AALBORG UNIVERSITET
Institut for elektroniske systemer
Datateknik

Titel: Automatiseret lagerdrift

Tema: Mikroprocessorsystemer

Projektperiode:
P3, efterårssemesteret 2005

Projektgruppe:
Projektgruppe 352

Deltagere:
Anders Grauballe
Mikkel Gade Jensen
Janne Dahl Rasmussen
Ulrik Wilken Rasmussen
Michael Rimestad
Morten Risager

Vejleder:
Karsten Jensen

Oplagstal: 9

Sidetal: xxx

Bilagsantal og –art: 1 stk. CD

Afsluttet den 19.12.05

Synopsis:

Rapporten beskriver udviklingen og implementeringen af et styresystem til en lagerrobot. Styresystemet er udviklet til microprocessoren MSP430F149.
For at teste systemet bygges en lagerrobot, af LEGO Mindstorms styret af mikroprocessoren. Derudover defineres et testlager, hvor robotten navigerer rundt ved hjælp af streger på gulvet. Forbindelsen til robotten er en standard RS232 seriel forbindelse over bluetooth. Robottens formål er at kunne flytte pallerne rundt på forskellige pladser på lageret, hvor der er ikke taget højde for eventuelle forhindringer. Udover automatisk navigation, er det desuden muligt at kontrollere robotten manuelt. Til dette er der lavet en brugerflade i GTK, der muliggør denne styring, samt tilføje og fjerne automatiske opgaver. Kravspecifikationen bliver opfyldt på de de fleste punkter, hvilket kan bekræftes ved accepttesten.

Forord

Denne rapport er udarbejdet af projektgruppe 352, 3. semester datateknik, Aalborg Universitet i perioden 2/9 til 19/12 2005. Det overordnede emne for semestret er "Mikroprocessorsystemer" og herunder er der valgt "Styresystem til lagerrobot" for dette projekt.

Målgruppen for denne rapport er andre studerende eller interesserede med samme tekniske niveau som studerende på tredje semester på datateknik. Derudover henvender rapporten sig til projektgruppens vejleder og censor.

Figurer og tabeller er i rapporten nummereret efter kapitel og nummer figur eller tabel i det kapitel, f.eks. figur 3.5 er den 5. figur i kapitel 3. Kildehenvisninger er i rapporten skrevet med det første 3 bogstaver i forfatterens efternavn evt. efterfulgt af årstallet for udgivelsen af litteraturen, f.eks. [Tan05] (Tanenbaum 2005), kildelisten findes bagerst i rapporten før diverse appendiks.

Læsevejledning:

- Kode eller syntaks: `Ser sådan ud`
- Ord til ordliste: Ser ud på følgende måde, "ord*" (efterfulgt af stjerne)
- Første gang der anvendes et nyt ord, er det skrevet fuldt ud, fulgt af den forkortelse, der vil blive anvendt efterfølgende

Der er vedlagt en CD, som indeholder følgende:

- Videooptagelser af test
- Billeder af robotten med MSP430F149
- Denne rapport i PDF-format
- Kildekode til mikroprocessoren

- PC-program til Linux, samt kildekode til dette
- Internetsider der er brugt som litteratur

Aalborg Universitet d. 19 december 2005.

Anders Grauballe	Ulrik Wilken Rasmussen
Mikkel Gade Jensen	Michael Rimestad
Janne Dahl Rasmussen	Morten Risager

Indhold

Kapitel 1

Indledning

Virksomheder efterspørger i dag i stigende grad nye metoder, der kan øge effektiviteten. Dette skyldes bl.a. det øgede konkurrenceniveau som f.eks. globaliseringen medfører. Det er vigtigt for et lille land som Danmark at følge med udviklingen og gerne være et skridt foran. For at bevare en god position på det internationale marked, er der flere midler som kan tages i brug.

Den store forskel i lønniveau, mellem Europa og Asien, er medvirkende til at stadig flere virksomheder flytter sine aktiviteter mod øst. Dette kaldes outsourcing og er en af de løsningsmodeller, der medfører en konkurrencedygtig pris på den vare eller ydelse som virksomheden tilbyder. Desværre betyder det også, at der forsvinder et stort antal arbejdspladser fra det danske erhvervsliv.

En anden løsningsmodel kunne være at automatisere en produktion eller en arbejdsgang, indenfor virksomheden, ved brug af robotter. Dette betyder at en given arbejdsrutine kan udføres hurtigere, billigere og mere præcist. Umiddelbart ser det ud til at denne metode også vil medføre et fald i antallet af arbejdspladser. Det betyder dog at virksomhederne lader deres aktiviteter forblive i landet og derfor vil en del af arbejdspladserne blive bevaret i Danmark. Formanden for Dansk Robot Forening, Leif Dalum, har udtalt:

> *"Med investeringer i automatisering kan man styrke sin konkurrenceevne så meget, at man kan bevare sin produktion og dermed arbejdspladser i forhold til lande, hvor arbejdslønningerne er meget lave."*[Bør99]

Dette bakkes op af Sebastian Swiatecki, som er journalist hos Ingeniøren. Han skriver at investeringen i robotteknologi er stigende og at disse investeringer også skyldes en stadig lavere pris på automatisering.[Swi04]

Der findes naturligvis også ulemper ved denne automatisering. En af ulemperne er at en robot stadig mangler den intelligens og fleksibilitet som en produktionsmedarbejder har, men det må dog vurderes at der er et væsentligt potentiale i robotter både nu og i fremtiden til at udføre praktiske rutineprægede opgaver.

Hovedparten af de robotter, som bliver implementeret i dag, er produktionsrobotter, som har en bestemt funktion ved samlebåndet. Disse er relativt simple at styre da de er stationære og skal udføre samme bevægelse igen og igen. Hvis der ønskes en mere fleksibel robot, der kan udføre en række forskellige opgaver, kræves der en form for kunstig intelligens. En sådan autonom robot kan ideelt set udføre samme praktiske opgaver som et menneske.

