. Den gir samme spenning for to forskjellige avstander. Dette må taes høyde for når mikrokontrolleren skal programmeres.

TESTDOKUMENT: PING ULTRALYD

22 HENSIKT

Hensikten med testen er å finne presisjonen til sensoren og avgjøre om koden som styrer sensoren er god nok.

23 TESTBESKRIVELSE

23.1 TESTOBJEKT

Vi skal teste ultralyd sensoren PING fra parallax. Det er en avstandsmåler med med rekkevidde 2-300 cm. Koden som styrer den skal også testes. For en nærmere beskrivelse av testobjekt og kode se prosjektets hovedrapport.

23.2 UTSTYR

- Parallax PING distance sensor
- Pic18f4520
- PICDDEM 2 plus demo board
- ICD 2
- Osciloskop
- Målestokk (1m)
- Div utstyr; ledninger, krokodilleklemmer...

23.3 TESTOPPSETT

Kobler PING sensoren til mikrokontrolleren(pic18f4520) gjennom pinne RC1. Kobler også oscilloskopet til denne pinnen for å overvåke kommunikasjonen mellom mikrokontroller og sensor.

Merket opp hver 30 cm opptil 300 cm. Satte opp PING sensoren på en eske ca 20 cm over bakken. Objektet PING sensoren skal måle avstanden til er en plastkasse på ca 90*50 cm med en flat og jevn overflate for å få god refleksjon.

23.4 UTFØRSEL

Første avstand som ble målt var 30 cm. Starter målingen med å aktivere koden på mikrokontrolleren fra pc. Ser så på oscilloskopet for å se om alt har foregått som det skal(en beskrivelse av kommunikasjonsprotokollen finnes i hovedrapporten eller i databladet). Får så resultatet fra målingen opp på pc'en.

Gjør tre målinger for hver avstand for å redegjøre for sensoren og kodens evne for gjentakelse. Gjør målinger for hver 30 cm opp til 300 cm. Gjør også målinger utenfor oppgitt rekkevidde for å se hva slags resultater som kommer.

24 RESULTAT

Resultatet fra målingen vises i tabellen under. Avviket ligger mellom 0,1- 1,2 cm ut i fra gjennomsnittet av målingene. Det største avviket for en enkelt måling er på 1,5 cm.

Avstand	målt 1	målt 2	målt 3	Gjennomsnitt	Avvik
30	30,2	30,2	30,1	30,2	0,2
60	60,9	60,7	60,7	60,8	0,8
90	91,5	91,1	91,1	91,2	1,2
120	121,2	120,8	120,8	120,9	0,9
150	151	150,4	150,5	150,6	0,6
180	180,3	180	180,5	180,3	0,3
210	209,9	209,9	210,4	210,1	0,1
240	240,6	239,4	240,4	240,1	0,1
270	270,5	270,7	269,8	270,3	0,3
300	300,5	300,2	299,9	300,2	0,2

Spredningen til målingene ligger alle under 1 cm, hvor den største er på spredningen på en måling er på 0,9 cm. Dette tyder på god gjentakelses evne til PING sensoren og koden.

Vi gjorde også målinger når sensoren ikke hadde noe objekt innen rekkevidde. Så da på oscilloskopet at PING sensoren avsluttet målingen etter ca 20 ms. Dette stemmer bra med databladet, som sier at målingen avsluttes etter 18,5 ms hvis ingen refleksjoner er registrert.

24.1 FEILKILDER

Mulige feilkilder kan være refleksjoner og interferens. Dette grunnet at målingene foregikk nærme en vegg og PING sensoren kun var plassert 20 cm over bordet. Den største feilkilden er nok plasseringen av objektet som avstanden skulle måles til.

25 KONKLUSJON

Testen viser en presisjon som er veldig bra, spesielt når vi ser på hva sensoren skal brukes til. Kan også konkludere med at gjentagelsesevnen er god.

PING sensoren oppfyller kravene til presisjon med god margin. Etter gjentatte forsøk med å aktivere PING sensoren ser vi at koden som styrer den fungerer.

TESTDOKUMENT: POSISJONERINGSSYSTEM

26 HENSIKT

Denne testen skal teste posisjoneringssystemet. Om den klarer å regne ut en posisjon. Den skal også finne presisjonen til avstandsmålingene og selve presisjonen til systemet.

27 TESTBESKRIVELSE

27.1 TESTOBJEKT

Objektet som skal testes er posisjoneringssystemet. Den beregner posisjonen ved å finne avstanden til to faste punkter i rommet ved ultralyd.

27.2 UTSTYR

- Posisjoneringssystem
- Spenningskilde 5V
- Osciloskop
- Målestokk (1m)
- Div utstyr; ledninger, krokodilleklemmer

27.3 TESTOPPSETT

Posisjoneringssystemet bli satt opp som vist på figuren. Oscilloskopet overvåker utgangen til forsterkerkretsen og utgangen til komparatoren.

27.4 UTFØRSEL

1. Tester først presisjonen i avtandsmålingen. Gjør dette ved å ta en avstandsmåling ved flere avstander.
2. Tester så selve posisjoneringssystemet. Dette gjøres ved å plassere mottakerenheten i kjente punkter og så kjøre en posisjonerings oppdatering.

28 RESULTAT

28.1 TEST 1

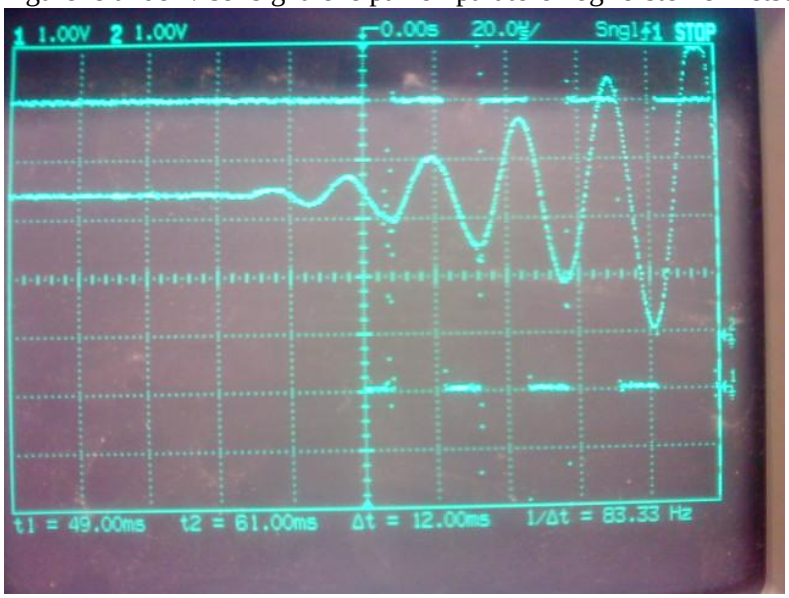
Avstand	Målt
20	29
40	43
60	64
80	79
100	104

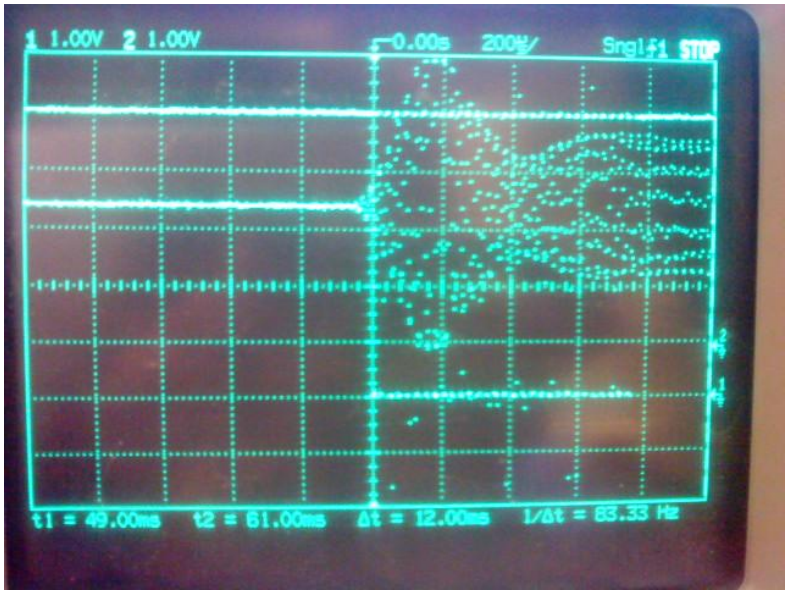
140	143
180	184

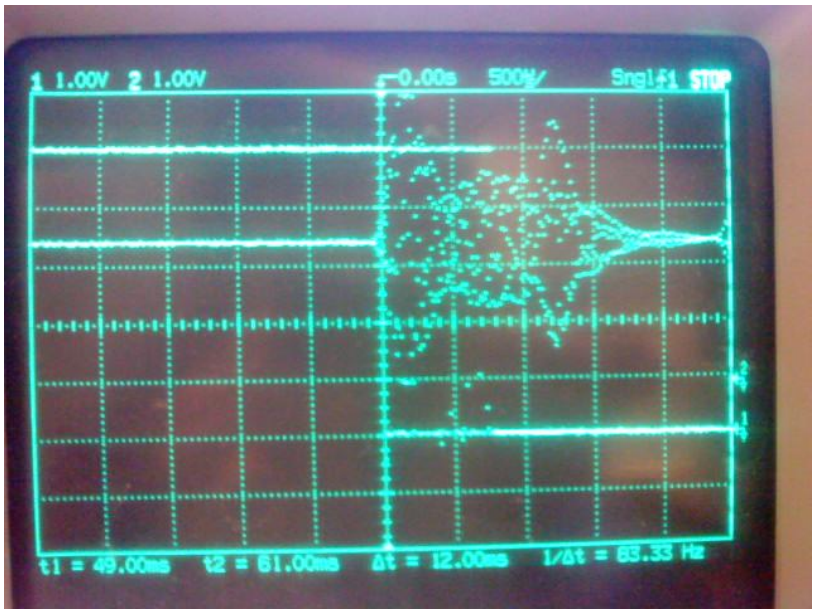
28.2 TEST 2

Posisjon		Målt	
x	y	x	y
140	35	141	34
140	50	140	53
120	50	123	49
120	20	125	21
100	35	107	33

Figurene under viser signalene på komparatoren og forsterkerkretsen.







29

30 KONKLUSJON

Ser på resultatene fra testene at presisjonen for systemet ligger på rundt 6 cm. Ser også på bildene fra oscilloskopet at det er mye refleksjoner.

TESTING AV KOMPLETT LADESYSTEM

30.1.1.1 HENSIKT

Kontrollere at vi får ladet batteriet på en tilfredstillende måte.

30.1.1.2 TESTBESKRIVELSE

30.1.1.2.1 TESTOBJEKT

Kretsen som skal tilføre og kontrollere strømmen inn til –og spenningen over batteriet.

30.1.1.2.2 UTSTYR

- Batterilader, Texas Instruments UC2906
- PNP transistor, Power Innovations BD896
- Motstander, 6 ohm, 470 ohm, 220 ohm, 390k, 43k, 18k og 70k
- Kondensator, 0,1uF
- Diode, BY251
- Blyakkumulator, Panasonic LC-R064R5P
- Spenningskilde, koblingsbrett, ledninger og digitale multimetre

30.1.1.2.3 TESTOPPSETT

Koblet opp ladekretsen på et test brett. Koblet så inngang til en spenningskilde og utgang til batteriet. Ladestasjonen vil ha en spenning på 12V. Vi brukte digitale multimetre til å måle strøm og spenning i forskjellige deler av kretsen.

30.1.1.2.4 UTFØRSEL

Testen er utført på lab.

30.1.1.3 RESULTAT

- Laderen tilfører maksimal strøm til batteriet, 1.8A.
- Spenningen over batteriet når 7.4 V, spenningen minker kraftig ned til 0.3 A.
- Spenningen holdes konstant og strømtilførselen stopper. Batteriet er fulladet.

30.1.1.4 FEILKILDER

Unøyaktighet i måleinstrumentene

30.1.1.5 KONKLUSJON

Testen var vellykket. Vi fikk de verdiene vi forventet.

TESTING AV APPLIKASJON TIL SUN SPOT

UART
For å teste UART kommunikasjonen mellom sun SPOT og mikrokontroller, ble det laget en applikasjon som kun sendte et hexadesimalt tall samt en kontrollutskift som ble skrevet ut etter at tallet var sendt. Ved å la sun SPOT enheten være koblet til datamaskinen mens vi testet, kunne vi lese av utskriften der. Det ble også laget en test applikasjon som først skulle sende ett tall, for deretter vente på å motta ett nytt tall fra MC. Dette skulle tallet skulle da skrives ut på skjermen.

Da vi hadde noe problem med kommunikasjonen mellom sun SPOT og MC, prøve vi å kjøre samme test mellom to sun SPOT enheter. SPOT1 var da som vanlig koblet til en datamaskin for kontrollere utskriften samtidig som den var koblet til SPOT2 via RX,TX og GRD. SPOT2 var kun koblet til SPOT1. Denne testen gikk ut på at SPOT 1 sendte en verdi X til SPOT2 som da, etter å ha mottatt X, returnerer X sammen med en ny verdi Y. SPOT1 skulle da skrive ut både verdien X og Y.
Dette fungerte som det skulle, ergo var det ikke selve applikasjonen det var noe galt med.

Testing av ulike verdier.
Etter å ha fått i gang kommunikasjonen mellom sun SPOT og MC begynte vi å sende ulike verdier. Flere verdier ble sendt etter hverandre med en liten pause i mellom for å starte og stoppe motoren.

I denne testen fant vi ut at alle byte verdier sendt til sun SPOT enheten som var høyere en 127 ble erstattet med ett negativt tall. I Java så dekker en byte tallene fra -128 til 127. Ikke fra 0-255 som vi først trodde. For å få oversikt på alle verdiene laget vi en ny liste slik at vi raskt og enkelt kunne bla oss frem i den.

Tolking av verdi.
Her hadde vi utvidet applikasjonen slik at to tråder jobbet samtidig. Den ene leste av verdier fra mikrokontrolleren og lagret disse i ett buffer mens den andre hentet verdiene fra bufferet, kontrollerte de og ut i fra resultatene sendte kommandoer tilbake til mikrokontrolleren. Dette gjorde det mulig å teste IR sensorene.Vi testet på en IR sensor av gangen. Først ble kjørestatus kontrollert, der etter IR sensoren verdien. Motoren skulle da stoppet om IR sensoren hadde en lavere verdi en 0x42. Var verdien høyre en 0x42 skulle motoren kjøre. Gjennom denne testen lot vi applikasjonen skrive ut alle verdiene mottatt fra mikrokontrolleren for å kontrollere de opp mot hvordan motoren ble styrt.

