AALBORG UNIVERSITET

Institut for Elektroniske Systemer

Titel: Robot Race

Tema: Apparat og system konstruktion

Projektperiode: 4. september - 20. december 2001

Storgruppe: E5 2001

Gruppe: 506

Gruppemedlemmer:

Mads Fogh
Jesper Holst
Jakob Kirkegaard
Morten Tofte Koch
Peder Andersen Lund
Morten Mikkelsen
Esben Juul Sørensen

Vejleder:
Henning Nielsen

Oplagstal: 10

Rapport sideantal: 90

Appendiks sideantal: 32

Total sideantal: 122

Afsluttet: 20. december 2001

Synopsis:

I dette projekt er der lavet et autonomt videostyret proceskontrolsystem, som ved hjælp af en LEGO-robot kan indsamle plastickopper på en afgrænset bane - i konkurrence med en tilsvarende robot.

Systemet er opdelt i tre stadier:

Billedserver. En Silicon Graphics computer finder gennem et kamera kopper og robotters positioner, som videresendes gennem et UDP/IP netværk.

Klient. En Intel baseret computer beregner ud fra modtagne billeddata hvilke kopper, det kan svare sig at køre efter, under hensyntagen til korteste rute samt konkurrentens bevægelser. Disse data videresendes til en radioserver via TCP/IP.

Robot. Et opbygget microcontrollersystem håndterer de modtagne data og styrer dc-motorer til fremdrift, samt gribearm til indsamling af kopper.

AALBORG UNIVERSITY

Institute of Electronic Systems

Title: Robot Race

Theme: Device and system construction

Time period: 4th of September - 20th of December 2001

Term: E5 2001

Group: 506

Members:

Mads Fogh
Jesper Holst
Jakob Kirkegaard
Morten Tofte Koch
Peder Andersen Lund
Morten Mikkelsen
Esben Juul Sørensen

Instructor:

Henning Nielsen

Number printed: 10

Number of pages: 90

Appendix pages: 32

Total number of pages: 122

Ended: 20th of December 2001

Abstract:

A complete autonomous video controlled LEGO robot regulation system was made during this project. The purpose was to collect plastic cups from within a limited area - in competition with a similar robot.

The system is divided in three stages:

Image server. A Silicon Graphics computer determines cup and robot positions through a video camera and sends these data over a UDP/IP network.

Client. An Intel based computer calculates which cups are priority targets, based on distance and movement of the other robot. Control data are sent to a radio communication server via a TCP/IP network.

Robot. A constructed microcontroller system handles the received data and controls the propulsion dc-motors and a cup grabbing mechanism.

1 Forord

Dette projekt er udarbejdet af gruppe 506 ved Institut for Elektroniske Systemer på Aalborg Universitet. Projektet er udarbejdet i P5-perioden, der strækker sig fra den 4. september til den 20. december 2001.

Formålet med P5-projektet er at få den tilførte tekniske viden fra PE-kurserne indført i den problemorienterede og projektorganiserede indlæringsform gennemført i grupper. Mere specifikt er formålet med projektet:

- at kunne foretage syntese af et samlet system bestående af en datamat og et fysisk system, med en interaktion via transducere og interfaces.
- at kunne anvende teorier og metoder til analyse og modellering af komplekse fysiske komponenter og systemer.
- at kunne anvende teorier og metoder til analyse, design og implementation af et datamatsystem, herunder inddragelse af sandtidsproblematikker.
- at kunne anvende metoder til systemkonstruktion, herunder valg af en hensigtsmæssig balance mellem hardware og software.

Målgruppen for dette projekt er studerende og andre interesserede med et teknisk niveau svarende til gruppens eget.

I rapporten er alle litteraturhenvisninger udført i henhold til Harvardmetoden, dvs. forfatter og årstal står i kantet parentes - eks. [Sedra & Smith, 1998]. Placering af henvisningen afgør, hvad der specifikt henvises til. Står kildehenvisningen før et punktum, refereres der til den foregående sætning - står henvisningen efter et punktum, refereres til hele det foregående afsnit.

Figurer og ligninger er nummereret efter kapitel og et fortløbende nummer. Eksempelvis kan figur 1.10 findes i kapitel 1, som den 10. figur.

Desuden benyttes punktum som decimalseparator, da dette benyttes i de fleste simulerings- og matematikprogrammer.

I rapporten er der lagt vægt på analyse og metodik af systemet, fremfor implementering. Dette gør, at det kun er de vigtige dele af implementeringen, der er fremhævet. Derfor indeholder rapporten ingen specifik kildekode, men visualiseres vha. pseudokode og flowchart

diagrammer. For nærmere inspektion af kildekoden, henvises der til den vedlagte cd-rom, der indeholder alle anvendte datablade samt rapport og kildekode.

Aalborg d. 20. december 2001

Mads Fogh	Jesper Holst	Jakob Kirkegaard
Morten Tofte Koch	Peder Andersen Lund	Morten Mikkelsen
	Esben Juul Sørensen	

Indhold

2 Indledning

Formålet med dette kapitel er kort at introducere problemstillingen, som behandles i projektet, for derefter at analysere det valgte problem. Dette gøres med henblik på at opstille en kravspecifikation, som skal danne grundlag for videre design og implementation af systemet.

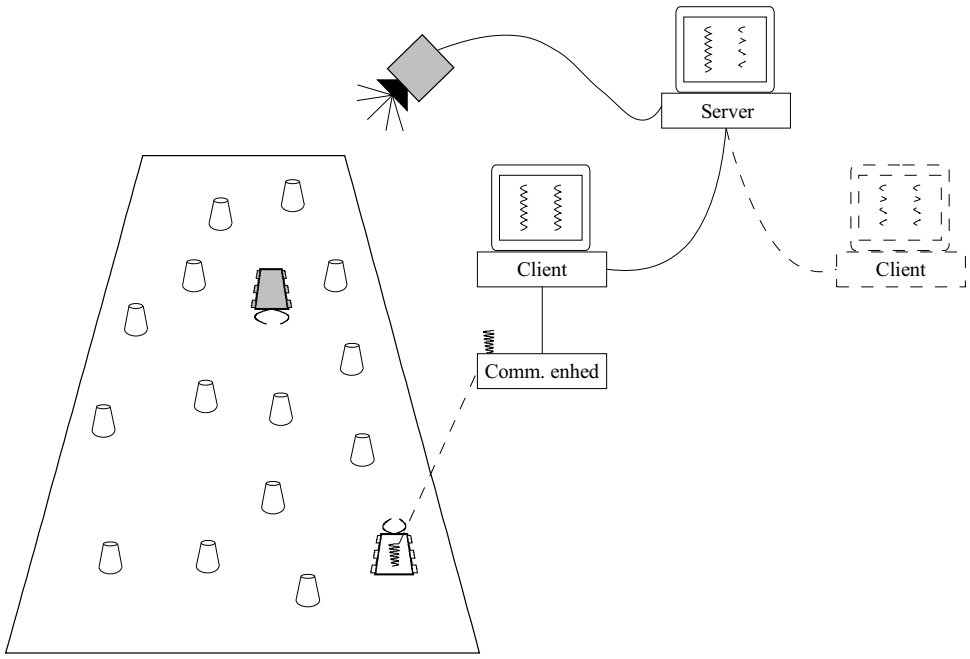
En robot er en maskine designet til at udføre en eller flere foruddefinerede opgaver gentagne gange med givne krav til hastighed og præcision. Mere præcist er en robot en programmérbar maskine, som i større eller mindre grad kan imitere et intelligent væsens handlinger. For at kunne opfattes som en robot, skal en maskine være i stand til at opsamle informationer fra det omgivende miljø, samt foretage handlinger i dette fysiske miljø. Hvis robotten på egen hånd uden menneskelig interaktion kan afgøre sine handlinger ud fra de informationer, som opsamles via sensorerne, siges robotten at være autonom.

Et meget centralt problem i forbindelse med robotter er dels at opnå kendskab til deres position og orientering i det miljø, som de befinder sig i, dels at bestemme position og type af andre objekter i deres omverden. Med henblik på at opnå denne viden om position, retning og andre egenskaber ved miljøet, kan robotter opsamle informationer fra omverdenen via en eller flere sensorer, der i visse tilfælde forsøger at efterligne det menneskelige sanseapparat. Lyssensorer kan fx benyttes til simpel kontrastnavigation eller som ultralyd til afstandsbedømmelse. Til mere komplekse målinger på omverdenen benyttes kameraer eller en CCD chip til opsamling af billeddata i miljøet. Ud fra disse opsamlede billeddata, kan robotten via en billedanalyse finde bestemte mønstre i billedet samt beregne afstande og positioner. [Borenstein, 1996]

Dette projekt beskæftiger sig med analyse, design og konstruktion af en robot. Robotten skal foretage en række handlinger på baggrund af de sensorer, som sidder på selve robotten samt informationer opsamlet vha. et kamera placeret over det område, som robotten skal operere i.

Mere specifikt ønskes at opbygge et system, hvor et bagvedliggende computer system skal opsamle billeddata, identificere objekter i området og på baggrund af disse oplysninger beregne strategien for robottens bevægelsesmønster i området. Ud fra disse beregninger skal robottens handlinger bestemmes, og disse transmitteres til selve robotten via en trådløs kommunikationskanal.

Spillekoncept regler. Idet to grupper beskæftiger sig med samme projekt, er der lagt op til en konkurrence, hvor målet er at identificere og indsamle fleste mulige objekter inden for en given tidsramme.



Figur 2.1: Oversigt over det samlede system.

På forhånd er der givet en Silicon Graphics maskine (SGI) med framegrabber og netkort. Til framegrabberen er tilsluttet et videokamera ved hjælp af hvilket, systemet kan grabbe 25 [fps]. Denne SGI maskine skal benyttes af begge grupper. Derudover skal hver gruppe tilslutte en klient til SGI maskinen, som skal stå for styring og kommunikation af hver deres respektive robot samt fungere som interface for brugeren af systemet.
Da der er tale om en konkurrence, er der på forhånd udfærdiget nogle regler mellem de to grupper.

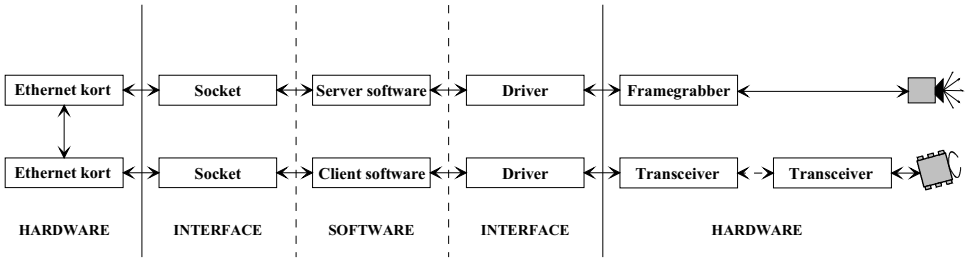
Regelsæt for Robot Race 5. semester 2001:

- Et ulige antal kopper (9).
- Kopperne skal være ligeligt fordelt på banen.
- Indsamlede kopper skal afleveres på basen.
- Kopper på basen må ikke indsamles af modstanderen.
- Baserne har samme størrelse som et A3 papir.
- Alle kopper skal være indsamlet, før spillet er slut.
- En runde tager maks. 10 min. Vinderen er holdet med flest indsamlede kopper.
- Vinderen kåres bedst af 3 runder.
- Der skiftes ikke base ved nye runder.
- Det skal undgås, at robotterne kolliderer med hinanden.

Ovenstående regelsæt er blevet udarbejdet af grupperne 506 og 508 i forbindelse med projektet på 5. semester.
Begge grupper står inde for reglerne, der indirekte skal indgå som parametre i udførelsen af projektet.

Kravspecifikation 3

I dette kapitel defineres de enkelte delsystemer, og der opstilles specifikke krav til hvert enkelt delsystem. Indledningsvis specificeres det samlede systems interne og eksterne grænseflader, hvorved en mere grundlæggende analyse af systemet muliggøres. Til sidst redegøres for de valgte metoder, der ligger til grund for løsningsmodellen i de enkelte delsystemer.



Figur 3.1: Systemets opdeling i blokke, hvor interne og eksterne grænseflader er vist.

På figur 3.1 er systemets snitflader beskrevet. Ud fra disse opstilles der følgende præcise krav til de enkelte dele af systemet samt krav til grænsefladerne imellem delsystemerne.

3.1 Server

Serveren skal være i stand til at opsamle billeddata via kameraet og framegrabberen. Serveren udgøres af en Silicon Graphics maskine med tilhørende IRIX operativsystem. Det antages, at opsætningen af framegrabberen er foretaget, samt der forefindes tilgængelig kode til initialisering af hardwaren og til opsamling af billeddata. Framegrabber delen af serveren betragtes således som en blackbox, der fra vores synsvinkel blot er i stand til at levere et givent antal billeder inden for en bestemt tidsramme, hvorfor der kun stilles krav til selve billedbehandlingen. Opgaven bliver da at analysere, designe og implementere software, som er i stand til at behandle de billeddata, der opsamles af systemet. Heraf uddrages relevante

data til videre brug. Disse data skal transmitteres videre til klienten, som foretager den egentlige regulering af robotten.
Mere specifikt er følgende givet på forhånd:

- Et sort/hvid videokamera, der overvåger robottens bevægelsesområde fra en kendt position og vinkel.
- SGI maskine med tilhørende framegrabber samt hardwaredriver, der muliggør grabbing af billeddata op til 25 [fps].
- Billedopløsning på 768×576 [pixels] med farvedybden 32 bit RGB.

3.1.1 Billedbehandling

Herunder findes kravene til billedbehandlingssoftwaren:

- Softwaren på serveren skal være i stand til at hente minimum 10 [fps], således robottens bevægelse løbende bestemmes ud fra opdaterede billeddata.
- Selve billedbehandlingen skal kunne nås inden for en tidsramme på 100 [ms].
- Billedbehandlingssoftwaren skal tage højde for, at kameraet er placeret i en på forhånd kendt vinkel i forhold til banen. Dette skal modelleres vha. en pinhole model, mhb. på transformation fra billedkoordinater til gulvkoordinater.
- Ud fra billederne skal der efter støj filterering identificeres 3 typer af objekter:

  Egen robot (massemidtpunkt og retning).

  Modstanderrobot (massemidtpunkt og retning).

  Kopper (massemidtpunkt)

3.1.2 Klient/server kommunikation

Til at varetage overførsel af billeddata i systemet, skal der benyttes en netværksprotokol. Netværket består i dette projekt af kommunikationen mellem de to maskiner, der hver varetager på forhånd bestemte funktioner. Netværket ønskes realiseret med udgangspunkt i en klient/server struktur i form af en dedikeret billedbehandlingsserver og en klient til styring af GUI og den autonome robot. Der sættes på forhånd som krav, at serveren sender billeddata til klienten, hver gang et billede er færdigbehandlet. Klienten skal således lytte efter indkommende data.

3.2 Hardware

Dette afsnit beskriver de krav, der skal gælde, for at data fra en klientcomputer kan styre en specifik hardware - robotten. Robotten skal opbygges af LEGO og udstyres med en microcontroller, som skal foretage styring af aktuatorer, kommunikationshardware samt opsamle data fra sensorerne på selve robotten.

Krav til den fysiske robot. Da robotten skal kunne manøvrere rundt inden for et begrænset areal (kameraets synsfelt), er det vigtigt med gode og hurtigt reagerende manøvreegenskaber. Robotten skal også kunne opsamle plastkopper samt sætte dem andetsteds. Tillige ønskes der logik, som skal kunne klare små opgaver autonomt, som fx at regulere hastighed, styre en gribemekanisme etc. Der skal være gode trådløse kommunikationsegenskaber i form af rækkevidde (min. 10 [m]), hastighed og fejlrate, således klientcomputeren ikke mister kontrollen over robotten. Endvidere skal det være muligt at kunne foretage analoge målinger på fx. spænding, afstand el.l.
De opstillede krav opsummeres således:

- Stor manøvredygtighed.
- God traktion og mulighed for justering af fart.
- Gribeaggregat til opsamling af plastkopper.
- Gode trådløse kommunikationsegenskaber.
- Mulighed for implementation af simple styringsprocedurer.
- Mulighed for måling af analoge signaler.

Krav til kommunikationsdelen. Kravene omfatter såvel klientcomputeren som robotten, imellem hvilke der skal være en fælles kommunikationsprotokol. Ligeledes skal det være muligt at sende data mellem klient og robot, uden afbrydelser, med høj hastighed og med en god immunitet over for andre datasignaler - specielt styredata til den anden robot. Kommunikationen skal foregå trådløst og skal ikke afbrydes, når robotten er i yderområderne af det afgrænsede areal. Ydermere skal det være muligt at få statusmeldinger tilbage fra robotten. Hovedkravene er kort sammenfattet således:

- Fælles protokol med udgangspunkt i en standardiseret protokol.
- Trådløs kommunikation med god rækkevidde i forhold til de fysiske rammer.
- Fejldetektion (kontrol bit/byte).
- Bidirektional kommunikation.

3.3 Klient

Klienten skal stå for beregning af robottens strategi ud fra de billeddata, som den modtager fra serveren. Derudover skal klienten indeholde en grafisk brugerflade. I forbindelse med klienten er der ikke på forhånd opstillet rammer for, hvordan systemet skal implementeres. Dvs. der er mulighed for at vælge arkitektur og operativsystem efter egne ønsker og krav. Dog skal klienten være i stand til at benytte de protokoller, som serveren stiller til rådighed, således kommunikationen mellem systemerne muliggøres. Derfor skal der i klienten tages stilling til følgende angående kommunikationen mellem delsystemerne:

- Valg af hardware tilsluttet klienten til kommunikation med robotten.
- Valg af arkitektur og operativsystem, der skal kunne kommunikere med serveren.

3.3.1 GUI

Klienten skal udstyres med et Grafisk User Interface (GUI) mhb. på at give brugeren en grafisk præsentation af systemets muligheder samt dets indstillinger, uden at man på forhånd har en dybere indsigt i systemets parametre. Kravene til GUI er følgende:

- Visning af spillefeltet med kopper og bevægelsen af robotterne ud fra koordinatberegning.
- Visning af klientcomputerens input/output data i form af informationer, som kommer fra serveren og afsendte kommandoer til robotten.

3.3.2 Strategi

Den primære strategiberegning skal foregå i klienten, idet den på baggrund af processerede billeddata fra serveren, skal udregne robottens kørselsstrategi. Taktikken i klient softwaren baseres på prioritering af robottens destinationer, samt ruten til den valgte destination. Dette skal ske i følgende rækkefølge:

- Klienten får informationer om objekterne fra serveren, herunder informationer om type og position.
- Robotternes bevægelse analyseres, og resultatet gemmes. Klienten skal være i stand til at bestemme robotternes retning, hastighed og position ud fra let identificérbare og forskellige mønstre på robotternes skjold.
- Data for objekterne analyseres, og destinationerne prioriteres. Herunder skal klienten være i stand til at bestemme robottens næste position samt modstanderens næste destination og ruten til denne. Endvidere skal der tages højde for, om robotterne er ved at hente eller bringe objekter.

- Den højst prioriterede destination vælges, og ruten beregnes. Den valgte destination kan prioriteres, selvom modstanderen antages at have valgt samme destination, men skal nedprioriteres, hvis modstanderen antages at nå destinationen først.
- Ruten fortolkes, hvorefter de ønskede kommandoer sendes til robotten. Klienten skal til enhver tid kunne ændre robottens bevægelse.

3.4 Metoder

Til løsning af de forskellige delsystemer vil følgende metoder blive benyttet:

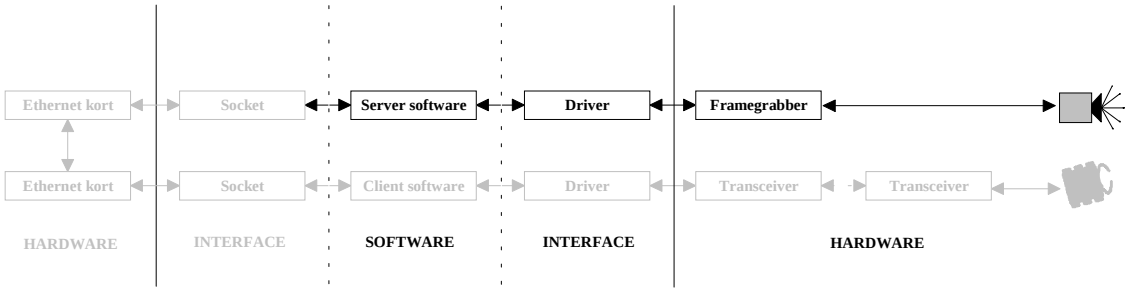
- **Server.** Da præcisionen af robotten afhænger af informationerne fra serveren, er det vigtigt, at billedbehandlingensprogrammet eksekveres hurtigst muligt. Programmeringssproget C er det foretrukne valg, idet C kode er hardwarenært, med hurtig eksekveringstid til følge. Programmet i serveren skal helst være enkelt, så der ses bort fra en decideret metode til løsning af delsystemet.
- **Hardware.** I robotten er det vigtigt at timingen af microprocessoren er præcis, idet denne skal styre tidskritiske komponenter. Der vælges at programmere microprocessoren i assemblerkode.
- **Klient.** Løsning af klientens strategi algoritmer, er programmingsmæssigt kompleks, hvorfor der er valgt at bruge objektorienteret systemudvikling. UML metoden vælges som dokumenteringssprog, og selve programmeringen foregår i det objektorienterede programmeringssprog C++. GUIen bliver programmeret vha. QT[1].

[1] Grafiske C++ include moduler, leveret af firmaet Trolltech

Billedserver

4

Dette afsnit har til formål at beskrive serveren, som varetager billedbehandlingen. Klienten skal modtage så præcise informationer om de fysiske forhold som muligt mht. identifikation af objekter og deres positioner, sådan at robotten kan opnå den hurtigste rute og den rigtige kurs til en nærstående kop. Kapitlet indeholder nogle generelle informationer om billedanalyse, de egenskaber billedbehandlingsprogrammet skal indeholde, samt hvordan det implementeres på serveren og sendes videre.



Figur 4.1:

4.1 Definition af grænseflader

Billedbehandlingsprogrammets input kommer fra framegrabberen, som genererer 25 [fps], hvilket er rigeligt i forhold til de opstillede krav på 10 [fps]. Output til klienten skal indeholde 3 informationer pr. objekt. Disse informationer bliver overført til klienten som integers (4 bytes), hvor der benyttes 2 til positionen og en til identifikationsnummer. Programmet skal designes, så kravene fastsat i kravspecifikationen afsnit 3.1.1 opfyldes, således at det systems samlede afviklingstid ikke sænkes.

4.2 Valg af løsningsmodel

Programstrukturen er blevet opstillet på baggrund af de konsekutive funktioner, som billedbehandlingen skal igennem, før billedet er analyseret. Objekterne skal identificeres og positionen bestemmes, hvorefter informationerne sendes til klienten.

Programstrukturen i billedbehandlingen bygges op efter følgende:

- **Initialisering af programmet.** Framegrabberen initialiseres, baggrundsbilledet samples og gemmes.
- **Filtrering.** De løbende billeder hentet og filtreret fra baggrundsbilledet.
- **Labelling og identifikation af objekter.** Hvert objekt tildeles en label, og ud fra areal og omkreds identificeres hhv kopper og robotter.
- **Beregning af massemidtpunkt.** Objekternes massemidtpunkt beregnes, således én pixel bestemmer positionen i et objekt.
- **Koordinattransformationer.** Massemidtpunkternes koordinater transformeres fra billedkoordinater til gulvkoordinater.
- **Kommunikation.** Data lagres i en buffer og sendes til klienten.

For at opnå en nærmere forståelse for, hvordan begrebet billedanalyse kan bruges som et værktøj til at agere robottens sanser til det omkringliggende miljø, gives i det følgende en nærmere beskrivelse af elementerne indeholdt i et billede. Et billede består af en 2-dimensionel lysintensitetsfunktion f(x,y), hvor værdien f er lysintensiteten også kaldet grayscalen for koordinatsættet (x,y). Disse koordinatsæt betegner billedelementerne i et digitalt billede,og benæves pixels. Pixels på 8 [bit] indeholder 256 niveauer (0-255), hvor 0 er sort og 255 er hvid. Mellemliggende niveauer er farvetoner, der kontinuert varierer mellem sort og hvid. Pixels kan derved karakteriseres ved 2 følgende elementer.

- Lysniveauet i en given situation.
- Mængden af lys der reflekteres fra objekter i og omkring det aktuelle billede.

De to elementer kaldes henholdsvis illuminations- og refleksions komponenterne. Omgivelserne på og omkring banen har følgeligt stor indflydelse på disse værdier. [Gonzalez, 1992]

4.2.1 Initialisering af progammet

Under initialiseringen af programmet, konfigureres framegrabberen, der i projektet betragtes som en blackbox. Et baggrundsbilledet af banen tages efterfølgende og gemmes til senere brug, idet dette subtraheres fra de billeder der løbende hentes under programafviklin-

gen. Denne proces sker under filtreringen. Baggrundsbilledet af banen tages uden kopper og robotter, hvorefter robotter og kopper kan placeres på banen. se figur 4.2

(a) Baggrundsbilledet (b) Det indkommende billede

Figur 4.2: De 2 billeder inden behandlingen.

4.2.2 Filtrering

Det første trin under filtreringen subtraherer baggrundsbilledet fra de løbende billeder, og sker pixel for pixel gennem hele billedet. Efter subtraktionen vil der kun være objekter tilbage sammen med utilsigtet støj, se figur 4.3:

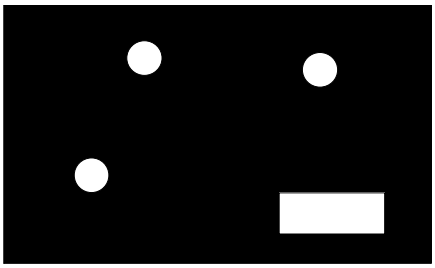
(a) Ideel billed (b) Billede med støj.

Figur 4.3: Billedet efter subtraktionen.

Støj i billedet opstår, fordi spredningen i kameraet kan ændre intensitetsværdien med op til $\pm$ 5 i forhold til de 256 diskrete værdier der findes pr. pixel. Dette afhænger dog af hvilket kamera der anvendes. Dermed kan grayscaleværdien f(x,y) variere fra billede til billede. Hvis grayscaleværdien for en pixel ændres bare et enkelt niveau i forhold til samme pixel på baggrundsbilledet, vil dette under filtreringen resultere i at den pågældende pixels betragtes som værende et objekt. En anden form for støj opstår, hvis lysforholdene ændres

omkring banen. Dermed vil ændrede skygger kunne identificeres som objekter. Dette stiller følgelig store krav til banens omgivelser. Disse former for støj kan der kompenseres for i softwaren.

Billedsegmentation. Når subtraktionen mellem 2 billeder har fundet sted, vil differensen udgøre et billede, som kan behandles. Første skridt i en billedanalyse er segmentering af objekter i billedet, sådan at kun de relevante informationer er til stede uden støj. Den enkleste måde at filtrere støj på er at bruge metoden "simple global threshholding", hvor der anvendes en tærskelværdi T.



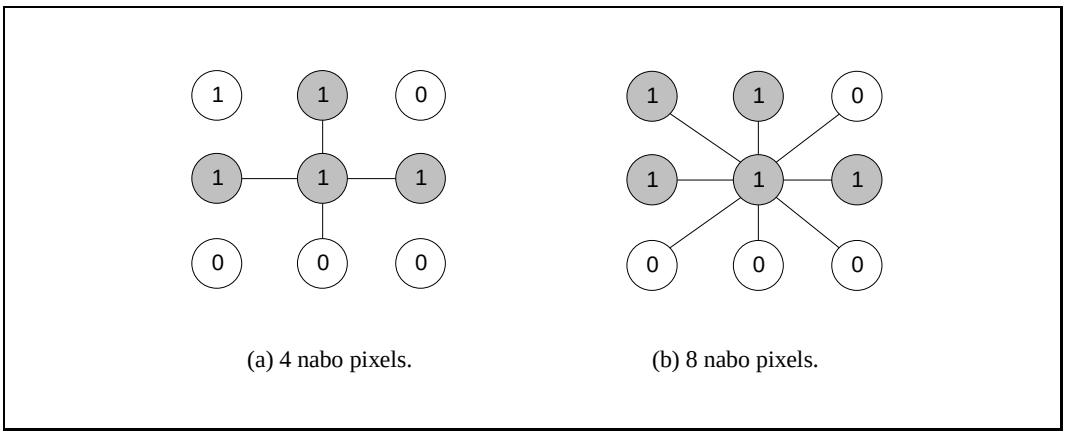
Figur 4.4: Segmenteret billede.

Ved simple global threshholding kan skygger og andet støj filtreres væk, sådan at kun objekter med en grayscale differense over eksempelvis 20, tages i betragtning. Billedet bliver således binært, idet grayscale differensen mellem 2 billeders pixels enten er større eller mindre end tærskelværdien T. Værdier større end T sættes til 255 (hvid) og værdier mindre end T sættes til 0 (sort). Dermed vil billedet se ud som på figur 4.4. [Gonzalez, 1992]

4.2.3 Labelling

Når billedet er segmenteret, undersøges dernæst, hvilke pixels, der hører sammen og udgør et objekt. Dette gøres ved at sammenligne relationerne mellem de enkelte pixels, idet der undersøges, om en pixel og dennes nabopixels har samme binære værdi. Relationen mellem

de enkelte pixels kan foretages ved at undersøge de omkringliggende 4 eller 8 nabopixels, alt efter hvor detaljeret og tidskrævende billedbehandlingen skal være, se figur 4.5:



(a) 4 nabo pixels. (b) 8 nabo pixels.

Figur 4.5: Sammenhæng mellem pixels for hhv 4 og 8 naboer set fra den midterste pixel.

Under labelling kontrolleres samtlige hvide pixels og de omkringliggende nabopixels, for at undersøge om de tilhører det samme objekt. Til programmeringsteknisk løsning af denne proces er der valgt at benytte rekursive funktioner[1]. [Gonzalez, 1992]

Identifikation af objekter. Der findes flere måder til at identificere et objekt på. Det kan være på arealet, hvor objekterne bliver inddelt efter størrelse. Ligger objektets areal inden for et givent område, vil det blive identificeret som et objekt af en given type. En anden metode kan være identifikation ud fra objektets figur. Er objektet kendt på forhånd, kan der laves en analyse af dets kanter. På den måde bestemmes objektet. En tredje måde, der implicerer de to første metoder, benytter et objekts kompleksitet defineret ved forholdet mellem areal og den kvadrerede omkreds. Skal et objekt nemt kunne identificeres, er det en god metode at øge omkredsen i forhold til arealet. Derved opnås en lille kompleksitet.