I en virksomhed, hvor produktionen allerede er automatiseret, kan det næste skridt være at automatisere lageret. Her er der brug for at varerne kan flyttes rundt på en effektiv måde. Derfor antages det, at en automatisk lagerrobot kan effektivisere det arbejde, der ellers bliver udført af en traditionel truck og en truckfører. En sådan robot skal kunne udføre opgaver som f.eks., at flytte paller fra en plads på lageret til en anden. Ud fra ønsket om at udvikle en autonom lagerrobot, fremsættes følgende initierende problem:

Hvordan konstrueres og programmeres et system til en lagerrobot, så opgaverne på et lager kan automatiseres?

Til dette projekt er der stillet følgende udviklingsværktøjer til rådighed:

- Texas Instruments MSP430F149 mikroprocessor, som er emnet for 3. semester. Denne skal programmeres til at styre robotten.
- Merlin Lab, Embedded Bluetooth Modul til seriel kommunikation.
- LEGO Mindstorms byggesæt, der skal bruges til konstruering af prototypen til en lagerrobot. Dette gør det let at bygge prototypen, så der kan lægges vægt på udviklingen af styresystemet.

Kapitel 2

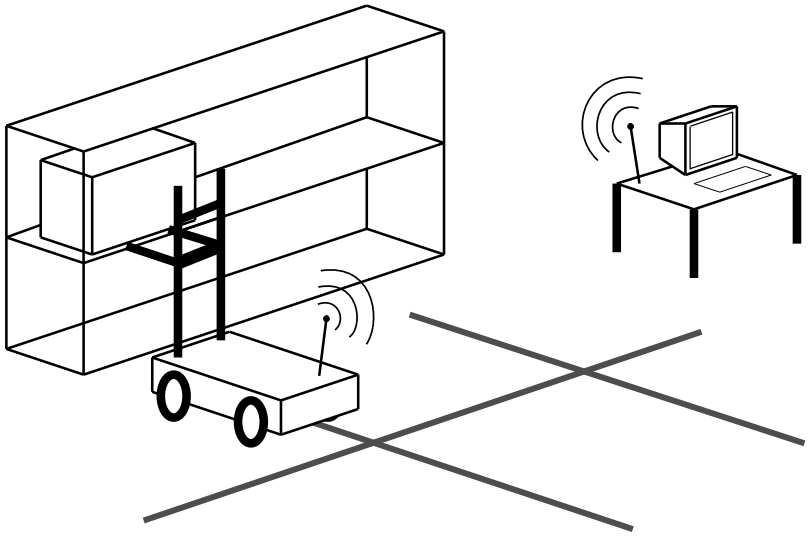
Systembeskrivelse

For at løse den problemstilling, der er opstillet sidst i indledningen, skal der udvikles software til mikroprocessoren, MSP430F149, så den kan styre en lagerrobot. Lagerrobotten skal kunne udføre de samme opgaver som en almindelig truck med en truckfører, det vil sige at den skal kunne hente en bestemt palle et bestemt sted på lageret og aflevere den et andet. For at mikroprocessoren i princippet kan fungere som en truckfører skal den kunne navigere rundt på lageret, samt modtage de opgaver den skal udføre.

Mikroprocessoren skal modtage data om de instruktioner den skal udføre fra en PC, som står på lageret. Denne skal også overvåge systemet og informere brugeren af systemet om lagerrobottens placering, nuværende instruktion og eventuelle fejlmeddelelser.

For at lagerrobotten kan navigere rundt, er der til dette system valgt at arbejde med lyssensorer der kan registrere streger på gulvet. For at lagerrobotten skal kunne hente en palle, skal der monteres både tryksensorer, der registrerer om der er en palle på gaflen og en triptæller der registrerer hvor højt gaflen er hævet. Som udgangspunkt skal lagerrobotten ikke håndtere forhindringer som f.eks. paller eller personer. Dette er gjort for at lægge fokus på mikroprocessoren og dermed lagerrobottens opbygning, uden for megen koncentration på det omkringliggende miljø. En skitse af systemet kan ses på figur 2.1 på den følgende side.

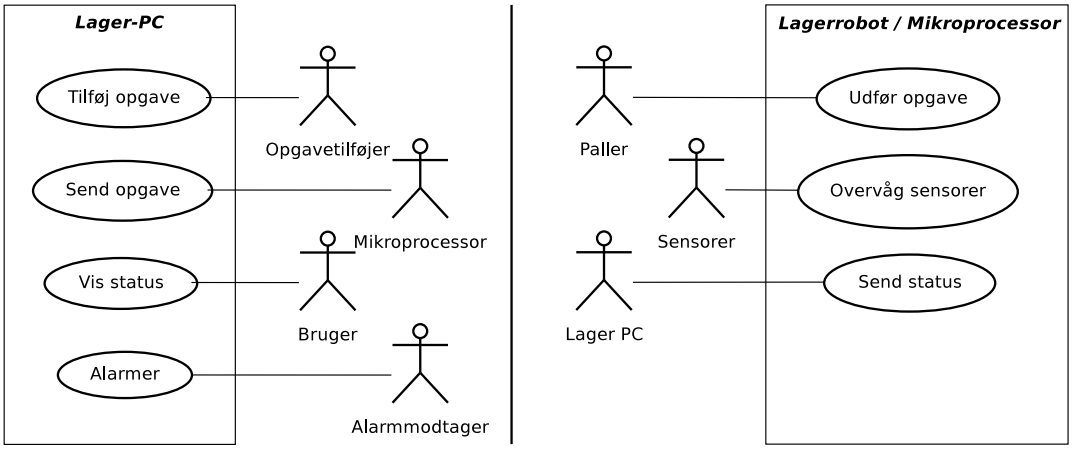
Der tages udgangspunkt i UML (Unified Modelling Language). SPU (Struktureret Program-Udvikling)[BS02] anvendes som udviklingsmodel, men modellen er tilpasset til behovet i projektet. Derudover anvendes en Use case-analyse, for at give et overblik over systemets funktionalitet.



Figur 2.1: Figuren viser en skitse af det ønskede system. Til venstre kører lagerrobotten og flytter paller, mens lager-PC'en til højre sender nye instruktioner til robotten.