Samme testen gjorde vi da vi skulle teste støtsensorene. Vi koblet først til en sensor, for der etter å legge til alle fire. Det ble gjort ulike tester, blant annet å trykke inne flere sensorer samtidig, for så sleppe den ene. Programmet skulle hele tiden ha kontroll på alle støt sensorene, Så da fikk vi test at motoren aldri begynne å kjøre så lenge en eller flere sensorer var trykt inn.

En ny test ble utført der en IR sensorene og alle støtsensorene ble tatt i bruk. Motoren skulle stoppe og starte som i de to forrige testene. Vi sjekka at programmet håndterte at to ulike sensorer ble registrerte ting samtidig, og at ikke motoren begynte å kjøre hvis bare den ene hindringen ble fjernet.

Applikasjonen ble utvidet ti tre tråder.
Der den tredje tråden skulle kontrollere at kjøreruten ble utført. Vi testet at denne tråden fortsatte arbeidet sitt selv etter å ha blitt avbrutt. Noe den gjorde, men den kunne miste ett halvt sekund. Den kjørte kun i 1,5 sekund i stede for 2. Dette kunne unngås hadde vi sjekket kjørestatusen oftere.

Testing med IR sensor.
Applikasjonen ble utvidet til å ta i mot og tolke verdiene fra en IR sensor. Hver gang IR sensoren hadde en verdi som var lavere en 67, skulle sun SPOT enheten sende en stopp verdi til MC. Dette fungerte utmerket. Men å utvide til at den også skulle sende en start verdi når verdien ble over 67 igjen fungerte ikke fult så bra. Motoren stoppet, men ville ikke starte igjen.
Det viste seg at da applikasjonen sendte en stopp kommando, gikk den inn i en loop og sendte ut stoppverdier helt til den fikk en ny beskjed fra MC. MC ble overlesset med så mange stopp kommandoer at den aldri fikk mulighet til å sende noe tilbake. Dette løste seg med å hindre sun SPOT til å sende den samme verdien mer enn en gang.

Testing av Støtsensor
En tilsvarende applikasjon som IR sensorene ble laget for støtsensorene, men byte verdiene ble byttet ut med boolean verdier.

