NTNU
Det skapende universitet

Eurobot 2009

Christian Wegger Kjølseth
Øystein Wergeland

Master i teknisk kybernetikk
Oppgaven levert: Juli 2009
Hovedveileder: Sverre Hendseth, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Oppgavetekst

Det skal designes og bygges en robot som skal delta i Eurobot Open 2009 i La Ferté-Bernard, 21- til 24. Mai. Dette innebærer:

- Utvikling av et beaconbasert deteksjonssystem for lokalisering av motstander til enhver tid, deteksjon av vinkler til posisjoneringsbeacons og deteksjon av flyttbar dispenserbeacon.
- Å benytte tidligere utviklet testrobot for navigasjonstesting parallelt med utvikling og sammensetning av årets konkurranserobot.
- Utvikling av et modulært kretskortsystem som muliggjør enkel utskiftning og plassering av benyttede kretskort.
- Utvikling av ny felles motordrivermodul for høyre og venstre motor med intern tilbakekobling.
- Implementering av en kunstig intelligens som kan ta i bruk informasjon fra sensorer og dynamisk velge optimal strategi ut fra dette.
- Viderutvikling og implementasjon av antikollisjonssystem som kan planlegge og velge optimale kollisjonsfrie ruter på brettet.
- Bygging av nytt konkurransebord.
- Organisering og oppfølging av to Eksperter i Team gruppers arbeid med å lage en modul for opplukking, bygging, og levering av templer på tilgjengelige poengområder.
- Å tilrettelegge for neste års deltagelse ved å lage moduler som fungerer og kan brukes videre, dokumentere alle utviklede moduler på ett sted, og videreføre kunnskap direkte til de personene som skal overta.

Oppgaven gitt: 08. februar 2009
Hovedveileder: Sverre Hendseth, ITK

Forord

Denne rapporten er i løpet av høsten 2009 skrevet av:

- Christian W. Kjølseth
- Øystein Wergeland

Det var av praktiske årsaker mest naturlig å levere en felles besvarelse, da oppgaven inneholdt mange samarbeidende elementer.

Vi vil gjerne benytte anledningen til å takke vår hovedsponsor, Kongsberg Gruppen ASA, for å ha bidratt med midler som har gjort det mulig å gjennomføre prosjektet.

I tillegg ønsker vi å takke deltagerne i Eksperter i Team, samt Glenn Angell for design og utvikling av tempelbyggermodul, Geir Mathisen for feilsøkingshjelp, komponentverkstedet for hjelp og støtte med komponenter, det mekaniske verkstedet for bruk av tid og materialer, og ikke minst Sverre Hendseth som har veiledet oppgaven.

Christian W. Kjølseth

Øystein Wergeland

Trondheim, 4. juli 2009

Sammendrag

Eurobot Open er en internasjonal robotkonkurranse for studenter og uavhengige organisasjoner, som arrangeres i Europa i mai hvert år. Institutt for teknisk kybernetikk har deltatt hvert år siden 2000, gjennom prosjekt- og diplomoppgaver i samarbeid med Eksperter i Team. I 2009 skal konkurransen foregå i La Ferté-Bernard, Frankrike, under tittelen "Temples of Atlantis". Robotens oppgave går i korte trekk ut på å plukke opp byggeklosser for å bruke disse til å bygge tempel, som senere kan plasseres på forskjellige poengområder med ulik høyde.

Dette arbeidet har hatt til hensikt å fullføre roboten ved å utvikle og produsere de forskjellige systemene som trengs for å delta i konkurransen. Gjennom prosjektoppgaven høsten 2008 ble det det utviklet en del basisfunksjonalitet, både software og hardwaremessig, som det nå er bygget videre på.

Det er utviklet et beaconbasert deteksjons- og posisjoneringssystem som gjør roboten i stand til å detektere motstanderrobotens posisjon til enhver tid, samt detektere vinkelen til tre posisjoneringstårn og en flyttbar dispenserbeacon med meget god oppløsning. Ved å kjenne til motstanderens posisjon er det tidligere utviklede antikollisjonssystemet tatt i bruk og gjør roboten i stand til å foreta meget smarte valg vedrørende kollisjonsunngåelse og rutevalg.

Det er foretatt et omfattende arbeid i å utvikle en ny kombinert motordrivermodul for høyre og venstre motor- og odometrienhet. I tillegg til tidligere funksjonalitet innehar denne modulen hastighetsregulatorer til de respektive motorene ved hjelp av tilbakekobling fra motorakslingene. Dette har medført at robotens evne til å kjøre langs rette linjer har økt betrakelig da sideavvik fanges opp momentant.

Det eksisterende rammeverket for kunstig intelligens er blitt brukt til å lage en ny implementasjon av strategisystemet som både har større funksjonalitet og er effektivt implementert. Ved å dele systemet inn i moduler og bruke høynivå funksjoner i strategiene er det enkelt å sette seg inn i virkemåten til systemet og videreutvikle dette. Det er også brukt en del ressurser på å vedlikeholde den generelle strukturen til programvaren og dokumentere eksisterende kode.

Nytt konkurransebord har blitt produsert da tidligere versjon var av så dårlig forfatning at roboten stadig ble stående å spinne. På bakgrunn av tilstanden til det eksisterende bordet, ble det foretatt nødvendige designvalg som sørger for at det nyutviklede bordet kan brukes til formålet i mange år fremover.

For å kunne teste navigasjonssystemet på et tidligere stadium enn foregående år, er det blitt tatt i bruk en tidligere utviklet testrobot. På denne måten har det vært mulig å bruke mer tid på utviklingen av konkurranseroboten uten at dette har medført tap av viktige og nødvendig testmuligheter.

For å kunne sørge for at de utviklede modulene samt softwaren benyttet skal kunne forstås, reproduseres, samt videreutvikles, er det blitt nedlagt en del tid på å utvikle to dokumentasjonspermer, henholdsvis dokumentasjon av software og hardware. Dette er gjort siden dokumentasjonen fra tidligere utviklede system opp igjennom årene er av svært variabel art og spredd utover utallige prosjekt og diplomoppgaver.

Det er lagt stor vekt på at alle moduler som er blitt utviklet skal være av god kvalitet, og at de skal kunne brukes videre i kommende eurobotkonkurranser. Dette gjelder både med tanke på håndtverk og design av modulene. Sett sammen med dokumentasjonsjobben som er gjort skal disse modulene kunne brukes videre slik de er, og være lett utskiftbare på grunn av den modulbaserte designen.

Underveis har det vært en del administrativt arbeide, hovedsakelig koordinering av ti studenter fra faget Eksperter i Team, samt en fulltidsansatt mekanikerlærling. I tillegg kommer elementer som økonomi, reise, rekruttering, samt deltagelse på flere arrangementer for å promotere eurobot, samt bidra til å inspirere unge til å velge realfag. Det er også lagt ned en del arbeid i å integrere de ulike delene sammen på roboten på en god, robust og effektiv måte.

Omfattende testing og resultater fra konkurransen viser at den utviklede roboten utfører de gitte oppgavene på en særdeles god måte. Roboten er i stand til å finne ledige poengområder og bygge templer som resulterer i svært høye poengsummer, samt unnvike motstanderroboten om denne skulle være i veien. Dette resulterte i at roboten, som den eneste med unntak av vinneren, gikk gjennom hele innledningsrunden uten å tape. Videre fortsatte den å seire både i åttendelsfinalen og kvartfinalen, slik at endelig plassering ble en meget sterk 4.plass av totalt 44 lag som hadde kvalifisert seg til den internasjonale finalen.

Innhold

Kapittel 1

Innledning

Denne rapporten beskriver arbeidet som er gjort med å utvikle de forskjellige modulene, samt den endelige sammenstillingen av konkurranseroboten som skal delta i Eurobot Open 2009. I tillegg til selve robotkonstruksjonen blir også administrative elementer beskrevet da disse i sum utgjør et vesentlig arbeid.

Oppgaven er først og frem veldig praktisk rettet, slik at mye av arbeidet dreier seg om design og utforming av de respektive moduler. Informasjon om den ferdige roboten fremkommer derfor av bilder, tester og resultater oppnådd under konkurransen.

Rapporten bør leses sammen med de to utviklede dokumentasjonspermene for henholdsvis hardware og software. Dette da rapporten hovedsakelig beskriver valgene og teknologiene benyttet, men refererer til dokumentasjonspermene der det er aktuelt for mer detaljert informasjon om modulens oppbygning og virkemåte.

Det har vært forsøkt å skape et godt skille mellom eksisterende og nyutviklede moduler, samt der man har fornyet enkeltelementer fra allerede eksisterende system. Siden arbeidet består i en del uavhengige oppgaver, er det valgt å dele rapporten inn i flere enkeltkapitler som hver inneholder en oversiktlig rapportstruktur.

Modulen som finner, plukker opp, bygger, samt leverer spilleelementene, er konstruert av ti studenter gjennom faget Eksperter i Team, og blir således ikke beskrevet i samme detalj som resten av systemet. Det blir dog referert til ekstern dokumentasjon der dette er aktuelt.

Hovedområdene i rapporten er beaconbasert posisjonerings- og deteksjonssystem, fremdriftssystem, kunstig intelligens, antikollisjon, robotkonstruksjon, og i tillegg kommer de administrative elementene samlet i kapittel diverse.

På et overordnet nivå finnes det kapitler som bakgrunn, testing og resultater, konklusjon og videre arbeid, der innholdet gjelder generelt for hele arbeidet.

Kapittel 2

Bakgrunn

2.1 Eurobot-konkurransen

Eurobot Open er en internasjonal robotkonkurranse for studenter og uavhengige organisasjoner fra hele verden. Konkurransen stammer fra Frankrike og arrangeres årlig i mai måned, et sted i Europa. Eurobot Open har blitt arrangert siden 1998 og NTNU, ved Institutt for teknisk kybernetikk, har vært representert med et lag hvert år siden 2000.

Lag fra i underkant av 30 nasjoner har deltatt de siste årene. Hver nasjon kan maksimalt stille med 3 lag, men inkludert de nasjonale kvalifiseringene har rundt 350 lag vært involvert hvert år. Frankrike er desidert størst med rundt 150 lag, men også andre land som f.eks. Tyskland og Italia har omfattende nasjonale kvalifiseringsrunder.

Institutt for teknisk kybernetikk har som tidligere nevnt representert NTNU i Eurobot siden 2000. Prestasjonene har vært noe varierende, men stort sett ganske bra, hvis man tar i betraktning lagstørrelse og ressursbruk i forhold til mange av de store utenlandske lagene. Mange av lagene har gjennomført flere runder med nasjonal kvalifisering og dermed kunnet videreutvikle sin robot. Dette har medført at nivået i finalene har vært veldig høyt. Tabell 2.1 viser NTNUs tidligere plasseringer i Eurobot (hentet fra [14] og [18]).

2.1.1 Reglement 2009

Reglene endres hvert år for å gi nye utfordringer til deltagerene, og årets konkurranse har fått navnet "Temples of Atlantis". Temaet er den mytiske sivilisasjonen som ifølge legendene var på havets bunn, og målet er å bygge tempel av sylindere og tverrliggere. Det er lagt opp til mulighet for samarbeid, ved at roboten kan bygge på motstanderens tempel, og i motsetning til fjorårets regler vil destruktiv adferd bli straffet hardt. Templene skal bygges på spesifikke poengområder, som har forskjellige former og høyder.

I denne konkurransen er det to roboter som konkurrerer mot hverandre på samme spillebrett. Spillebrettet har samme dimensjoner som tidligere år, 3,0

År/konkurransenavn	**NTNUs plassering**	**Antall lag i finalen (antall lag totalt)**
2000 Fun Fair	8.	13
2001 Space Odyssey	13.	18
2002 Flying Billiards	17.	25
2003 Heads or Tails	16.	32
2004 Coconut Rugby	21.	41
2005 Bowling	11.	50
2006 Funny Golf	44.	50 (350)
2007 Robot Recycling Rally	25.	39 (ca. 350)
2008 Mission to Mars	31.	39 (292)

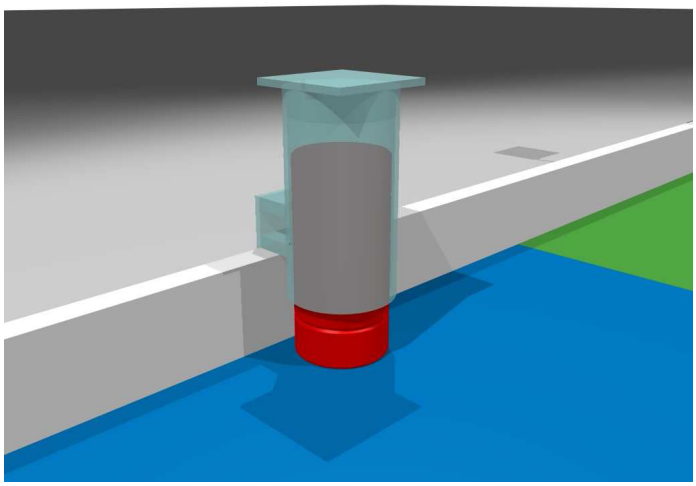
Tabell 2.1: NTNUs tidligere plasseringer i Eurobot

x 2,1 meter, der robotene starter i hvert sitt hjørne. Den største forskjellen på spillebrettet fra tidligere er poengområdene. Disse er plassert som et sirkulært område midt på brettet, og et rektangulært område langs den ene langsiden. Siden det vil bli høy straff for å rive ned templer som er bygget på disse områdene er det viktig å ikke krasje i disse områdene. Robotene skal operere fullstendig autonomt, og skal prøve å score så mange poeng som mulig innen tidsfristen på 90 sekunder. Når tiden er ute skal robotene stoppe av seg selv. Robotene får tildelt hver sin farge før start, enten rød eller grønn. De skal plukke opp spilleelementer i sin farge, som er plassert rundt på spillebordet og i faste dispensere, for så å bygge disse oppå hverandre på poengområdet. Poenggivningen er slik at jo høyere hvert element ligger, jo flere poeng vil elementet gi. Det er satt en begrensning på at roboten ikke får holde på mer enn 4 sylindere om gangen, men det er lov å holde så mange tverrliggere man vil.

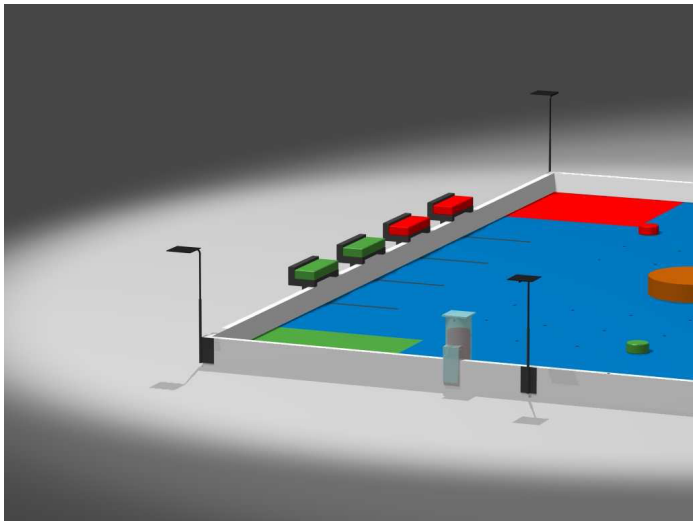
Det finnes to typer spilleelementer: sylindere og tverrliggere, som skal brukes til å bygge templene på de forskjellige poengområdene. Sylinderene kan plukkes opp på to forskjellige måter. Det finnes to dispensere langs kanten av spillebrettet som inneholder 5 sylindere hver. Den ene dispenseren har en fast posisjon, mens den andre kan befinne seg på en av to mulige posisjoner. Figur 2.1a viser hvordan disse dispenserene er utformet.

Den andre måten å få tak i sylinderene på er 12 markerte posisjoner for hver farge på spillebrettet. Her skal det plasseres til sammen 6 sylindere: 2 har fast posisjon, mens de 4 andre ligger i et av 10 forskjellige mønstre. Figur 2.2 viser fire av disse forskjellige mønstrene.

Tverrliggerene er plassert 2 stk av hver farge på faste lagringsplasser på brettet. Figur 2.1b viser hvordan disse henteområdene er konstruert. I tillegg til disse to tverrliggerene har hver robot lov til å starte med en tverrligger plassert inne i roboten. Figur 2.3 viser årets spillebrett med kun de faste sylinderene plassert, og dispenseren satt på en av de mulige posisjonene.

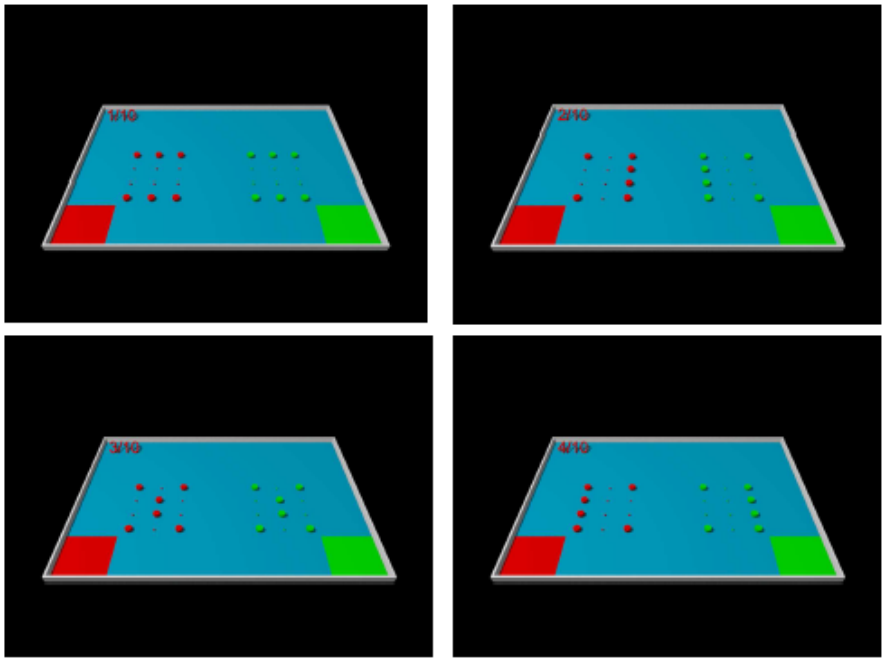


(a) Sylinderdispenser

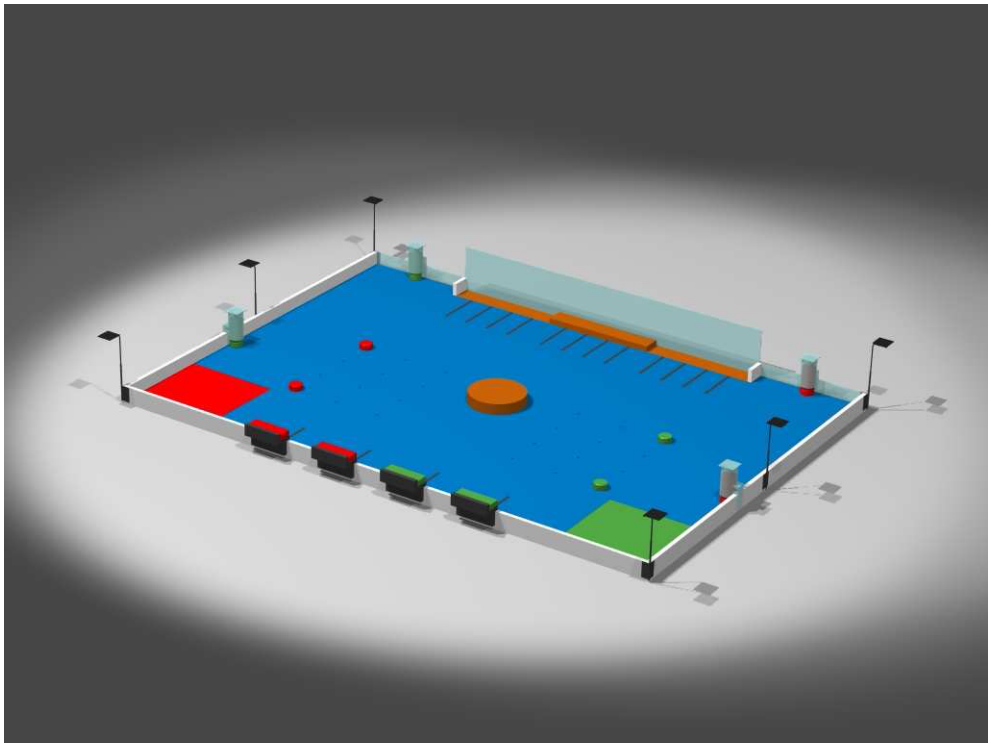


(b) Tverrliggerdispensere

Figur 2.1: Utformingen av dispensere på spillebrettet



Figur 2.2: Fire forskjellige oppsett for sylinderene på bordet

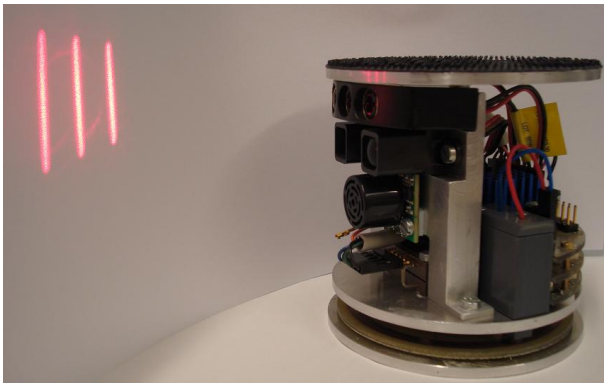


Figur 2.3: Illustrasjon av spillebrettet for Eurobot Open 2009

Kapittel 3

Beaconbasert posisjonerings- og deteksjonssystem

En autonom robot er avhengig av et godt posisjoneringssystem for å kunne navigere til nøyaktig ønsket destinasjon. Et slik system må i sanntid oppdatere robotens egen posisjon og retning i det lokale koordinatsystemet. Robotens posisjoneringssystem er basert på tidligere forbedret odometrimodul godt beskrevet i både [13] og [9], og blir dermed ikke ytterligere gjennomgått i denne rapporten.



Figur 3.1: Det roterende tårnet

Dette kapittelet går grundig gjennom hvordan flere kjente teknologier har blitt kombinert, og sammen sørget for posisjonering av motstanderrobot med rask oppdateringsfrekvens og god nøyaktighet i det lokale koordinatsystemet. På denne måten kan det tidligere utviklede antikollisjonssystemet beskrevet i [9] benyttes, og få en robot som er i stand til å ta smarte avgjørelser vedrørende valg av kjørebane og strategi, på et meget tidlig tidspunkt. I tillegg til å detektere og plassere motstanderbeacon, kan det roterende tårnet bli satt til å finne vinkelen til tre posisjoneringstårn samt en dispenserbeacon, som er plassert på kjente referanser rundt konkurransebordet. Absolutt posisjonering

kan dermed estimeres, samt deteksjon av den flyttbare dispenseren. Kapittelet beskriver i hovedsak bakgrunnen for teknologi og designvalg, mens informasjon om endelig fysisk løsning og utforming er grundig beskrevet i [10]

Det må innledningsvis nevnes at den eksisterende algoritmen for absolutt posisjonering ikke viste seg å fungere godt på hele spilleflaten, slik at det på nåværende tidspunkt kun eksisterer ferdig utviklet hardware for deteksjon av posisjoneringstårn.

3.1 Teori

3.1.1 Generelle posisjoneringsmetoder

Som det blir beskrevet i [2] eksisterer det i hovedsak syv forskjellige posisjoneringsmetoder, vist under, som kan benyttes på roboter, hvorav to er relative og fem er absolutte posisjoneringssystemer.

- Relativ posisjonering
  - Odometri
  - Inertial navigasjon
- Absolutt posisjonering
  - Magnetisk kompass
  - Posisjoneringstårn
  - GPS
  - Landemerke gjenkjenning
  - Modellgjenkjenning

Odometri

Ved odometri avleses bevegelsesinformasjon fra flere løpehjul ved kjente tidsintervaller. Bevegelsesendringen som da fremkommer integreres videre opp til posisjonsendring som benyttes for posisjonsestimering. Dette kan gjøres svært nøyaktig over korte tidsperspektiv, med billige komponenter og med høy oppdateringsfrekvens. Metoden er svært mye brukt som posisjoneringssystem for innendørs roboter, men den har sine svakheter. En liten konstant feil økes uten øvre avgrensning etter integrasjonen og sørger for akkumulerende avvik over tid. Det er spesielt avvik i robotens avleste retning som sørger for feil i robotens sidebevegelse, en feil som øker proposjonalt med tilbakelagt distanse. I tillegg kommer faktorer som spinn, glieffekt, fysiske hindringer som dumper og sprekker, ulik diameter på avlesningshjul og unøyaktig akselavstand. Metoden er derfor best egnet til innendørs bruk og på rene flater som ikke er glatte.

Inertial navigasjon

Inertial navigasjon bruker gyroskop og akselerometere til å måle rotasjonsrate og akselerasjon, for deretter å estimere posisjonsendringen ved å integrere opp henholdsvis en eller to ganger. Disse systemene har fordelen med at de er selvfungerende og således ikke trenger eksterne referanser, men ulempen er akkumulerende avvik over tid. En liten konstant feil økes også her uten øvre avgrensning etter integrasjonen, og gjør spesielt akselerometere uskikket for robotposisjonering. Gyroskop blir funnet noe mer egnet men i likhet med aktuelle akselerometere er prisen på slike system svært dyre.

Magnetisk kompass

Magnetisk kompass vil i utgangspunktet gi en sikker retningsindikator, en variabel som ved feil bidrar mest til det akkumulerende avviket som oppstår i et relativt posisjoneringssystem nevnt under odometri tidligere. Problemet med å basere seg på et magnetisk kompass er at jordens magnetfelt ofte er forstyrret i nærheten av kraftledninger og stålstrukturer, noe som gjør denne teknologien sårbar og upålitelig innendørs.

Posisjoneringstårn

Posisjoneringstårn ved kjente referanser som posisjoneringssystem brukes i stor grad idag både i fly og båtindustrien, samt til å posisjonere innendørs roboter. Disse posisjoneringsenhetene står fast på kjente posisjoner og kan detekteres på en enkel og nøyaktig måte, enten ved å finne vinkelen til dem, kalt triangulering, eller ved å finne avstanden til dem, kalt trilaterering. Informasjonen kan deretter brukes til å estimere posisjonen. Ved trilaterering vil det ikke være mulig å estimere rotasjonsretningen som roboten måtte inneha, en parameter som kun kan finnes ved triangulering sett fra robotens eget koordinatsystem. Ulempen med denne teknologien er at den krever et større og komplisert system, samt mer vedlikehold. I tillegg vil nøyaktig plassering av tårnene, samt fri sikt til dem, være en forutsetning for at absolutt posisjonering kan benyttes.

GPS

"Global Positioning System" benytter seg av trilaterering, som tidligere nevnt betyr avstandsestimering til tre eller flere kjente posisjoner, i dette tilfellet satelitter i omløp rundt jorda. Systemet fungerer svært godt for posisjonering av objekter utendørs, men kan ikke benyttes innendørs. I tillegg estimeres robotens rotasjon ved å sammenligne nåværende og forrige måling, slik at objektene som skal posisjoneres er avhengig av tilstrekkelig forflyttning og hastighet for nøyaktig retningsestimering. Systemet har en oppdateringsfrekvens på ett sekund og en nøyaktighet på ca en meter.

Landemerkegjenkjenning

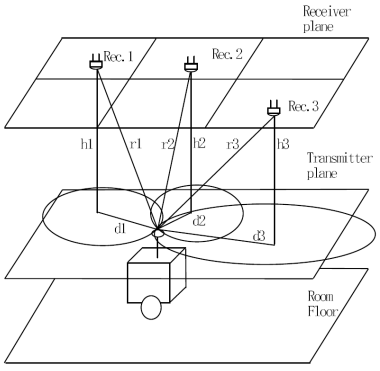
Landemerkegjenkjenning benytter seg av ett eller flere kameraer til å kjenne igjen spesielle strukturer som rektangler, linjer eller sirkler. Systemet fungerer godt om forholdene er kjent, med det menes farger på gjenstander, størrelser på objekter, samt like lysforhold i miljøet der det benyttes. Svakheten ligger i bruk av ressurser, samt at det ved endring av miljø er nødvendig med kamerakalibrering.

Modellgjenkjenning

Modellgjenkjenning går ut på det samme som landemerkgjenkjenning, men i denne situasjonen er landermerkene kjent på forhånd i en ferdig konstruert modell. På bakgrunn av sensorinformasjon fra henholdvis ett eller flere kamera posisjoneres roboten i modellen. Store og tunge matematiske operasjoner er nødvendig for å analysere og knytte bildene opp mot modellen, og således krever denne metoden mest regnekraft av de nevnte metodene.

3.1.2 Posisjonering basert på infrarødt lys og ultralyd

Studie [6] beskriver godt hvordan infrarødt lys i kombinasjon med ultralyd kan benyttes til å posisjonere en robot brukt i hjemmet. Systemet er bygget opp av en multimodul plassert på roboten som kan sende modulert ultralyd og infrarødt lys samtidig, samt motta informasjon over radio. I taket er det montert minimum tre mottagere på kjente plasser, som alle er i stand til å detektere det modulerte lyset og ultralyden. Disse er også i stand til å sende informasjon videre til en datamaskin ved hjelp av radio. Datamaskinen foretar videre en posisjonsestimering basert på den mottatte informasjonen, og sender posisjonskoordinater tilbake til roboten ved hjelp av radio. Figur 3.2 viser oversikt over systemet.



Figur 3.2: Posisjonering av roboten ved hjelp av IR, ultralyd og radiokommunikasjon

Informasjonen som benyttes til posisjonsestimeringen er avstanden mellom roboten og de forskjellige mottagerne i taket. Avstandene blir estimert ved at multimodulen på roboten sender et infrarødt lysblink samtidig med en ultralydsekvens. Grunnet lysets hastighet vil lysblinket nå alle mottagerne tilnærmet samtidig og trigge deres tellere til å starte. Når ultralyden ankommer en mottager vil telleren stoppe, og ved følgende formler kan avstanden estimeres.

$$t = n\frac{1}{f} - t_d \quad \text{s} \tag{3.1}$$

$$c(T) = 331,3176\sqrt{1 + \frac{T}{273,15}} \quad \text{m/s} \tag{3.2}$$

$$r = tc(T) \quad \text{m} \tag{3.3}$$

Hvor t = tid mellom sendt og mottatt ultralyd, n = tellerverdi, f = tellerfrekvens, t_d = deteksjonsforsinkelse, c = lydhastighet, T = temperatur og r = avstand.

Ved å studere figur 3.2 er både h_i og r_i kjent, og ved projeksjon av r_i utspinnes det i locuser hvor d_i kan finnes ved pytagoras.

$$d_i = \sqrt{r_i^2 - h_i^2} \tag{3.4}$$

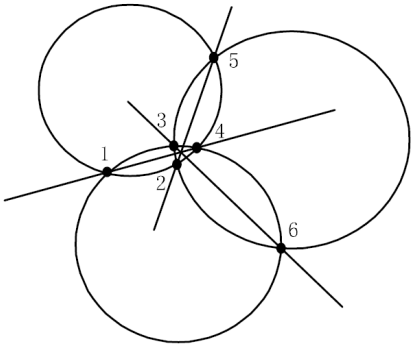
Kryssningspunkt mellom to eller flere av disse locusene representerer mulige posisjoner roboten kan befinne seg i, og disse blir funnet ved analytisk geometri.

$$(x_1 - x_r)^2 + (y_1 - y_r)^2 = d_1^2 \tag{3.5}$$

$$(x_2 - x_r)^2 + (y_2 - y_r)^2 = d_2^2 \tag{3.6}$$

$$(x_3 - x_r)^2 + (y_3 - y_r)^2 = d_3^2 \tag{3.7}$$

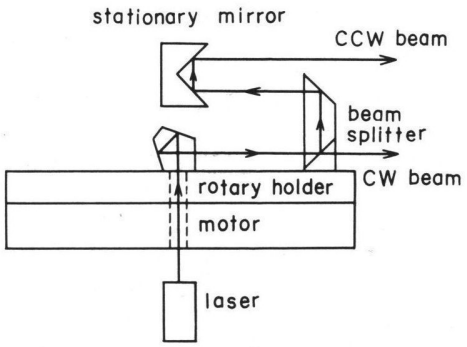
Hvor x_i, y_i er de kjente koordinatene til mottagerne, d_i er radius til locusene, og x_r,y_r er estimert posisjon til roboten. Resultatet er seks mulige posisjoner vist i figur 3.3. Grunnet tilfeldig avvik ved avstandsestimering er ikke posisjon 2, 3 og 4 plassert på nøyaktig samme sted, men ved å benytte minste kvadraters metode [4], kan et godt posisjoneringsestimat finnes ved å benytte et punkt som har minimum avstand til samtlige seks kryssningspunkter.



Figur 3.3: Mulige posisjoner

3.1.3 Posisjonering basert på symmetrisk roterende lasere

Denne fremgangsmåten er ment for å posisjonere en robot som beveger seg over et stort, flatt område og er hentet fra [17]. Konseptet går ut på å benytte to posisjoneringstårn som begge sender ut to roterende laserstråler i forskjellig høyder, se figur 3.4. Den nederste laserstrålen roterer med klokken, mens den øverste laserstrålen roterer mot klokken.

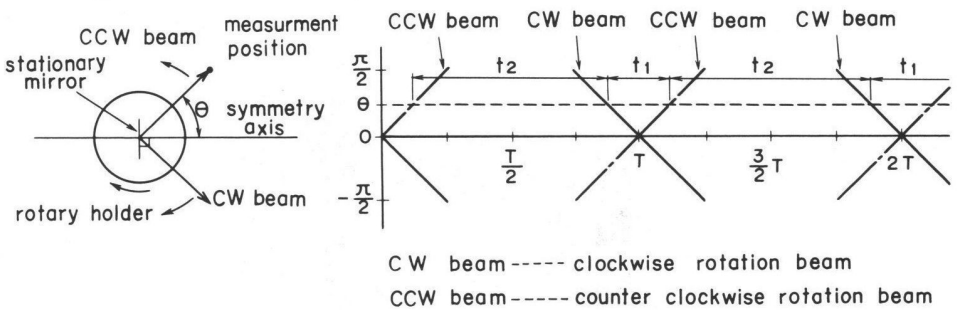


Figur 3.4: Roterende tårn med to symetriske lasere som roterer hver sin vei

Ved å ha en laserdetektor plassert på roboten som er i stand til å detektere den lave og den høye laserstrålen fra et tårn, vil det ved bruk av tellere kunne estimeres hvilken vinkel roboten har i forhold til tårnet. t_1 representerer tidsintervallet mellom deteksjonen av CW (med klokken) laser og CCW (mot klokken) laser, mens t_2 representerer tidsintervallet mellom deteksjonen av CCW laser og CW laser. Relasjonen mellom t_1, t_2 og dreining θ er vist i figur 3.5 og er basert på ligning 3.8 og 3.9

$$\theta = \frac{t_1}{t_1 + t_2} \quad \text{1. og 2. kvadrant} \tag{3.8}$$

$$\theta = \frac{-t_2}{t_1 + t_2} \quad \text{3. og 4. kvadrant} \tag{3.9}$$



Figur 3.5: Sammenheng mellom t_1, t_2 og dreining θ

På bakgrunn av ligning 3.8 og 3.9 eksisterer det to løsninger, og det må derfor stadfestes på et annet vis hvilken side av symmetriaksen til posisjoneringstårnet roboten befinner seg. Dette kan enten gjøres ved at laserdetektor på roboten er i stand til å detektere rotasjonsretningen til laserne, eller ved å plassere posisjoneringstårnet slik at kun en løsning vil være gyldig til enhver tid. Et eksempel på det sistnevnte kan være å plassere et posisjoneringstårn inntil en sidekant av robotens arbeidsfelt, med henholdvis første og andre kvadrant rettet ut mot arbeidsfeltet. Ved deteksjon av laser på roboten vil eneste løsning på θ være ligning 3.8.

3.1.4 Avstandsmåling basert på ultralyd

Ultralydsensorer har vært benyttet i utallige systemer i mange år, og produseres i flere forskjellige forpakninger og med forskjellig frekvenser og strålingsmønstre. Konseptet beskrevet grunding i [12], går ut på å sende langsgående trykkbølger ut i et medium som kan bære dem, hvorpå objekter med større dimensjoner enn trykkbølgens bølgelengde, vil reflektere trykkbølgen tilbake. Denne reflekterte trykkbølgen kalles et ekko, og avstanden til objektet kan finnes ved å ta tiden fra signalet ble sendt, til echoet kom tilbake. Frekvensene benyttet varierer fra 20kHz som befinner seg rett over det hørbare området, og helt opp til 1MHz, og innehar ulike egenskaper. Som nevnt finnes det utallige varianter og valg av riktig ultralydenhet varierer med applikasjonen den skal implementeres i [11].

For å kunne benytte tiden det tok fra signalet ble sendt til ekkoet ble mottatt er det nødvendig å kjenne hastigheten til trykkbølgen, heretter referert

til som lyd, i transmisjonsmediumet som benyttes. I eurobot vil transmisjonsmediumet til lyden være luft, og hastigheten avhenger av temperaturen vist i påfølgende ligning.

$$c(T) = 331,3176\sqrt{1 + \frac{T}{273,15}} \quad \text{m/s} \tag{3.10}$$

Videre endres bølgelengden som funksjon av hastighet og frekvens gitt av ligning 3.11.

$$\lambda = \frac{c(T)}{f} \quad \text{m} \tag{3.11}$$

En tredje faktor som er viktig å ta hensyn til er friksjonen som skapes av transmisjonsmediumet. Signalet som sendes ut må være sterkt nok til å traversere frem og tilbake den gitte avstand uten å bli borte på veien grunnet dempning. Dempningen er avhengig av frekvens, temperatur og luftfuktighet, men verdien av temperatur og luftfuktighet som gir maks dempning er ikke de samme for alle frekvenser. For eksempel vil lydfrekvenser på 40kHz få maks dempning grunnet luftfuktighet ved 100% RH (relativ luftfuktighet), mens lyd på 125kHz vil få maks dempning ved 50% RH. Siden en ultralydmodul skal kunne operere under alle typer luftfuktighet og ved forskjellige temperaturer, benyttes største dempningsverdi som luftfuktighet og temperatur kan påføre signalet. Dermed blir et godt estimat for dempning av lydsignaler gitt av følgende ligninger hentet fra [11].