2.1 Use Cases

Der er udarbejdet en Use Case-analyse, som overordnet beskriver systemets funktionaliteter. Denne kan ses på figur 2.2.



Figur 2.2: Use case for det samlede system opdelt i lager-PC'en og lagerrobotten/mikroprocessoren

Tilføj opgave

Målbeskrivelse
Lager-PC'en modtager opgavedata fra en opgavetilføjer og denne opgave tilføjes til en database.

Normalscenarie

- Der indtastes data på lager-PC'en om hvilken palle, destination og evt. prioritet den tilføjede opgave har. Der arbejdes med 3 prioriteter: Høj, standard og lav.
- Opgaven tilføjes til databasen, som sorteres efter prioritet og kronologi.

Undtagelser

- Hvis ikke prioritering specificeres, sættes denne til standard.
- Opgaven kan ikke tilføjes hvis pallens placering eller destination ikke er angivet.

Send opgave

Målbeskrivelse
Næste opgave hentes i databasen, dekodes til simple navigationsinstruktioner og sendes til lagerrobotten.

Normalscenarie

- Den næste opgave i databasen indlæses.
- Opgaven dekodes til følgende simple instruktioner:
  - Kør ligeud til næste kryds.
  - Bak til foregående kryds.
  - Drej om egen akse 90 grader til højre.
  - Drej om egen akse 90 grader til venstre.
  - Løft pallen fra niveau x.
  - Sæt pallen på niveau y.

- Instruktionerne sendes til lagerrobotten.

Undtagelser
Der skal vises en fejlmeddelelse, hvis der ikke er forbindelse til lagerrobotten.

Vis status

Målbeskrivelse
Den modtagede status skal vises på lager-PC'ens skærm, så brugeren af systemet kan se hvad lagerrobotten foretager sig.

Normalscenarie

- Der modtages en statusmeddelelse fra lagerrobotten.
- En liste med statusmeddelelser vises på lager-PC'ens skærm.

Undtagelser
Hvis der ikke modtages status fra robotten i max 5 sekunder, skal der vises en alarmmeddelelse på lager-PC'ens skærm.

Alarmér

Målbeskrivelse
Alarmen bruges i de tilfælde hvor der er opstår en fejl. Alarmmodtageren får et signal om alarmen, og tilkaldes for at undersøge fejlen nærmere. Fejl som kræver at alarmmodtageren skal tilkaldes:

- Robotten kommer væk fra dens rute og ikke kan finde tilbage igen.
- Robottens batteriniveau er så lavt, at den ikke selv har mulighed for at køre til opladning.

Normalscenarie

- Alarmtiltilstand opstår.
- Alarmen sendes til alarmmodtager.

- Alarmen skal indeholde problemtypen og hvad robottens status er.
- Alarmmodtageren løser problemet.
- Alarmen stoppes og robotten arbejder videre.

Undtagelser

- Hvis alarmmodtageren ikke kommer og stopper alarmen indenfor en angivet periode gensendes alarmen.
- Hvis der er flere alarmer på samme tid, sendes alle alarmer til alarmmodtageren.

Udfør opgave

Målbeskrivelse
Robotten skal køre hen til den palle der skal flyttes, og flytte den til dens nye destination. Dette gøres ved at styre motorer og gaffel, ud fra de modtagne instruktioner fra lager-PC'en og input fra sensorer.

Normalscenarie

- Der modtages simple instruktioner.
- Instruktionerne udføres.

Undtagelser

- Hvis der ikke er nogle instruktioner, skal robotten køre til ladestationen og lade op.
- Hvis robotten er ved at løbe tør for strøm, skal den ikke udføre en ny instruktion, men køre over for at lade op, samt sende status til lager-PC'en.

Overvåg sensorer

Målbeskrivelse
Lagerrobotten skal kunne følge forskellige streger på gulvet, for at gøre det skal den

reagere på forskelligt input fra lyssensorerne, der er placeret under den. Derudover skal robotten vide om der er en palle på gaflen eller ej, derfor overvåges også en tryksensor.

Normalscenarie

- Sensorerne bliver overvåget med konstant interval og når status for de forskellige sensorer skal bruges, bliver de hentet af mikroprocessoren.
- Ved aktivering af tryksensor, sendes status til lager-PC'en.
- Tryksensoren aktiveres kun når der er en palle på gaflen.
- Lyssensorerne skal registrere de streger robotten skal følge.

Undtagelser

- Hvis lyssensorerne ikke registrerer en streg skal der sendes en fejlmeddelelse til lager-PC'en og robotten skal stoppe.

Send status

Målbeskrivelse
Lagerrobotten skal sende status til lager-PC'en om følgende:

- Klar til at modtage instruktioner.
- Batteriniveau.
- Udført instruktion.
- Sensorernes status.
- Alarmer.

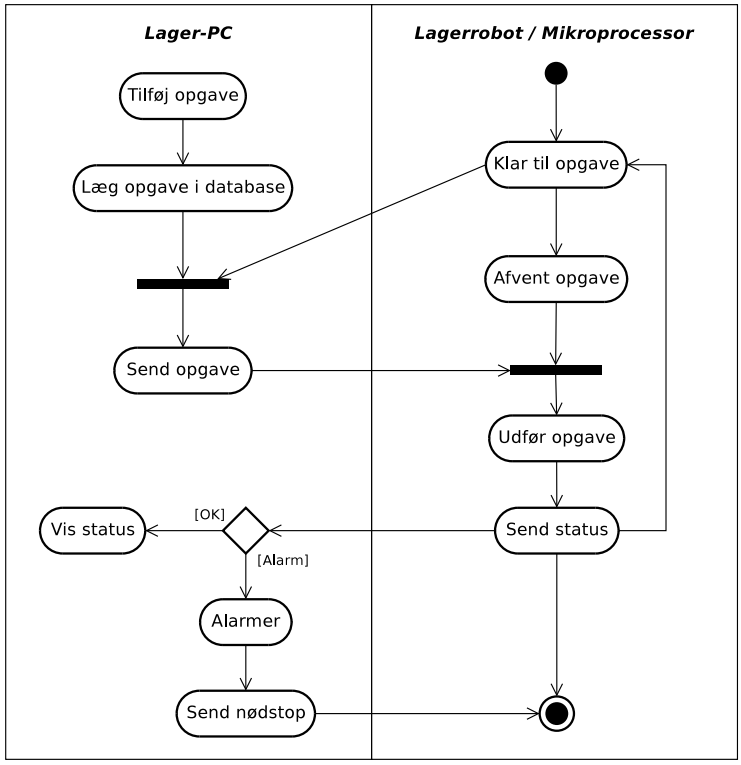
Normalscenarie

- Lagerrobotten har udført en instruktion eller der er udløst en alarm.
- Lagerrobotten sender en meddelelse til lager-PC'en.
- Meddelelsen vises på lager-PC'ens skærm.