Vedlegg 4

Kildekode - Applet

31 KILDEKODE FOR JAVA APPLET

AppInterface-KLASSEN

```
import javax.imageio.ImageIO;
import javax.swing.JApplet;
import javax.swing.JDesktopPane;
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import javax.swing.JButton;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import javax.swing.SwingUtilities;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JTextField;
import javax.swing.UIManager;
import java.awt.Color;
import java.awt.Font;
import javax.swing.JProgressBar;
import javax.swing.JSeparator;
import javax.swing.JToggleButton;
import javax.swing.JTextArea;
import javax.swing.JComboBox;
import javax.swing.JRadioButton;


public class AppInterface extends JApplet {
	JPanel		topPanel, subPanel, buttonPanel, frontPanel, endPanel,
			listPanel1, listPanel2, listPanel3, listPanel4,
listPanel5;
	JButton leftButton, stopButton, rightButton, forwardButton,
reverseButton,
			deleteButton, previousButton, nextButton, sendButton;
	JLabel		processLabel, lblVelgHvilkenRobot,
			lblHibuBugster, lblProgrammeringspanel,
			headlabel2, headlabel1, headlabel3, headlabel4,
headlabel5;
	JTextArea textArea, textArea1, textArea2, textArea3, textArea4;
	JComboBox comboBox, comboBox_2, comboBox_3, comboBox_4, comboBox_5;
	JToggleButton robotChoiceBtn1;
	JProgressBar progressBar;
	JSeparator separator, separator_2;


	// Numerical values for the Bugster commands.
	// ------------------------------------------
	String ryggkommando = "C";
	String kjørkommando = "A";
	String høyrekommando = "E";
	String stoppkommando = "B";
	String venstrekommando = "D";
	String string;
	String commandline;
```

```
	String[] comboString = {"Hopp over", "Reaksjon 10cm ", "Reaksjon
20cm", "Reaksjon 30cm"};

	ArrayList<String> array;
	ArrayList<String> array_1;
	ArrayList<String> array_2;
	ArrayList<String> array_3;
	ArrayList<String> array_4;

	Client socket;  // Reference to the socket-class

	int count = 1;  //
	int count1 = 1;
	int count2 = 1;
	int count3 = 1;
	int count4 = 1;


	// Creating the UserInterface with the ActionEvents
	// ---------------------------
	public AppInterface() {
		getContentPane().setBackground(Color.DARK_GRAY);

		getContentPane().setLayout(null);
		array = new ArrayList<String>();
		array_1 = new ArrayList<String>();
		array_2 = new ArrayList<String>();
		array_3 = new ArrayList<String>();
		array_4 = new ArrayList<String>();

		frontPanel = new JPanel();
		frontPanel.setBounds(0, 98, 650, 343);
		getContentPane().add(frontPanel);
		frontPanel.setBackground(Color.LIGHT_GRAY);
		frontPanel.setLayout(null);

		lblVelgHvilkenRobot = new JLabel("Velg hvilken robot du vil
styre?");
		lblVelgHvilkenRobot.setBounds(199, 59, 254, 14);
		lblVelgHvilkenRobot.setFont(new Font("Rockwell Extra Bold",
Font.PLAIN, 13));
		lblVelgHvilkenRobot.setForeground(Color.BLACK);
		frontPanel.add(lblVelgHvilkenRobot);

		robotChoiceBtn1 = new JToggleButton("Prototype");
		robotChoiceBtn1.setBounds(239, 114, 139, 87);
		frontPanel.add(robotChoiceBtn1);

		listPanel1 = new JPanel();
		listPanel1.setBackground(Color.LIGHT_GRAY);
		listPanel1.setBounds(0, 98, 165, 343);
		getContentPane().add(listPanel1);
		listPanel1.setLayout(null);

		textArea = new JTextArea();
		textArea.setEditable(false);
		textArea.setBounds(6, 52, 134, 246);
		listPanel1.add(textArea);

		listPanel2 = new JPanel();
		listPanel2.setBackground(Color.LIGHT_GRAY);
```

```
listPanel2.setBounds(0, 98, 165, 343);
getContentPane().add(listPanel2);
listPanel2.setLayout(null);
listPanel2.setVisible(false);

comboBox = new JComboBox(comboString);
comboBox.setBounds(6, 21, 134, 29);
listPanel2.add(comboBox);
comboBox.setSelectedIndex(0);

textArea1 = new JTextArea();
textArea1.setEditable(false);
textArea1.setBounds(6, 52, 134, 246);
listPanel2.add(textArea1);

deleteButton = new JButton("Slett");
deleteButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
            textArea1.setText("");
            count1 = 1;
            deleteArray();
      }
});

deleteButton.setBounds(6, 310, 134, 23);
listPanel2.add(deleteButton);

listPanel3 = new JPanel();
listPanel3.setBackground(Color.LIGHT_GRAY);
listPanel3.setBounds(0, 98, 165, 343);
getContentPane().add(listPanel3);
listPanel3.setLayout(null);

comboBox_2 = new JComboBox(comboString);
comboBox_2.setSelectedIndex(0);
comboBox_2.setBounds(6, 21, 134, 29);
listPanel3.add(comboBox_2);

textArea2 = new JTextArea();
textArea2.setEditable(false);
textArea2.setBounds(6, 52, 134, 246);
listPanel3.add(textArea2);

listPanel4 = new JPanel();
listPanel4.setBackground(Color.LIGHT_GRAY);
listPanel4.setBounds(0, 98, 165, 343);
getContentPane().add(listPanel4);
listPanel4.setLayout(null);

comboBox_3 = new JComboBox(comboString);
comboBox_3.setSelectedIndex(0);
comboBox_3.setBounds(6, 21, 134, 29);
listPanel4.add(comboBox_3);

textArea3 = new JTextArea();
textArea3.setEditable(false);
textArea3.setBounds(6, 52, 134, 246);
listPanel4.add(textArea3);

listPanel5 = new JPanel();
listPanel5.setBackground(Color.LIGHT_GRAY);
```

```
            listPanel5.setBounds(0, 98, 165, 343);
            getContentPane().add(listPanel5);
            listPanel5.setLayout(null);

            comboBox_4 = new JComboBox(comboString);
            comboBox_4.setSelectedIndex(0);
            comboBox_4.setBounds(6, 21, 134, 29);
            listPanel5.add(comboBox_4);

            textArea4 = new JTextArea();
            textArea4.setEditable(false);
            textArea4.setBounds(6, 52, 134, 246);
            listPanel5.add(textArea4);

            topPanel = new JPanel();
            topPanel.setBackground(Color.LIGHT_GRAY);
            topPanel.setBounds(0, 0, 650, 98);
            getContentPane().add(topPanel);
            topPanel.setLayout(null);

            endPanel = new JPanel();
            endPanel.setBounds(0, 99, 650, 343);
            getContentPane().add(endPanel);
            endPanel.setBackground(Color.LIGHT_GRAY);
            endPanel.setLayout(null);
            endPanel.setVisible(false);

            buttonPanel = new JPanel();
            buttonPanel.setBackground(Color.LIGHT_GRAY);
            buttonPanel.setBounds(165, 98, 485, 343);
            getContentPane().add(buttonPanel);
            buttonPanel.setVisible(false);
            buttonPanel.setLayout(null);

            // Creating all the different buttons, and assigning some
Events to them.
            // ------------------------------------------------------------
----------
            leftButton = new JButton("VENSTRE");
            leftButton.setBounds(173, 127, 85, 70);
            buttonPanel.add(leftButton);
            leftButton.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {

                        if(listPanel1.isVisible())
                        {
                              setTextArea("Svinge til venstre", 1);
                              addToArray(venstrekommando, 1);
                        }
                        else if(listPanel2.isVisible())
                        {
                              setTextArea("Svinge til venstre", 2);
                              addToArray(venstrekommando, 2);
                        }
                        else if(listPanel3.isVisible())
                        {
                              setTextArea("Svinge til venstre", 3);
                              addToArray(venstrekommando, 3);
                        }
                        else if(listPanel4.isVisible())
                        {
```

```
                        setTextArea("Svinge til venstre", 4);
                        addToArray(venstrekommando, 4);
                }
                else if(listPanel5.isVisible())
                {
                        setTextArea("Svinge til venstre", 5);
                        addToArray(venstrekommando, 5);
                }
        }
});

rightButton = new JButton("H\u00D8YRE");
rightButton.setBounds(367, 127, 85, 70);
buttonPanel.add(rightButton);
rightButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                if(listPanel1.isVisible())
                {
                        setTextArea("Svinge til høyre", 1);
                        addToArray(høyrekommando, 1);
                }
                else if(listPanel2.isVisible())
                {
                        setTextArea("Svinge til høyre", 2);
                        addToArray(høyrekommando, 2);
                }
                else if(listPanel3.isVisible())
                {
                        setTextArea("Svinge til høyre", 3);
                        addToArray(høyrekommando, 3);
                }
                else if(listPanel4.isVisible())
                {
                        setTextArea("Svinge til høyre", 4);
                        addToArray(høyrekommando, 4);
                }
                else if(listPanel5.isVisible())
                {
                        setTextArea("Svinge til høyre", 5);
                        addToArray(høyrekommando, 5);
                }
        }
});

forwardButton = new JButton("FREM");
forwardButton.setBounds(270, 50, 85, 70);
buttonPanel.add(forwardButton);
forwardButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                if(listPanel1.isVisible())
                {
                        setTextArea("Kjøre frem", 1);
                        addToArray(kjørkommando, 1);
                }
                else if(listPanel2.isVisible())
                {
                        setTextArea("Kjøre frem", 2);
                        addToArray(kjørkommando, 2);
                }
                else if(listPanel3.isVisible())
                {
```

```
                        setTextArea("Kjøre frem", 3);
                        addToArray(kjørkommando, 3);
                }
                else if(listPanel4.isVisible())
                {
                        setTextArea("Kjøre frem", 4);
                        addToArray(kjørkommando, 4);
                }
                else if(listPanel5.isVisible())
                {
                        setTextArea("Kjøre frem", 5);
                        addToArray(kjørkommando, 5);
                }
        }
});

reverseButton = new JButton("RYGG");
reverseButton.setBounds(270, 203, 85, 70);
buttonPanel.add(reverseButton);
reverseButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                if(listPanel1.isVisible())
                {
                        setTextArea("Rygge", 1);
                        addToArray(ryggkommando, 1);
                }
                else if(listPanel2.isVisible())
                {
                        setTextArea("Rygge", 2);
                        addToArray(ryggkommando, 2);
                }
                else if(listPanel3.isVisible())
                {
                        setTextArea("Rygge", 3);
                        addToArray(ryggkommando, 3);
                }
                else if(listPanel4.isVisible())
                {
                        setTextArea("Rygge", 4);
                        addToArray(ryggkommando, 4);
                }
                else if(listPanel5.isVisible())
                {
                        setTextArea("Rygge", 5);
                        addToArray(ryggkommando, 5);
                }
        }
});
stopButton = new JButton("STOPP");
stopButton.setBounds(270, 127, 85, 70);
buttonPanel.add(stopButton);
stopButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                if(listPanel1.isVisible())
                {
                        setTextArea("Stopp", 1);
                        addToArray(stoppkommando, 1);
                }
                else if(listPanel2.isVisible())
                {
                        setTextArea("Stopp", 2);
```

```
                        addToArray(stoppkommando, 2);
                    }
                    else if(listPanel3.isVisible())
                    {
                        setTextArea("Stopp", 3);
                        addToArray(stoppkommando, 3);
                    }
                    else if(listPanel4.isVisible())
                    {
                        setTextArea("Stopp", 4);
                        addToArray(stoppkommando, 4);
                    }
                    else if(listPanel5.isVisible())
                    {
                        setTextArea("Stopp", 5);
                        addToArray(stoppkommando, 5);
                    }
                }
            });

            sendButton = new JButton("S E N D");
            sendButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    SwingUtilities.invokeLater(
                                new Runnable()
                                {
                                    public void run()
                                    {
                                                convertArray();
                                    }
                                });
                }
            });
            sendButton.setBounds(233, 144, 148, 72);
            endPanel.add(     sendButton);

            subPanel = new JPanel();
            subPanel.setBackground(Color.LIGHT_GRAY);
            subPanel.setBounds(0, 440, 650, 60);
            getContentPane().add(subPanel);
            subPanel.setLayout(null);

            processLabel = new JLabel("Prosess: ");
            processLabel.setFont(new Font("Tahoma", Font.PLAIN, 12));
            processLabel.setForeground(Color.BLACK);
            processLabel.setBounds(10, 29, 55, 14);
            subPanel.add(processLabel);

            progressBar = new JProgressBar(0,6);
            progressBar.setBounds(63, 26, 253, 23);
            subPanel.add(progressBar);

            previousButton = new JButton("Tilbake");
            previousButton.setEnabled(false);
            previousButton.addActionListener(new ActionListener() {

                // Management of the panels, related to the PREVIOUS-
button.
                // -------------------------------------------------------
---
                public void actionPerformed(ActionEvent e) {
```

```
if(listPanel1.isVisible())
{
      frontPanel.setVisible(true);
      buttonPanel.setVisible(false);
      listPanel1.setVisible(false);
      listPanel2.setVisible(false);
      listPanel3.setVisible(false);
      listPanel4.setVisible(false);
      listPanel5.setVisible(false);
      endPanel.setVisible(false);
      previousButton.setEnabled(false);
      progressBar.setValue(0);
      headlabel1.setVisible(false);
}
else if(listPanel2.isVisible())
{
      frontPanel.setVisible(false);
      listPanel1.setVisible(true);
      listPanel2.setVisible(false);
      listPanel3.setVisible(false);
      listPanel4.setVisible(false);
      listPanel5.setVisible(false);
      endPanel.setVisible(false);
      progressBar.setValue(1);
      headlabel1.setVisible(true);
      headlabel2.setVisible(false);
}
else if( listPanel3.isVisible())
{
      frontPanel.setVisible(false);
      listPanel1.setVisible(false);
      listPanel2.setVisible(true);
      listPanel3.setVisible(false);
      listPanel4.setVisible(false);
      listPanel5.setVisible(false);
      endPanel.setVisible(false);
      progressBar.setValue(2);
      headlabel2.setVisible(true);
      headlabel3.setVisible(false);
}
else if( listPanel4.isVisible())
{
      frontPanel.setVisible(false);
      listPanel1.setVisible(false);
      listPanel2.setVisible(false);
      listPanel3.setVisible(true);
      listPanel4.setVisible(false);
      listPanel5.setVisible(false);
      endPanel.setVisible(false);
      progressBar.setValue(3);
      headlabel3.setVisible(true);
      headlabel4.setVisible(false);
}
else if( listPanel5.isVisible())
{
      frontPanel.setVisible(false);
      listPanel1.setVisible(false);
      listPanel2.setVisible(false);
      listPanel3.setVisible(false);
```

```
                        listPanel4.setVisible(true);
                        listPanel5.setVisible(false);
                        endPanel.setVisible(false);
                        progressBar.setValue(4);
                        headlabel4.setVisible(true);
                        headlabel5.setVisible(false);
                  }

                  else if(endPanel.isVisible())
                  {
                        frontPanel.setVisible(false);
                        listPanel1.setVisible(false);
                        listPanel2.setVisible(false);
                        listPanel3.setVisible(false);
                        listPanel4.setVisible(false);
                        listPanel5.setVisible(true);
                        buttonPanel.setVisible(true);
                        endPanel.setVisible(false);
                        progressBar.setValue(5);
                        nextButton.setEnabled(true);
                        headlabel5.setVisible(true);
                  }
            }
      });
      previousButton.setBounds(441, 26, 89, 23);
      subPanel.add(previousButton);

      nextButton = new JButton("Neste");
      nextButton.addActionListener(new ActionListener() {

            // Management of the panels, related to the NEXT-button.
            // -----------------------------------------------------
            public void actionPerformed(ActionEvent e)
            {
                  if(robotChoiceBtn1.isSelected())
                  {
                        if(frontPanel.isVisible())
                        {
                              buttonPanel.setVisible(true);
                              frontPanel.setVisible(false);
                              listPanel1.setVisible(true);
                              listPanel2.setVisible(false);
                              listPanel3.setVisible(false);
                              listPanel4.setVisible(false);
                              listPanel5.setVisible(false);
                              endPanel.setVisible(false);
                              headlabel1.setVisible(true);
                              previousButton.setEnabled(true);
                              progressBar.setValue(1);
                        }
                        else if ( listPanel1.isVisible())
                        {
                              frontPanel.setVisible(false);
                              listPanel1.setVisible(false);
                              listPanel2.setVisible(true);
                              listPanel3.setVisible(false);
                              listPanel4.setVisible(false);
                              listPanel5.setVisible(false);
                              endPanel.setVisible(false);
                              headlabel2.setVisible(true);
                              headlabel1.setVisible(false);
```

```
                    progressBar.setValue(2);

                }
                else if( listPanel2.isVisible())
                {
                    frontPanel.setVisible(false);
                    listPanel1.setVisible(false);
                    listPanel2.setVisible(false);
                    listPanel3.setVisible(true);
                    listPanel4.setVisible(false);
                    listPanel5.setVisible(false);
                    endPanel.setVisible(false);
                    headlabel3.setVisible(true);
                    headlabel2.setVisible(false);
                    progressBar.setValue(3);

                }
                else if( listPanel3.isVisible())
                {
                    frontPanel.setVisible(false);
                    listPanel1.setVisible(false);
                    listPanel2.setVisible(false);
                    listPanel3.setVisible(false);
                    listPanel4.setVisible(true);
                    listPanel5.setVisible(false);
                    endPanel.setVisible(false);
                    headlabel4.setVisible(true);
                    headlabel3.setVisible(false);
                    progressBar.setValue(4);
                }
                else if( listPanel4.isVisible())
                {
                    frontPanel.setVisible(false);
                    listPanel1.setVisible(false);
                    listPanel2.setVisible(false);
                    listPanel3.setVisible(false);
                    listPanel4.setVisible(false);
                    listPanel5.setVisible(true);
                    endPanel.setVisible(false);
                    headlabel5.setVisible(true);
                    headlabel4.setVisible(false);
                    progressBar.setValue(5);
                }
                else if( listPanel5.isVisible())
                {
                    frontPanel.setVisible(false);
                    listPanel1.setVisible(false);
                    listPanel2.setVisible(false);
                    listPanel3.setVisible(false);
                    listPanel4.setVisible(false);
                    listPanel5.setVisible(false);
                    buttonPanel.setVisible(false);
                    endPanel.setVisible(true);
                    nextButton.setEnabled(false);
                    headlabel5.setVisible(false);
                    progressBar.setValue(6);
                }

                }
            }
        });
```

```
                    nextButton.setBounds(540, 26, 100, 23);
                    subPanel.add(nextButton);

                    separator = new JSeparator();
                    separator.setBounds(10, 11, 630, 2);
                    subPanel.add(separator);

                    lblHibuBugster = new JLabel("HiBu Bugster");
                    lblHibuBugster.setForeground(Color.BLACK);
                    lblHibuBugster.setFont(new Font("Rockwell Extra Bold",
Font.PLAIN, 14));
                    lblHibuBugster.setBounds(10, 11, 123, 19);
                    topPanel.add(lblHibuBugster);

                    lblProgrammeringspanel = new
JLabel("Programmeringspanel");
                    lblProgrammeringspanel.setFont(new Font("Rockwell Extra
Bold", Font.PLAIN, 11));
                    lblProgrammeringspanel.setForeground(Color.BLACK);
                    lblProgrammeringspanel.setBounds(10, 33, 169, 14);
                    topPanel.add(lblProgrammeringspanel);

                    separator_2 = new JSeparator();
                    separator_2.setBounds(10, 56, 630, 2);
                    topPanel.add(separator_2);

                    headlabel1 = new JLabel("Bruk knappene for \u00E5
bestemme hva Bugster skal gj\u00F8re");
                    headlabel1.setBounds(98, 66, 442, 14);
                    topPanel.add(headlabel1);
                    headlabel1.setFont(new Font("Rockwell Extra Bold",
Font.PLAIN, 12));
                    headlabel1.setForeground(Color.BLACK);
                    headlabel1.setVisible(false);

                    headlabel2 = new JLabel("Hva vil du Bugster skal
gj\u00F8re hvis noe blokkerer i front ? ");
                    headlabel2.setBounds(98, 65, 494, 18);
                    topPanel.add(headlabel2);
                    headlabel2.setForeground(Color.BLACK);
                    headlabel2.setFont(new Font("Rockwell Extra Bold",
Font.PLAIN, 12));
                    headlabel2.setVisible(false);

                    headlabel3 = new JLabel("Hva vil du Bugster skal
gj\u00F8re hvis noe blokkerer bak ?");
                    headlabel3.setForeground(Color.BLACK);
                    headlabel3.setFont(new Font("Rockwell Extra Bold",
Font.PLAIN, 12));
                    headlabel3.setBounds(98, 68, 494, 15);
                    topPanel.add(headlabel3);
                    headlabel3.setVisible(false);

                    headlabel4 = new JLabel("Hva vil du Bugster skal
gj\u00F8re hvis noe blokkerer til venstre ?");
                    headlabel4.setForeground(Color.BLACK);
                    headlabel4.setFont(new Font("Rockwell Extra Bold",
Font.PLAIN, 12));
                    headlabel4.setBounds(98, 68, 494, 15);
                    topPanel.add(headlabel4);
                    headlabel4.setVisible(false);
```

```
                headlabel5 = new JLabel("Hva vil du Bugster skal
gj\u00F8re hvis noe blokkerer til h\u00F8yre ?");
                headlabel5.setForeground(Color.BLACK);
                headlabel5.setFont(new Font("Rockwell Extra Bold",
Font.PLAIN, 12));
                headlabel5.setBounds(98, 68, 494, 15);
                headlabel5.setVisible(false);
                topPanel.add(headlabel5);

            }

            // Method that adds Messages into the textarea.
            public void setTextArea(String dispMessage, int textAreaNr)
            {
                if(textAreaNr == 1)
                {
                    textArea.append(count + ": " + dispMessage + "\n");
                    count++;
                }
                else if(textAreaNr == 2)
                {
                    textArea1.append(count1 + ": " + dispMessage +
"\n");
                    count1++;
                }
                else if(textAreaNr == 3)
                {
                    textArea2.append(count2 + ": " + dispMessage +
"\n");
                    count2++;
                }
                else if(textAreaNr == 4)
                {
                    textArea3.append(count3 + ": " + dispMessage +
"\n");
                    count3++;
                }
                else if(textAreaNr == 5)
                {
                    textArea4.append(count4 + ": " + dispMessage +
"\n");
                    count4++;
                }
            }


            // Method that add the command into an ArrayList.
            public void addToArray(String insertToArray, int arrayNr)
            {
                if( arrayNr == 1)
                {
                    array.add(insertToArray);
                    System.out.println(array);
                }
                else if( arrayNr == 2)
                {
                    array_1.add(insertToArray);
                    System.out.println(array_1);
                }
                else if( arrayNr == 3)
```

```
                {
                        array_2.add(insertToArray);
                        System.out.println(array_2);
                }
                else if( arrayNr == 4)
                {
                        array_3.add(insertToArray);
                        System.out.println(array_3);
                }
                else if( arrayNr == 5)
                {
                        array_4.add(insertToArray);
                        System.out.println(array_4);
                }
        }

        // Method that empties the ArrayList.
        public void deleteArray()
        {
                if(listPanel1.isVisible())
                {
                        array.clear();
                }
                else if(listPanel2.isVisible())
                {
                        array_1.clear();
                }
                else if(listPanel3.isVisible())
                {
                        array_2.clear();
                }
                else if(listPanel4.isVisible())
                {
                        array_3.clear();
                }
                else if(listPanel5.isVisible())
                {
                        array_4.clear();
                }
        }

        // Method that converts the ArrayList to a String, and starts
the Client class.
        public void convertArray()
        {
                SwingUtilities.invokeLater(
                                new Runnable()
                                {
                                        public void run()
                                        {
                                                checkComboBoxes();
                                                initSocket(string);
                                        }
                                });

        }

        // Method that checks the ComboBox-values that the user have
chosen.
        public void checkComboBoxes()
        {
```

```
int content1 = comboBox.getSelectedIndex();
int content3 = comboBox_2.getSelectedIndex();
int content4 = comboBox_3.getSelectedIndex();
int content5 = comboBox_4.getSelectedIndex();

if (content1 != 0)
{
      if(content1 == 1)
      {
            array_1.add(0, "F");
      }
      else if(content1 == 2 )
      {
            array_1.add(0, "G");
      }
      else if(content1 == 3)
      {
            array_1.add(0, "H");
      }
}
else if (content3 != 0)
{
      if(content3 == 1)
      {
            array_2.add(0, "I");
      }
      else if(content3 == 2 )
      {
            array_2.add(0, "J");
      }
      else if(content3 == 3)
      {
            array_2.add(0, "K");
      }
}
else if (content4 != 0)
{
      if(content4 == 1)
      {
            array_2.add(0, "L");
      }
      else if(content4 == 2 )
      {
            array_2.add(0, "M");
      }
      else if(content4 == 3)
      {
            array_2.add(0, "N");
      }
}
else if (content5 != 0)
{
      if(content5 == 1)
      {
            array_2.add(0, "O");
      }
      else if(content5 == 2 )
      {
            array_2.add(0, "P");
      }
      else if(content5 == 3)
```

```
                        {
                                array_2.add(0, "Q");
                        }
                }

                // Converts the arrays into strings, and remove the
start- and end-brackets.
                String str = array.toString();
                String fin = str.substring(1,str.length()-1);
                String str1 = array_1.toString();
                String fin1 = str1.substring(1,str1.length()-1);
                String str2 = array_2.toString();
                String fin2 = str2.substring(1,str2.length()-1);
                String str3 = array_3.toString();
                String fin3 = str3.substring(1,str3.length()-1);
                String str4 = array_4.toString();
                String fin4 = str4.substring(1,str4.length()-1);

                // Adding all the string pieces into one huge string.
                string = fin + "2" + fin1 + "3" + fin2 + "4" + fin3 + "5"
+ fin4 + "6";
                System.out.println(string);
            }

            // Starts up the socketclass, only if the busySock is false.
            public void initSocket(String commandline)
            {
                        socket = new Client();
                        socket.sendList(commandline);

                        System.out.println(commandline);


            }
        }

import java.io.EOFException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.Socket;

import javax.swing.SwingUtilities;


public class Client{

        AppInterface Appref;
        Socket socketclient;


        String ip = "158.36.70.52";
        int host = 9001;
        ObjectOutputStream output;
        ObjectInputStream input;
        String a;
        String list = "";
        public Client()
        {
```

```
            System.out.println("SystemCheck: SysCheck socketclass
initiated!");

                        System.out.println("SysCheck: run start");
                        try
                        {
                              socketclient = new Socket(ip, host);
                              output = new
ObjectOutputStream(socketclient.getOutputStream());

                              output.flush();
                              input = new
ObjectInputStream(socketclient.getInputStream());
                              System.out.println("SystemCheck: Socket OK!");
                        }
                        catch(EOFException eofException)
                        {
                              System.out.println("SysCheck: eof");
                        }
                        catch(IOException ioException)
                        {
                              System.out.println("SysCheck: io");
                              ioException.printStackTrace();
                        }
                        System.out.println("SysCheck: run finish");


      }
      public void sendList(String list)
      {
            this.list = list;
            System.out.println("SystemCheck: skriver ut liste");
            System.out.println("SystemCheck: skriver ut liste" + list);

            try
            {     output.writeObject(list);
                  output.flush();
                  System.out.println("SystemCheck: SendList completed");
                  list = "";
            }
            catch(IOException ioException)
            {
                  ioException.printStackTrace();
            }
      }
}
```