4.2.4 Massemidtpunkt

Når objekterne i billedet er identificeret, er det efterfølgende nødvendigt at bestemme deres endelige position. Positionen bestemmes i forhold til punktet (0,0) i øverste venstre hjørne af billedet. For de enkelte objekter, der optræder i billedet, vil det være hensigtsmæssigt at anvende en algoritme, som kan bestemme positionen af massemidtpunktet i hvert objekt. Objekternes position kan dermed betragtes som en vektor til koordinatsættet i massemidtpunktet (x,y). Det vil således være muligt for de objekter der er i bevægelse, at bestemme positionenændringen af massemidtpunktet mellem de enkelte frames. På denne måde kan retning og evt. hastighed bestemmes vha. vektorregning. I et binært billede kan massemidtpunktet betragtes som centrum af arealet og findes ved summering af samtlige pixels i x-

[1] Rekursiv funktion: En funktion som kalder sig selv.

og y-retningen. For et givent objekt i billedmatricen [c×r], findes massemidtpunktet ud fra ligning 4.1 og 4.2:

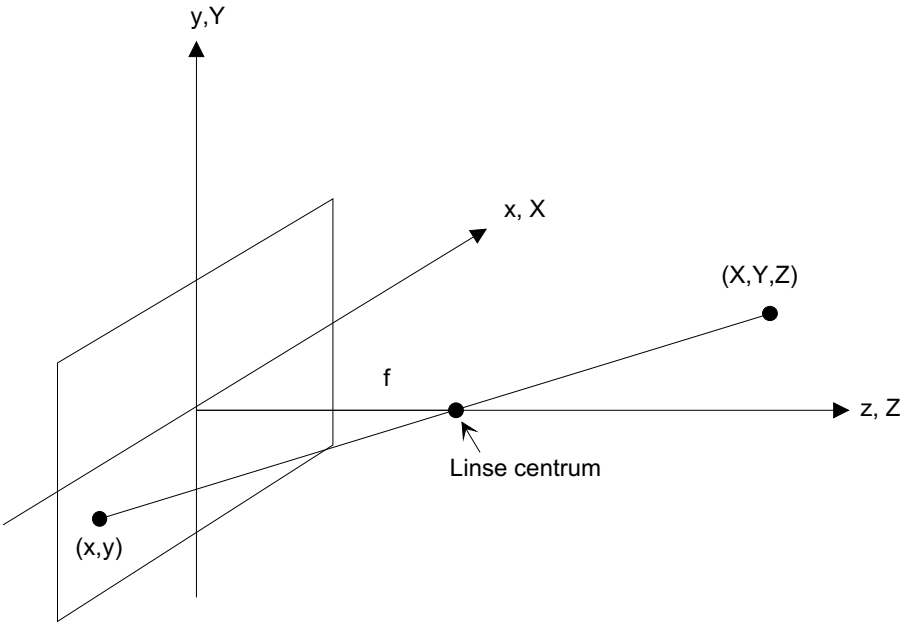
$$\overline{x} = \frac{1}{A}\sum_{i=1}^{c}\sum_{j=1}^{r} i \cdot f[i,j] \tag{4.1}$$

$$\overline{y} = \frac{1}{A}\sum_{i=1}^{c}\sum_{j=1}^{r} j \cdot f[i,j] \tag{4.2}$$

hvor f kan antage værdien 1 for hvide pixels i objektet og 0 for sorte pixels. A betegner arealet som antallet af sammenhængende pixels i objektet, og c og r er hhv. kolonner og rækker i billedmatricen.

4.2.5 Koordinattransformationer

Under billedbehandlingen er det væsentligt at inddrage perspektivtransformation, også kaldet billedtransformation. Perspektivtransformation er anvendelig, når et 3D-punkt projekteres ind på et billedplan. En skitsering af 3D-projektering ses på figur 4.6.



Figur 4.6: 3D punkt projekteret ind på et billedplan.

Definitionen af billedkoordinatsystemet (x,y,z) sker ved at indlægge billedplanet i xy-planet og den optiske akse i form af linsens centrum, langs z-aksen. Centrum i billedet er derfor i origo, og kameraets linse i (0,0,f). Længden f benævnes linsens focal længde, når kameraet

er i fokus. For et givent 3D-punkt (X,Y,Z) benævnes punkterne i koordinatsystemet som verdenskoordinater. Som det ses af figur 4.6, er orienteringen af de to koordinatsystemer sammenfaldende, hvilket er en nødvendighed i den følgende analyse. For at finde billedkoordinaterne (x,y), projekteret fra et 3D-punkt (X,Y,Z), betragtes følgende sammenhæng:

$$\frac{x}{f} = -\frac{X}{Z-f} = \frac{X}{f-Z} \tag{4.3}$$

og

$$\frac{y}{f} = -\frac{Y}{Z-f} = \frac{Y}{f-Z} \tag{4.4}$$

Det anførte minus følger af geometrien, idet et 3D-punkt projekteret ind på billedplanet inverteres omkring z-aksen.

Som det fremgår af ligning 4.3 og 4.4, er sammenhængen mellem billedkoordinater og verdenskoordinater ulineær, idet der divideres med variablen Z.

Når et 3D-punkt omvendt skal bestemmes ud fra et billedplan, kan dette ske ved isolering mht. X og Y i ligning 4.3 og 4.4, hvorved følgende udtryk fås:

$$X = \frac{x}{f}(f-Z) \tag{4.5}$$

$$Y = \frac{y}{f}(f-Z) \tag{4.6}$$

Af ligningerne 4.5 og 4.6 ses det, at rekonstruktion af et 3D-punkt ud fra et billede kræver kendskab til mindst én 3D koordinat som eksempelvis Z. [Gonzalez, 1992]

Som førnævnt kan ovenstående betragtninger kun anvendes, når de to koordinatsystemer er orienteret ens, og dermed sammenfaldende. Kameraet sidder imidlertid ikke lodret over gulvet, men i en vinkel ϕ på 27° i YZ-planet, hvorfor ligning 4.6 ikke kan anvendes.

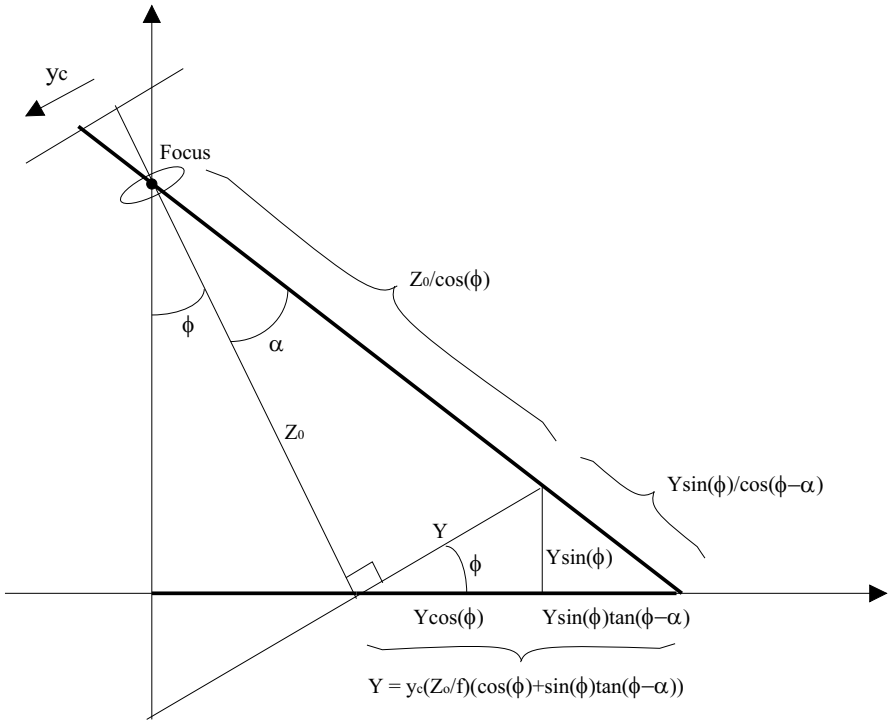
Det vil derfor være ønskeligt at vide, hvordan Y-gulvkoordinaten kan afbildes ud fra billedkoordinaterne. Ved betragtning af figur 4.7, hvor kameraet er orienteret modsat Y-retningen på gulvet, ses sammenhængen mellem følgende størrelser :

$$tan(\alpha) = \frac{y_c}{f} \tag{4.7}$$

Dvs

$$\alpha = arctan\left(\frac{y_c}{f}\right) \tag{4.8}$$

$$Y = \frac{y_c Z_0}{f} \tag{4.9}$$



Figur 4.7: Transformation af y-koordinater.

Da gulv- og kamerakoordinater er modsat orienteret i Y-retningen, bliver gulvkoordinaterne for YZ-planet således:

$$Y_{Gulv} = y_c \left(\frac{Z_0}{f} \right) (cos(\phi) + sin(\phi) tan(\phi - \alpha)) \tag{4.10}$$

hvor længden Z_0 måles.

I XZ-planet sidder kameraet vinkelret på gulvplanet. Som det fremgår af figur 4.8 er gulv- og kamerakoordinaten i X-retningen modsat orienteret. Ligning 4.5 kan derfor skrives som

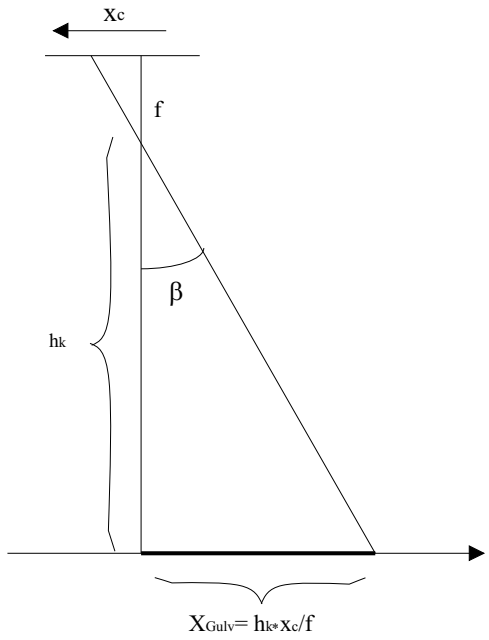
$$X = \frac{x}{f}(Z - f) \tag{4.11}$$

Med definering af kameraets højde over gulvet som $h_k = Z - f$, findes denne ved:

$$h_k = Z_0 \cdot cos(\alpha) = 3.56[m] \cdot cos(27^\circ) = 3.17[m] \tag{4.12}$$

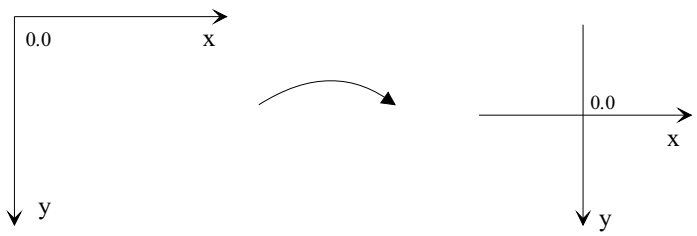
Gulvkoordinaterne i X-retningen findes således ud fra figur 4.8:

$$X_{Gulv} = h_k \frac{x_c}{f} \tag{4.13}$$



Figur 4.8: Transformation af x-koordinater.

Da koordinaterne for massemidtpunkt i billedet har origo i øverste venstre hjørne, transformeres origo til midten af billedet som vist på figur 4.9, således at ligning 4.10 og 4.13 kan anvendes. Transformationen af massemidtpunktet findes dermed ved $X_{mean} = x_{mean} - 384$ [pixels] og $Y_{mean} = y_{mean} - 288$ [pixels].



Figur 4.9: Transformation af origo.

Focallængde. For det anvendte kamera er focallængden opgivet i mm, men ønskes istedet angivet i pixels, sådan at focallængden kan sammenholdes med pixelbredden i billedet. Det er derfor nødvendigt at bestemme focallængden udtrykt i pixels eksperimentelt. Dette gøres ved at tage et billede af et vilkårligt mønster med en given bredde, og i en vilkårligt afstand l fra kameraet. Bredden af mønsteret aflæses i pixels ud fra billedet, og ud fra de tre bekendte, kan focallængden f beregnes af ligning 4.3. I praksis er focallængden fundet ud fra et ternet mønster med en bredde X på 32 [cm] i en afstand l fra kameraet på 110 [cm]. I billedet er mønsterbredden x fundet til 260 [pixels], hvorfor focallængden f beregnes til:

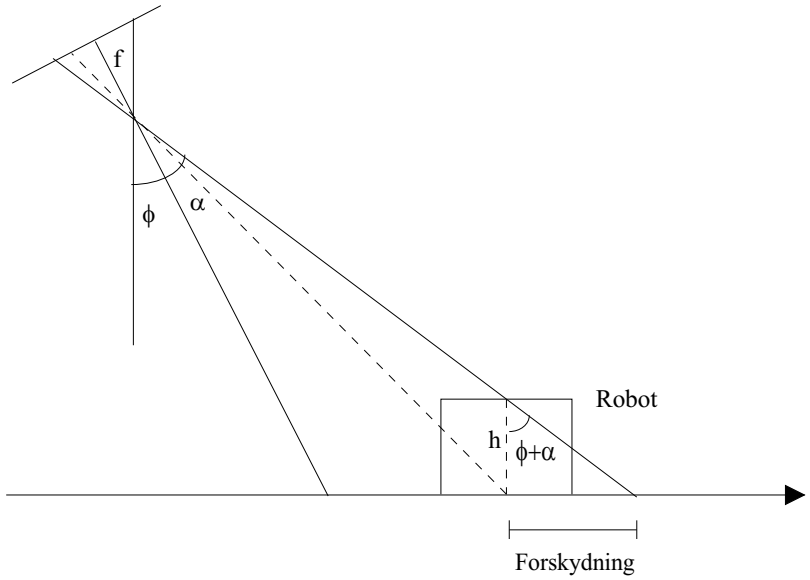
$$\frac{x}{f} = \frac{X}{l} \tag{4.14}$$

$$\Updownarrow$$

$$f = \frac{l}{X} \cdot x = \frac{110}{32} \cdot 260 \approx 895 pixels \tag{4.15}$$

Forskydning. På grund af robottens højde og kameraets vinkel, vil robotten blive genkendt i højden h over gulvet jvf. figur 4.10. Da robotten er væsentlig højere end koppernes massemidtpunkt, bliver robottens position dermed forskudt, hvorfor det er ønskeligt at finde den egentlige position i gulvhøjde. Ved betragtning af figur 4.10 findes forskydningen i Y-retningen ud fra ligning 4.16

$$Y_{forskydning} = h \cdot tan(\phi + \alpha) \tag{4.16}$$



Figur 4.10: Forskydning af robottens position.

I x-retningen vil der ligeledes være en forskydning, når robotten ikke befinder sig under kameraet. Forskydningen kan findes på samme måde ved:

$$X_{forskyning} = h \cdot tan(\beta) \tag{4.17}$$

hvor β findes som $arctan(\frac{x_c}{f})$. Indsat i ligning 4.17 fås forskydningen til:

$$X_{forskyning} = h \cdot \frac{x_c}{f} \tag{4.18}$$

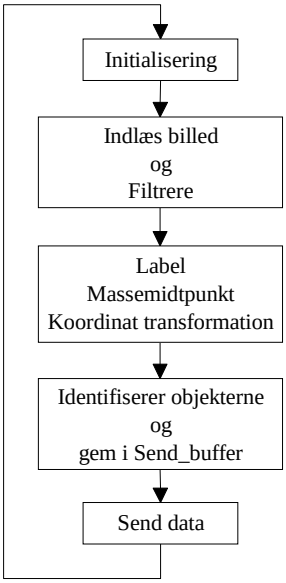
Med de to udtryk for gulvkoordinaterne i henholdsvis X og Y retningen, er det dermed muligt ud fra et punkt i billedplanet at finde det tilsvarende koordinatpunkt på gulvet. På grund af forvrængning i kameraet vil den samme afstand mellem to punkter blive målt forskellig, afhængigt af hvor på banen punkterne er placeret. Forvrængningen ses som parabler, der udbreder sig fra side til side med toppunkt i midten, hvor hældningen er værst fra hjørnerne af billedet ind til midten nederst i billedet.

4.2.6 Sendeprocedure

Til at varetage transmission af billeddata mellem server og klient, udvikles et softwaremodul i serveren. Dette modul kaldes når hvert billede er færdigbehandlet.
Klienten forventer på forhånd data fra serveren. Da datamængden er begrænset, stilles der ingen krav til errorkontrol samt flowkontrol, der er indbygget i TCP. UDP understøtter kun anvendelsen af portnumre og giver derved en hurtigere dataoverførsel end TCP. Det vælges derfor at benytte UDP/IP til dataoverførsel mellem server og klient. For yderligere dokumentation omkring netværksprotokoller og standarder herfor, henvises til appendiks E.

4.3 Implementation af server software

Initialisering. Opbygningen af billedbehandlingsprogrammet kan deles op i 4 elementer, som udgør den løkke, der skal behandle de indkommende billeder. Derudover består programmet af en initialiseringsdel, se figur 4.11. Alle 5 enheder på figuren beskrives kort for at få indblik i løsningen af programmeldelen. En stor del af initialiseringen foregår med opsætning af framegrabberen. Her benyttes biblioteket IRIS Video Library functions. I programmet defineres endvidere, hvordan billederne er opbygget mht. pixels, bredde og længde, så andre billedstørrelser kan bruges. Billederne, der bliver benyttet i dette projekt, er RGB billeder med en bredde på 768 [pixels] (kolonner) og højde på 576 [pixels] (rækker). Hver pixel består, for den valgte hardware, af 32bit, 4 bytes, hvor den første er tom (byte0), og de 3 næste er intensitetsniveauet for hhv. blå (byte1), grøn (byte2), rød (byte3). Til sidst i initialiseringen opsættes sendeproceduren og baggrundsbilledet indlæses i en buffer.



Figur 4.11: Programflow.

Indlæsning og filtrering af billede. Efter initialiseringen fortsættes i en løkke, der først stoppes, når denne afbrydes af brugeren. I løkken bliver der oprettet en pointer til baggrundsbilledet og en pointer til det indkommende billede. For at kunne måle lysændringen i en enkelt pixel, er det kun nødvendig at benytte én af farverne i billedet, hvilket giver mulighed for en hurtigere eksekveringstid. Dette gøres ved at behandle hver fjerde byte i billederne, så beregningsgrundlaget kun består af den blå intensitetsværdi for hver pixel. Når baggrundbilledet og det indkommende billede skal subtraheres, sker det med en pixel fra baggrundsbilledet og den tilsvarende pixel fra det indkommende billede, som pointerne peger på. For at mindske beregningsmængden bliver subtraktionen kun foretaget ved hver 5. pixel pr. kolonne og hver 5. pixel pr. række. Hvis differensen mellem de to pixels er større end tærskelværdien T, hvor T er sat til 20, vil lig-

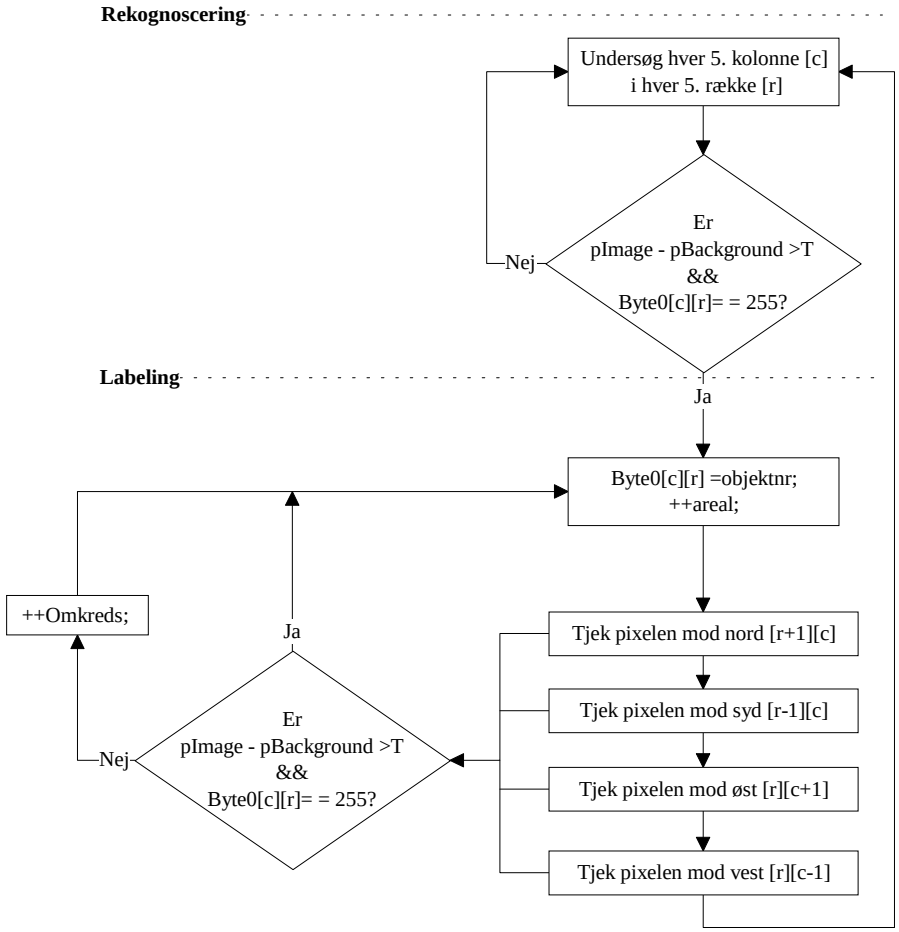
ning 4.19 være sand, og en pixel i et objekt vil være fundet. Her kan det også ses, at objekter mørkere end banen vil blive filtreret fra.

$$pImage[c,r] - pBackground[c,r] > T \tag{4.19}$$

Grunden til at en rekognoscering af billedet kan foretages ved at kontrollere hver 25. pixel skyldes, at det kun er nødvendigt at finde én pixel pr. objekt for at filtrere det. Et kvadrat på 25 pixels er mindre end arealet af det mindste objekt, der ønskes identificeret på banen. Når den første pixel er fundet i et objekt kaldes labelrutinen, hvorefter hele objektet kan identificeres.

Labelling, massemidtpunkt og koordinattransformation. Som før beskrevet benyttes der rekursion til labelling af objekterne. Det foregår ved, at det første objektpixel der rammes ved gennemløb af rekognosceringen, får et objektnummer, der lagres i den tomme byte i pixlen (byte0). Derefter kontrolleres, om ligning 4.19 er sand mht. de 4 nabo pixels. Er dette tilfældet, får disse en label med det samme objektnummer. Nabo pixels kontrolleres én ad gangen, og hvis den første er sand, bliver funktionskaldet for denne lagt på stacken sammen med funktionskaldet for de resterende kald af den første pixel. På den måde findes hele objektet og tildeles en label inden gennemløbet af rekognosceringen forsætter, hvor

den er nået til. Det er under labelling, at arealet og omkredsen af et objekt bestemmes. Rekognoscerings og label rutinen er illustreret på figur 4.12.



Figur 4.12: Programflow over labelling og rekognosceringen.

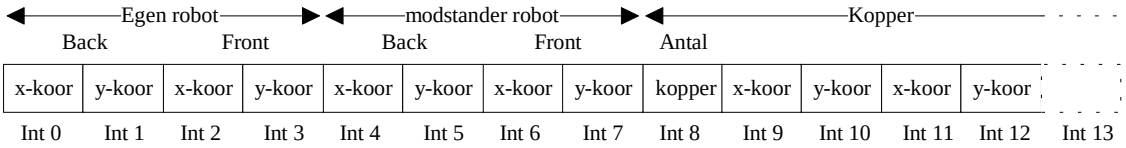
Implementeringen af massemidtpunkt og koordinattransformationen foretages under den rekursive labelrutine, hvor massemidtpunktet i pixels findes ved opsummering af objektets areal samt x- og y-koordinaterne. Når hele objektet er fundet, kan massemidtpunktet findes ved $x_{mean} = \frac{xsum}{A}$ og $y_{mean} = \frac{ysum}{A}$. Herefter kan koordinattransformationen beregnes.

Identifikation og lagring af objekter. Når alle objekter er blevet positioneret, skal de efterfølgende identificeres. Dette kan, som før beskrevet, gøres på flere forskellige måder. Der er 4 forskellige objekter, der skal identificeres på banen. Kopperne, modstandernes robot der består af 2 identitets mærker og egen robot, der består af tre ens cirkler, hvis centre udgør punkterne. På grund af kameraets indfaldsvinkel på banen og dets forvrængning, er der stor forskel på objekternes størrelse og figur, alt efter hvor på banen objektet er placeret.

Derfor kigges der på flere betingelser, hver gang et objekt skal identificeres. Disse er areal, omkreds, kompleksitet og position. De krav, der er blevet stillet til identifikationen, er blevet fastlagt på baggrund af eksperimentelle værdier foretaget kritiske steder på banen. Se testafsnit4.4
Identifikationsbetingelserne for de forskellige objekter er følgelig:

- **Kopper.** De største objekter på banen er kopperne, der tilnærmelsesvis er runde. Dermed vil arealet være det største og kompleksiteten den mindste værdi af objekterne på banen.
- **Modstanders front.** Dette objekt er det mindste på banen både mht. areal og omkreds.
- **Modstanders bagende.** Det bagerste objekt på modstanderen har et areal, der overlapper koppens nogen steder på banen. Omkredsen er imidlertid stor, hvilket bevirker, at dens kompleksitet er større end koppens.
- **Egen robot.** Disse objekter har et areal, der ligger mellem modstanderrobottens 2 objekter, men som på kritiske steder har samme areal som fronten på modstandernes robot. Sammenholdes både areal og omkreds for objektet sker ingen sammenfald med de øvrige objekter.

De identificerede objekter og deres position skal nu sendes til klienten. Resultaterne gemmes i en buffer, hvor de første 4 integers er reserveret til egen robot, og de 4 næste til modstanderens. Næste byte indeholder antallet af kopper og konsekutivt herefter koppernes positioner. Ved at reservere de enkelte bytes i send_bufferen på denne måde, spares informationen om identiteten af objekterne, da denne er fastlagt i bufferen (se figur 4.13).



Figur 4.13: Opdeling af send_bufferen.

Sendeprocedure. Under initialisering af serversoftwaren oprettes en socket, der returnerer en file descriptor. Denne benyttes til at kommunikere over netværket ved forsendelse af billeddata. Ved brug af funktionen *sendto*, hvor blandt andet parametre som IP nummer og portnummer for klienten specificeres, sendes de ønskede data til klienten ved brug af file descriptoren.

4.4 Test

For at opstille nogle kriterier til identifikation af objekterne er det nødvendigt at bestemme de enkelte objekters areal og omkreds i grænseområderne på banen. På grund af kameraets vinkel i forhold til banen, vil areal og omkreds som tidligere nævnt have en meget stor varians afhængig af objektets placering på banen. Målingerne af de enkelte objekter er derfor foretaget i 6 forskellige punkter placeret sådan, at banens kritiske punkter dækkes tilfredsstillende. Som forventet viser resultaterne et maksimum for areal og omkreds i positionen umiddelbart tættest på kameraet, hvor koncentrationen af pixels er størst. Areal og omkreds mindskes omvendt i de to hjørner længst væk fra kameraet.

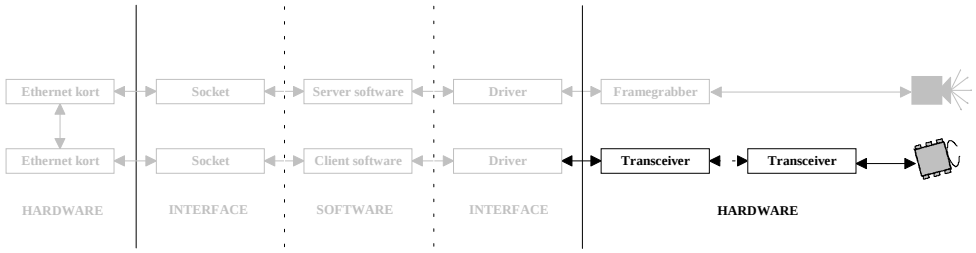
Da det på forhånd er antaget, at billedbehandlingen er det mest tidskrævende element i systemet, vil det være interessant at undersøge eksekveringstiden for hver frame, samt om denne varierer markant under indflydelse af antallet af objekter i billedet. Målingerne viser, at dette ikke har nogen væsentlig indflydelse og med en eksekveringstid mellem 10 [ms] og 40 [ms], kan de ønskede 10 billeder i sekundet altid realiseres. Da kameraet maksimalt kan tage 25 [fps], vil en ekskveringstid på 40 [ms] være optimalt.

4.5 Delkonklusion

Som før beskrevet er nøjagtigheden af billedbehandlingen vigtig for robottens præcision på banen. Dette kapitel har kort gennemgået nogle teorier inden for billedbehandling, som har været relevante for dette projekt. Derudover er de mest markante steder i den programmelle del af serveren blevet gennemgået i generelle vendinger til at skabe et indblik i løsningsmodellen for kildekoden. Ud fra de krav, der blev stillet i kravspecifikationen, og de tests der er foretaget af delsystemet, er opgaven blevet løst tilfredsstillende. Programmet kan nå at behandle samtlige billeder fra framegrabberen og filtrere, positionere, identificere og sende informationerne for alle kendte objekter, uanset hvor de befinder sig på banen.

5 Robothardware

I dette kapitel behandles de forskellige aspekter vedrørende analyse og design af hardware i forbindelse med robot og kommunikation. Ydermere beskrives opbygningen af de fysiske systemer.



5.1 Analyse af hardware

I det følgende beskrives, hvordan de specificerede krav fra afsnit 3.2 ved hjælp af analyse kan realiseres i et design. Der tages udgangspunkt i robottens fysiske design, hvorefter forskellige aspekter ved kommunikationen behandles. Analyserne skal munde ud i valg af microprocessor og hardwaredesign til den endelige konstruktion. Analysen er foretaget ud fra overordnede krav til snitflader til de øvrige dele af det samlede system:

- Kortvarig og simpel kommunikation mellem robot og klientcomputer.
- Mulighed for feedback fra robot, således forskellige status kan udlæses.
- Hurtige databeregninger fra billedserver, idet korrektion af robot kan gøres ved højere hastighed.