Frekvenser under 50kHz:

$$\alpha(f) = 10^{-6} f \quad \text{dB/ft} \tag{3.12}$$

Frekvenser mellom 50 til 300Khz:

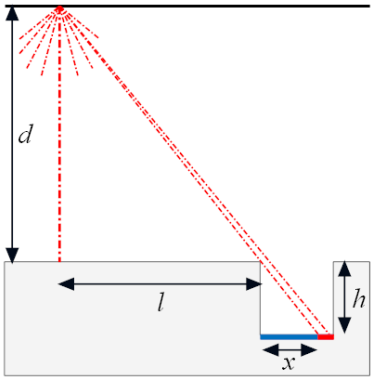
$$\alpha(f) = 22 \cdot 10^{-6} f - 0.6 \quad \text{dB/ft} \tag{3.13}$$

Overstående ligninger viser at dempningen øker vesentlig med frekvensen, noe som fører til at valg av frekvens er meget viktig i forhold til hvilke avstander som skal estimeres.

En siste faktor er at støyen reduseres betraktelig desto høyere frekvens som benyttes. Dette er et resultat av at det er generelt færre kilder som genererer høyfrekvent lyd, samt at når høyfrekvent lyd først er sendt, dempes denne raskt bort. På denne måten blir valg av ultralydenhet som avstandsestimator en balansegang mellom ønsket rekkevidde og støyrobusthet.

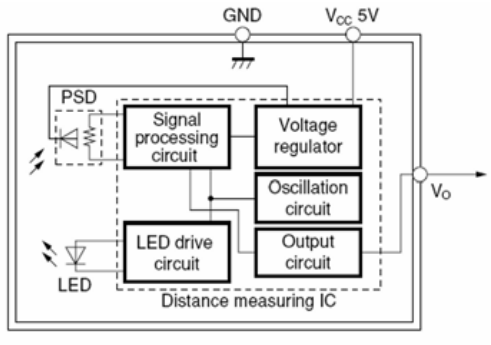
3.1.5 Avstandsmåling basert på infrarødt lys

En form for infrarød avstandsmåling er fremstilt i figur 3.6 og går ut på å sende modulert infrarødt lys rett frem. Når lyset treffer et objekt splittes lyset og en del reflekteres tilbake til en PSD (Position Sensing Detector) plassert nederst i et hulrom. Avstandene som resulterer i at PSD-sensoren enten blir truffet over hele linsen, eller ikke truffet i det hele tatt bestemmes av parametrene l og h, og må tilpasses til hvilket rekkeviddeintervall som er ønskelig å bruke. En signalprosessor, se figur 3.7, estimerer hvor stor del av PSD-sensoren som blir truffet av det modulerte IR-lyset, på figur 3.6 referert til som x. Basert på formel for trekantlikhet gitt i ligning 3.14, kan ulineræ analog distansemåling avleses på utgangen V_o



Figur 3.6: Prinsipp for avstandsestimering ved hjelp av IR og PSD-sensor

$$\mathrm{d} = \frac{lh}{x} \tag{3.14}$$

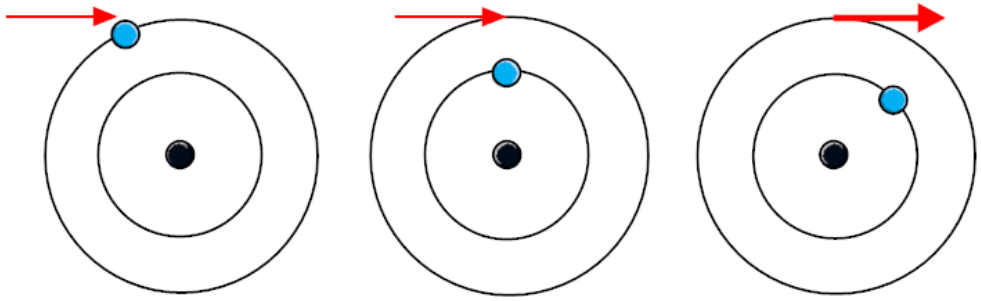


Figur 3.7: Oppbyggning av IR avstandssensor

3.1.6 Laserteknologi

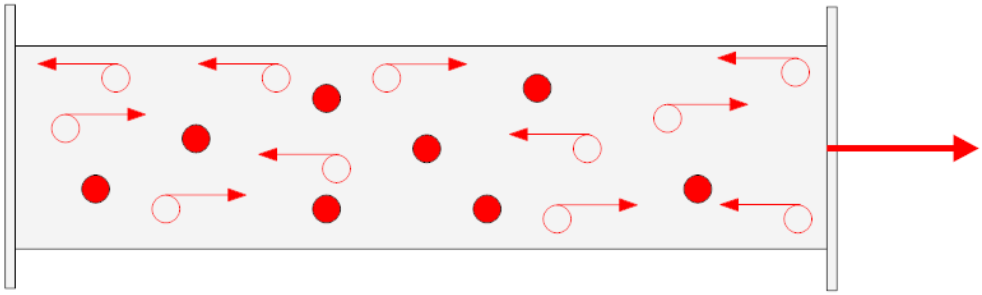
Lys skapes av at atomer i et stoff sender ut lysfotoner under gitte betingelser. Disse betingelsene blir i [22] beskrevet som stoffenes egne ønsker til å lette på trykket, lett sammenlignbart med for eksempel glødende metall og en lyspære. Glødende metall, samt en lyspære er gode eksempler på atomer som sender ut lysfotoner i tilfeldige retninger, og ved ulike tidspunkt.

Ved laserlys derimot, vil alle atomene avfyre sine fotoner på samme tid, og i samme retning. Dette skjer ved at det benyttes en metode beskrevet i [16]. Prinsippet går ut på at fotoner fra et atom som passerer et annet, inspirerer det passerte atomet til å også sende ut fotoner i samme retning, se figur 3.8.



Figur 3.8: Fotonutsending fra atom inspirert av forbipasserende foton

Ved å la mange slike prosesser foregå lukket inne i et rør som tilføres energi med speil på hver ende, se figur 3.9, vil det skapes mye lys som reflekteres frem og tilbake, og igjen inspirerer nye atomer til å slippe løs sine fotoner. Ved å la det ene speilet ikke være perfekt reflekterende, men la noe av lyset bli sluppet igjennom, vil en svært fokusert lysstråle sendes ut. Det er dette prinsippet som benyttes for å skape laserlys.



Figur 3.9: Laserprinsipp

Klassifisering av lasere

Bruksområdene til en laser varierer stort, fra å bli benyttet i enkle laserpenner til å kutte ut store metallseksjoner med en nøyaktighet på èn μm. Intensiteten i lyset kan dermed bli så sterk at den medfører en helserisiko for omkringstående personer som kan bli truffet av laserstrålen. På bakgrunn av dette er lasere delt inn i forskjellige klasser, hvor klassene representerer hvilken helserisiko laseren medfører ved feil bruk, da spesielt ved treff mot øyne.

Laserklasser:

- **1**

  Ufarlig

- **1M**

  Ufarlig, men farlig dersom laserstrålen passerer magnetiske eller optiske instrumenter som mikroskop eller teleskop

- **2**

  Ufarlig ved tilfedig treff på $\leq$ 0,25s

- **2M**

  Ufarlig ved tilfedig treff på $\leq$ 0,25s, men farlig dersom laserstråle passere magnetiske eller optiske instrumenter som mikroskop eller teleskop

- **3R**

  Kan være skadelig, men det er lav risiko

- **3B**

  Skadelig, men refleksjoner fra matte underlag er ufarlig

- **4**

  Uansett skadelig og brannfarlig

3.2 Bakgrunn

3.2.1 Eksisterende system

Odometri

Som nevnt innledningsvis har det tidligere blitt utviklet et posisjoneringssystem basert på odometri, godt beskrevet i både [13] og [9]. Dette systemet består av software, hardware, samt den mekaniske oppbygningen til modulen.

Softwaren som omdanner rotasjon fra løpehjulene til et posisjonsestimat fungerer meget bra og trenger ingen yttligere utvikling. Hardwaren fungerer også godt, men består av to store separate deler som opptar unødvendig mye plass, og bør derfor endres. Den mekaniske oppbyggningen blir sett på som tilfredstillende, men at det er svært vanskelig å oppnå skikkelig friksjon mellom løpehjul og underlag, samt at det eksisterer dødgang. I tillegg er det liten akselavstand mellom løpehjulene som bidrar til dårlig oppløsning på estimert vinkeldreining.

Absolutt posisjoneringssystem

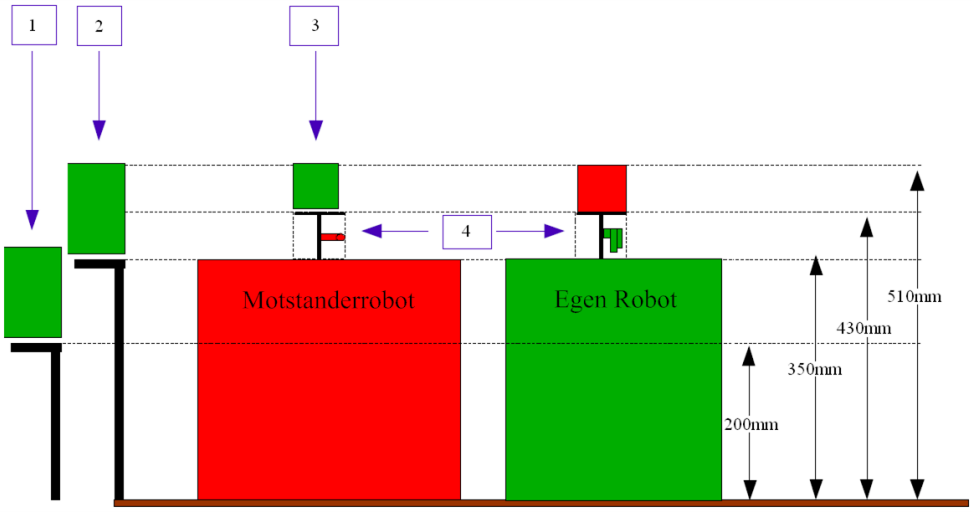
Det har gjennom flere år blitt forsøkt utviklet et absolutt posisjoneringssystem basert på å benytte tre kjente referanser rundt konkurransebordet. På disse posisjonene er det blitt plassert enheter med infrarøde lysdioder som sender modulert lys utover spillebordet, som igjen detekteres av et roterende tårn på egen robot. Dette systemet har kun basert seg på infrarødt lys som deteksjonsprinsipp, et konsept som har vist seg å ikke fungere tilfredstillende. Dette grunnet at systemet er svært sensitivt for utvendig lysstøy, samt at det bruker lang tid på å finne referansepunktene. Hele systemet er derfor forkastet mens den negative erfaringen ved bruk av infrarød deteksjonsmetode er tatt med videre.

Software til dette systemet, som i hovedsak består av en trianguleringsalgoritme, har tidligere blitt utviklet og er beskrevet i [14], men har aldri blitt simulert eller testet.

3.2.2 Beacons i eurobot

En beacon er en generell beskrivelse av en ekstern enhet som kan plasseres på spesifiserte plasser på konkurransebordet. Disse enhetene kan enten være passive, med andre ord refleksbaserte, eller aktive med vilkårlig kommunikasjonsteknologi, så lenge den ikke medfører en helserisiko for personer i nærheten. Det kan utplasseres tre posisjoneringsbeacons, heretter kalt posisjoneringstårn, på kjente plasser rundt konkurransebordet. I tillegg kan det plasseres to dispenserbeacons, henholdsvis plassert på hver dispenser, hvorpå den ene er flyttbar, samt en motstanderbeacon som kan plasseres på motstanderroboten. Figur 3.10 viser de strenge størrelsesbegrensningene til en beacon, samt at

plasseringshøyden varierer med hvilken type beacon som skal plasseres. Hensikten med å bruke en beacon er å kunne detektere den, og således benytte informasjonen til posisjonsestimering av egen robot, motstander robot, eller dispenser.



Figur 3.10: Plassering av beacons

1. Dispenserbeacon 80x160mm
2. Posisjoneringstårn 80x160mm
3. Motstanderbeacon 80x80mm
4. Søkemodul 80x80mm

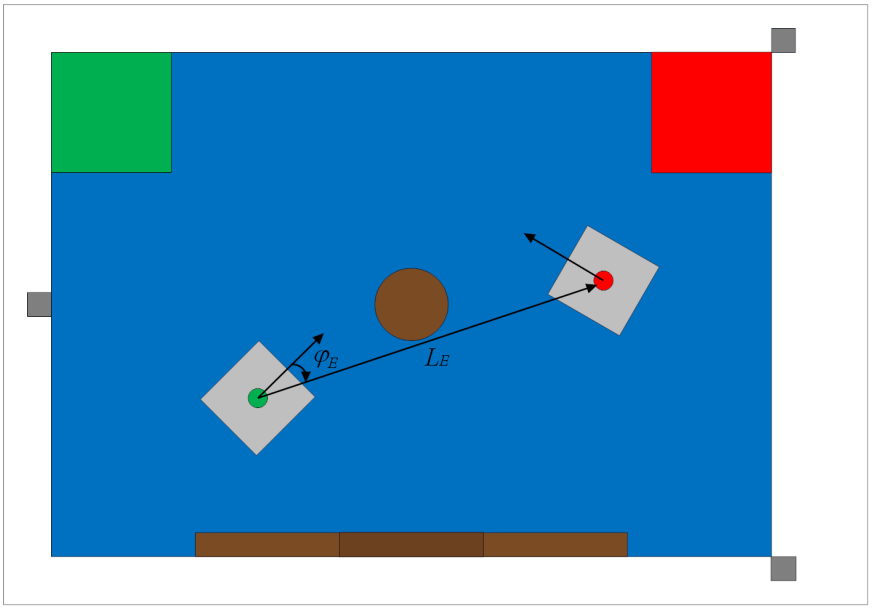
3.3 Design

3.3.1 Parametre posisjoneringssystemet må estimere

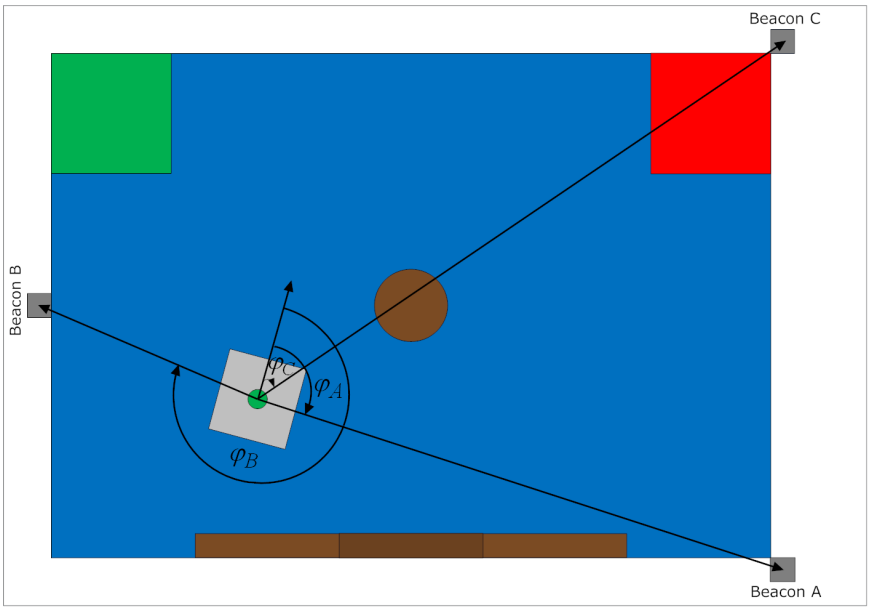
For å kunne ta i bruk det tidligere utviklede ruteplanlegginssystemet beskrevet i [9], er hyppige oppdateringer på hvor motstanderrobot befinner seg i forhold til egen robot en nødvendighet. I tillegg må systemet også kunne detektere vinkelen til samtlige posisjoneringstårn for å kunne benytte seg av absolutt posisjonering med rotasjonsestimering, se seksjon 3.1.1. Årets konkurranse inneholder i tillegg en dispenser som kan plasseres på to forskjellige steder, slik at systemet må være i stand til å detektere vinkelen eller avstanden til hvor denne befinner seg. Parametre systemet må være i stand til å estimere kan dermed summeres opp i tabell 3.1, og er visuelt fremstilt i henholdsvis figur 3.11, 3.12 og 3.13.

Finn motstander	*Finn egen posisjon*	*Finn dispenser*
φ_E L_E	φ_A φ_B φ_C	φ_{Da} φ_{Db} eller L_{Da} L_{Db}

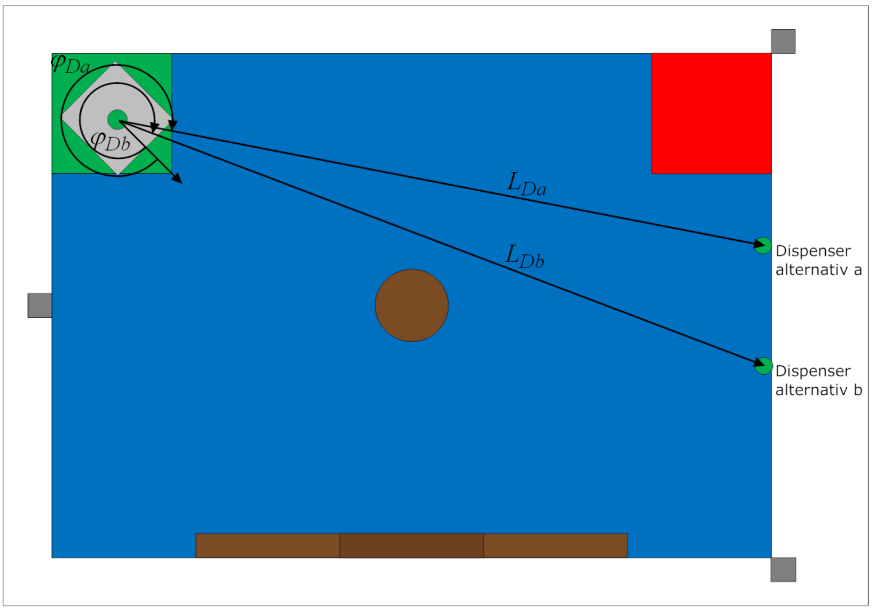
Tabell 3.1: Nødvendige parametre



Figur 3.11: Posisjonering av motstander



Figur 3.12: Absolutt posisjonering basert på posisjoneringstårn



Figur 3.13: Deteksjon av den flyttbare dispenseren under oppstart

3.3.2 Beacons benyttet i konkurransen

På bakgrunn av parametrene systemet skal være i stand til å finne, beskrevet i seksjon 3.3.1, er det funnet fornuftig å utvikle et system med 3 posisjoneringstårn, en dispenserbeacon på den flyttbare dispenseren, samt en motstanderbeacon plassert på motstanderroboten i bevegelse. Bakgrunn for valget er at enhetene muliggjør enkel deteksjon av de ønskede parametrene, og at disse også kan detekteres uavhengig hvor roboten befinner seg i forhold til dem. Da roboten har et fungerende posisjoneringssystem basert på odometri, samt at det legges til rette for absolutt posisjonering, vil det ikke være nødvendig å plassere en beacon på den stillestående dispenseren.

3.3.3 Metode for beacondeteksjon

Ved bruk av posisjoneringstårn rundt bordet, motstanderbeacon plassert på motstander og dispenserbeacon plassert på flyttbar dispenser, må det foregå en form for kommunikasjon som kan detektere disse. Søk etter litteratur på feltet viser i likhet med valg av posisjoneringssystem, at det ikke finnes en god rettesnor for å sammenligne de eksisterende løsningene da utganspunktet og begrensningene gitt av de forskjellige oppsett varierer stort. Med dette menes størrelse på søkemodul, signalstyrker, pris, gjeldende avstander og hvilke miljøer roboten skal operere i. I tillegg kreves det i denne sammenheng å finne den bevegelige motstanderen i tillegg til stillestående posisjoneringstårn og dispenser. På tross av dette kan allikevel nyttige enkeltobservasjoner fra forskjellige system ekstraheres og gi en god pekepinn på hva som er egnet teknologi i eurobotsammenheng.

Seksjon 3.1.2 beskriver hvordan infrarødt lys i kombinasjon med ultralyd og radiokommunikasjon kan brukes til å posisjonere en robot brukt innendørs. Denne teknologien fungerer svært godt til å posisjonere en innendørs robot, og kan lett modifiseres til å kunne bli tatt i bruk i eurobot. Problemet ligger i at systemet ikke er i stand til å kunne finne samtlige parametre beskrevet som nødvendige i seksjon 3.3.1. Systemet er ikke i stand til å detektere en eneste vinkel, men kun x- og y- posisjonen til roboten. Dette medfører at retningen til egen og motstanders robot vil være ukjent, og er grunnlaget for at dette systemet ikke er tatt i bruk. Hva som derimot kan stadfestes er at lys som triggerkilde, ultralyd som avstandsestimator, samt radio som kommunikasjonsstruktur kan gi et godt resultat.

Videre viser studie [24], at bruk av en spesiell og robust infrarød sensor til å estimere avstand, med minimal størrelse, rask hastighet, og lite strømforbruk, kan egne seg som en redundant avstandsmåler. All tidligere erfaring viser nytten av å gjøre slike system redundante, da ultralyd og infrarøde signaler er sårbare for interferens med andre ultralyd- og infrarøde kilder.

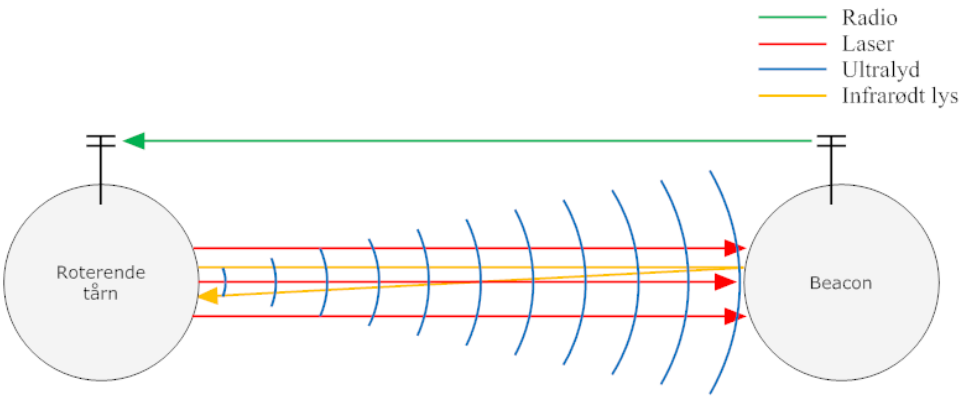
En tredje studie beskrevet i seksjon 3.1.3 tar for seg bruk av posisjoneringstårn med to roterende symmetriske lasere. I drift vil systemet være i stand

til å detektere vinkelen til roboten sett fra det aktuelle posisjoneringstårnet, og ved bruk av to tårn kan posisjonering av robot bli foretatt. Dette kan også implementeres lett i eurobotsammenheng, men lider samme skjebne som den kombinerte ultralyd/infrarøde teknologien. Det er kun x- og y-posisjonen som kan stadfestes, ingen retning på hverken egen eller motstanders robot. Dette da vinklene som estimeres er vinklene sett fra posisjoneringstårnene mot roboten, altså relativt til bordet, ikke relativt til egen robot. Det vil derfor ikke utgjøre noen forskjell om det adderes ekstra posisjoneringstårn inn i systemet, og gjør at systemet isolert sett er uegnet for posisjonering av egen og motstanders robot i eurobotsammenheng.

Allikevel viser studien, sammen med [16] at laserlys egner seg bedre som triggerkilde enn vanlig lys da intensiteten er mye sterkere grunnet parallelle stråler, samt at bølgelengdespekteret er mye smalere. Ved å benytte flere lasere, vil det være mulig å estimere vinkeldreiningen på det roterende tårnet mellom hvert treff ved å bruke tellere ved mottager, informasjon som kan være svært nyttig.

Oppsummering

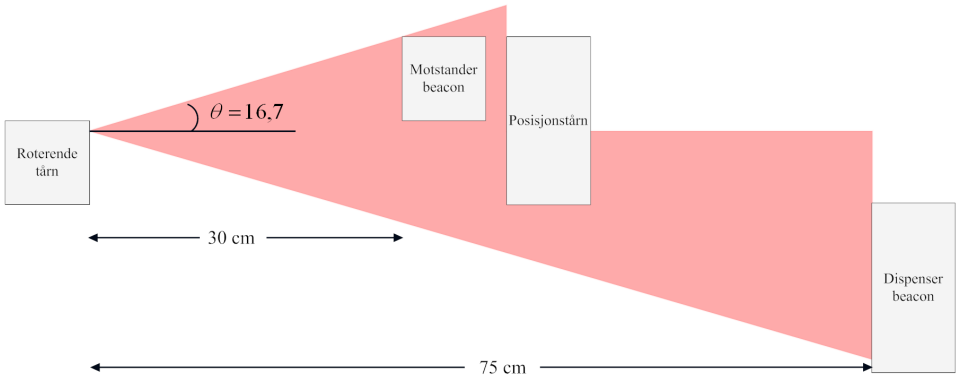
Ved å kombinere de tidligere nevnte positive faktorene fra overstående studier, samt foreta en tilpasning til forholdene i eurobot, er det blitt valgt en deteksjon og kommunikasjonsstruktur vist i figur 3.14. Denne baserer seg på lasere som triggerkilde, ultralyd, infrarødt lys og laser som redundante avstandsestimatorer, samt radiokommunikasjon for informasjonsflyt mellom enhetene. Når en beacon blir truffet av laserlys blir det umiddelbart sendt ut en melding via radiogrensesnittet om at aktuell beacon har blitt truffet. Om dette er dispenserbeacon eller et posisjoneringstårn vil kun en vinkelavlesning på det roterende tårn bli foretatt, mens ved treff av motstanderbeacon vil rotasjonen opphøre med påfølgende vinkelavlesing og avstandsestimering. På denne måten vil de nødvendige parametrene beskrevet i seksjon 3.3.1 kunne estimeres.



Figur 3.14: Deteksjon og kommunikasjonsstruktur

3.3.4 Laser som triggerkilde

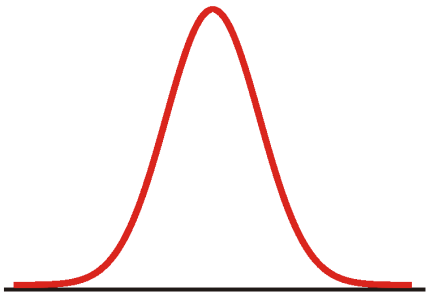
Enhetene som skal lokaliseres befinner seg på forskjellige høyder, tidligere vist i seksjon 3.2.2, og laserstrålen må derfor splittes tilstrekkelig for å kunne bli detektert. Figur 3.15 viser at spredningsvinkelen θ må være 33,4 grader om motstander skal kunne detekteres på en avstand på 30 cm og dispenserbeacon på en avstand på 75 cm. Posisjoneringstårnene står i samme høyde og kan således detektere laserlinjen uavhengig avstand.



Figur 3.15: Spredning av laser for å treffe enheter som skal lokaliseres

Gaussisk spredning og mindre spredningsvinkel

For å splitte laserstrålen benyttes en linse i forkant av laseren som har den egenskap at linjen som tegnes får sterkere intensitet på midten av linjen, og svakere intensitet desto lenger ut på sidene man kommer, se figur 3.16.

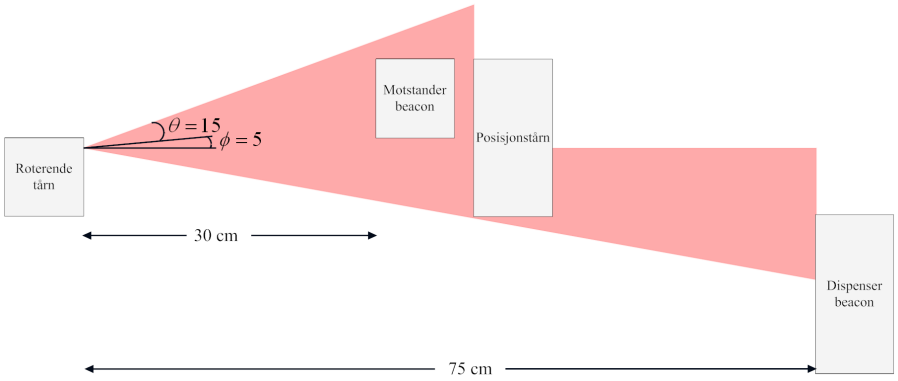


Figur 3.16: Laserintensitet langs linjen, gaussisk

Dette kalles gaussisk spredning og er et fenomen som er vanskelig å unngå i aktuelle prisklasser. Dette kan resultere i at laserlyset ikke blir detektert ved bruk av en spredningsvinkel på 33,4 grader, noe som er lite ønskelig spesielt når motstanderrobot kun befinner seg 30 cm unnna. Å øke spredningsvinkel er

heller ikke ønskelig da laserlinjegeneratorer leveres med en gitt totaleffekt, og større spredningsvinkel resulterer i tap av laserintensitet.

En aktuell løsning vil derfor være å minimalisere spredningsvinkel θ og samtidig rotere utgangsposisjonen ϕ grader slik som vist i figur 3.17. Roboten må således befinne seg lenger unna dispenserbeacon for å detektere denne, men dette har ingen praktisk betydning da deteksjonen av dispenser kan foretaes umiddelbart etter oppstart med en avstand på over to meter.



Figur 3.17: Minimal spredning av laser, samt sterkere intensitet på laser som treffer motstanderbeacon

Laserlinjegenerator FLL-3.5P-650-30

Det er strenge regler ved bruk av lasere i eurobot da disse ved for stor intensitet kan skade omkringstående deltagere og publikum. En lasers intensitet er delt inn i klasser, beskrevet i kapittel 3.1.6, hvorav klasse 1M er maksimal intensitet som er lov å benytte gitt av gjeldene regler i eurobot. Laserlinjegenerator FLL-3.5P-650-30 fra World Star Tech, se figur 3.18 er derfor valgt da denne er en av svært få laserlinjegeneratorer som befinner seg i denne klassen. Laserspredningen er valgt til 30 grader grunnet krav om deteksjon av forskjellige beacons beskrevet tidligere, og av praktiske årsaker er det valgt å benytte en bølgelengde på 650nm som er synlig rødt lys. Enheten er også svært liten slik at implementasjon i det roterende tårn vil gå smertefritt.

Figur 3.18: Laserlinjegenerator

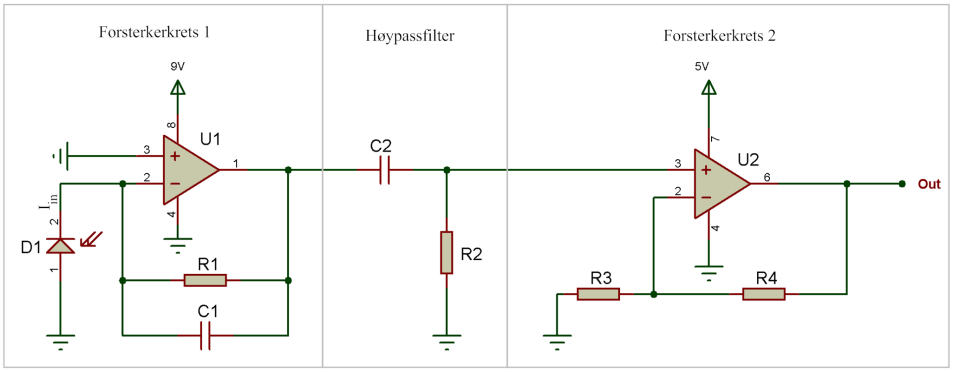
3.3.5 Deteksjon av laser

Laser er lys med sterk intensitet og med et smalt bølgelengdespekter og kan dermed detekteres enkelt ved hjelp av fotodioder eller fototransistorer. Fotodioder har sin styrke i svært rask responstid, men de krever ekstra elektronikk til å forsterke signalet samt omgjøre den svake strømmen til målbar spenning.

Fototransistorer slipper ekstra elektronikk da det skapes en naturlig forsterkning ved å la strømmen generert av lyset styre basen på transistoren. Denne naturlige forsterkningen resulterer i at det skapes mer ut av det samme signalet enn ved bruk av en fotodiode. Svakheten til en fototransistor ligger i treg responstid og er bakgrunnen for at fotodioder med tilhørende elektronikk er valgt som detektoroppbyggning i dette prosjektet.

Da en laser har smalt bølgelengdespekter er det ønskelig med en lysdetektor som kun detekterer lys innenfor disse bølgelengdene. Det er dessverre ikke mulig å skaffe en fotodiode som er i stand til å kun detektere bølgelengder innenfor et så smalt spekter, men i et stort område rundt. Det er derimot mulig å velge hvilken bølgelengde fotodioden skal generere mest strøm på, og således få maks forsterkning ved ønsket bølgelengde, og svakere forsterkning etterhvert som bølgelengden på det mottatte lys beveger seg bort fra denne. På bakgrunn av dette er fotodioden S1787-12 med maksimal forsterkning ved 650nm produsert av Hamamatsu valgt som lasersensor.

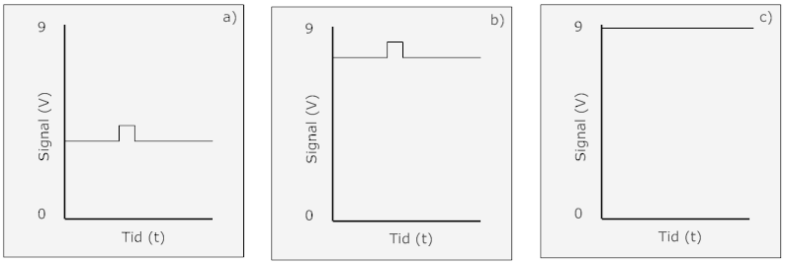
Tidligere nevnt krever en fotodiode ekstern elektronikk som kan forsterke den svake signalstrømmen til målbar spenning. Til dette er det blitt designet en elektronikkdel basert på tre deler, se figur 3.19, med bakgrunn i informasjon hentet fra [15]



Figur 3.19: Nødvendig elektronikk for å kunne detektere laserlys under støyfulle lysforhold

Forsterkerkrets 1

Forsterkerkrets 1 har som oppgave å forsterke det svake strømsignalet generert av fotodioden og omgjøre signalet til målbar spenning. Som tidligere nevnt genererer fotodioder strøm over et vidt bølgelengdespekter og således vil belysningen rundt sørge for en konstant forstyrrelse. Ved å forsterke signalet vil denne bakgrunnsbelysningen tre frem som et konstant DC-ledd, hvorpå signalet fra laser vil medføre addisjon av lys og en liten endring, se figur 3.20 a) og b). Siden bakgrunnstøyen også forsterkes er det viktig å minimalisere forsterkningen foretatt på dette trinnet, da operasjonsforsterkeren går i metning ved for mye bakgrunnslys, vist i figur 3.20 c), og ingen laserdeteksjon vil være mulig. Det fremkommer av figur 3.20 at en tilførselsspenning til operasjonsforsterkeren på 9 volt istedet for 5 volt, vil sørge for et større operasjonsområde med tanke på bakgrunnsstøy. Det er likevel viktig at det velges en operasjonsforsterker som tåler dette.



Figur 3.20: Spenningsignal ut fra forsterkerkrets 1. a) Lite støy b) Mye støy c) For mye støy

Omgjøring til målbar spenning avhenger av generert strøm og er gitt av følgende ligning:

$$V_{out} = I_{in}R_1 \tag{3.15}$$

I tillegg er det nødvendig med en kondensator C1 i pF-område for å stabilisere operasjonsforsterkeren. Dette da det opereres med svært lav inngangsstrøm, stor motstand og stor forsterkning, noe som kan få operasjonsforsterkeren til å oscillere frem og tilbake.

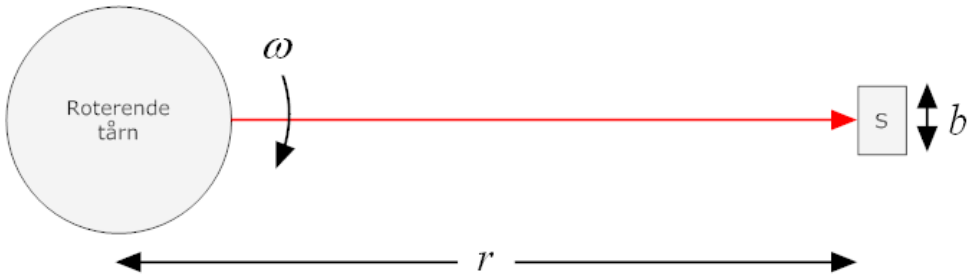
Høypassfilter

Høypassfilter er nødvendig for å fjerne støyen representert ved konstant DC spenning, se tidligere figur 3.20, samt andre forstyrrende lyskilder med lav frekvens. Frekvenser som passerer filteret er nødt til å være høyere en cutoff-frekvensen gitt av ligning 3.16

$$f_c = \frac{1}{2\pi R_2 C_2} \quad \text{Hz} \tag{3.16}$$

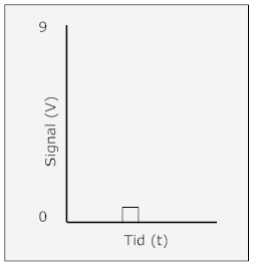
Det er videre viktig at denne cutoff-frekvensen velges lavere en frekvensen skapt når laser treffer laserdetektor. Figur 3.21 viser systemet sett ovenifra da en laser fra det roterende tårnet treffer fotodioden. Periodetiden på pulsen som blir skapt av at lasersensoren blir truffet av laserlys, avhenger av hvor raskt det roterende tårnet roterer, avstanden mellom tårnet og sensor, samt hvor bredt det aktive området på lasersensoren er. Frekvensen på signalet som skapes kan dermed finnes ved å benytte ligning 3.17.

$$f_l = \frac{\omega r}{b} \quad \text{Hz} \tag{3.17}$$



Figur 3.21: Systemet sett ovenifra idet laseren treffer fotodioden

f_c må derfor velges lavere enn f_{lmin} for å kunne utelukke at systemet ikke filtrerer ut deteksjonen med omkringliggende lavfrekvent støy. f_{lmin} finnes ved å bestemme minimum distanse r til motstanderroboten, samt minimum rotasjonshastighet ω det roterende tårnet skal benytte seg av. Signalet avlest på utgangen av dette trinnet påført signal a) eller b) fra figur 3.20, er fremstilt i figur 3.22.