Undtagelser

- Hvis der ikke kan kommunikeres med lager-PC'en skal robotten stoppe.

Ud fra use case analysen udarbejdes et aktivitetsdiagram, der kan ses på figur 2.3. Det giver et overblik over flowet i systemets arbejdsrutine.



Figur 2.3: Aktivitetsdiagram som viser det overordnede flow i systemets arbejdsrutine. De omtalte use cases er indsat som aktiviteter.

2.2 Kravspecifikation

I det følgende afsnit opstilles kravspecifikationen for det ønskede system. Den er delt op i krav til lager-PC'en, krav til mikroprocessoren og krav til hardware.

2.2.1 Krav til lager-PC'en

Formålene med lager-PC'en er at afvikle den beregningstunge software, så som ruteberegning, der ikke er hensigtsmæssig at afvikle på mikroprocessoren. Lager-PC'en skal give brugeren mulighed for at indtaste opgaver til robotten samt at overvåge hvad den foretager sig. Det er også lager-PC'en der skal stå for at oprette forbindelsen til mikroprocessoren.

Tilføj opgave
Lagerrobottens opgaver skal overordnet håndteres på lager-PC'en. Dette gøres ved at brugeren trykker på en knap der hedder "Add Task", hvorefter en dialog boks skal komme frem hvor brugeren skal kunne specificere følgende:

- Pallens nuværende placering, hvor den skal flyttes fra, givet ved hyldeplads og niveau.
- Pallens destination på lageret, givet ved hyldeplads og niveau.
- Prioritering af opgaven (høj, standard eller lav).

Hyldepladsen skal gives ved et nummer, og niveau skal gives som A eller B, for gulvet eller første etage. Hyldeplads og niveau skal kunne vælges fra en liste. Dette skal gøres for at sikre at brugeren ikke indtaster en plads på lageret der ikke eksisterer. Det skal også være muligt at fjerne opgaver fra databasen ved at trykke på knappen "Remove Task", og derefter vælge den opgave man ønsker fjernet. Derudover skal det også være muligt at følge med i hvor langt robotten er kommet i databasen.

Send opgave
Når der er tilføjet en opgave til databasen og mikroprocessoren er klar til at modtage en opgave, skal den næste opgave i databasen sendes over til lagerrobotten. Dette gøres ved at splitte opgaven op i en række simple instruktioner, der skal sendes til mikroprocessoren som en tekststreng. Strengen skal have følgende format:

`$TASK,`*`instruktionsnr`*`,`*`instruktionstype`*`,`*`parameter`*`,`

"Instruktionsnr" skal være et tal mellem 0 og 999. Der skal altid sendes 3 cifre, det vil sige at hvis instruktions nummeret er 10, skal det skrives som 010. Instruktionerne skal have et nummer, som er én større end den sidste instruktion der er blevet sendt. De forskellige instruktioner kan ses i tabel 2.1.

intruktionstype	**parameter**
FORWARD	antal
BACK	antal
TURN	retning
PALLET,GET	niveau
PALLET,PUT	niveau

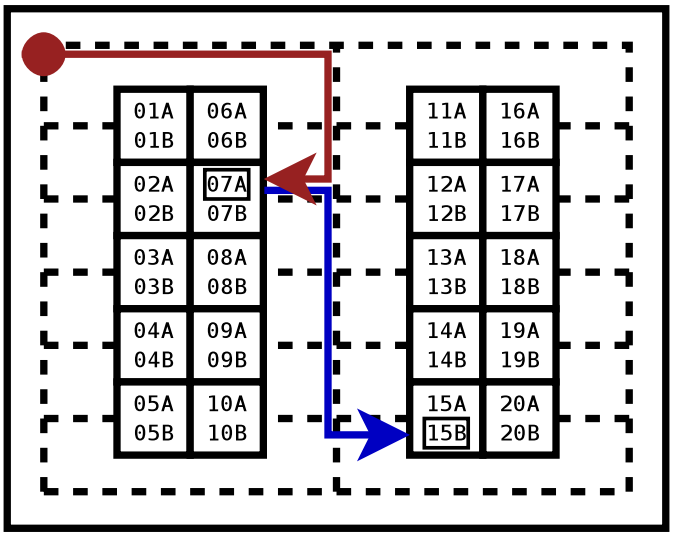
Tabel 2.1: Instruktioner som robotten skal modtage med tilhørende parametre. "Antal" skal være et tal mellem 1 og 9, "retning" skal være enten "R" for højre eller "L" for venstre og nivau skal være "A" for gulvet eller "B" for første etage

Et eksempel på en opgaves instruktionsopdeling kan være som følger:

- `$TASK,001,FORWARD,1,`
- `$TASK,002,TURN,R,`
- `$TASK,003,FORWARD,2,`
- `$TASK,004,TURN,R,`
- `$TASK,005,PALLET,GET,A,`
- `$TASK,006,BACK,1,`
- `$TASK,007,TURN,L,`
- `$TASK,008,FORWARD,3,`
- `$TASK,009,TURN,L,`
- `$TASK,010,PALLET,PUT,B,`

Ved at sende ovenstående instruktioner til robotten, vil den udføre den opgave der er vist på figur 2.4 på den følgende side.