5.1.1 Den fysiske robot

For at opnå maksimal manøvredygtighed vælges at anvende samme styring, som benyttes i forbindelse med kampvogne. Denne styring består i, at der på hver side af robotten er et sæt trækkende hjul (et eller flere hjul). Hver side af de trækkende hjul styres hastighedsmæssigt ens. Når der skal drejes, decelereres eller accelereres hjulene på den ene side af robotten, og den vil således dreje.
Styremekanismen er særdeles smidig, men kræver kraftige motorer, da der ved anvendelse af flere hjul, dannes en del friktionsmodstand ved kursændring. Dette skyldes, at hjulene ikke drejes, så de følger en cirkelbue, men slæbes rundt.
Da der ikke er planer om at montere tachometre på robotten anvendes fire hjul, to på hver side, monteret som på en almindelig bil. Dette gøres, da det må forventes, at robotten bliver lettere at styre, ved kørsel ligefrem, idet kraftforskellen mellem de to hjulsæt skal være relativt stor, før bilen drejer. Placeres et hjul på hver side af robotten, skal der kun en ganske lille kraftforskel til, før robotten vil dreje.
God traktion sikres ved at anvende flere hjul, således motoreffekten ikke afsættes på et enkelt hjul, men over et hjulsæt. Til formålet anvendes hjul, som kan give en god frihøjde, således små forhindinger nemt kan forceres samt kraftige tandhjul til kraftudveksling, hjulene imellem.
Til opsamling af plastkopper, skal der implementeres en gribearm, som skal kunne gribe en kop og fastholde den under kørsel.
Rent teknisk udføres konstruktionen i LEGO, som dels er udleveret til brug for formålet, og dels er særdeles velegnet til prototypekonstruktioner. Denne beslutning underbygges også af, at der under konstruktionsfasen må påregnes en del ændringer og forbedringer på konstruktionen.

5.1.2 Kommunikation

Da to robotter uafhængigt af hinanden skal kommunikere med hver deres klientcomputer, stiller det visse krav til kommunikationen.
Da RCX controlleren fra LEGO blev udleveret til projektet, faldt det naturligt at anvende denne i og med, der i controlleren er indbygget alle de features, der ønskes at gøre brug af. Denne controller inkluderer også et IR (infrarødt) sende- og modtagemodul. Desværre sender alle de udleverede controllere fra LEGO på samme frekvens[1].
Måden hvormed data overføres over en infrarød kanal er, at ved at modulere data på en bærebølge[2]. Problemet er således, at to forskellige bærebølger (med samme frekvens) interfererer med hinanden, og signalerne dermed ikke kan skelnes fra hinanden, med mindre det ene er meget kraftigt, sammenholdt med det andet.
Problemet kan løses ved at vælge en helt anden bærebølgefrekvens, således signalerne kan filtreres entydigt fra hinanden, eller ved at udskifte dioder i såvel controller som klientcom-

[1] I forskellige RCX-moduler anvendes lysdioder med samme fysiske egenskaber - og dermed samme infrarøde frekvens.

[2] I tilfældet med LEGO er bærebølgen på 38 [kHz] som mange almindelige fjernbetjeninger til fx fjernsyn.

puterens IR-sender. Dette fører til to nye iagtagelser.
For det første skal den udleverede controller samt den udleverede IR-sender adskilles med henblik på at modificere den hardware, som systemet oprindeligt består af.
For det andet er transmissionshastigheden for controlleren på 2400 [bps]. Denne hastighed vurderes til at være for lav, til at kunne tilfredsstille kravene til en hurtig transmission ud fra det faktum, at det vil tage:

$$\frac{1}{2400[bps]} \approx 417[\mu s] \tag{5.1}$$

at sende én bit. Datapakkerne til LEGOs IR-sender består af:

1. 1 startbit
2. 8 databits
3. 1 paritetsbit (ulige)
4. 1 stopbit

Disse data kodes envidere, således bitantallet af hhv. 0 og 1 bliver 50 %. Dette opnås idet der, når der sendes en databyte, efterfølgende sendes en 'modsat' databyte. Gennemsnitlig fås på denne måde lige mange 0 og 1 i en datapakke. Ovenstående overvejelser medfører en effektiv transmissionshastighed på:

$$\frac{2400[bps]}{11[bit] \cdot 2} \approx 109[databytes/s] \tag{5.2}$$

Dette medfører at der, når der mindst behandles 10 billeder pr. sekund, er plads til at sende:

$$\frac{109[databytes/s]}{10[fps]} \approx 11[databytes/frame] \tag{5.3}$$

Dette giver plads til maksimalt at sende 11 bytes mellem hvert billede, hvis der behandles 10 billeder pr. sekund. I denne overvejelse er der desuden ikke taget hensyn til at robotten skal nå at behandle data og svare igen, som er et krav stillet til robotten. Ydermere skal denne datamængde deles imellem de to robotter, hvis samme RCX-moduler skulle anvendes til transmission.

RCX fra LEGO anvendes derfor *ikke* i dette projekt, hvorfor det er nødvendigt at fremstille egen hardware. Med udgangspunkt i førnævnte transmissionsaspekter, vælges en kommunikationsmetode, som kan ligge til grund for konstruktionen af et controllersystem.
For helt at undgå problemer med, at flere enheder benytter sig af infrarød kommunikation og de deraf følgende problemer, vælges det at benytte radiolinks til transmission af data.
Det er valgt at benytte en integreret radiotransceiver[3] fra Radiometrix, der sender på et licensfrit bånd (433 [MHz]) og samtidigt er i stand til at sende data med en hastighed på op

[3]Transceiver - Et integreret modul som både kan modtage og sende data moduleret på en bærebølge.

til 64 [kbps]. Ved at studere sammenhængen mellem fejlrate, rækkevidde, dataformat og datahastighed er det besluttet at anvende en transmissionshastighed på 9600 [bps] uden paritetsbit [Radiometrix, 2001]. Som ved LEGOs controller, er det stadig nødvendigt at kode data, så antallet af 0 og 1 er 50 % af hensyn til den dataslicer, som i radiomodulet skal bestemme, om der er modtaget et 0 eller et 1. Når der ses bort fra initialisering af radio etc., at det er muligt at sende:

$$\frac{9600[bps]}{10[bits] \cdot 2} \approx 480[databytes/s] \tag{5.4}$$

Ved 10 billeder pr. sekund giver dette:

$$\frac{480[databytes/s]}{10[fps]} \approx 48[Bytes/frames] \tag{5.5}$$

Denne datahastighed er vurderet passende til de styredata, som skal udveksles klient og robot imellem.
Som klient anvendes en almindelig PC, der er udstyret med en RS-232[4] kompatibel serielport, i stand til at sende og modtage data med 9600 [bps]. RS-232 er derfor valgt som kommunikationsprotokol, med den undtagelse, at alle spændinger fra klienten konverteres til CMOS kompatible spændinger, som radiomodulet kan benytte. Sammenhængen mellem RS-232 spændinger og CMOS spændinger kan ses i tabel 5.1. [Nelson, 2001]

	Logisk 0 UD	Logisk 1 UD	Logisk 0 IND	Logisk 1 IND
RS-232	5 til 15 [V]	-5 til -15 [V]	5 til 25 [V]	-5 til -25 [V]
CMOS	0 til 0.4 [V]	3.0 til 5.0 [V]	0 til 0.3 [V]	0.8 til 5.0 [V]

Tabel 5.1: Spændinger i RS-232 og CMOS.

For at opnå den korrekte fordeling af hhv. 0 og 1 i datapakkerne vælges en lidt anden indgangsvinkel, end den der er valgt i RCX. Dette skyldes, at radiomodulet er mere følsomt over for overvægt af enten 0 eller 1 i datapakkerne. Det er således ikke godt nok at sende en inverteret byte efter en databyte.
Metoden valgt til kodning af dataene kaldes Manchesterkodning og er almindeligt brugt i forbindelse med dataoverførsel over radiolink. Metoden går i sin enkelthed ud på, at hver bit der ønskes sendt, sendes som den inverterede bit efterfulgt af den oprindelige bit. Eksempel på Manchesterkodet byte:

MAN('00011100') = '10101001 01011010'

På denne måde bliver vægten pr. datapakke inkl. start- og stopbits præcis 50 %.
Sålænge der anvendes hardware med tilgængelig UART (Universal Asynchronous Receiver

[4]RS-232. Recommended Standard udarbejdet af IEEE (Institute of Electrical and Electronic Engineers) og godkendt af EIA (Electronic Industries Alliance)

Transmitter), og RS-232 protokollen overholdes, kan flowcontrol af datapakkerne overlades til hardwaren, hvilket væsentligt forsimpler kommunikationen. Således kan serielle data sendes ved brug af standard software uden hensyntagen til timing mellem de enkelte bits. Således er de nederste niveauer i OSI modellen (Open Systems Interconnection, se endvidere afsnit E.1), som beskæftiger sig med fysisk forbindelse og rutning af data på plads. Øverste niveau af protokollen, som beskriver dataformat, syntax, etc., kan således fastlægges.

Kommunikation til robotten. Da robotten primært skal modtage forskellige styrekoder med dertil hørende parametre, er det besluttet, at data sendes som opkoder (operationskoder) efterfulgt af en parameter. Der allokeres en byte til opkoden og en byte til parametren. Disse to bytes Manchesterkodes før afsendelse og kommer derved til at fylde fire bytes. En femte byte medsendes de fire databytes som et simpelt fejlcheck, bestående af de øvrige fire bytes XOR[5] resultat. Resultatet af denne operation gemmes i den femte byte, som også medsendes over radiolinket. På modtagersiden XORes alle fem bytes og er resultatet 0, er transmissionen med stor sandsynlighed fejlfri.

De implementerede opkoder er:

Opkode	Handling	Parameter
00000000	N/A	N/A
00100000	Frem	Fart
01000000	Tilbage	Fart
01100000	Venstre	Værdi venstre hjul kører langsommere end højre
10000000	Højre	Værdi højre hjul kører langsommere end venstre
10100000	Stop	N/A
11000000	Action	Brugerdefinerede handlinger
11100000	N/A	N/A

Tabel 5.2: Oversigt over implementerede operationskoder.

Kommunikation til klienten. Efter data er modtaget af robotten, skal den udføre et XOR fejlcheck som ovenfor beskrevet. Hvis denne kontrol viser, at data er modtaget korrekt, skal robotten sende et acknowledge tilbage til klienten. Dette signal skal sendes, inden robotten begynder at udføre den handling, den har modtaget gennem kommunikationskanalen, for at få svar fra klienten hurtigst muligt.
Det vedtages, at acknowledge fra robotten skal opfylde følgende datastruktur:

- Byte: Den oprindeligt modtagne opkode.

[5] eXclusive OR - binær regneart.

- Halv byte: Data ok (0000b) eller data ikke ok (0110b).
- Halv byte: Har kop (0000b) eller har ikke kop (0110b).

De to bytes Manchesterkodes, og der tilføjes en XOR kontrolpakke, hvorved den samlede datamængde, som sendes tilbage til klienten, bliver fem bytes.
Når den tilgængelige transmissionskapacitet tages i betragtning, ses det, at det nemt kan lade sig gøre at udvide antallet af sendte pakker mhb. på implementering af fx kryptering, fejlkorrektionsmulighed eller lignende.

5.1.3 Microcontroller

Der vælges en MCU (Micro Controller Unit), som kan styre hardwaren i robotten, og som kan tilfredsstille samtlige krav, stillet af såvel kommunikation som den fysiske robot:

- Hurtig processor af hensyn til bl.a. kommunikation.
- RISC struktur (Reduced Instruction Set Computing) da processoren skal kodes hardwarenært.
- Ca. 16 I/O porte (Input/Output) - svarende I/O for to bytes.
- ADC (Analog Digital Converter) til måling af analoge værdier.
- PWM (Pulse Width Modulation) til at styre motorhastigheder.
- Mulighed for interrupts, således vigtige procedurer kan prioriteres.
- UART til håndtering af data i forbindelse med radiolinket.

En PIC16F877 microcontroller fra Microchips anvendes til formålet, idet den opfylder alle de opstillede minimumskrav. Valget underbygges også af et a priori kendskab til processortypen opnået i forbindelse med tidligere opgaver.
Fordelene ved den valgte PIC er, at den er hurtig (20 [MHz]), har indbygget PWM, UART, ADC, etc. med tilhørende hardware interrupt. Desværre findes der kun interrupts i ét niveau, hvorfor en prioritering mellem forskellige interrupts skal klares softwaremæssigt. Der kan derfor ikke opbygges et 'hard real-time'[6] system, men dog alligevel et hurtigt system med flere realtids aspekter.

I en PIC16F877, samt de fleste andre mindre microprocessor systemer, haves kun meget begrænset fysisk hukommelse (8 [KB]), ingen virtuel hukommelse og hardwarenært interface. Disse ting stiller store krav til den måde, processoren styres på. Operativsystemet eller taskmanageren skal med andre ord være meget effektiv og hurtigt tildele ressourcer til de tidskritiske operationer, således der ikke opstår store stakke af data.

[6] Hard real-time: Et system hvor processer køres præcist og deterministisk til eksakt givne tider - overskredne deadlines tolereres ikke.

Da det på forhånd vides hvilke processer, som ønskes udført, og hvilke der er tidskritiske, vælges statisk cyklisk programafvikling. På denne måde fastsættes programafvikling og prioritering *før* processoren startes, og der er således ikke mulighed for løbende at ændre prioriteter for forskellige procedurer. Dog anvendes interruptstyring til at sikre, at de tidskritiske procedurer afvikles til tiden. Fx er det vigtigt at tømme receive bufferen for UARTen til tiden, således der ikke opstår overflow, og data går tabt.
Kommunikationen har meget høj prioritet for at sikre korrekt adfærd hos robotten, derfor opprioriteres kommunikationsdelen i controlleren. Ligeledes ønskes resultatet af analoge målinger udlæst, så snart data er klar, hvorfor også dette aspekt prioriteres. Endelig skal kalibreringer, justeringer, etc. udføres med faste tidsrum, hvorfor en timer i controlleren vælges til formålet. Timeren prioriteres også, således funktioner tilknyttet den, udføres med højere præcision end almindelig programafvikling.
Den resterende behandling af data køres som almindelig programafvikling, idet behandlingstiden for små datasæt er meget kort, da processoren kan udføre 5,000,000 operationer pr. sekund[7].
Programafviklingen kan ud fra ovenstående overordnet vises således:

1. Initialisering: Generel opsætning af processor, porte, PWM, UART, ADC, radiolink, etc.

2. Interrupts (forekommer når et hardware interrupt trigges): Tømning af UART buffer, udlæsning af ADC data, kørsel af timerbaserede procedurer.

3. Programafvikling: Øvrig databehandling, udførsel af ikke tidskritiske funktioner.

Således er den indledende analyse af den samlede hardware færdig, og designet kan betragtes.

5.2 Design - PIC

I dette afsnit behandles opsætningen af hardware i microprocessoren. Der tages udgangspunkt i et minimumssystem, således microprocessoren kan opereres og aftestes. Minimumssystemet opbygges på et beadboard[8], funktionskontrolleres og overføres så til en printplade, således der ikke opstår løse forbindelser under kørsel.

PIC Minimumssystem. PICen tilhører en familie af digitallogiske komponenter, som er afhængige af en fast spænding på 5 [V] for korrekt operation. Afviges der fra denne spænding, med selv små spændingsforskelle ($\pm$ 0.5 [V] for PIC16F877 ved 20 [MHz]), kan det få fatale konsekvenser for processoren, idet den ofte resettes, så programafviklingen starter forfra. Vigtigheden af en konstant spænding er således stor, og for at tilfredsstille dette

[7] Det tager internt mindst fire clockpulser ved 20 [MHz] at udføre én instruktion

[8] Beadboard. Et brædt med indbyggede ledninger, som komponenter kan isættes uden brug af ekstra ledninger eller loddekolbe.

krav, opbygges en spændingsregulator. Til formålet anvendes en LM78L05 fra National Semiconductors, der udmærker sig ved at kunne regulere en indgangsspænding på 7 [V] og derover, ned til 5.0 ± 0.25 [V] ved en kontinuert strøm på 100 [mA]. Denne komponent danner således basis for spændingsforsyningen af de øvrige digitallogiske komponenter, som ønskes anvendt i konstruktionen. [National-Semiconductors, 2000]
For at operere PICen kræves, ud over en fast spænding, et krystal, som skal sikre en clockfrekevens på 20.000 [MHz], en pullup modstand, samt to kapacitorer [Microchip-16F87X, 2001]. Til dette system skrives et lille program i assembler, som kan teste alle I/O porte, således det sikres, at processoren fungerer som den skal. Debugging foregår på dette stadie ved hjælp af lysdioder. Efter det er konstateret, at systemet fungerer, programmeres en kerne til processoren.

Kernen. Til at styre de forskellige processer i microprocessoren, laves en kerne, som skal holde styr på de ressourcer, der er til rådighed. Den skrives i assembler, da der i dette sprog er mulighed for at styre processoren ned til de enkelte instruktioner, der ønskes udført. Dette er især nyttigt i forbindelse med kalibrering og timing af ekstern hardware, kommunikation, etc.
Ud fra analysen bestemmes hvilke moduler i processoren, der er nødvendige at anvende. Disse moduler initialiseres og vedligeholdes af kernen, som på den måde sørger for den grundlæggende styring. Det er efterfølgende muligt at lægge forskellige procedurer ind, der anvender de data, som kommer ud eller ind til de forskellige moduler. I det følgende gennemgås de anvendte moduler og den overordnede funktionalitet:

USART. USARTen (Universal Synchronous Asynchronous Receiver Transmitter) anvendes i forbindelse med radiomodulet. Det er denne enhed, som sørger for, data præsenteres i korrekt format og afsendes med en fast bitrate. For at dette kan lade sig gøre, skal tilknyttede registre i PICen sættes korrekt op. I registrene vælges bithastigheden til 9600 [bps] med 1 stopbit og ingen paritetsbit. Ligeledes sættes USARTen op til at interrupte, altså afbryde den kørende kode, hvis der er modtaget en komplet byte. Dette sker for at sikre, at alle data fjernes fra den serielle buffer, så hurtigt som muligt. I modsætning til en UART i en almindelig pc, som kan rumme omkring 16 bytes, er der kun plads til to bytes i bufferen på PICen - modtages der flere, uden bufferen tømmes, overskrives de først modtagne data. Hvis dette sker, sættes et advarselsflag, som gør det muligt at opdage sådanne fejl. Tilsvarende kan der polles på 'framing errors', som fremkommer, hvis UARTen ikke modtager en korrekt sekvens af hhv. start-, data-, paritet- og stopbits.
Alle fejlbehæftede serielt modtagne datapakker kasseres med det samme, og de pakker, som modtages fejlfrit, kontrolleres med en paritetsbyte, før de anvendes. Dette sker for at forbedre støjimmunitet, modvirke jamming og lignende, således der ikke udføres utilsigtede operationer, da mange trådløse apparater, som fx termometre, telefoner, bilalarmer, etc. opererer på netop samme frekvens, som de anvendte radiomoduler.

PWM. Til styring af fremdriftsmotorerne, anvendes puls vidde modulations (PWM) modulerne i PICen. PWM er en meget anvendt metode til at regulere spændinger. I stedet for at bruge en transistor som en variabel modstand, hvor overskydende energi omdannes til varme, anvendes PWM til at trække en transistor, som åbner og lukker. Ved at justere den tid transistoren er lukket, i forhold til den tid den er åben, fås en gennemsnitlig spænding, lavere end den initielt påtrykte spænding. Da transistorerne enten er lukkede eller åbne forekommer der næsten intet spild ved metoden, og det er derfor muligt at regulere relativt store effekter på denne måde.
I processoren styres PWM ved hjælp af timere og registre. Ved at indstille hvor meget en given tæller skal løbe til, defineres en periodetid, som modsvarer den switchfrekvens, der ønskes anvendt. Når timeren starter, sendes et højt logisk signal fra processoren. Dette signal forbliver højt, indtil en justérbar værdi, mindre end eller lig med perioden, er opnået, hvorefter processoren sætter signalet logisk lavt. På denne måde fås et forhold mellem det logisk høje signal og hele periodens varighed kaldet duty cyclen. De to indbyggede PWM moduler kan justeres med en præcision på 10 bit, hvilket vil sige 1024 forskellige niveauer, som er betydeligt mere end nødvendigt i forbindelse med styring af robotten. Processoren har 8 bit datastruktur, hvorfor det besluttes, kun at anvende de otte højeste PWM bits (256 niveauer).
Kernen initialiserer PWM, således korrekt frekvens opnås, men sørger også for at justere duty cyclen. Når der fx modtages en ordre fra radioen, om at give motorerne maksimal spænding, vil kernen således gradvist øge duty cyclen, således motoren accelereres gennem alle 256 trin. Dette gøres dels for at spare effekt, idet der bruges megen effekt til pludselig opstart af elektromotorer, og dels for at beskytte såvel gear som give bedre traktion.

ADC. Processorens analog til digital converter anvendes også i robotten. Helt specifikt er der tale om en 10 bit converter, som kan måle på mellem en og otte forskellige I/O porte på processoren. De analoge værdier skal bruges i forbindelse med afstandsmåling, således robotten får mulighed for selv at kunne måle afstanden til et givent objekt. Ligesom beskrevet for PWM, anvendes også for ADC kun de otte mest betydende bits, idet 256 niveauer af spænding regnes for en tilfredsstillende præcision.

De analoge værdier omsættes til digitale gennem en metode, kaldet successiv approksimation. Helt overordnet kan metoden opdeles i følgende skridt:

1. Sampling. En 120 [pF] kapacitor i ADCen oplades til den analoge spænding, som ønskes målt.

2. Hold. Kapacitoren afkobles fra den analoge kilde.

3. Successiv approximation. Et indbygget 10-bit register tilsluttet en DAC (Digital til Analog Converter) påtrykkes værdier, startende med den mest betydende bit (512), hvorefter resultatet af PICens egen DAC sammenholdes med det analoge resultat i en komparator. Er spændingen over DACen højere end det analoge input, sættes første

bit til 0. Er DACens værdi lavere, sættes bitten til 1. På denne måde arbejder ADCen sig igennem alle 10 bits, indtil en digital værdi er fundet.

Grundet sample kapacitoren, skal der dels gå et stykke tid, mens målingen foretages og dels et stykke tid til at nulstille DAC-registret. Disse tider er (worst case) hhv. 19.7 og 12.8 [μS]. Det ses således, at det er muligt at foretage godt 30,000 målinger pr. sekund, hvilket er mere end rigeligt til at opnå gode afstandsmålinger, for den hastighed robotten forventes at bevæge sig med. [Microchip-Midrange, 1997]

Timer. For at servicere de anvendte moduler, og for at styre opdateringen af forskellige data, anvendes en intern timer i processoren. Denne timer sættes til at interrupte almindelig programafvikling, hver gang den udløber, hvilket vil sige når timeren går fra 255 til 0. Denne tid vælges under initialiseringen til 3.28 [ms] ud fra følgende:

$$\begin{aligned} T_{CPU} &= \frac{1}{5[MHz]} = 200[ns] \\ T_{PSA} &= 64 \\ T_{TMR} &= T_{CPU} \cdot T_{PSA} \cdot 256 \approx 3.28[ms] \end{aligned} \tag{5.6}$$

T_{CPU} er periodetiden mellem hver CPU instruktion, T_{PSA} er en intern justérbar prescaler og T_{TMR} er periodetiden mellem hvert interrupt. Kernen i PICen kører således alle prioriterede modifikationer af data, motorer, etc. hvert 3.28 [ms] og kan således hele tiden holde forskellige registre ajour, mens andre procedurer kører med mindre tidskritiske funktioner.

5.3 Design - periferi

I det følgende beskrives de komponenter, som er tilsluttet PICen, samt deres logiske tilslutning.

Proximity detektor. Objekter foran robotten findes ved hjælp af en proximity detektor[9]. Detektoren opbygges ved hjælp af en IR (infrarød) lysdiode, samt en IR modtagediode. Disse monteres parallelt, således modtageren ikke kan se lyset fra senderen, med mindre et objekt reflekterer lyset fra senderen ind i modtageren. Grunden til der er valgt infrarøde dioder er, at disse fås med indbygget optik, således lyset sendes meget koncentreret fremad ($\pm 8^\circ$ grader for den valgte type), og dermed er detektérbart i en længere afstand. Den infrarøde modtager har en modtagevinkel på $\pm 12^\circ$ og er afskærmet for dagslys. Dioden placeres i et spændingsdelerkredsløb, således den sænker spændingen ved belysning.
Når der skal bestemmes, om et objekt er i nærheden af robotten, måles den analoge spænding over modtagedioden med ADCen, når IR-senderen er slukket. Umiddelbart efter denne måling, tændes IR-senderen, og der tages en ny måling. Hvis første og anden måling begge

[9] Proximity detektor: Detekterer objekter i nærheden.

giver ca. samme resultat, er lyset ikke reflekteret, hvorfor der følgelig ikke er et objekt foran detektoren. Er der omvendt forskel på de to målinger, er dele af lyset fra senderen reflekteret ind i modtageren. Denne tilnærmelse kan anvendes, så længe der kun går meget kort tid mellem målingerne, da det så antages, de øvrige ydre forhold ikke når at ændre sig. For at sikre målingernes validitet, foretages 20 målinger, hvoraf mindst 15 skal være positive, før detektoren skal melde, at et objekt er fundet. Dette stiller krav til mange og hurtige målinger, hvorfor dette også styres af kernen i microprocessoren. Det skal således være muligt at aktivere detektoren via radio, mens målingerne styres af kernen.

Radiomodulet. Transceiveren fra Radiometrix er i stand til at sende og modtage data med 64 [kbps]. En stor fordel ved modulet er, at det er integreret i én komponent, således det kun er nødvendigt at tilslutte en datacontroller, samt noget simpelt logik til at styre forskellige indbyggede funktioner. Modulet har fire forskellige indstillinger, styret ved hjælp af to ben, og disse består af:

- Sleep. Radiomodulet kan ikke modtage eller sende, men afventer i standby, og der bruges kun meget lidt strøm ($< 1[\mu A]$).

- Receive. Når modulet sættes til at modtage data, går der et stykke tid, før data fra radioen er valide, hvilket skyldes, der i radiomodulet anvendes en adaptiv datasliser. Datasliceren måler det indkommende RF signal, efter diverse forstærkninger og filtreringer, og sammenligner det med en tærskelværdi. Er signalet højere end tærskelværdien, betragtes signalet som værende logisk højt. Er signalet lavere, fås et logisk lavt signal. Da datasliceren er adaptiv betyder det, at den skal initialiseres, før den kan skelne høje og lave signaler. Dette gøres ved at sende en preamble, bestående af '01010101', i mindst 3 [ms]. Når dette er overstået, er det muligt at modtage valide data fra radioen, så længe de afsendte data overholder en mark/space ratio på 50 % - de data, som sendes skal med andre ord også overholde det kriterium, at halvdelen af de afsendte bits skal være hhv. høje og lave.
  Der er vedtaget, at robotten ikke selv må initiere radiokommunikation. Derfor opereres radiolinket primært i receivemode, således kommandoer fra klienten opfanges hurtigst muligt.
  Så snart radiolinket opereres i receivemode, kommer der støjgenereret data på udgangen. Disse data er selvsagt ikke valide og ønskes ikke modtaget af microprocessorens UART, da der statistisk set på ét eller andet tidspunkt vil modtages støj, som microprocessoren kan opfatte som valide data.
  Løsningen på problemet er, at anvende radiomodulets carrierdetect, et signal som går logisk lavt ved modtagelse af en bærebølge på 433 [MHz], i forbindelse med datasignalet. Den digitale filtrering klares med en logisk inverter samt en logisk AND. Resultatet er bevist med perfekt induktion i sandhedstabellen på figur 5.3.

RXD	C	$\overline{RXD}$	$\overline{C}$	$\overline{RXD \cdot C}$	$RXD \cdot C$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

Tabel 5.3: Sandhedstabel for digital filtrering.

På denne måde sikres, at datalinien til microprocessorens UART holdes høj, indtil der kommer data, som foreskrevet i RS-232.

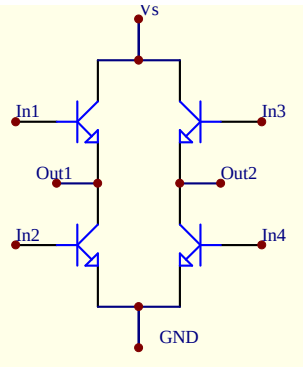
- Transmit. Radiomodulet sættes i transmit-mode, og data moduleres på bærebølgen. Efter ca. 100$[\mu s]$ udsendes bærebølgen med fuld signalstyrke.

- Loopback self test. Radiomodulet kan aftestes for funktionalitet.

I tilfælde af såvel modtagelse, som afsendelse, forsinkes data gennem radiomodulet med 15 $[\mu s]$, men det har ingen betydning for dataintegriteten, når timingen for hele systemet tages i betragtning.
For at sikre gode sende- og modtageforhold, uden at skulle være afhængig af en lang antenne, er det valgt at udstyre radioen med en helixantenne. Antennen består af et stykke legeret kobbertråd, viklet 24 gange om en 3.2 [mm] form. Derfor bliver antennen meget kompakt og udgør dermed mindst mulig gene, når robotten skal køre rundt. Da rækkevidden for radiomodulet i databladet er angivet til 200 [m] med en almindelig trådantenne, er den reducerede rækkevidde ved valg af helixantenne, negligeabel, når den ønskede rækkevidde for robotten er 10 [m].
Øvrige informationer om radiomodulet kan findes i appendiks A.

H-bro. H-broen er den fysiske komponent, som skal omforme de logiske PWM signaler til effektregulering. En h-bro opbygges af fire transistorer, placeret i et H, som på figur 5.1,

således strømmen gennem dem kan løbe gennem de forskellige transistorer. Konstruktionen er yderligere omtalt i appendiks B.



Figur 5.1: Opbygning af h-bro.

Der anvendes en integreret h-bro til robotten, idet der i disse kredse er indbygget logik til at drive de forskellige transistorer. Den integrerede kreds fra STMicroelectronics består af to separate h-broer integreret i én komponent, som hver styres ved hjælp af følgende ben:

- Enable. Aktiverer h-broen.
- Input1. Aktiverer output1.
- Input2. Aktiverer output2.