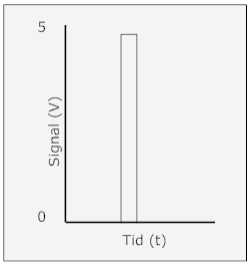


Figur 3.22: Signal i etterkant av høypassfilter

Forsterkerkrets 2

Forsterkrets 2 sørger for å forsterke signalet slik at det kan tolkes og bearbeides av en mikrokontroller. Det filtrerte signalets amplitude varierer avhengig av hvor langt unna, og på hvilken høyde detektoren befinner seg. Dette da laserintensiteten som nevnt tidligere avhenger av hvor på den spredte laserlinjen treffet oppstår, samt at laserens intensitet svekkes over lengre distanser. Dette gjør imidlertid ingen ting da støyen er filtrert bort, og det kan benyttes meget stor forsterkning som resulterer i at operasjonsforsterkeren går i metning uansett. Det er likevel viktig at spenningstilførselen på denne operasjonsforsterkeren er på samme nivå som mikrokontrolleren, i dette tilfellet 5 volt. Forsterkningen er gitt av ligning 3.18, og den resulterende pulsen sendt til mikrokontroller er vist i figur 3.23.

$$\frac{V_{out}}{V_{in}} = 1 + \frac{R4}{R3} \tag{3.18}$$

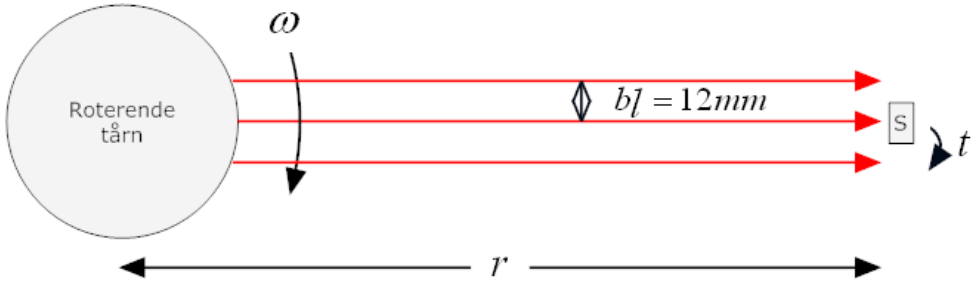


Figur 3.23: Signal sendt til mikrokontroller fra forsterkerkrets 2

Forhindring av feildeteksjon

Systemet beskrevet så langt fungerer under svært vanskelige lysforhold så lenge lysstøyen har en frekvens lavere enn f_c. Lys som derimot har høyere frekvens og befinner seg innenfor bølgelengdespekteret til laserdetektoren, kan resultere i at uriktig deteksjon forekommer. Lyskilder som kan ha disse egenskapene er hovedsakelig blitz fra diverse kameraer, samt lasere og lysglimt fra andre konkurrerende robotsystemer. Dette kan forhindres ved å benytte tre parallelle lasere, vist i figur 3.24, og måle tiden mellom treffene. For at deteksjon skal aksepteres må to lasere ha truffet lasersensoren innenfor et gitt tidsvindu. Det vil være laseren plassert i senter som genererer en akseptert deteksjon da denne laseren alltid vil være nummer to, uavhengig rotasjonsretning.

Størrelsen på tidsvinduet kan finnes ved følgende ligninger



Figur 3.24: Deteksjonssikkert system med tre parallelle laserlinjegeneratorer

$$t_{min} = \frac{b_l}{\omega r_{maks}} \quad \text{s} \tag{3.19}$$

$$t_{maks} = \frac{b_l}{\omega r_{min}} \quad \text{s} \tag{3.20}$$

Valg av verdier

For å ikke sette forsterkerkrets 1 i metning ved støyfulle forhold har kretsen blitt testet sammen med fotodioden S1787-12 fra Hamamatsu og det er funnet fornuftig å velge R1 og C1 til henholdsvis 100k og 22pF. Med disse verdiene er det kun direkte solbelysning som setter operasjonsforsterkeren i metning. Pulsene fra lasersensoren som skapes når laserstrålene passerer er i tillegg markante innenfor alle aktuelle distanser benyttet i eurobotkonkurransen.

Ved å måle minste rotasjonshastighet ω til 4,48 rad/sek (0,71RPS), og velge minimum distanse $r = 30$cm og aktiv bredde på fotodiode $b = 2$mm, kan minste frekvens på pulsen skapt av forbipasserende laser bli funnet til:

$$f_{lmin} = \frac{\omega_{min} r_{min}}{b} = 672\text{Hz} \tag{3.21}$$

Dermed kan f_c velges betydelig mindre enn f_{lmin} ved å sette $R_2 = 10$k og $C2$=100nF. Dette resulterer i:

$$f_c = \frac{1}{2\pi R_2 C_2} = 159\text{Hz} \tag{3.22}$$

Videre viser forsøk at en forsterkning på over 30 ganger resulterer i at forsterkerkrets 2 går i metning uansett størrelsen på pulsen skapt når laserstrålen passerer sensor ved aktuelle avstander. Ønsket forsterkning blir skapt ved å velge $R_3 = 5$k og $R_4 = 150$k.

$$\frac{V_{out}}{V_{in}} = 1 + \frac{R4}{R3} = 31 \tag{3.23}$$

3.3.6 Avstandsestimering

Som et resultat av seksjon 3.3.3 ble det valgt å benytte 3 redundante avstandsmålingsprinsipper da alle metodene isolert sett kan bli forhindret i å operere korrekt grunnet støypåvirkning. Andre elementer enn støybekjempelse som underbygger valget av flere redundante avstandsmålere er deres egenskaper i form av hvilke objekter som detekteres innenfor forskjellige områder. Ultralyd brer seg henholdsvis utover i rommet og fører til en deteksjon av objekter innenfor et dråpeformet område, mens laser og infrarødt lys finner objekter som befinner seg på samme høydenivå, og tilnærmet rett foran sensor. I eurobotsammenheng vil derfor ultralydmodulen måle avstand til det opphøyde poengområdet midt på konkurransebordet om motstanderroboten befinner seg på den andre siden, noe som ikke er ønskelig. En siste grunn til bruk av flere avstandsmålere er at de har sine styrker ved forskjellige avstander. Ultralyd er meget sikker ved korte distanser grunnet smalt dråpeområde, konseptet med infrarødt lys er godt ved korte og medium distanser, mens laserprinsippet benyttet er svært godt ved medium og store distanser. Således kan en votering eller algoritme som vekter de forskjellige avstandene gjennomføres før endelig avstand bestemmes.

Ultralydmodul LV-MaxSonar-EZ1

Valg av avstandsmåler basert på ultralyd er en balansegang mellom ønsket støyundertrykking og påkrevd rekkevidde. Desto høyere frekvens, jo mindre støypåvirkning, men også mindre rekkevidde. I tillegg må modulen være i stand til å måle temperaturen i miljøet sensoren skal operere i, grunnet varierende lydhastighet ved ulike temperaturer. Det er på bakgrunn av dette ultralydmodulen EZ1 produsert av MaxSonar er valgt til formålet, se figur 3.25. Denne opererer på 42kHz og kan derfor garantere en rekkevidde fra 15 til 645 cm, samtidig som en temperatursensor sørger for riktig kalibrering av lydhastigheten. Implementasjonsmessig er den også svært enkel da avstand avleses direkte fra en digital PWM (pulse width modulaton) utgang. Formel for omgjøring til avstand er gjengitt i ligning 3.24. Minimal størrelse er også viktig da størrelsen til det roterende tårnet er svært begrenset grunnet gjeldene regler.

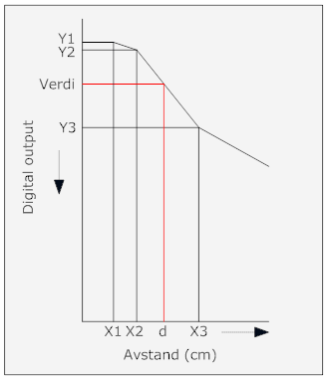
Figur 3.25: Ultralydsensor

$$d_u = \frac{pulsewidth(\mu s)}{57,874} \quad \text{cm} \tag{3.24}$$

Infrarød avstandsmåler Sharp GP2Y0A02YK0F

Sharp GP2Y0A02YK0F er en avstandsmåler som baserer seg på teknologien beskrevet i seksjon 3.1.5 og den blir sett på som den mest robuste avstandsmålermetoden basert på infrarødt lys. Sensoren har en garantert rekkevidde fra 0,2 til 1,5m og gir ut en analog ulineær spenning som representerer avstanden. Denne spenningen er funnet noe støyfull, slik at det er lagt inn en kondensator på $1\mu F$ mellom signalbane og jord, før signalet sendes til AD-konverter på mikrokontrolleren.

Den ulineære signalkurven som fremkommer deles opp opp i ni lineære linjer som gir en god tilnærming, hvorav et utdrag er gjengitt i figur 3.26. For fullstendig signalkurve, samt den stykkevise lineariserte kurven, henvises det til tillegg A.



Figur 3.26: Oppbyggning av stykkevis linearisert modell

Videre blir utregnet distanse d foretatt ved å finne nærmeste $Y_i \leq Verdi$ og $Y_{i-1} \geq Verdi$ med tilhørende X_i og X_{i-1}, og benytte ligning 3.25. Verdier på X_i og Y_i, samt diagrammer over ulineær samt stykkvis lineærisert måling er vedlagt i tillegg A

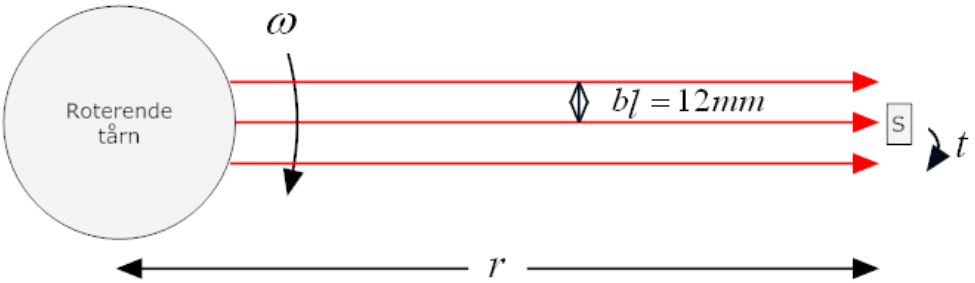
$$d_{ir} = \frac{Y_{i-1} - Verdi}{\frac{Y_{i-1}-Y_i}{X_i - X_{i-1}}} + X_{i-1} \quad \text{cm} \tag{3.25}$$

Avstandsmåling ved hjelp av deteksjonslasere

I studien beskrevet i seksjon 3.1.3 fremkommer det at ved å måle tiden mellom to lasertreff fra et roterende tårn med konstant hastighet, kan vinkelen tårnet har dreid i mellomtiden estimeres. Dette konseptet er blitt videreført og tilpasset systemet med parallelle lasere, og istedet for vinkeldreining, estimeres avstanden mellom roterende tårn og laserdetektor. Nødvendig informasjon er allerede tilgjengelig, da tiden mellom lasertreffene er blitt funnet på den aktuelle beacon. Denne tiden må være innenfor et gyldig tidsvindu beskrevet i

Figur 3.27: Infrarød avstandsensor

seksjon 3.3.5, og ved å sende over tiden i tillegg til treff id kan avstand estimeres ved ligning 3.26. Nøyaktig avstand kan kun oppnås når roterende tårn og aktuell beacon står stille seg i mellom, men ved avstander på over 1,5 meter er hastigheten på laseren så høy, at eventuell bevegelse på egen robot eller truffet beacon, kun vil resultere i en marginal distansefeil. På denne måten brukes systemet til å estimere avstand til motstanderbeacon, men kan også utvides til å finne distansen til posisjoneringstårn og dispenserbeacon om ønskelig.



Figur 3.28: Lasere som passerer sensor med tidsintervall t

$$d_l = \frac{b_l}{\omega t} \quad \text{cm} \tag{3.26}$$

Tolkning av avstandsinformasjon

De forskjellige avstandsteknologiene benyttet har som nevnt innledningsvis forskjellige styrker hva angår avstand, støyrobusthet, samt triggesensitivitet på andre omkringliggende elementer. Det er med bakgrunn i dette utviklet en softwarealgoritme, se algoritme 3.1, som er i stand til å finne og forkaste den sensoren som befinner seg lengst unna gjennomsnittet. Ved å videre ta gjennomsnittet av de gjenværende målingene, samt forrige avstandsestimat, vil ny benyttet avstand bli estimert. Dette er nødvendig da ultralydmodulen lett kan detektere byggeplassen på midten av konkurransebordet istedet for motstanderrobot, se seksjon 3.4.1, om motstander befinner seg i nærheten av byggeplassen. I tillegg vil feil som følge av støy på en sensor ikke medføre feilaktig avstandsestimering, da denne vil filtreres bort.

Input: measurement $dist$ of size $size$
Result: distance to opponent, $removed\_average$
$sum \leftarrow 0$;
for $i \leftarrow 0$ **to** $size$ **do**
  $sum \leftarrow sum + dist[i]$;
end
$average\_dist \leftarrow sum/size$;
$max\_deviation \leftarrow 0$;
$max\_deviation\_index \leftarrow 0$;
for $i \leftarrow 0$ **to** $size$ **do**
  $deviation \leftarrow fabs(average\_dist - dist[i])$;
  if $deviation > max\_deviation$ **then**
    $max\_deviation \leftarrow deviation$;
    $max\_deviation\_index \leftarrow i$;
  end
end
$sum \leftarrow sum - dist[max\_deviation\_index]$;
$removed\_average \leftarrow sum/(size - 1)$;

Algoritme 3.1: Algoritme for vekting av måleverdier fra forskjellige sensorer

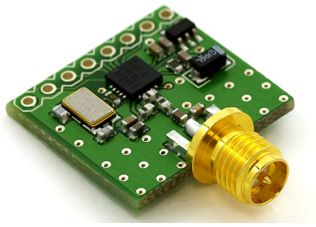
3.3.7 Radiokommunikasjon

Det roterende tårnet styres ved hjelp av radiokommunikasjon fra kommunikasjonsnode på egen robot. Det er via dette grensesnittet det roterende tårnet blir satt til å søke etter motstanderbeacon, posisjoneringstårn eller dispenserbeacon, hvorpå robot får tilbake informasjon om hvor motstanderbeacon, posisjoneringstårn eller dispenserbeacon befinner seg. Radiokommunikasjon er også avgjørende mellom de forskjellige beacons og det roterende tårnet, da det roterende tårnet blir underrettet om treff på aktuell beacon ved hjelp av dette grensesnittet.

Radiomodul nRF24L01

Radiokommunikasjon innebærer en viss form for risiko, da andre enheter som opererer på samme frekvens vil medføre tap av informasjon. Det er derfor viktig å kunne håndtere pakketap på en enkel og oversiktlig måte uten å belaste mikrokontrolleren for mye. I tillegg bør det være mulig å benytte frekvenser som berøres minst mulig av trådløse nettverk, mikrofoner, walkie talkier eller lignende. Ved oppdagelse av mye støy bør det være mulig å kunne endre frekvens på en enkel måte. Enheten må også ha svært raskt transmisjonstid, da denne tiden representerer en forsinkelse fra faktisk treff til systemet er i stand til å detektere det.

På bakgrunn av det overnevnte er det blitt valgt å benytte en trådløs transciever fra Nordic Semiconductor, se figur 3.29, som tillater frekvensbruk helt opp til 2525MHz, overføringshastighet på 2Mb/s, samt har flere programmerbare feilhåndteringsmekanismer.



Figur 3.29: Radio transceiver fra Nordic Semiconductor

Antennekonstruksjon

Transceiveren nRF24L01 fra Nordic Semiconductor trenger ekstern antenne for å kunne kommunisere med andre radionoder. Alle tilgjengelige antennealternativ medførte at radiomodulen opptok for mye plass på de respektive beacons, slik at deres størrelsesbegrensninger gitt av reglene ble overskredet. I tillegg er det viktig med omnidireksjonelle antenner da stillestående og bevegende radionoder på forskjellige høyder skal kunne kommunisere med hverandre.

Det ble derfor utviklet en egen kvartbølgeantenne basert på ligning 3.27

$$\lambda = \frac{c}{f} \tag{3.27}$$

hvor λ = bølgelengde, c = lysets hastighet, og f = frekvens. Ved å benytte $c = 3 \cdot 10^8$ og $f = 2500$MHz vil lengden på en bølgelengde være 120 mm, og således har utviklede kvartbølgeantenner en lengde på 30 mm. Disse er i tillegg svakt buet og montert slik at de vil være i stand til å sende radiobølger til alle andre radionoder i systemet uavhengig posisjon på de bevegelige enhetene.

Sende- og mottagermodus

Posisjoneringstårn, motstanderbeacon og dispenserbeacon er satt til å være i kontinuerlig sendemodus, og sender over treffinformasjon til det roterende tårn når et treff oppstår. Dette for å oppnå rask tilbakemelding om at den respektive beacon er truffet, siden skifte mellom sendemodus og mottagermodus representer en forsinkelse. Det roterende tårnet er hovedsakelig i mottagermodus, men endrer tilstand til sendemodus når motstander, posisjoneringstårn eller dispenserbeacon er lokalisert, avhengig av hva det roterende tårnet er satt til å søke etter. Informasjonen sendes til kommunikasjonsnode på egen robot som hovedsakelig også befinner seg i mottagermodus. Kommunikasjonsnode på egen robot endrer kun tilstand til sendemodus når en styringskommando skal sendes til det roterende tårnet.

Feilhåndtering

For å forhindre tap av pakker som følge av kollisjon med andre radiokilder som trådløse nettverk, mikrofoner, wakie talkie, eller lignende, er det valgt å benytte en frekvens på 2523MHz. Dette grunnet at de fleste andre kommersielle radiokilder opererer mellom 2400 og 2500MHz. Radionodene er programmert til å automatisk generere bekreftelse på mottatt pakke. Dersom bekreftelsen uteblir, vil radionoden som sendte, forsøke å resende pakken tre ganger med et tidsintervall på $250\mu s$. Om pakken etter tre resendinger ikke har blitt bekreftet mottatt, vil mikrokontroller bli underrettet og kunne forholde seg til situasjonen, som er avhengig av hvor feilen har oppstått.

- **Mellom beacon og det roterende tårn**

  Om en beacon ikke får bekreftet sin melding til det roterende tårnet om at den er truffet, vil den ikke forsøke å resende flere ganger, men gjøre seg klar til å bli truffet igjen som om ingen ting har skjedd. Dette grunnet at en forsinket pakke fra en beacon vil resultere i at det roterende tårnet avleser treffvinkel på feil tidspunkt, noe som ikke er ønskelig.

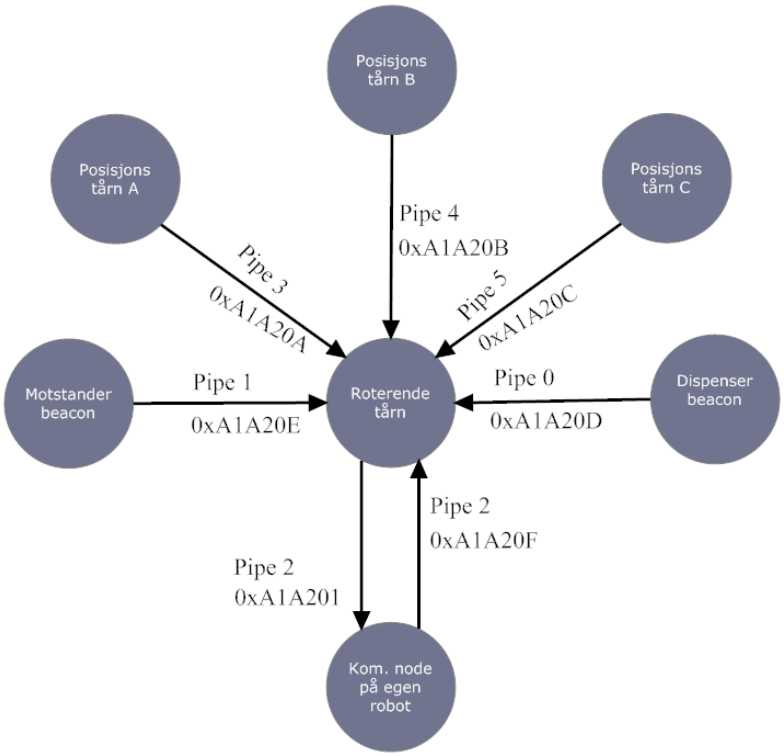
- **Mellom roterende tårn og kommunikasjonsnode på egen robot**

  Om en styringskommando blir sendt til det roterende tårnet fra kommunikasjonsnode samtidig som det roterende tårnet sender over treffinformasjon, vil begge nodene være i sendemodus og således ikke være i stand til å motta noe som helst. Dette er blitt løst ved at det roterende tårnet uansett status på sendt melding endrer tilstand tilbake til mottagermodus, mens kommunikasjonsnode sender melding til AI om at styringskommando ikke nådde frem, hvorpå AI setter igang en ny sending. Informasjonen det roterende tårnet ikke fikk sendt går dermed tapt, men dette representerer ingen svakhet da ny styringskommando i prinsipp er et spørsmål om å kunne motta annen type informasjon. Eksempelvis søker det roterende tårnet etter dispenserbeacon og skal oversende

dispenserens vinkel, samtidig som kommunikasjonsnode oversender kommando om å finne motstanderbeacon. Informasjon om dispenseren vil derfor ikke lenger være aktuelt og er grunnlaget for at det er valgt å la denne informasjonen gå tapt.

Pipes og adresser

Alle radionodene er i stand til å kunne motta informasjon fra seks andre noder samtidig, via adresserbare pipes. I systemet beskrevet tidligere er det kun det roterende tårnet og kommunikasjonsnode på egen robot som skal motta data, og således er det kun deres pipes som må adresseres. Figur 3.30 viser oversikt over pipes med tilhørende adresser, og hvilke noder som sender til dem.

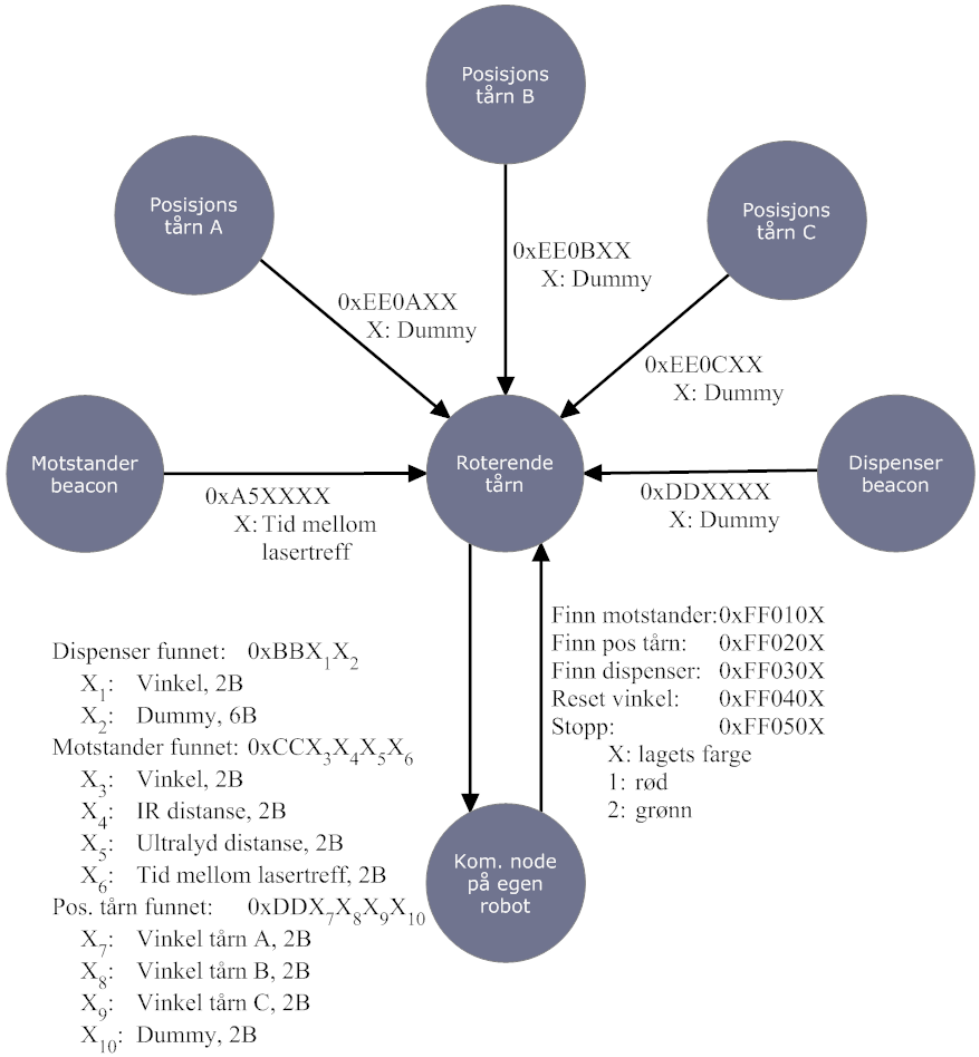


Figur 3.30: Mottagerpipes med tilhørende adresser

Overført data

Når laser blir detektert ved en beacon blir det umiddelbart utsendt en radiomelding om at den respektive beacon har blitt truffet. Om det er motstanderbeacon som er truffet vil radiomeldingen i tillegg inkludere tidsintervallet mellom de to lasertreffene som genererte den gyldige deteksjonen, beskrevet i seksjon 3.3.5. Det roterende tårnet overfører så videre informasjonen til kom-

munikasjonsnode på egen robot som oversetter informasjonen til CAN. Fra den andre siden sendes det styringskommandoer fra kommunikasjonsnoden til det roterende tårnet, som bestemmer om det roterende tårnet skal søke etter posisjoneringstårn, dispenserbeacon eller motstanderbeacon, samt om det skal foreta en nullstilling av vinkelavleser, eller stoppe opp. Av praktiske årsaker er det valgt å operere med en datalengde på 3B fra beacons til det roterende tårnet, og 9B fra det roterende tårnet til kommunikasjonsnoden. For detaljert informasjon over hva de forskjellige enhetene sender og mottar henvises det til [10], samt figur 3.31.



Figur 3.31: Data overført mellom enhetene

3.3.8 Utforming av det roterende tårn

Denne seksjonen lister opp de funksjonelle kravene satt til det roterende tårnet, se tabell 3.2, for at valgt løsning for beacondeteksjon funnet i 3.3.3 kan gjennomføres. Dette dreier seg hovedsakelig om krav til plassutnyttelse, slik at alle benyttede teknologier kan implementeres uten at modulen overskrider den strenge størrelsesbegrensningen på 80x80x80mm. I tillegg kommer softwarebaserte krav som kommunikasjon og kontrollerbarhet. Med unntak av krav 4 og 5 er kravene enten tidligere beskrevet eller av fysisk art, slik at for detaljert beskrivelse og skjemategninger over endelig løsning, henvises det til [10, kap 1].

1	Størrelse innenfor 80x80x80mm
2	Toveis radiokommunikasjon
3	Kontrollerbar via radio
4	Motor med konstant rotasjonshastighet
5	Mulighet for momentan rotasjonsstopp
6	Fri sikt foran sensorer
7	Laserlinjegeneratorer plassert så høyt som mulig
8	Uavhengig stillestående topplate
9	Enkle tilkoblinger
10	Plass til:
	3 laserlinjegeneratorer
	Ultralydmodul
	Infrarød modul
	Vinkelavlesningsmodul
	Radiomodul
	Roterende spenningstilførsel
	Motor med gir
	Kretskort
	Debuggingsmuligheter

Tabell 3.2: Funksjonelle krav til det roterende tårn

Motor med konstant rotasjonshastighet

Det er benyttet en svært liten motor med metallgir ved navn GM12a, se figur 3.32, som kan operere opp til 6V med 170RPM. Motoren er valgt da den innehar et gir som gjør den svært sterk, selv med liten størrelse og med et lavt strømforbruk, samt at den er svært enkel å montere.

Krav 6 og 8 medfører at det er nødvendig med stillestående aksling, og at motoren således er nødt til være plassert på platen som roterer rundt akslingen. Motoren vil dermed rotere seg selv og platen den står fastmontert på rundt den stillestående akslingen.

Figur 3.32: Motor GM12a

O-ring er valgt som overføringsmedium mellom motoren og den stillestående akslingen da tykkelse, strekkegenskaper, samt diameter lett kan endres. Messinghjulene montert på motorakslingen og den stillestående akslingen med et forhold på 1:1,8 sørger for en tilnærmet konstant rotasjonshastighet på 0,71RPS ved en tilført spenning på 5V. For mer informasjon samt kretskjema over motordriverkrets henvises leser til [10, kap 1].

Mulighet for momentan rotasjonsstopp

Ved deteksjon av motstander skal rotasjonen opphøre og avstandsmålinger foretas. Det er derfor svært viktig at rotasjonen opphører slik at infrarød- og ultralydmodul står i retning mot motstanderen. Motordriverkretsen benyttet har dessverre ingen form for "stop and hold" funksjon, slik at motoren fortsetter å rotere noen grader etter at spenningen fjernes. Det har derfor vært nødvendig å implementere en enkel PI-regulator som roterer tårnet tilbake til treffvinkelen før avstandsmodulene aktiveres. Forsinkelsen som da oppstår er minimal og har derfor ingen praktiske betydninger for systemet.

3.3.9 Utforming av beacons

Denne seksjonen går kortfattig igjennom grunnlaget for valg gjort under utvikling av de forskjellige beacons benyttet. I hovedsak dreier dette seg om kravene som stilles til de forskjellige enhetene slik at laserstrålen kan detekteres enten om beacon befinner seg i et stilletående hjørne som et posisjoneringstårn, eller er i bevegelse plassert på motstanderrobot. Derfor vil kravene til den respektive beacon bli gjennomgått i denne seksjonen, mens for detaljert beskrivelse og skjemategninger over endelig løsning henvises det til [10, kap 3,4,5].

Mens elektronikken, tidligere beskrevet i seksjon 3.3.5, er lik ved alle beacons, er plassering av lasersensor, samt fra hvilken vinkel sensoren skal bli truffet fra, vesentlig forskjellig for posisjoneringstårn, dispenerbeacon og motstanderbeacon. Dette setter krav til hvilket vinkelområde deres lasersensorer skal være i stand til å detektere laserlys fra, samt hvordan resten av modulen skal være oppbygd for at ikke unødvendig lysstøy skal tre inn i systemet og påvirke sensoren.

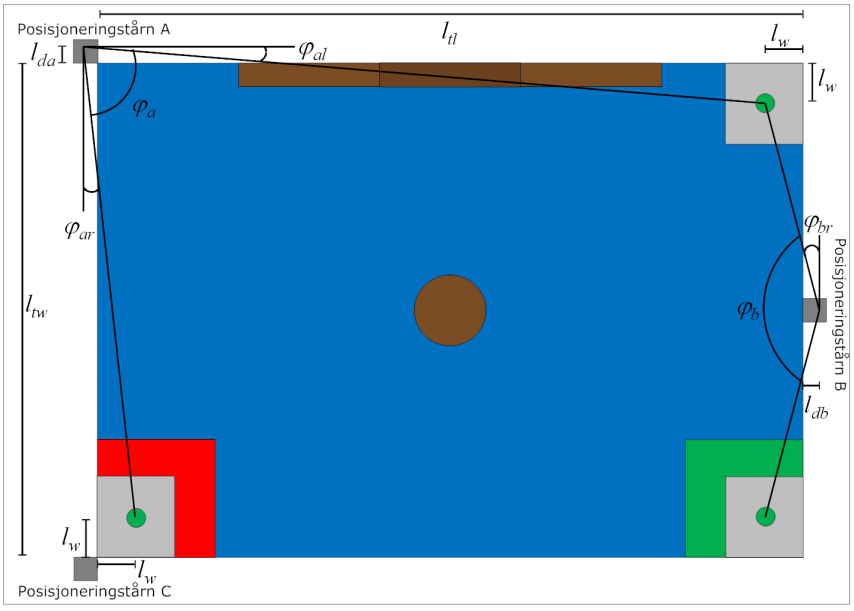
Posisjoneringstårn

Følgende krav ble stilt under utvikling av posisjoneringstårn

1 Størrelse innenfor 80x80x160mm
2 Enveis radiokommunikasjon
3 Fri sikt foran sensorer
5 Debuggingsmuligheter
6 Enkle i bruk
7 Støyforhindring
  Kun tillate vertikalt lys fra en spesifik høyde å treffe sensor
  Kun tillate horisontalt lys innenfor deteksjonsområde å treffe sensor

Tabell 3.3: Funksjonelle krav til posisjoneringstårn

Som det fremkommer av figur 3.33, befinner posisjoneringstårn A og C seg i to av konkurransebordet sine hjørner, mens posisjoneringstårn B befinner seg midt på den ene kortsiden. Lasersensor på posisjonstårn A og C må således være i stand til å detektere laserlys fra det roterende tårn innenfor et område på φ_a, mens lasersensor på posisjonstårn B er nødt til å kunne detektere laserlys fra et område på φ_b. Størrelsen på disse områdene kan finnes ved å benytte ligning 3.28 og 3.29.



Figur 3.33: Deteksjonsområde for posisjoneringstårn

$$\varphi_a \geq 90 - \arctan\frac{l_{da} + l_w}{l_{da} + (l_{tl} - l_w)} - \arctan\frac{l_{da} + l_w}{l_{da} + (l_{tw} - l_w)} \tag{3.28}$$

$$\varphi_b \geq 180 - 2\arctan\frac{l_w + l_{db}}{\frac{l_{tw}}{2} - l_w} \tag{3.29}$$

Ved å bestemme $l_{da} = 4{,}5$ cm, $l_{db} = 6{,}0$ cm, samt bruk av en minimal robot med $l_w = 15{,}0$ cm, må laserdetektor kunne detektere laserlys fra følgende vinkelområder for å kunne posisjonere roboten på hele spilleflaten.

$$\varphi_a \geq 80,56 \tag{3.30}$$

$$\varphi_b \geq 153,73 \tag{3.31}$$

Lys fra vinkler utenfor φ_a og φ_b blir sett på som støy og avskjæres ved hjelp av utfreste beskyttelseshetter, plassert over lasersensor. I tillegg sørger disse beskyttelseshettene for at lysstøy fra andre høyder ikke når lasersensoren da de sørger for at lasersensoren får tunnelsyn. Et konsept som vil fungere da alle posisjoneringstårnene befinner seg i samme høyde som det roterende tårnet.

Dispenserbeacon

Følgende krav ble stilt under utvikling av dispenserbeacon

1 Størrelse innenfor 80x80x160mm
2 Enveis radiokommunikasjon
3 Fri sikt foran sensorer
4 Debuggingsmuligheter
5 Enkel i bruk
6 Ikke stå i veien for robot
7 Støyforhindring:
  Kun tillate vertikalt lys innenfor $\theta = 5°$ å treffe sensor, se figur 3.34
  Kun tillate horisontalt lys innenfor deteksjonsområde å treffe sensor

Tabell 3.4: Funksjonelle krav til dispenserbeacon

Figur 3.34: Beskyttelseshette tredd over lasersensor

Søk etter dispenserbeacon skal kun foretas initielt og derfor trenger ikke laserdetektor på enheten stort deteksjonsområde, da den kan rettes mot robotens startområde initielt. Av produksjonsmessige årsaker, samt at det vil være vanskelig å justere dispenserbeacon til å se mot startområdet ved for smalt deteksjonsområde, er det valgt å benytte samme deteksjonsområde φ_a som for posisjoneringstårn A og C.

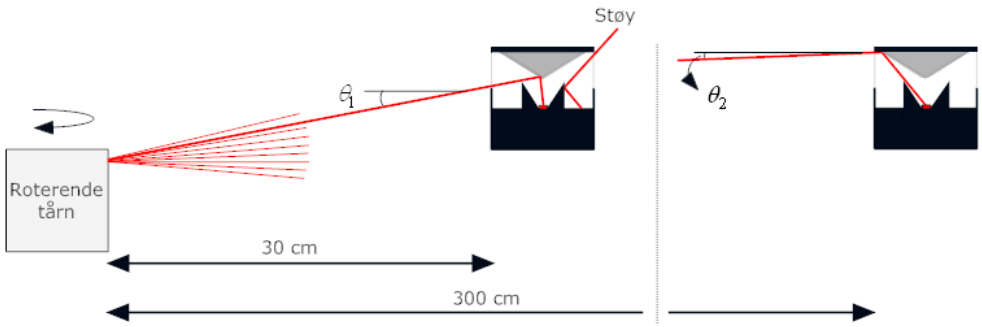
Modulen skiller seg fra et posisjoneringstårn, da den befinner seg på annen høyde enn det roterende tårnet, og må således ha fri sikt svakt oppover for å kunne detektere laserstrålene. I tillegg har enheten strengere størrelsesbegrensninger enn dem gitt av reglene, da deler av modulen laget av Eksperter i Team-gruppen vil treffe dispenserbeacon, og dispenserbeacon må derfor ha hulrom for disse.

Motstanderbeacon

Følgende krav ble stilt under utvikling av motstanderbeacon

1 Størrelse innenfor 80x80x80mm
2 Enveis radiokommunikasjon
3 360 graders deteksjonsområde
4 Debuggingsmuligheter
5 Enkel i bruk
6 Støyforhindring:
  Kun tillate vertikalt lys mellom θ_1 og θ_2 å treffe sensor, se figur 3.35

Tabell 3.5: Funksjonelle krav til motstanderbeacon



Figur 3.35: Prinsipp av utviklet omnidireksjonell laserdetektor

Motstanderbeacon skal plasseres på motstanderroboten i bevegelse og trenger derfor en omnidireksjonell laserdetektor i horisontalplanet. I tillegg er plasseringsnivået til motstanderbeacon 80 mm høyre enn det roterende tårnet, noe som resulterer i at vinkelen ned til det roterende tårn varierer fra $\theta_1 = 10,4°$ til $\theta_2 = 1,5°$ avhengig av avstanden. Det er derfor blitt utviklet en refleksjons-

kjegle som er i stand til å reflektere laserlinjen mellom minimum avstand på 30 cm og ved maksimum avstand på 300 cm ned til lasersensor, se figur 3.35. I tillegg er det utviklet en beskyttelseskappe som er plassert over lasersensoren slik at direkte belysning av denne ikke vil være mulig, og således gjør denne enheten svært robust for ytre støypåvirkning. For mer detaljert beskrivelse og skjemategninger over endelig løsning henvises det til [10, kap 2].