Vis status
Denne funktion skal vise brugeren hvad robotten fortager sig og give brugeren mulighed for at se hvad status er for robottens sensorer og batteri. Status skal på lager-PC'en vises under menupunktet "Status bar". Data modtaget og sendt til lagerrobotten skal henholdsvis vises i et input- og outputvindue.



Figur 2.4: Eksempel på en opgave hvor lagerrobotten skal hente en palle fra plads 7A og køre den til 15B

- Det skal være muligt at se følgende af robottens status.
  - Hvilken instruktion robotten er i gang med.
  - Lyssensorer.
  - Batteristatus.
- Lagerrobottens placering skal vises på et kort over lageret, ved at vise hvilket punkt robotten sidst er kørt forbi samt hvilken retning den har.

Alarmér
I tilfælde af fejl som lagerrobotten ikke selv kan udbedre, skal der vises hvilken fejl der er opstået. Fejl defineres som fejl i udførelse af en instruktion. Disse fejl opstår når robotten ikke detekterer det forventede.

Ved fejl skal lagerrobotten:

- Stoppe.
- Slå over til manuel styring.

Det skal være brugerens ansvar at udbedre fejlen og derefter starte robotten igen.

2.2.2 Krav til mikroprocessoren

Udfør opgave
Mikroprocessoren modtager et antal instruktionsstrenge fra lager-PC'en. Disse lægges ind i hukommelsen, hvorefter de udføres en ad gangen. Der skal være plads til minimum 20 instruktioner på mikroprocessoren.

Lagerrobotten skal udføre instruktionerne på følgende måde:

- ***FORWARD***: Robotten skal følge en streg frem til det næste tværgående streg eller til lyssensorerne registrere advarselsfarven (se " Overvåg sensorer").
- ***BACK***: Robotten skal bakke til forrige kryds.
- ***TURN***: Robotten skal dreje om sin egen akse til den side der er angivet i instruktionen. Den skal dreje en kvart omgang og derefter stoppe.
- ***PALLET,GET***: Robotten skal indstille gaflens højde til det niveau der er angivet i instruktionen, køre frem til pallen er på, hæve gaflen og køre tilbage.
- ***PALLET,PUT***: Robotten skal indstille gaflens højde til det niveau der er angivet i instruktionen, køre frem til pallens destination, stille pallen og køre tilbage.

Overvåg sensorer

- Robotten skal kunne registrere 3 forskellige farver på gulvet.
    1. gulvets farve.
    2. stregens farve.
    3. advarselsfarve.
- Under kørsel skal der hentes og testes på information fra sensorerne minimum 100 gange i sekundet, for at sikre at alle ændringer registreres.
- Lagerrobotten skal have en omdrejningstæller på hejset der registrerer hvor højt gaflen er hævet, den tælles op/ned hver gang der køres med gaflen.
- Det skal kunne registres om der er en palle på gaflen eller ej.

Send status
Mikroprocessoren skal sende status til lager-PC'en under kørsel.

Der skal sendes følgende status:

- Når en instruktion er udført.
- Når en ny instruktion er påbegyndt.
- Alarmer.

Følgende status skal sendes minimum 1 gang/sekund:

- Status på sensorer.
- Batteristatus.

2.2.3 Krav til hardware

I dette afsnit beskrives kravene til de hardware-enheder, som skal kobles til mikroprocessoren.

Sensorer

- Lyssensorer.
  - Skal ikke kunne påvirkes af lyskilder fra omgivelserne.
  - Output fra disse skal være en spænding mellem 0 og 2,5 V.
- Tryksensor.
  - Skal kunne registrere når der er en palle på gaflen.
- Triptæller.
  - Skal give et signal, hvor der minimum er 1 puls/cm, når der køres op eller ned med gaflerne.

Strømforsyning

- Skal kunne levere tilstrækkelig spænding til at forsyne 9 V jævnstrømsmotorer.
- Skal være et batteri.

Motorer

- Skal være jævnstrømsmotorer.
- Skal have mulighed for at kunne køre begge veje.
- Skal kunne trække robotten og løfte gaflen.

2.3 Accepttestspecifikation

I det følgende afsnit beskrives accepttestspecifikationen, som er opstillet, for at teste om kravspecifikationen opfyldes for det ønskede system. Strukturen for denne test er bygget op efter kravspecifikationen. Efter de enkelte testpunkter beskrives der hvordan testen vil blive udført i praksis.

Lager-PC

Til at starte med tændes robotten og PC softwaren startes.

Tilføj opgave

1. PC programmet testes for om der kan tilføjes en ny opgave.
   (a) Der trykkes på knappen “Add Task” og der skal komme en dialogboks frem.
   (b) I dialogboksen specificeres pallens nuværende placering og destination i forhold til hyldeplads, ved nummer og niveau samt hvor høj prioritet opgaven har.
   (c) Når de enkelte felter er udfyldt, trykkes på knappen “Ok” og opgaven tilføjes til databasen, hvilket kan ses i boksen “Task Queue”.
2. PC programmet testes for om det muligt at fjerne en opgave fra databasen.
   (a) I boksen “Task Queue” vælges en opgave og der testes for om denne forsvinder, hvis der trykkes på knappen “Remove Task”.
3. PC programmet testes for om det kan vise status i afvikling af opgaver.
   (a) Når en opgave er udført, kontrolleres der om opgaven får et kryds i feltet “Done”.

Send opgave

1. Det verificeres at PC programmet kan konstruere ruten mellem nuværende og kommende placering.
   (a) Der vælges en opgave i boksen “Task Queue”, hvorefter der trykkes på knappen “Details”. Det kontrolleres at instruktionslisten indeholder korrekte instruktioner.

2. Hver gyldig instruktion med en gyldig parameter og instruktionsnummer, sendes til robotten. Det verificeres at robotten godkender instruktionen.

   (a) I outputvinduet kontrolleres at den rigtige streng genereres. I inputvinduet kontrolleres at robotten returnerer "Task added".