Client-KLASSEN:

```
import java.io.EOFException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.Socket;

import javax.swing.SwingUtilities;


public class Client{
```

```
	AppInterface Appref;
	Socket socketclient;


	String ip = "158.36.70.52";
	int host = 9001;
	ObjectOutputStream output;
	ObjectInputStream input;
	String a;
	String list = "";
	public Client()
	{

		System.out.println("SystemCheck: SysCheck socketclass
initiated!");

			System.out.println("SysCheck: run start");
			try
			{
				socketclient = new Socket(ip, host);
				output = new
ObjectOutputStream(socketclient.getOutputStream());

				output.flush();
				input = new
ObjectInputStream(socketclient.getInputStream());
				System.out.println("SystemCheck: Socket OK!");
			}
			catch(EOFException eofException)
			{
				System.out.println("SysCheck: eof");
			}
			catch(IOException ioException)
			{
				System.out.println("SysCheck: io");
				ioException.printStackTrace();
			}
			System.out.println("SysCheck: run finish");


	}
	public void sendList(String list)
	{
		this.list = list;
		System.out.println("SystemCheck: skriver ut liste");
		System.out.println("SystemCheck: skriver ut liste" + list);

		try
		{	output.writeObject(list);
			output.flush();
			System.out.println("SystemCheck: SendList completed");
			list = "";
		}
		catch(IOException ioException)
		{
			ioException.printStackTrace();
		}
	}
}
```

Vedlegg 5

Kildekode - Server

32 KILDEKODE FOR JAVA SERVER

SunSpotHostApplication-KLASSEN:

```
package org.sunspotworld;


import com.sun.spot.io.j2me.radiogram.*;

import com.sun.spot.peripheral.ota.OTACommandServer;
import java.text.DateFormat;
import java.util.Date;
import javax.microedition.io.*;

public class SunSpotHostApplication {

    private void run()
    {

            Server server = new Server();
            server.runServer();
            SunSPOTServer data = new SunSPOTServer();
    }

    public static void main(String[] args) throws Exception {

        OTACommandServer.start("SendDataDemo");

        SunSpotHostApplication app = new SunSpotHostApplication();
        app.run();
    }
}
```

Server-KLASSEN:

```
package org.sunspotworld;



import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.EOFException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingUtilities;
```

```
public class Server extends JFrame{
	private JFrame frame;
	private JTextField enterfield;
		private JLabel label1;
		private JButton startbutton;
		private JTextArea textarea;
		private ObjectOutputStream output;
		private ObjectInputStream input;
		private ServerSocket server;
		private Socket connection;
		private int counter = 1;
	SunSPOTServer data;



		//Constructor - Set up GUI
		public Server()
		{

		  data = new SunSPOTServer();

			// Loading and adding all the GUI elements.
		  textarea = new JTextArea(30,15);
		  add(new JScrollPane(textarea), BorderLayout.CENTER);
		  setSize(400,200);
		  setVisible(true);

		}
		// Set up and run server
		public void runServer()
		{
			try
		  {
				server = new ServerSocket(9001);

				while (true)
				{
				try
					{
						waitForConnection();
				getStreams();
				processConnection();
			}
					catch(EOFException ioException)
					{
						displayMessage("\nServer teminated connection");
					}
					finally
					{
						closeConnection();
						counter++;
					}
				}
			}
			catch(IOException ioException)
			{
				ioException.printStackTrace();
			}
		}
```

```
  // wait for a connection
  private void waitForConnection() throws IOException
  {
InetAddress thisIP = InetAddress.getLocalHost();
displayMessage("Denne pc'en har følgende ip-address: " + thisIP.getHostAddress());
displayMessage( "\n\nVenter på tilkobling\n");

connection = server.accept();

if(connection.isBound())
{
   displayMessage( "Tilkobling " +counter + " mottatt fra: " +
connection.getInetAddress().getHostName());
}
  }
  // gets streams to send and receive data
  private void getStreams() throws IOException
  {
            output = new ObjectOutputStream( connection.getOutputStream());
            output.flush();

            input = new ObjectInputStream( connection.getInputStream());

            displayMessage("\nMottar I/O strøm\n");

  }
  // process connection from client
  private void processConnection() throws IOException
  {
String message = "Tilkobling vellykket\n";
//sendData(message);

do{
   try
   {
      message = (String ) input.readObject();
      displayMessage("\n" + message);
   }
   catch(ClassNotFoundException exception)
   {
      displayMessage("fail");
   }

   checkString(message);

}while(!message.equals("STENG"));
  }
  // close streams and socket
  private void closeConnection()
  {
displayMessage("\nKutter tilkobling\n\n");
try
{
   output.close();
   input.close();
   connection.close();
}
catch(IOException ioException)
{
   ioException.printStackTrace();
```

```
        }

          }
          // sending a message to client
          private void sendData(String message)
          {
        try
        {
            output.writeObject(message);
            output.flush();
            displayMessage(message);
        }
        catch(IOException ioException)
        {

        }
          }
          // shifting the showing text in textarea
          private void displayMessage(final String messageToDisplay)
          {
        SwingUtilities.invokeLater(
                new Runnable()
                {
                    public void run()
                    {
                        textarea.append( messageToDisplay );
                    }
                });
          }

    // Method for checking what the string contains, and who that is to receive it.
    public void checkString(String string)
    {

        string = string.replaceAll("[[,]] ", "");

        System.out.println(string);
        textarea.append(string);

        sendString(string);

    }
    public void sendString(String string)
    {
      try{
        data.RobotVerdi(string);
      }
      catch(Exception e)
      {
          e.printStackTrace();
      }
    }


}
```

SunSPOTServer – Klassen:

```
package org.sunspotworld;

import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.peripheral.ota.OTACommandServer;
import java.text.DateFormat;
import java.util.Date;
import javax.microedition.io.*;


public class SunSPOTServer {

    private static final int HOST_PORT = 67;
    private String value = null;
    DatagramConnection Connection = null;
    Datagram dg = null;

    public SunSPOTServer()
    {

    }

    public void RobotVerdi(String value)throws Exception
    {

        this.value = value;
        System.out.println("Value = " + value);

        if(value != null)
        {
        try {


            Connection = (DatagramConnection) Connector.open("radiogram://broadcast:" + HOST_PORT);
            dg = Connection.newDatagram(Connection.getMaximumLength());


        } catch (Exception e) {
             System.err.println("setUp caught " + e.getMessage());
             throw e;
        }

        // Main data collection loop
        while (true)
        {
            dg.reset();
            dg.writeUTF(value);
            Connection.send(dg);


            try {

            } catch (Exception e) {
                System.err.println("Caught " + e +  " while reading sensor samples.");
                throw e;
            }
        }
      }else
      {
         System.out.println("feil verdi");
      }
```

```
  }
}
```