3.3.10 Utforming av kommunikasjonsnode på egen robot

Kommunikasjonsnode på egen robot er utviklet for å være et interface mellom robotens hovedkommunikasjonsstruktur, CAN-bus, og det nye kommunikasjonssystemet basert på radio, brukt mellom robot, det roterende tårnet, og beacons. I tillegg har også modulen ansvar for å kunne detektere brytere plassert rundt på roboten, brukt til å posisjonere roboten inntill forskjellige poengområder, samt dispensere.

Denne seksjonen går i likhet med den foregående kun igjennom kravene stilt til modulen, hvorpå mer detaljert informasjon om skjemategninger, virkemåte og endelige løsning er å finne i [10, kap 6].

Følgende krav ble stilt under utvikling av kommunikasjonsnode på egen robot

1	Kunne implementeres i kretskortramme beskrevet i kapittel 7.2.2
2	Toveis radiokommunikasjon
3	CAN kommunikasjon
4	Tilkobling for eksterne brytere
5	Enkle retningsstyrte tilkoblinger
6	Muligheter for enkel utskiftning
7	Debuggingsmuligheter
8	Enkel i bruk

Tabell 3.6: Funksjonelle krav til kommunikasjonsnode på egen robot

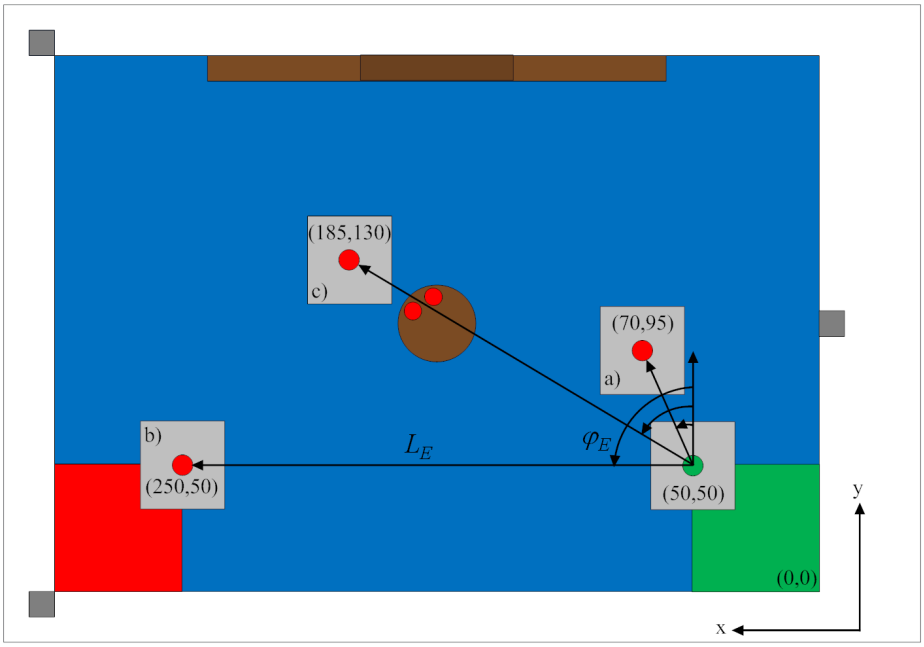
3.4 Test og resultater

Denne seksjonen tar for seg de forskjellige testplanene som er blitt utviklet for å kunne verifisere at systemet fungerer, samt bidra til å forstå styrker og svakheter til de forskjellige måleprinsippene benyttet. Dette vil igjen bidra til å kunne tolke forskjellig sensorinformasjon riktig i AI, slik at korrekt posisjonering av motstander, posisjoneringtårn, samt dispenser kan foretaes.

All testing er ellers gjort under så vanskelige lysforhold som mulig ved å benytte flere halogenspotter direkte vinklet mot de forskjellige beacons.

3.4.1 Deteksjon av motstander

Det er blitt utviklet en testplan som gjør bruk av en minimal motstander påsatt motstanderbeacon. Med minimal menes det at motstanderbeacon kun er plassert på et svært tynt tårn slik at det er mulig å få verifisert at avstandsensorene er i stand til å detektere roboter av små størrelser. Testplanen innebærer deteksjon og plassering av motstanderen ved kort distanse, lang distanse, og i tilfellet der byggeplassen plassert midt på konkurransebordet befinner seg imellom robotene. I figur 3.36 er disse plasseringene skilt med henholdsvis oppsett a), b) og c).



Figur 3.36: Plassering av egen og motstanders robot under test

Ønsket oppførsel

Ved å se på systemet analytisk vil ideell oppførsel resultere i avstander og vinkler gjengitt i tabell 3.7. Teoretisk avstand er dog fra senterpunkt til senterpunkt, slik at estimert avstand må forventes noe mindre.

Oppsett	$L_E(cm)$	$\varphi_E(grader)$
a	48,24	23,96
b	200,00	90
c	156,92	59,35

Tabell 3.7: Analytisk φ_E og L_E til motstanderrobot ved oppsett a), b), og c)

Resultater

Test av de forskjellige oppsettene a), b), og c) er gjengitt i tabell 3.8, 3.9, og 3.10.

Oppsett a				
Forsøk	$L_E(cm)$			$\varphi_E(grader)$
	Ir	*Ultralyd*	*Laser*	
1	40	39	47	24,00
2	56	39	48	24,00
3	41	39	48	24,25
4	39	39	48	24,00
5	39	39	48	24,00

Tabell 3.8: Målt avstand og detektert vinkel ved testoppsett a)

Oppsett b				
Forsøk	$L_E(cm)$			$\varphi_E(grader)$
	Ir	*Ultralyd*	*Laser*	
1	233	139	187	89,75
2	112	139	190	89,75
3	244	142	191	89,75
4	95	139	191	90,00
5	100	140	191	89,75

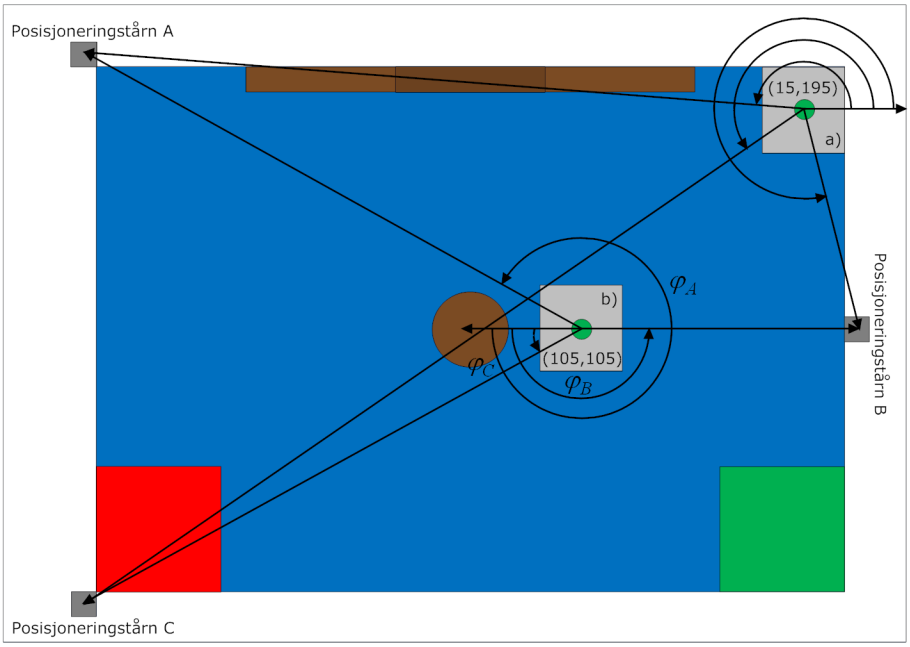
Tabell 3.9: Målt avstand og detektert vinkel ved testoppsett b)

Oppsett c				
Forsøk	$L_E(cm)$			$\varphi_E(grader)$
	Ir	*Ultralyd*	*Laser*	
1	128	103	143	59,50
2	273	103	160	59,50
3	151	103	147	59,25
4	151	102	146	59,25
5	158	103	144	59,25

Tabell 3.10: Målt avstand og detektert vinkel ved testoppsett c)

3.4.2 Deteksjon av posisjoneringstårn

For å kunne benytte triangulering som absolutt posisjoneringssystem er gode vinkelavlesninger til tre tårn sett fra roboten nødvendig, uansett hvor på spillebrettet roboten befinner seg. I seksjon 3.3.9 ble nødvendig deteksjonsområde funnet til å være $\geq 80,56°$ for posisjoneringstårn A og C, mens $\geq 153,73°$ for posisjoneringstårn B. Det er derfor utviklet en testplan, se figur 3.37, som har til hensikt å teste systemet ved randen av detekesjonsområdet, samt midt i, ved henholdsvis oppsett a) og b). I tillegg endres rotasjonsretningen til det roterende tårnet hver gang det settes til å lete etter posisjoneringstårn, slik at systemet er testet både med rotasjon mot høyre og mot venstre.



Figur 3.37: Testoppsett for deteksjon av trianguleringsvinkler

Ønsket oppførsel

Ved å se på systemet analytisk vil ideell oppførsel resultere i vinkler gjengitt i tabell 3.11.

Oppsett	φ_A	φ_B	φ_C
a	176,14	283,13	214,57
b	331,23	180	28,76

Tabell 3.11: Analytisk φ_A, φ_B og φ_C ved oppsett a) og b)

Resultater

Test av de forskjellige oppsettene a) og b) med forskjellige rotasjonsretninger på det roterende tårnet er gjengitt i tabell 3.12, 3.13, 3.14 og 3.15.

Oppsett a) rotasjon mot høyre			
Forsøk	φ_A	φ_B	φ_C
1	176,00	284,50	214,50
2	176,00	284,25	214,50
3	176,00	284,25	214,25
4	176,00	284,25	214,50
5	176,25	284,25	214,25

Tabell 3.12: Vinkler til posisjoneringstårn

Oppsett a) rotasjon mot venstre			
Forsøk	φ_A	φ_B	φ_C
1	175,75	283,50	214,50
2	176,00	283,75	214,25
3	176,00	283,50	214,50
4	176,00	283,75	214,50
5	176,25	283,50	214,25

Tabell 3.13: Vinkler til posisjoneringstårn

Oppsett b) rotasjon mot høyre			
Forsøk	φ_A	φ_B	φ_C
1	331,25	180,25	28,50
2	331,25	180,00	28,50
3	331,25	180,25	28,50
4	331,25	180,00	28,75
5	331,25	180,00	28,50

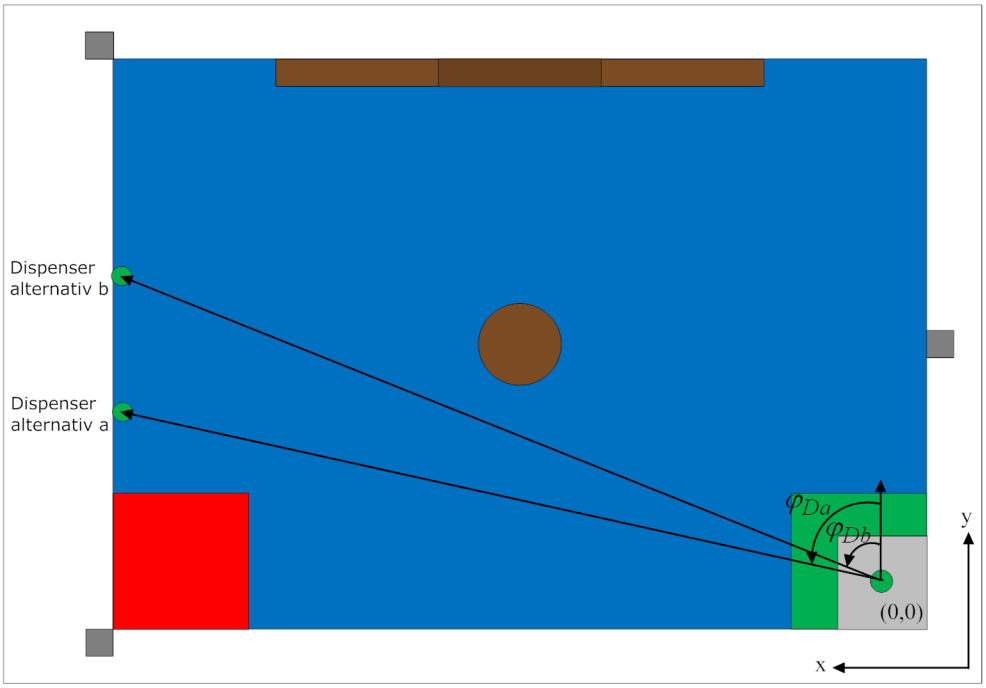
Tabell 3.14: Vinkler til posisjoneringstårn

Oppsett b) rotasjon mot venstre			
Forsøk	φ_A	φ_B	φ_C
1	331,00	180,75	28,50
2	331,25	180,50	28,50
3	331,00	180,75	28,50
4	331,25	180,00	28,50
5	331,00	180,00	28,50

Tabell 3.15: Vinkler til posisjoneringstårn

3.4.3 Deteksjon av dispenserbeacon

Søk etter dispenseren skal kun foretaes initielt og således vil det ikke være nødvendig å finne vinkel til dispenserrn fra områder utenfor startområdet. Testplanen er derfor gitt av figur 3.38.



Figur 3.38: Testoppsett for deteksjon av flyttbar dispenser

Ønsket oppførsel

Ved å se på systemet analytisk vil ideell oppførsel resultere i vinkler gjengitt i tabell 3.16.

Alternativ	φ_D
a	77,06
b	67,88

Tabell 3.16: Analytisk φ_D til dispenser på posisjon a og b

Resultater

Test av de forskjellige plasseringsalternativene for dispenser gitt av testplanen er gjengitt i tabell 3.17.

	Alternativ a	Alternativ b
Forsøk	φ_{D_A}	φ_{D_B}
1	76,75	67,75
2	76,75	67,50
3	76,75	67,50
4	76,75	67,75
5	76,50	67,50

Tabell 3.17: Detekterte vinkler til dispenser plasseringsalternativ a og b

3.5 Diskusjon

Denne seksjonen drøfter systemet valgt for deteksjon av motstanderbeacon, posisjoneringstårn og dispenserbeacon, og baserer seg på resultatene funnet i seksjon 3.4, samt erfaringer gjort under utvikling og ved bruk.

3.5.1 Avstandsestimering

Å basere seg på flere redundante system for avstandsestimering har vist seg å være svært gunstig da de forskjellige teknologier har ulike styrker hva angår støytoleranse, og hvilke avstander de egner seg til å måle. Som det vil bli beskrevet har alle teknologier mulighet til å feile, eller finne avstand til feil objekt, slik at det har vært svært viktig å ha en god algoritme for å filtrere ut feilmålinger. Denne algoritmen er beskrevet i seksjon 3.3.6, og har således vært en forutsettning for å kunne ta i bruk det benyttede systemet.

Ultralydmodul

Ultralydmodulen har vist seg å være den mest stabile avstandsestimatoren benyttet, og den fungerer svært godt ved korte avstander $\leq 1,5m$. Grunnet det dråpeformede deteksjonsområdet vil modulen lett kunne detektere andre elementer som spillebrikker, byggeplasser, eller gulvet. Dette resulterer i at ved lengre distanser vil målt avstand estimeres noe kortere enn virkelig avstand, noe som spesielt testtabell 3.10 viser. Sett fra et antikollisjonsperspektiv er det allikevel mye viktigere å kunne ha et stabilt og pålitelig system ved korte avstander, og heller tolerere noe distanseavvik ved større avstander. Om avviket i tillegg er stort, vil målingen bli filtrert bort og således ikke ha noen betydning for systemet i det hele tatt.

Infrarød modul

Den infrarøde sensoren er i utgangspunktet av svært robust type, men har vist seg å være god ved korte distanser $\leq 1m$, men generelt ha stor varians over hele avstandspekteret. Sensoren har et rekkeviddeintervall fra 0,2m til 1,5m, slik at distanser utenfor dette resulterer i upålitelige målinger, noe som kommer frem i testtabell 3.9. Den store variansen, som øker ved større distanser, skyldes hovedsakelig den ulineære spenningen på signalet modulen leverer, se figur A.1. Ved større distanser vil en liten endring av spenningsnivå medføre stor distanseendring, og således kan en bedre tilnærming enn nåværende stykkevise lineariseringsmetode forbedre variansen ved store avstander.

I tillegg vil feilmålinger som følge av lysstøy kunne reduseres ved å beskytte sensoren mer enn hva som er gjort i nåværende konsept. Dette kan gjøres ved å flytte sensoren lenger inn i det roterende tårnet og lage en beskyttelseshette slik som gjort på posisjoneringstårnene og dispenserbeacon.

Selv med varians, da spesielt ved avstander $\geq 1m$, samt feilmålinger grunnet støypåvirkning, blir modulen allikevel sett på som en god ressurs ved avstandsestimering. Dette grunnet at eventuelle feilmålinger blir filtrert bort ved hjelp av algoritmen beskrevet i seksjon 3.3.6.

Avstand basert på lasermodul

Avstand generert ved å benytte tiden mellom lasertreffene har vist seg å bli meget god, gitt at motstander befinner seg $\geq 1,5m$ unna det roterende tårn, eller kjører med begrenset hastighet. I testene foretatt er det av praktiske årsaker benyttet en motstanderrobot som står stille, noe som resulterer i godt avstandsestimat også ved korte distanser. Dette vil dog ikke være tilfelle under reel bruk, da det ved korte distanser vil oppstå stor varians, avhengig av hvilken retning motstander og egen robot har i forhold til hverandre. Ved distanser $\geq 1,5m$ vil hastigheten på laserstrålen være såpass høy at bevegelsesendringer mellom egen og motstanders robot kun vil medføre marginale distanseavvik. Om rotasjonshastigheten på det roterende tårn økes ytterligere, vil meget god

distanseestimering til en bevegende motstanderrobot kunne utføres ved mindre distanser.

Ekstra avviksfaktorer er at det roterende tårnet ikke har hastighetsregulator, slik at varierende rotasjonshastighet kan føre til et lite avvik i avstandsestimeringen. I tillegg har det vist seg svært vanskelig å plassere tre laserlinjegeneratorer helt parallellt, slik at avstanden mellom laserstrålene varierer fra 12 til 14 mm ved en distanse fra 0 til 3 m, noe som igjen resulterer i feil tid estimert mellom lasertreffene. Allikevel er modulen laget med stor nøyaktighet, samt at rotasjonshastigheten varierer svært lite, og avvikene som skapes på bakgrunn av dette kan dermed bli sett på som marginale.

3.5.2 Vinkeldeteksjon

For å kunne unngå å kollidere med motstanderen er det nødvendig å kjenne vinkelen samt avstanden til motstander, hvorav avstanden kan ha varians, mens vinkelen er svært viktig å kunne estimere nøyaktig. I tillegg til å kunne forsikre om korrekt vinkel til motstander vil det i et system med posisjoneringstårn også være viktig å kunne detektere disse med tilstrekkelig oppløsning for god triangulering.

Tre parallelle lasere som triggerkilde, svært støybeskyttet deteksjonskilde med tilhørende optimalisert elektronikk, samt radiokommunikasjon på de respektive beacons, har resultert i et tilnærmet feilfritt system og med en oppløsning på $0,25°$. Testresultatene viser at vinkelavvikene er svært små i forhold til teoretisk verdier, og dette kan igjen forklares med at de respektive beacons ikke har blitt plassert helt korrekt, noe som er tildels vanskelig. Plassering av de forskjellige beacons, samt startposisjonen på det roterende tårnet, er nødt til å være helt riktig for at korrekt deteksjonsvinkel i forhold til egen robot skal kunne estimeres nøyaktig, og blir sett på som systemets akilleshæl.

Krav til nøyaktighet under oppbygningen av spillebordet, med tilhørende plattformer for posisjoneringstårn, er derfor store for at et system basert på posisjoneringstårn kan benyttes, uansett systemets hastighet og oppløsning.

3.6 Konklusjon

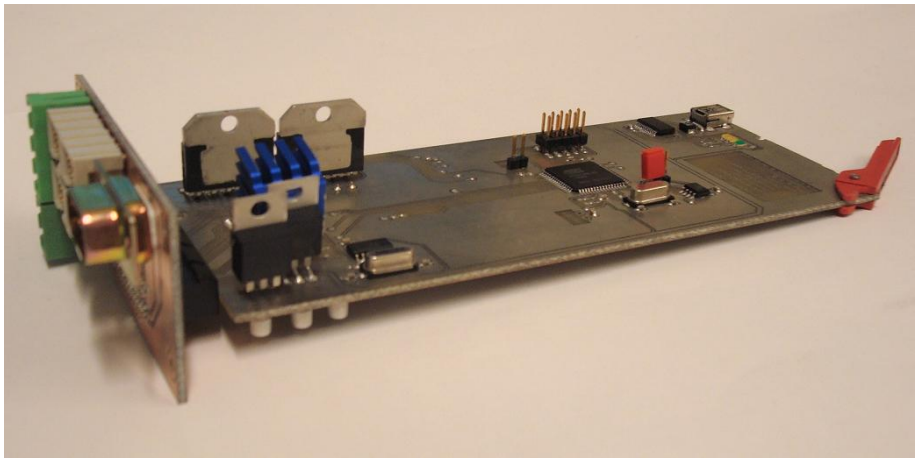
Det har blitt utviklet et velfungerende beacon-basert system, som er i stand til å detektere vinkel og avstand til motstanderbeacon uansett plassering i forhold til egen robot, samt vinkler til posisjoneringstårn og dispenserbeacon. De detekterte vinklene har oppløsning på minimale $0,25°$, og eventuelle vinkelavvik som oppstår skyldes hovedsakelig at det er svært vanskelig å plassere beacons korrekt, da avstandene er store og krav til nøyaktighet likeså. Systemet er raskt og dynamisk slik at det er i stand til å detektere en motstanderrobot som flytter seg raskt, og det fungerer under særdeles vanskelige lysforhold. Systemet utviklet gjør egen robot i stand til å benytte triangulering for absolutt posisjonering, til å detektere flyttbar dispenser, samt til å kunne bruke tidligere

utviklet antikollisjonssystem. Systemet er laget modulært slik at det kan benyttes i flere forskjellige sammenhenger i fremtidige konkurranser, og systemtes egenskaper samt erfaringer er godt dokumentert i [10]. Avsluttningsvis må det derfor konkluderes med at systemet fungerer meget godt, og at det forventes at det implementeres og videreutvikles i fremtidige systemer.

Kapittel 4

Fremdriftssystem

For at roboten skal være i stand til å utføre oppgaver forskjellige steder på konkurransebordet er det viktig med et fremdriftssystem som sørger for rask posisjonsforflytning. Dette kapittelet tar for seg hvordan tidligere eksisterende fremdriftssystem har blitt funksjonelt og implementasjonsmessig forbedret, slik at roboten beveger seg med betraktelig større nøyaktighet, og med et mindre og mer oversiktlig modulært drivsystem. For mer informasjon enn den gitt i dette kapittelet henvises det til [10, kap 7].



Figur 4.1: Nyutviklet motordrivermodul

4.1 Bakgrunn

4.1.1 Eksisterende system

Det eksisterende systemet består av to motorer med tilhørende motordrivermoduler som styres over kommunikasjonsgrensesnittet CAN beskrevet i [7, kap 4.3.2]. Disse motordrivermodulene fungerer ved at en referanse sendt fra AI blir

omdannet til spenning og satt på den respektive motoren. I tillegg har motordrivermodulene oppgaven med å lese av kvadraturtellerene som tilhører høyre og venstre odometrimodul. Det eksisterende system er derfor operativt, men erfaring ved bruk av modulen fra tidligere år har medført, at en del svakheter har blitt avdekket. Disse svakhetene dreier seg først og fremst om de eksterne motordrivermodulene, ikke motorene eller den mekaniske oppbygningen med motor, gir, og odometri.

- **Ingen intern tilbakekobling**

  Det eksisterer ingen tilbakekobling som kan forsikre om at tilført spenning har resultert i ønsket rotasjon, og således vil motorene ha forskjellig rotasjonshastighet selv ved samme spenning tilført, grunnet ulik intern friksjon. Motorene er vendt samme vei med tilkobling til hjul på hver sin side, noe som gjør at motorene er nødt til å rotere hver sin vei for at roboten totalt sett skal bevege seg fremover. Da motorene har noe forskjellig friksjon avhengig av rotasjonsretningen medfører dette et unødvendig avvik som ikke oppdages før odometrienheten fanger opp feilposisjoneringen. Begge motorene har implementert moduler som kan avlese rotasjonsendringen på akslingene, men disse modulene er ikke tatt i bruk.

- **Spenningstilførsel**

  Elektronikken til motordrivermodulene styres av spenning som tilføres via CAN-bussen, mens spenning til motorene tilføres separat. Dette medfører at CAN-bussen må være operativ for at motordriverene skal kunne fungere, og at en CAN-node i den andre enden forsyner motordriverelektronikken med spenning. Dette medfører at flere system er nødt til å være påskrudd for at motordrivermodulene skal kunne være operative.

- **Størrelse og tilkoblinger**

  Motordrivermodulene opptar svært stor plass, og har ingen gode festemekanismer slik at borrelås har blitt valgt som en nødløsning. I tillegg må modulene demonteres for at tilkoblinger kan foretas, og i sum bidrar dette til å gjøre enhetene lite modulære og dårlig egnet for flytting mellom roboter.

- **Synkronisering**

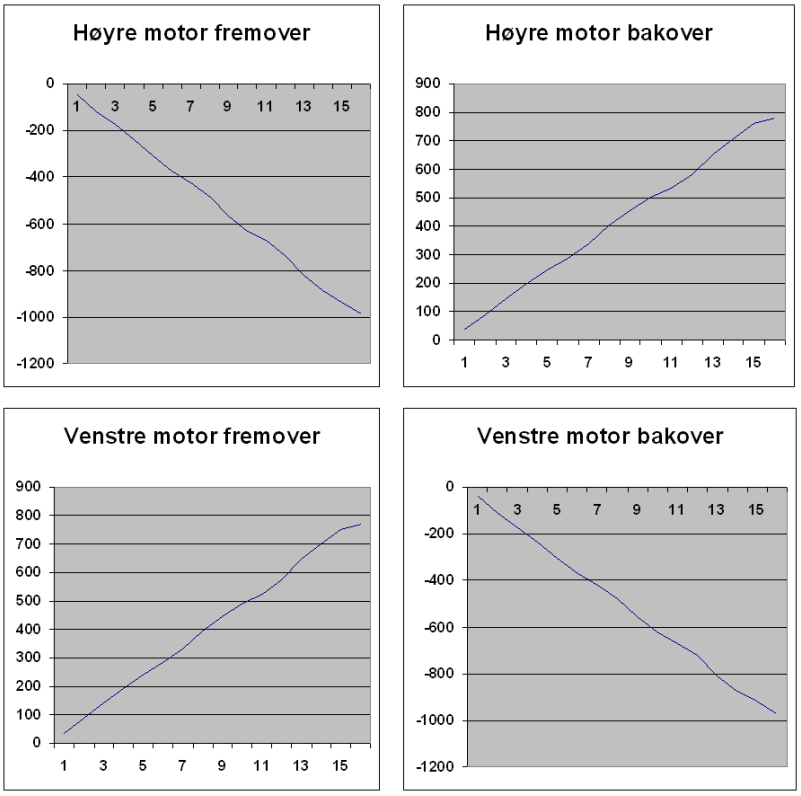
  Det er svært viktig å få informasjon fra løpehjulene tilhørende odometrien ved faste tidspunkt. Dette skjer ved at AI, som befinner seg på den bærbare pc'en, spør om å få informasjon ved gitte tidspunkt. Disse tidspunktene er ikke helt identiske, da den bærbare pc'en ikke innehar et sanntids-operativsystem, som medfører at benyttet samplingstid varierer. I tillegg må begge modulene oversende odometriinformasjon separat, som resulterer i et system som krever synkronisering.

- **Dokumentasjon**

  Eksisterende dokumentasjon av modulen er ikke sammenfallende med hva den faktisk inneholder, og er derfor en siste faktor som gjør endring av motordrivermodulen nødvendig. Ved studering av kretskort er det påsatt flere komponenter og foretatt endringer som ikke er beskrevet i tidligere rapporter eller noen andre steder.

4.1.2 Motorenes spenning/hastighets-karakteristikk

Figur 4.2 viser resultatene fra utlesning av hastigheten til motorene ved forskjellige spenninger. Her er spenning gitt langs x-aksen og hastigheten er gitt langs y-aksen. Ut fra fra disse resultatene kan det konkluderes med at motorene har forskjellig hastighet forover og bakover, og at karakteristikken forover på den ene motoren er tilnærmet identisk med karakteristikken bakover på den andre. Denne forskjellen i karakteristikk forover og bakover, gjør at roboten ikke vil gå rett frem ved lik spenning tilført begge motorene, og dette er den største grunnen til at det trengs en egen regulator for spenning tilført motorene.



Figur 4.2: Spenning/hastighets-karakteristikk for de to motorene både forover og bakover

4.2 Design

4.2.1 Krav til ny motordrivermodul

Basert på svakhetene til det eksisterende systemet, beskrevet i seksjon 4.1, ble det utviklet en rekke krav til ny motordrivermodul, se tabell 4.1, som vil forbedre systemet betraktelig, både funksjonelt og implementasjonsmessig.

1	Èn modul som styrer begge motorene
2	Èn spenningstilkobling
3	Enkle retningsstyrte tilkoblinger
4	Muligheter for enkel utskiftning
5	Minimal størrelse
6	CAN bus
7	Debuggingsmuligheter
8	Forskjellige spenningsramper på motorene
9	Intern hastighetsregulator
	Avlesning av høyre og venstre motoraksling
10	Odometri
	Avlesning av høyre og venstre odometrimodul

Tabell 4.1: Funksjonelle krav til motordrivermodul

4.2.2 Utforming av Hardware

På bakgrunn av designkravene er det blitt utviklet en ny motordrivermodul som håndterer kontroll av begge motorene. I tillegg er vinkelinformasjon fra motorakslingene tatt i bruk og benyttes for intern hastighetsregulator beskrevet i seksjon 4.2.3.

Modulen er laget for å bli plassert i kretskortramme beskrevet i kapittel 7.2.2, og består derfor av bakplan og hovedkretskort. For enkel utskiftning vil all tilkobling til modulen bli foretatt på bakplanet, og hovedkretskortet kan således skyves ut og inn i bakplanet etter eget ønske. Tilkoblingene på bakplanet er alle retningsstyrte slik at sannsynligheten for feilkoblinger reduseres betraktelig.

Spenningstilførselen er ikke lenger delt, slik at hovedspenningen tilført bakplanet blir fordelt mellom motorer og modulens elektronikk. For å forhindre eventuelle variasjoner i spenningen som følge av endring i motoreffekt, er det montert flere store avkoblingskondensatorer rundt motordriverkretsene samt spenningsregulatorene.

For mer inngående informasjon om virkemåte, konstruksjon, samt erfaringer gjort under bruk henvises det til [10, kap 7].

4.2.3 Utforming av Software

Programvaren på motordriverkortet består av to hoveddeler som svarer til oppgavene kortet skal utføre: avlesning av enkodermodulene tilknyttet robotens løpehjul og regulering av motorhastighet. Begge disse oppgavene krever en fast samplingstid, og dette løses ved hjelp av interne timere med interruptrutiner på mikrokontrolleren.

Posisjonsavlesning

Ved å sette opp en av de interne timerene til å generere et interrupt med frekvens på 100 Hz blir avlesningene fra odometrimodulen meget stabile. Dette er viktig for at navigasjonssystemet skal få gode og stabile målinger, og fordi hver posisjonsmåling, som kan sees på som en tilnærmet derivert av posisjonen, benyttes som robotens hastighet. Slik systemet var bygget før, ble denne hastigheten svært varierende på grunn av varierende samplingstid.

Motorregulator

Det overordnede navigasjonssystemet sender en referanseverdi til motordrivermodulen. Tidligere har denne referansen blitt direkte omformet til spenning som er satt ut til motoren. Det nye systemet har en egen regulatorsløyfe som justerer spenningen ut til motoren avhengig av avviket mellom den avleste hastigheten på motoren og referanseverdien mottatt fra navigasjonssystemet. Dette minimerer feilen som oppstår av at motorene kjører med forskjellig hastighet ved lik spenning, siden spenningen vil legge seg på det nivået som er nødvendig for å holde en gitt hastighet, uavhengig av hvilken motor som kjøres.

I tillegg til å få hver motor til å kjøre like fort ved lik referanse har motorregulatorene ratebegrensing. Det ble oppdaget at løpehjulene lett kunne gli mot underlaget dersom roboten hadde raske endringer i hastighet. Det er derfor lagt inn en ratebegrenser i regulatorene slik at motorene ikke vil gå fra stillestående til makshastighet på et tidsskritt. Ved innføringen av denne ratebegrenseren kom det tilbakevendende problemet med forskjellig fart på motorene. På grunn av at dette problemet viste seg å bli betydelig, ble det valgt å lage en individuell ratebegrenser med forskjellig stigning for hver av motorene. Ut fra resultatene gitt i figur 4.2 ble maksimal stigningsverdi bestemt for hver av motorene.

4.3 Konklusjon

Det er utviklet en motordrivermodul med intern tilbakekobling fra begge motorer som sørger for helt lik rotasjonshastighet på begge drivhjul, uavhengig av intern eller ekstern friksjon som måtte påføres. Interruptrutiner sørger for oversending av odometri-informasjon fra begge løpehjulene med fast samplingstid, noe som sørger for bedre posisjoneringsestimering, samt at synkronisering ikke lenger er nødvendig i AI. I tillegg er posisjoneringsestimeringen forbedret ved

at det har blitt utviklet individuelle ratebregrensninger på motorpådragene, som sørger for at løpehjulene ikke lenger glir ved oppstart. Modulen som før bestod av to store motordrivere med utilgjengelige tilkoblinger og dårlige festemekanismer, er blitt erstattet med et lett utskiftbart kretskort med tilhørende passive bakplan plassert i kretskortramme.

I tillegg er det foretatt et grundig arbeid i å dokumentere modulens software- og hardware-egenskaper, slik at det skal være lett å sette seg inn i systemet.

Kapittel 5

Kunstig intelligens

Dette kapittelet tar for seg hvordan den kunstige intelligensen til roboten er utviklet. Basert på et rammeverk som er brukt de siste to årene er det laget et nytt og utvidet planleggingssystem. Det legges vekt på å vise hvordan denne nye implementasjonen er laget ikke bare for at roboten skal ha riktig funksjonalitet, men også for at neste års eurobot-studenter lett skal kunne forstå og sette seg inn i koden for videre bruk.

Flere av områdene som blir gjennomgått i dette kapittelet er beskrevet i programvaredokumentasjonen [25]. Det anbefales derfor å lese denne dokumentasjonen på forhånd da dette gir god forståelse av systemet og forklaringer på en del begreper brukt i dette kapittelet.

5.1 Teori

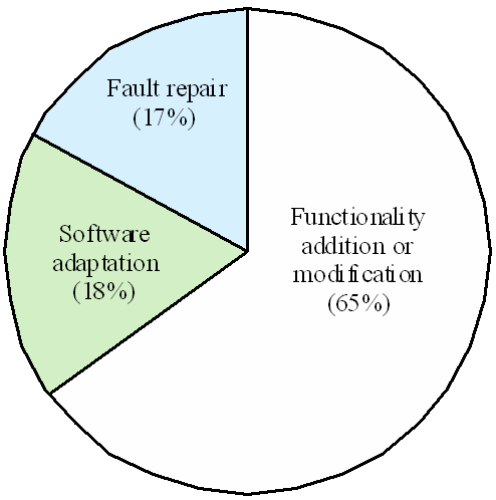
5.1.1 Vedlikehold av programvare

Ethvert system som skal programmeres og brukes av flere personer over en lengre periode har behov for vedlikehold. Det kan være flere grunner til behov for endringer i programmet. Ian Sommerville [21] beskriver disse tre faktorene som viktige i vedlikeholdsarbeid:

- Implementasjon av ny funksjonalitet
- Feilretting
- Tilpassing til nye omgivelser

I et større system vil fordelingen av disse typene vedlikehold typisk være som i figur 5.1. Her kan man se at den desidert største delen av vedlikehold er å legge til ny funksjonalitet til systemet.

Det er flere måter å løse problemene på når det oppstår behov for endringer i programvaren. Vedlikehold, arkitekturtransformasjon og re-engineering er tre ulike fremgangsmåter som brukes ved forskjellig grad av endringsbehov.



Figur 5.1: Prosentvis andel av forskjellige typer vedlikehold av et system

Vedlikehold

Med vedlikehold menes endringer av programmet mens den grunnleggende strukturen ligger fast. Dette er både den enkleste og mest brukte formen for vedlikehold. Etter hvert vil mange slike vedlikeholdsoperasjoner gjøre systemet mer komplekst og vanskeligere å vedlikeholde.

Arkitekturtransformasjon

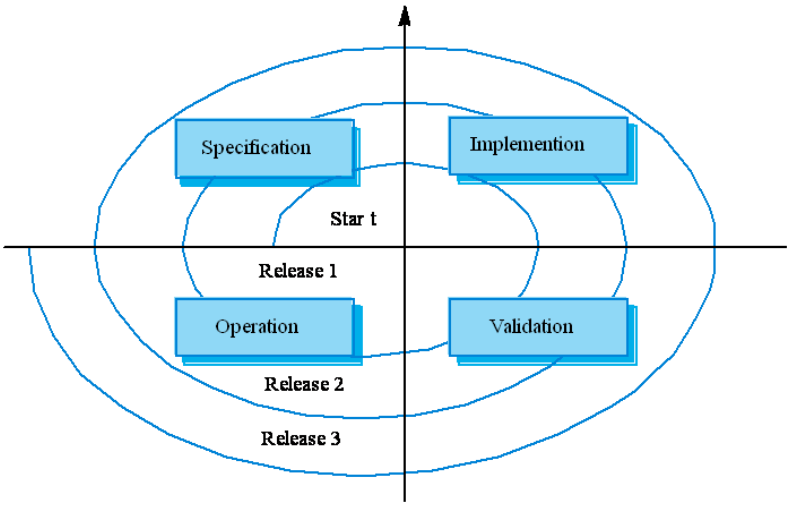
Dersom et system har blitt utsatt for så mye vedlikehold at den opprinnelige strukturen begynner å bli ugjenkjennelig kan det være nødvendig å gå dypere til verks. Arkitekturtransformasjon er endringer som medfører betydelige forandringer både i programmet og arkitekturen. Det krever mye ressurser i forhold til å utføre vanlig vedlikehold og kan medføre mer ekstraarbeid ved at det blir innført nye feil under transformasjonen.