3. PC programmet testes for om det modtager en fejl fra robotten, hvis der sendes en ugyldig instruktion med ugyldig parameter eller instruktionsnummer.

   (a) Fra et terminalvindue sendes en streng med ugyldig instruktionstype, hvis der modtages "Task not valid" bekræfter dette at instruktionen er ugyldig.

4. Fra terminalvinduet sendes der en tilfældig tekststreng, uden "$", til lagerrobotten. Det kontrolleres at der ikke modtages svar og robotten ikke reagerer.

Vis status

1. Under udførelse af instruktioner med robotten, kontrolleres det at det er muligt at se robottens status.

   (a) I PC programmet kontrolleres de viste værdier i "Status bar". Det kontrolleres at lyssensorerne, batterispænding og gaflens position stemmer overens med de faktiske værdier som robotten har.

2. I PC programmet kontrolleres om robottens position på kortet passer med robottens faktiske position.

Alarmér

1. Det kontrolleres at PC programmet reagerer korrekt hvis der modtages en alarm. Hvis robotten placeres udenfor stregen eller skubbes af stregen, kontrolleres det, at:

   (a) Der modtages en fejl fra robotten.

   (b) Robotten stopper.

   (c) Robotten slår over til manuel styring.

   Det kontrolleres om ovenstående instruktioner udføres og at der i PC programmet modtages en fejlmeddelelse i inputvinduet og der fremkommer en dialogboks med en fejlbeskrivelse.

Mikroprocessoren

Udfør opgave

1. Det verificeres at robotten kan modtage op til 20 instruktioner.
   (a) Igennem PC programmets “Test run’s” sendes 20 instruktioner i forlængelse af hinanden, uden at der modtages fejl.
2. Det verificeres af robotten udfører det forventede ved hver instruktion.
   (a) Det kontrolleres at der efter hver udført instruktion modtages “Task done” i inputvinduet og at robotten fysisk har udført instruktionen.

Overvåg sensorer

1. Der kontrolleres om sensorerne fungerer korrekt.
   (a) Det verificeres at robotten kan følge en streg på gulvet, ved at sende instruktionen “Forward” fra PC programmet.
   (b) Det verificeres at robotten kan stoppe midt i et kryds, ved at sende instruktionen “Forward” og kontrollere at robotten sender “Task done” og stopper ved et kryds.
   (c) Der kontrolleres om robotten kan registrere forskel på gulvfarve, stregfarve og advarselsfarve. Ved manuelt at placere robotten på de tre forskellige farver, kontrolleres det at der vises de rigtige farver under “Status bar” i PC programmet.
   (d) Der kontrolleres om robotten stopper, når der registreres en advarselsfarve. Robotten placeres før en advarselsfarve og sættes til at køre fremad med “Forward”. Det kontrolleres at robotten stopper ved advarselsfarven og der modtages “Task done” i inputvinduet.
   (e) Der kontrolleres at robotten tester på lyssensorerne minimum 100 gange i sekundet. Det kontrolleres ved at skrive et tegn ud til lager-PC’en, hver gang der testes på lyssensorerne.
   (f) Der kontrolleres om robotten kan registrere gaflens højde. Dette kontrolleres ved se værdien for “Lift” under “Status bar” i PC programmet. Denne værdi sammenlignes med gaflens faktiske højde.
   (g) Det kontrolleres at robotten kan registrere at der er en palle på gaflen. Robotten sættes til at hente en palle ved tryk på knappen “Take pallet from”. Når pallen er på gaflen kontrolleres det at robotten stopper med at køre frem.

Send status

1. Der kontrolleres at status sendes korrekt.

    (a) Det verificeres at robotten sender sin status minimum 1 gang i sekundet. Det kontrolleres på lager-PC'en at der modtages status med 1 sekunds interval.

    (b) Det kontrolleres at robotten sender status når den påbegynder en ny instruktion og når instruktionen er udført. Dette kontrolleres ved at der vises "Task added" i inputvinduet ved ny instruktion og "Task done" når instruktionen er udført.

    (c) Det verificeres at robotten sender en alarm, når der opstår en fejl. Ved at placere eller skubbe robotten udenfor stregen, kontrolleres det om der kommer en fejlmeddelelse i inputvinduet.

Hardware

1. Der kontrolleres at hardwaren fungerer korrekt.

    (a) Det kontrolleres at lyssensorerne ikke bliver påvirket af andre lyskilder. Dette testes ved at slukke og tænde lyset, og det kontrolleres at spændingen er den samme.

    (b) Det kontrolleres at lyssensorerne giver et output på 0-2,5 V og tryksensorer samt triptæller giver et output på enten 0 eller 3,3 V.

    (c) Det kontrolleres at triptælleren minimum giver 1 puls/cm. Dette testes ved at se på værdien "Lift" i "Status bar"på lager-PC'en og sammenligne med den fysiske højde.

    (d) Det kontrolleres at batteriet minimum giver en spændingsforsyning på 9 V. Dette testes ved at måle spændingen med et voltmeter.

    (e) Det kontrolleres at motorerne kører på jævnstrøm, kan køre begge veje og kan trække robotten samt gaflen. Dette testes ved at tilføre motorerne jævnstrøm til motorerne og se om de kan trække robotten og løfte gaflen.

2.4 Opbygning af testlager

Som testmiljø for systemet, skal der opbygges en model af et lager, som lagerrobotten kan køre rundt på. Dette testlager vil blive opbygget af moduler så det, alt afhængig af den ønskede størrelse på lageret, kan konstrueres ved hjælp af et eller flere moduler. På den måde vil det være lettere at udbygge lageret uden at lave større ændringer i systemets software.

Som det fremgår af use case analysen, afsnit 2.1 på side 11, skal lageret i projektet bygges i to niveauer. Det betyder, at systemet senere let kan udbygges med et ekstra niveau ved små ændringer i softwaren. Afstanden mellem varereolerne skal være så stor, at lagerrobotten kan dreje om sin egen akse uden at ramme reolerne. På lageret skal der udover reolerne være en plads, hvor pallerne der skal til og fra lageret, kan hentes og stilles af lagerrobotten. På figur 2.4 på side 18 ses en skitse over hvordan et testlager kan opbygges. Ud fra dette kan disse krav stilles til lageret.