Vedlegg 6

Kildekode - BugsterApp

```
package org.sunspotworld;

import com.sun.spot.peripheral.Spot;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.service.BootloaderListenerService;
import com.sun.spot.io.j2me.radiostream.*;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.util.*;

import java.io.*;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
import java.io.IOException;

public class Control extends MIDlet {

    private static final int Host_Port = 67;
    private String value = null;
    private byte IrFront = 0x43;
    private byte IrLeft = 0x73;
    private byte IrRight = 0x63;
    private byte IrBack = 0x53;
    private boolean IrSensor = false;
    private byte uart = 0;
    private byte handel = 0;

    //Hver sensor registrere sin forrige kommando fra mikrokontrolleren
    private byte OldUart[] = new byte[12];
    private int drive = 0;
    private int stop = 0;
    private int reverse = 0;


    private int count;
    //Hindre at programmet skal sende samme verdi til mikrokontoller mer en en gang.
    private int oneStop = 0;
    private int oneStop2 = 0;

    //Støtsensorene
    private boolean SFront = false;
    private boolean SBack = false;
    private boolean SLeft = false;
    private boolean SRight = false;

    //Felles variabel for alle støtsensorene
    private boolean SSensor = false;
    private int bDrive = 0;
```

```
private String userString;

//Verdien brukeren har satt for hver infrarødsensor
private byte IrFrontVerdi;
private byte IrBackVerdi;
private byte IrLeftVerdi;
private byte IrRightVerdi;

//Verdien som forteller hvilken posisjon sevoen er
private int Turn = 0;

//Siferene som skiller kjøreruten og hendelsemønsteret for hver IR-sensor
private int A2 = 0;
private int A3 = 0;
private int A4 = 0;
private int A5 = 0;
private int A6 = 0;

//Starter hendelsesmønsteret
private int StartFunction;

//Hele brukerens string.
private char UserArray[] = new char [100];

//Kjøremønsteret til roboten som bruker har satt
private char DriveArray[] = new char[100];


Buffer buffer = new Buffer();
DrivingRoute dr = new DrivingRoute(buffer);
EDemoBoard board = EDemoBoard.getInstance();



protected void startApp() throws MIDletStateChangeException {



  RadiogramConnection connect = null;
  Datagram dg = null;
  Datagram tg = null;

  readUart();


  board.initUART(9600, false);
 try
   {
```

```
    //Prøver å oppnå kontakt med server
    connect = (RadiogramConnection) Connector.open("radiogram://:" + Host_Port);
    tg = connect.newDatagram(connect.getMaximumLength());

 }catch(Exception e){
    System.err.println("Caught " + e + " in connection initialization.");

 }

while(true)
 try
     {

          tg.reset();
          //mottar datagram fra server
          connect.receive(tg);
          value = tg.readUTF();

          if (value != null)
          {

            controlString(value);
            dr.drivingPattern(DriveArray);

            while(true)
            {
               try
               {
                  //lagrer brukerens bestemelser av motoren
                  bDrive = buffer.getbDrive();

                  //Brukerens kjøreprogram er færdig, applikasjonen skal avslutte
                  if(bDrive == 5)
                  {
                    notifyDestroyed();
                  }


                  drive = buffer.getDrive();
                  reverse = buffer.getReverse();
                  Turn = buffer.getTurn();
                  SSensor = buffer.getSSensorene();
                  IrSensor = buffer.getIrSensorene();

                  //SSensor er tru om en av sensorene er satt true. Her sjekker vi alle sensorene.
                  if (SSensor == true)
                  {
                     try
```

```
        {
            SFront = buffer.getSFront();
            SBack = buffer.getSBack();
            SLeft = buffer.getSLeft();
            SRight = buffer.getSRight();

        }catch(Exception e){}
    }else
    {
        // Oppdaterer støtsensorene, ungår å kalle alle metodene på nytt
        SFront = false;
        SBack = false;
        SLeft = false;
        SRight = false;

    }

    //Hvis noen av IR-Sensorene er satt true
    if(IrSensor)
    {
        try{


            IrFront = buffer.getIrFront();
            IrBack = buffer.getIrBack();
            IrLeft = buffer.getIrLeft();
            IrRight = buffer.getIrRight();

        }catch (Exception e){}
    }else
    {
        //Setter alle IR-sensorene til høyeste nivå, ungår å kalle alle metodene på nytt
        IrFront = 0x47;
        IrBack = 0x57;
        IrLeft = 0x77;
        IrRight = 0x67;
    }

}catch(Exception e){}

/*Her blir det sjekka hvilken veg bruker ønsker å svinge.
 * Verdiene 1,2,3 står for venstre, høyre, rettfrem.
 */
if(Turn == 1)
    {
        if(oneStop !=1)
        {
            try
```

```
                {
                    //Svinger til Venstre
                    board.writeUART((byte)0x23);
                }catch(Exception e){}
                oneStop = 1;
            }
        }else if(Turn ==2)
        {
            if(oneStop !=2)
            {
                try
                {
                    //Svinger til Høyre
                    board.writeUART((byte)0x22);
                }catch(Exception e){}
                oneStop = 2;
            }
        }else if(Turn ==3)
        {
            if(oneStop!=3)
            {
                try
                {
                    //Hjulene skal stå beint frem
                    board.writeUART((byte)0x21);
                }catch(Exception e){}
                oneStop=3;
            }
        }


    //Dette er kjøre retningen som BRUKER ønsker
    if(bDrive == 1)
    {

     if ( IrFront <= IrFrontVerdi && drive == 1)
     {
        if(oneStop!= 1 && A2 < A3)
            {
                try{
                    //Sjekker alle verdiene mellom verdi 2 og 3 i bruker stringen
                    for(count = A2+2; count < A3; count++)
                    {
                        handel = handling(count);
                        board.writeUART((byte) handel);

                        //Passe på at robot rygger i to sekunder
                        if(handel == 0x13)
```

```
                {
                    Thread.sleep(2000);
                }
            }
                oneStop = 1;
                count = 0;

        }catch(Exception e) {}
    }else{}

}else if (IrRight <= IrRightVerdi && drive == 1)
{
    if(oneStop!= 2 && A3 < A4)
        {
        try{
            //Sjekker verdiene mellom verdiene 3 og 4 i brukerstringen
            for(count = A3+2; count < A4; count++)
            {
                handel = handling(count);
                board.writeUART((byte) handel);

                //passer på at robot svinger eller kjører i to sekunder
                if(handel == 0x13 || handel == 0x11)
                {
                    Thread.sleep(2000);
                }
            }
                oneStop = 2;
                count = 0;

        }catch(Exception e) {}
    }else{}
}else if (IrLeft <= IrLeftVerdi && drive == 1)
{
    if(oneStop!= 3 && A4 < A5)
        {
        try{
            //Sjekker verdiene mellom veridene 4 og 5 i bruker stringen
            for(count = A4+2; count < A5; count++)
            {
                handel = handling(count);
                board.writeUART((byte) handel);
                //passer på at robot svinger eller kjører i to sekunder
                if(handel == 0x13 || handel == 0x11)
                {
                    Thread.sleep(2000);
                }
            }
```

```
                    oneStop = 3;
                    count = 0;

                }catch(Exception e) {}
            }else{}

        // Denne er forløbig ikke i bruk.. IR-sensor skal ignoreres når robot kjører fremover
        }else if (IrBack <= IrBackVerdi && drive == 1)
        {
            if(oneStop!= 4)
                {

                    oneStop = 4;
            }else{}


        //Sjekker om noen av støtsensorene har slått ut mens robot kjører
        }else if((SFront == true || SBack == true || SLeft == true || SRight == true) && drive == 1)
        {
            if (oneStop != 5)
            {
                try{
                    //Sender stopp til mikrokontroller
                    board.writeUART((byte) 0x12);
                    oneStop = 5;
                }catch(Exception e){}
            }else
            {

            }

        //Hvis ingen av støtsenorene eller IR-sensorene er falske, skal motor starte
        }else if(SFront == false && SBack == false && SLeft == false && SRight == false &&
                IrFront > IrFrontVerdi && IrLeft >= 0x73 && IrRight >= 63 && drive == 0 )
        {
            if (oneStop != 6)
            {
                try{
                    //Sender start verdi til mikrokontroller
                    board.writeUART((byte) 0x11);
                    oneStop = 6;
                }catch(Exception e){}
            }else{}
        }else{}

    //Brukeren har satt roboten til å stoppe
    }else if(bDrive == 0)
    {
```

```
    if (oneStop != 7)
    {
      try{
        //sender stopp til mikrokontroller
        board.writeUART((byte) 0x12);
        oneStop = 7;
      }catch(Exception e){}
    }
  }
    //Brukeren har valt at roboten skal stoppe, eller robot rygger pga en hindring
    else if (bDrive== 2 || reverse ==1)
    {
        //Robot skal ignorere IR-sensor i front når den rygger
        if (IrFront <= IrFrontVerdi && reverse == 1)
        {
          if(oneStop!= 8)
          {
            oneStop = 8;
          }else{}

         //IR-sensor er lavere en brukerens ønske
        }else if (IrRight <= IrRightVerdi && reverse == 1)
        {
          if(oneStop!= 9 && A3 < A4)
          {
            try{

              //Sjekker verdiene mellom verdi 3 og 4 i bruker string
              for(count = A3+2; count < A4; count++)
              {
                handel = handling(count);
                board.writeUART((byte) handel);
                //Passer på at robot rygger eller kjører i to sekunder
                if(handel == 0x13 || handel == 0x11)
                {
                  Thread.sleep(2000);
                }
              }
                oneStop = 9;
                count = 0;

            }catch(Exception e) {}
          }else{}
        }else if (IrLeft <= IrLeftVerdi && reverse == 1)
        {
          if(oneStop!= 10 && A4 < A5)
          {
            try{
```

```
            //Sjekker verdiene mellom verdi 4 og 5 i bruker string
            for(count = A4+2; count < A5; count++)
            {
               handel = handling(count);
               board.writeUART((byte) handel);
               if(handel == 0x13 || handel == 0x11)
               {
                  Thread.sleep(2000);
               }
            }
               oneStop = 11;
               count = 0;

         }catch(Exception e) {}
      }else{}

   }else if (IrBack <= IrBackVerdi && reverse == 1)
   {

      if(oneStop!= 12 && A5 < A6)
         {
           try{
               //Sjekker verdiene mellom verdi 5 og 6 i bruker string
               for(count = A5+2; count < A6; count++)
               {
                  handel = handling(count);
                  board.writeUART((byte) handel);
                  //Sørger for at robot rygger i to sekund
                  if(handel == 0x11)
                  {
                     Thread.sleep(2000);
                  }
               }
                  oneStop = 12;
                  count = 0;

            }catch(Exception e) {}
      }else{}

      //Sjekker om en eller flere av søtsensorene er satt til true. Støtsensor
      //front blir bare ignorert
   }else if((SBack == true || SLeft == true || SRight == true) && reverse == 1)
   {
      if (oneStop != 14)
      {
         try{
            //Sender stopp komando til mikrokontroller
            board.writeUART((byte) 0x12);
```

```
                    oneStop = 14;
                }catch(Exception e){}
            }else{}

            // Sjekker om alle støt- og IR-sensorene er satt til false og motor står stille
        }else if(SBack == false && SLeft == false && SRight == false &&
         IrLeft >= 0x73 && IrRight >= 63 && IrBack >= 53 && reverse == 0 )
        {
            if (oneStop != 15)
            {
                try{

                    //Sender start verdi til mikrokontroller
                    board.writeUART((byte) 0x13);
                    oneStop = 15;
                }catch(Exception e){}
            }else{}
        }else{}
    } else{}
  }
}else
{

}

}catch(IOException ioe) {}
}


/*****************************************************************************
*********************/

public void readUart()
  {
    new Thread(){
        public void run()
        {

            EDemoBoard board = EDemoBoard.getInstance();
            board.initUART(9600, false);

     while(true)
      try{
          //Largrar innkommende kommando fra mikrokontrolleren.
           uart = board.readUART();

           if(uart == 0x11)
           {
```

```
    try{
       //Oppdaterer status til motor
        buffer.setDrive(1);
        buffer.setStop(0);
        buffer.setReverse(0);
    }catch(InterruptedException e){}

//mottatt en stopp verdi
}else if (uart == 0x12)
{
    try{
        //Oppdaterer status til motor
        buffer.setDrive(0);
        buffer.setStop(1);
        buffer.setReverse(0);
    }catch(InterruptedException e){}

//mottatt en rygge verdi
}else if(uart == 0x13)
{

    try{
        //Oppdaterer status til motor
        buffer.setDrive(0);
        buffer.setStop(0);
        buffer.setReverse(1);
    }catch(InterruptedException e){}
}
// kontrollerer om innkommende byte er en verdi for IR-sensoren i front
else if(uart >= 0x41 && uart <= 0x47)
{
    if(uart != OldUart[0])
    {
        try
        {
          buffer.setIrFront(uart);
        }catch(InterruptedException e){}
        OldUart[0]=uart;
    }else{}
 // kontrollerer om innkommende byte er en verdi for IR-sensoren bak
}else if(uart >= 0x51 && uart <= 0x57)
{
    if(uart != OldUart[1])
    {
        try
        {
          buffer.setIrBack(uart);
        }catch(InterruptedException e){}
```

```
        OldUart[1]=uart;
    }else{}
 // kontrollerer om innkommende byte er en verdi for IR-sensoren til høyre
}else if(uart >= 0x61 && uart <= 0x67)
{
    if(uart != OldUart[2])
    {
        try{
             buffer.setIrRight(uart);
        }catch(Exception e){}
        OldUart[2]=uart;
    }else{}
    // kontrollerer om innkommende byte er en verdi for IR-sensoren til venstre
}else if(uart >= 0x71 && uart <= 0x77)
{
    if(uart != OldUart[3])
    {
        try
        {
          buffer.setIrLeft(uart);
        }catch(InterruptedException e){}
        OldUart[3]=uart;
    }else{}
//Støtsensor registrert
}else if(uart == -111)
{
    if( uart != OldUart[4])
    {
        try
        {
            buffer.setSFront(true);
        }catch(Exception e){}
        OldUart[4] = uart;
    }else{}

//Støt front fjerna
}else if(uart == -110)
{
     if( uart != OldUart[5])
    {
        try
        {
            buffer.setSFront(false);
        }catch(Exception e){}
        OldUart[5] = uart;
    }else{}

//Støt bak registrert
```

```
}else if(uart == - 95)
{
    if( uart != OldUart[6])
   {
      try
      {
         buffer.setSBack(true);
      }catch(Exception e){}
      OldUart[6] = uart;
   }else{}

//Støt bak fjerna
}else if(uart == -94)
{
   if( uart != OldUart[7])
   {
      try
      {
         buffer.setSBack(false);
      }catch(Exception e){}

      OldUart[7] = uart;
   }else{}

//Støt høyre registrert
}else if(uart == -79)
{
   if( uart != OldUart[8])
   {
      try
      {
         buffer.setSRight(true);
      }catch(Exception e){}

      OldUart[8] = uart;
   }else{}

//Støt høyre fjerna
}else if(uart == -78)
{
   if( uart != OldUart[9])
   {
      try
      {
         buffer.setSRight(false);
      }catch(Exception e){}

      OldUart[9] = uart;
```

```
                    }else{}

                //Støt venstre registrert
                }else if(uart == -63)
                {
                    if( uart != OldUart[10])
                    {
                        try
                        {
                            buffer.setSLeft(true);
                        }catch(Exception e){}

                        OldUart[10] = uart;
                    }else{}

                 //Støt venstre fjerna
                }else if(uart == -62)
                {
                    if( uart != OldUart[11])
                    {
                        try
                        {
                            buffer.setSLeft(false);
                        }catch(Exception e){}

                        OldUart[11] = uart;
                    }else{}
                    //Kontrollerer om innkommende verdi er for servoen. Har ingen funskjon enda
                }else if(uart == 0x23 || uart == 0x22 || uart == 0x21)
                {

                }else
                {

                }

              }catch(IOException ioe) {}

              }
          }.start();
  }



/*""""""""""""""""""""""""""""""""""""""""*/
  public void controlString(String userString)
  {
      this.userString = userString;
```

```
for(int i = 0; i < userString.length(); i++)
        {
            UserArray[i] = userString.charAt(i);
            System.out.println("Array: "+ i +" "+ UserArray[i]);
            if(UserArray[i] == '2')
            {
                for (int b = 0; b < i; b++)
                {
                    DriveArray[b] = UserArray[b];