Re-engineering

Dersom systemet er i så dårlig forfatning at det vil kreve like mye ressurser å vedlikeholde det som å begynne på nytt, er re-engineering et alternativ. Dette er den mest arbeidskrevende strategien og består av å lage systemet på nytt slik at det blir lettere å forstå og vedlikeholde. I stedet for å ta utgangspunt i en kravspesifikasjon vil man ved re-engineering ta utgangspunkt i den eksisterende koden, tolke denne, og komme frem med et nytt og bedre design som løser samme problemstilling.

5.1.2 Programvare-evolusjon

Lehman og Belady studerte utviklingsdynamikken til programvare tidlig på 70- og 80-tallet. Ut fra mange empiriske studier foreslo de i 1985[1] at det fantes et sett med lover(Lehmans lover) som gjaldt for alle system som blir vedlikeholdt. Uttrykket "systemevolusjon" ble brukt om systemers endring over tid og kan beskrives av figur 5.2. I denne figuren vises det at et system som stadig blir endret kan beskrives av en spiralmodell, der syklusen går mellom de fire fasene: spesifikasjoner, implementering, validering og aktiv operasjon.



Figur 5.2: Spiralmodell for evolusjonen til et system

Kontinuerlig forandring

Lehmans første lov konstaterer at vedlikehold av system er uunngåelig. Alle programmer som brukes i et miljø, må tilpasse seg de endringene som skjer i omgivelsene. Når nye krav fra omgivelsene er tatt hånd om og implementert i programmet, medfører dette nye endringer i omgivelsene og en ny sykel påbegynnes.

Økende kompleksitet

Den andre av Lehmans lover konstaterer at vedlikehold av et system ødelegger den opprinnelige strukturen litt etter litt og gjør programmet mer komplekst. Den eneste måten å hindre dette i å skje er å drive preventivt vedlikehold der man bruker tid på å forbedre systemets struktur uten å legge til ny funksjonalitet eller fikse programfeil.

Store programmers utvikling

Den tredje loven påstår at programevolusjonen er en selvregulerende prosess. Når et system når en viss minimumsstørrelse blir det vanskelig å endre. Fordi det er stort og komplekst er systemet vanskelig å forstå, og programmererene gjør oftere feil og innfører svakheter og feil i systemet. Dette gjør at store endringer sannsynligvis vil innføre flere feil som begrenser nytteverdien av systemendringen. Små endringer derimot unngår å redusere systemets stabilitet. Ethvert system kan dermed sies å ha en egen maksimal størrelse, og jo nærmere denne grensen man kommer, jo mindre må endringene for hver gang være, helt til programmet må restruktureres for å kunne ta ny funksjonalitet.

Sammen med disse tre lovene ble det lagt frem fem lover til som ikke nevnes i denne rapporten.

5.1.3 Objektorientert programmering

Objektorientert programmering (OOP) er et paradigme for programmering som har sitt utspring fra programmeringsspråket *Simula*. Dette språket ble utviklet av nordmennene Ole-Johan Dahl og Kristen Nygaard i 1967[23]. Etter dette fikk konseptet stor utbredelse gjennom andre mer moderne programmeringsspråk, som Pascal, C++ og Java. OOP har flere sentrale prinsipper som kan bidra til mer vedlikeholdbar kode, og i de neste avsnittene kommer en gjennomgang av de viktigste prinsippene.

Objekter

Data og funksjoner kan samles i forskjellige enheter i programmet, som kan kommunisere med hverandre gjennom tilgjengelige funksjoner som finnes i hvert av objektene. Dette er basisen for modularitet, som gjør at det er mulig å jobbe med deler av systemet uten å måtte vite hvordan resten av systemet fungerer. Hvert objekt har en metode kalt konstruktør som brukes for å opprette objektet.

Abstraksjon

Abstraksjon er å forenkle virkeligheten ved å modellere klasser som er tilpasset for problemet som skal løses. I stedet for å ha et stort objekt som skal gjøre all jobben, kan det modelleres ned til mindre deler som sammen kan utføre samme oppgave. Dette gjør at hovedobjektet som samler sammen alle disse objektene, kan jobbes med på et høyere nivå uten å måtte bry seg om virkemåten til de enkelte delene.

Innkapsling

Ved å gjøre interne variabler i et objekt utilgjengelig for andre objekter oppnår man at den interne virkemåten er beskyttet fra eksterne endringer og kan

ses på som et eget system. Endring av interne variabler tillates kun gjennom funksjoner som kan begrense hvordan variablene skal endres.

Polymorfi

Polymorfi er objekters egenskap til å kunne reagere forskjellig på samme melding avhengig av hvilken type objektet er. Så lenge to objekt oppfyller samme grensesnitt, kan de interne funksjonene som svarer til grensesnittet være vidt forskjellige.

Arv

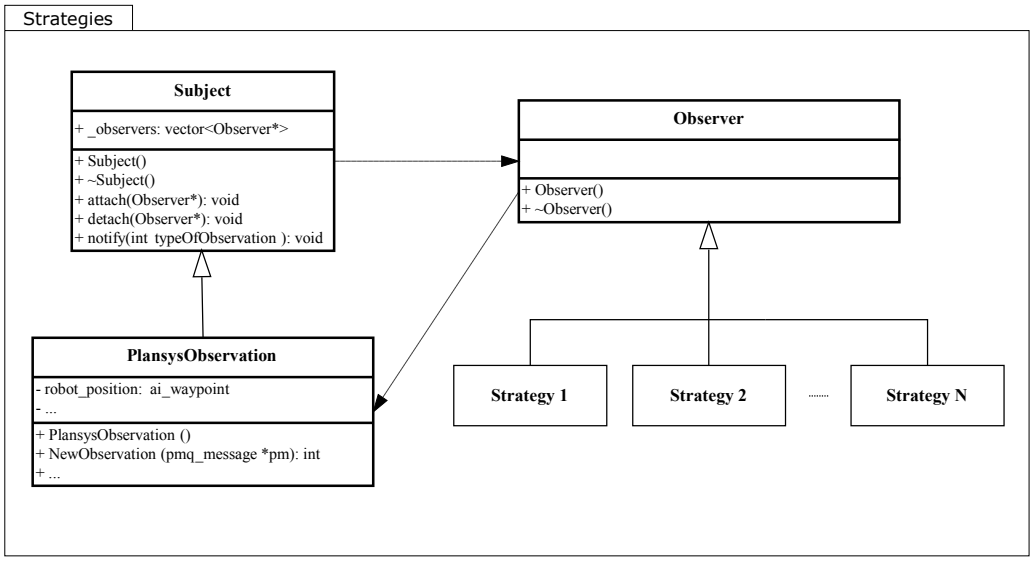
Mange objekter har noen felles egenskaper og felles metoder som kan utføres. Ved å definere et generelt objekt som inneholder alle disse egenskapene og funksjonene, og gjøre dette objektet felles for de opprinnelige objektene, unngår man å måtte gjøre mye dobbeltarbeid. Dette prinsippet kalles arving, og gjør at hvert av de arvende objektene blir en spesialisert versjon av foreldreobjektet. Den felles klassen kalles superklasse og de arvende klassene kalles subklasser.

5.2 Bakgrunn

Det er blitt arvet et rammeverk for kunstig intelligens som har blitt utviklet de to siste årene innenfor Eurobot-prosjektet. Dette rammeverket er beskrevet i [7] og [14]. I grove trekk baserer denne kunstige intelligensen seg på å dynamisk velge en av flere mulige strategier basert på hver enkelt strategis prioritet. Prioriteten til de enkelte strategiene er beregnet ut fra hvilken tilstand spillebrettet og roboten er i. For å oppnå den dynamiske funksjonaliteten blir prioritetene kontinuerlig oppdatert basert på input fra ulike sensorer.

I arbeidet med prosjektoppgaven i høst [9] ble det funnet at strukturen og tankegangen bak dette rammeverket er meget godt egnet for videre bruk i konkurransen, men at implementasjonen desverre ikke holder mål for å jobbe videre med. Det største problemet med koden er at den ser ut til å være objektorientert, ikke drar nytte av noen av fordelene objektorientert struktur medfører. Figur 5.3 viser hvordan strukturen til det arvede planleggingssystemet er lagt opp. Her ser man at alle strategiene har en felles superklasse, kalt Observer. Kode som skrives i denne klassen blir arvet av alle strategiene og man oppnår dermed at felles kode for strategiene kun trengs å bli skrevet en gang. Dette er også en stor fordel dersom felles kode må endres. Listing 5.1 viser hele implementasjonen av klassen Observer, mens listing 5.2 viser konstruktøren til en tilfeldig strategi. Her kan man se at superklassen ikke inneholder kode i det hele tatt, mens strategien inneholder mange variabler som er felles for alle strategiene. Dette er stikk motsatt av hvordan god og vedlikeholdbar kode bør skrives. Alle de implementerte strategiene har ca 500 kodelinjer, og er grunnlaget for at det bare er gitt et utdrag fra denne koden.

Andre svakheter med denne implementasjonen er at hver enkelt strategi-implementasjon er meget vanskelig å få oversikt over. Dette kommer delvis av at strategiene består av mye kode, men også fordi hver strategi benytter seg av lavnivå kommandoer til navigasjonssystemet og til de distribuerte mikro-kontrollerene rundt om på roboten.



Figur 5.3: Klassediagram for det arvede strategisystemet

```
#include "Observer.h"

Observer::Observer( ){
        teamColor = -1;
}
Observer::~Observer(){

}
```

Listing 5.1: Observer.cpp

```
#include "WhiteBlueDispenserStrategy.h"

WhiteBlueDispenserStrategy::WhiteBlueDispenserStrategy(
        PlansysObservation *s , int theTeamColor ){
        teamColor = theTeamColor;
        initWaypoints();
        blueWhiteDispenser = GetWaypoint(BLUE_WHITE_
        DISPENSER_WAYPOINT);
        ai_waypoint_3d = GetWaypoint(ZERO_WAYPOINT);
        state = FIRST_RUN;
        strategyDone = FALSE;
        strategyCompleted = FALSE;
        priority = 5;
        internalState = 5;
        extraBall = 0;
        timeLeft = 90;
        numberOfRuns = 0;
        numberOfTries = 0;
        Init();
        _subject = s;
        _subject->Attach(this);
        _subject->SetFirstRunVariable(TRUE);
        cout << "Strategy added" << endl;
}
```

Listing 5.2: WhiteBlueDispenserStrategy.cpp

5.3 Krav for implementasjon av strategisystemet

Kravene for strategisystemet er i utgangspunktet de samme som nevnt i [7]. Disse kravene er listet opp under.

1. Ha mulighet for et stort antall forskjellige uavhengige strategier
2. Prioritere målorienterte strategier
3. Velge strategi ut fra hvilken strategi som har høyest prioritet
4. Hver strategi skal ha mulighet til å ha all tilgjengelig oppdatert informasjon til enhver tid
5. Robust ved at den kan fjerne strategier som ikke virker
6. Må kunne håndtere avbrudd uten at dette påvirker aktive strategier
7. Aktiv strategi skal beholdes til den har oppnådd sitt mål, eller en annen strategi gir betydelig høyere prioritet

De fem første kravene er allerede løst ved strukturen som finnes og vil bli benyttet videre. Det sjette kravet derimot, er basert på en virkemåte i systemet som ikke lenger finnes. Tidligere har antikollisjonssystemet fungert slik at det har avbrutt den aktive strategien og gjort nødvendige korrigeringer. Ved denne funksjonaliteten er det viktig at strategien kan fortsette der den ble avbrutt. Som man kan lese i kapittel 6 og i programvaredokumentasjonen [25] er årets antikollisjonssystem integrert i planleggingsystemet på en slik måte at strategiene ikke blir avbrutt. Dette gjør kravet om avbruddshåndtering unødvendig.

På bakgrunn av erfaringer fra konkurransen i 2008 er det også et ønske om å endre på det syvende kravet. Problemet som oppsto i denne konkurransen var at roboten kjørte seg fast på en skrue tidlig i omgangen. Dette gjorde at roboten ble stående og spinne i luften helt til det var et minimum av tid igjen. Ved dette tidspunktet slo leveringsstrategien inn fordi den fikk betydelig høyere prioritet enn den aktive strategien, og gjorde at roboten klarte å komme seg løs fra skruen. Kravet som ønskes på bakgrunn av denne observasjonen er at hver strategi må ha en grense for hva som er rimelig tid å bruke på gitte operasjoner, og ha mulighet til å avbryte strategien dersom denne tidsgrensen overskrides. Det nye kravet blir dermed:

7. Aktiv strategi skal beholdes til den har oppnådd sitt mål, eller blir avbrutt på grunn av tidsoverskridelse i en tilstand

Alle disse kravene beskriver planleggingssystemets funksjonalitet, men sier ingen ting om hvordan systemet bør implementeres. I tillegg til disse kravene er det derfor blitt definert noen flere krav som skal gjøre systemet lettere å vedlikeholde. Disse kravene gjør det både enklere å legge til nye strategier, endre de eksisterende og å forstå hvordan hver enkelt strategi fungerer.

8. Skrive all generell kode for strategiene i superklassen
9. Gjøre den spesifikke delen av strategiene så liten som mulig
10. Lage høynivå funksjoner for bruk i strategiene som er lette å bruke og forstå
11. Gi variabler og funksjoner tydelige og forståelige navn

5.4 Strategiene

Årets oppgave var lett å dele inn i forskjellige typer oppgaver siden den gikk ut på å plukke opp spilleelementer og levere disse på spesifikke poengområder. Strategiene har dermed blitt delt opp i fire forskjellige deler: BoardDispenserStrategy, BuildingStrategy, DispenserStrategy og LintelStrategy. I tillegg har det blitt laget en egen strategi som er brukt kun for testing og kalibrering.

Ingen av strategiene er detaljbeskrevet i dokumentasjonen [25]. Dette fordi disse strategiene er meget spesifikke for årets oppgave, og uansett må skrives på nytt til neste år.

5.4.1 Implementasjon

Ved implementasjon av strategisystemet ble det lagt vekt på å gjøre koden mest mulig vedlikeholdbar. Ved å bruke teknikker fra objektorientert programmering, som arv, innkapsling og abstraksjon er strukturen til strategisystemet endret betydelig. All generell kode er lagt i superklassen Observer, og hver av strategiene har kun et minimum av funksjoner: konstruktør, destruktør og 3 metoder som er de eneste metodene som skiller de ulike strategiene. Ingen av strategiene inneholder variabler utenom de som arves fra Observer, og koden ligger på mellom 100 og 200 kodelinjer per strategi. Dette er betydelig forskjell fra fjorårets kode, der hver av de 15 strategiene hadde kode på mellom 400 og 500 linjer. Samtidig er gjennomsnittlig andel kommentarer og utskrifter hevet fra ca 5% til ca 20%. Dette gjør at det er mye enklere å forstå koden og holde oversikt over den.

5.4.2 Timing

Strategienes funksjon som utfører den gitte oppgaven er implementert som en stor switch/case-blokk, med hver enkelt kommando som en egen tilstand. Hver tilstand har en egen grense for hvor lang tid som er rimelig å bruke i denne tilstanden, og denne tiden blir sjekket ved hvert funksjonskall. Dersom tiden går ut er det mulig å trå til med forskjellige tiltak avhengig av hvilken tilstand det var som timet ut. Et eksempel er dersom roboten ikke får respons fra en trykkbryter ved forsøk på å kjøre inntil poengområdet. Dette kan komme av flere grunner; roboten kan være kommet så mye ut av posisjon at den bommer

på poengområdet eller det kan ligge en sylinder mellom roboten og poengområdet. Siden roboten ikke har mulighet til å finne ut hva som har skjedd, må det iverksettes tiltak som løser problemet uavhengig av årsak. Roboten kan dermed avbryte den aktive strategien, rygge bakover et lite stykke for å komme fri fra en eventuell obstruksjon og velge en ny strategi. Robotens status er blitt oppdatert i løpet av strategiens kjøring, og en ny prioritering av strategiene vil velge den strategien som på dette tidspunktet er høyest prioritert. Denne funksjonaliteten gjør at det syvende kravet gitt i forrige seksjon blir oppfylt.

5.4.3 Prioritering

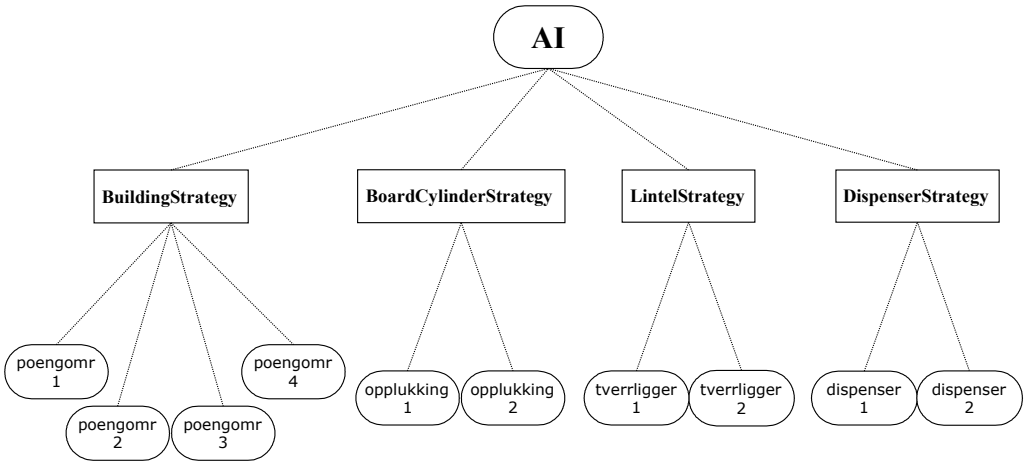
Prioriteringen av hvilken oppgave roboten skal utføre skjer i to runder, først velges en av strategiene på bakgrunn av både robotens og spillebrettets tilstand. Dette valget skjer på bakgrunn av hva roboten skal utføre, men hvor den skal gå for å utføre oppgaven vektlegges ikke. Under følger en liste over alle variablene som blir tatt hensyn til ved prioritering av de ulike strategiene. Hver strategi bruker kun de variablene som er relevant for prioriteringen av den enkelte strategien.

- Gjenstående tid
- Om strategien er kjørt før eller ikke
- Antall sylindre igjen på bordet
- Antall sylindre igjen i dispensere
- Antall sylindre i roboten
- Antall tverrliggere igjen i dispensere
- Antall tverrliggere i roboten
- Avstand/kostnad til nærmeste ledige dispenser tilknyttet den gitte strategien
- Om alle dispensere tilknyttet den gitte strategien er blokkert av motstanderen

Når strategivalget er gjort har roboten bestemt seg for hvilken handling den skal utføre, men for alle strategiene finnes det flere valg for å utføre denne handlingen: Byggestrategien har 4 forskjellige leveringsområder å velge mellom, dispenserstrategien kan hente fra to forskjellige dispensere, tverrliggerstrategien har to forskjellige hentepunkt og bordsylinder-strategien har to forskjellige opplukkingsrutiner. Strategien utfører dermed en prioriteringsrunde til for å finne ut hvilken av disse posisjonene som er best å velge. Denne andre prioriteringsrunden bruker andre variabler for å velge mellom de ulike posisjonene. Disse variablene er gitt i den påføgende listen.

- Antall objekter igjen i den gitte dispenseren/opplukkingsrutinen
- Om den gitte posisjonen er blokkert av motstanderen
- Avstand/kostnad til den gitte posisjonen
- Poengområdets høyde
- Om det er bygget en konstruksjon allerede på dette poengområdet

Figur 5.4 viser en oversikt over hvordan de to prioriteringene fungerer sammen. Denne hierarkiske prioriteringsstrukturen gjør at det er mulig å ha så få som 4 strategier selv om det er 10 ulike posisjoner som kan velges. I tillegg gjør strukturen det mulig å dynamisk endre ønsket posisjon under utføring av en strategi. Dersom motstanderen blokkerer ruten til den posisjonen som har blitt valgt, blir denne posisjonen lavere prioritert, og en annen posisjon med høyere prioritet kan velges. Det er lagt inn en rutine som hvert sekund sjekker om det er andre posisjoner innad i strategien som har høyere prioritet enn den valgte og eventuelt bytter til denne. Når roboten har nådd frem til den høyest prioriterte posisjonen avsluttes denne sjekkingen, siden roboten da er i gang med utførelsen av oppgaven som skal gjøres.



Figur 5.4: Oversikt over de to forskjellige prioriteringsrundene for valg av strategi og posisjon

5.4.4 Presisjon

Alle strategiene er basert på sekvensielle lister av waypoint for å komme seg frem til det ønskede objektet og for å rygge bak i god nok avstand for å ikke kollidere med det. Disse waypointene har forskjellige krav til nøyaktighet. Det første waypointet, som kan bli brukt for å krysse store avstander på brettet, har en stor *circle of acceptance*. Dette er greit fordi det neste waypointet som ligger nærme objektet har mye mindre *circle of acceptance* og dermed større nøyaktighet for robotens posisjon.

Alle strategiene har i tillegg lagt inn støtte for å bruke absolutt posisjonering. Ved det første av waypointene i listen blir det roterende tårnet bedt om å finne robotens absolutte posisjon. Når da roboten skal bevege seg til det neste waypointet vil den akkumulerte relative feilen bli rettet opp og roboten har bedre mulighet for å treffe nøyaktig riktig posisjon. Denne funksjonaliteten ble desverre ikke benyttet i konkurransen da den arvede trianguleringsalgoritmen ikke fungerte.

5.4.5 Strategiutførelse

Her kommer en oversikt over de forskjellige strategiene som er implementert på roboten. Hver strategi har en gitt oppgave den skal løse, og går gjennom en sekvens av operasjoner for å fullføre oppgaven sin.

BoardCylinderStrategy

Denne strategien skal plukke opp sylinderene som ligger på bordet i et av ti mulige forhåndsbestemte mønstre. Det skal være mulig å kjøre denne strategien to ganger for å plukke opp alle sylinderene på bordet. På grunn av at strategien skal ha forskjellig virkemåte ved første og andre kjøring, finnes det en variabel som holder på denne informasjonen. Første gang strategien kjøres begynner roboten med å be kameraet om å finne hvilke posisjoner sylinderene ligger i. Samtidig som kameraet finner dette mønsteret, blir det roterende tårnet bedt om å finne den flyttbare dispenseren. Når denne startprosedyren er fullført begynner roboten å plukke opp sylinderene på bordet. Siden det ligger 6 sylindere på bordet, og roboten er begrenset av reglementet til å oppbevare maks 4 sylindere om gangen, er det satt opp en optimal rute for hver enkelt strategi. Roboten tar den korteste veien via 4 sylinderposisjoner til det sentrale poengområdet, og passer på å ikke dytte noen av de andre sylinderene ut av posisjon. Når 4 sylindere er plukket opp og det er første gang denne strategien kjøres, avslutter strategien sin rutine.

Ved oppstart av strategien for andre gang sjekkes variabelen som sier om strategien er kjørt før eller ikke, for å velge riktig funksjonalitet. Nå er det bare 2 sylindere igjen, og disse plukkes også opp i bestemt rekkefølge for å bruke minst mulig tid. Når begge sylinderene er plukket opp, avslutter strategien og roboten er klar til å gå løs på neste strategi.

BuildingStrategy

Dette er strategien som skal bygge templer på et ledig poengområde. Første gang denne strategien kjøres, velger den å gå til den sektoren på det sentrale poengområdet som er nærmest robotens posisjon. Roboten stopper i en gitt avstand fra poengområdet for at kameraet kan ta bilde og analysere om det allerede finnes en bygning på de 8 sektorene som poengområdet er delt inn i. Dersom det finnes en konstruksjon innenfor de sektorene som vårt tempel kommer til å dekke, bruker roboten en algoritme for å finne den nærmeste sektoren som har nok etterfølgende ledige sektorer for å levere et tempel. Dersom det ikke finnes et eneste område på dette poengområdet som har nok sammenhengende ledige sektorer, blir dette poengområdet fjernet fra strategiens liste over mulige leveringssteder, og strategien startes på nytt. På denne måten vil roboten gjøre akkurat den samme prosedyren på det nest høyest prioriterte poengområdet.

Dersom roboten har funnet et ledig område, vil roboten kjøre til denne sektoren og gjøre seg klar til levering. Før roboten kjører inntil poengområdet, forsikrer den seg om at det ikke ligger noe i veien ved å sveipe en gang med sylinderfangeren mellom roboten og poengområdet. For å komme helt inntil poengområdet benyttes det to trykkbrytere, en på hver side av robotens front. Disse bryterene kommuniserer med den kunstige intelligensen slik at hvert hjul på roboten fortsetter å gå med konstant hastighet helt til trykkbryteren på den respektive siden er blitt trykket inn. Dette gjør at roboten justerer vinkelen sin og står perfekt inntil poengområdet uansett om det er et lite avvik i posisjoneringen.

Når denne strategien startes for andre gang, husker den hvor det forrige tempelet ble bygd, og hvilken høyde dette hadde. Roboten kjører derfor direkte dit, og gjør seg klar til å levere oppå det allerede bygde tempelet. Kameraet sjekker fortsatt om det er ledig plass, men på grunn av problemer med posisjonering av kameraet, ble denne informasjonen sjelden til å stole på. Roboten vil derfor uansett prøve å levere oppå sitt eget tårn uansett hva kameraet forteller. Dette gjør at dersom motstanderen er i stand til å bygge oppå vår konstruksjon, kan vi stå i fare for å velte det som eventuelt er bygd der. Dette ble sett på som svært lite sannsynlig på grunn av presisjonen som kreves for å bygge oppå ukjente konstruksjoner, og viste seg under konkurransen å være helt riktig avgjørelse. Det var kun ett lag som klarte å bygge oppå andre konstruksjoner, og dette var vinneren av hele konkurransen. Poengmessig ville vi tapt mot denne konkurrenten uansett.

Dersom motstanderen står i veien for å komme seg til den ønskede sektoren på poengområdet, velges det å gå til sektoren på motsatt side av poengområdet, fordi motstanderen mest sannsynlig vil bygge på poengområdet der den står, og vi vil at roboten skal gå dit det mest sannsynlig er ledig. Samtidig unngår man problemene som kan oppstå ved å navigere i nærheten av motstanderen.

Ved levering velger roboten om tverrliggeren skal leveres basert på antall

sylindere i klypene. På grunn av at tverrliggere kun gir poeng dersom de ligger vannrett vil tverrliggeren kun leveres dersom det er 2 eller 4 sylindere i klypene til roboten. Tverrliggere er heller ikke poenggivende dersom de ikke ligger på minst 2 andre spilleelementer, så tverrliggeren vil heller ikke leveres dersom det er 0 sylindere. På grunn av den mekaniske konstruksjonen til klypene er det umulig å levere tverrliggeren samtidig med sylinderene dersom det er kun 2 sylindere. Det ble derfor laget en ekstra leveringsrutine for dette tilfellet, der sylinderene først blir levert, hvorpå roboten rygger til klypene er utenfor tårnet og legger tverrliggeren pent oppå toppen av konstruksjonen. Når tempelet er levert, rygger roboten bakover til den er utenfor kollisjonsfare med det bygde tempelet før strategien avsluttes.

DispenserStrategy

Det er to dispensere for hvert lag som begge inneholder 5 sylindere plassert på spillebrettet. Plasseringen av den ene av disse dispenserene bestemmes før hver kamp til å være på en av to mulige posisjoner. Denne strategien skal plukke opp sylindre fra disse to dispenserene. Dersom posisjonen til den variable dispenseren ikke er bestemt ved oppstart av denne strategien, gjøres det et nytt forsøk på å finne dispenserens plassering ved hjelp av det roterende tårnet før det går tilbake til sin faste oppgave med å finne motstanderen.

Etter dette finner roboten ut hvilken av dispenserene som er best å gå til, avhengig av hvor mange sylindere som er igjen i dispenserene og hvor motstanderen står. Framme ved dispenseren blir opplukkeren bedt om å plukke opp 4 sylindre eller så mange sylindere som er igjen i dispenseren dersom det er færre. Når den kunstige intelligensen får bekreftelse fra opplukkingsmodulen om at det riktige antallet sylindre er plukket opp, vil den rygge bakover slik at den er utenfor kollisjonsfare med dispenseren før strategien avsluttes.

LintelStrategy

Dette er strategien som skal sørge for at roboten kan plukke opp tverrliggere fra de to henteområdene på bordkanten. Strategien begynner med å velge det henteområdet som har høyest prioritet og kjører til det første waypointet. Samtidig sendes det melding til byggemodulen at heisene skal justeres til riktig høyde for tverrliggerhenting. Dersom absolutt posisjonering er aktivert vil den ta en måling ved dette waypointet før den kjører videre til det neste waypointet som er like foran henteområdet. På denne måten blir eventuell akkumulert relativ posisjoneringsfeil rettet opp. Fremme ved det neste waypointet blir armen på midtheisen sendt ut samtidig som roboten kjører sakte rett fremover. Roboten fortsetter å kjøre frem helt til trykkbryteren på tuppen av armen blir trykket inn. Dette trykket får roboten til å stoppe og byggemodulen tar med seg tverrliggeren ved å løfte heisen. Byggemodulen gir klarsignal til den kunstige intelligensen så snart tverrliggeren er i luften, slik at roboten kan

rygge bakover mens tverrliggeren blir løftet opp i en låst posisjon. Strategien avsluttes når roboten har rygget utenfor kollisjonsfare.

CalibrationStrategy

Dette er en strategi som ble opprettet for testing av navigasjonssystemet og kalibrering av odometrien. Den fungerer som alle de andre strategiene men er ikke ment for å bli brukt under konkurranse. For sikkerhets skyld har denne strategien lavere prioritet enn alle andre strategier slik at om den ved et feilgrep kommer til å være aktiv i en konkurranse vil den bli overkjørt av alle andre strategier. For å kjøre denne strategien må altså ingen av de andre strategiene være lagt til ved kompilering.

5.5 Input fra sensorer

Den kunstige intelligensen er avhengig av input fra sensorer for å kunne tolke tilstanden til omvigelsene på spillebrettet. På roboten er det integrert en rekke slike sensorer. I dette delkapittelet kommer en oversikt over sensorene, hvilke tilstander hver av de har mulighet til å oppdage og hvordan den kunstige intelligensen reagerer på de forskjellige inputene.

5.5.1 Roterende tårn

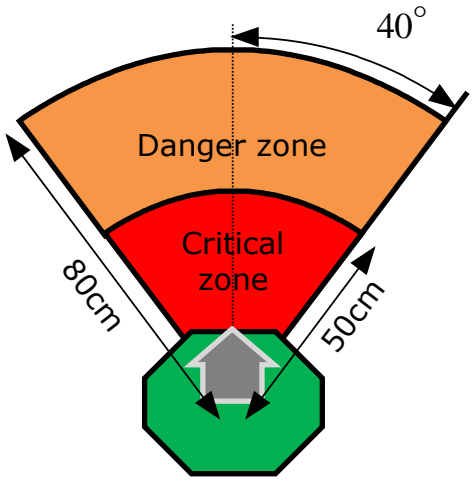
Det roterende tårnet er beskrevet i kapittel 3. Dette tårnet har mulighet for å detektere motstanderens posisjon relativt robotens egen posisjon. Den har også mulighet til å finne posisjonen til den flyttbare dispenseren, samt å finne robotens absolutte posisjon ved hjelp av navigeringsbeacons plassert rundt spillebrettet.

Motstanderrobotens posisjon

Denne informasjonen blir i hovedsak benyttet av antikollisjonssystemet for å finne en egnet rute frem til det ønskede waypointet. Planleggingssystemet bruker imidlertid denne informasjonen også til to andre formål. Ved prioritering av de forskjellige strategiene er en av variablene det tas hensyn til, nettopp motstanderens posisjon. For eksempel vil et waypoint som blokkeres av motstanderen ha betydelig lavere prioritet enn et tilgjengelig waypoint.

Den andre måten planleggingssystemet bruker denne informasjonen på er et sikkerhetssystem for å forhindre kollisjon som et tillegg til antikollisjonssystemet. Dette sikkerhetssystemet kommuniserer direkte med navigasjonssystemet og setter robotens maksimale hastighet basert på motstanderens posisjon. Figur 5.5 viser hvordan det er definert to ulike soner foran roboten der maksimal translasjons- og rotasjonshastighet blir begrenset ved deteksjon av motstanderroboten. I utgangspunktet vil antikollisjonssystemet sørge for at kollisjoner

unngås, men det er vanskelig å forutse alle forskjellige situasjoner som kan oppstå med en motstanderrobot som både har ukjent form, hastighet og udefinerbar oppførsel. Siden det også er spesifikt uttrykt i reglene at kollisjoner er strengt forbudt og kan medføre diskvalifisering, er dette sikkerhetssystemet implementert som en del av den kunstige intelligensen.



Figur 5.5: Sikkerhetssoner foran roboten som medfører redusert translasjons- og rotasjonshastighet

Dersom motstanderroboten detekteres i den orange faresektoren blir både maksimal translasjons- og rotasjonshastighet halvert. Dersom motstanderen befinner seg i den røde kritiske sonen blir maksimal translasjonshastighet satt til 0, mens rotasjonshastigheten fortsatt begrenses til halvparten av opprinnelig makshastighet. På denne måten vil det bli umulig for vår robot å kjøre inn i motstanderroboten, mens det fortsatt er mulig å komme seg ut av situasjonen ved å rotere til motstanderen kommer utenfor den kritiske sonen.

Plassering av dispenser

På spillebrettet er det plassert fire dispensere, to for hvert lag. Den ene av disse dispenserene, for hvert lag, har to mulige posisjoner den kan være plassert på. Når den kunstige intelligensen får vite av det roterende tårnet hvor dispenseren befinner seg, blir denne informasjonen brukt til to ting. Det første og mest opplagte er at dispenserstrategien blir oppdatert med riktig posisjon for den flyttbare dispenseren. I tillegg blir antikollisjonssystemet oppdatert med informasjon om både egen dispenser og motstanderens dispenser slik at roboten unngår å kollidere med disse objektene, som kan sees på som obstruksjoner på brettet.

Absolutt posisjonering

Langs sidene på spillebrettet er det plassert plattformer der hvert lag kan plassere ut navigasjonsbeacons for hjelp til navigeringen på brettet. I år er det laget beacons som kan kommunisere med det roterende tårnet og gi den informasjonen som trengs for å beregne robotens posisjon. Det er arvet en algoritme fra tidligere år som skulle beregne robotens posisjon ut fra vinklene som blir funnet. Desverre har denne algoritmen vist seg å ikke være til å stole på, og derfor har ikke absolutt posisjonering blitt brukt i konkurransen. Det som var meningen med denne funksjonen var å kunne overstyre posisjonen gitt av det relative posisjoneringssystemet i situasjoner som krever høy presisjon.

5.5.2 Kameramodul

Kameramodulen består av to identiske kameraer med et tilhørende program for å tolke data fra bilder og kommunisere med den kunstige intelligensen. Modulen har to oppgaver. Den ene oppgaven er å finne hvordan sylinderene som ligger på bordet er plassert, og den andre er å finne ut om det er ledig plass for å levere en konstruksjon på et poengområde.

Detektering av sylindre på bordet

På spillebrettet skal det plasseres 6 sylindere for hvert lag innenfor et gitt område. Sylinderene kan ligge i et av 10 forhåndsbestemte mønstre og dette mønsteret velges tilfeldig før begynnelsen av hver kamp. Det ene kameraet på roboten er stilt inn slik at det kan ta bilde av akkurat det området som mønsteret dekker når roboten står i sin startposisjon. Siden mønsteret kan konstrueres ved å plassere de 6 sylinderene rundt på 12 forskjellige faste posisjoner, søker kameraet etter den gitte lagfargen på akkurat disse punktene. Når bildet er tolket sendes informasjonen til den kunstige intelligensen som en tabell bestående av 0 og 1 avhengig av om det finnes en sylinder på den tilhørende posisjonen.

Ved mottak av denne tabellen gjør den kunstige intelligensen først en validering av resultatet. Den første testen resultatet må gjennom er om det finnes riktig antall 1-ere i tabellen. Dersom det ikke gjør det, er ikke alle sylinderene registrert og valideringen feiler. Den andre testen baserer seg på det faktum at alle mønstre er symmetriske om x-aksen. Dersom en av disse testene feiler betyr det at bildet som er tatt enten er korrupt eller at roboten kan være kommet litt ut av startposisjonen, slik at det er vanskeligere å finne sylinderene. Ved feil gang på gang vil kameramodulen bli bedt om å ta nytt bilde opp til 3 ganger før den kunstige intelligensen gir opp og velger en vilkårlig oppstilling for å forsøke å plukke opp sylinderene.

Detektering av ledige sektorer på poengområde

På spillebrettet er det 4 poengområder med forskjellig høyde og fasong. Poengområdene er felles for begge lag og byr dermed på potensielle konflikter ved at begge robotene vil levere på det samme området. Det er gitt av reglene at det er strengt forbudt å rive ned et byggverk som er satt opp av motstanderen, og det er her behovet for en sensor dukker opp.

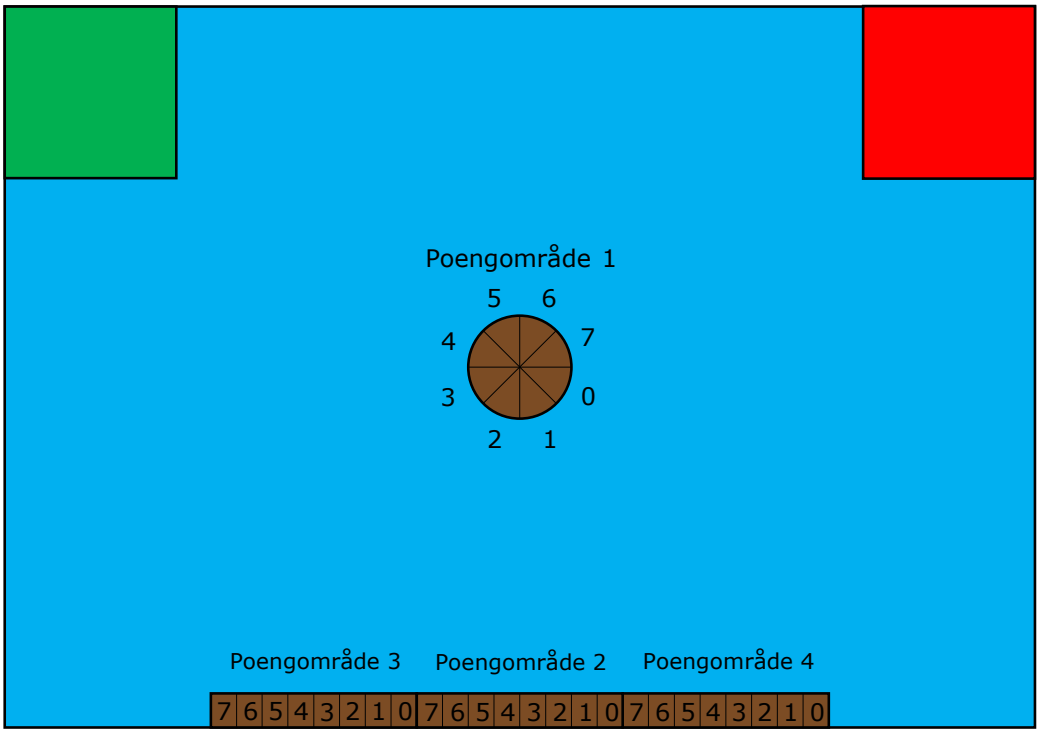
Kamera nr. 2 på roboten er satt i en vinkel slik at det kan ta bilde av hvilket som helst av de 4 poengområdene når roboten står i en gitt avstand fra området. Siden det både er ulik form og høyde på poengområdene må den kunstige intelligensen gi beskjed til bildeanalyse-metoden om hvilket poengområde roboten står foran, slik at den kan tolke bildet på riktig måte.