Hvis både input1 og input2 har samme status (logisk høj eller lav), bremses motoren. Er de forskellige, går der en strøm, gennem den ene eller den anden af h-broens diagonaler. Deaktiveres hele h-broen, kører motorerne i frigang.

For at microprocessoren kan styre h-broen med PWM i både frem og tilbagegående retning, kræver det, at der af h-broen kan skelnes mellem frem og tilbage. Måden dette løses på, er ved at anvende tre ben på microcontrolleren, således der tildeles en bit til at aktivere h-broen og en bit til hhv. frem og tilbageløb. En ekstern AND kreds sørger for at lave AND operation på PWM signalet, som udtages fra et andet I/O ben, og frem- eller tilbage bitten for h-broen. Det kræver på denne måde syv I/O ben på processoren at styre den dobbelte h-bro på den valgte måde, ud fra følgende tilslutning:

- Enable 1+2 (for den dobbelte h-bro).
- PWM1 (PWM for motor 1).
- Enable 1 (motor 1 fremad).
- Enable 2 (motor 1 tilbage).

- PWM2 (PWM for motor 2).
- Enable 3 (motor 2 fremad).
- Enable 4 (motor 2 tilbage).

Hver af h-broerne kan klare en kontinuert dc strøm på 2 [A], hvilket er mere end de valgte motorer bruger ved maksimal belastning.
PWM frekvensen til de to motorer vælges ud fra to forskellige kriterier:

1. Frekvensen skal ligge over det hørbare område, således hørbar akustisk støj undgås (>20 [kHz]).
2. Frekvensen må ikke nærme sig den maksimale kommuteringsfrekvens for den valgte h-bro (<40 [kHz]).

Det vælges at anvende en PWM frekvens på 25 [kHz], således begge krav opfyldes.
En af fordelene ved at vælge en høj PWM frekvens er, at den switchede spænding varierer mindre over tid, til gengæld afsættes der mere varmeenergi, da transistorerne ikke åbner momentant, hvorfor nyttevirkningen nedsættes en smule.
Når transistorer tændes og slukkes med høj hastighed, fremkommer der også en del transitionsstøj, som kan genere de øvrige elektriske komponenter, hvorfor h-broen støjmæssigt afkobles jf. databladet [STMicroelectronics, 2000].

5.4 Delkonklusion

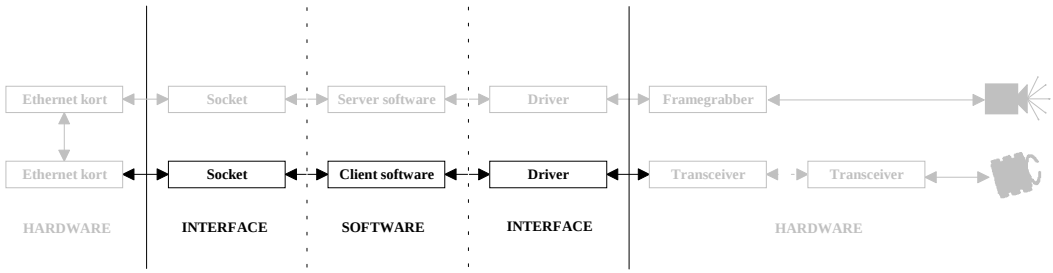
Det samlede hardwaresystem er designet, således det er dynamisk og nemt at ændre, da der i forbindelse med aftestning etc. må forventes at skulle foretages justeringer. Hardwaren er som specificeret opbygget på beadboard og er aftestet med hensyn til generel funktionalitet med særligt henblik på motorstyring, kommunikation og proximity detektor.
Efterfølgende er hele systemet overført til printplade ved hjælp af Protel[10] og igen intensivt testet. Efter endt aftestning indbygges printpladen i den fysiske robot, og aftestningen af det samlede system kan foretages.
Dokumentation af elektriske diagrammer, printplade, etc. findes i apppendix C.

[10] Protel. CAD program til design af elektriske diagrammer og print.

Datahåndtering i klient

6

Formålet med dette kapitel er at beskrive de dele af klienten, som står for håndtering af data flowet. Det vil sige modtagelse af data fra billedserveren samt afsendelse af kommandoer til robotten. Herunder beskrives, hvorledes de forskellige kommunikationsrutiner er opbygget, hvordan de forskellige processer koordineres samt hvilke interfaces, der tilbydes til den ovenliggende del af klient programmet, som står for den egentlige analysering af de indkommende billeddata med henblik på afgørelse af robottens handling.



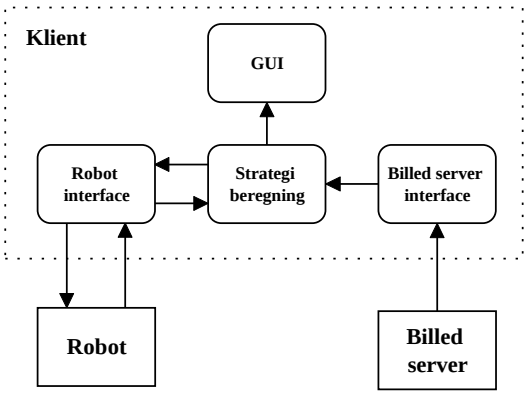
6.1 Definition af grænseflader

I dette afsnit beskrives de delsystemer, som klienten er opdelt i. Det vil sige delsystemerne, der fungerer som interfaces mod de eksterne systemer i form af robotten og billedserveren, samt delsystemerne som står for håndteringer og koordinering af dataflowet internt i klienten. Denne opdeling i delsystemer er endvidere vist ved figur 6.1, som danner udgangspunkt for den følgende diskussion af de egentlige krav til og formål med delsystemerne.

6.1.1 Klienten overordnet

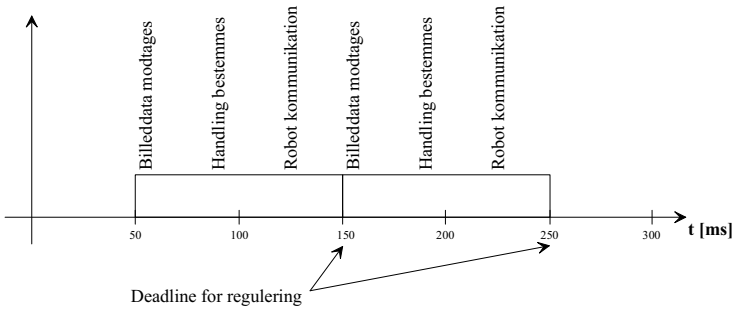
Klienten kan overordnet betragtes som controlleren i et realtidsproces kontrol system, som overvåger processen via et kamera. Processen, der kontrolleres, er selve robottens, mens

aktuatorerne som controlleren benytter til styring af processen, er robottens motorer. Controllerens kommunikation med sensoren og akuatorerne foregår henholdvis vha. en netværksforbindelse og et radiolink.



Figur 6.1: Klientens grænseflader mod serveren, mod robotten og mod det ovenliggende klient lag.

I kravspecifikationen afsnit 3 er det valgt, at der som udgangspunkt kan forventes billeddata klar med en frekvens på 10 [Hz], samt at disse 10 [fps] alle skal danne grundlag for styringen af robottens handlinger - dvs. alt billeddata modtaget med 10 [Hz], skal behandles. Disse krav til timingen er endvidere vist ved figur 6.2.



Figur 6.2: De overordnede krav til timingen i klienten, hvor billeddata ankommer med frekvensen 10 [Hz], skal behandles inden der er gået 100 [ms].

Disse restriktioner i forbindelse med modtagelse af billeddata, sætter en række krav til valg af hardware og operativsystem for klienten, som kort omtales i det følgende.

- Krav til hardware:
  - Håndtering af **seriel kommunikation**, der benyttes til kommunikation med aktuatorer.

  - Mulighed for **netværkskommunikation**, som benyttes til opsamling af data fra sensorerne.
  - Skal kunne **behandle data** inden for de givne **tidsrammer** - dvs. fra der modtages billedata, må der maksimalt gå 100 [ms], før klienten igen er klar til at foretage en ny regulering af robotten.

- Krav til operativsystem:
  - Håndtering af **seriel kommunikation** på hensigtsmæssige måde.
  - Mulighed for **netværkskommunikation**.
  - Kørsel af **multiple processer** mhb. på at kunne modtage data fra robotten og fra billedserveren på samme tid, samt generel optimering af programflowet.
  - Skal introducere mulighed for brug af forskellige **software interrupts**. Dvs. processer skal kunne reagere på forskellige hændelser, som eksempelvis, at der er data fra robotten, data fra serveren el.lign.

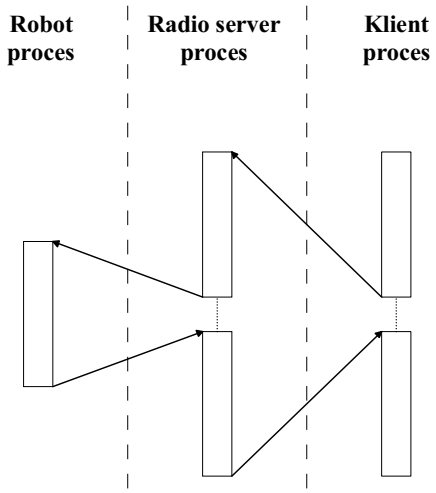
6.1.2 Robot interface

Formålet med robot interfacet er at indføre mulighed for kommunikation med robotten i form af udveksling af kommandoer, hvor kommandoer fra klienten til robotten er en række foruddefinerede styrekommandoer, mens data flowet fra robotten til klienten består af status informationer omkring robottens nuværende tilstand samt acknowledge informationer vedr. seneste datatransmission.

Overordnet set skal robot interfacet på initiativ af kald fra en anden proces, sende kommandoer til robotten, straks den anden proces anmoder om dette. Dernæst skal robot interfacet vente, til robotten sender data tilbage, hvorefter disse skal videresendes til den kaldende proces. Det er endvidere vist ved figur 6.3.

Mere specifikt er kravene til robot interface:

- Skal kunne sende kommandoer til robotten, når en given proces ønsker dette.
- Skal kunne modtage informationer fra robotten vedr. status for grabberen, samt informationer om den seneste transmission lykkes.
- Skal ikke forstyrre andre processer i klienten - herunder i særdeleshed andre I/O håndterende processer. Det modsatte skal endvidere være gældende.
- Skal ikke introducere væsentlige forsinkelser i det samlede system. Dvs. transmissionstiden skal være så lav, at der er tid til den egentlige behandling af data i klienten indenfor de givne tidsrammer (< 100 [ms]).



Figur 6.3: Princippet i opbygningen af robot interfacet, hvor en kaldende proces blokeres, indtil der er svar fra robotten.

6.1.3 Billedserver interface

Billedserver interfacets opgave er at opsamle de billeddata, som sendes fra billedserveren. Som beskrevet i afsnit 4.2.6, sender billedserveren efter hver billedbehandling en UDP pakke indeholdende de givne informationer omkring objekternes position og orientering på banen. Billedserver interfacet skal således lytte efter disse UDP pakker, opsamle dem når de ankommer, samt præsentere dem for den egentlige analysedel af klienten, som benytter billeddataene til at beregne robottens handling.

Ud fra kravspecifikationen afsnit 3 er det som nævnt bestemt, at der som minimum kan forventes 10 UDP pakker pr. sekund. Idet det præcise tidspunkt for afsendelse af billeddata er ukendt for klienten, skal den, via billedserver interfacet, være klar til at modtage data hele tiden.

Endeligt skal billedserver interfacet, ligesom robot interfacet, ikke introducere forstyrrelser, som påvirker andre processer.

6.2 Valg af løsningsmodel

I dette afsnit beskrives den nærmere implementation af de enkelte delsystemer, på baggrund af de beskrivelser og krav foretaget i foregående afsnit.

6.2.1 Klienten overordnet

Til opfyldelse af de krav som blev stillet afsnit 6.1.1, vælges klient hardwaren til at bestå af en Intel baseret PC indeholdende serielport samt netkort. Operativsystemet vælges endvidere ud fra de ovennævnte krav til at være Linux 2.4.3, idet dette operativsystem opfylder kravene på alle punkter. Nærmere bestemt introducerer operativsystemet følgende muligheder:

- Mulighed for parallelle processer.
- Understøtter Interproces Kommunikation (IPC) i form af pipes, sockets, fælles memory, semaforer samt beskedkøer.
- Terminal håndtering som følger POSIX standarden.
- Understøtter direkte TCP, UDP og IPv4 samt andre netværksprotokoller.
- Uniform håndtering af I/O, idet dette altid sker igennem læse/skrive operationer på filer.

[Rémy Card, 1997]

6.2.2 Robot interface

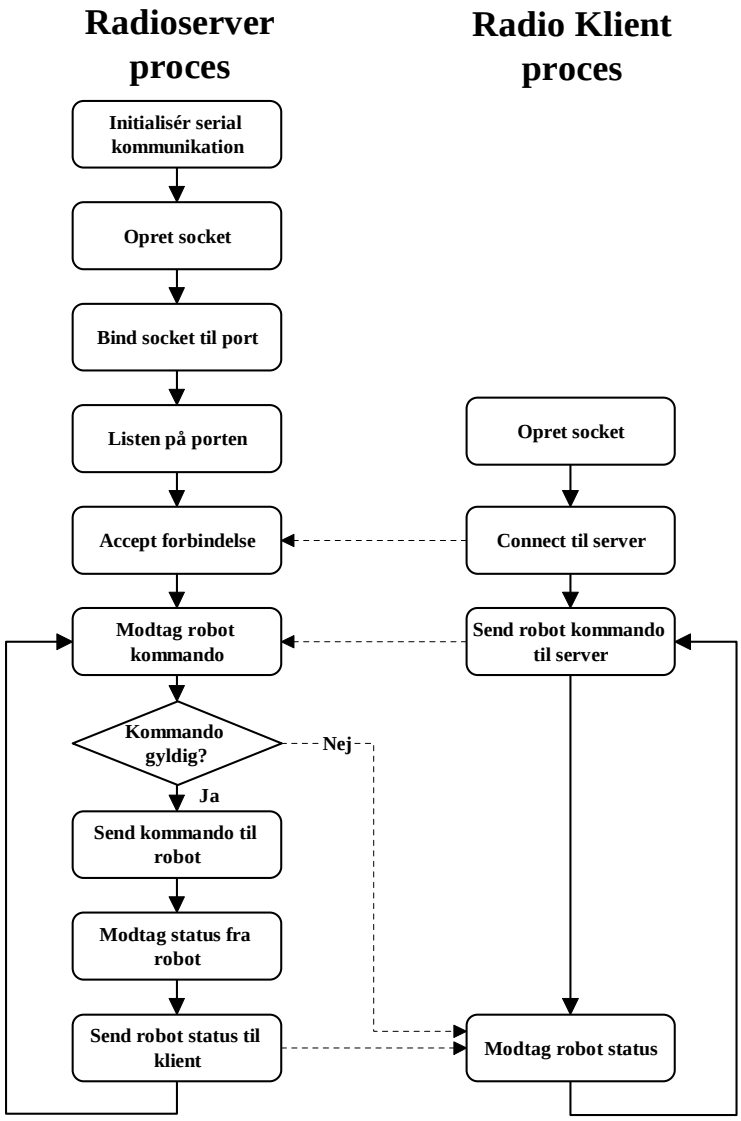
Interfacet mod robotten, der håndterer kommunikationen, er valgt implementeret som en selvstændig proces, der via en bidirektional streamsocket forbindelse kan kommunikere med andre processer. Mere specifikt er denne selvstændige proces implementeret som en dedikeret server (se afsnit E.5), der muliggør, at processer lokalt på samme maskine eller processer, der kører på andre maskiner i samme netværk, kan forbinde sig og styre kommunikationen med robotten. Hovedformålet med denne opbygning er at indkapsle selve opsætningen af kommunikationshardware, håndtering af timing o.l. fra processen, som ønsker at kommunikere med robotten og derved skabe et interface, som umiddelbart er muligt at benytte for andre processer.

Opsætning af serielport. Inden robot interface processen tilbyder at servicere andre processer, initialiseres hardwaren, som benyttes til selve kommunikationen. Dvs. baudrate, data format og andre transmissionsspecifikke indstillinger opsættes for serielporten. Se endvidere afsnit D.1.4 for yderligere information.

Opsætning af streamsocket forbindelse. Efterfølgende påbegynder processen initialisering af den streamsocket forbindelse, der skal benyttes til kommunikation med den anden proces. Første trin er at oprette selve socket file descriptoren, hvorefter der bindes en port til denne file descriptor. Endelig påbegynder processen en uendelig løkke, der fortsætter indtil en klient i form af en anden proces connecter til serveren. (Se endvidere afsnit F.2)

Håndtering af input fra anden proces. Efter kommunikationslinien er etableret kan processen påbegynde udveksling af kommandoer. Indledningsvis venter serveren på, at klienten i form af en anden proces, afsender en kommando (2 bytes), som den ønsker sendt til robotten. Serveren analyserer kommandoen, og sender den til robotten i så fald, der er tale om en gyldig kommando. Efterfølgende venter server processen på svar fra robotten, som når det modtages, sendes videre til klient processen. Det er i rutinen indbygget en timeoutfunktion, således det samlede system ikke går i uendelig løkke, i tilfælde af manglende svar fra radioen.

Efter gennemløbet løkke, startes forfra og serveren venter på en ny kommando fra klienten.



Figur 6.4: Programflow i radioserveren. Til venstre er radioserveren vist, mens højre side viser processen, som benytter radioserveren. Stiplede linier indikerer data, mens fuldtoptrukne linier viser programflow.

6.2.3 Billedserver interface

Formålet med billedserver interfacet er at opsamle alle informationer, som sendes fra billedserveren og videresende disse til de ovenliggende klient lag, der håndterer analyse af de indkommende billeddata.

Idet billeddataene ankommer uregelmæssigt, er det hensigtsmæssigt, at en proces bliver interruptet, når der ankommer nye data. Det er derfor valgt, at implementere billedserver interfacet som en selvstændig proces, hvis eneste formål er at lytte efter billeddata, opsamle disse når de ankommer og sørge for videredistribution. Når processen modtager interruptet hidrørende, at der er nye data klar til læsning, flyttes disse nye data til en buffer (beskedkø), hvorefter processen genoptager lytningen efter billeddata. Den omtalte buffer kan da aflæses af de ovenliggende klient processer, når disse måtte ønske det.

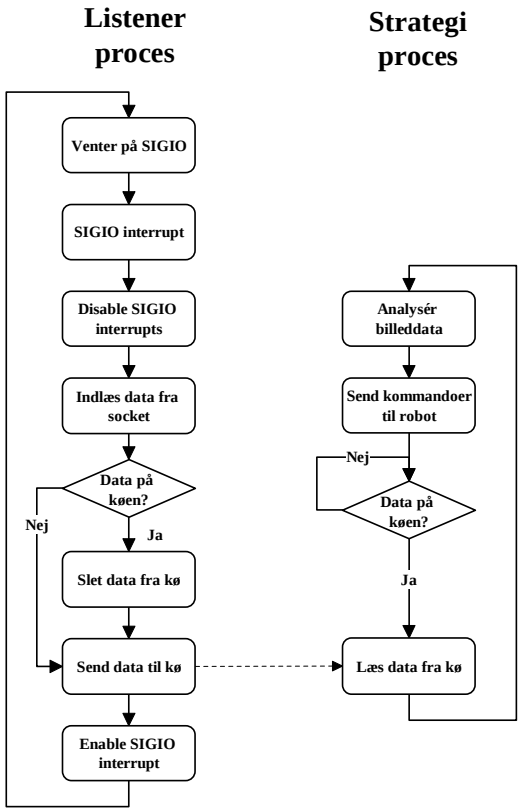
Ovenstående procesbeskrivelse kræver implementering af tre forskellige hovedområder, der omtales i det følgende.

Opsætning af netværksforbindelse. I stil med robot interface delsystemet, skal der også for billedserver interfacet opsættes og initialiseres en socket, hvorpå der kan modtages data fra billedserveren. Det sker på tilsvarende måde som ved radioserveren, blot er den socket, der oprettes af typen datagram (imodsætning til stream) socket, idet kommunikation mellem billedserveren og klienten foregår via transportprotokollen UDP.

Opsætning af interrupt og interrupt handler. Efterfølgende skal processen lytte på den oprettede socket, mhb. på at opsamle de data, der måtte være. Dette kan enten ske ved kontinuert at polle på den oprettede socket, eller det kan ske ved hjælp af software interrupts. Det sidstnævnte er ønskeligt, idet der benyttes langt mindre CPU ressourcer ved denne metode samtidig med, at der er mulighed for, at processen kan lave andet, sålænge den ikke læser på socket descriptoren. (se endvidere afsnit F.2.1)

Opsætning af beskedkø. Når der modtages data fra billedserveren, skal dette hurtigst muligt flyttes til en buffer, som andre processer kan læse fra, hvorefter der igen kan lyttes efter nye billeddata. I det der er tale om adskilte parallelle processer, der skal udveksle disse informationer, er det nødvendigt at benytte et af de midler, som operativsystemet stiller til rådighed i forbindelse med interproces kommunikation. Disse forskellige metoder er beskrevet nærmere i afsnit F.2. Til implementation af bufferen med billeddata er der valgt at benytte en beskedkø, idet denne udover at være meget fleksibel mht., hvilke data der kan udveksles, også besidder en række midler til synkronisering mellem de involverede processer.

Princippet, der er valgt implementeret, bygger på, at strategi delen i klienten ikke kan foretage sig noget, medmindre den har billeddata at arbejde ud fra. Dvs. at strategi delen af klienten venter på, der sendes data til køen, hvis denne er tom. Omvendt sørger processen, der lytter på data fra billedserveren, også for, at der hele tiden kun ligger én besked (den nyeste) på køen. Derved opnås en synkronisering mellem de to processer.



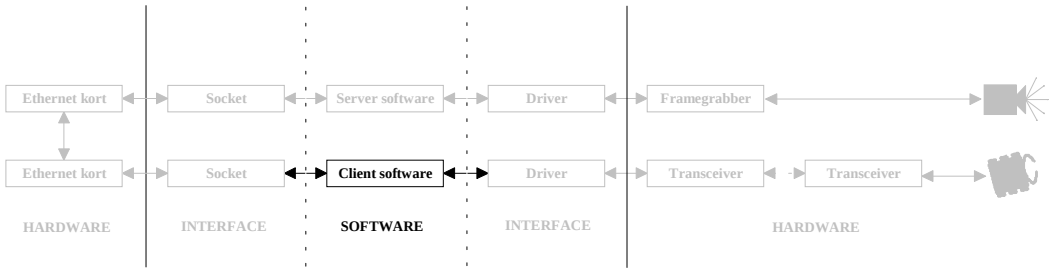
Figur 6.5: Program flowet i billedserver interfacet. Til venstre er vist listener processen og til højre er vist processen, som analyserer de indkommende billeddata. Den stiplede linie er dataflow.

6.3 Delkonklusion

I det ovenstående er design og implementation beskrevet for de dele af klient programmet, som tager sig af de kommunikationsmæssige opgaver i systemet i form af håndtering af data fra billedserveren og data fra og til robotten. Håndtering af forskellige samtidige I/O processer, der skal samarbejde uden at forstyrre hinanden, er problematisk. Ved at anvende parallelle processer, som kommunikerer vha. beskedkøer og bidirektionale sockets er der implementeret interfaces til de ovenliggende klient lag. Delsystemerne opfylder de tidsmæssige krav, der er opstillet, idet systemerne muliggører 10 reguleringer af robotten pr. sekund. Der er således skabt grundlag for det videre design og implementation af klient delsystemerne, som håndterer selve analysen af billeddata samt bestemmelse af robottens strategi.

7 Strategi i klient

Formålet med dette kapitel er at beskrive analyse, design og implementation af de dele af klienten, som står for den egentlige styring og kontrol af systemet. Dette sker indledningsvis ved en analyse af de opstillede krav til systemet, hvorefter det egentlige system designes og implementeres.



7.1 Definition af grænseflader

Klientens opgave er overordnet at analysere de billeddata, som indsamles fra billedserveren via billedserver interfacet (beskrevet i afsnit 6.2.3), beregne en handling for robotten og efterfølgende sende kommandoen til robotten via robot interfacet (beskrevet i afsnit 6.2.2). Endvidere har denne del af klienten en grænseflade imod GUI processen, som benyttes til visningen af robotterne samt koppernes position på spillefeltet.

7.2 Analyse

I det følgende analyseres kravene til det overordnede styresystem til kontrolleringen af robotten. Dette sker indledningsvis ved en gennemgang af de krav, som er stillet til klienten i afsnit 6.1, samt en række krav til selve styringsrutinerne i klienten.

Formål. Formålet med klienten, er at styre robottens færden på spillefeltet. Den skal kunne fortolke informationerne givet omkring udviklingen på spillefeltet og efterfølgende give robotten den bedst mulige instruktion, således den ønskede handling udføres.

Systemdefinition. Der ønskes et system, hvorom følgende gælder

- Betingelser. Klientens styring af robotten skal foregå autonomt. Dog skal det være muligt at overvåge systemets afvikling.
- Teknologi. I foregående kapitel er det valgt at benytte en standard Intel pc med Linux operativ system, serielport samt netkort. Styringssoftware skal følgelig udvikles på denne platform.
- Funktionalitet. Klienten skal fungere som den kontrollerende enhed i styringen af robotten og være det sammenkoblende link mellem server og robot.

Anvendelsesområde For at få et overblik over systemet og funktionaliteten af dette, vil problemområdet i det følgende blive analyseret. Der foretages en foranalyse, i form af en brainstorm på krav til klient softwaren. Denne er efterfølgende blevet gennemarbejdet, hvorefter den egentlige kravspecifikation er foretaget.

Krav til klient software. Følgende er en opsummering af de krav, som stilles til styringssoftwaren i klienten.

- Opsamling af data. Der skal mellem hver regulering af robotten, opsamles data fra billedserver interfacet omkring spillefeltet status.
- Analyse af robotternes bevægelse. Robotternes retning, hastighed og position skal bestemmes ud fra de opsamlede billeddata. Herunder skal modpartens formodede mål og vej til dette forudsiges.
- Prioritering af mål. Prioritering af destinationer skal foretages ud fra robotternes vinkel og afstand til objekterne, samt antallet af forhindringer på vejen. Hvis et mål formodes at være modstanderens destination, skal dette opprioriteres. Modsat skal det nedprioriteres såfremt det er analyseret, at modstanderen vil det først.
- Bestemmelse af vejen til målet. Vejen til destination skal vælges ud fra hensyn til at undgå kollision med modpart og kopper. Endeligt skal det være muligt at udarbejde waypoints, som robotten følger til målet.
- Regulering af robot. Via robot interfacet skal der kunne gives en kommando til robotten, således den kører efter den valgte destination.

7.3 Strategi

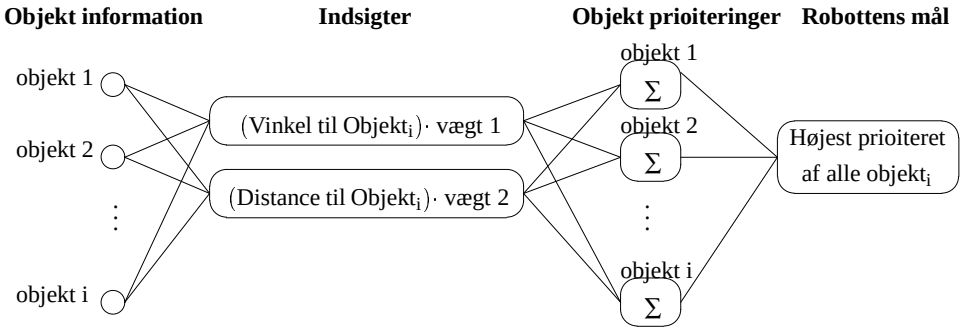
Ud fra de opstillede krav kan der drages den konklusion, at en bestemmelse af robottens handlinger på spillefeltet er kompliceret og den afhænger af en række forskellige beslutningsprocesser med hver deres indsigt.

For at simplificere den samlede løsningsmodel, vil det være oplagt at give de individuelle beslutningsprocesser en vægtning efter deres betydning for den samlede bestemmelse af robottens strategi.

Bestemmes robottens strategi efter denne metode, vil der kunne opstilles et minimumsystem, hvor kompleksiteten og intelligensen af den samlede beslutningsproces kan forøges eller mindskes, alt efter hvor mange selvstændige beslutninger der foretages.

I det følgende designes minimumsystemet til klienten, der skal fungere som grundlag for udviklingen af robottens intelligens. Designet er konstrueret således, at det kan udvikles på baggrund af indsamlede erfaringer, hvilket der som udgangspunkt er mangel på, idet robottens præcise adfærdsmønster ikke kendes på forhånd.

Den grundlæggende strategi kan nu opstilles som diagrammet vist på figur 7.1.



Figur 7.1: Strategi for klienten. Diagrammet viser hvorledes informationer opsamlet vedr. en række objekter, vægtes i forhold til deres afstand og vinkel til robotten. De enkelte prioriteter summeres, hvorefter det højeste prioriterede objekt vælges som destination.

Efterfølgende kan der overordnet opstilles en grundlæggende taktik for klienten:

1. Henter informationer om objekternes position fra billedserver interfacet.
2. Robotternes bevægelse analyseres, og informationerne til hver objekt findes.
3. Objekterne prioriteres i henhold til figur 7.1.
4. Det højst prioriterede mål vælges som destination.
5. Ruten til destination fortolkes, og kommandoer til robotten oprettes.
6. Kommandoerne sendes til robot interfacet.

Ovenstående taktik skal således gennemløbes hver gang, der hentes nye informationer fra billedserver interfacet. Informationerne om objekternes placering på banen vil som udgangspunkt blive betragtet som nye, hver gang der hentes billeddata - dog vil robotternes positioner blive lagret, således bestemmelse af hastigheder muliggøres. Hver gang der modtages nye billedata, foretages nye beregninger på objekterne, hvorved prioriteterne opdateres.

Designet af minimumssystemet udvikles således, at systemet baseres på oprettelse af ruter. Dette vil som udgangpunkt kunne gøres ved at oprette waypoints som objekter på spillefeltet, og prioritere disse højest. Dette vil være fordelagtigt i forbindelse med undvigelse af modpartens robot.

7.4 Design

Strategien i opgaven kan nu objektorienteres. Der vil blive oprettet grundlæggende klasser, der kan benyttes som udgangspunkter i den videre udvikling af programmet.