Krav til opbygning af testlager:

- Skal være opbygget af moduler, således at det let kan gøres større eller mindre.
- Hylderne skal være i to niveauer, da det gør det enkelt at udbygge systemet med endnu et niveau.
- Afstanden mellem reolerne skal være så stor, at robotten skal kunne dreje om sin egen akse. På testlageret sættes mindste afstand mellem reolerne til robottens længde + 20 cm.
- På lageret skal der være et sted, hvor pallerne kan stilles og hentes af robotten.

2.5 Opbygning af lagerrobot

Da det ikke er hensigten at udvikle en ny type truck, skal denne prototype konstrueres ud fra nogle af de samme principper, som anvendes i manuelt styrede trucks. Robotten skal dog konstrueres så navigation og manøvrering sker med størst mulig præcision. Det skal derfor undersøges hvilke principper der vil være hensigtsmæssige i forhold til den prototype der skal bygges.

Hjul
En typisk truck har enten tre eller fire hjul og kan enten dreje på baghjulene eller på alle hjul. Trucks med tre hjul er typisk små og beregnet til indendørs brug og

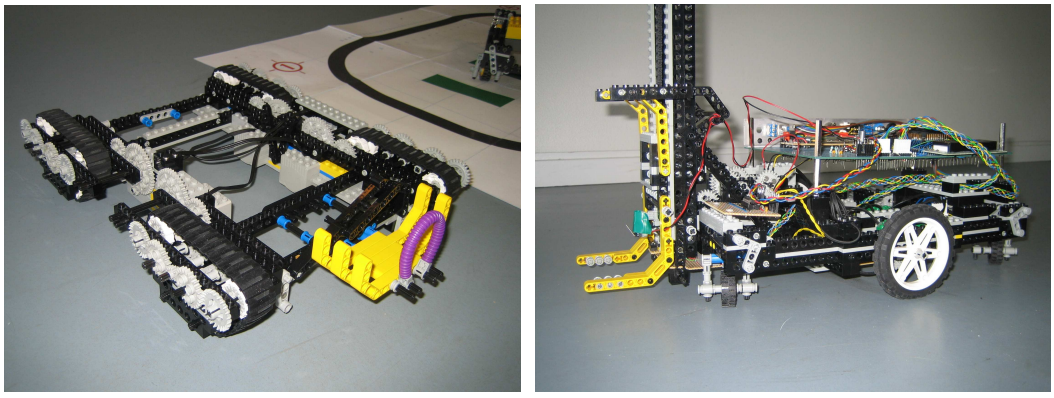
transport af almindelige paller. De tre hjul gør den fleksibel og nem at manøvrere på lidt plads. Denne konstruktion gør dog robotten mere ustabil.

For at opnå den største kontrol med robottens retnings-skift, blev det defineret i kravspecifikationen i afsnit 2.2.2 på side 19, at robotten skal kunne dreje om sin egen akse. På den måde kan robottens rute beskrives ud fra rette linier og rette vinkler. En drejning om sin egen akse kan opnås på 2 måder: (se figur 2.5)

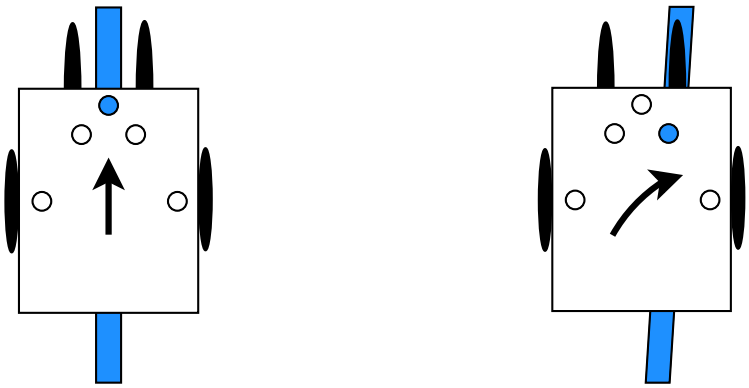
- Larvefødder på hver side.
  Larvefødderne fungerer efter samme princip som Forsvarets kampvogne eller pansrede manskabsvogne. Larvefødderne i hver side kører i hver sin retning og køretøjet roterer.

- Et hjul i hver side.
  Samme princip som larvefødder, men siden køretøjet kun drives af 2 hjul skal der sættes støttehjul på så stabiliteten sikres.

Der blev konstrueret 2 modeller ud fra det ovenstående og det blev konstateret, at modellen med larvefødder var meget ustabil i sin omdrejning på grund af den høje gnidningsmodstand mellem larvefødderne og underlaget. Både vægtfordeling og snavs på underlaget havde indflydelse på modellens rotation. Derimod var gnidningsmodstanden minimal på modellen med hjul. Støttehjulene fungerer efter samme princip som indkøbsvogne, dvs. at hjulene justerer sig og ændrer retning efter robottens bevægelser. På denne måde kan der foretages en mere kontrolleret rotation. Der vil blive arbejdet med en robot med 2 store hjul på midten og 4 små støttehjul i hvert hjørne.



Figur 2.5: Robotten er her opbygget med: Til venstre, larvefødder. Til højre, fire små løse hjul og to træk hjul



Figur 2.6: Her ses hvordan robottens lyssensorer påvirkes, når den kører hhv. lige og skævt på stregen.

Navigation

En effektiv truck skal kunne bevæge sig frit rundt i dens omgivelser, uden at være afhængig af et kabel til strømforsyning og kommunikation. Den ønskede prototype skal have samme egenskab og være trådløs. Både hvad angår strømforsyningen som udgøres af et batteri, og kommunikationen via bluetooth.

På en manuel truck er det naturligvis føreren som tager højde for omgivelserne og sørger for at undgå forhindringer. Føreren kan se hvor trucken er placeret og om der er en palle på gaflen. Disse egenskaber skal også gælde for den autonome lagerrobot, de opnås ved hjælp af lyssensorer og tryksensorer. Lyssensorerne kan detektere en farvet streg på gulvet som robotten kan følge, for at finde rundt på lageret og tryksensorerne kan mærke hvorvidt der er en palle på gaflen eller ej.