For å forenkle algoritmen som skal tolke bildet er hvert poengområde delt opp i sektorer som kan undersøkes separat. Figur 5.6 viser hvordan sektorene er inndelt og nummerert. På de tre poengområdene som står langs siden av spillebrettet vil roboten stille seg opp midt foran poengområdet for å ta bilde, for så å kjøre fram til riktig sektor på bakgrunn av dataene som tolkes av bildet. På det sentrale poengområdet er det derimot ikke like lett. Her kan roboten i teorien stille seg opp hvor som helst rundt poengområdet for å ta bilde, noe som vil gjøre tolkningen av hvor de forskjellige konstruksjonene befinner seg vanskelig. Måten dette er løst på er at roboten er begrenset til å kjøre inn til poengområdet kun på grensen mellom to sektorer. Når kameraet blir bedt om å ta bilde, blir det samtidig overført informasjon om hvilke sektorer roboten står mellom. Algoritmen for analyse av bildet kan dermed rotere returtabellen ut fra denne informasjonen slik at indeksene i returtabellen tilsvarer sektornummereringen slik den er definert i figur 5.6

Dataene som returneres til den kunstige intelligensen er en tabell med et element for hver sektor på poengområdet. Elementene har verdiene 0, 1, 2 eller 3 avhengig om den gitte sektoren har et byggverk og hvilken farge dette byggverket består av. Tabell 5.1 viser hvilke situasjoner de forskjellige verdiene tilsvarer. Så lenge en sektor delvis er dekket av en konstruksjon blir hele sektoren regnet som okkupert av denne konstruksjonen, og dersom det finnes konstruksjoner av begge farger innenfor en sektor blir den regnet som okkupert av motstanderen.

Verdi	Tolkning
0	Ledig sektor
1	Egen konstruksjon
2	Motstanderens konstruksjon
3	Konstruksjon fra begge lag

Tabell 5.1: Definisjoner for forskjellige tilstander til en sektor på poengområdet



Figur 5.6: Definisjon av sektorer for hvert poengområde

5.5.3 Trykkbrytere

På roboten er det plassert 3 trykkbrytere som er beskrevet i kapittel 7.2.3. Informasjonen fra disse bryterene brukes av den kunstige intelligensen til å stoppe roboten når den er kommet inntil poengområdet eller en dispenser.

Når roboten er klar til å kjøre helt inntil et poengområde blir det gitt beskjed til navigasjonssystemet om å kjøre rett frem med en gitt hastighet. Den kunstige intelligensen går i en tilstand der den venter på signal fra trykkbryterene som er montert i bunnplaten. Når en av bryterene blir trykket inn blir det gitt beskjed om å stoppe motoren på denne siden. Den andre motoren vil fortsette å gå helt til trykkbryteren på denne siden også er trykket inn. På denne måten stopper roboten først når den er helt inntil poengområdet. Når den kunstige intelligensen har registrert at begge bryterene er blitt trykket inn, går den over til neste tilstand i den aktive strategien.

Ved opplukking av tverrligger er det trykkbryteren i armen på midtheisen som brukes. Idet den kunstige intelligensen registrerer at denne bryteren er blitt trykket inn, blir begge motorene stoppet og den aktive strategien går til neste tilstand.

5.6 Kommunikasjon med andre moduler

Som beskrevet i dokumentasjonen [25] kommuniserer den kunstige intelligensen, som ligger i prosessen plansys, med navigasjonssystemet(navsys), kameramodulen(cvsys) og kretskortene rundt på roboten (via prosessen gateway). Kommunikasjonen innad mellom prosessene som kjøres på datamaskinen gjøres ved hjelp av posix-meldinger, mens kommunikasjonen fra prosessen gateway og ut til kretskortene går via CAN-bus.

Flere moduler på roboten har blitt utviklet av forskjellige personer. Ved å sammen bestemme et fast grensesnitt for kommunikasjon mellom modulene har det vært mulig for alle parter å utvikle systemet sitt uavhengig av å vite fullstendig virkemåte for de andre modulene. Dette har spesielt gjort seg gjeldende i samarbeidet med EiT-gruppen. I tillegg til å definere selve grensesnittet til disse modulene har det vært viktig å kontrollere utviklingen av modulene slik at de har nådd alle kravspesifikasjonene satt på forhånd. Dette er nærmere beskrevet i kapittel 8.6.

5.6.1 Grensesnitt til byggemodul

Det ble på et tidlig stadium i utviklingsprosessen definert et grensesnitt mellom den mekaniske byggemodulen utviklet av EiT-gruppen og den kunstige intelligensen. Grensesnittet består av meldinger som sendes mellom modulene med kommandoer og eventuell tilleggsinformasjon. Underveis i utviklingen ble grensesnittet oppdatert og endret, og det endelige grensesnittet er gitt i tabell 5.2.

Kommando	**Meldings-ID**	**Data**	
		byte 0	byte 1
Fra AI til byggemodul			
CAN_STOP_BUILDER	0x180		
CAN_RESTART_BUILDER	0x181		
CAN_READY_ARMS	0x182	next_claw	
CAN_LEVEL_ARMS	0x183	height	
CAN_PICKUP_BOARD	0x184	which_claw	next_claw
CAN_PICKUP_LINTEL	0x185		
CAN_PICKUP_DISPENSER	0x186	num_elements	
CAN_DELIVER	0x187	with_lintel	
CAN_CLEAN_UP	0x188	direction	
CAN_SET_WIPE	0x189	position	
CAN_PREP_LINTEL_PICKUP	0x18A		
CAN_DELIVER_ONLY_LINTEL	0x18B		
Fra byggemodul til AI			
CAN_ARMS_READY	0x190		
CAN_PICKUP_ACK	0x191	num_elements	type
CAN_DELIVER_ACK	0x192	with_lintel	
CAN_CLEAN_UP_ACK	0x193	direction	
CAN_SET_WIPE_ACK	0x194	position	

Tabell 5.2: Grensesnitt for meldinger mellom kunstig intelligens og byggemodul

5.6.2 Grensesnitt til kameramodul

Kameramodulen er utviklet av EiT-gruppen, slik at definering av et fast grensesnitt også her var en nødvendighet. Dette er gitt i tabell 5.3. Kameramodulen har ganske spesifikke og avgrensede oppgaver, så grensesnittet ble relativt enkelt i forhold til grensesnittet mot byggemodulen.

5.7 Testing

På grunn av at det tok lang tid å bli ferdig med den fysiske opplukkermodulen har det blitt liten tid til å teste det komplette systemet med kjøring på brettet, opplukking og levering av spilleelementer. Det er derfor lagt vekt på å teste den kunstige intelligensen underveis i utviklingen. Dette er gjort på forskjellige måter avhengig av hvilke deler på roboten som har vært ferdig utviklet.

5.7.1 Simulering av robot

Roboten er blitt simulert på flere forskjellige måter for å få testet funksjonaliteten til den kunstige intelligensen og de implementerte strategiene.

Manuell rotering av løpehjul

For å få testet hva som skjer når navigasjonssystemet kommer frem til ønskede waypoints uten å kunne kjøre på testbordet har løpehjulene blitt snurret manuelt til navigasjonssystemet har nådd målet sitt. Her er det via utskrifter i prosessen plansys blitt verifisert at den kunstige intelligensen registrerer at roboten er fremme og den aktive strategien går inn i neste tilstand for å utføre oppgaven på denne posisjonen.

Simulerte svar fra moduler

Ved å hardkode forskjellige simulerte svar fra andre moduler inn i de enkelte strategiene ble det mulig å kontrollere programflyten i strategiene, og det ble undersøkt om hver strategi oppførte seg på riktig måte i forhold til den tiltenkte funksjonaliteten.

Simulerte svar via CAN-bus

Denne fremgangsmåten går ut på å simulere meldinger fra alle de ulike kretskortene ved hjelp av et CAN-program på PC. Den viste seg å bli veldig nyttig for testing, siden den gjorde det mulig å gi forskjellige svar uten å måtte endre i kildekoden og kompilere på nytt for hver endring.

Kommando	**Meldings-ID**	**Datastruktur**
Fra AI til kameramodul		
MINOR_CV_TAKE_SCORING_PICTURE	42	camera_message_request_picture
MINOR_CV_TAKE_BOARD_CYLINDER_PICTURE	43	camera_message_request_picture
Fra kameramodul til AI		
MINOR_CV_BOARD_CYLINDERS	40	camera_message_board
MINOR_CV_SCORING_AREA	41	camera_message_scoring

Tabell 5.3: Grensesnitt for meldinger mellom kunstig intelligens og kameramodul

Oppsummering

Ved å kombinere disse tre teknikkene for å simulere forskjellige deler av roboten har det vært mulig å teste funksjonaliteten til den kunstige intelligensen uten å ha fysisk opplukkermodul, kamera eller navigasjonssystem tilgjengelig.

5.7.2 Kjøring med testrobot

I løpet av prosjektoppgaven i høst [9] ble det konstruert en testrobot som kunne kjøre rundt på spillebrettet, men ikke utføre noen oppgaver. Denne ble brukt for å teste hvordan den kunstige intelligensen valgte forskjellige strategier og kjørte til posisjonene for disse strategiene. Siden denne testroboten ikke hadde noen opplukkermodul måtte meldingene til den kunstige intelligensen simuleres ved hjelp av de overnevnte teknikkene.

5.7.3 Integrering av fysisk modul

Ved integrering av to systemer som er blitt utviklet parallellt av forskjellige personer, er det stor sjanse for at det har oppstått misforståelser underveis og at det komplette systemet ikke fungerer som forventet. Dette var også tilfelle ved integreringen av EiT-modulen og den kunstige intelligensen til roboten. På tross av et grundig utarbeidet grensesnitt, se seksjon 5.6.1, oppstod det situasjoner som ikke var forutsett på forhånd og måtte håndteres. Dette gjaldt spesielt hvordan enkelte sekvenser av meldinger fra den kunstige intelligensen førte til at byggemodulens forskjellige deler kom i veien for hverandre og kolliderte.

Når så dette kom på plass ble det mulig å gjennomføre en fullskala test av systemene på roboten. For å få roboten til å utføre oppgavene raskt ble det lagt ned mye arbeid i hvor lang tid det skulle gå før meldinger ble sendt fra byggemodulen ved opplukking og levering. Med en gang den delen av operasjonen som krevde at roboten stod i ro var ferdig, kunne modulen gi beskjed til den kunstige intelligensen om at operasjonen var fullført. Dermed kunne roboten begynne å bevege seg mot neste posisjon mens byggemodulen fullførte operasjonen.

5.8 Resultater og diskusjon

Den nye implementasjonen av strategisystemet har vist seg gjennom testing å oppfylle alle funksjonskrav for konkurransen. Siden det også ble stilt krav til hvordan systemet skulle implementeres, vil det her gjøres et forsøk på å sammenligne den nye implementasjonen med den som fantes ved starten av denne oppgaven. Listing 5.3 viser konstruktøren for superklassen til alle strategiene. I tillegg er en metode som kalles fra konstruktøren tatt med for å vise alle variablene som tilhører superklassen og som brukes der. Dette er akkurat den

samme klassen som er beskrevet i bakgrunnskapittelet, listing 5.1. Den tidligere implementasjonen var på 8 kodelinjer, mens den nye implementasjonen har vokst seg til ca 650 linjer ved å implementere generell kode som tidligere var plassert rundt i hver enkelt strategi. Dette er en rimelig størrelse med tanke på at hver enkelt strategi tidligere var på ca 400-500 kodelinjer, og at det fantes 10 aktive strategier i tillegg til 5 strategier som ikke ble brukt. En klasse bør heller ikke overstige denne størrelsen da det blir vanskelig å holde oversikt over klassens funksjoner.

For å gi et inntrykk av hvordan en strategi er implementert, er konstruktøren til en av strategiene gitt i listing 5.4. Ved sammenligning av konstruktøren til en av de gamle strategiene 5.2, ser man flere positive forskjeller i den nye implementasjonen:

1 Superklassen blir brukt til å sette generelle variabler
2 Det er flere og mer spesifikke utskrifter som viser statusen til den kunstige intelligensen
3 Det brukes enkle og forståelige funksjoner for å aksessere eller manipulere variabler i superklassen

Hver av de 5 nyimplementerte strategiene er bygget opp med denne strukturen og består av 100-250 kodelinjer. Tabell 5.4 viser noen nøkkeldata som viser forskjeller i strategisystemets kompleksitet før og nå. I denne tabellen er størrelsen gitt i antall semikolon, og er derfor noe mindre enn tidligere oppgitte antall kodelinjer. Antall semikolon angir mer presist kompleksiteten til en kildekode, siden det ikke tar hensyn til om programmereren skriver luftig eller kompakt kode. Den prosentvise delen av kommentarer og utskrifter er antall utskrifter delt på antall semikolon i koden.

Nøkkelinformasjon	Implementasjon	
	Opprinnelig	Ny
Antall strategier	15	5
Gjennomsnittlig størrelse per strategi	225	115
Størrelse til superklasse	1	350
Total størrelse av systemet	3387	928
Gjennomsnittlig mengde kommentarer og utskrifter	6,6%	22,6%

Tabell 5.4: Nøkkeltall som beskriver kompleksisteten til strategisystemet før og nå

Listing 5.5 er også tatt med for å vise h-filen til en av strategiene. Her kan man se at det kun finnes 3 funksjoner i strategien og at det ikke er noen variabler. Dette er gjort for å gjøre hver enkelt strategi så enkel som mulig. Det er lett å lage en ny strategi eller å endre på en av de eksisterende strategiene siden det meste av koden allerede finnes i superklassen.

```
#include "Observer.h"

Observer::Observer(PlansysObservation *s,
        int theTeamColor){
        _subject = s;
        _subject->attach(this);
        time_left = 90;
        teamColor = theTeamColor;
        cout << endl;
        reset_strategy();
        circle_building_sector = -1;
        first_run = TRUE;
}
void Observer::reset_strategy(){
        current_pos_number = 0;
        current_pos = -1;
        current_sector = -1;
        waypoint_reached = FALSE;
        strategy_done = FALSE;
        arms_ready = FALSE;
        camera_ready = FALSE;
        tower_ready = FALSE;
        pickup_ack = FALSE;
        deliver_ack = FALSE;
        left_switch = FALSE;
        right_switch = FALSE;
        dispenser_switch = FALSE;
        wipe_ack = FALSE;
        adjust_count = 0;
        time_left_second_counter = time_left;
        current_state_start = time_left;
        priority = 0;
        state = 0;
        cout << "Strategy reset!" << endl;
        _subject->setActiveWaypoint(FALSE);
}
```

Listing 5.3: Observer.cpp

```
#include "BoardCylinderStrategy.h"
#include "../include/observations.h"
#define PRINT 0

BoardCylinderStrategy::BoardCylinderStrategy
( PlansysObservation *s , int theTeamColor )
        : Observer(s,theTeamColor)
{
        choose_waypoints();
        cout << "Strategy added: ";
        cout << "BoardCylinderStrategy" << endl;
        print_current_waypoints();
        id = BOARD_CYLINDER_STRATEGY;
}
void BoardCylinderStrategy::choose_waypoints(){
        positions[0] = WP_BOARD_CYLINDERS;
        positions[1] = WP_BOARD_CYLINDERS_2;
        positions_size = 2;
}
```

Listing 5.4: BoardCylinderStrategy.cpp

```
class BoardCylinderStrategy: public Observer {
  public:
        BoardCylinderStrategy(PlansysObservation*,int);
        ~BoardCylinderStrategy();
        void updatePriority();
        void performStrategy( MessageHandler*,
        MessageHandler*, MessageHandler*);
        void choose_waypoints();

  private:
};
```

Listing 5.5: BoardCylinderStrategy.h

5.9 Konklusjon

Det er blitt uviklet et system for kunstig intelligens basert på et eksisterende rammeverk fra tidligere år. Denne kunstige intelligensen styrer roboten ved hjelp av forskjellige strategier som blir valgt ut fra hvilken tilstand roboten og spillebrettet befinner seg i. I implementasjonen er det lagt vekt på å gjøre koden så vedlikeholdbar som mulig ved å bruke modulbasert oppbygning, høy grad av dokumentasjon og prinsipper fra objektorientert programmering som abstraksjon, arv og innkapsling. Siden det arvede systemet allerede har blitt endret betydelig fra det ble designet for over 2 år siden, er det også lagt ned arbeid for å bygge opp igjen og beholde den opprinnelige strukturen til rammeverket.

Gjennom testing har den kunstige intelligensen vist seg å reagere på input som kommer fra spillebrettet via forskjellige sensorer, og bruke denne informasjonen til å ta riktige valg for å få så mange poeng som mulig i konkurransen.

Kapittel 6

Antikollisjonssystemet

Dette kapittelet tar for seg robotens antikollisjonssystem. Arbeidet med antikollisjonssystemet begynte allerede i prosjekteoppgaven i høst [9], og er blitt viderført i denne oppgaven. Det nyutviklede tårnet som beskrives i kapittel 3 har åpnet opp for store muligheter for å få roboten til å oppføre seg smart og beregne beste kollisjonsfrie rute frem til ønsket mål basert på motstanderens posisjon.

Antikollisjonssystemets oppbygning og virkemåte er grundig beskrevet i programvaredokumentasjonen [25], så dette kapittelet vil først og fremst ta for seg den praktiske delen med testing og verifisering av virkemåten til systemet.

6.1 Bakgrunn

Høsten 2008 ble det utviklet et design for antikollisjonssystem til bruk i Eurobot-konkurransen. Dette antikollisjonssystemet baserer seg på at det finnes sensorer på roboten som kan oppdage hvor motstanderen er på spillebrettet til enhver tid, og skal beregne kollisjonsfrie ruter for roboten som holder seg med størst mulig avstand til motstanderen. Virkemåten for dette systemet er nærmere beskrevet i [9]. Dette antikollisjonssystemet er først og fremst utviklet som en simulator og er skrevet i Java. Siden rammeverket til roboten baserer seg på C og C++ er det nødvendig å skrive om koden til C++ før det kan testes i full skala på roboten. Arbeidet med porteringen fra Java til C++ ble påbegynt i prosjektoppgaven, men ble ikke fullført. Blant annet gjenstår det å tilpasse koden slik at den kan integreres i rammeverket for kunstig intelligens på roboten.

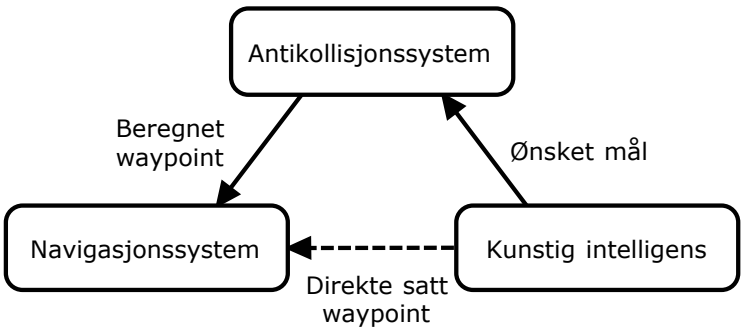
6.2 Hardware

Som beskrevet i kapittel 3 er det utviklet et konsept for detektering av motstanderens posisjon relativt egen posisjon. Denne modulen finner motstanderen over hele spillebrettet og kan gi vinkelen til motstanderen med 0,25 grads

oppløsning. Avstanden mellom robotene blir målt av 3 forskjellige måleprinsipper, og er derfor robust mot feilmålinger som kan komme av spilleobjekter på poengområdene eller ytre forstyrrelser. Disse målingene av motstanderens posisjon kommer med en frekvens på 2-3 Hz. Selv om dette er noe lavere enn det som var forventet ved utvikling av antikollisjonssystemet, kan denne oppdateringsfrekvensen ses på som tilfredsstillende for å unngå kollisjon.

6.3 Software

Antikollisjonssystemet er implementert som en selvstendig modul som kan brukes av den kunstige intelligensen til å planlegge en kollisjonsfri rute frem til det ønskede punktet. Denne modulen består av klassene Node, Edge, Obstruction og Tree, og knyttes sammen med resten av planleggingssystemet via filen Board.cpp. Selve implementasjonen er godt beskrevet i programvaredokumentasjonen [25]. Figur 6.1 viser hvordan antikollisjonssystemet fungerer som et mellomledd mellom den kunstige intelligensen og navigasjonssystemet. Slik systemet fungerte før, sendte den kunstige intelligensen waypoints direkte til navigasjonssystemet slik den stiplede linjen viser. Dette medfører at det må spesifiseres en hel liste med waypoints fra den kunstige intelligensen dersom roboten skal passere et objekt på vei til målet. Det nyutviklede antikollisjonssystemet fordeler arbeidet med å komme seg fram til målet slik at den kunstige intelligensen kun trenger å vite hvor den skal. Antikollisjonssystemet mottar denne meldingen og beregner hvordan roboten kan komme seg frem dit. Er det flere valg, vil systemet finne ut hvilken rute som er best basert på flere variabler. Deretter sender det kontinuerlig meldinger til navigasjonsystemet om hvor det bør kjøre for å holde seg på denne ruten. Ruten blir oppdatert med en frekvens på 10 Hz slik at enhver endring av egen eller motstanderens posisjon vil gjøre at ruten blir endret og at roboten fortsatt vil velge optimal rute frem til målet spesifisert i den kunstige intelligensen.



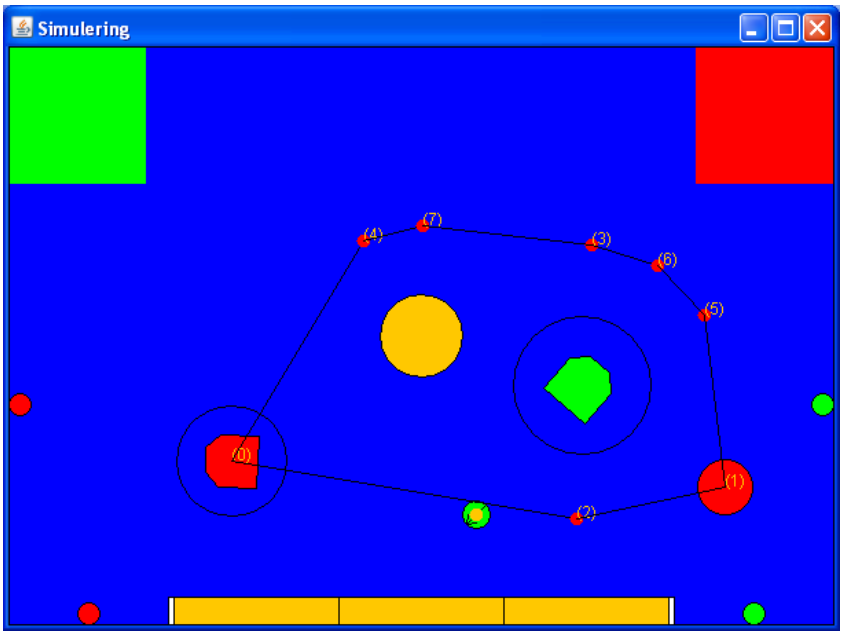
Figur 6.1: Kommunikasjon mellom den kunstige intelligensen, antikollisjonssystemet og navigasjonssystemet

6.4 Testing og resultater

Det er gjort tester på mange forskjellige nivåer under utviklingen av antikollisjonssystemet. Tester av spesifikke situasjoner er utført i simulatoren som ble utviklet i prosjektet. Etter hvert som testroboten kom på plass med et fungerende navigasjonssystem ble det mulig å teste den reelle C++-implementasjonen. Siden det roterende tårnet ble utviklet parallellt med dette systemet måtte motstanderen midlertidig simuleres ved å hardkode ulike posisjoner og ruter inn i koden til planleggingssystemet. Når så alle systemene på roboten var oppe og gikk, ble det mulig å montere motstanderbeacon på en dummyrobot laget av tre. Dette muliggjorde testing av antikollisjonssystemet basert på faktiske målinger fra det roterende tårnet.

6.4.1 Simulator

Som nevnt innledningsvis ble det utviklet en simulator i løpet av prosjektoppgaven som gjorde det mulig å teste virkemåten til antikollisjonssystemet i forskjellige situasjoner. Et skjermbilde av simulatoren er gitt i figur 6.2. I denne simulatoren er det meget lett å definere hvordan motstanderen skal oppføre seg, samt definere forskjellige scenarier som påviser om systemet har riktig funksjonalitet. Simulatoren har derfor blitt brukt i tidlig fase for å teste om de gitte algoritmene klarer å håndtere alle tenkelige problemstillinger roboten kan komme opp i.



Figur 6.2: Skjermbilde fra det grafiske grensesnittet i simulatoren

I tillegg til å vise grafisk hvordan roboten navigerer på brettet kan simula-

toren på kommando skrive ut forskjellig nyttig informasjon. Blant annet kan den skrive ut det komplette treet som beregnes frem til ønsket posisjon på formen gitt under:

```
Node (x-pos,y-pos), (Avstand til mål/Kostnad for denne noden) | ->
->(Kant til denne noden/Avstand), ...flere kanter...
```

Siden simulatoren har vært kjernen i utviklingen av antikollisjonssystemet, er det viktig at implementasjonen som skal kjøres på roboten har akkurat samme virkemåte. Ved å lage identiske testoppsett i simulatoren og C++-implementasjonen har det vært mulig å verifisere at porteringen fra Java til C++ har blitt gjort uten å endre funksjonaliteten til systemet. Resultatene fra begge systemene ved samme testoppsett er vist i utskrift 6.1. Her ser man at begge systemene har tilnærmet identisk funksjonalitet, det største avviket mellom waypointene som blir satt er på 4 mm.

6.4.2 Uten motstander

Etter å ha verifisert implementasjonen opp mot simulatoren var det mulig å teste kjøring på spillebrettet. Det første som ble gjort var å teste et fast eksempel fra simulatoren uten motstander på spillebrettet. I dette oppsettet skulle roboten kjøre rundt det sentrale poengområdet for å komme seg til målet i hjørnet av spillebrettet. Ved dette og flere andre eksempeloppsett ble det verifisert at roboten klarte å unngå forhåndsdefinerte hindere. Testene ble også brukt for å justere den nødvendige avstanden mellom roboten og hver av obstruksjonene på spilleflaten.

6.4.3 Simulert motstander

Neste steg på testhierarkiet bestod av å se hvordan roboten håndterer en bevegelig motstander. Siden det på dette tidspunktet i utviklingen ikke fantes sensorer for å oppdage motstanderen, ble det hardkodet inn i programmet en motstanderrobot. Denne motstanderen startet på et punkt på spillebrettet og flyttet seg i rett linje frem til et annet punkt. Figur 6.3 viser hvordan dette oppsettet så ut ved start. Den grønne roboten representerer egen robot, mens den røde forestiller motstanderroboten. Oppsettet ble valgt på denne måten for å teste om antikollisjonssystemet klarte å endre kurs dersom motstanderen blokkerte den optimale ruten. Ved utførelse av testen viste det seg at roboten oppførte seg akkurat slik den skulle. Roboten kjørte fremover mot målet på høyre side av poengområdet fordi dette var ruten som passerte motstanderen med størst avstand. Når roboten var kommet på høyde med poengområdet var den simulerte motstanderen kommet så nærme den valgte ruten at algoritmen valgte å kjøre på venstre side av poengområdet selv om dette medførte å kjøre en litt lenger rute. Roboten snudde derfor og kjørte i en fin halvsirkel rundt poengområdet, før den kjørte frem til målet.

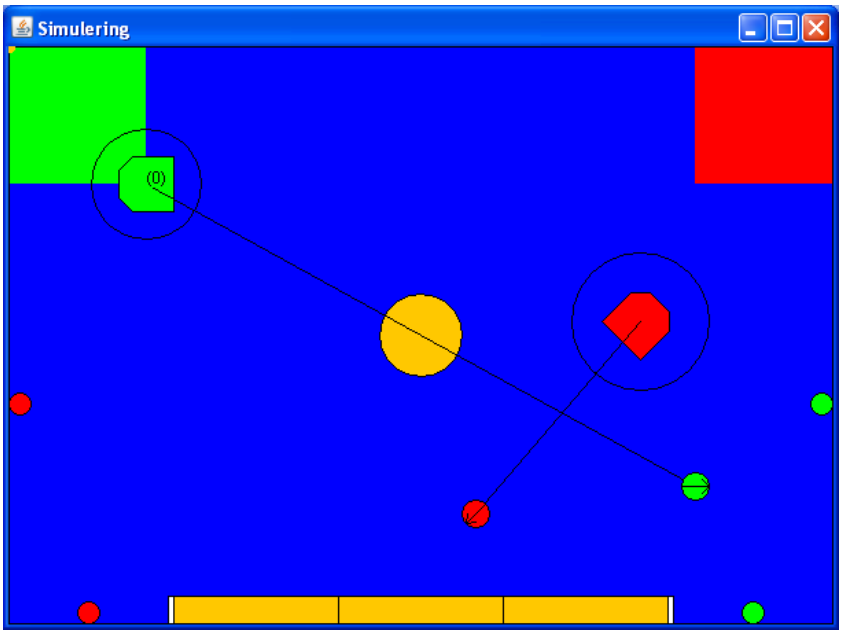
```
Fra C++:

Nodes | Edges
-------------
0 (500,500), (2282.54/4.4919) | (2/1208.82), (3/1208.5),
1 (2500,1600), (0/66.8386) |
2 (1307,1400), (1209.65/16.4944) | (1/1209.65),
3 (1692,699), (1210.23/54.6105) | (6/253.338), (7/625.991),
4 (1964,1300), (614.244/199.298) | (1/614.244),
5 (2635,699), (911.058/199.422) | (8/383.131),
6 (1864,885), (956.933/198.55) | (9/220.009),
7 (2300,550), (1068.88/200) | (5/366.642),
8 (2745,1066), (587.521/200.212) | (1/587.521),
9 (1862,1105), (807.508/199.346) | (4/220.066),

Fra Java:

Nodes | Edges
-------------
0 (500,500), (2282.54/4.4919) | (2/1209.33), (3/1209.33),
1 (2500,1600), (0.0/66.8385) |
2 (1305,1400), (1209.33/16.4956) | (1/1209.33),
3 (1690,695), (1209.33/54.8201) | (6/252.659), (7/625.420),
4 (1960,1300), (613.253/200.0) | (1/613.253),
5 (2635,695), (910.450/199.999) | (8/383.245),
6 (1860,880), (956.718/199.095) | (9/220.919),
7 (2300,550), (1068.86/200.0) | (5/366.915),
8 (2745,1065), (586.836/200.0) | (1/586.836),
9 (1860,1105), (806.472/199.999) | (4/219.759),
```

Utskrift 6.1: Utskrift fra simulatoren og robotimplementasjonen ved samme testoppsett



Figur 6.3: Oppsett i simulator for fysisk test av roboten med simulert motstander

6.4.4 Fysisk motstander

Den endelige testen av antikollisjonssystemet ble først mulig når sensorsystemet var ferdig utviklet. Det ble mye enklere å få til varierende tester med motstander siden ingenting måtte hardkodes inn i programmet. Under disse forsøkene ble det oppdaget og fikset flere problemer som oppstod i kommunikasjonen mellom antikollisjonssystemet og den kunstige intelligensen. I tillegg gjorde disse forsøkene oss oppmerksomme på problemene som kunne oppstå dersom motstanderen var veldig rask eller at antikollisjonssystemet gjorde en feilmåling av motstanderposisjonen. På bakgrunn av disse erfaringene ble det opprettet et sikkerhetssystem for å gjøre det umulig å kollidere i motstanderroboten. Dette systemet er beskrevet i kapittel 5.5.1.

6.5 Konklusjon

Basert på utviklingen av en simulator for ruteplanlegging og antikollisjon i høst, er det blitt implementert og testet et robust antikollisjonssystem på roboten. Dette antikollisjonssystemet hindrer ikke bare kollisjoner mellom roboten og obstruksjoner på brettet, men kan planlegge ruter framover i tid for å forhindre fremtidige kollisjoner. Dette er mulig på grunn av nye sensorer som er utviklet i løpet av oppgaven, i tillegg til at roboten hele tiden vet hvor den er på spillebrettet. Antikollisjonssystemet gjør at det er mulig å kjøre med større

hastighet siden motstanderens posisjon er kjent og ruter som beregnes lages med så stor avstand til motstanderen som mulig. Dette forenkler den videre utviklingsjobben av strategisystemet, siden man slipper å tenke på om det oppstår kollisjoner på vei til den ønskede posisjonen. Antikollisjonssystemet fungerer altså som et mellomledd mellom den kunstige intelligensen og navigasjonssystemet, og gjør at roboten hele tiden velger en kollisjonsfri rute frem til målet. Siden disse rutene blir oppdatert med høy frekvens, vil den også kunne reagere på dynamiske endringer som en bevegelig motstanderrobot, og fortsatt generere kollisjonsfrie ruter.

Kapittel 7

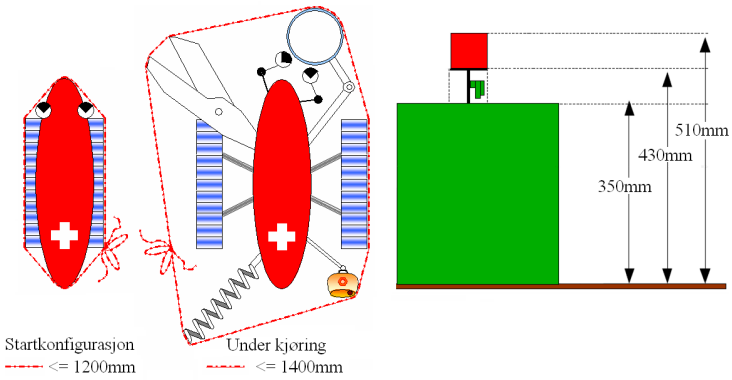
Robotkonstruksjon

For at roboten skal være i stand til å løse årets oppgave, tidligere beskrevet i kapittel 2.1.1, kreves det at mange forskjellige system må kunne integreres uten at de strenge størrelsesbegrensningene til roboten overskrides. Dette kapittelet har til hensikt å vise de forskjellige modulene roboten består av, og hvordan den totale sammenstillingsprosessen har foregått.

7.1 Bakgrunn

7.1.1 Størrelsesbegrensninger

For å kunne sørge for likt utgangspunkt for samtlige deltagende lag eksisterer det meget strenge størrelsesbegrensninger. Ved overskridelse av disse vil roboten ikke passere kvalifiseringen, og således ikke kunne delta i hverken innledende runder eller sluttspill. Reglene [20] er delt opp i hva som tillates av maksimal omkrets under oppstart og under kjøring, samt hvilken totalhøyde roboten kan ha, se figur 7.1.



Figur 7.1: Størrelsesbegrensninger gitt av gjeldene regler

7.1.2 Platå for motstanderbeacon

Om ønskelig kan et lag velge å plassere en beacon på motstanderens robot for enklere deteksjon og kollisjonsunngåelse. Det er derfor viktig at robotene innehar et platå som motstanderen kan sette sin beacon på. Det er ikke påbudt med et slikt platå, men om motstanderen krever det, vil motstanderen vinne på "walk over" om dette ikke er installert. Ved konstruksjon av platå gjelder følgende regler.

- Toppen av platået skal befinne seg på 430mm
- Platåets diameter skal være 80mm
- Oversiden skal være dekket med borrelås, type "krok" side
- Plassert nært horisontalt senter. Ved startkonfigurasjon skal distansen mellom platå og maksimal utstrekning til en side, ikke være mindre enn 50% av tilsvarende distanse på motsatt side

7.1.3 Eksisterende moduler

Fremdriftssystem

For at roboten skal være i stand til å utføre oppgaver forskjellige steder på konkurransebordet er det viktig med et fremdriftssystem som sørger for rask posisjonsforflytning. Til dette er det benyttet motormoduler bestående av motor og gir, med påhengte odometrimoduler, utviklet høsten 2006 [8]. Motordrivermodulene som drev motorene og avleste høyre og venstre løpehjul ble dog ikke sett på som gode, og derfor er disse blitt fornyet beskrevet i kapittel 4.

Brytermodul

Da roboten innehar flere forskjellige uavhengige systemer er det hensiktsmessig å dele opp spenningstilførselen i forskjellige kurser med tilhørende sikringer og felles nødstoppsbryter. I tillegg er det en nødvendighet å ha et grensesnitt mellom den bærbare pc'en og brytere, som kan gi beskjed til kunstig intelligens om startkommando og valg av lagets farge.

I forbindelse med prosjektoppgaven i høst, se [9, kap 4.1] ble det gjort et grundig arbeid med å utvikle en modulær brytermodul som innehar disse egenskapene. Panelet er i tillegg optimalisert med hensyn på plassopptagelse, samt at det er svært enkelt å flytte mellom roboter. For mer detaljert informasjon henvises det til [10, kap 8].

Acer Aspire One pc

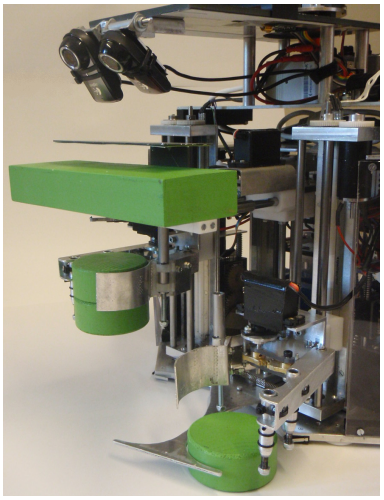
Da utviklingen av små billige bærbare pc'er med god ytelse har kommet langt, samt at det har vært en del uheldige episoder med åpne kretskort plassert rundt

på roboten, ble det foretatt et arbeid i å bytte ut den sentrale dataenheten til en Acer Aspire One pc, høsten 2008 [9, 4.2]. Denne pc'en er liten og kompakt, har egne batterier, samt alle nødvendige tilkoblinger for å kunne ta i bruk CAN bus og flere kameraer.