7.4.1 Objekt klynge

Med udgangspunkt i design af kontrolsystemet, vil der kunne opstilles et virtuelt spillefelt, der er en kopi af det virkelige spillefelt. Grundidéen i designet er, at alle objekter på spillefeltet har et (x,y) koordinatsæt, hvorfor der kan oprettes en klasse ved navn *object*, der har tilknyttet denne egenskab. Med udgangspunkt i den ønskede prioriteringsfunktionalitet tilknyttes hvert *object* endvidere en prioritering. De forskellige objekter på spilletfeltet kan efterfølgende nedarves af klassen *object*, hvorved disse tildeles prioritet og koordinatsæt.

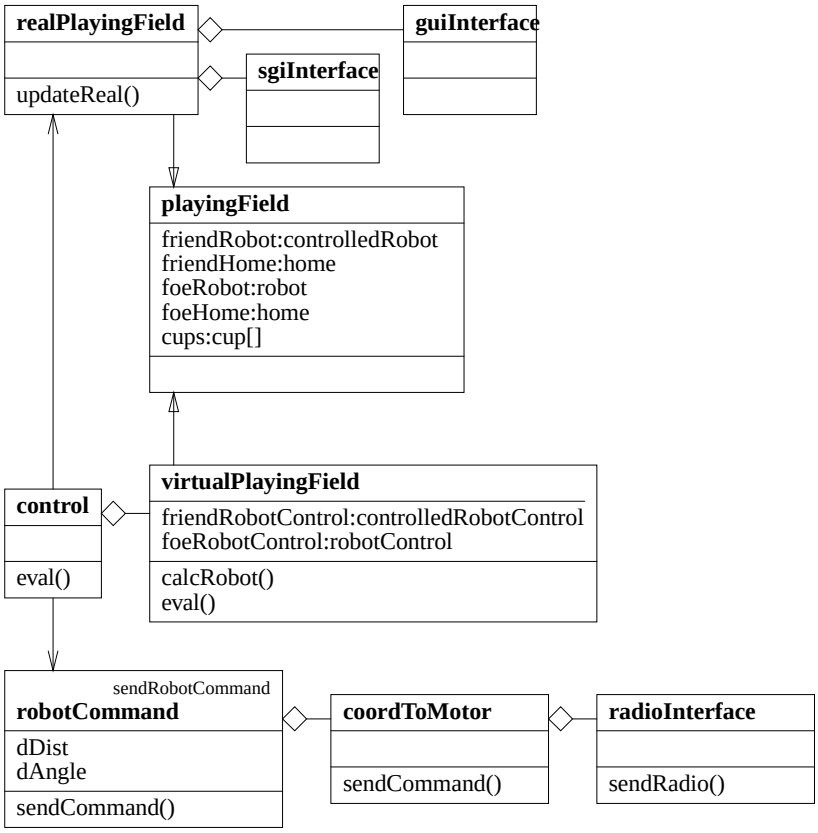
Idet objekterne er blevet klassificeret af billedserveren, kan de derfor umiddelbart oprettes på det virtuelle spillefelt med hver deres type og egenskaber. Efterfølgende er en analyse af objekterne på det virtuelle spillefelt muligt.

Dernæst oprettes et egentligt klassediagram, hvor de forskellige typer af objekter kan defineres i form af deres egenskaber og metoder, hvilket er vist ved figur 7.2. Hovedtrækkene i klassestrukturen vil i det følgende gennemgås.

Klassen *robot* er her tilknyttet en destination (**heading*), en hastighed (*velocity*) og en vinkel i planet (*angle*). Robottens destination er en pointer til den instans af *object*, som er robottens mål. Idet robotten samt dens destination er af samme type, vil der kunne beregnes en række informationer vedrørende robottens forhold til destination (eks. afstand, vinkel osv.). Dette foretages i *robotControl*, hvor robottens prioritering af det pågældende *object* ligeledes beregnes.

Et særtilfælde af klassen *robot*, er klassen *controlledRobot*, der tilføjer en række funktionaliteter til *robot* klassen, således den egentlige styring (afsendelse af kommandoer til radioserveren) af robotten muliggøres. Håndteringen af dette foretages ved, at en pointer til

den ønskede destination videresendes fra *controlledRobot* klassen til *sendRobotCommand* klassen.



Figur 7.2: Objekternes klassestruktur.

Disse klasser bliver nu samlet i en klynge, hvorefter deres struktur kan oprettes som objekter i andre klasser.

7.4.2 PlayingField

En af de centrale klasser i opbygningen af klienten softwaren er *playingField*, hvor der oprettes en række objekter, som beskrives i det følgende:

foeRobot. Klassen *foeRobot* er af typen *robot*. Denne klasse indeholder informationer om modpartens placering på spillefeltet og er i stand til at analysere modpartens bevægelse.

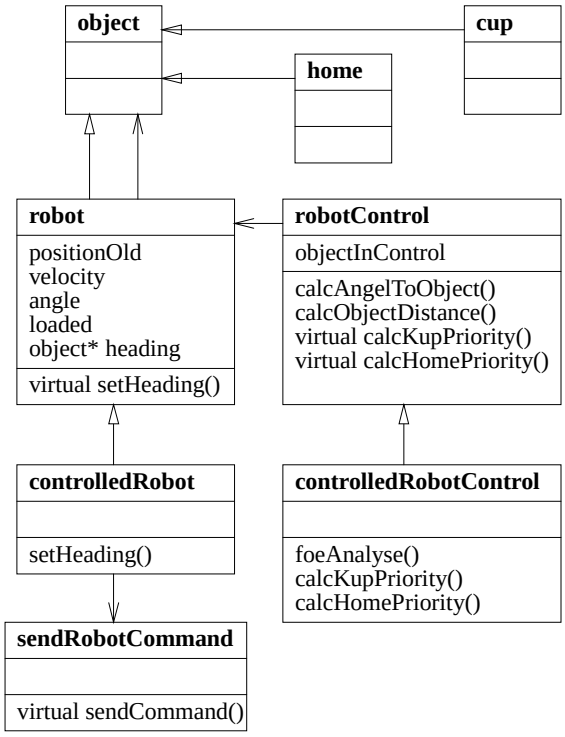
friendRobot. Klassen *friendRobot* er af typen *controlledRobot*. Denne klasse indeholder information om placeringen af egen robot, der skal styres. Klassen er i stand til at

analysere, hvilken af kopperne der skal prioriteres højest, samt videregive informationerne således robotten udfører de ønskede handlinger.

friendHome. Klassen *friendHome* er af typen *home*. Dens formål er at udpege hjemmebasen for *friendRobot*. Denne klasse vil blive oprettet ved initialiseringen af spillefeltet og betragtes herefter som værende et statisk punkt på spillefeltet. Det vil dog være muligt at flytte punktet, således robotten ikke placerer kopperne på samme sted.

foeHome. Klassen *foeHome* har samme attributer som *friendHome*.

cups. Klassen *cups* er et array af typen *cup*. Dettee bliver opdateret på ny hver gang der modtages nye billeddata fra billedserver interfacet, som beskrevet i afsnit 7.3.



Figur 7.3: Klassestruktur på klienten.

7.4.3 Klyngestruktur

De resterende klasser i diagrammet (figur 7.3) er alle relaterede til den grundlæggende klasse *playingField*, beskrevet i afsnit 7.4.2. I det følgende beskrives disse klasser.

control. Denne klasse står for den egentlige afvikling og styring af klienten.

realPlayingField. Denne klasse har til opgave at håndtere de informationer som kommer fra billedserveren (via klassen *sgi interface*), samt at opdatere GUIen (via klassen *gui interface*). Klassen sørger overordnet for, at klassen *control* har opdaterede informationer omkring spilletfeltet.

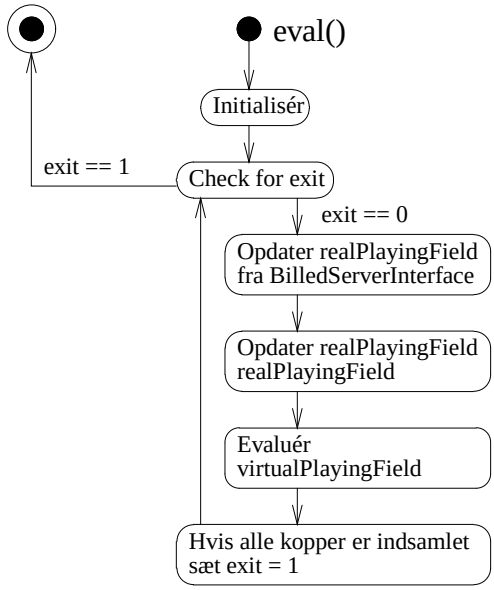
virtualPlayingField. Denne klasses opgave er at kontrollere de objekter, som befinder sig på spilletfeltet. Det vil sige prioritering af de forskellige objekter.

robotCommand. Denne klasse sørger for kommunikationen med robotten. Dette sker via klassen *coordsToMotor*, som omregner en destination til de egentlige kommandoer, der sendes til robotten. Selve afsendingen foregår i klassen *radio interface*.

I det følgende beskrives disse klasser mere indgående.

7.4.4 Control

Denne klasse sørger for afvikling og kontrol af det kørende system på klienten. Aggregeret til klassen *control*, er klassen *virtualPlayingField*, som står for selve kontrollen af objekterne på spillefeltet. Klassen control er den styrende klasse for klienten. Evalueringen af control klassen er vist på figur 7.4.



Figur 7.4: Evalueringen af *control* klassen.

Der startes en løkke, der bliver eksekveret sålænge exit er forskellig fra 1.

- Først vil funktionen, der opdaterer *realPlayingfield* blive eksekveret. Efterfølgende indhentes der data fra billedserver interfacet. Dataformatet er som beskrevet i afsnit

4.13 givet ved et array, hvorfor disse objektorienteres i den henseende, at der oprettes objekter af forskellige type i *realPlayingfield*.

- Efterfølgende overføres de opdaterede billeddata i *realPlayingfield* til *virtualPlayingfield*. Dette sker ved hjælp af polymorfisme, idet begge de nævnte klasser er nedarvet fra samme grundklasse.

- Efterfølgende bliver det virtuelle spillefelt (*virtualPlayingfield*) evalueret.

7.4.5 realPlayingField

Denne klasse er nedarvet af *playingField* klassen og sørger for kommunikation med billedserveren. Den har overordnet til opgave at tage sig af ude- og indefra kommende informationer, der har tilknytning til *control* klassen. Her vil informationerne om objekterne, der er blevet klassificeret af serveren, blive objektorienteret. Klassen skal selv sørge for tilgangen til disse informationer. Ligeledes skal klassen også selv stå for kommunikation med GUIen.

Til klassen *realPlayingfield* er der aggregeret interfacet *sgiInterface*. Opgaven for dette interface er at hente informationer fra billedserveren. Selve opdateringen af spillefeltet, bliver håndteret af klassen selv i funktionen *updateReal()*. Her fortolker klassen informationerne fra *sgiInterfacet*, og objektorienterer dem ud fra de givne objekter i klassen *playingField*. Informationerne kan efterfølgende sendes videre til GUIen ved hjælp af funktionen *updateGui* i interfacet *guiInterface*.

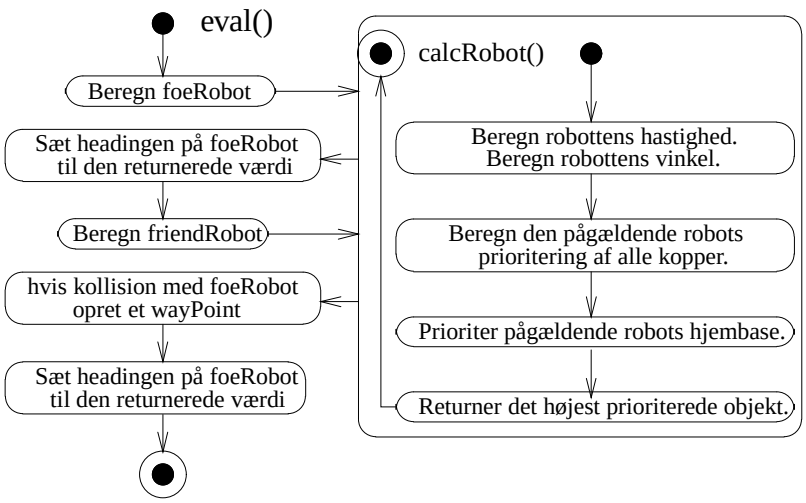
7.4.6 virtualPlayingfield

Klassen *virtualPlayingfield* har til opgave at analysere de modtagne data fra *realPlayingfield*, med henblik på prioritering af de givne destinationer og efterfølgende foretage et endeligt valg. Til dette formål er der blandt andet oprettet følgende underklasser i *virtualPlayingfield*.

foeRobotControl. Klassen *foeRobotControl* er af typen *robotControl*. Denne klasse sørger for analyse af modstander robotten bevægelser samt prioritering i forhold til *foeRobot*.

friendRobotControl. Klassen *friendRobotControl* er af typen *controlledRobotControl*. Denne klasse sørger for analyse, og beregning af objekternes prioitet, i forhold til *friendRobot*.

Evalueringen af det virtuelle spilletfelt (*virtualPlayingfield*) er vist på figur 7.5.



Figur 7.5: Evalueringen af det virtuelle spillefelt (*virtualPlayingfield*).

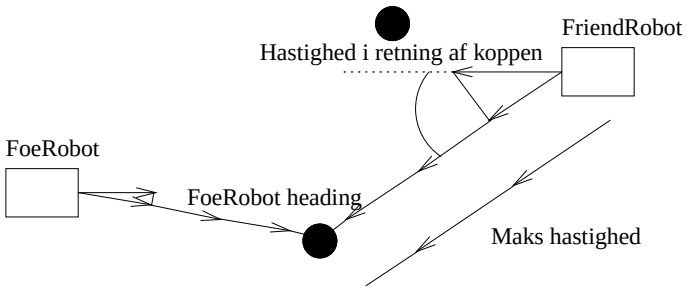
Her bliver prioriteringerne for de to robotter udregnet ved kald af funktionen *calcRobot*. Dette bliver gjort med en pointer til den pågældende robots *control* klasse og dens *home* klasse. Prioriteringen for robotterne kan beregnes forskelligt, alt efter om der er tale om en pointer til et *object* af typen *robotControl* (*foeRobot*) eller *controlledRobotControl* (*friendRobot*). Ligeledes vil kald af funktionen *setHeading*, kunne håndteres forskelligt, alt efter hvilken robot der er tale om.

- I beregningen af robotten, vil robottens pågældende vinkel og hastighed først blive beregnet.
- Ud fra dette vil prioriteten for hver kop blive beregnet, hvorefter den højeste prioriterede udvælges.
- Efterfølgende bestemmes prioriteten af den pågældende robots hjemmebase (*home*). Denne bliver sat højt, hvis robotten har indsamlet en kop.
- Dernæst returneres en pointer til det objekt (hjemmebase eller kop), som er blevet prioriteret højest.
- Ved beregning af *friendRobot* kaldes funktionen *checkCollisionFoe*, hvor det undersøges, om friendRobot er på kollisionskurs.
- Er dette tilfældet, kaldes funktionen *makeWayPoint*, der returnerer en pointer til et waypoint objekt, der overskriver det højst prioriterede objekt.
- Den returnerede pointer, vil nu blive sat som den pågældende robots destination.

Der bliver som udgangspunkt ikke taget højde for, at støde ind i kopper, som ikke er robottens destination. Dette vil kun kunne ske, hvis 2 kopper er placeret tæt, eller *friendRobot* er på vej til hjemmebasen. Skulle det ske, at en kop vælter, er robotten designet, således den stadig har mulighed for at opsamle denne.

Prioritering. Ved prioritering af kopperne, beregnes en række indsigter, der kan fortælle, hvor egnet en kop vil være som mål. Disse indsigter vil som udgangspunkt være afstanden til koppen, og vinklen robotten skal dreje, for at køre mod koppen. For at kunne sammenligne de 2 indsigter, vil de blive skaleret i forhold til deres maksimale værdi. Prioriteringen af koppen vil efterfølgende være en sum, af resultatet af de 2 indsigter, multipliceret med en vægtninig af indsigtens betydning. Dette er vist på figur 7.1, i afsnit 7.3.

Den omtalte prioriteringsberegning vil blive foretaget for *foeRobot*. For *friendRobot* opstilles endnu en indsigt, som medtages i beregningen af prioriteterne. Den tilføjede indsigt vægter muligheden for, hvorvidt robotten kan nå et objekt før modstanderrobotten. Er dette muligt, vil den pågældende kop blive opprioriteret. Er det ikke muligt, nedprioriteres denne kop. Dette scenarie er vist på figur 7.6.



Figur 7.6: Prioriterings indsigter, til gene for foeRobot.

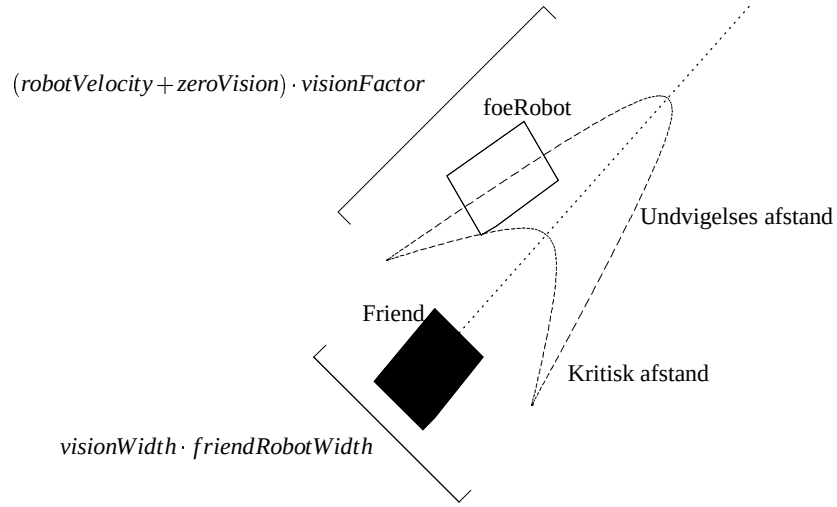
Først findes ud af, om det er muligt at nå koppen først. Dette vil gøres ved at splitte indsigten op i 2 dele.

Indsigt 1
Sammenligning af hastigheder og afstande for de to robotter.

Indsigt 2
Sammenligning af hastigheder og afstande for de to robotter, hvis *friendRobot* accelereres til maksimal hastighed.

Disse to indsigter er hver blevet vægtet ud fra en betragtning af hvor hurtigt, der kan skiftes bevægelses retning. Heraf bestemmes prioriteten, og den summeres med den først beregnede.

checkCollision. Denne funktions formål er at undersøge, om *friendRobot* er på kollisionskurs med *foeRobot*. Der defineres et synsområde ud for *friendRobot*, ved hjælp af formlen for en ellipse. Dette er illustreret ved figur 7.7.



Figur 7.7: Undersøgelse af om *foeRobot*, er på kollisions kurs.

Det undersøges om destinationen for *friendRobot* er inden for synsfeltet. Er dette tilfældet, undersøges om *foeRobot* er indenfor synsfeltet, idet målet skal kunne nås, uden gener for oprettelsen af wayPoints. Er dette ikke tilfældet, og *foeRobot* kan ses, indikeres det endvidere til *makeWayPoint*, om *foeRobot* er inden for den kritiske afstand. Dette princip vil være følgende:

x og y koordinaterne til *foeRobot* transformeres således de beskrives ud fra et koordinatsystem, der har *friendRobot* i centrum og er orienteret således retningen for *friendRobot* er parallel med x aksen. Der sker vha. følgende formler.

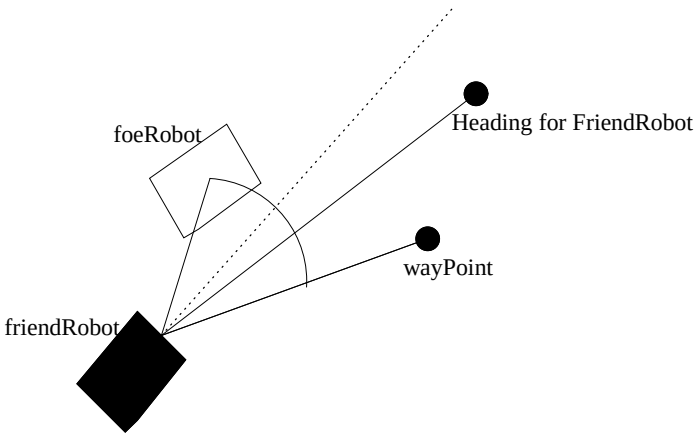
$$\begin{aligned} foeRobot_{NewXPos} &= (foeRobot_{XPos} - friendRobot_{XPos}) \cdot cos(frinedRobot_{angle}) \\ &\quad + (foeRobot_{YPos} - friendRobot_{YPos}) \cdot sin(frinedRobot_{angle}) \\ foeRobot_{NewYPos} &= (foeRobot_{YPos} - friendRobot_{YPos}) \cdot cos(frinedRobot_{angle}) \\ &\quad - (foeRobot_{XPos} - friendRobot_{XPos}) \cdot sin(frinedRobot_{angle}) \end{aligned}$$

Det undersøges efterfølgende om *foeRobot* er inden for den halve ellipse udspændt fra friendRobot, som det er illustreret i figur 7.7. Dette vil være sandt hvis $foeRobot_{NewXPos}$ er større end 0 og uligheden vist ved ligning 7.1 er opfyldt.

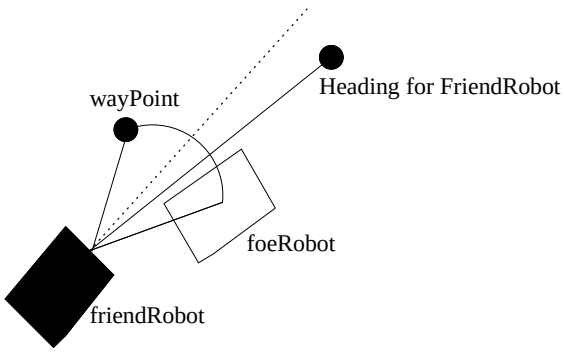
$$\frac{foeRobot^2_{NewXPos}}{((friendRobot_{absVelocity} + zeroVision) \cdot visonFactor)^2} + \frac{foeRobot^2_{NewyPos}}{(visionwidth \cdot width)^2} < 1 \quad (7.1)$$

Ellipsens stor akse, der ligger i kørselsretningen for *friendRobot*, bliver udspændt af robottens absolutte hastighed, adderet med et udtryk for dens ønskede syn, når den holder stille *zeroVision*. Dette multipliceres med en faktor for robottens fremadrettede syn *visionFactor*. Ellipsens lille akse, bliver udspændt af robottens bredde, multipliceret med en faktor for robottens synsvidde *visionwidth*.

makeWayPoint. Denne funktion opretter et waypoint til *friendRobot*, uden om *foeRobot*, så en kollision ikke opstår, men målet for *friendRobot* stadig nås. Der vil derfor altid prøves at holde den numeriske værdi, af vinklen til destinationen for *friendRobot*, mindre end den numeriske værdi af vinklen til *foeRobot*. Denne situation er illustreret på figurene 7.8 og 7.9.



Figur 7.8: Strategi ved undvigelse af *foeRobot*.



Figur 7.9: Strategi ved undvigelse af *foeRobot*, hvor situationen er kritisk.

Waypoints vil nu oprettes som følger:

- Vinklen til waypointet findes.
  - Er vinklen til destinationen for *friendRobot* mindre end vinklen til *foeRobot* skal den gerne forblive sådan. Vinklen bliver derfor vinklen for *foeRobot* minus en given afvigelsesvinkel. Dette er vist på figur 7.8.
  - Er vinklen til destinationen for *friendRobot* større end vinklen til *foeRobot* skulle den ligeledes gerne forblive sådan. Vinklen bliver derfor vinklen for *foeRobot* plus en given afvigelsesvinkel. Dette er vist på figur 7.9.
- Afstanden til waypointet findes.
  - Er det indikeret, at *foeRobot* befinder sig inden for den kritiske afstand fra *friendRobot*, vil afstanden til way pointet være afstanden til *foeRobot*, som det er tilfældet på figur 7.8. Her vil wayPointet ligge tæt på *friendRobot*, så robotten sænker hastigheden.
  - Er dette ikke tilfældet, vil afstanden til waypointet være afstanden til *foeRobot*, plus en ekstra afstand, så robotten ikke sænker hastigheden.
- Waypointet oprettes nu som et objekt og dens koordinater beregnes.

7.4.7 robotCommand

Denne klasse sørger for kommunikation med robotten. Til klassen *robotCommand* er aggregeret klassen *coordsToMotor*, der opretter kommandoen til robotten og sender dem til interfacet *radioInterface*. Her bliver kommandoerne sendt til robotten, og status vil blive modtaget.

Hvis objektet som kontrolleres, er af typen *controlledRobot*, vil dens *heading* også blive evalueret. Dette bliver gjort i klassen *controlledRobot*, ved et kald til funktioen *sendCommand(*heading)* i klassen *sendRobotCommand*.

Klassen *robotCommand*, er nedarvet af klassen *sendRobotCommand*. I funktionen *sendCommand(*heading)* vil distancen og vinklen til objektet blive beregnet, ved hjælp af funktionerne i *robotControl*. Efterfølgende vil klassen *coordsToMotor*, blive evalueret, med et kald af dens funktion *robotLoopControl(dAngel,dDist)*, med parametrene der før blev udregnet. Funktionaliteten i klassen *coordsToMotor*, vil blive yderligere beskrevet i kapitel 8.

7.5 Delkonklusion

I dette kapitel er beskrevet opbygningen af strategi delen af klienten softwaren. Styresystemets grundlæggende funktion består opsamling af billedata, der efterfølgende er blevet objektorienteret og analyseret på et virtuelt spillefelt. I systemet er der indbygget prioriteringsrutiner, som vægter robottens destinationer efter afstand og vinkel, samtidig med, at der er indbygget rutiner til undvigelse af modstanderrobotten. Systemet opfylder de opstillede krav, og der er endvidere gjort mulighed for senere udvidelser af systemet.

Styring af robot

8

Dette afsnit har til formål at dokumentere, hvordan robotten på grundlag af beregningerne i klienten styres hen til koppen med den højeste prioritet, indsamles og føres tilbage til homebasen. Indledningsvis beskrives mekanikken bag robottens bevægelsesmønster indbygget i styringssoftwarens feedback algoritme i klienten.

8.1 Definition af grænseflader

Ud fra billeddata beregnes vinklen mellem robottens hastighedsvektor og den prioriterede kop samt afstanden mellem robotten og koppen. Sammen med en statusparameter, der efter hver transmission modtages fra robotten, benyttes disse variable som input til en styringsalgoritme, der bestemmer robottens bevægelsesmønster. Som output beregnes en opkode, der sendes til robotten. Mængden af opkoder er på forhånd valgt ud fra ønsket om en simpel og fleksibel styring. Disse er beskrevet i afsnit 5.1.2.

Styringsalgoritmen er i henhold til den ønskede funktionalitet for robottens styremekanisme underlagt følgende overordnede krav:

Prioritering af kopper. Samtlige kopper opfattes som objekter med en prioritet. Der skal altid køres mod punktet med den højeste prioritet, der også kan være et waypoint, der specificerer ruten til den prioriterede kop.

Feedback af styring. Når robotten bevæger sig mod det højest prioriterede objekt, skal styringsalgoritmen kompensere for forsinkelse ved transmission af data gennem hele systemet. Hver gang input til styringsalgoritmen opdateres, ønskes det at regulere robottens retning og hastighed, så der opnås så præcis en styring som muligt. Reguleringen skal være progressiv, så den øges i takt med størrelsen af robottens afvigelse fra den ønskede retning og skal endvidere tage højde for de foruddefinerede accelerations- og decelerationsprofiler for motorerne.

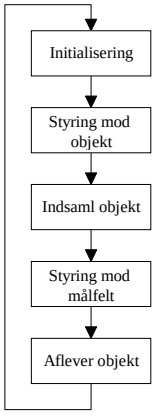
Status for robotten. For at sikre robotten har indsamlet den prioriterede kop, skal styringssoftwaren kontrollere den tilbagesendte statusparameter fra robotten. Ved udeblivende

svar eller ved fejl under transmission kan det ud fra statusparametren evt. vælges at sende opkoden, indtil der modtages et svar fra robotten, før næste opkode beregnes.

Spilleregler. Styringsalgoritmen skal overholde de opsatte spilleregler for indsamlingen af kopper vedtaget i fællesskab af de to opponerende grupper.

8.2 Valg af løsningsmodel

Der er i udviklingen af styringsalgoritmen taget udgangspunkt i forløbet for indsamling af et objekt:



Figur 8.1: Forløb for indsamling af et objekt.

Initialisering. Hver gang indsamlingen af en kop påbegyndes, skal robotten initialiseres. Grabberen skal aktiveres, så den automatisk indfanger en kop, når den er tæt nok på robotten. Herefter igangsættes robotten.

Styring mod objekt. Umiddelbart efter igangsættelsen af robotten skal retningen justeres mod det prioriterede objekt med så høj hastighed som muligt. Retningen tilpasses automatisk af en feedback algoritme under bevægelsen mod koppen, så der opnås en høj retningsstabilitet under høj hastighed. Koppen skal rammes så præcist, at den kan indfanges af grabberen, der er styret af robotten.

Indsaml objekt. Tæt ved objektet foretages en nedbremsning, hvorefter de sidste justeringer af retningen foretages, før koppen kan indsamles. Robotten stoppes kort, mens den venter på en ny opkode.

Styring mod målfelt. Efter indsamling af koppen foretages en pivotvending på ca. 180°. Dermed undgås at vælte andre kopper i nærheden af den indsamlede kop under kørslen mod målfeltet. Feedback algoritmen skal igen give den ønskede retningstabilitet.

Aflever objekt. Som ved indsamling af koppen nedbremses kort før målfeltet, hvorefter grabberen åbnes og koppen placeres midt i målfeltet. Når grabberen er helt åbnet, bakker robotten tilbage i et specificeret tidsrum, hvorefer algoritmen påbegyndes forfra, såfremt der er flere kopper tilbage.