                }

                A2 = i;
                StartFunction = i+1;

                System.out.println("A2: "+ A2);

            }else if(UserArray[i] == '3')
            {
                A3 = i;
                System.out.println("A3: "+ A3);

            }else if(UserArray[i] == '4')
            {
                A4 = i;
                System.out.println("A4: "+ A4);

            }else if(UserArray[i] == '5')
            {
                A5 = i;
                A6 = userString.length();
                System.out.println("A5: "+ A5);
            }else{}
        }

        for(int k = StartFunction; k <=userString.length(); k++)
        {
            if(UserArray[k] == 'F')
            {
                IrFrontVerdi = 0x41;
                try{
                    buffer.setIrFrontVerdi(IrFrontVerdi);
                }catch(Exception e){}
            }else if(UserArray[k] == 'G')
            {
                IrFrontVerdi = 0x42;
                try{
                    buffer.setIrFrontVerdi(IrFrontVerdi);
                }catch(Exception e){}
```

```
}else if(UserArray[k] == 'H')
{
   IrFrontVerdi = 0x43;
   try{
      buffer.setIrFrontVerdi(IrFrontVerdi);
   }catch(Exception e){}
}else if(UserArray[k] == 'I')
{
   IrBackVerdi = 0x51;
   try{
      buffer.setIrBackVerdi(IrBackVerdi);
   }catch(Exception e){}
}else if(UserArray[k] == 'J')
{
   IrBackVerdi = 0x52;
   try{
      buffer.setIrBackVerdi(IrBackVerdi);
   }catch(Exception e){}
}else if(UserArray[k] == 'K')
{
   IrBackVerdi = 0x53;
   try{
      buffer.setIrBackVerdi(IrBackVerdi);
   }catch(Exception e){}
}else if(UserArray[k] == 'L')
{
   IrLeftVerdi = 0x71;
   try{
      buffer.setIrLeftVerdi(IrLeftVerdi);
   }catch(Exception e){}
}else if(UserArray[k] == 'M')
{
   IrLeftVerdi = 0x72;
   try{
      buffer.setIrLeftVerdi(IrLeftVerdi);
   }catch(Exception e){}
}else if(UserArray[k] == 'N')
{
   IrLeftVerdi = 0x73;
   try{
      buffer.setIrLeftVerdi(IrLeftVerdi);
   }catch(Exception e){}
}else if(UserArray[k] == 'O')
{
   IrRightVerdi = 0x61;
   try{
      buffer.setIrRightVerdi(IrRightVerdi);
   }catch(Exception e){}
```

```
                    }else if(UserArray[k] == 'P')
                    {
                        IrRightVerdi = 0x62;
                        try{
                            buffer.setIrRightVerdi(IrRightVerdi);
                        }catch(Exception e){}
                    }else if(UserArray[k] == 'Q')
                    {
                        IrRightVerdi = 0x63;
                        try{
                            buffer.setIrRightVerdi(IrRightVerdi);
                        }catch(Exception e){}
                    }else{}
                }

     }

  /*""""""""""""""""""""""""""""""""""""*/
    //Kontrollerer innholdet i UserArray av angitt plassering. Retunerer en kommado ut i fra hva innholdet er.
   public byte handling(int count)
   {

       int location = count;

          if(UserArray[location] == 'A')
          {
            return 0x11;
          }else if(UserArray[location] == 'B')
          {
              return 0x12;

          }else if(UserArray[location] == 'C')
          {
              return 0x13;
          }else if(UserArray[location] == 'D')
          {
              return 0x23;
          }else if(UserArray[location] == 'E')
          {
              return 0x22;
          }else if(UserArray[location] == 'R')
          {
              return 0x24;
          }else{}

       return 0;
```

```
    }

    protected void pauseApp() {}

    protected void destroyApp(boolean unconditional) throws MIDletStateChangeException
    {


    }
}
```

```
package org.sunspotworld;

/**
 *
 * @author Helene
 *
 *
 *Denne klassen lagrer verdiene til sensorene, motorene og bruker.
 */
public class Buffer
{
    //IR-sensorverdiene
    private byte IrFront = 0x43;
    private byte IrLeft = 0x73;
    private byte IrRight = 0x63;
    private byte IrBack = 0x53;

    //Støtsensorene
    private boolean SFront = false;
    private boolean SBack = false;
    private boolean SLeft = false;
    private boolean SRight = false;

    //brukerns valgte reaksjonsavstand på IR-sensorene
    private byte IrFrontVerdi;
    private byte IrBackVerdi;
    private byte IrLeftVerdi;
    private byte IrRightVerdi;

    private byte uart = 0;
    private int drive = 0;
    private int stop = 0;
    private int reverse = 0;
    private int bDrive;
    private int turn = 0;

    private boolean occupiedIrFront = false;
    private boolean occupiedDrive = false;
    private boolean occupiedStop = false;
    private boolean occupiedReverse = false;
    private boolean occupiedIrBack = false;
    private boolean occupiedIrLeft = false;
    private boolean occupiedIrRight = false;
    private boolean occupiedSFront = false;
    private boolean occupiedSBack = false;
    private boolean occupiedSLeft = false;
    private boolean occupiedSRight = false;
```

```
private boolean occupiedSSensor = false;
private boolean occupiedbDrive = false;
private boolean occupiedIrFrontVerdi = false;
private boolean occupiedIrBackVerdi = false;
private boolean occupiedIrLeftVerdi = false;
private boolean occupiedIrRightVerdi = false;
private boolean occupiedTurn = false;



public Buffer()
{
}

public synchronized void setDrive(int drive)throws InterruptedException
{
   while (occupiedDrive)
   {
       System.out.println("1");
      wait();
   }

   occupiedDrive = true;
   this.drive = drive;

   occupiedDrive = false;

   notifyAll();


}

public synchronized int getDrive()throws InterruptedException
{
   while(occupiedDrive)
   {
      wait();
   }

   notifyAll();

   return drive;
}

public synchronized void setStop(int stop)throws InterruptedException
{
   while (occupiedStop)
   {
```

```
        wait();
    }
    occupiedStop = true;
    this.stop = stop;


    occupiedStop = false;


    notifyAll();


}

public synchronized int getStop()throws InterruptedException
{
    while(occupiedStop)
    {

        wait();
    }


    notifyAll();


    return stop;
}

  public synchronized void setReverse(int reverse)throws InterruptedException
{
    while (occupiedReverse)
    {

        wait();
    }

    occupiedReverse = true;
    this.reverse = reverse;

    occupiedReverse = false;
    notifyAll();


}

public synchronized int getReverse()throws InterruptedException
```

```
    {
        while(occupiedReverse)
        {
            wait();
        }


        notifyAll();

        return reverse;
    }



    public synchronized boolean getIrSensorene() throws InterruptedException
    {

        if(IrRight > IrRightVerdi && IrLeft > IrLeftVerdi && IrBack >IrBackVerdi && IrFront
>IrFrontVerdi)
        {


            notifyAll();
            return false;
        }else
        {
            notifyAll();
            return true;
        }


    }

    public synchronized void setIrFront(byte IrFront) throws InterruptedException
    {
        while (occupiedIrFront)
        {
            wait();
        }
        occupiedIrFront = true;

        this.IrFront = IrFront;
        occupiedIrFront = false;
        notifyAll();
    }

    public synchronized byte getIrFront() throws InterruptedException
    {
```

```
    while(occupiedIrFront)
    {
        wait();
    }

    notifyAll();

    return IrFront;
}

public synchronized void setIrBack(byte IrBack) throws InterruptedException
{
    while (occupiedIrBack)
    {

        wait();
    }
    occupiedIrBack = true;

    this.IrBack = IrBack;

    occupiedIrBack = false;
    notifyAll();
}

public synchronized byte getIrBack() throws InterruptedException
{
    while(occupiedIrBack)
    {
        wait();
    }

    occupiedIrBack = true;
    notifyAll();
    occupiedIrBack = false;
    return IrBack;
}

  public synchronized void setIrLeft(byte IrLeft) throws InterruptedException
{
    while (occupiedIrLeft)
    {

        wait();
    }
    occupiedIrLeft = true;
    this.IrLeft = IrLeft;
```

```
    occupiedIrLeft = false;
    notifyAll();
}

public synchronized byte getIrLeft() throws InterruptedException
{
    while(occupiedIrLeft)
    {
        wait();
    }

    notifyAll();

    return IrLeft;
}

  public synchronized void setIrRight(byte IrRight) throws InterruptedException
{
    while (occupiedIrRight)
    {

        wait();
    }
    occupiedIrRight = true;

    this.IrRight = IrRight;

    occupiedIrRight = false;
    notifyAll();
}

public synchronized byte getIrRight() throws InterruptedException
{
    while(occupiedIrRight)
    {
        wait();
    }

    notifyAll();

    return IrRight;

}


/*
 *
 *  Nærhetssensorene!
```

```
 *
 */


 public synchronized boolean getSSensorene() throws InterruptedException
{

    if(SRight == false && SLeft == false && SBack == false && SFront == false)
    {


        notifyAll();
        return false;
    }else
    {


        notifyAll();
        return true;
    }
}


public synchronized void setSFront(boolean SFront) throws InterruptedException
 {
    while (occupiedSFront)
    {

        wait();
    }
    occupiedSFront = true;

    this.SFront = SFront;

    occupiedSFront = false;
    notifyAll();
 }

 public synchronized boolean getSFront() throws InterruptedException
 {
    while(occupiedSFront)
    {
        wait();
    }


    notifyAll();
```

```
    return SFront;
}



public synchronized void setSLeft(boolean SLeft) throws InterruptedException
{
    while (occupiedSLeft)
    {


        wait();
    }
    occupiedSLeft = true;
    this.SLeft = SLeft;

    occupiedSLeft = false;
    notifyAll();
}

public synchronized boolean getSLeft() throws InterruptedException
{
    while(occupiedSLeft)
    {

        wait();
    }

    notifyAll();

    return SLeft;
}


public synchronized void setSRight(boolean SRight) throws InterruptedException
{
    while (occupiedSRight)
    {
        wait();
    }

    this.SRight = SRight;
    notifyAll();
}

public synchronized boolean getSRight() throws InterruptedException
{
    while(occupiedSRight)
```

```
    {
        wait();
    }

    notifyAll();

    return SRight;
}


public synchronized void setSBack(boolean SBack) throws InterruptedException
{
    while (occupiedSBack)
    {
        wait();
    }
    occupiedSBack = true;
    this.SBack = SBack;

    occupiedSBack = false;
    notifyAll();
}

public synchronized boolean getSBack() throws InterruptedException
{
    while(occupiedSBack)
    {
        wait();
    }

    notifyAll();

    return SBack;
}









public synchronized void setbDrive(int bDrive) throws InterruptedException
{
    while (occupiedbDrive)
    {
```

```
            wait();
        }
        occupiedbDrive = true;
        this.bDrive = bDrive;
        occupiedbDrive = false;
        notifyAll();
    }

    public synchronized int getbDrive() throws InterruptedException
    {
        while(occupiedbDrive)
        {
            wait();
        }

        notifyAll();

        return bDrive;

    }




/*****************************************************************************
************/

    public synchronized void setTurn(int turn) throws InterruptedException
    {
        while (occupiedTurn)
        {
            wait();
        }
        occupiedTurn = true;
        this.turn = turn;
        occupiedTurn = false;
        notifyAll();
    }

    public synchronized int getTurn() throws InterruptedException
    {
        while(occupiedTurn)
        {
            wait();
        }
        notifyAll();

        return turn;
```

```
}

public synchronized void setIrFrontVerdi(byte IrFrontVerdi) throws InterruptedException
{
   this.IrFrontVerdi = IrFrontVerdi;
   occupiedIrFrontVerdi = true;
   notifyAll();
}

public synchronized byte getIrFrontVerdi() throws InterruptedException
{
   while(!occupiedIrFrontVerdi)
   {
      wait();
   }

   notifyAll();
   return IrFrontVerdi;

}

public synchronized void setIrBackVerdi(byte IrBackVerdi) throws InterruptedException
{
   this.IrBackVerdi = IrBackVerdi;
   occupiedIrBackVerdi = true;
   notifyAll();
}

public synchronized byte getIrBackVerdi() throws InterruptedException
{
   while(!occupiedIrBackVerdi)
   {
      wait();
   }

   notifyAll();
   return IrBackVerdi;
}

public synchronized void setIrRightVerdi(byte IrRightVerdi) throws InterruptedException
```

```
    {
        this.IrRightVerdi = IrRightVerdi;
        occupiedIrRightVerdi = true;
        notifyAll();
    }

    public synchronized byte getIrRightVerdi() throws InterruptedException
    {
        while(!occupiedIrRightVerdi)
        {
            wait();
        }

        notifyAll();
        return IrRightVerdi;
    }

     public synchronized void setIrLeftVerdi(byte IrLeftVerdi) throws InterruptedException
    {
        this.IrLeftVerdi = IrLeftVerdi;
        occupiedIrLeftVerdi = true;
        notifyAll();
    }

    public synchronized byte getIrLeftVerdi() throws InterruptedException
    {
        while(!occupiedIrLeftVerdi)
        {
            wait();
        }

        notifyAll();

        return IrLeftVerdi;
    }
}
```

```
package org.sunspotworld;

import com.sun.spot.sensorboard.EDemoBoard;
import java.lang.StringBuffer;


public class DrivingRoute
{
     private int CountTime = 4;
     private final Buffer buffer;
     private char userString[] = new char[100];
     //Kjørestatusen som bruker har bestemt roboten skal være i
     private int DriveStatus;
     private int turn;

    public DrivingRoute(Buffer delt)
    {
        buffer = delt;

    }

    //Sjekker en og en verdi i kjøremønsteret til bruker og utfører handlingen
    public void drivingPattern(char value[])
    {
        this.userString = value;
        new Thread(){
            public void run()
            {
                for(int j = 0; j < userString.length; j++)
                {
                  //roboten skal kjøre fremover
                  if( userString[j] == 'A')
                  {
                      System.out.println("A");
                       userDrive(1);

                   //Roboten skal stoppe
                  }else if(userString[j] == 'B')
                  {
                      try{
                          // gir beskjed om at bruker ønsker å stoppe motoren
                           buffer.setbDrive(0);
                           Thread.sleep(2000);
                      }catch(Exception e){}

                  //Roboten skal å rygge
                  }else if(userString[j] == 'C')
```

```
                {
                    userDrive(2);

                //Roboten skal svinge til venstre
                }else if(userString[j] == 'D')
                {
                    try{
                        buffer.setTurn(1);
                    }catch(Exception e){}

                //Roboten skal svinge til høyre
                }else if(userString[j] == 'E')
                {
                    try{
                        buffer.setTurn(2);

                     }catch(Exception e){}

                //Roboten skal kjøre rett frem
                }else if(userString[j] == 'R')
                {
                    try{
                        buffer.setTurn(3);
                    }catch(Exception e){}
                }else
                {
                    //Kjøreprogram er ferdig
                    try{
                            //Gir beskjed om å stoppe motor
                            buffer.setbDrive(0);
                            Thread.sleep(1000);
                            //Gir beskjed om å avslutte applikasjonen
                            buffer.setbDrive(5);
                        }catch(Exception e){}
                }
            }
        }
    }.start();
}

public void userDrive(int DriveStatus)
{
  this.DriveStatus = DriveStatus;

    int status = 0;
    int i = 0;

        try{
```

```
  //Gir besked om hvilken kjørestatus bruker har valgt
   buffer.setbDrive(DriveStatus);

}catch(Exception e){}

//Passer på at robot kjører i x antall sekunder
while(i < CountTime)
{
   //Sjekker om roboten kjører
   try{
       if(DriveStatus == 1)
       {
         status = buffer.getDrive();
       }else if(DriveStatus == 2)
       {
          status = buffer.getReverse();
       }
      }catch(Exception e){}

   //Fungerer som en stoppeklokke som sjekker staus hvert halve sekund
   //Status = faktisk kjøre status
   while(status == 1 && i< CountTime)
   {
      try{
         Thread.sleep(500);
      }catch(Exception e){}

      i++;

      try{
         //bruker har satt robot til å kjøre fremover
         if(DriveStatus == 1)
          {
            //Sjekker om roboten faktisk kjører fremover
            status = buffer.getDrive();

           //bruker har satt roboten til å rygge
          }else if(DriveStatus == 2)
          {
            //sjekker om roboten faktisk rygger
            status = buffer.getReverse();
          }
      }catch(Exception e){}

   }

   //robot står stille, skal ikke øke i
   while (status == 0 && i< CountTime)
```

```
            {
              try{
                //bruker har satt roboten til å kjøre fremover
                if(DriveStatus == 1)
                 {
                   //Sjekker faktisk status til motor
                   status = buffer.getDrive();
                   Thread.sleep(200);

                 //bruker har satt roboten til å rygge
                }else if(DriveStatus == 2)
                {
                   //sjekker faktisk status til motor
                   status = buffer.getReverse();
                   Thread.sleep(200);
                }
              }catch(Exception e){}

            }
          }
    }
}
```