7.2 Utviklede moduler

7.2.1 EiT modul

Eurobot er et prosjekt som krever mye jobb, og således er det blitt en god tradisjon å kunne hyre inn to prosjektgrupper bestående av totalt ti studenter fra 4. årskurs via faget eksperter i team, se kapittel 8.6. Oppgaven disse ble tildelt var å konstruere modulen som finner de tilfeldig plasserte spilleelementene, og benytte disse til å bygge tempel på forskjellige poengområder. Konseptet som ble utviklet består av tre heiser og en sylinderfanger plassert på robotens fremside, se figur 7.2, hvorav sideheisene er utstyrt med klyper som er i stand til å plukke opp sylindere, og midtheisen, en arm som er i stand til å plukke opp tverrliggere. Sylinderfangeren er plassert helt nederst, og har til hensikt å fange sylindere som befinner seg innenfor et område foran roboten og føre dem inn i klypene.



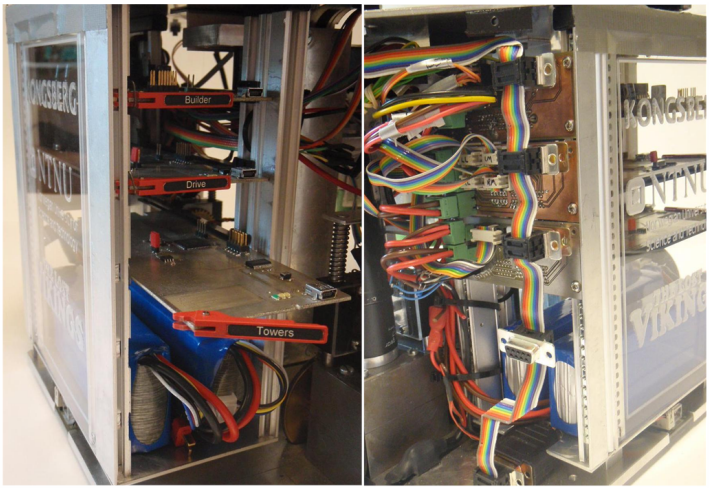
Figur 7.2: Tempelbyggermodul laget av EiT-studenter

I tillegg er det benyttet to kameraer, det ene for å finne sylindere utlagt på tilfeldige plasser, og det andre for å se til at roboten ikke bygger på et poengområde som allerede er bebygd av motstanderen. Heisenes høyde er valgt slik at roboten er i stand til å plassere et tempel oppå et annet tempel som allerede står plassert på det høyste poengområdet, slik at det er mulig å oppnå svært høye poengsummer. For mer informasjon om modulen henvises det til [3].

7.2.2 Kretskortramme

Kretskortene som styrer de forskjellige modulene er avhengig av sensorinformasjon og kommunikasjonsmuligheter for å utføre de gitte oppgaver. Dette resulterer i at svært mange ledninger må kunne tilkobles de respektive kretskortene, noe som igjen fører til et ledningskaos, og at det tar tid å bytte ut et kretskort.

Som en løsning på dette har det blitt tatt i bruk en modifisert 2RU kretskortramme, som har plass til 5 kretskort av størrelse 165,0x67,5mm, eller 3 kretskort av samme størrelse og to batterier. Ved bruk av kretskortramme flyttes alle tilkoblinger til passive bakplan, som tilhørende kretskort kan skyves ut og inn i, se figur 7.3.



Figur 7.3: Kretskortramme med utviklede kretskort

På denne måten kan samme kretskort trekkes ut fra en robot og skyves inn i en annen, uten at noen form for ekstra tilkoblinger er nødvendig. Dette kun ved å benytte to like kretskortrammer, med to like bakplan som begge er ferdig oppkoblet. På bakgrunn av dette er både testrobot, beskrevet i [9], og konkurranseroboten utstyrt med like kretskortrammer.

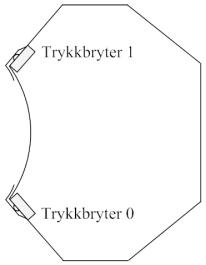
Overføringsmediumet mellom kretskort og bakplan er valgt til å være en 64pins PCI Express kontakt, som gir nok fleksibilitet hva angår antall sensorer og enheter kretskortet skal kontrollere. I tillegg innehar rammen mulighet for å låse kretskortene ved hjelp av håndtak og fjær, slik at de ikke sklir ut av PCI Express kontakten under vibrasjoner som måtte oppstå under kjøring.

Kretskortene som er utviklet for å passe i kretskortrammen er kommunikasjonsnode på egen robot, motordrivermodul, samt styringskortet til EiT modulen.

7.2.3 Trykkbrytere

Posisjonering ved poengområde

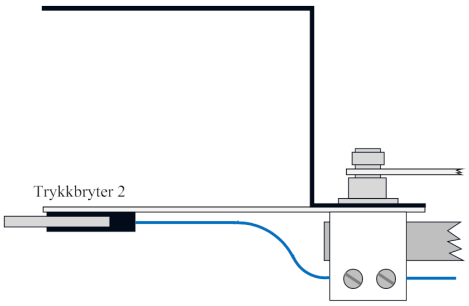
Ved bygging av tempel er det svært viktig å kunne forsikre seg om at de nederste byggelementene til et tempel befinner seg innenfor poengområdet, da grunnelementer plassert noe utenfor dette området gjør hele konstruksjonen null verdt. Det har derfor blitt utviklet et system basert på to brytere, som sørger for å fortelle roboten at den befinner seg helt inntil det ønskede poengområdet, se figur 7.4. Disse bryterene er konstruert slik at de vil bli trykket inn både ved levering på det sirkulære poengområdet, og på de rektangulære poengområdene.



Figur 7.4: Trykkbrytere innfelt i bunnplate for posisjonering ved poengområder

Posisjonering ved tverrliggerdispenser

Konseptet benyttet for å hente tverrligger fra dispenser går ut på å kjøre ut en arm under den aktuelle tverrliggeren, for så å løfte denne opp av dispenseren. Denne metoden medfører at armen som løfter tverrliggeren opp, må kunne kjøres tilstrekkelig under tverrliggeren for å kunne skape et stabilt feste. Dette resulterer i en kollisjonsrisiko mellom armen og veggen plassert bak dispenseren, og kan føre til at armen ødelegges ved kollisjon. Det er derfor montert en trykkbryter i armen, se figur 7.5, som sørger for at armen går maksimalt inn under tverrliggeren, men stopper opp før den kolliderer med veggen.



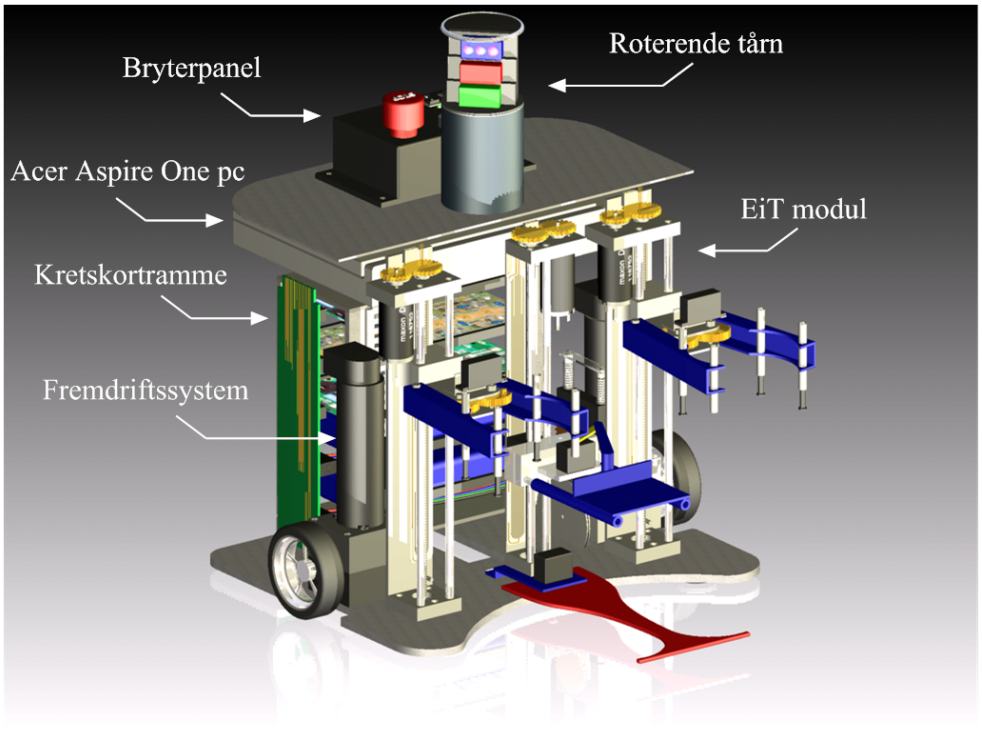
Figur 7.5: Trykkbryter innfelt i arm for posisjonering ved tverrliggerdispenser

7.2.4 Det roterende tårn

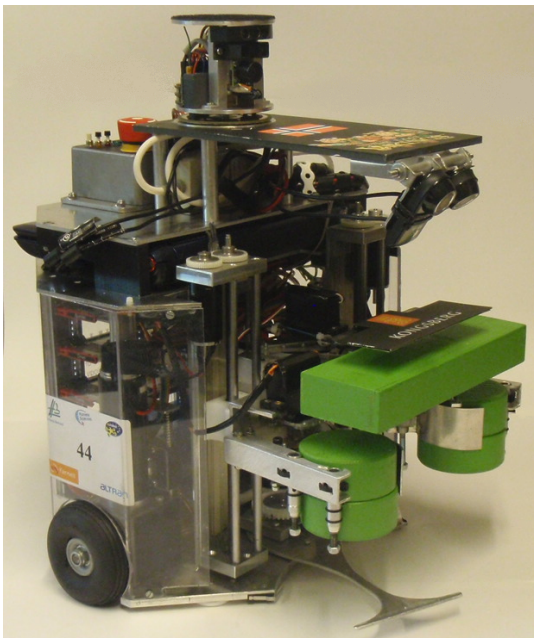
Roboten er avhengig av et godt og pålitelig antikollisjonssystem som er i stand til å detektere motstanderroboten før eventuell kollisjon oppstår. Til dette har det blitt utviklet et roterende tårn, beskrevet i kapittel 3.3.8, som må plasseres på en eksakt høyde lik 350mm og midt i robotens sentrum. Dette fordi toppen til modulen også fungerer som plasseringssted for motstanderens beacon, og at det er strenge regler på hvor dette området kan befinne seg.

7.3 Endelig sammenstilling

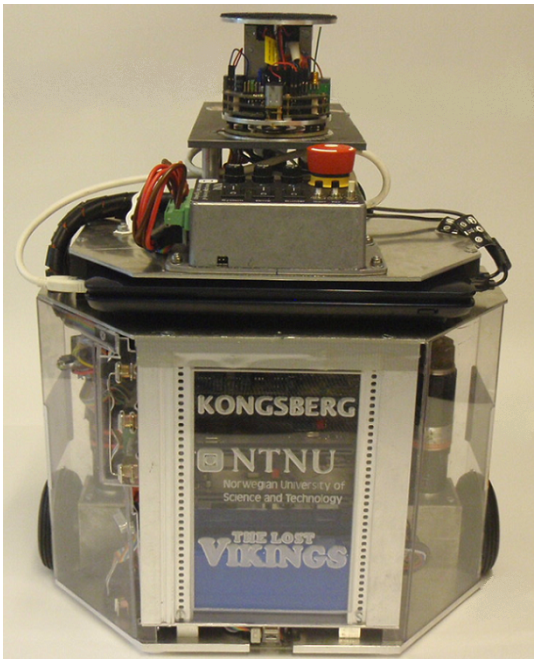
Basert på regler beskrevet i seksjon 7.1 ble alle eksisterende og nyutviklede moduler sammenstilt, i tett samarbeid med studenter fra Ekspterter i Team og instituttets verksted. Det ble i forkant utarbeidet en datamodell, se figur 7.6, som muligjorde maksimal plassutnyttelse, slik at størrelsesbegrensningene ble ivaretatt med god margin. Montering av kameraer, deksler, samt ledningsgater med tilhørende robust beskyttelse ble påmontert helt til slutt og figur 7.7 og 7.8 viser endelig konstruksjon av roboten.



Figur 7.6: Datamodell av robot i forkant av endelig sammenstilling



Figur 7.7: Endelig sammenstilling



Figur 7.8: Endelig sammenstilling

7.4 Konklusjon

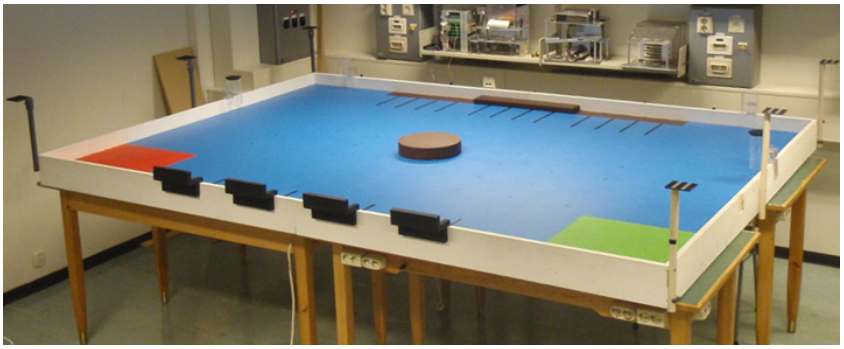
Robotens forskjellige moduler har blitt integrert smart og plasseffektivt, slik at roboten er i stand til å utføre årets oppgave på en svært tilfredstillende måte. Roboten er med denne konstruksjonen i stand til hente tverrliggere fra respektive dispensere, samt detektere tilfeldig utlagte sylindere og benytte disse til å bygge templer på alle tilgjengelige poengområder. Heisenes store rekkevidde muliggjør plassering av et tempel oppå et annet tempel som allerede står plassert på det høyeste poengområdet, og sørger for at svært høy poengsum er oppnåelig. I tillegg er roboten utstyrt med et roterende tårn som fungerer som antikollisjonssystem og som kontinuerlig oppdaterer den sentrale dataenheten om hvor motstanderroboten befinner seg.

Takket være god planlegging befinner totalsystemets størrelse seg innenfor plassbegrensningene gitt av reglene, og ledninger, kretskort, samt andre sårbare enheter er blitt beskyttet enten ved robuste ledningsgater eller plastikkdeksler plassert rundt roboten.

Kapittel 8

Diverse

8.1 Bygging av konkurransebord



Figur 8.1: Konstruert konkurransebord

Det har blitt konstruert et helt nytt konkurransebord da eksisterende bord var av så dårlig forfatning at roboten ikke var i stand til å forflytte seg, men kun ble stående å spinne. Det eksisterende konkurransebordet bar preg av dårlig konstruksjon, og at det har blitt flyttet flere ganger. Spilleflaten som var dekket av utallige tynne finèrplater hadde begynt å trekke seg sammen, slik at forskjellige partier på flaten var svært bølgete. Det ble derfor utviklet et nytt konkurransebord, se figur 8.1, basert på krav listet i tabell 8.1.

1	Helt jevn spilleflate
2	Bestå av to deler
3	Tåle å flyttes
4	Enkel sammensveising
5	Kunne benyttes i mange år

Tabell 8.1: Krav til nytt konkurransebord

Jevn spilleflate

Den store størrelsen på spilleflaten medfører at det ikke er mulig å benytte èn plate da plater av slik størrelse ikke produseres. Det er derfor nødvendig å benytte flere plater og sparkle skjøtene. For å unngå at disse platene over tid ikke endrer fasong grunnet flere malingstrøk, vannsøl, eller at de har blitt utsatt for noe tyngde, ble det valgt å benytte 16mm MDF-plater (Medium Density Fibreboard). MDF-plater har meget gode bearbeidelsesegenskaper samt at de egner seg godt for lakkering og maling. I tillegg er det ingen spenninger i materialet som kan bli utløst over tid, slik at flaten vil forbli jevn og rett, uavhengig maling, vannsøl, eller påført tyngde.

Tåle å flyttes

Det er benyttet 36x66mm bjelker som skjelett under MDF-platene, satt i et spesifikt system, slik at borddelene er stive uansett hvilken side et løft utføres på. Dette er vesentlig da stivhet sørger for at ikke skjøtene mellom MDF-platene sprekker opp og skaper en uønsket kant. Ved sammenstilling av de to borddelene benyttes gjennomgående bolter med tilhørende muttere til å klemme borddelene sammen på en enkel måte. Tykkelsen på MDF-platene gjør at disse kan klemmes meget hardt sammen uten at de ødelegges, og gjør det mulig å påføre stor tyngde på bordet uten av skjøten mellom bordelene blir synlig.

Oppsummering

Ved å benytte mer robuste materialer til skjelett, samt tykke MDF-plater til spilleflate, er vekten vesentlig økt i forhold til eksisterende spillebord. Dette er allikevel akseptabelt da et bord med lav vekt, men som ikke tåler å flyttes, uansett ikke vil være egnet. I tillegg vil bruk av tykke MDF-plater gjøre det mulig å pusse ned eventuelle riper og avvik som måtte oppstå underveis, og gjør bordet godt egnet som konkurransebord i flere år til.

Poengområdene er videre festet til spilleflaten ved hjelp av kitt, samt skruer skrudd inn fra sidekantene. Dette for å ikke påføre spilleflaten unødvendige skader, da poengområdene endres og flyttes hvert år mens spilleflaten forblir lik. Dette sørger for at et meget robust, solid, flyttbart og nøyaktig laget bord, står klart for neste års konkurranse uten unødvendige hull og arr.

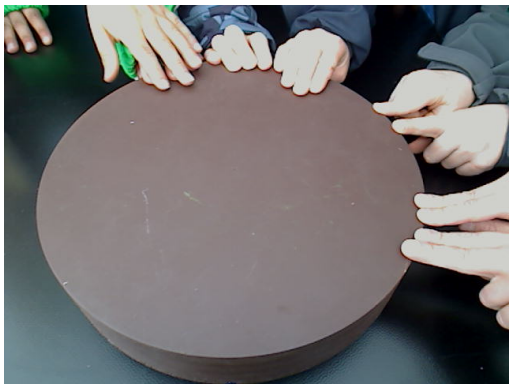
8.2 Instituttreklame

Roboten har blitt vist frem ved flere anledninger for å rekruttere studenter til NTNU, eller inspirere barn og unge til å velge realfag. I tillegg til å inspirere barn å unge til å velge studieløp ved NTNU, har disse arenaene også vært gode plasser for å promotere eurobotprosjektet, og i tabell 8.2 er de forskjellige anledningene opplistet.

To jentedager
Kreator09
Kolstaddagen
Diverse omvisninger

Tabell 8.2: Visninger av robot

Ved disse arrangementene har det vært tilskuere fra alle aldre, og bildet i figur 8.2 viser entusiastiske barn som holder fast poengområdet under arrangementet på Kolstaddagen. Bildet er tatt av kameraet på roboten som skal sjekke om det er mulig å levere på området.



Figur 8.2: Barnehender som holder fast poengområdet på Kolstaddagen

Det også blitt holdt et populærforedrag ved instituttet i etterkant av konkurransen for å øke kunnskapen lokalt om hva eurobotprosjektet innebærer, og hvilke reklameeffekter en god plassering i fremtiden vil kunne innebære. TV-adressa fulgte teamet før konkurransen, og skulle etter planen dekke konkurransen i etterkant, men av uvisse årsaker ble ikke dette fulgt opp fra deres side, selv om laget kom på en sterk 4.plass. Dette har gitt nyttige erfaringer slik at det vil bli knyttet flere mediekontakter i god tid før konkurransen neste år, samt at Norge Rundt allerede har sagt seg svært interressert i å følge det kommende laget.

8.3 Dokumentasjonspermer

Etter ti år med eurobotlag fra instituttet har det hopet seg opp med elektronikk og moduler som ingen aner hvordan fungerer. I tidligere diplomrapporter kan modulene stå beskrevet, mens mål, tilkoblingsoversikt, styringsprotokoller, samt skjemategninger ofte er av variabel art. Ved flere anledninger stemmer heller ikke aktuelle skjemategninger gjengitt i rapport med virkelige konstruksjoner, da komponenter og ledningsbaner er endret i etterkant. I tillegg er det svært sjelden slått fast at modulene faktisk fungerer etter hensikten, og således er motivasjonen lav for å forsøke å videreutvikle systemene.

Det som skiller de virkelig gode lagene er at disse er i stand til å videreutvikle moduler som fungerer til å bli enda bedre, samt fokusere på andre elementer når en modul fungerer godt nok. Dette krever god dokumentasjon på hva som er utviklet, som modulens virkemåte, mål, skjemategninger, endringer foretatt, samt noe av det viktigste, erfaringer gjort ved bruk. Dette er informasjon som ikke kommer godt nok til syne i en diplomrapport, samt at informasjonen ikke vil være samlet på ett oversiktlig sted.

På bakgrunn av dette er det utviklet to dokumentasjonspermer, se figur 8.3, en for software og en for hardware, som er tiltenkt å være et oppslagsverk for hele softwaresystemet og for alle fungerende moduler. Permene har som hensikt å vise systemene fullstendig og på en oversiktlig måte.

Figur 8.3: Dokumentasjonspermer

Hardwareperm

Retningslinjene ved skriving av hardwarepermen er at modulen beskrevet skal fungere. På denne måten kan en overflødig modul legges til side ett år, men allikevel ikke gå tapt da fremtidige studenter er i stand til å se at modulen er operativ. Permen er laget med et brennende ønske om at leser skal være i stand til å forstå budskapet, og om uformell tone kan bidra til dette, er dette lov, men ellers må følgende struktur gitt av tabell 8.3 følges.

1	Overskrift med bilde
2	Modulens funksjon
3	Nøkkelinfo
4	Hovedkomponenter
5	Kommunikasjon
6	Versjoner
7	Merknader
8	Kretsskjemaer
9	Layouts
10	Tilkoblingsoversikt
11	Tegning og mål
12	Sammenstillingsprosess (ved behov)
13	Kretskort transparenter (der dette er tilgjengelig)

Tabell 8.3: Krav til dokumentasjon gitt i hardwarepermen

Fungerende moduler som ved dags dato er beskrevet med gjeldene struktur i dokumentasjonsperm for hardware er:

1	Roterende tårn (Wall-E)
2	Motstanderbeacon (Eva)
3	Posisjonsbeacon A&C
4	Posisjonsbeacon B
5	Dispenserbeacon D
6	Kommunikasjonsnode
7	Motordrivermodul
8	Brytermodul

Tabell 8.4: Kapittelindeling i hardwarepermen

Softwareperm

Softwarepermen er laget for to formål. For det første skal den fungere som et oppslagsverk for all kode som kjøres på hovedprosesseringsenheten på roboten. Det andre formålet er at den skal kunne brukes for nye prosjekt- og masterstudenter til å bli kjent med systemet og lett kunne sette seg inn i strukturen og virkemåten til de forskjellige programmene. Det har vært et problem tidligere år at opplæring av nye studenter har blitt utelatt eller ikke gjort grundig nok. I år har dette også vært tilfelle siden det ikke var klart hvem som kom til å ta over før sommerferien var i gang. Dokumentasjonspermen blir derfor et viktig hjelpemiddel for denne kunnskapsoverføringen.

Permen er ment å videreføres fra år til år, og skal dokumentere det komplette fungerende systemet slik det ble brukt under konkurransen. Det er ønskelig at de tidligere års dokumentasjoner beholdes bakover i permen slik at det er mulig å se tilbake på hvordan systemet har opprinnelig blitt designet, og hvordan det har endret seg opp gjennom årene. Det er mulig at ideer og funksjonalitet som har blitt forkastet ett år på grunn av endring i oppgave viser seg å kunne brukes i en senere konkurranse. Sammen med dokumentasjonen på papir legges det ved en CD som inneholder all kildekode, samt latex-koden som er brukt til å skrive dokumentasjonen for å forenkle neste års dokumentasjon.

8.4 Modulbasert design

De modulene som er blitt utviklet i løpet av årets oppgave har blitt laget med tanke på kvalitet og brukervennlighet. Ved å gjøre følgende designknep oppnår man et modulbasert design som gjør at enhetene kan brukes videre av studenter flere år fremover:

- Standardiserte tilkoblinger
- Kompakt design der interne virkemåter ikke trenger å være kjent
- Enkle festemekanismer
- Grundig dokumentasjon

Sammenlignet med tidligere års håndverk er kretskortene produsert dette året av en egen klasse og er derfor mindre utsatt for feil som oppstår av dårlige loddinger eller andre produksjonssvakheter. Samtidig er det prøvd å innkapsle kretskortene i lukkede moduler der dette er mulig for å beskytte mot skader, støv og skitt.

8.5 Økonomi

Budsjettet for årets konkurranse ble satt opp i løpet av prosjektoppgaven og er gitt i [9]. Forrige års oppgave gikk med et overskudd på ca 20.000 kr som ble overført til årets oppgave. På grunn av høy utviklingsaktivitet, og at mange ble med på reisen til finalen ble derimot årets budsjett overskredet med 11.000 kr. Det er likvel en rest av fjorårets overskudd som overføres til neste års oppgave.

For å forsvare overskridelsen av budsjettet er det i løpet av dette årets oppgave kjøpt inn og laget mange moduler som kan brukes videre til neste års oppgave. Den bærbare pc'en representerer det største enkeltinnkjøpet og er allerede lovprist av studenter som planlegger å videreføre oppgaven. Det er også kjøpt inn flere motorer, servoer og potmetere som har veldig høy bruksfaktor for den mekaniske konstruksjonen som hvert år må lages på nytt.

Den andre faktoren som gjorde at budsjettet ble overskredet var at det var så mange av EiT-studentene som ble med på reisen til finalen. Det var på forhånd bestemt at hver enkelt skulle få dekket 2.500 kr for reise, i tillegg til overnatting, frokost, lunsj og transport fra Paris til La Fertè Bernard. Det viste seg at hele 10 personer ble med på reisen, og dette medførte en utgift på ca 40.000 kr. Det kan vurderes å høyne egenandelen til neste år dersom det er mange som ønsker å være med, for bare reisingen representerer langt over halvparten av budsjettets rammer.

Hovedsponsor

Kongsberg Gruppen har vært hovedsponsor for 4. år på rad, og samarbeidet med bedriften har fungert meget bra. Alle utgifter vedrørende utviklingskostnader og reise har blitt dekket, og til gjengjeld er det blitt forsøkt å promotere kongsberg ved alle aktuelle anledninger. I tillegg har kontaktpersoner i Kongsberg Gruppen blitt kontinuerlig oppdatert om hvordan prosessen har gått fremover, samt at disse har fått se demonstrasjoner av de utviklede systemene under karrieredagen høsten 2008, og rekrutteringsdagen våren 2009.

8.6 Administrering av EiT-grupper

I arbeidet med utviklingen av roboten har vi som tidligere nevnt hatt hjelp av to grupper fra faget "'Eksperter i Team"'(EiT). Disse studentene har laget den mekaniske delen som skal plukke opp spilleelementer og gjøre den poenggivende delen av konkurransen.

8.6.1 Formulering av oppgave

Av erfaring fra fjorårets EiT-samarbeid ble det valgt å slå sammen begge gruppene til en stor fellesgruppe. Dette ble gjort fordi det meget lett kan oppstå konflikter og konkurransepreg dersom det legges opp til å la hver gruppe komme

opp med et designkonsept, for så å velge et felles design. Som oppgavestillere ble det valgt å gi gruppen mulighet til å selv velge hvordan de ville løse oppgaven, innenfor rammene satt av spillereglene og størrelsesbegrensniger gitt av eksisterende moduler på roboten. Samtidig som at gruppene fikk frie tøyler for valg av design ble det satt flere krav til funksjonalitet og tidsfrister som måtte overholdes.

8.6.2 Arbeidsmengde

Det har vist seg å være mer arbeid å administrere disse gruppene enn først antatt. Selv med tett samarbeid opp mot disse gruppene ble det etter hvert klart at noen av kravspesifikasjonene som var satt for modulen ikke ble overholdt, samtidig som at de gitte tidsfrister ble overskredet. Det ble derfor gjennomført et krisemøte like før påske for å minne om kravene som modulen måtte innrette seg etter, og for å motivere studentene til å stå på for å klare å bli ferdig innen nye frister som ble satt.

Oppgaven som er gitt til gruppene har vært for stor og arbeidskrevende i forhold til hva faget på 7,5 studipoeng er lagt opp til. Dette har ført til at på tross av iherdig innsats og mye arbeid på fritiden, har ikke EiT-gruppene klart å overholde tidsfristene som ble satt. Det må vurderes å endre EiT-gruppenes funksjon fremover for å få til et godt samarbeid der konkurransedeltagelsen ikke står og faller på at EiT-studentene bruker fritiden sin på å komme i mål med sin oppgave.

8.6.3 Konklusjon

Bortsett fra problemene med den store arbeidsmengen, har samarbeidet med EiT-gruppene fungert veldig fint. Studentene klarte å utvikle et fungerende design, og konstruerte den mekaniske oppbygningen sammen med mekanikerlærlingen akkurat i tide før avreise til finalen i Frankrike. Modulen som ble laget er beskrevet nærmere i kapittel 7.2.1.

8.7 Rekruttering av nye studenter

Hvert år har det vært en eller flere studenter som har jobbet med eurobot som prosjekt- og masteroppgave. Alle disse studentene har vært tilknyttet institutt for teknisk kybernetikk, og de siste årene har det vært to studenter som har jobbet sammen med å utvikle roboten for deltakelse i konkurransen.

8.7.1 Samarbeid mellom flere institutter

Arbeidet med roboten er spredt på mange ulike fagfelt. Her er det oppgaver innenfor elektronikk, mekanisk design og programmering, for å nevne noen av hovedfeltene. For å finne spesialkompetanse innenfor alle områdene må det

derfor søkes utenfor teknisk kybernetikk, som til nå har styrt prosjektet alene. Det er derfor satt i gang et arbeid med å knytte sammen flere institutter til oppgaven, spesielt gjelder dette institutt for produktutvikling og produksjon og institutt for datateknikk. Arbeidet begynte allerede i høst med at det ble formulert en diplomoppgave på institutt for datateknikk innen datasyn. Dessverre var det ingen som valgte oppgaven, men det ble knyttet kontakt med faglærer på instituttet for datateknikk, som er meget positiv til å få til et fremtidig samarbeid.

8.7.2 Kontinuitet i prosjektet

Eurobot har den fordelen med at 4. klasse-studenter får mulighet til å være med på prosjektet gjennom faget Eksperter i Team. Ved at noen fra denne gruppen går videre med prosjekt- og diplomoppgave beholdes det en viss kontinuitet i prosjektet, og erfaringer kan lettere overføres gjennom samarbeidet før og under konkurransen. Ved at det har vært et sterkt fokus på å knytte sammen flere institutter til oppgaven, ble det tydelig lagt frem for alle deltagerene i EiT-gruppen at det var muligheter for å jobbe videre med eurobot som prosjekt- og diplomoppgave. Det endte opp med at to datastudenter og en maskinstudent bestemte seg for å ta eurobot som prosjektoppgave, mens ingen av deltagerene fra teknisk kybernetikk valgte å jobbe videre med eurobot. Dette medførte at det ble tatt kontakt med andre mulige kandidater fra teknisk kybernetikk, og funnet en som hadde ønske om å jobbe med oppgaven.

8.7.3 Oppsummering

Den nye sammensetningen for eurobot-teamet 2010 er dermed:

- En student fra institutt for teknisk kybernetikk
- En student fra institutt for produktutvikling og produksjon
- To studenter fra institutt for datateknikk

Der den ene av datastudentene skal jobbe med datasyn, den andre skal jobbe med kunstig intelligens, maskinstudenten har ytret ønske om å jobbe med design av roboten, samt forbedre odometri/motor-modulen og kybstudenten skal jobbe med robotens navigasjonssystem. Denne tverrfaglige kombinasjonen ser ut til å være en ideell sammensetning, og åpner for muligheter som aldri tidligere har latt seg gjøre i prosjektet. Det er interessant å se at denne sammensetningen nesten er identisk med den som er foreslått i videre anbefalinger i [7, kap. 8.2]. Dette viser at det er flere som har tenkt på denne tverrfaglige sammensetningen, men at det først nå er blitt en realitet.

8.8 Frakt

8.8.1 Pakking

I år som i fjor ble det lagt ned en del arbeid i å pakke roboten forsvarlig. Grunnet frykt for at klyper, heiselementer, samt beaconsystemet ikke ville tåle frakten ble disse demontert og surret inn i bobleplast for seg selv. For sikker frakt av roboten ble det benyttet en meget robust plastkasse med en størrelse som muliggjør bruk av mye boble- og skumplast rundt roboten. Alt nødvendig utstyr og verktøy ble ellers fordelt jevnt på alle som skulle reise, noe som ikke var noe problem da hele ti personer fra laget ble med. Av utstyr som ikke ble tatt med, men som anbefales sterkt å ta med neste år, er oscilloscope. Under feilsøking er dette verktøyet svært avgjørende for å kunne detektere eventuelle feilkilder raskt.

En siste ting som anbefales er at kretskort, da spesielt kretskortene som befinner seg i kretskortrammen, ikke bør være integrert i robot under frakten. ESD-skader som følge av pakking med statiske beskyttelseselementer som skumplast er svært vanskelig å håndtere, og således burde alle kretskort tas ut og pakkes i ESD-sikre poser for seg selv.

8.8.2 Forebygging av fraktskader

Det ble valgt å ikke pakke utstyr og verktøy i samme kasse som roboten, da dette ble gjort i fjor og at daværende robot ble påført en del fraktskader. I tillegg ble det etter uttallige telefoner til KLMs hovedkontor i nederland godkjent at roboten kunne fraktes som en skjør gjenstand, og at roboten derfor ville bli behandlet skånsomt. Da hele ti personer var med på reisen medførte denne servicen heller ingen ekstra kostnad.

Det skulle vise seg at en slik ekstra service, i praksis kun gjelder frakt av gjenstander inn på flyet. Stuere som tar bagasjen ut av flyet, aner ingen ting om at gjenstanden de henter ut er skjør, da de ikke legger merke til "Fragile" og "This side up" merknader. Skrekkslagne lagdeltagere ble vitne til roboten i et flere meter fritt svev over toppen av en baggasjevogn. Det er derfor ikke nødvendig å bruke tid og ressurser på å skaffe slik ekstra service, men undersøke mulighetene for å bestille eget sete til roboten, spesielt på veien ned til konkurransen. Om dette gjøres er det viktig å undersøke mulighetene for at roboten kan bli stoppet i sikkerhetskontrollen. Avslutningsvis må det nevnes at roboten ikke tok skade av den dårlige behandlingen og at pakkingen derfor ble sett på som svært god.

8.9 Reservedeler

Det ble under utvikling av roboten til konkurransen i heidelberg 2008 innført gode rutiner på å fremskaffe nødvendige reservedeler. Disse rutinene går ut på

å kontinuerlig detektere hvilke enheter som er utsatt for slitasje, samt enheter som blir sett på som svært nødvendige for at systemet skal kunne fungere, og sørge for innkjøp eller produksjon av disse på et tidlig tidspunkt. I år var det en del enheter som ble utsatt for slitasje da systemet inneholdt tre heiser samt fire servoer, godt beskrevet i [3], og det var derfor nødvendig med en betydelig mengde reservedeler. Eventuell dødtid hos mekaniker ble derfor fylt opp med produksjon av ekstra deler, samt at det ble foretatt nødvendig innkjøp av sensorer, servoer og nødvendige komponenter fortløpende.

Kapittel 9

Testing og resultater

9.1 Testing før avreise

Den siste uken før avreise ble det en meget hektisk periode. EiT-modulen ble ikke ferdigstilt før dette tidspunktet, og det var endelig mulig å teste det komplette systemet. Den opprinnelige strategien som var tenkt med å plukke opp sylindere fra dispenser måtte forkastes på grunn av for dårlig robusthet og hastighet til EiT-modulen. I stedet ble det laget en ny strategi som skulle plukke opp de to siste sylinderene fra bordet etter å ha plukket opp de fire første. Det endelige systemet utførte følgende sekvens av strategier dersom motstanderroboten ikke kom i veien eller bygde på aktuelt poengområde:

1. Opplukking av fire sylindere fra bordet
2. Levering av et stort tempel (to stabler à to sylindre og en tverrligger oppå) på nærmeste sektor på poengområdet
3. Opplukking av to sylindere fra bordet
4. Henting av tverrligger fra dispenser
5. Levering av et lite tempel (to sylindere med en tverrligger oppå) oppå det eksisterende tempelet

Når denne sekvensen av strategier var ferdigutført var det mellom 4 og 8 sekunder igjen av konkurransetiden på 90 sekunder, så dette er den absolutte maksimale poengsummen roboten kan klare. Dersom roboten klarer å bygge dette gir konstruksjonen 62 poeng.

9.2 Konkurransen

Årets konkurranse befant seg i La Fertè-Bernard i Frankrike, samme sted som for to år siden. Før deltagelsen i denne finalen har lagets navn blitt endret fra Legend of Norway til Lost Vikings, og fått matchende logo til navnet.

9.2.1 Utpakking og klargjøring av roboten

Ved ankomst til byen La Fertè-Bernard var det allerede blitt sent på kvelden. Siden roboten skulle gjennom homologering i løpet av morgenen dagen etter, ble det bestemt at roboten burde pakkes opp og monteres på ankomstkvelden. Etter at roboten var montert og undersøkt for eventuelle skader fra flyturen, skulle den bare testes en gang for å verifisere at alt var i orden. Fokuset ble endret da det plutselig ble klart at det var noe alvorlig galt med roboten; heisene på roboten kjørte ikke til riktig posisjon når de ble startet opp. Mange forskjellige tester ble kjørt for å finne ut hva som var galt, og det ble bestemt at mikrokontrolleren på EiT-kortet skulle byttes ut med den vi hadde i reserve. Etter dette var gjort så det ut som roboten var i orden, og det var på tide å få litt søvn, da klokken var blitt 5 på natten.