8.3 Implementation af feedbackregulering

Det er ønsket, at feedback algoritmen ved brug af de i forvejen definerede opkoder kan benyttes under svingmanøvrer samt give retningsstabilitet under kørsel ligeud. Opkoderne giver mulighed for at svinge robotten til højre og venstre med en bestemt forskel i vinkelhastigheden for højre og venstre hjulpar. Ved svingmanøvrer bevæger robotten sig følgelig i cirkelbaner. Ved at lade radius i cirklen bestemme vinklen til objektet, opnås en progressiv styring. Til at kalibrere styringen indføres en tidsfaktor, der angiver svingningstiden for robotten uanset vinklen til objektet. På dette grundlag beregnes opkoden til robotten.

Ved løbende at sende opkoder til robotten beregnet på grundlag af opdaterede billeddata, opnås en regulering, hvor robotten tilbagelægger en ganske kort buelængde i cirkler med varierende radius. Når vinklen til objektet nærmer sig nul, vil robotten svinge med en hastighedsforskel på nul mellem de to hjulpar. Dette svarer til at køre med konstant hastighed.

Der opnås heraf en regulering med en høj opløsning, så robottens bevægelse er kontrolleret og forudsigelig, samtidig med at der køres med så høj en hastighed som muligt.

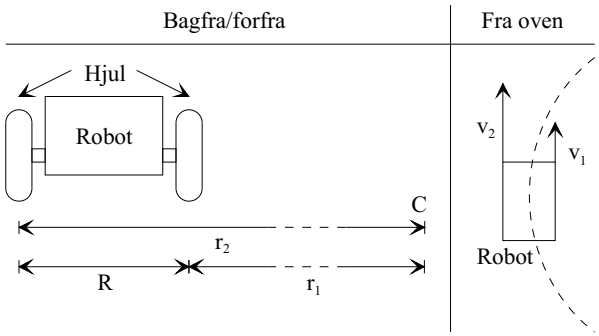
Krav til feedbackregulering. Det samlede system stiller krav til reguleringen. Robottens hardware og firmware specificerer, hvordan motorerne indbyrdes styres, ligesom ændringer i motorhastigheden bestemmes af indbyggede foruddefinerede accelerations- og decelerationsprofiler. Dermed undgås pludselige ændringer af momentet, se afsnit 5.2. Forsinkelse ved datatransmission og databehandling i hele systemet giver endvidere anledning til usikkerheder.

Reguleringen skal tage udgangspunkt i en model, hvor robotten med konstant fart foretager sving langs cirkelstykker. Vinkelhastigheden for robotten skal variere alt efter vinklen til det prioriterede objekt. På denne måde forsøges det for små vinkler at minimere effekten af accelerations- og decelerationstiderne for robotten samt forsinkelserne i systemet.

Tilnærmelser og antagelser. På baggrund af robottens fysiske konstruktion opstilles der i de følgende afsnit en sammenhæng mellem input og output i feedback algoritmen, hvor der ses bort fra accelerations- og decelerationsprofiler og forsinkelser under transmission. Det antages i denne diskussion, at robottens massemidtpunkt befinder sig i centrum af den fysiske konstruktion.

8.3.1 Svingradius for robotten

Dette afsnit har til formål at opstille udtryk for sammenhængen mellem den translatoriske hastighed for hvert af robottens hjulpar, v_1 og v_2 samt robottens ydre svingradius r_2. Grafisk er denne sammenhæng afbildet på figur 8.2:



Figur 8.2: Svingradius for robotten som funktion af hjulenes translatoriske hastigheder.

Det fremgår, at radius r_2 kan skrives

$$r_2 = r_1 + R \tag{8.1}$$

hvor r_1 er robottens indre svingradius, og R er bredden mellem robottens hjulpar, der på forhånd er kendt. r_1 udtrykkes nu vha. hastighederne v_1 og v_2 for hvert af robottens hjulpar. På figur 8.3.1

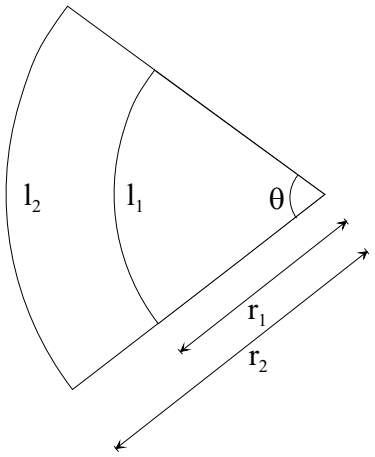
angiver l_1 og l_2 buelængden af en cirkel over en vinkel θ, robotten tilbagelægger under et sving. Forholdet mellem buelængderne l_1 og l_2 er da

$$\frac{l_1}{l_2} = \frac{2\pi r_1}{2\pi(r_1+R)} = \frac{r_1}{r_1+R} = \frac{\frac{v_1}{t_l}}{\frac{v_2}{t_l}} = \frac{v_1}{v_2} \tag{8.2}$$

hvor t_l er tiden, det tager at tilbagelægge strækningerne l_1 og l_2. Ved omskrivning af 8.2 fås for r_1:

$$r_1 = R \cdot \frac{v_1}{v_2 - v_1} \tag{8.3}$$

Indsættelse af 8.3 i 8.1 giver:



Figur 8.3: Buelængdens afhængighed af radius.

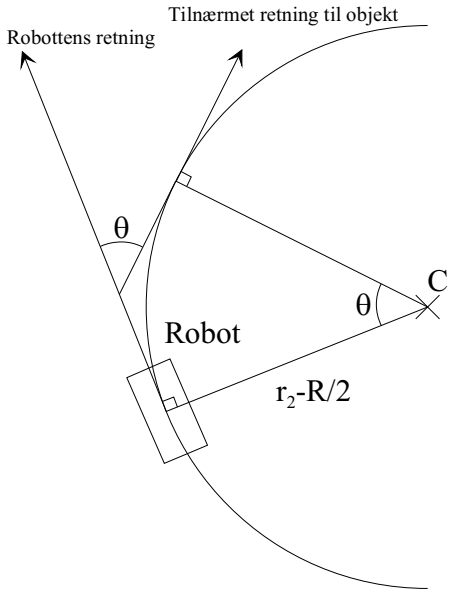
$$r_2 = R \cdot (1 + \frac{v_1}{v_2 - v_1}) \tag{8.4}$$

Ud fra den translatoriske hastighed bestemt af hjulenes rotationshastighed i hver side, er det muligt at beregne robottens svingradius. Det er et krav, at svingradius er mindre end afstanden til det prioriterede objekt, for at robotten kan nå objektet under svingmanøvren.

8.3.2 Hastighedsrelation

Som omtalt i afsnit 8.2 skal robotten under et sving bevæge sig langs cirkelstykker. I dette afsnit udvikles, for robotten, en relation mellem vinkelhastigheden ω og de translatoriske hastigheder v_1 og v_2.

På figur 8.4 ses robotten køre langs en buelængde l over vinklen θ i en cirkel med centrum C:



Figur 8.4: Tilnærmelse af vinklen til et objekt.

Radius i cirklen defineres som afstanden fra cirklens centrum til robottens centrum. Denne afstand er givet ved $r_2 - \frac{R}{2}$.

På figur 8.4 er vinklen θ også indtegnet som vinklen mellem de to tangenter til cirklen. Vinklen til objektet tilnærmes nu vinklen θ. For store værdier af radius $r_2 - \frac{R}{2}$ er tilnærmelsen dårlig, idet tangenternes skæringspunkt er langt fra robottens aktuelle position. Således skal robotten tilbagelægge en større buelængde af cirklen end den indtegnede buelængde, for at den aktuelle vinkel til objektet bliver 0.

Tilnærmelsen er imidlertid god for små værdier af radius $r_2 - \frac{R}{2}$ og gælder eksakt i grænsetilfældet hvor radius $r_2 - \frac{R}{2} \to 0$. Dette svarer følgelig til at robotten drejer om sit eget massemidtpunkt.

Robottens translatoriske hastighed i afstanden $r_2 - \frac{R}{2}$ fås af buelængden l:

$$v_1 + \frac{1}{2}(v_2 - v_1) = \frac{v_1 + v_2}{2} = \frac{l}{t_l} \tag{8.5}$$

t_l er tiden, det tager at tilbagelægge buelængden l. Buelængden kan også skrives

$$l = (r_2 - \frac{R}{2})\theta \tag{8.6}$$

Ved indsættelse af ligning 8.4 i ligning 8.6 fås for ligning 8.5 med hensyn til v_1:

$$\begin{aligned} \frac{v_1 + v_2}{2} &= \frac{l}{t_l} \\ &= \frac{\left(r_2 - \frac{R}{2}\right)\theta}{t_l} && (8.7) \\ &= \frac{\left(R\left(1 + \frac{v_1}{v_2 - v_1}\right) - \frac{R}{2}\right)\theta}{t_l} && (8.8) \end{aligned}$$

Det bemærkes, at $\omega = \frac{\theta}{t_l}$ er vinkelhastigheden, idet farten er konstant under svingmanøvren. Ved omskrivning af ligning 8.8 fås

$$\begin{aligned} (v_1 + v_2)\left(-\frac{t_l}{R\theta}v_1 + \frac{t_l}{R\theta}v_2 - 1\right) &= 0 \\ \Leftrightarrow v_1 = -v_2 \wedge v_1 = v_2 - \frac{R\theta}{t_l} &= v_2 - R\omega && (8.9) \end{aligned}$$

Da der for v_1 og v_2 skal gælde, at de har samme fortegn, når robotten svinger, er den søgte hastighedsrelation givet ved

$$v_1 = v_2 - \frac{R\theta}{t_l} \tag{8.10}$$

For en fast værdi for t_l opnås en vinkelhastighed ω afhængig af vinklen θ. Hastigheden v_1 kan derfor beregnes, såfremt v_2 på forhånd er kendt. v_2 bestemmes til at være robottens fremadrettede hastighed umiddelbart før svingmanøvren, og v_1 kan således beregnes. Det bemærkes, at relationen er udviklet for et højresving men kan uden videre generaliseres til også at gælde venstresving ved at bytte om på v_1 og v_2 i ligning 8.10.

8.3.3 Modelbegrænsninger

På baggrund af modellens tilnærmelser og antagelser har modellen et begrænset anvendelsesområde. Formålet med dette afsnit er at klarlægge modellens anvendelsesområder.

Kørsel tæt på et objekt. Afstanden fra robotten til det prioriterede objekt er ikke medtaget i modellen. Buelængden robotten tilbagelægger ved sving er således kun afhængig af vinklen θ og en af hastighederne v_1 og v_2. Modellen kan følgelig kun benyttes, når afstanden til objektet er større end buelængden, det kræver for robotten at dreje mod objektet. Der bør derfor køres med lavere hastighed og simplere styring, når robotten er tæt på objektet, så retningen kan finjusteres.

Understyring. På baggrund af tilnærmelsen mellem vinklen til objektet og vinklen θ opnås en understyring af robotten, idet vinklen til objektet ikke er 0, efter robotten har tilbagelagt buelængden l.

Krav til vinkelhastighed. Ud fra den på forhånd bestemte tid t_l, det skal tage robotten at tilbagelægge buelængden l, kan et interval for vinkelhastigheden ω fastlægges for $0 < \theta < \pi$. For at robotten skal bevæge sig langs et cirkelstykke, er det krav, at v_1 og v_2 er positive. Af ligning 8.10 gælder der således for t_l:

$$v_2 - \frac{R\theta}{t_l} > 0$$

$$\Updownarrow \tag{8.11}$$

$$t_l > \frac{R\theta}{v_2} \tag{8.12}$$

Styringen kalibreres følgelig, så det ene hjul er standset for $\theta = |\pi|$, idet $|\theta|$ er defineret i intervallet $0 \leq |\theta| \leq \pi$. Ved venstresving vil det samme gælde for v_1.

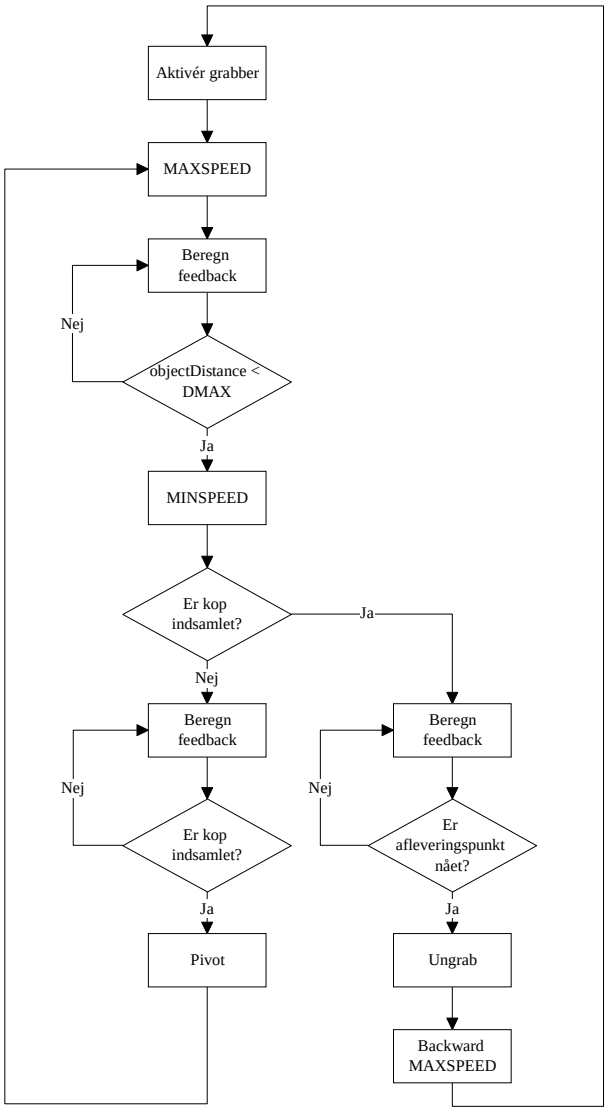
8.4 Implementation af styringsalgoritme

På baggrund af den opstillede løsningsmodel beskriver dette afsnit, hvorledes det er valgt at implementere løsningsmodellen for styring af robotten.

Programflow. Efter implementation af feedback algoritmen i styringsalgoritmen består den resterende regulering i at kontrollere hvor i processen, robotten befinder sig ved indsamlingen af en kop. Når indsamlingen af kopper påbegyndes startes algoritmen ved at aktivere grabberen. De øvrige situationer begrænser sig til indsamling og aflevering af en kop som vist på figur 8.1. Imellem disse situationer styres robotten med feedback.

Hver eneste gang en opkode er sendt, returnere styringsalgoritmen til andre dele af klientsoftwaren, der opdaterer vinkel og afstand til objektet. Herefter kaldes styringsalgorit-

men med de nye værdier og fortsætter programafviklingen. Hele forløbet er indeholdt i følgende figur for programflowet for styringsalgoritmen:



Figur 8.5: Program flow for styringsalgoritmen.

På vej mod koppen køres hele tiden med den valgte hastighed MAXSPEED, der er en parameter i styringsalgoritmen. I det øjeblik afstanden til objektet *objectDistance* bliver mindre end afstanden DMAX bremses robotten, og farten er nu MINSPEED. Feedbackreguleringen ændres nu så der drejes med en fast vinkelhastighed så robotten rammer koppen så præcist som muligt. Når koppen er indsamlet, foretages en pivotvending, og algoritmen køres igen

som vist på figur 8.5, indtil afleveringspunktet er nået, hvor koppen afleveres. Efterfølgende køres algoritmen forfra, såfremt der er kopper tilbage at indsamle.

Ved justering af de omtalte parametre optimeres styringsalgoritmen efterfølgende ved forsøg i laboratoriet. Eksempelvis kan farten sættes op og ned efter behov, ligesom decelerationsprofilen for robotten stiller krav til afstanden DMAX.

8.5 Delkonklusion

Der er i dette afsnit opstillet krav til robottens styringsalgoritme. Ud fra disse krav er der udviklet en model for feedbackmekanismen til beregning af robottens hjulhastigheder ud fra vinklen fra robotten til et prioriteret objekt og kravet til svingningstiden t_l. Anvendelsesområdet for modellen er bestemt, så det er muligt at implementere den i styringsalgoritmen.

Modellen gør det muligt hurtigt at overskue og justere variable ved tilpasning af feedbackmekanismen gennem forsøg, så styringen af robotten optimeres mhb. på at minimere indflydelsen af usikkerhederne. Ligeledes indeholder styringsalgoritmen parametre der kan justeres mhb. på eksperimentel optimering.

Forsøg og test

9

Formålet med dette kapitel er at opstille en række tests af systemet. Herunder præcisionen af robottens manøvredygtighed, effektforbrug og målinger på grabberens følsomhed.

9.1 Forsøgsserie

Robottens præcision er afgørende for om systemet virker efter hensigten og reagerer korrekt på klientens input, således en orientering på banen er mulig. Hastigheden af robotten, samt dens acceleration ved forskellige indstillinger på klienten, er nødvendigt at kende, for herved at give robotten korrekte instruktioner. Det samme gælder følsomheden af lyssensoren til grabberarmen, for at kunne beregne hvornår grabberen skal lukke omkring en indfanget kop. Endvidere vil et estimeret effektforbrug også være af interesse, idet en vurdering af hvor lang tid robotten kan køre ved et fuldt opladt batteri, kan have betydning ved konkurrencen.
Der er foretaget følgende forsøg med systemet:

- Kørsel efter ret linie.
- Hastighedsmåling.
- Afstandsmåling af sensorens rækkevidde.
- Effektforbrug af robot.

Forsøgene foretages med størst præcision efter de anvendte måleredskaber.

9.2 Kørsel efter en ret linie

I forsøget sættes robotten til at køre 6 [m] ligeud. Ud fra en ret linie måles afvigelsen fra robottens bane.

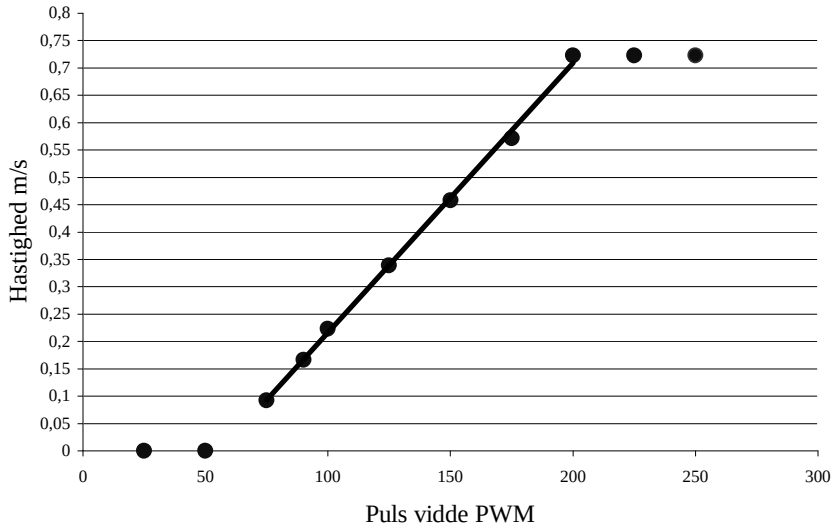
Pulsvidde	Afvigelse [m]	Afvigelse i grader
100	0.44	4.2
150	0.41	3.9
175	0.21	2.0
200	0.00	0.0

Tabel 9.1: Robottens afvigelse ved forskellige hastigheder.

Af tabel 9.1 fremgår den største afvigelse til 4.2 °. Afvigelserne er ikke ens over den givne måleserie, hvilket betyder at de to motorers omdrejningstal er forskellig. Forsøget viser, at afvigelserne er relativt små, hvorfor der i projektet ikke vil tages højde for usikkerheden.

9.3 Hastighedsmåling

I dette forsøg skal robotten, med konstant hastighed, køre langs en 6 [m] bane. Ud fra tiden det tager at tilbagelægge afstanden, udregnes robottens hastighed. Målingerne er foretaget ved forskellige PWM opløsninger. Der kan herved findes en sammenhæng mellem robottens hastighed i [m/s] og angivelsen i PWM.
Nedenstående graf 9.1 viser en afbildning af bilens hastighed ved forskellige PWM værdier.



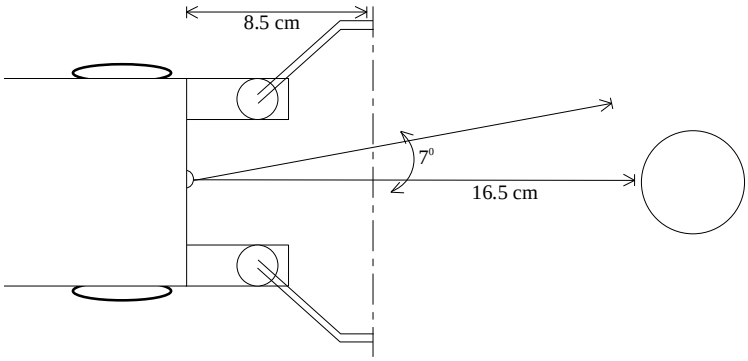
Figur 9.1: Hastigheden som funktion af PWM.

Grafen afbilleder målinger med PWM værdier fra 75 til 200. Målinger over og under de angivet værdier er ikke medtaget, da robotten holder stille ved PWM værdier under 75 og værdier over 200 ikke øger hastigheden.

Grafen viser en tilnærmelsesvis ret linie, hvormed det kan konkluderes, at der findes en linearitet mellem PWM og robottens hastighed.

9.4 Afstandsmåling af sensorens rækkevidde

I forsøget er afstanden mellem kop og sensoren målt, for hvornår grabberen lukker. På figur 9.2 er opstillingen illustreret, med de tilhørende måledata.



Figur 9.2: Sensorens måleområde inden grabberen lukker.

Grabberen lukker, når en kop kommer inden for 25 [cm] fra sensoren, samt i en vinkel af 7° målt fra midterlinien. Grabberens indfangnings område er 16.5 [cm], hvilket betyder at robotten skal komme kørende ind på koppen, for at den bliver fanget.

9.5 Effektforbrug af robot

I forsøget måles batterispændingen og strømforbruget ved forskellige hastigheder. Tabel 9.2 viser måleresultaterne:

Batt. [V]	PWM 75	PWM 100	PWM 150	PWM 200	Gns. [A]
10.7 [V]	0.24 [A]	0.29 [A]	0.40 [A]	0.51 [A]	0.36 [A]

Tabel 9.2: Strømforbrug ved forskellig pulsvidde.

Af måleresultaterne bruges et gennemsnitligt stømforbrug, udregnet til 0.36 [A], hvormed effektforbruget bliver 3.85 [W].

Til projektet haves to genopladelige batterier til rådighed på hhv. 0.8 og 1.42 [Ah]. Udregning af driftstiden bliver:

$$\begin{aligned} \frac{800[mAh]}{0.36[A]} &= 2.2[h] \\ \frac{1420[mAh]}{0.36[A]} &= 3.9[h] \end{aligned} \tag{9.1}$$

Disse tider er ved konstant kørsel med en gennemsnitlig hastighed.

9.6 Delkonklusion

Forsøgsrækken har vist, hvilken præcision og fejltolerance robotten opererer med. Disse resultater bruges bla. til optimering af robottens styre- og kontrolrutiner, samt til vurdering af systemets stabilitetsforhold.

Ved kørsel efter en ret linie drejer robotten blot en smule. Denne fejlmargin får ingen betydning for robottens præcision, idet både feedback rutine og kamerarets periodiske opdatering af banen sørger for konstant tilpasning af robottens position.
Motorernes relativt store moment muligør, at robotten kan geares til at køre med en høj hastighed, hvilket bevirker, at robotten hurtigt kan indsamle kopper. Indsamlingen af kopper er under et forsøg målt til at vare mellem 15 og 35 sekunder pr. kop, alt efter placering. Forsøget er foretaget med ni kopper på banen for således at få bekræftet at alle klientcomputerens prioritering, justerings og kontrolrutiner virker efter hensigten.

Måleresultatet kan kun tilnærmelsesvis sammenlignes med, hvorledes robotten vil reagere under et spil, idet der i forsøget ikke er taget højde for modstanderens robot. Forsøget giver dog et indtryk af, at alle systemets enkeltdele gennem grænsefladerne arbejder fejlfrit sammen.

Ud fra målinger af robottens effektforbrug er det vurderet, at batteriernes kapacitet er tilstækkelig til brug ved konkurrencen.

En samlet vurdering af robottens hardware og styringrutiner, må derfor siges at leve op til målsætningen om en hurtig og stabil robot.

Konklusion 10

Formålet med dette kapitel er at konkludere på projektet, hvilket vil sige sammenligne de opstillede krav med det realiserede system, samt diskutere hvilke fremtidige udvidelsesmuligheder systemet kunne underlægges.

10.1 Overordnet formål

Projektets formål var som udgangspunkt at konstruere en autonom robot, der via en række sensorer og et bagvedliggende controller netværk, kunne opsamle en række objekter på en bane og samtidigt gøre dette hurtigere end en opponerende gruppes robot, der havde samme formål. Igennem projektet er mange fagområder blevet berørt, og det har været opgaven at indhente, samle og koordinere denne viden med henblik på at analysere, designe og implementere systemet. Dette er indledningvis sket igennem opdeling af det samlede system, hvorefter der er foretaget en kravssætning til delsystemerne, samt til de enkelte interfaces mellem disse. Ud fra kravene er delsystemerne analyseret, designet og implementeret, hvorefter det samlede system er gennemtestet i laboratoriet.

I det følgende gennemgås delsystemerne med henblik på at undersøge, om kravene til disse er opfyldt, samt kort at opsummere hvorledes delproblemerne er blev løst i projektet.

10.1.1 Server

Opbygning og arkitektur. Serverens arkitektur var på forhånd givet, hvilket også var tilfældet for kamera og driver software. Dette blev opfattet som en blackbox, der var i stand til at leverer 25 [fps], der efterfølgende skulle analyseres.

Objekt identifikation. Kravene til billedanalysen var at identificere de to robotter samt kopperne på banen, inden for en tidsramme på 100 [ms]. Ud fra identifikation vha. areal og omkreds betragtninger blev objekterne identificeret, og efter en optimering af software rutinerne, blev det muligt at analysere op til 25 [fps], hvorved framegrabberen blev flaskehalsen i delsystemet.

I forbindelse med identifikation af objekter, blev det fundet nødvendigt at benytte en empirisk indgangsvinkel, med henblik på at tage højde for variationer i objekternes

egenskaber (areal, omkreds) som følge og skiftende lysforhold og forvrængning i kameraet. Mere præcist blev der taget målinger af robotternes egenskaber forskellige steder på banen, hvorefter disse data blev implementeret i programmet. Dette skete i modsætning til at opstille en egentlig matematisk model for variationerne.

10.1.2 Hardware

Manøvredygtighed og køreegenskaber. Opbygningen af robotten blev foretaget i LEGO, og det løste mange praktiske problemer mht. at ændre på funktionaliteten igennem designfasen. Kravene til selve robotten blev opfyldt, og der blev konstrueret en robot, der var i stand at manøvrere på banen under hensyntagen til obstruktioner og objekter.

Kommunikation med klient. Kommunikation mellem klient og robot blev indledningsvis valgt til at skulle være trådløs, bidirektional og støjresistent. Dette blev implementeret vha. et radiolink, der opfyldte de givne krav til bidirektionalitet samt transmissionshastighed. Yderligere blev kommunikationen forbedret vha. fejlkontrolkodning mhb. på minimering af fejltransmissioner. Overordnet muliggjorde radiokommunikationen, at robotten kunne reguleres mellem 15 og 20 gange pr. sekund.

Realtidsegenskaber. Opbygningen og implementation af MCU hardware og software blev foretaget ud fra en række krav til og overvejelser omkring realtids problematikker. Til honorering af disse krav, blev PIC MCUen valgt og dens egenskaber blev benyttet til at muliggøre over 20 reguleringer af robotten pr. sekund.

10.1.3 Klient

Opbygning og arkitektur. Opbygning og arkitektur for klienten blev valgt til en standard PC med Linux operativ system, idet dette kunne opfylde de opstillede krav til kommunikation med eksterne enheder, samt håndtering af multiple processer. Systemet viste sig i stand til at opfylde de opstillede krav til timing samt håndtering af data fra billedserveren samt til og fra robotten.

Prioriteringsalgoritmer. Ud fra en objektorienteret indsgangsvinkel blev klienten analyseret, og der blev udviklet rutiner til prioritering af de forskellige destinationer, samt omsætning af destination til en egentlig robot handling. Systemet blev simuleret, og det viste sig senere i laboratoriet, at systemet var i stand til at prioritere kopperne efter parametre som afstand, vinkel og modstandernes formodede destination.

Reguleringsalgoritmer. Til styring af robotten fra et punkt til et andet, blev der udviklet en regulerings algoritme, som løbende regulerede robottens kørsel ud fra de indkommende billeddata. Idet regulerings algoritmen i samarbejde med prioriterings algoritmen og det underliggende data håndteringssystem viste sig i stand til at regulere robotten tilfredsstillende, må formålet med realiseringen af klienten siges at være opfyldt.

10.1.4 Proceskommunikation

Klient / server Til at forbinde de forskellige processer lokalt og over netværk blev der brugt en række forskellige metoder. Kommunikationen mellem klienten og serveren blev foretaget i henhold til UDP/IP protokollen, som viste sig i stand til at opfylde de givne krav til transmissionshastighed. De viste sig endvidere ikke nødvendigt at implementere tidsstempling af pakker, idet transmissions-hastighed og -regelmæssighed var tilpas stabil.

Internt i klienten. Kommunikationen mellem de enkelte processer i klienten blev implementeret som beskedkøer, og opfyldte endvidere de opstillede krav til dataoverførsel samt synkronisering mellem processerne.

10.2 Udvidelsesmuligheder

- Forbedrede matematiske modeller indenfor:

  Kamera model. Der kunne opstilles og benyttes en forbedret kameramodel, der tager højde for forvrængning i kameraet.

  Proceskontrol model. Forbedret model af hele styringssystemet, der tager højde for motormodeller, friktion, forsinkelse i systemet osv.

- Udvidelse af robotten

  Tachometre. Med henblik på opnå større mulighed for regulering af robotten kunne denne udstyres med tachometre med henblik på bestemmelse af kørselsretning.

  Proximity detektorer. Robotten kunne udstyres med flere proximity detektorer, således de opsamlede kamera data, kunne komplementeres af flere målinger fra selve robotten.

  Forbedret transmission. LEGO tandhjulene viste sig ikke at kunne overføre momentet fra motoren over en længere periode uden at det blev nødvendigt at udskifte disse. Tandhjulene kunne derfor udskiftes med mere holdbare slags.

- Forbedret billedbehandling

  Forbedret objektidentifikation. Identifikationen af objekter kunne forbedres ved at benytte mere avancerede teknikker til identifikation af former og kanter i billedet.