Vedlegg 7

Kildekode - Mikrokontroller

```
#include <stdlib.h>
      //importerer nødvendige bibiloteker
#include <pwm.h>
#include <delays.h>
#include <portb.h>
#include <timers.h>
#include <adc.h>
#include <usart.h>
#include <capture.h>

#pragma config WDT = OFF                                        //skrur av
watch dog timer
#pragma config LVP = OFF                                        //skrur av
low voltage programming

void chk_isr(void);
      //prototyper
void commandCenter(void);
void ADC(void);
void startUltra(void);
void ultralyd(void);
void beregnAvstand(void);
void stotSensor(void);

//Ultralyd
unsigned int res1, res2;                                //resultat fra
ultralydsensor
int tDiff, u;                                                 //differansen
mellom res1 og res2
double avstand;                                               //avtstand fra
ultralydsensor til objekt

//ADC
unsigned int resADC;                                    //resultat fra ADC
char ch;                                                      //ADC kanal
int bk;
int a;
      //Batterikontroll; denne holder styr på når spennningsnivået over
batteriet skal måles

//UART
char mottatt, kommando, adresse;
int oe = 0;
int fe = 0;
int c;
      //settes til 1 når programmet skal kjøres commandcenter()
//støt
int sf, sb, sh, sv, s;                                        //variabler til
støtsensor

//generelle tilstandsvariabler
int k = 0, c=0;
int i = 0;



#pragma code My_HiPrio_Int=0x0008               //ved høyprioritets
interrupt gå til denne adressen
void My_HiPrio_Int (void)
{
      _asm GOTO chk_isr  _endasm
```

```
}
#pragma code
#pragma interrupt chk_isr


void chk_isr(void)                                          //interrupt
service routine
{
      while(PIR1bits.RCIF == 1)                       //løkke kjøres hvis
noe er mottatt fra på UART 1
      {
            mottatt = Read1USART();                   //henter verdi fra
mottakerenhet
            c=1;                                            //sier fra at
programmet skal gå til commandcenter
            k++;
      }
      while(INTCONbits.TMR0IF == 1)             //hvis Timer0 har telt en
runde
      {
            INTCONbits.TMR0IF = 0;                    // senker tmr0 int
flagg
            ConvertADC();                                   //starter AD
konvertering
            bk++;                                           //inkrementerer
batterikontroll variable
            s++;                                            //inkrementerer
s; når denne blir høy nok sjekkes støtsensorene
      }
      while(PIR1bits.ADIF == 1)                       //hvis fullført AD
konvertering
      {
            PIR1bits.ADIF = 0;                              // senker ad int
flagg
            resADC = ReadADC();                             //henter
resultat fra ad konverter
            a=1;                                            //sier i fra at
programmet skal kjøre ADC()
      }
      if(PIR2bits.CCP2IF == 1)                        //hvis tilbakemelding
fra ultralydsensor
      {
            ultralyd();                                     //kjør
ultralyd()
      }
}

void main(void)
{
//--------initaliserer UART----------------------------------------
      TRISCbits.TRISC6 = 0;
                        //setter pinne RC6 til utgang
      TRISCbits.TRISC7 = 1;
                        //setter pinne RC7 til inngang

      Open1USART(USART_TX_INT_OFF & USART_RX_INT_ON & USART_ASYNCH_MODE &
USART_EIGHT_BIT & USART_CONT_RX & USART_BRGH_HIGH, 25); //Åpner UART
funksjonen; BAUDE RATE 9600

      PIE1bits.RC1IE = 1;
            //interrupt ved mottak
```

```
      PIR1bits.RC1IF = 0;
            //nuller interrupt flagg

//-------initialiserer ADC-----------------------------------------
      TRISAbits.TRISA0 = 1;
                                  // setter port A til inngang

      OpenTimer0(TIMER_INT_ON & T0_16BIT & T0_SOURCE_INT & T0_PS_1_4);
                  //STARTER TIMER0, 16 bit, prescaler 1/4; denne gir et
interrupt hvert 0.26 sekund

      OpenADC(ADC_FOSC_8 & ADC_RIGHT_JUST & ADC_6ANA, ADC_CH0 & ADC_INT_ON
& ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS); //åpner AD konverter; ADC

      PIE1bits.ADIE = 1;
                              //enabler AD interrupt
      PIR1bits.ADIF = 0;
                              //senker AD interrupt flagg
      INTCONbits.TMR0IF = 0;
                        //senker timr0 interrupt flagg


//---------------Motor----------------------------------------
      OpenTimer2(TIMER_INT_OFF & T2_PS_1_16 & T2_POST_1_1);
      //timer 2 styrer PWM funksjonene

      //Fremdrift
      OpenPWM3(255);                                      //åpner pwm
funskjon med perioden på 4ms
      SetDCPWM3(0);                                       //motor stopp

      //styring
      OpenPWM4(255);
                              //periode på 4 ms
      SetDCPWM4(300);                                     //rett frem

      // retning ultralyd
      OpenPWM5(255);                                      //periode på 4ms
      SetDCPWM5(300);                                     //rett frem

//---------------ultralyd-----------------------------------
      OpenCapture2(CAPTURE_INT_OFF & C2_EVERY_RISE_EDGE);
                                        //setter ccp2 pinnen til capture
mode
      OpenTimer1(TIMER_INT_OFF & T1_16BIT_RW & T1_SOURCE_INT & T1_PS_1_1 &
T12_SOURCE_CCP);  //åpner en 16 bits timer
      avstand=0;
//--------------støtsensorer--------------------------------
      MEMCONbits.EBDIS = 1;
                              //gjør port D til IO pinner
      PORTD = 0;
      TRISD = 1;
                                  //port D til inngang
      sf=0;
      sb=0;
      sh=0;
      sv=0;

//--------------Generelle instillinger----------------------
      OSCCON = 0b01100010;
                        //OSC 4MHz
```

```
	INTCONbits.GIE = 1;
	                        //enabler interrupt
	INTCONbits.PEIE = 1;

	k=0;


	while(1)
	{
//----------UART----------------------
		if(RCSTAbits.OERR == 1)
	//sjekk overrun feil
		{
			RCSTAbits.CREN = 0;
			oe++;
			RCSTAbits.CREN = 1;
		}

		if(RCSTAbits.FERR == 1)
		//sjekk etter framing error
		{
			Read1USART();
			//kaster ugyldig resultat
			fe++;
		}
//-------commandcenter---------------------------
		if(c==1)
			//starter commandCenter
		{
			commandCenter();
			c=0;
		}
//---------------ADC----------------------------
		if(a==1)
			//hvis resultat fra AD konvertering er klar
		{
			ADC();
				//starter ADC()
			a=0;
			//setter ADC status til 0: ingen ADC resultat klar
		}
//-------------støtsensor-----------------
		if(s>=1)
			//sjekker støtsensor når s>=1: hvert 0.26 sekund
		{
			stotSensor();
			//starter stosSensor()
			s=0;
			//nuller støt status:
		}

	}

}



//------------------Støtsensor---------------------------------------
void stotSensor(void)
{
```

```
            if(PORTDbits.RD1 == 1)
            //støtsensor fram, hvis bryter aktiv (trykket inn)
            {
                  if(sf==0)
                  //hvis sf=0; sf=0 betyr at forrige gang støtsensoren ble
sjekket var den ikke aktiv
                  {
                        while(Busy1USART());
            //venter til UART sender er ledig
                        Write1USART(0x91);
                  //sender at fremre bryter er aktiv
                        sf=1;
                  //oppdaterer status til aktiv
                  }
            }
            if(PORTDbits.RD1 == 0)
            //hvis fremre bryter ikke er aktiv
            {
                  if(sf==1)
                  //sjekker om bryter var aktiv sist støtsensor ble sjekket
                  {
                        while(Busy1USART());
            //venter til UART sender er ledig
                        Write1USART(0x92);
                  //sender at fremre bryter ikke er aktiv
                        sf=0;
                  //oppdaterer status til ikke aktiv
                  }
            }

            if(PORTDbits.RD2 == 1)
            //støtsensor bak, hvis bryter aktiv
            {
                  if(sb==0)
                  //hvis sb=0; sb=0 betyr at forrige gang støtsensoren ble
sjekket var den ikke aktiv
                  {
                        while(Busy1USART());
            //venter til UART sender er ledig
                        Write1USART(0xA1);
                  //sender at bakre bryter er aktiv
                        sb=1;
                  //oppdaterer status til aktiv
                  }
            }
            if(PORTDbits.RD2 == 0)
            //hvis bakre bryter ikke er aktiv
            {
                  if(sb==1)
                  //sjekker om bryter var aktiv sist støtsensor ble sjekket
                  {
                        while(Busy1USART());
            //venter til UART sender er ledig
                        Write1USART(0xA2);
                  //sender at bakre bryter ikke er aktiv
                        sb=0;
                  //oppdaterer status til ikke aktiv
                  }
            }
```

```
            if(PORTDbits.RD3 == 1)
            //støtsensor høyre, hvis bryter aktiv
            {
                  if(sh==0)
                  //hvis sh=0; sh=0 betyr at forrige gang støtsensoren ble
sjekket var den ikke aktiv
                  {
                        while(Busy1USART());
            //venter til UART sender er ledig
                        Write1USART(0xB1);
                  //sender at høyre bryter er aktiv
                        sh=1;
                  //oppdaterer status til aktiv
                  }
            }
            if(PORTDbits.RD3 == 0)
            //hvis høyre bryter ikke er aktiv
            {
                  if(sh==1)
                  //sjekker om bryter var aktiv sist støtsensor ble sjekket
                  {
                        while(Busy1USART());
            //venter til UART sender er ledig
                        Write1USART(0xB2);
                  //sender at høyre bryter ikke er aktiv
                        sh=0;
                  //oppdaterer status til ikke aktiv
                  }
            }

            if(PORTDbits.RD4 == 1)
            //støtsensor venstre, hvis bryter aktiv
            {
                  if(sv==0)
                  //hvis sv=0; sv=0 betyr at forrige gang støtsensoren ble
sjekket var den ikke aktiv
                  {
                        while(Busy1USART());
            //venter til UART sender er ledig
                        Write1USART(0xC1);
                  //sender at venstre bryter er aktiv
                        sv=1;
                  //oppdaterer status til aktiv
                  }
            }
            if(PORTDbits.RD4 == 0)
            //hvis venstre bryter ikke er aktiv
            {
                  if(sv==1)
                  //sjekker om bryter var aktiv sist støtsensor ble sjekket
                  {
                        while(Busy1USART());
            //venter til UART sender er ledig
                        Write1USART(0xC2);
                  //sender at venstre bryter ikke er aktiv
                        sv=0;
                  //oppdaterer status til ikke aktiv
                  }
            }

}
```

```
//------------------------ADC-----------------------------------------------
---
void ADC(void)
{
      ch = ADCON0 & 0b00111100;
            //henter ut bitsene som innholder informasjonen om hvilken
kanal
//Vref = 5
/*    if(ch==0b0)
                  //kanal 0 AN0. IRfram
      {
            if(resADC<=1023 && resADC>=274) Write1USART(0x41);
      //sender resultat; 10-20
            if(resADC<274 && resADC>=191) Write1USART(0x42);
      //20-30
            if(resADC<191 && resADC>=148) Write1USART(0x43);
      //30-40
            if(resADC<148 && resADC>=120) Write1USART(0x44);
      //40-50
            if(resADC<120 && resADC>=102) Write1USART(0x45);
      //50-60
            if(resADC<102 && resADC>=90) Write1USART(0x46);
      //60-70
            if(resADC<90 && resADC>=0) Write1USART(0x47);
      //70->

            SetChanADC(ADC_CH1);
            //bytter til kanal 2
      }
      if(ch==0b000100)
            //kanal 1 AN1. IRbak
      {
            if(resADC<=1023 && resADC>=274) Write1USART(0x51);
            if(resADC<274 && resADC>=191) Write1USART(0x52);
            if(resADC<191 && resADC>=148) Write1USART(0x53);
            if(resADC<148 && resADC>=120) Write1USART(0x54);
            if(resADC<120 && resADC>=102) Write1USART(0x55);
            if(resADC<102 && resADC>=90) Write1USART(0x56);
            if(resADC<90 && resADC>=0) Write1USART(0x57);
            SetChanADC(ADC_CH2);
            //bytter til kanal 2
      }
      if(ch==0b001000)
            //kanal 2 AN2. IRhøyre
      {
            if(resADC<=1023 && resADC>=274) Write1USART(0x61);
            if(resADC<274 && resADC>=191) Write1USART(0x62);
            if(resADC<191 && resADC>=148) Write1USART(0x63);
            if(resADC<148 && resADC>=120) Write1USART(0x64);
            if(resADC<120 && resADC>=102) Write1USART(0x65);
            if(resADC<102 && resADC>=90) Write1USART(0x66);
            if(resADC<90 && resADC>=0) Write1USART(0x67);
            SetChanADC(ADC_CH3);
            //bytter til kanal 3
      }
      if(ch==0b001100)
            //kanal 3 AN3. IRvenstre
      {
            if(resADC<=1023 && resADC>=274) Write1USART(0x71);
            if(resADC<274 && resADC>=191) Write1USART(0x72);
```

```
            if(resADC<191 && resADC>=148) Write1USART(0x73);
            if(resADC<148 && resADC>=120) Write1USART(0x74);
            if(resADC<120 && resADC>=102) Write1USART(0x75);
            if(resADC<102 && resADC>=90) Write1USART(0x76);
            if(resADC<90 && resADC>=0) Write1USART(0x77);

            SetChanADC(ADC_CH0);
            //bytter til kanal 0
      }*/
//Vref = 3,3
      if(ch==0b0)
                  //kanal 0 AN0. IRfram
      {
            if(resADC<=1023 && resADC>=407) Write1USART(0x41);
      //sender resultat; 10-20
            if(resADC<407 && resADC>=284) Write1USART(0x42);
      //20-30
            if(resADC<284 && resADC>=220) Write1USART(0x43);
      //30-40
            if(resADC<220 && resADC>=180) Write1USART(0x44);
      //40-50
            if(resADC<180 && resADC>=153) Write1USART(0x45);
      //50-60
            if(resADC<153 && resADC>=133) Write1USART(0x46);
      //60-70
            if(resADC<133 && resADC>=0) Write1USART(0x47);
      //70->

            SetChanADC(ADC_CH1);
            //bytter til kanal 2
      }
      if(ch==0b000100)
            //kanal 1 AN1. IRbak
      {
            if(resADC<=1023 && resADC>=407) Write1USART(0x51);
            if(resADC<407 && resADC>=284) Write1USART(0x52);
            if(resADC<284 && resADC>=220) Write1USART(0x53);
            if(resADC<220 && resADC>=180) Write1USART(0x54);
            if(resADC<180 && resADC>=153) Write1USART(0x55);
            if(resADC<153 && resADC>=133) Write1USART(0x56);
            if(resADC<133 && resADC>=0) Write1USART(0x57);
            SetChanADC(ADC_CH2);
            //bytter til kanal 2
      }
      if(ch==0b001000)
            //kanal 2 AN2. IRhøyre
      {
            if(resADC<=1023 && resADC>=407) Write1USART(0x61);
            if(resADC<407 && resADC>=284) Write1USART(0x62);
            if(resADC<284 && resADC>=220) Write1USART(0x63);
            if(resADC<220 && resADC>=180) Write1USART(0x64);
            if(resADC<180 && resADC>=153) Write1USART(0x65);
            if(resADC<153 && resADC>=133) Write1USART(0x66);
            if(resADC<133 && resADC>=0) Write1USART(0x67);
            SetChanADC(ADC_CH3);
            //bytter til kanal 3
      }
      if(ch==0b001100)
            //kanal 3 AN3. IRvenstre
      {
            if(resADC<=1023 && resADC>=407) Write1USART(0x71);
```

```
            if(resADC<407 && resADC>=284) Write1USART(0x72);
            if(resADC<284 && resADC>=220) Write1USART(0x73);
            if(resADC<220 && resADC>=180) Write1USART(0x74);
            if(resADC<180 && resADC>=153) Write1USART(0x75);
            if(resADC<153 && resADC>=133) Write1USART(0x76);
            if(resADC<133 && resADC>=0) Write1USART(0x77);

            SetChanADC(ADC_CH0);
            //bytter til kanal 0
      }
      if(ch==0b010000)
            //kanal 4
      {
            if(resADC<930) Write1USART(0xE1);
      //sier ifra hvis roboten må lades, 930 er grensen ved ref 3,3 v
            SetChanADC(ADC_CH0);
            //bytter til kanal 0
      }

      if(bk>=120)
      {
            bk=0;
                  //nuller batterikonroll
            SetChanADC(ADC_CH4);
            //bytter til kanal 4
      }
}

//--------------ultralyd-------------------------------------
void startUltra(void)
{
      TRISCbits.TRISC1 = 0;                                     //gjør RC1
til utgang
      PORTCbits.RC1 = 1;
      //sender start signal til ultralyd sensor
      Delay1TCY() Delay1TCY();                                  //vent 5
us
      PORTCbits.RC1 = 0;
      TRISCbits.TRISC1 = 1;                                     //gjør RC1
til inngang
      PIE2bits.CCP2IE = 1;                                      // enabler
interrupt
      PIR2bits.CCP2IF = 0;                                      //nuller
interupt flagg
      u=1;
      //oppdaterer ultralyd status;
}

void ultralyd(void)
{

      if(u == 2)                                          //hvis u=2;
mottat falling edge fra ultralyd sensor;
      {
            PIE2bits.CCP2IE = 0;                      // skrur av interrupt
            res2 = ReadCapture2();                    //henter resultat fra
timer1
            CCP2CONbits.CCP2M0 = 1;                   //setter capture modus
til rising edge
            u=0;                                            //oppdaterer
ultralyd status
```

```
            beregnAvstand();                        //beregner avstand
      }
      if(u==1)                                            //hvis u=1;
mottat rising edge fra ultralyd sensor
      {
            PIE2bits.CCP2IE = 0;                    // skrur av interrupt
            res1 = ReadCapture2();                  //henter resultat fra
Timer1
            CCP2CONbits.CCP2M0 = 0;                 //setter capture modus
til falling edge
            PIE2bits.CCP2IE = 1;                    //enabler interrupt
            PIR2bits.CCP2IF = 0;                    //nuller int flagg
            u=2;                                            //oppdaterer
ultralyd status
      }
}

void beregnAvstand(void)
{
      if(res1 > res2)                                     //hvis res1 er
større enn res2; betyr at timer1 har rullet over mens ultralyd sensoren
ventet på retur puls
      {
            tDiff = 65535-res1+res2;                //finner differansen
mellom res1 og res2
      }
      else
      {
            tDiff = res2 - res1;                    //finner differansen
mellom res1 og res2
      }
      avstand = tDiff * 0.0172;                     //beregner avstanden i
cm

      if(avstand>=0 && avstand<=30) Write1USART(0x82);     //sender
avstand; delt opp i avstander på 30 cm
      if(avstand>30 && avstand<=60) Write1USART(0x83);
      if(avstand>60 && avstand<=90) Write1USART(0x84);
      if(avstand>90 && avstand<=120) Write1USART(0x85);
      if(avstand>120 && avstand<=150) Write1USART(0x86);
      if(avstand>150 && avstand<=180) Write1USART(0x87);
      if(avstand>180 && avstand<=210) Write1USART(0x88);
      if(avstand>210 && avstand<=240) Write1USART(0x89);
      if(avstand>240 && avstand<=270) Write1USART(0x8A);
      if(avstand>270 && avstand<=300) Write1USART(0x8B);
      if(avstand>300) Write1USART(0x8C);
}

//---------------------kommandosentral-----------------------------
void commandCenter(void)
{
      adresse = mottatt>>4;
      //henter ut adresse bits
      kommando = mottatt & 0b00001111;
      //henter ut kommando bits

      switch(adresse)
            //finner hvilken komponent som skal styres
      {
            case 0b0001:
                  //motor_fremdrift
```

```
            switch(kommando)
            {
                  case 0b0001:
      //motor frem
                        SetDCPWM3(325);
      //setter dc (dutycycle) slik at roboten går fremover; dc=1,3ms

                        Write1USART(0x11);
      //sier i fra at motor har startet
                        break;

                  case 0b0010:
      //motor stopp
                        SetDCPWM3(0);
      //stopper motor
                        Write1USART(0x12);
      //sier ifra at motor er stoppet
                        break;

                  case 0b0011:
      //motor rygg
                        SetDCPWM3(500);
      //setter dc slik at roboten går bakover
                        Write1USART(0x13);
      //sier ifra at roboten rygger
                        break;

                  default:Write1USART(0x1);
//ingen gyldig kommando
                        break;
            }
            break;

      case 0b0010:
            //motor_styring
            switch(kommando)
            {
                  case 0b0001:
      //retning frem
                        SetDCPWM4(300);
      //setter dc slik at servo peker fremover
                        Write1USART(0x21);
      //sier ifra at kommando er utført
                        break;

                  case 0b0010:
      //retning høyre
                        SetDCPWM4(400);
//setter dc slik at servo peker 45`til høyre
                        Write1USART(0x22);
      //sier ifra at kommando er utført
                        break;

                  case 0b0011:
      //retning venstre
                        SetDCPWM4(200);
      //setter dc slik at servo peker 45`til venstre
                        Write1USART(0x23);
      //sier ifra at kommando er utført
                        break;
```

```
                        default: Write1USART(0x1);
        //ingen gyldig kommando
                                break;
                        }
                        break;

                case 0b0011:
                        //motor_ultra
                        switch(kommando)
                        {
                                case 0b0001:
                        //retning frem
                                SetDCPWM5(300);

                                Write1USART(0x31);
                        //sier ifra at kommando er utført
                                break;

                                case 0b0010:
                //retning høyre
                                SetDCPWM5(400);

                                Write1USART(0x32);
                //sier ifra at kommando er utført
                                break;

                                case 0b0011:
                //retning venstre
                                SetDCPWM5(200);

                                Write1USART(0x33);
                //sier ifra at kommando er utført
                                break;

                                default: Write1USART(0x1);
        //ingen gyldig kommando
                                break;
                        }
                        break;

                case 0b1000: startUltra();
                                //ultralyd  her skal kall til ultralyd være
                        break;

        }
}
```