Neste morgen skulle roboten testes før den skulle kontrolleres i homologeringen. Det viste seg at nattens reparasjoner ikke hadde løst problemet likevel. Vi var heldige å få låne et oscilloscope fra et av de andre lagene, og med dette hjelpemiddelet ble problemet identifisert. Ved produksjon av kretskortene på roboten er det brukt et flussmiddel som er ledende og som må vaskes ordentlig for å forsvinne. Ved inspeksjon av EiT-kortet var det tydelig å se at det ikke var ordentlig vasket, og har ført til delvise kortslutninger rundt om på kretskortet. Kortet ble demontert, skrapet og vasket, og ved hjelp av multimeter ble det verifisert uendelig motstand mellom alle lederene. Roboten var dermed klar for homologering.

9.2.2 Homologering

Homologering kommer av det franske ordet "homologation" og betyr at roboten må godkjennes for de fysiske dimensjonene spesifisert i reglene, i tillegg til en funksjonstest av roboten. Roboten må vise at den kan score poeng uten en interfererende motstander og at den klarer å unngå en dummyrobot uten å kollidere.

Roboten gikk greit gjennom den fysiske kontrollen, selv om flere av dommerene kommenterte laserene på det roterende tårnet. Når det kom til funksjonstesten ble denne delt opp i to runder. Først skulle roboten score poeng uten motstander på bordet. Roboten gjorde som på testbordet hjemme i Trondheim og leverte to tårn oppå hverandre.

I den neste testen skulle roboten unngå en dummyrobot laget av tre. Dommeren bevegde dummyroboten rundt på spillebrettet, og roboten stoppet opp like foran motstanderen hver eneste gang. Siden roboten kun var satt opp til å kjøre på egen side av spillebrettet fikk den ikke vist fram så mye smart kjøring, men den unngikk å kollidere med motstanderen, og roboten var dermed godkjent for å delta i eurobot-finalen. I motsetning til hva som har blitt gjort tidligere år, ble det faktiske programmet som skulle kjøres under konkurransen brukt under homologeringen.

9.2.3 Innledende runder

Eurobot-konkurransen består av innledende runder der hvert lag får spille 5 kvalifiseringskamper før sluttspillet begynner. Siden de franske lagene hadde sin egen nasjonale kvalifisering parallellt med den internasjonale konkurransen, ble disse lagene først med i de to siste kvalifiseringsrundene. Disse måtte derfor spille henholdsvis to og tre kamper i fjerde og femte runde for å få likt antall kamper som de andre lagene.

1. runde

Den første runden ble spilt på kvelden etter homologeringen. Roboten hadde bestått homologeringen tidligere på dagen og alt var klart til at konkurransen skulle begynne. Laget vi skulle møte i denne kampen var "Robosib" fra Romania. Roboten startet fint og plukket opp og leverte det første tempelet problemfritt. Etter å ha plukket opp de neste to sylinderene skulle roboten kjøre for å plukke opp tverrliggeren. Roboten traff perfekt midt på tverrliggeren, men ble stående der i noen sekunder før den rygget tilbake uten å plukke opp tverrliggeren. Roboten fikk likevel levert de to sylinderene oppå det eksisterende tårnet og endte opp med å få 41 poeng for byggverket og 10 poeng ekstra for seier. Med disse 51 poengene havnet Lost Vikings på 2. plass, og stemningen gikk rett i taket hos lagets deltagere.

Feilsøking etter 1. runde

Etter den første runden ble det kjørt mange tester for å finne ut hvorfor tverrliggeren ikke ble plukket opp. Igjen ble vi overrasket av mystiske feil, denne gangen var det det roterende tårnet som begynte å oppføre seg feil. Kretskortene på denne modulen har blitt vasket ordentlig ved produksjon, så det var ingen mistanke om at samme feilen som på EiT-kortet slo ut her. Det var sent på kvelden, og siden alle var trøtte fra forrige natt var det ikke lurt å ta en ny natt med feilsøking. Oppgitte gikk laget Lost Vikings og la seg uten å ha en fungerende robot for morgendagens kamper.

Dagen etter ble det etter mye testing og tenking oppdaget at den høye luftfuktigheten der nede førte til feil som ikke oppsto i Norge. Selv kretskortene som var blitt ordentlig rengjort fikk problemer på grunn av flussmiddelet som var brukt under loddingen. Når problemet nå endelig var oppdaget måtte man finne en løsning. Utformingen på kretskortene på det roterende tårnet og selve tårnets oppygning gjorde det tilnærmet umulig å demontere og vaske kretskortene slik det ble gjort med EiT-kortet. Løsningen ble en hårtørker kjøpt på det lokale supermarkedet som kunne varme opp kretskortene før kamp slik at det ikke oppstod kortslutninger.

2. runde

Siden roboten nå lå på 2. plass ble den møtt av andre lag som lå helt i toppen av poenglisten. Neste utfordrer var "Unict team" fra Italia. Med godt mot og hårføner backstage entret Lost Vikings sin andre kamp. Men desverre oppstod det små komplikasjoner under oppstartsprosedyren. Hvert lag har 3 minutter på å forberede roboten til hver kamp, og dette skulle være tilstrekkelig også for vår oppstartsprosedyre. På grunn av at ledninger ble satt inn i feil rekkefølge måtte programmene på roboten startes på nytt og tidsgrensen på 3 minutt ble overskredet. I forvirringen og stresset med å få roboten klar ble det glemt å starte et av programmene på roboten, nemlig kameraprogrammet. Dette medførte at roboten ikke klarte å gjenkjenne sylinderenes plassering, og standardplasseringen ble valgt. Siden roboten da gikk i blinde klarte den kun å plukke opp to sylindere, men den klarte faktisk å levere disse sylinderene på poengområdet, og scoret dermed 7 poeng. Når roboten videre skulle plukke opp sylindere fra bordet kjørte den slik at den plukket opp to sylindere på en gang, og dette medførte at hele systemet ble "jammet". Siden vi ikke ville at roboten skulle bli ødelagt ba vi dommerene om å stoppe roboten ved å trykke på nødstopp-bryteren.

På grunn av overskridelsen av tidsgrensen og fordi dommerene måtte stoppe roboten fikk laget to advarsler denne runden. På tross av uhellet denne runden hadde motstanderen enda mer uflaks, og de 7 poengene var nok til seier. Total poengsum denne runden ble da 17 poeng.

3. runde

Den tredje runden ble spilt senere på dagen, og i denne runden var motstanderen "Dynamo Rapperswil" fra Sveits. Før denne runden var oppstartsrutinen nøye gjennomgått og skrevet opp på en lapp for at samme som forrige runde ikke skulle skje igjen. Roboten ble startet uten problemer denne gangen og plukket opp fire sylindere fra bordet som vanlig. Motstanderen klarte å levere sitt tårn nesten på akkurat samme tidspunkt som vår, men klarte å rive ned sitt eget tårn. Dette medførte at tverrliggeren på vår konstruksjon ble liggende såvidt oppå motstanderens veltede tverrligger og ble ikke poenggivende. Heller ikke denne runden ble tverrliggeren plukket opp, og nå ble det et stort mysterium på hvorfor trykkbryteren ikke registrerte trykket inntil tverrliggerdispenseren. Det ble seier også i denne runden med 14 poeng pluss 10 for seieren, totalt 24 poeng.

Feilsøking etter 3. runde

Siden roboten nå hadde unnlatt å plukke opp tverrliggeren 2 av 2 ganger, var det på tide å finne ut hva som var feilen. Det ble gjennomført storstilt testing av trykkbryteren, og det viste seg at ca. 1 av 100 meldinger ikke kom frem til den kunstige intelligensen. Det er meget lite sannsynlig at denne feilen har slått

inn i 2 av 2 forsøk. Det ble derfor i tillegg til å fjerne kondensatoren like ved bryteren, gjort modifikasjoner på heisarmen som kjøres ut under tverrliggeren. Modifikasjonen sørget for at trykkbryteren kom lenger frem, og gjorde det mulig for roboten å komme inn under tverrligger med skjev vinkel.

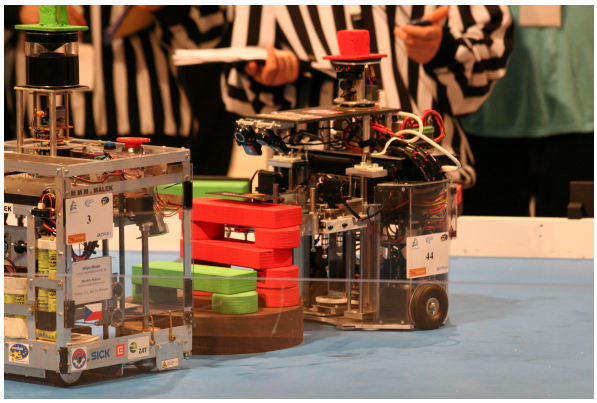
Til slutt ble det oppdaget at tverrliggerene som ble brukt under konkurransen var 2mm for brede i forhold til spesifikasjonen. Dette gjorde at trykkbryteren aldri hadde blitt trykket inn ved tverrliggerdispenseren, og siden vi hadde med egne spilleelementer til testing, ble dette ikke oppdaget. Problemet ble løst ved å justere trykkbryteren litt frem og blikkplaten som skal holde tverrliggeren på plass litt bakover.

4. runde

Fjerde runde var den første kampen på den siste konkurransedagen. Hver av de neste kampene ville bli spilt med svært korte mellomrom. Denne runden var første mulighet for å møte et fransk lag, og siden vi fortsatt lå blant de 5 beste lagene møtte vi et av disse. "ETS Montreal - ENSAM Paris" var navnet på laget og bestod av en blanding av studenter fra Frankrike og Canada. Denne gangen var vi sikre på at alt skulle fungere og at tverrliggeren kom til å bli plukket opp. Etter å ha levert det første tempelet begynte roboten plutselig å oppføre seg unormalt. Med meget høy hastighet kjørte den frem og tilbake og roterte rundt egen akse i helt tilfeldig mønster. Dette var et utslag av luftfuktigheten på enda et kort, nemlig motordriverkortet. Roboten fikk vist at den har mulighet til å kjøre med mye høyere hastighet enn den som blir brukt under konkurransen, men klarte ikke å score flere poeng. Heldigvis krasjet ikke roboten i verken motstanderen eller andre obstruksjoner og stoppet etter 90 sekund. Kampen endte uavgjort siden begge hadde bygget et stort tempel på det sentrale poengområdet. Dette resulterte i 29 poeng pluss 6 poeng for uavgjort = 35 poeng totalt.

5. runde

Etter så mange forskjellige feil og uhell var det på tide å få en runde der roboten fikk vist hva den dugde til. Hårføneren var nå satt i sving på samtlige kretskort, og i kampen mot "MMM. Málek" gikk det endelig. Roboten gjorde alt som den skulle og leverte et stort tempel med et lite tempel oppå. Bildet i figur 9.1 viser hvordan roboten vinner suverent over motstanderen med byggverket sitt. Med seier ble dette 72 poeng, og Lost Vikings hoppet rett opp på 4. plass igjen, inkludert de franske lagene. Total poengsum etter de 5 innledende rundene var 199 poeng, 2 poeng bak 3. plassen.



Figur 9.1: Roboten leverer to templer oppå hverandre og vinner over motstanderen "MMM. Málek"

9.2.4 Sluttspill

Sluttspillet i Eurobot består av 8-delsfinale, kvartfinale, semifinale, bronsefinale og finale. Alle disse kampene ble spilt rett etter hverandre, så det var ingen tid til testing eller verifisering av roboten mellom kampene. I sluttspillet ble det ikke gitt poeng for seier eller uavgjort slik det ble i de innledende rundene, men kun for konstruksjonene som ble bygget.

8-delsfinale

Siden laget vårt lå på 4. plass, møtte vi 13. plassen i 8-delsfinalen. Dette var laget "IA Bears" fra Romania. I denne kampen klarte motstanderen å bygge en konstruksjon før vår robot kom frem til poengområdet. Kameraet oppdaget konstruksjonen og roboten justerte derfor sin posisjon til den ene siden for ikke å rive ned det som allerede var bygget. På grunn av denne justeringen vurderte den kunstige intelligensen at det ikke var tid til å hente tverrliggeren, så den leverte kun to sylindere oppå tempelet. Dette holdt til seier, og Lost Vikings var videre til kvartfinalen. Med dette var ambisjonene som var satt på forhånd om å komme blant topp 10 allerede nådd, og det som var spennende nå, var hvor høyt opp vi kunne komme.

Kvartfinale

I kvartfinalen ble det et gjensyn med det Fransk-Canadiske laget "ETS Montreal - ENSAM Paris". Dette laget hadde vi tidligere spilt uavgjort mot, og det så ut til å bli en veldig spennende kamp. Under kjøringen så vi at løpehjulene gled mot underlaget, og roboten var tydelig ute av posisjon. Heldigvis klarte roboten å treffe alle sylinderene på bordet og kjøre helt inntil poengområdet. Når roboten kom bort til tverrliggerdispenseren derimot, var avviket fra odometrien såpass stort at roboten traff helt på siden av dispenseren og klarte

ikke å plukke opp tverrliggeren. Tross dette klarte roboten å vinne også denne kampen, og semifinale var et faktum.

Semifinale

I semifinalen ble vi satt opp mot det beste franske laget, nemlig "Microb Technologies", og vi visste vel egentlig på forhånd hvordan resultatet kom til å bli. Roboten kjørte fint og leverte det første tempelet mens motstanderen allerede hadde bygd to små templer oppå hverandre. Mens roboten fortsatt holdt på å levere, kjørte motstanderen tvers over vår side av spillebrettet og dyttet dermed klossene ut av posisjonene de sto i. Av denne grunn klarte ikke roboten å plukke opp flere sylindere, og det var umulig å få flere poeng siden en tverrligger må være støttet opp av sylindere for å være poenggivende. Det ble dermed det første tapet for laget, men det er jo greit å tape mot vinneren av konkurransen.

Bronsefinale

Tapet i semifinalen gjorde at vi havnet i bronsefinalen. Her var det igjen "Dynamo Rapperswil" som skulle til pers. Forventningene steg siden vi allerede hadde vunnet over dette laget, og det var tydelig at de hadde problemer før oppstart. I kampen derimot kom vi fort ned igjen på jorda da vi oppdaget at det igjen hadde skjedd en feil under oppstartsprosedyren. En mekanisk del på en av klypene hadde blitt forskjøvet under forrige kamp og stilt seg i en posisjon som gjorde at det var svært vanskelig å oppdage feilstillingen. Dette gjorde at viskeren som skal dytte sylinderene inn til klypene stoppet på midten, og roboten klarte ikke å plukke opp en eneste sylinder. Med 0 poeng i denne runden ble det tap, men likevel var det supert å konkludere med at Lost Vikings havnet på en fantastisk 4. plass.

9.2.5 Oppsummering

På tross av mange uhell gjennom de forskjellige rundene har roboten vært stabilt bra og klart å score poeng selv når ikke alle programmene kjører som de skal. Gjennom konkurransen har det vært en kamp bare for å holde roboten i live, og det er tydelig at vi hadde hatt godt av å fått testet totalsystemet i mer enn et par dager før konkurransen, slik det var tilfelle i år. I tillegg har det vært mye arbeid siden uforutsette problem som luftfuktighet har vært meget nærme ved å stoppe hele deltagelsen i konkurransen. Med stor innsats og arbeidsvilje har problemene blitt løst akkurat i tide til hver kamp, og har ført til den fantastiske prestasjonen det er å komme på 4. plass i denne konkurransen.

Kapittel 10

Konklusjon

Det har blitt utviklet en robot som er i stand til å utføre årets oppgave på en slik måte at svært høy poengsum er oppnåelig. Roboten er i stand til å detektere hvor de tilfeldig utlagte sylinderene befinner seg på meget kort tid, og benytte disse sammen med tverrliggere fra dispensere til å konstruere templer. Templene kan bli plassert på alle tilgjengelige poengområder, samt oppå ett annet tempel plassert på det høyeste poengområdet. For å unngå å rive ned motstanderens tempel sørger datasyn for å finne nærmeste ledige byggeplass, slik at tidsforbruket minskes, og høyest mulig poengsum gitt av situasjonen er oppnåelig.

For å unngå kollisjon er roboten i stand til å detektere og plassere motstanderroboten til enhver tid, med en kvart grads oppløsning og med et meget godt avstandsestimat. Dette sørger for at det tidligere utviklede antikollisjonssystemet kan foreta smarte valg i form av kollisjonsunngåelse og ruteplanlegging.

I likhet med deteksjon av motstander er det utviklede beaconbaserte deteksjons- og posisjoneringssystemet i stand til å detektere en dispenserbeacon, samt tre posisjoneringstårn plassert rundt konkurransebordet, med også her en kvart grads oppløsning. Dette sørger for at roboten er i stand til å finne posisjonen til den flyttbare dispenseren fylt med sylindere, samt muliggjør bruk av absolutt posisjonering om ny trianguleringalgoritme utvikles.

Det har blitt utviklet en helt ny kombinert motordrivermodul for høyre og venstre motor som benytter intern tilbakekobling, og sørger for lik rotasjonshastighet på begge motorene. Dette har medført en enorm forbedring i robotens evne til å kjøre langs rette linjer da sideavvik fanges opp momentant. Grunnet tidsmessige årsaker ble den mekaniske oppbygningen til den kombinerte motor og odometrimodulen sett på som tilfredstillende, noe som viste seg å være riktig. Allikevel bør den mekaniske oppbygningen endres, samt skilles, om roboten skal kunne kjøre raskere og således over lengre distanser.

Den kunstige intelligensen har både fått bedre funksjonalitet og blitt implementert på en måte som lettere kan forstås og vedlikeholdes for neste års studenter. Ved å bruke informasjon fra de forskjellige sensorene på roboten

velges den strategien som passer best for den gitte tilstanden til roboten og spillebrettet. I tillegg er hver utførte operasjon i en strategi utformet som en tilstand, slik at det er mulig å registrere om operasjonen tar for lang tid. Dersom dette er tilfelle har roboten flere mulige løsninger avhengig av hvilken tilstand som timet ut.

Håndtverket som er lagt ned i årets utviklede moduler er av et høyt nivå, og alle modulene som er utviklet er beregnet for at de skal kunne brukes videre uten å måtte endre på dem. Ved å bruke modulbasert design skal det være lett å integrere enhetene på en ny robot, og å bytte ut enheter som ikke er egnet for oppgaven som skal løses det gjeldende året.

Ulikt tidligere har det blitt opprettet to dokumentasjonspermer som meget grundig beskriver hvordan hardware og software til fungerende moduler fungerer, noe som sørger for at kommende års eurobotstudenter er i stand til å forstå, reprodusere, samt videreutvikle det som har blitt laget.

Arbeidet har vært svært tidkrevende og har i tillegg til konstruksjon av konkurranserobot innebært administrering av studenter fra Eksperter i Team, samt mekanikerlærling på instituttets verksted. Nytt konkurransebord har blitt utviklet og vil kunne benyttes i flere år fremover. I tillegg har det blitt deltatt på flere arrangementer for å promotere eurobot, samt inspirere unge til å velge realfag.

I forkant av konkurransen var roboten meget stabil, og oppnådde svært høye poengsummer under tester foretatt. Dette var dog ikke tilfelle under oppstarten i Frankrike, men etter at problemet med kortslutninger grunnet høy luftfuktighet ble oppdaget og løst, jobbet roboten seg frem med en imponerende stabilitet og robusthet. Dette underbygges med at den utviklede roboten, som eneste robot med unntak av vinneren, ikke tapte en eneste kamp i innledende runder, samt seiret både i åttendelsfinale og kvartfinale.

Det kan dermed til slutt oppsummeres med at det har blitt utviklet flere velfungerende moduler samt foretatt store og viktige softwaremessige endringer vedrørende strategi-implementering, som sørget for at den utviklede roboten til slutt endte på en meget sterk 4.plass.

Kapittel 11

Videre arbeid

Siden oppgavene forandrer seg fra år til år er det stor sannsynlighet for at det blir andre behov fra både programvaren og mekanikken enn de behovene man har hatt dette året. Det har også blitt oppdaget svakheter på roboten underveis som ikke har blitt forbedret på grunn av tidsmangel. Dette kapittelet tar for seg de forbedringspotensialene som har blitt observert dette året, samt teknikker og moduler vi mener bør videreføres.

11.1 Programvare

11.1.1 Generelt vedlikehold

Noe av grunnen til at det har vært vanskelig å sette seg inn i koden som ble arvet, var at det var veldig mye kode innimellom den aktive koden som viste seg å aldri bli brukt. Det har blitt lagt ned mye arbeid i å finne og fjerne slik kode som bare gjør programmereren forvirret og gjør koden mindre vedlikeholdbar. Det oppfordres derfor for de som skal fortsette å jobbe med systemet, å fjerne kode som er oppgavespesifikk og som nødvendigvis ikke kan gjenbrukes. Dette istedet for å kommentere ut deler, eller bare la funksjonene ligge ubrukt slik som gjort tidligere.

11.1.2 Systemets struktur

Selve strukturen på programmene har vist seg å fungere godt. Alle meldingene som må sendes mellom prosessene skaper et visst overhead, men har vist seg å være svært nyttig for å skille mellom de ulike delene av systemet. Strukturen gjør det enkelt for flere personer å jobbe sammen med hver sin del, og definere klare grensesnitt mellom de ulike delene. Slik det har vært i år, har meldingssystemet gitt en del ekstraarbeid og til tider virket tungvint, siden det har kun vært en person som har jobbet med hele systemet. Det anbefales likevel å beholde strukturen for å legge til rette for et mulig samarbeidsprosjekt for senere

år og for at hver del kan utvikles videre uten å måtte vite eksakt virkemåte til andre deler av systemet.

11.1.3 Navigasjonssystemet(Navsys)

Navsys har blitt nedprioritert dette året, på grunn av at det har vært større fokus på utvikling av hardware, og dermed kun har vært en person som i hovedsak har jobbet med programmeringen. Her trengs det mye opprydning, delvis fordi den strukturen som ble laget i 2007 ikke egnet seg så godt for en robot på hjul, men også fordi de endringene som ble gjort i 2008 ikke tok bort disse strukturelle svakhetene, men bygde endret funksjonalitet oppå dette systemet.

Navigasjonssystemet fra 2007 bygger svært mye på faget "Navigasjon og fartøystyring" som flere studenter på kybernetikk tar i 4. årstrinn. I dette faget lærer man mange nyttige ligninger og modeller for navigasjon. Det som er problemet her er at faget har veldig sterkt fokus på sjøfart, og alle modellene man lærer om bærer preg av dette. Det ser man også igjen i koden som er skrevet i navsys. Det er satt opp et stort og unødvendig komplekst system for bevegelsesligning og regulator for roboten. Siden dynamikken til roboten er svært enkel: kun bevegelse i lengderetning og rotasjon om den vertikale aksen er mulig, ingen sidelengs bevegelse, kan disse modellene kuttes ned til en brøkdel av den nåværende kompleksiteten.

Regulatoren som ble laget i 2007 er kalt LOSGuidance. LOS står for Line Of Sight, og er en algoritme som brukes på skip for å holde seg best mulig på en gitt kurs mellom to punkter på tross av ytre påvirkning som vil dytte skipet sidelengs [5]. Dette har ingen relevans for en robot som for det første ikke har ytre påvirkninger som flytter roboten sidelengs, og som heller ikke trenger å huske hvor den kom fra, men kun kjøre i rett linje mot målet sitt. Ved gjennomgang av regulatoralgoritmen i navsys har det blitt observert at det i stor grad henger igjen tankegang fra dette prinsippet, men at det har blitt gjort flere snarveier for å fikse problemene som oppstår på grunn av feil struktur.

Det er i tillegg definert to forskjellige regulatorer som er kaskadekoblet uten noe fysisk system imellom. Dette kan med fordel forenkles til en regulator for å gjøre det enklere å tune koeffesienter i regulatoren for å få ønsket oppførsel fra roboten.

Det anbefales derfor å lage regulatoren i navsys på nytt fra bunnen av. Her er det mye kode som ikke er gjennomgått dette året, og det er stor sannsynlighet for at systemet kan gjøres mindre og mer forståelig ved å ta en kritisk gjennomgang av eksisterende kode. Programvaredokumentasjonen [25] kan brukes som referanse for hvordan systemet skal kommunisere med andre moduler ved en eventuell omskrivning.

11.1.4 Den kunstige intelligensen(Plansys)

Prinsippet for den kunstige intelligensen er nå brukt de siste 3 årene, og har fungert meget bra. Det er også gjort et grundig arbeid med dette systemet for å gjøre koden forståelig og effektiv. Det menes derfor at strukturen bør beholdes, men at alle strategiene må skrives på nytt for den nye oppgaven.

11.1.5 Motordriverkortet

Det har vist seg at kvadraturtellerene som er brukt til å lese av både motorhastighet og løpehjulene til odometrimodulen ikke fungerer som beskrevet i databladet. Ved resetting av innholdet i telleren klarer ikke kvadraturtelleren å få med seg eventuell samtidig bevegelse av encoderskiven. Dette gjør at den avleste verdien ikke blir korrekt, og reduserer nøyaktigheten til posisjonereingssystemet. Dette kan fikses ved å ikke resette kvadraturtellerene og lese av den absolutte verdien for rotasjonen istedet. Siden kvadraturtellerene kan lagre verdier opp til 32 bit er det ikke problemer knyttet til overflyt ved dette designet, siden det er mulig med opp til 1 million rotasjoner i hver retning. Denne endringen vil sannsynligvis øke nøyaktigheten til det relative posisjoneringssystemet basert på odometri betraktelig.

11.1.6 Trianguleringsalgoritme

Det har blitt utviklet et beaconbasert system beskrevet i kapittel 3 som er i stand til å detektere vinklene til posisjoneringstårn plassert på kjente plasser rundt konkurransebordet med svært høy oppløsning. Dette systemet sammen med tidligere utviklet trianguleringsalgoritme, [14], skulle sørge for absolutt posisjonering for å rette eventuelle feil som hadde oppstått ved det relative posisjoneringssystemet basert på odometri. Trianguleringsalgoritmen viste seg dog å ikke fungere tilfredstillende på hele spilleflaten, noe som det ikke fantes tid til å se nærmere på. Det er derfor nødvendig å utvikle en ny trianguleringsalgoritme for å ta i bruk absolutt posisjonering. Ved utvikling av ny trianguleringsalgoritme kan det også være hensiktsmessig å vite omtrentlig avstand til posisjoneringstårnene, ikke bare nøyaktige vinkler. Ved svært enkel omprogrammering av posisjoneringstårnene kan disse i likhet med motstanderbeacon, oversende tiden det tok mellom to lasertreff, en tid som representerer avstanden. Grunnet noe varierende rotasjonshastighet på det roterende tårnet representerer denne tiden på nåværende tidspunkt kun en omtrentlig avstand, men kan gjøres mer nøyaktig om hastighetsregulator på det roterende tårn implementeres.

11.2 Mekanikk

11.2.1 Odometri

Posisjoneringssystemet basert på odometri har gjennomgått store forbedringer i forbindelse med prosjektoppgave høst 2008 [9], og gjennom arbeid gjort med motordrivermodulen i denne oppgaven, beskrevet i kapittel 4. Det som gjenstår er fornyelse av den mekaniske delen som har vist seg å ha svakheter, og som bidrar til at estimert posisjonsavvik øker unødvendig mye over tid. I dagens system er det benyttet flere tannhjul, slik at dødgang eksisterer, samt at det er vanskelig å oppnå tilstrekkelig friksjon mot underlaget. Ved bevegelse av roboten rolig frem og tilbake, er det observert at hjulene ikke roterer tilbake til den opprinnelige startposisjonen, selv om gli ikke har blitt visuelt observert under selve bevegelsen.

Det bør derfor utvikles et nytt mekanisk system med løpehjul som har større kontaktflate og press mot underlaget, samt at rotasjonsavlesning bør gjøres direkte på løpehjulakslingene. Som en siste faktor er det observert flere svært gode roboter som har løpehjulene på utsiden av drivhjulene istedet for på innsiden som på nåværende system. Med dette vil avstanden mellom løpehjul bli forlenget og sørge for bedre oppløsning på vinkelestimeringen, en parameter som ved feil sørger for stort posisjoneringsavvik over tid.

11.2.2 Motor og gir

Motor og gir fungerer meget godt, men det er satt spørsmål ved om størrelsen på modulen er nødt til å være så stor. Modulen er idag også kombinert med den mekaniske delen til odometrisystemet, og dette gjør systemet svært lite modulært. Da utseende til robotene endrer seg betraktlig hvert år grunnet forskjellige oppgaver, er det også ytret et ønske om å kunne velge om motorene skal plasseres horisontalt eller vertikalt i girmodulen. I tillegg har motorene ulik intern friksjon, beskrevet i kapittel 4.1.2, som medfører større kompleksitet vedrørende kjøring langs en rett linje. På bakgrunn av dette bør det vurderes om andre typer motorer er mer hensiktsmessig i eurobotsammenheng.

11.3 Elektronikk

Det har blitt benyttet flussmiddel på alle utviklede kretskort for å skape gode og solide konkave loddinger. Å bruke flussmiddel anbefales sterkt, men det har vist seg at det eksisterer forskjellige varianter og at feil type har blitt benyttet. Dette medfører at selv etter vasking av et kretskort eksisterer det fortsatt ett tynt lag flussmiddel på overflaten, som begynner å lede strøm ved store luftfuktigheter. Dette ble oppdaget under konkurransen og løst ved å varme opp samtlige kretskort med varmepistol eller hårtørker. Kretskortetene bør derfor vaskes skikkelig nok en gang, hvorpå en funksjonstest gjennomføres med på-

følgende beskyttelseslakkering av kortene. Eventuelt kan kortene produseres på nytt med annet flussmiddel, da disse er godt dokumentert i [10]. Det sistnevnte vil også bidra til et ekstra sett med kretskort, noe som systemet med kretskortramme legger opp til da utskiftning av kretskort kan gjøres meget hurtig.

11.4 Dokumentasjon

Som nevnt i kapittel 8.3 er det lagt vekt på å dokumentere fungerende moduler og software-systemet i sin helhet, for å gjøre det lettere for neste års studenter å overta og jobbe videre med prosjektet. Det anbefales å videreføre denne tradisjonen og følge samme retningslinjer som er lagt for permene. Dette vil forhåpentligvis legge til rette for god oversikt over moduler som fungerer, og at det er mulig å se tilbake på tidligere års programvaredokumentasjon for oppsnapping av gode ideer og designvalg.

Bibliografi

[1] L. Belady and M. Lehman. *Program Evolution: Processes of Software Change*. Academic Press, London, UK, 1985.

[2] J. Borenstein, H.R. Everett, L. Feng, and D. Wehe. Mobile Robot Positioning - Sensors and Techniques. *Journal of Robotic Systems, vol 14 No.4,pp.231-249*, 1996.

[3] Fagrapport EiT. Eurobot 2009. Technical report, Institutt for Teknisk Kybernetikk, NTNU, 2009.

[4] R.W. Farebrother. *Linear least squares computations*. Marcel Dekker Inc, 1988.

[5] Tor Inge Fossen. *Marine Control Systems: Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*. Marine Cybernetics, Norway, 2002.

[6] S.S. Ghidary, T. Tani, T. Takamori, and M. Hattori. A new Home Positioning System (HRPS) using IR switched multi ultrasonic sensors. Technical report, Dept. og computer and systems engineering, Fac. of Engineering, Kobe University, Japan, 2007.

[7] Gunnar Kjempehol. Eurobot 2007. Master's thesis, Institutt for Teknisk Kybernetikk, NTNU, 2007.

[8] Kjemphol and Knausgård. Prosjektoppgave: Eurobot 2007. Technical report, Institutt for Teknisk Kybernetikk, NTNU, 2006.

[9] Kjølseth and Wergeland. Prosjektoppgave: Eurobot 2009. Technical report, Institutt for Teknisk Kybernetikk, NTNU, 2008.

[10] Christian Kjølseth. Dokumentasjonsperm, hardware. Technical report, Institutt for Teknisk Kybernetikk, NTNU, 2009.

[11] Donald P. Massa. Choosing an Ultrasonic Sensor for Proximity or Distance Measurment. *Sensors*, 1999.

[12] Frank Massa. Ultrasonic Transducers for Use in Air. *IEEE, vol 53, NO 10*, 1965.

[13] Mørkrid and Øien. Prosjektoppgave: Eurobot 2008. Technical report, Institutt for Teknisk Kybernetikk, NTNU, 2007.

[14] Mørkrid and Øien. Eurobot 2008. Master's thesis, Institutt for Teknisk Kybernetikk, NTNU, 2008.

[15] James W. Nilsson and Susan A. Riedel. *Electric Circuits*. Pearson, 2005.

[16] Nobelprize.org. What is a laser? `http://nobelprize.org/educational_games/physics/laser/facts/index.html`, 2009.

[17] Tatsuyuki Ochi. A positioning system for mobile robots using symmetrical rotating laser beams. In *Advanced Robotics*, pages 217–222. VSP and Robotics Society of Japan, 1990.

[18] Eurobot Organisation. Eurobot.org. `http://www.eurobot.org`.

[19] Walter Savitch. *Absolute C++*. Pearson Education, 3rd edition, 2008.

[20] Planète Science. Eurobot 2009 rules. `http://www.planete-sciences.org/robot/data/file/coupe/2009/E2009_rules_and_drawings-EN-final-v1.pdf`.

[21] Ian Sommerville. *Software Engineering*, chapter 21. Pearson Education, 8th edition, 2007.

[22] Ingrid Spilde. Hva er laser? `http://www.forskning.no/artikler/2002/november/1036663547.37`, 2002.

[23] Wikipedia. Object oriented programming. `http://en.wikipedia.org/wiki/Object_oriented`.

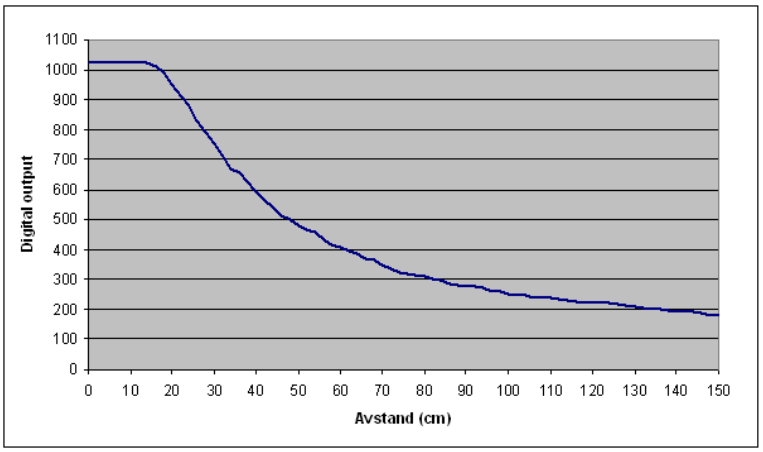
[24] Yong Ju Yang, Jae Woo Shin, Seung Jin Jang, Hyun Sook Lee, and Young Ro Yoon. Resarch of body movement during sleep with an infrared triangulation distance sensor, wavelets and neuro-fuzzy reasoning. Technical report, Department of Biomedical Engineering, Department of Oriental Biomedical Engineering, Republic of Korea, 2006.

[25] Øystein Wergeland. Dokumentasjon av programvare - Eurobot 2009. Technical report, Institutt for Teknisk Kybernetikk, NTNU, 2009.

Tillegg A

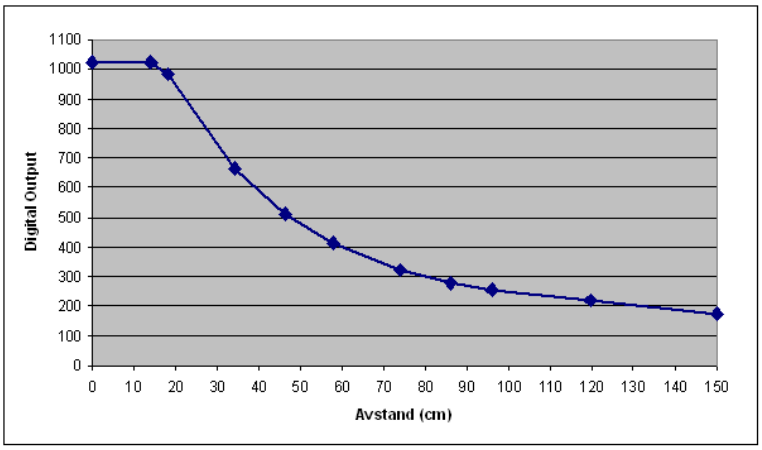
Kalibrering av infrarød avstandssensor

Signalkurve i etterkant av AD konverterer.



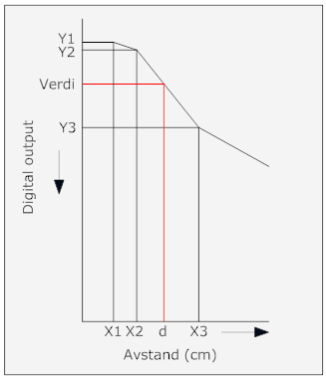
Figur A.1: Avlest avstandssignal i etterkant av AD konverterer

Signalkurve etter stykkevis linearisering



Figur A.2: Stykkevis linearisert avstandssignal i etterkant av AD konverterer

Den stykkevise lineariseringen er delt opp ved Y_i og X_i, som vist i figur A.3, og med verdier gitt av tabell A.1. Utregning av distansen d blir gjort ved å finne nærmeste $Y_i \leq \mathit{Verdi}$ og $Y_{i-1} \geq \mathit{Verdi}$ med tilhørende X_i og X_{i-1}, og benytte ligning A.1.



Figur A.3: Oppbyggning av stykkevis linearisert tilnærming

X_i	14	18,4	34,4	46,5	57,8	73,8	85,9	96,1	119,5	279
Y_i	1022	985	665	510	415	325	280	255	220	0

Tabell A.1: Knekkpunkter $X_1, Y_1 \rightarrow X_{10}, Y_{10}$

$$\mathrm{d} = \frac{Y_{i-1} - \mathit{Verdi}}{\frac{Y_{i-1}-Y_i}{X_i - X_{i-1}}} + X_{i-1} \quad \mathrm{cm} \tag{A.1}$$

Tillegg B

CD-ROM

Vedlagte CD inneholder all kildekode til roboten, dokumentasjon av hardware og software, kildekode for denne rapporten og diverse dokumenter som kan være nyttig for kommende eurobot-studenter.

Strukturen til CD'en er som gitt under

- 2009
  - Hardware
    - Roterende tårn
    - Motstanderbeacon
    - Posisjonsbeacons
    - Dispenserbeacon
    - Kommunikasjonsnode
    - Motordrivermodul
    - Brytermodul
    - Hardwareperm
  - Software
    - Kode
    - Dokumentasjon
  - Diverse
    - Filmer
    - Bilder
    - Logoer
    - Plakat
  - Rapport