- Udvidelse af robot intelligens

  Neurale netværk. Simple neurale netværk kunne benyttes til, at robotten kunne lære af sine fejl. Eksempelvis kunne rutinen som benyttes til at dreje ind på en kop, underlægges kontrol af et neuralt netværk, der således lærer af de sidste handlinger mhb. på optimering af de næste.

10.3 Tilegnede færdigheder

Idet systemet overordnet, som i de enkelte dele, opfylder de krav, der er blev stillet, og samtidig fungerer efter hensigten, må formålet med projektet siges at være opnået. Derudover har gruppen opnået viden indenfor en lang række fagområder, hvilke blandt andet omfatter:

- Billedanalyse.
- Objektorienteret analyse, design og programmering.
- Interproces kommunikation og netværksprogrammering i Linux og Irix.
- Radiokommunikation og MCU systemer.

Idet kravene beskrevet i projektenhedsbeskrivelsen for E5 alle er opfyldt samtidig med, at de realiserede systemer fungerer efter hensigten, må formålet med projektet siges at være opfyldt.

Radiolink

A

Formålet med dette afsnit er at fremsætte en nærmere beskrivelse af de anvendte radiomoduler. Der vil fremgå, hvilke karakteristika radiomodulet besidder i en mere hardwaremæssig henseende. Dette omfatter den fysiske benopsætning, samt timingen af kommunikationen med henblik på at opnå en korrekt dataoverførsel.

A.1 Transceiver modul

Til den trådløse dataoverførsel benyttes to transceiver radiomoduler, BiM2-433-64. En transceiver er i stand til at operere i half duplex mode, hvilket giver mulighed for både at sende og modtage data, i henholdsvis transmit og receive mode. Radiomodulet indeholder en FM-modulator samt FM-demodulator, der opererer ved udsendelse af UHF FM radiobølger med en frekvens på 433.920 [MHz]. Den maksimale overførselshastighed af data er 64 [kbit/s], samt en rækkevidde på op mod 200 [m] med en korrekt konstrueret antenne og frit udsyn. BiM-433-64 er opbygget således, at der er mulighed for tilslutning til microcontrollere og anden logisk seriel datastream.

Ben konfiguration. Tilslutning af en microcontroller foregår ved radiomodulets RXD og TXD benforbindelser, for henholdsvis receive og transmit af data.

Til konfiguration af radiomodulets funktioner anvendes yderligere to benforbindelser, hhv. $\overline{TXSelect}$ og $\overline{RXSelect}$. Benenes logiske tilstand afgør, hvilken tilstand modulet befinder sig i. Opsætningen kan ses ud fra tabel A.1. Afhængig af indstillingen konfigureres radiomodulet til at sende eller modtage data. Endvidere er der mulighed for at bringe modulet i sleep tilstand samt en selfloop test.

$\overline{TXselect}$	$\overline{RXselect}$	Funktion
1	1	Power down
1	0	Receive mode
0	1	Transmit mode
0	0	Test loop back

Tabel A.1: Opsætning af BIM-433-64.

Ud over de 4 omtalte ben, som er et minimum for at operere modulet i half duplex mode, er der yderligere et ben med betegnelsen CD Carrier Detect. Dette ben er en aktiv lav udgang, som trækkes lav, når radiomodulet i receive mode opfanger en bærebølge, fra et andet tilsvarende modul.
For at radiomodulet kan forbindes til et TTL kompatibelt in- og output, i eksempelvis en microprocessor, er det nødvendigt, at radiomodulets benforbindelser får den rette impedans tilpasning. Dette kan opnås ved at drive benforbindelserne gennem logiske CMOS enheder. Hertil benyttes en CMOS hexinverter, der opkobles, således at signalet inverteres to gange. Vigtigt er især, at CD og RXD benet tilpasses, da disse har en høj udgangsimpedans på henholdsvis 50 [k Ω] og 10 [k Ω].

Transmissionspakker. Når en mængde data ønskes transmitteret, skal de specifikke data sendes i datapakker gennem radiomodulerne. Pakkernes størrelse kan frit bestemmes og defineres af brugeren. Pakkerne sendes som asynkron seriel data efter en nærmere bestemt RS-232 protokol. Dog er det en betingelse, at radiomodulerne synkroniseres, og datasliceren på RXD opnår stabilitet forud for afsendelse af data. Dette kan opnås ved forud for hver datatransmission at sende en preamble sekvens bestående af '1010101...' i minimum 3 [ms].

Timing. For at sikre en pålidelig dataoverførsel har radiomodulerne en række krav til timing. Der skal tages hensyn til timingen mellem de forskellige ben og ved skift mellem de to modes. Når RX benet trækkes lavt, og modulet befinder sig i receive mode, vil CD benet først være aktiv lavt efter ca. 0.5 [ms] ved detektion af en bærebølge.
Ved skift mellem transmit og receive mode og vice versa, vil radiomodulet befinde sig i stabil tilstand efter ca. 1 [ms].
For de givne radiomoduler er der mulighed for at sende data med en hastighed mellem 1 [kbit/s] og op til 64 [kbit/s]. I projektet anvendes kun en standard hastighed på 9.6 [kbit/s], der er typisk for den anvendte RS-232 potokol. [Radiometrix, 2001]

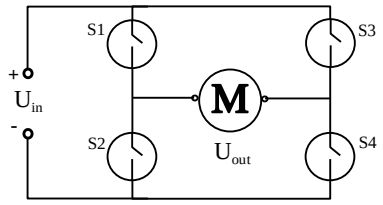
Motor driver

B

Formålet med dette afsnit er at redegøre for de tekniske aspekter ved styring af de anvendte motorer. Den hardwaremæssige teori, der ligger til grund for valget af komponenter, bliver analyseret, og opsætningen af denne beskrives.

B.1 H-bro

Det er på forhånd valgt at anvende dc-motorer til navigation af robotten. Disse kan ikke direkte tilsluttes den valgte microprocessor, da motorernes effektforbrug langt overskrider, hvad der kan trækkes fra en microprocessor. I stedet anvendes et effektgivende hjælpekredsløb, der styres af microprocessoren. Dertil findes forskellige topologier, men fælles er konceptet baseret på en H-bro konstruktion, hvor fire kontakter styrer én motor. Kontakterne kan åbnes og lukkes i en rækkefølge, der muliggør, at polariteten over motoren kan skifte, således strømretningen ændres, og motoren roterer den modsatte vej. På figur B.1 er en simpel H-bro illustreret.



Figur B.1: H-bro i form af fire kontakter og en motor.

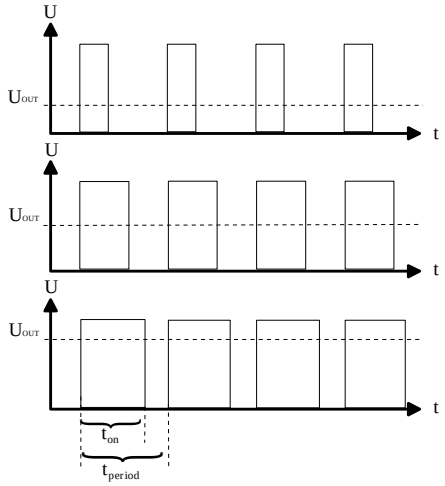
Motor retning. Hvis kontakten er indstillet efter figur B.1, løber der ingen strøm, og motoren står i friløb. Sluttes kontakterne $S1$ og $S4$, mens $S2$ og $S3$ er åbne, vil strømmen gå gennem motoren fra venstre mod højre, og motoren drejer. Modsat retning opnås, når $S2$, $S3$ er lukket og $S1$ og $S4$ står åbne. For at opnå en bremsning af motoren kan $S1$ og $S3$ eller $S2$ og $S4$ sluttes. Det skal dog bemærkes, at bremsepositionen bør undgås, idet motoren trækker et unødigt strømforbrug. Sluttes alle kontakter, vil batteriet blive kortsluttet.

Motor hastighed. Spændingspotentialet over en dc-motor vil ændre motorens rotationshastighed. Spændingen kan reguleres ved terminal tilslutning U_{in}, eller ved benyttelse af switchmode teknikken 'PWM' Puls Width Modulation.

Ved PWM benyttes transistorer i stedet for kontakter til at åbne og lukke for strømmen til motoren. Typisk bruges power transistorer såsom MOSFET teknologi, da disse kan trække store strømme. Det resulterende spændingspotentiale over motoren bliver, ved anvendelse af PWM, forholdet mellem tiden, hvor transistoren leder, og den samlede periode tid, dvs. duty cyclen multipliceret med U_{in}.:

$$U_{out} = \frac{t_{on}}{t_{period}} \cdot U_{in} \tag{B.1}$$

Af figur B.2 ses hvorledes spændingen falder, når duty cyclen mindskes.

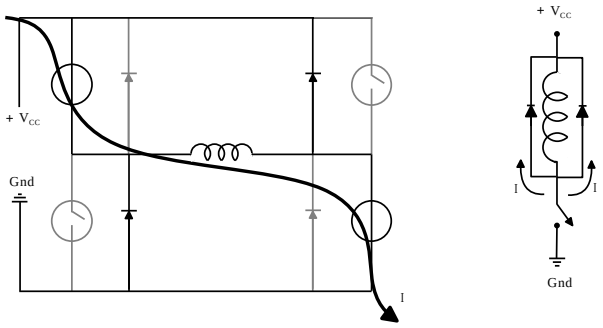


Figur B.2: Spændingsregulering med PWM. De stiplede linier angiver den resulterende spænding ved tre forskellige t_{on} tider.

Ved benyttelse af PWM forudsættes, at switchfrekvensen holdes tilstrækkelig høj. Herved har ripplespændingen mulighed for at blive udglattet ved brug af kondensatorer, således at den stiplede linie på figur B.2 tilnærmelsesvis opnås.

I konstruktioner med switchning opstår et velkendt problem med induktive ladninger, forårsaget af motorens spole. Idet spolens opladede energi ikke momentant kan forsvinde, når transistoren åbner, vil spændingsfaldet over spolen øjeblikkeligt opnå høje spidsværdier. Problemet kan afhjælpes ved at indsætte en flyback diode, der leder strømmen væk fra spo-

len, når switchen åbner. På figur B.3 er der vist hvorledes H-broen er opkoblet med fire dioder, idet denne indeholder fire switchende transistorer.



Figur B.3: Fire dioder indsat i H-broen, samt model for ækvivalentdiagrammet over motoren.

I modellen vil to dioder altid være aktive, alt efter hvilken retning motoren drejer. Dioderne vil således sørge for, at den inducerede strøm ledes tilbage til spændingsforsyningen, således spændingspotentialet over spolen, og dermed motoren, kan gå mod nul, når switchene åbner. [Jones, 1993]

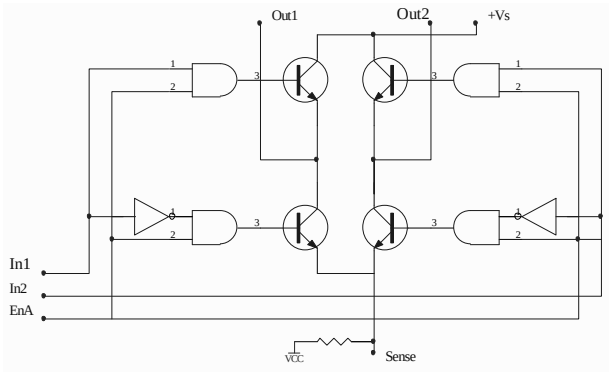
B.2 ST L298 Dual Full-Bridge Driver

I projektet er der af pladsmæssige årsager valgt at anvende integrerede H-bro løsninger til styring og regulering af motorerne. Motorerne er i laboratoriet testet med henblik på strømforbruget ved det største moment, for således at afgøre hvilken H-bro der kan bruges.

Valget er faldet på dual H-broer fra STMicroelectronics model ST L298, der for hver bro tillader et strømforbrug på op til 2 [A], der til projektet er tilstrækkeligt. Endvidere er det muligt at operere med motorspændinger op til 46 [V]. Alle logiske indgange er fuldt TTL kompatible, der muliggør direkte tilslutning af en microprocessor.

Transistor styring. I det følgende tages udgangspunkt i diagrammet for en halv L298, da den anden halvdel er identisk. Indtroduktionen til H-broerne er basis for opbygningen af

L298, der benytter NPN bipolære transistorer i stedet for kontakter. Simpel logik med AND kredse sørger for, at transistorerne åbner og lukker i korrekt rækkefølge.



Figur B.4: Blokdiagram af en halv ST L298

På figur B.4 ses det elektriske blokdiagram af L298. Out_1 og Out_2 er udgangen til én dc-motor. Tre logiske indgange In_1, In_2 samt En_A aktivere AND kredsen. Ud fra sandhedstabellen tabel B.2, hvor En_A er sat høj, fremgår det, om den tilsluttede transistor leder '1' eller er åben '0'. Motorens drivretning er også angivet.

In_1	In_2	S_1	S_2	S_3	S_4	Motor
1	1	1	0	1	0	Hold
1	0	1	0	0	1	Frem
0	1	0	1	1	0	Tilbage
0	0	0	1	0	1	Hold

Tabel B.1: Sandhedstabel for H-broens indgange, med En_A sat høj.

Vælges det at holde benet En_A logisk lavt, bliver alle output fra AND kredsene '0', hvormed motoren står i friløb.

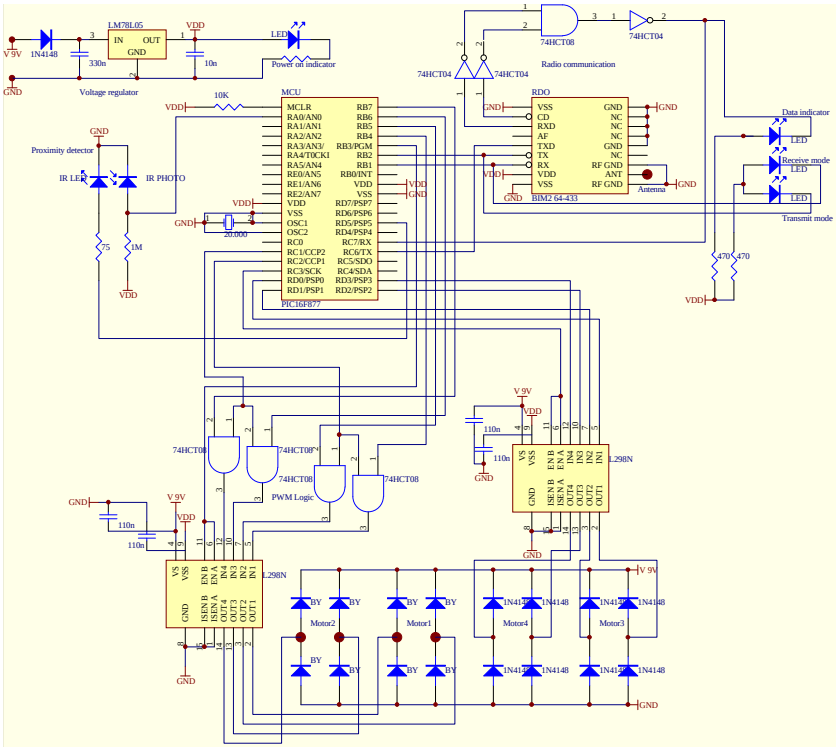
Ben konfiguration. Alle L298's logiske kredse opererer ved 5 [V], der forsynes ved indgangsterminalen V_{ss}. Spændingsforsyningen, der driver motorerne, tilsluttes indgangen V_s. *Sense* udgangen, der er beregnet til overvågning af motorernes strømforbrug, benyttes ikke i denne opstilling. Endelig skal fire effektdioder tilkobles. For et overblik over den samlede opstilling henvises til side C.1. [STMicroelectronics, 2000]

Hardware diagrammer

C

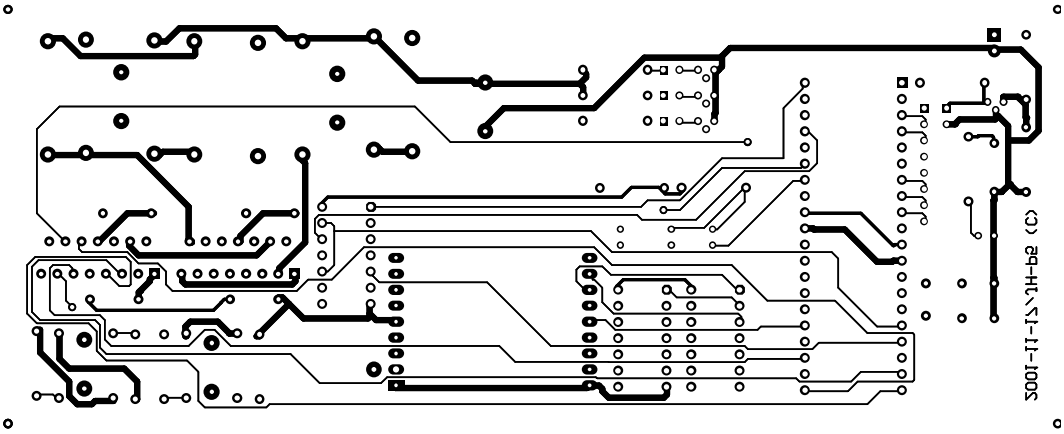
Dette appendiks omhandler de mere tekniske detaljer omkring hardwaren i robotten.

Diagrammet for hele hardwarekonstruktionen i robotten er vist på figur C.1:

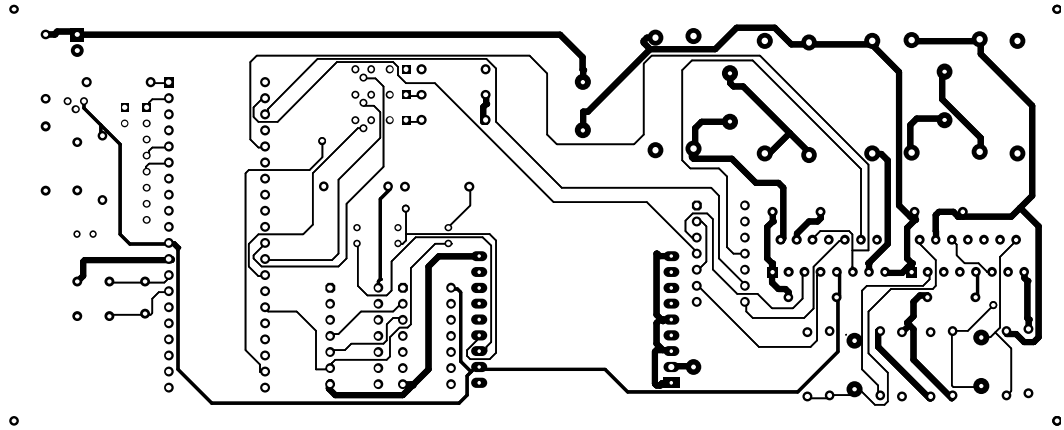


Figur C.1: Diagram over hardware implementeret i robotten.

Ved hjælp af diagrammet er der udlagt et dobbeltsidet printlayout, som kan ses på figur C.2 og C.3:

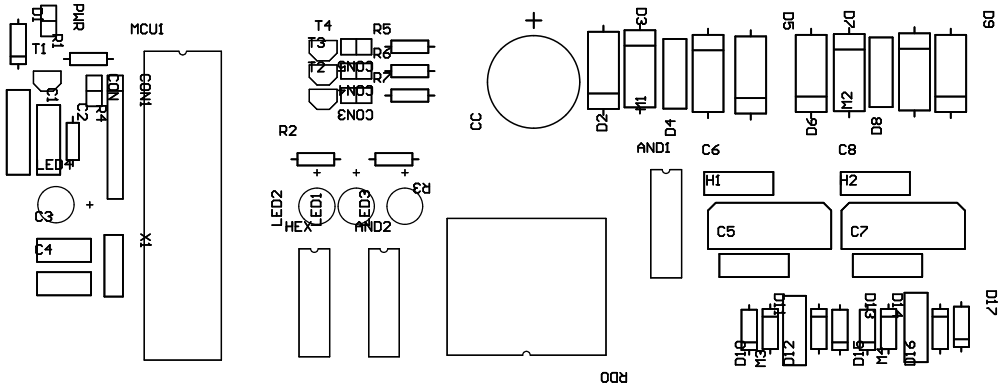


Figur C.2: Oversiden af det udlagte print.



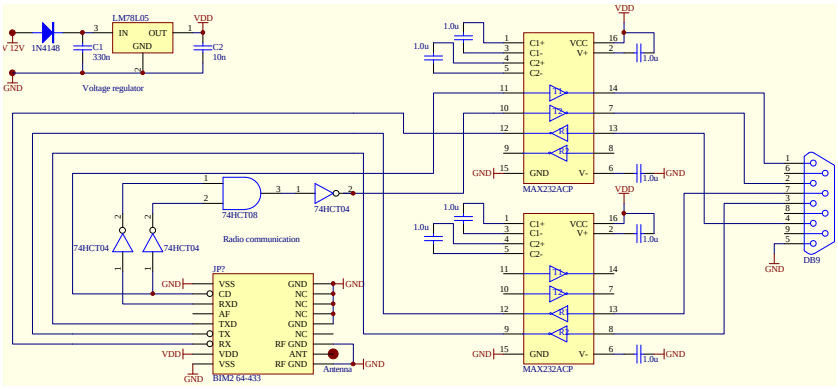
Figur C.3: Undersiden af det udlagte print.

Printlayout set fra oven, med omrids af komponentplacering, kan ses på figur C.4:



Figur C.4: Komponenternes placering på det endelige print.

På figur C.5 ses det færdige diagram over radiolinket til klientpcen.



Figur C.5: Diagram over RS-232 kompatibelt radiolink til pc.

Seriel kommunikation

D

Kommunikationen mellem klienten og robotten foregår via et radiolink, der benytter en på forhånd specificeret protokol. Dette afsnit omhandler denne protokols karakteristik samt opsætning af driver hardware.

D.1 RS-232

Serielkommunikation er en transmissionsmetode, der ofte er anvendt til at forbinde computere til periferi enheder, såsom modems, printere, mus og i dette tilfælde et radiokommunikationsmodul. Protokollen, der benyttes til kommunikationen, er specificeret ud fra en international ISO standard RS-232, der beskriver, hvorledes data sendes over en seriel linie. RS-232 er en asynkron kommunikations metode, der benytter det binære system til at transmittere data i ASCII[1] format, ved at sende og modtage spændinger, der enten er høje eller lave, svarende til hhv. **1** (kaldet mark) og **0** (space). En sender kaldet en DTE enhed (Data Terminal Equipment) og en modtager DCE (Data Communications Equipment) skal forbindes således, at DTE enhedens output føres til DCEens input. Endvidere er det vigtigt at både DTE og DCE opsætningen er ens, således at de kan kommunikere fejlfrit. Følgende indstillinger er nødvendige:

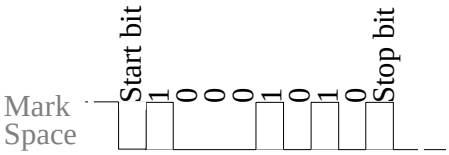
Baud rate. Da en asynkron forbindelse ikke indeholder et clocksignal, skal DCE enheden kunne sample de afsendte bits efter samme frekvens, som DTEen sender med. Denne hastighed er defineret ud fra antallet af bits, der kan sendes over linien pr. sek. kaldet bit raten.

Start/Stop bit. Synkroniseringen starter, når det første bit modtages. I følge RS-232 skal dette være et startbit med værdien **0**. Efterfølgende sendes databyten og afsluttes med et eller flere stopbits, der antager værdien 1.

Paritet. Til verifikation, om den afsendte byte er korrekt, sættes et paritetsbit. Pariteten kan udregnes på tre forskellige måder, idet lige, ulige og ingen paritet kan benyttes. Denne verifikation er af ringe karakter, idet fejl kun detekteres, hvis et ulige antal bits af databyten er ukorrekte. Oftest suppleres der med andre kontrolkodningssystemer. Fx. Hamming og Golay koder.

[1] American Standard Code for Information Interchange

Data format. Ved opsætning af en DTE eller DCE enhed specificeres RS-232 opsætningen efter formatet: *Bit rate - data bits - paritet - stop bits*. Opsætningen, der bruges i projektet, antager strengen 9600-8-N-1, der betyder: 9600 [bps], 8 databit, paritetstype none, og et stopbit. På figur D.1 vises bitsekvensen, der latches ud på linien ved afsendelse af hexadecimalværdien 45h (=1000101b), ved den anvendte opsætning.



Figur D.1: Bitsekvensen af 45h ved 9600-8-N-1

D.1.1 UART

For at aflaste processoren, der ellers konstant skal lytte på datalinien efter nye data, kan, der anvendes en UART[2]. En UARTs opgave er at holde styr på alle operationerne der skal til, for at transmittere data over kommunikationslinien med hensyn til overholdelse af RS-232 protokollen.

Dette gælder signal timing, paritet udregning, check af start og stop bit samt bit raten. Foruden sende og modtage linierne, har UARTen flere linier, der kan benyttes til kontrol. Nedenstående datalinier er at finde i et 9 polet D-stik:

Data Carrier Detect. Input signal der markerer, at DCE har skabt forbindelse.

Receive Data. Modtager serielle data.

Transmit Data. Sender serielle data.

Data Terminal Ready. Signaler at UARTen er parat til at sende data.

Data Set Ready. Input fra DCE'en, at denne er klar.

Request To Send. Linien trækkes høj, når DTE'en er klar til datamodtagelse.

Clear To Send. Linien trækkes høj, når DCE'en er klar til datamodtagelse.

Ring Indikator. Linien bruges kun, når DCE er et modem.

Ground. Stelforbindelse.

[2]Universal Asynchronous Receiver / Transmitter

Signalerne, der sendes over UARTen, er liniespændinger med en amplitude, der afgør om signalet er logisk 0 eller 1. Sammenhængen er som følger:

	Logisk 0	Logisk 1
Linie indgang	5 til 25 [V]	-5 til -25 [V]
Linie udgang	5 til 15 [V]	-5 til -15 [V]

D.1.2 Signalkonvertering

Idet radiomodulets linie ind- og udgange er lavere end de spændinger, UARTen udsender på linien, og polariteten er modsat, er det nødvendigt at konvertere liniesignalet. Til formålet anvendes driver kredse fra MAXIM model MAX232, der konverterer signalet fra UARTen til TTL kompatible spændinger. Kredsen indeholder to indgange og to udgange. Således kan radiolinkets udgangssignaler også transformeres til RS-232 niveau. MAX232 tilkobles en forsyningsspænding på 5 [V], og kredsen stabiliseres ved fire 1.0 [μF] kapacitorer, jf. datablad.

D.1.3 Tilkobling til radiomodul

Da klienten er en standard PC, forefindes allerede en indbygget UART. Anvendelsen af operativsystemet Linux muliggør en enklere programmering af PC'ens serielport. Data, der ønskes overført til den serielle linie, sendes som en fil til UARTen, der er ansvarlig for, at RS-232 protokollon overholdes.

PCens UART tillader bidirektional afsendelse og modtagelse af data, denne feature anvendes ikke, idet radiomodulet kun kommunikerer over én kanal ad gangen. Af appendiks A fremgår hvorledes radiomodulet skifter mellem modtagelse og afsendelse af data, vha. dets $\overline{RXSelect}$ og $\overline{TXSelect}$ ben.

Ud fra UARTens mulige ind- og udgange vælges følgende forbindelser mellem radiomodulet og PCens serielport:

PC serielport		BiM2-433-64
Request to send	$\rightarrow$	$\overline{TXSelect}$
Transmit Data	$\rightarrow$	TXD
Data carrier detect	$\leftarrow$	Carrier detect
Data terminal ready	$\rightarrow$	$\overline{RXSelect}$
Receive data	$\leftarrow$	RXD
Ground	-	Ground

Endvidere indsættes MAX232 driverkredsene, disse kan ses ud fra RS-232 diagrammet i afsnit C.

Tilkoblingen af radiomodulet i robotten foregår uden brug af MAX232 kredse, idet modulet tilkobles PIC16F877 indbyggede UART, som er TTL kompatible.

D.1.4 Software opsætning

I det følgende beskrives kort opsætningen af seriel porten i Linux, samt de muligheder I/O modes, som seriel porten kan benyttes i. Linux understøtter de standarder, som POSIX standarden definerer for terminal interfaces. Dvs. der er en række foruddefinerede parametre og funktioner, som kan benyttes til konfigurering af porten. Disse parametre kan inddeles i en række kategorier:

c_cflag. Indeholder en række **hardware kontrol** indstillinger, som benyttes til opsætning af eksempelvis baudrate og dataformat. I forbindelse med opsætning af seriel porten, benyttes følgende kontrol indstillinger.

CLOCAL. Ignorer modem status linier. (slået til)

CREAD. Sætter port i stand til at kunne modtage data. (slået til)

CS8. 8 bits pr. pakke. (slået til)

PARENB. Benyt paritet. (slået fra)

c_lflag. Indeholder en række **software kontrol** indstillinger, som afgører hvorledes seriel driveren behandler input fra porten. Herunder benyttes følgende.

ICANON. Afgør om porten skal benytte kanoniske input, hvilket vil sige, at indlæsningen af data sker i linier, der termineres af NL (newline). I modsætning til kanoniske input, kan benyttes non-kanoniske input hvor der kan indlæses et bestemt antal karakterer ved hver læsning. Idet robotten hver gang sender et bestemt antal karakterer, benyttes non-kanoniske input. (slået fra)

ISIG. Afgør om processen, der benytter seriel porten, skal modtage forskellige signaler, når der eksempelvis modtages bestemte karakterer. (slået fra)

c_iflag. Når processer læser fra en terminal. kan der benyttes forskellige input processeringer, således data behandles på forskellig vis inden processen modtager data. I projektet benyttes ikke nogen form for input processeringer, idet data skal modtages i det format som det blev sendt. Følgende parametre benyttes.

IGNPAR. Benyttes til at undersøge om der er paritetsfejl. (slået fra)

c_oflag. Tilsvarende som ved input processering, kan output processeres inden de sendes. Dette benyttes heller ikke, da data skal sendes i det rå format, som sendes til terminalen.

OPOST. Benyttes til at post processere data inden de sendes. (slået fra).

c_cc. Indeholder en række kontrol karakterer samt timeout funktioner. Idet input processering ikke benyttes, er det kun timeout indstillingerne, der benyttes. Følgende indstillinger benyttes i forbindelse med non-kanonisk input:

VTIME. Specificerer antallet af $0.1 \cdot$ *sekunder*, der må gå mellem hver karakter, der modtages. (sat til 16)

VMIN. Antal af karakterer, som skal modtages. (sat til 1)

Opsætningen af VTIME og VMIN afgør hvorledes porten skal opfører sig, når der modtages data.