Vedlegg 8

Kildekode – posisjonering
(kontrollenhet)

```
#include <stdlib.h>           //importerer nødvendige bibiloteker
#include <pwm.h>
#include <delays.h>
#include <portb.h>
#include <timers.h>
#include <math.h>

#pragma config WDT = OFF
      //konfigurer mikrokontrolleren
#pragma config LVP = OFF

void sendPuls(void);                // prototyp
void puls(char a, char b);
void delay(int d);

int p, d;
int n = 2;              //antall UL sendere.
int k = 1;              //hvilken UL sender
char a,b;


void main(void)
{
      OSCCON = 0b01110010;                            //OSC 8MHz
      TRISC = 0;                                            //setter port c
til utgang
      TRISA = 0;
      ADCON1 = 0b00001111;                            //gjør ADC pinner til
IO pinner
      PORTC = 0;

      OpenTimer2(TIMER_INT_OFF & T2_PS_1_4 & T2_POST_1_1);
      //åpner timer to, brukes av PWM funksjonen

      SetDCPWM2(28);
                        //setter dutycycle
//    OpenPWM2(13);
      while(1)
      {
            while(k <= n)
            {

                  sendPuls();
                              //sender 1ms puls
                  delay(12);
                              //vent 12ms
                  ClosePWM2();
                              //skru av IR
                  delay(40);
                              //vent 40ms, refleksjoner har forsvunnet
                  k++;
                              //bytter UL sender
            }
            k = 1;
            delay(800);
                              //vent 800ms, skaper en periode på ca 1s


/*          PORTC=0b00000001;
            OpenPWM1(12);
            PORTC=0b00000000;
```

```
            OpenPWM1(12);*/
//          puls();


      }
}

void sendPuls(void)
{
      if(k==1)                                          //UL sender 1
      {
            a=0b00000001;                               //RA0
            b=0b00000010;                               //RA1

            OpenPWM2(13);                               //starter IR led
            puls(a,b);                                  //40 kHz puls på 1 ms
      }
      if(k==2)                                          //UL sender 2
      {
            a=0b00000100;                         //RA2
            b=0b00001000;                         //RA3
            OpenPWM2(13);                         //starter IR led
            puls(a,b);
      }
/*    if(k==3)
      {
            PORTCbits.RC0 = 1;
      //skru på IR
            //send puls på UL3
            puls();
      }
      if(k==4)
      {
            PORTCbits.RC0 = 1;
      //skru på IR
            //send puls på UL4
            puls();
      }*/
}

void puls(char a, char b)
{
      while(p < 40)
      {
            PORTA = a;
            Delay10TCYx(1); Delay1TCY() Delay1TCY() Delay1TCY()
Delay1TCY();            //12,5 us delay
            Delay1TCY() Delay1TCY() Delay1TCY() Delay1TCY() Delay1TCY();
            p++;
            PORTA = b;
            Delay10TCYx(1);Delay1TCY() Delay1TCY() ;
                        //12,5 us delay
      }
      p = 0;
}

void delay(int d)
{
      if(d==12)                                                     //12 ms
delay;
      {
```

```
            T0CON = 0b00001000;
      //oppretter timer0 8 bits
            TMR0H = 0xA2;
            TMR0L = 0x40;
            T0CONbits.TMR0ON = 1;                           //skrur på
timer0
            while(INTCONbits.TMR0IF == 0);                  //venter 12ms
            T0CONbits.TMR0ON = 0;                           //skrur av
timer0
            INTCONbits.TMR0IF = 0;                          // senker Int
flagg
      }

      if(d==40)                                                   //40ms
delay
      {
            T0CON = 0b00000000;
      //oppretter timer0 16 bits, prescaler 4
            TMR0H = 0x63;
            TMR0L = 0xC0;
            T0CONbits.TMR0ON = 1;                           //skrur på
timer0
            while(INTCONbits.TMR0IF == 0);                  //venter 45ms
            T0CONbits.TMR0ON = 0;                           //skrur av
timer0
            INTCONbits.TMR0IF = 0;                          // senker Int
flagg
      }
      if(d==800)                                                  //800ms
delay
      {
            T0CON = 0b00000100;
      //oppretter timer0, 16 bits, prescaler 16
            TMR0H = 0x3C;
            TMR0L = 0xB0;
            T0CONbits.TMR0ON = 1;                           //skrur på
timer0
            while(INTCONbits.TMR0IF == 0);                  //venter 800ms
            T0CONbits.TMR0ON = 0;                           //skrur av
timer0
            INTCONbits.TMR0IF = 0;                          // senker Int
flagg
      }
}
```

Vedlegg 9

Kildekode – posisjonering
(mottaker)

```
#include <stdlib.h>           //importerer nødvendige bibiloteker
#include <delays.h>
#include <portb.h>
#include <timers.h>
#include <capture.h>
#include <math.h>

void chk_isr(void);                 //prototype
void beregnAvstand(void);
void beregnPosisjon(void);

int k,  i, tDiff=0, n, s, j, l, m;
unsigned int res1, res2, res3,irON;
char res = 0;
float avstand1, avstand2, a1, a2, Sp, a, x, y, d1, d2, z2, temp1, temp2;

#pragma config WDT = OFF
#pragma config LVP = OFF

#pragma code My_HiPrio_Int=0x0008              //ved høyprioritets
interrupt gå til denne adressen
void My_HiPrio_Int (void)
{
      _asm GOTO chk_isr  _endasm
}
#pragma code
#pragma interrupt chk_isr

void chk_isr(void)
{
      if(PIR2bits.CCP2IF==1)                          //hvis mottatt
ultralydpuls
      {
            PIE2bits.CCP2IE = 0;                      //"skrur av" ultralyd
mottakere
            PIR2bits.CCP2IF = 0;                      //nuller int flagg ccp
2
            res2 = ReadCapture2();                    //henter verdi fra
timer1
            j++;

            beregnAvstand();                          //beregner avstand

            if(k==1) Delay100TCYx(10);                //mottvirker
refleksjoner
            if(k==2)
            {
                  beregnPosisjon();                   //beregner posisjon
            }
            PIR2bits.CCP2IF = 0;                      //nuller int flagg ccp
2
      }

      if(PIR1bits.CCP1IF==1)
      {
            if(CCP1CONbits.CCP1M0==0)                       //falling edge
            {
                  PIE2bits.CCP2IE = 1;                      //"skrur på"
ultralyd mottakere
                  PIR2bits.CCP2IF = 0;                      //nuller int
flagg ccp 2
```

```
                  res1 = ReadCapture1();                    //henter verdi
fra tmr1
                  CCP1CONbits.CCP1M0 = 1;                   //setter til
rising edge
                  PIR1bits.CCP1IF = 0;                      //nuller int
flagg ccp 1
                  k++;                                            //hvilken
sender
                  m++;
                  PIR2bits.CCP2IF = 0;                      //nuller int
flagg ccp 2
            }

            else if(CCP1CONbits.CCP1M0==1)                        //rising
edge
            {
                  CCP1CONbits.CCP1M0 = 0;                   //setter til
falling edge
                  PIE2bits.CCP2IE = 0;                      //"skrur av"
ultralyd mottakere
                  PIR2bits.CCP2IF = 0;                      //nuller int
flagg ccp 2
                  PIR1bits.CCP1IF = 0;                      //nuller int
flagg ccp 1
                  l++;
                  if(k>=n) k=0;                                   //hvis
alle ultralyd pulser er sendt.
            }
      }
}

void main(void)
{
      OSCCON = 0b01100010;                            //OSC 4MHz

      OpenTimer1(TIMER_INT_OFF & T1_16BIT_RW & T1_SOURCE_INT & T1_PS_1_1 &
T1_SOURCE_CCP);               //åpner timer1; 16 bits teller.

      INTCONbits.GIE = 1;                                   //globald
interrupt enable
      INTCONbits.PEIE = 1;                            //peripheral interrupt
enable

      TRISCbits.TRISC1 = 1;                           //RC1 og RC2 til
inngang
      TRISCbits.TRISC2 = 1;
      TRISCbits.TRISC0 = 0;


      TRISAbits.TRISA1 = 1;                           //Inngang til
komparatoren
      TRISAbits.TRISA2 = 1;
      TRISAbits.TRISA5 = 0;                           //settes til utgang.
utgangen til komparatoren

      OpenCapture1(CAPTURE_INT_ON & C1_EVERY_FALL_EDGE);                //IR
      OpenCapture2(CAPTURE_INT_OFF & C2_EVERY_FALL_EDGE);
      //ultralyd

      PIR2bits.CCP2IF = 0;                      //nuller int flagg ccp 2
      PIR1bits.CCP1IF = 0;                      //nuller int flagg ccp 1
```

```
	ADCON1 = 0b00001010;
	CMCON = 0b00110011;                                //setter opp
komparatoren

	l=0;
	m=0;
	j=0;
	n = 2;
                        //antall sendere
	k = 0;
	s = 100;
                    //avstanden mellom sendere cm
	z2 = 40000;
                    //høyde til sendere i annen cm

	avstand1=0;
	avstand2=0;

	while(1)
	{

	}

}


void beregnAvstand(void)
{
	if(res1 > res2)
		//hvis res1 er større enn res2
	{
		tDiff = 65535-res1+res2 - 200;

	}
	else
	{
		tDiff = res2 - res1 - 200;
	}

	if(tDiff<12000)
	{
		if(k==1) avstand1 = tDiff * 0.034;                    //finner
avstanden til sender 1
		if(k==2) avstand2 = tDiff * 0.034;                    //finner
avstanden til sender 2
	}

}

void beregnPosisjon(void)
{
// ultrlydsendere er plassert høyere en sensor
/*	a1 = avstand1*avstand1;
	a2 = avstand2*avstand2;

	d1 = sqrt(a1-z2);
	//finner sidene av trekanten
	d2 = sqrt(a2-z2);
```

```
      Sp = (s+d1+d2)*0.5;
      //finner semi perimeter
      a = sqrt(Sp*(Sp-s)*(Sp-d1)*(Sp-d2));
      //finner arealet av trekanten
      x = 2*(a/s);
      //finner x koordinaten
      y = sqrt((d1*d1)-(x*x));
      //finner y koordinaten
*/

//sendere er plassert på lik høyde med sensor
      Sp = (s+avstand1+avstand2)*0.5;                          //semi
perimeter
      a = sqrt(Sp*(Sp-s)*(Sp-avstand1)*(Sp-avstand2));     //finner arealet
      x = 2*(a/s);
      //finner x koordinaten
      y = sqrt((avstand1*avstand1)-(x*x));                     //finner y
koordinaten

}
```