[Lewine, 1991]

Internet protokoller

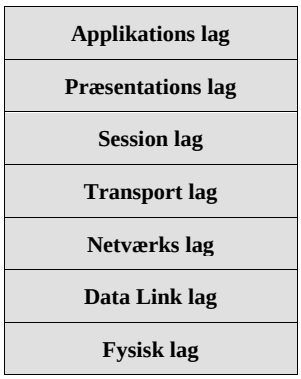
E

Formålet med dette kapitel er kort og overordnet at beskrive de netværksprotokoller, som benyttes i projektet. Det vil sige protokollerne som benyttes til kommunikation mellem klienten og billedserveren henholdsvis radioserveren.

E.1 Open Systems Interconnection

Sidst i 1970'erne nåede udviklingen af datanetværks konceptet et stadie, hvor der opstod behov for, på en standardiseret måde at kunne beskrive kommunikationsprocessen med henblik på modellering og videreudvikling. Løsningen af dette problem fandtes i 1984, hvor den Internationale Organisation for Standardisering (ISO) færdiggjorde Open Systems Interconnection modellen (OSI). Efterfølgende blev OSI modellen gældende standard inden for netværkskommunikation. [Ray, 1999]

OSI modellen består af en lagdelt model af netværksprocessen, hvor hvert lag i modellen håndterer en adskilt del af kommunikationsprocessen. Ved at foretage denne opdeling af kommunikationsprocessen i forskellige lag, simplificeres beskrivelsen af interaktionen mellem hardware og software. OSI modellen består af 7 lag, som i det følgende kort beskrives.



Figur E.1: OSI modellens 7 lag.

Applikations laget. Giver adgang til netværks services som filoverførsel og håndtering af beskeder mv. Dette lag kontrollerer endvidere adgangen til netværket samt transmission fra sender til modtager (flow control) på et generelt plan.

Præsentations laget. Kontrollerer og styrer data-format informationer for netværkskommunikationen. Laget konverterer udgående beskeder til et generisk format, som kan overføres via netværket, mens indgående data konverteres fra det generisk format, til et format, som applikationerne kan benytte. Laget er endvidere ansvarlig for protokol konvertering samt data kryptering og dekryptering.

Session laget. Tillader, at to netværks ressourcer kan opretholde en vedvarende kommunikationsforbindelse (session) over et netværk. Dvs. applikationer i hver ende af den pågældende session, er i stand til at udveksle data sålænge sessionen opretholdes. Laget holder således styr på opsætning af sessionen, kontrollerer data udveksling mellem parterne i form af synkronisering og sikkerhed mv.

Transport laget. Kontrollerer overførslen af data mellem de involverede parter i netværket. Dette sker ved at segmentere lange datapakker til pakker, som er passende for det pågældende netværk. Laget sørger endvidere for at udgående pakker vedhæftes fejlkontrol information, således modtageren kan afgøre om der er sket fejl under transmissionen, samtidig med at laget resekvenserer pakker, som ikke er modtaget i samme rækkefølge, som de er afsendt. Endelig kan laget bede om en retransmission, hvis der modtages pakker med fejl.

Netværks laget. Håndterer adresseringen af data, samt translateringen af de logiske netværks adresser til fysiske adresser. Laget afgøre endvidere, hvilken transmissionsrute, der skal benyttes, og tager herunder hensyn til Quality of Service informationer, alternative ruter samt leveringsprioriteter. Netværkslaget håndterer endvidere packet switching samt data routing.

Data Link laget. Håndterer specielle data frames mellem Netværks laget og det Fysiske lag. Hos modtageren, sørger laget for konvertering fra rå data format til data frames, som benyttes af Netværks laget. Den modsatte proces foregår hos afsenderen. Den omtalte data frame er den mest basale enhed for netværkstrafik.

Det fysiske lag. Konverterer de data frames, som modtages fra Data Link laget til elektriske signaler, som kan overføres via det fysiske medium i netværket (eks. kabler). Det fysiske lag håndterer således interfacet mellem computeren og det benyttede netværks medium.

En protokol stak er en samling af protokoller, som samarbejder om at muliggøre en netværksforbindelse. Generelt kan en sådan protokol stak opdeles i 3 sektioner, som alle kan mappes til OSI modellens lag: Netværk, Transport og Applikation. Idet hvert lag udfører en specifik funktion med egne regler, har en protokol stak ofte forskellige protokoller til hvert lag. I det følgende er listet en række eksempler på protokoller i de forskellige lag.

- Netværks protokoller.
  - IP (Internet Protocol). En del af TCP/IP protokolpakken som håndterer adressering og routning informationer.
  - IPX (Internetwork Packet Exchange). Protokol som benyttes til packet routing og forwarding.
- Transport protokoller.

  - SPX (Sequenced Packet Exchange). En forbindelsesorienteret protokol, som benyttes til kontrol af data levering.
  - TCP (Transmission Control Protocol). Den del af TCP/IP protokolpakken, som sørger for pålidelig data transmission.
  - UDP (User Datagram Protocol). Protokol som bygger oven på IP protokollen, og benyttes til hurtig men mindre pålidelig data overførsel.

- Applikations protokoller.
  - FTP (File Transfer Protocol) En del af TCP/IP protokol pakken, som benyttes til filoverførsel.
  - SMTP (Simple Mail Transport Protocol) En del af TCP/IP protokol pakken, som sørger for mail overførsel.

[Ray, 1999]

I forbindelse med dette projekt benyttes Netværks protokollen IP, samt Transport protokollerne TCP og UDP. Applikationsprotokollerne, som benyttes, er ikke foruddefinerede protokoller, men egne tilpassede protokoller, som opfylder de krav, som projektet stiller.

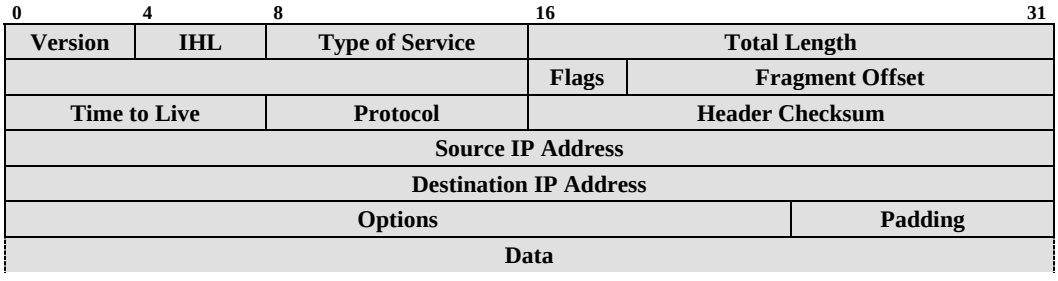
I det følgende beskrives kort principperne i IP, TCP og UDP.

E.2 Internet Protokollen

IP protokollen i den nuværende form er beskrevet i RFC1340, som danner grundlag for den følgende beskrivelse. Den primære funktion for IP protokollen er adressering og fragmentering. IP sikrer en uniform måde at adressere maskiner på netværk, uafhængigt af det fysiske nets opbygning.

Idet forskellige typer netværk kan benytte sig af forskellige pakkestørrelser, varetager IP protokollen endvidere fragmentering af datapakker, hvis disse er for store.

En IP header som vist på figur E.2 benytter sig af en række parametre, for at levere de omtalte fragmenterings og adresserings services.



Figur E.2: IP header.

I det følgende omtales de 4 vigtigste parametre, som er Type of Service, Time to Live, Options samt Header Checksum.

Type of Service. Kan benyttes til at kræve en bestemt transmissions kvalitet. Eksempelvis kan feltet benyttes til at kræve høj transmissionshastighed eller høj data pålidelighed. Denne type serviceindikation bruges af gateways til at udvælge de aktuelle transmissionsparametre til det enkelt datanet.

Time to Live. Afgøre levetiden for IP datagrammet, og bliver afgjort af afsenderen. Hver enhed på netværket som passeres af datagrammet, skal dekrementere værdien, og hvis værdien bliver 0, skal IP datagrammet destrueres.

Options. Feltet i IP headeren kan benyttes til at give specifikke informationer om routningen. Eksempelvis kan afsenderen ved at ændre i options feltet afgøre ruten, som IP datagrammet skal tage.

Header Checksum. Benyttes til at teste om IP datagrammet er beskadiget. Hvis checksummen er ugyldig, destrueres datagrammet. Idet felter såsom Time to Live bliver opdateret ved hver enhed på netværket, bliver checksummen også løbende beregnet og ændret.

[Postel, 1992]

IP protokollen indfører ingen form for sikkerhed i forbindelse med kommunikationen, samtidig med at der ikke kan opnås nogen form for tilkendegivelse for, om transmissionen er lykkes. Ud over den simple Header Checksum eksisterer der ingen form for fejlkontrol, ligesom redskaber til flowcontrol heller ikke er tilstede.

Disse omtalte emner skal alle varetages af transport protokollerne, som omtales i det følgende.

E.3 Transmission Control Protokollen

Transmission Control Protokollen (TCP) er beregnet til meget pålidelig kommunikation mellem parter i et packet switched computer netværk. TCP er en forbindelsesorienteret og sikker protokol, der muliggør overførsel af data, som ankommer sikkert og i rigtig rækkefølge. Med henblik på at stille disse services til rådighed, varetager TCP en række områder, som alle er beskrevet i standarden RFC793. Disse er følgende:

Grundlæggende data overførsel. TCP muliggør overførsel af en kontinuert strøm af data i begge retninger. Dette gøres ved at segmentere, at de data som ønskes sendt, er pakker af en given størrelse, der kan acceptere de lavere liggende lag. Generelt bestemmer TCP selv, hvornår dataene sendes, hvilket sker med henblik på opnåelse af maksimal udnyttelse af kommunikationskanalen.

Pålidelighed. Når TCP modtager en række pakker, skal der være mulighed at undersøge om pakker er blevet ødelagt, tabt, duplikeret eller er ankommet i forkert rækkefølge. Det sker ved at give hver pakke et sekvens nummer. Når TCP modtager en pakke med et sekvens nummer, skal den sende en tilkendegivelse tilbage. Sker dette ikke foretages en retransmission hos senderen. Endeligt kan modtageren benytte disse sekvensnumre, til at undersøge om pakkerne er modtaget i rigtig rækkefølge.

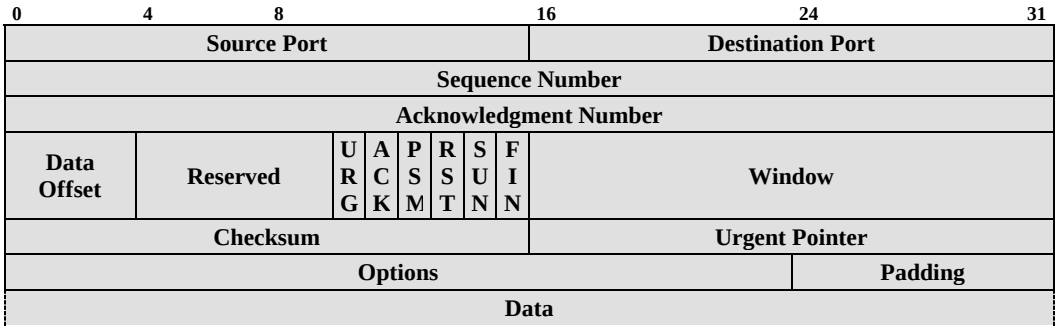
Flow kontrol. Sammen med den føromtalte tilkendegivelse, sender modtageren af en TCP pakke informationer tilbage om hvor meget plads den har tilbage. Mere præcist sender modtageren et vindue med gyldige sekvensnumre, som senderen kan benytte.

Multipleksing. For at tillade forskellige processer på den samme maskine at benytte TCP er det nødvendigt med en form for multipleksing. I praksis indføres dette ved, at TCP giver mulighed for at der kan bruges en række adresser (porte) på den samme maskine. Den port sammenholdt med adressen som opnås fra IP laget, definerer en socket. En socket kan deltage i flere forskellige forbindelser, mens et socket par definerer en forbindelse mellem to maskiner.

Forbindelser. Pålideligheden og kontrol af dataflowet, som er beskrevet ved de ovennævnte punkter, kræver alle, at TCP initialiserer og vedligeholder en række informationer for hver data strøm. Kombinationen af denne information (sockets, sekvens numre og vindue størrelser) kaldes en forbindelse. Når to processer ønsker at kommunikere over et socket par, skal TCP først etablere en forbindelse, samtidig med at forbindelsen nedlægges efter brug. For at oprette denne forbindelse benytter TCP en handshake mekanisme med clock baserede sekvensnumre.

Præcedens og sikkerhed. Brugere af TCP kan endvidere angive præcedens og kræver sikkerhed af deres forbindelse. Dette er dog ikke altid nødvendigt.

For at implementere de egenskaber, som kort er blevet gennemgået, benytter TCP en header i stil med IP, som vedhæftes hver datapakke. Denne er vist ved figur E.3.



Figur E.3: TCP header.

TCP headeren indeholder en række oplysninger, hvoraf følgende omtales kort.

Sequence number. Indeholder et sekvens nummer (32 bit), som specificerer, hvor i datastrømmen pakken tilhører.

Acknowledge number. Indeholder det sekvens nummer (32 bit), som forventes næste gang.

URG, ACK, PSH, RST, SYN, FIN. 6 enkle bits, som benyttes til at indikere forbindelsens nuværende tilstand.

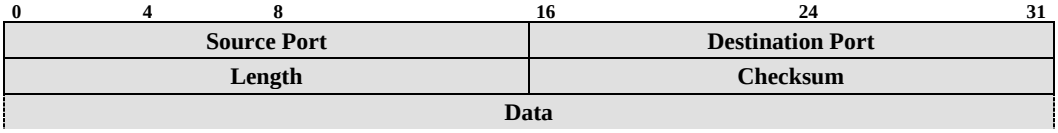
Window. Indeholder 16 bit som fortæller, hvor meget data senderen af pakken vil modtage.

[Postel, 1981]

E.4 User Datagram Protokollen

User Datagram Protokollen (UDP) er beregnet til overførsel af data mellem parter i et packet switched computer netværk. UDP beskriver en procedure til udveksling af data mellem processer med et minimum af protokol mekanismer. I modsætning til TCP er UDP transaktionsorienteret og stiller ingen garanti for at pakkerne modtages i rigtig rækkefølge og uden fejl.

Som følge af dette er UDP headeren noget mindre end TCP headeren, hvilket er medvirkende til den større transmissionshastighed og mindre pålidelighed.



Figur E.4: TCP header.

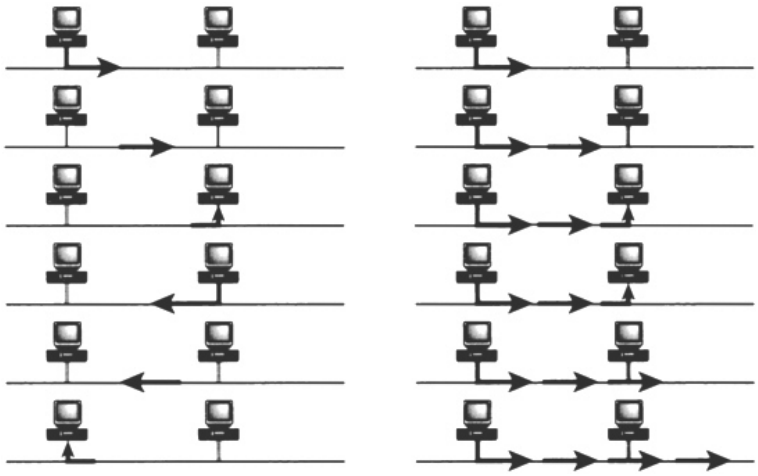
Som vist ved figur E.4 ses, at UDP header indeholder følgende felter.

Source Port. Porten, som datagrammet sendes fra. Er ikke påkrævet, men kan medtages for at vise modtageren hvor pakken kom fra.

Destination Port. Porten på maskinen som datagrammet sendes til.

Length. Længden af datagrammet inklusiv headeren.

Checksum. Indeholder en checksum af de indeholdte data. Beregnes på samme måde som ved TCP headeren.



Figur E.5: Beskrivelse af TCP og UDP. Til venstre er vist netværkskommunikation via TCP, mens der til højre benyttes UDP [Ray, 1999].

[Postel, 1980]

E.5 Klient/server paradigmet

Kommunikation mellem applikationer over datanet i almindelighed samt de dataforbindelse som benyttes i dette projekt, foregår efter en model, som beskrives ved klient/server paradigmet. Hovedprincippet i dette paradigme er, at en server applikation venter passivt på kontakt fra en eller flere klienter, som aktiv kontakter serveren med henblik på at benytte en service, som serveren stiller til rådighed.

Mere generelt kan følgende egenskaber beskrive en klient.

- En vilkårlig applikation kan opfattes som klient, når der er brug for fjernadgang til en anden applikation.
- En klient starter en aktiv forbindelse til en eller flere servere, men kan kun kommunikere med en ad gangen.

En server kan beskrives ved følgende egenskaber.

- En server applikation er et program dedikeret til at yde en service til en eller flere klienter.
- Venter passivt på kontakt fra vilkårlige klienter og accepterer kontakt fra vilkårlige klienter, men tilbyder kun en service.

[Stevens, 1998b]

Interproces kommunikation

F

Formålet med dette kapitel er at beskrive de forskellige metoder, som findes i Linux til implementation af interproces kommunikation (IPC). Indledningsvis beskrives selve proces begrebet, hvorefter de forskellige metoder til interaktion mellem processer kort omtales.

F.1 Processer i Linux

Generelt kan en proces opfattes som et kørende program, der på sekventiel vis udfører en instruktion ad gangen. En proces er dog mere end blot eksekveringen af et program, idet en proces også karakteriseres ved en række parametre, som operativsystemet varetager:

- Status for processen (eks. eksekverende, stoppet el.lign.).
- Identifikationsnummer.
- Værdi af processor registre (herunder bla. program tælleren).
- Informationer som benyttes af kernen til skedulering - eks. omkring prioritering.
- Informationer om processens adresserum (programkode, data og stak).

[Rémy Card, 1997]

Når der benyttes parallelle processer, kan det være nødvendigt at have en form for kommunikation og/eller synkronisering mellem de enkelte processer. Linux introducerer en lang række mere eller mindre sofistikerede metoder og værktøjer til håndtering af denne problemstilling. I det følgende omtales en række metoder, som umiddelbart er tilgængelige i Linux.

F.2 Overblik over IPC i Linux

Kommunikation mellem processer kaldet interproces kommunikation (IPC) baserer sig som oftest på enten fælles variable eller besked systemer. Ved fælles variable metoden, har de processer som ønsker at kommunikere, begge råderet over et hukommelsesområde, som

begge kan skrive og læse fra (dog ikke samtidigt). Besked systemer baserer sig derimod på, at processer sender beskeder via en form for kø, til en anden proces, som efterfølgende kan aflæse beskeden. Hvilken metode, der vælges, er underordnet, idet begge kan benyttes til at implementere samme funktionalitet. [Burns and Wellings, 2001]

Signaler. Er software genererede interrupts, som sendes til en proces på baggrund af en given hændelse. Signaler kan være synkrone (eks. hvis de genereres af fejl i en applikation), men er dog hovedsageligt asynkrone. Processer kan sende signaler til hinanden, men kan også modtage signaler direkte fra kernen på baggrund af et hardware interrupt (eks. bus fejl).

Pipes. Udgør den simpleste form for interproces kommunikation. Metoden består i at forbinde standard output for en proces til standard input for en anden proces, hvorved der muliggøres en unidirektionel kommunikationskanal, som kan benyttes til udveksling af data. Pipes giver ikke mulighed for struktureret kommunikation mellem processer, man kan benyttes til simpel data udveksling.

Beskedkøer. Kan benyttes af to eller flere processer til udveksling af informationer. Afsender processen placerer via et besked modul en meddelelse i køen, som efterfølgende kan læses af en anden proces. Til hver meddelelse kan der gives en type og identifikation, således processer kan udvælge de meddelelser, som er relevante for dem. For at processer kan benytte den fælles kø, er det nødvendigt, at de begge kender til en på forhånd aftalt nøgle (integer værdi).

Semaforer. Er simple integer værdier, der benyttes til kontrol af adgang til delte ressourcer for multiple processer. Til semaforer er knyttet to funktioner - signal og wait. Signal inkrementerer værdien af semaforen, mens wait dekrementerer værdien. Wait har endvidere den egenskab, at den blokerer hvis værdien af semaforen bliver 0 efter dekrementeringen. Ud fra semaforer er det således muligt at styre adgangen til eks. et fælles memory område, hvor to processer ikke kan tillades samtidig adgang.

Fælles hukommelse. Er den hurtigste metode til udveksling af informationer mellem processer. En proces opretter et hukommelsesområde, som andre processer kan få adgang til, hvis de har de rigtige rettigheder. Som ved beskedkøer er det nødvendigt at begge processer kender til en fælles nøgle, for at identificere det fælles område. For at sikre at begge processer ikke skriver til området samtidigt, er det nødvendigt at benytte semaforer til kontrol af dette.

Sockets. Er en af de mest fundamentale begreber i forbindelse med computernetværk og proces kommunikation. Mere præcist er sockets et API (Application Programming Interface), der befinder sig mellem applikationer og netværksprotokol mekanismerne, som operativsystemet stiller til rådighed. To processer kan kommunikerer ved at oprette sockets og sende beskeder imellem disse. Sockets udemærker sig ved, at de kan benytte en lang række protokoller. Som udgangspunkt opdeles disse i tre typer:

Stream sockets. Benyttes ved forbindelsesorienterede protokoller (eks. TCP).

Datagram sockets. Benyttes i forbindelse med transaktionsorienterede protokoller (eks. UDP).

Raw sockets. Benyttes til lowlevel protokoller, som går uden om operativsystemets indbyggede transportprotokoller (eks. IP kommunikation).

[Stevens, 1998a]

I det følgende beskrives mere indgående de metoder, som er valgt benyttet i projektet.

F.2.1 Signaler

Signalhåndtering i Linux muliggør, at processer kan reagere på hændelser skabt af dem selv og andre processer. Hvert signal svarer til en specifik hændelse i systemet, og kan som udgangspunkt genereres på følgende metoder:

- På baggrund af en given **fysisk omstændighed**. Eksempelvis en proces der forsøger at skrive til et uallokeret hukommelsesområde, hvilket bevirker at et flag bliver sat. Dette flag genkendes af kernen, som sender signalet SIGSERV til processen.
- På baggrund af **input fra terminal**. Det kan være brugeren ved keyboardet der taster [CTRL-C], hvorved der generes et SIGINT signal. Dette betyder som udgangspunkt, at processen skal termineres.
- Signaler kan opstå på baggrund af kald af **kill** kommandoen, der kan benyttes til at sende vilkårlige signaler til processer.
- Signaler kan endvidere genereres som følge af **hændelser**, der **kontrolleres** og **overvåges** af kernen. Eksempelvis når en alarm får time-out, hvorved SIGALRM signalet sendes til processen.

For de fleste signaler er det muligt selv at ændre funktionaliteten af programkoden, der eksekveres, når der modtages et bestemt signal. Dette gælder dog ikke SIGKILL (dræb proces) og SIGSTOP (suspend proces). Den funktion, som varetager den nye funktionalitet, der udføres, når et signal modtages, kaldes en signal handler. [Rémy Card, 1997]

I forbindelse med projektet benyttes signaler af kategorien input fra terminal, idet processen som overvåger input fra billedserveren bliver interruptet, når der forefindes ny data til indlæsning. Det specifikke signal, der benyttes til dette er SIGIO. Proceduren i benyttelse af SIGIO i forbindelse med UDP sockets er følgende:

1. En signalhandler installeres for SIGIO signalet.
2. Signaldrevet I/O skal enables for den pågældende UDP socket.
3. Efterfølgende genereres SIGIO signalet, når der modtages et nyt datagram.

[Stevens, 1998a]
Det er således nødvendigt, vha. signal handleren, at foretage den egentlige indlæsning af data fra datagram socketen. Sålænge der indlæses data, skal SIGIO signalet slås fra, således indlæsningen ikke forstyrres. Straks efter indlæsningen er overstået, slås SIGIO til igen, så nye data kan modtages. Det er således ønskeligt af holde signal handlerens størrelse og følgelig eksekveringstiden på et minimum, idet data fra billedserveren ellers kan gå tabt.

F.2.2 Beskedkøer

Beskedkøer kan sammenlignes med et system af postbokse, hvor flere forskellige processer kan skrive og læse data. Beskedkøer identificeres ud fra en beskedkø nøgle, som processer skal kende for at kunne skrive og læse fra køen (givet de tilstrækkelige rettigheder). For hver beskedkø vedligeholder kernen en række parametre, der blandt andet omfatter følgende:

- Rettighedsoplysninger for hvem der må skrive og læse fra køen.
- Pointere til begyndelse og slutning af køen.
- Antallet af bytes samt beskeder på køen.
- Proces ID numre for de processer som seneste har sendt til og læst fra køen.
- Tidspunkter for seneste læsning, skrivning og ændring af køen.

Til oprettelse, styring og brug af beskedkøer, indeholder Linux en række funktioner, der kort omtales i det følgende.

msgget. Benyttes til at oprette nye beskedkøer eller søge efter allerede oprettede beskedkøer.

msgsnd. Benyttes til at sende beskeder til køen.

msgrecv. Benyttes til at modtage beskeder fra køen. En af de mulige parametre til msgrecv er IPC_NOWAIT, som bevirker at funktionen returnerer straks, selvom der ingen data er på køen. Ved at slå dette flag fra, venter processen indtil der kommer data på køen. Dette benyttes i projektet til synkronisering mellem billedserver interfacet og selve strategidelen af klienten.

msgctl. Benyttes til kontrol af køen, eksempelvis ændring af rettigheder eller nedlæggelse af beskedkø.

[Rémy Card, 1997]

F.2.3 Stream og datagram sockets

Fælles for alle typer af sockets i Linux er, at de indledningsvis skal oprettes vha. et kald til funktionen socket(). Parametrene til denne funktion specificerer, hvilken familie af protokoller der ønskes benyttet (eks. IPv4 eller IPv6), samt hvilken type protokol der benyttes (stream, datagram eller raw).

Efterfølgende opsætning af socket forbindelsen er afhængig af, om der er tale om en klient eller server. I det følgende omtales de forskellige trin, som skal foretages i forbindelse med opsætning af en stream server.

Socket. Socket oprettes og der afgøres hvilken protokol familie og typer, der ønskes benyttet.

Bind. Funktionen tildeler en lokal protokol adresse til den oprettede socket. I forbindelse med TCP kan dette være en IP adresse, en port eller begge dele.

Listen. Når der indledningsvis oprettes en socket, antages denne at være aktiv, hvilket vil sige, at en klient kan benytte den til at forbinde sig til en server. Ved at benytte listen funktionen, konverteres den oprettede socket til en passive socket, hvilket indikerer at kernen skal acceptere indkommende forbindelser til denne socket.

Accept. benyttes til at returnere den næste fuldførte forbindelse fra forbindelseskøen. Hvis forbindelseskøen er tom, venter processen indtil en klient connecter.

Er der tale om en stream klient, udføres følgende trin:

Socket. Socket oprettes og der afgøres hvilken protokolfamilie og type, der ønskes benyttet.

Connect. Benyttes til at forbinde en stream klient til en stream server. Parametre til funktionen er IP adresse og port nummer for serveren. Klienten behøver ikke som serveren at benytte bind funktionen, idet kernen selv vælger en port samt IP adresse.

Er der istedet tale om klient og server, som kommunikerer via datagram sockets, skal disse igennem et færre antal trin for opsætning - nærmere bestemt benyttes listen, accept ikke for serveren, mens connect udelades for klienten. Efterfølgende skal parterne således ved hver transaktion angive, hvor beskeden skal sendes hen. Mere præcist benyttes funktionerne **recvfrom** og **sendto** fremfor **recv** og **send**.

Litteratur

Borenstein, J. [1996]. Where am i?
*Vedlagt på CD

Burns, A. and Wellings, A. [2001]. *Real-Time Systems and Programming Languages*, 3rd edn, Addison-Wesley, Essex, England. ISBN: 0-201-72988-1.

Gonzalez, R. E. W. . R. C. [1992]. *Digital image processing*, Addison-Wesley Publishing Inc. ISBN:0-201-50803-6.

Jones, A. M. F. . J. L. [1993]. *Mobile Robots*, A K Peters, Wellesley, USA. ISBN:1-568-81011-3.

Lewine, D. [1991]. *POSIX Programmers Guide*, 1st edn, O'Reilly and Associates, Inc., USA. ISBN: 0-937175-73-0.

Microchip-16F87X [2001]. *PIC16F87X Data Sheet*, Microchip Technology Inc., USA.

Microchip-Midrange [1997]. *PICmicro^{tm} Mid-Range MCU Family Reference Manual*, Microchip Technology Inc., USA.

National-Semiconductors [2000]. *LM78LXX Series*, National, USA.

Nelson, R. [2001]. Lego mindstorms internals.
*HTTP://www.crynwr.com/lego-robotics/

Postel, J. [1980]. *RFC768-UDP*, J. Postel.

Postel, J. [1981]. *RFC793-TCP*, J. Postel.

Postel, J. [1992]. *RFC1340-IP*, J. Postel.

Radiometrix [2001]. *433MHz high speed FM radio transceiver module*, Radiometrix Ltd, England.

Ray, J. [1999]. *Special Edition Using TCP/IP*, 1st edn, Que Corporation, USA. ISBN: 0-7897-1897-9.

Rémy Card, r. D. o. F. M. [1997]. *The Linux Kernel book*, 1st edn, John Wiley and Sons, Chichester, England. ISBN: 0-471-98141-9.

Stevens, R. W. [1998a]. *UNIX Network Programming - Interprocess Communications*, 2nd edn, Prentice Hall PTR, Upper Saddle River, USA. ISBN: 0-13-081081-9.

Stevens, R. W. [1998b]. *UNIX Network Programming - Networking APIs*, 2nd edn, Prentice Hall PTR, Upper Saddle River, USA. ISBN: 0-13-490012-x.

STMicroelectronics [2000]. *L298 DUAL FULL-BRIDGE DRIVER*, STMicroelectronics, Italien.