Hejsesystemet fungerer efter samme princip som fra gaffeltrucks, hvor gaflen føres ind under det objekt som ønskes løftet eller transporteret. Selve hejset drives af en motor som trækker en kæde. På den måde kan det kontrolleres meget præcist i hvilken højde gaflen befinder sig.

Udfra ovenstående opstilles der følgende krav til opbygningen af prototypen. Robottens skal:

- Have 2 trækhjul midt på, og 4 små støttehjul i hver hjørne.
- Være batteridrevet og trådløs.
- Kunne transportere mikroprocessoren og det medfølgende board.
- Trækkes af 2 9V LEGO motorer. En til hver af de store hjul.

- Være udstyret med et hejsesystem med gaffel, som kan håndtere paller i samme antal niveauer som findes på lageret. Hejsesystemet trækkes af en kæde og en 9V LEGO motor.
- Orientere sig ved hjælp af lyssensorer.
- Registrere om der er en palle på gaflen ved hjælp af en tryksensor.

Kapitel 3

Mikroprocessoren MSP430F149

I dette kapitel beskrives mikroprocessoren MSP430F149, der arbejdes med i projektet. Her vil den fysiske opbygning, samt de funktioner som mikroprocessoren stiller til rådighed blive gennemgået. Der vil kun blive beskrevet de funktioner der bliver benyttet i projektet.

I modelbetegnelsen står "MSP" for Mixed Signal Processor, dvs. at processoren kan behandle flere forskellige slags input/output. Det er bl.a. muligt at sende og modtage data via RS232-standarden for seriel kommunikation, konvertere analoge og digitale signaler, og styre andre enheder bl.a. ved hjælp af de indbyggede timere.

"F" står for Flash, som betyder at der er indbygget flashhukommelse. Det giver den fordel at processoren let kan programmeres igen og igen. Derudover er det meget let at finde softwarefejl da debugging kan foregå direkte på processoren. Skema over mikroprocessorens opbygning kan ses på figur 3.1 på næste side. [Nag03]

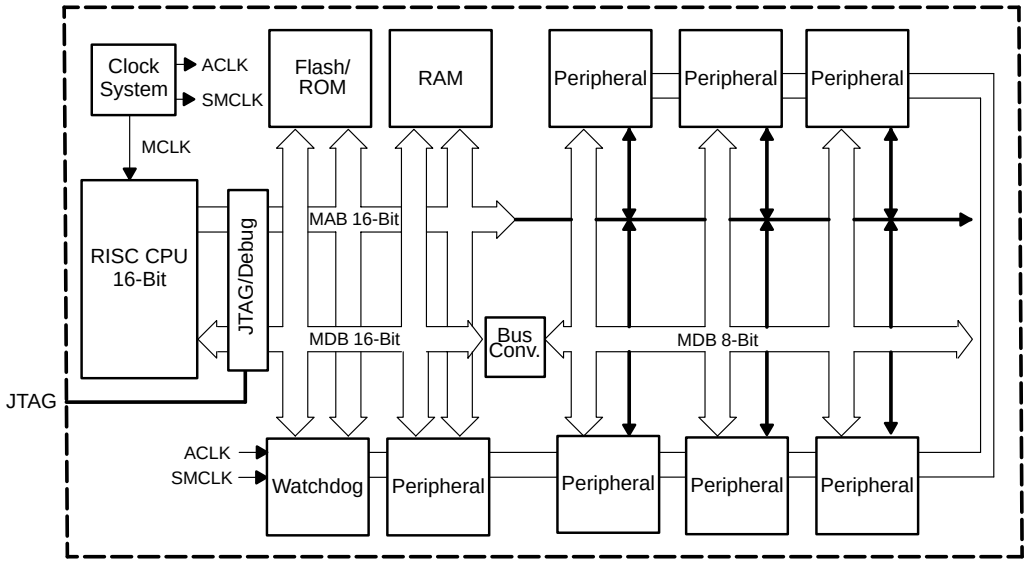
Fremover refereres der til mikroprocessoren med "MSP".

MSP'en indeholder bl.a. følgende indbyggede moduler [Ins04]:

- 16-Bit RISC CPU med 27 kerneinstruktioner.
- 64 KB hukommelse.
- 12-Bit ADC med en samplingsfrekvens på 200 kHz.
- 12-Bit DAC.
- 2 x USART til ekstern kommunikation.

- 2 x 16-bit timere.
- 6 I/O-porte med hver 8 ben som kan konfigureres individuelt i begge retninger.

  Mulighed for ekstern interrupt på port 1 og 2.



Figur 3.1: MSP'ens opbygning. Til venstre ses de indbyggede moduler: CPU, Clock system og hukommelse. Til højre ses de ydre enheder, det er for eksempel I/O porte og USART. [Ins04]

MSP'en har et areal på 10 x 10 mm, og er opbygget efter Von Neumann arkitekturen. Dvs. at den består af følgende dele:

- Hukommelse.
- Aritmetisk Logisk Enhed (ALU).
- Kontrolenhed.
- Input/output forbindelser (I/O).

Forbindelsen mellem de forskellige enheder består af parallelle kabler som kaldes en bus. Denne består af MAB (Memory Address Bus) der anvendes til at udvælge den lagercelle, der skal læses fra eller skrives til, og MDB (Memory Data Bus) der anvendes til at transportere data mellem lager og CPU.

MAB er på 16 bit (hvilket betyder at det maksimale antal adresserbare lagerceller for CPU'en er $2^{16} = 64k$) og MDB er også på 16 bit (hvilket betyder at disse lagerceller kan antage $2^{16} = 64k$ forskellige værdier). [Tan05]

3.1 CPU

MSP'ens CPU er en 16-bit RISC (Reduced Instruction Set Computer) CPU. Den indeholder 16 registre, hvoraf de 4 er reserverede til specifikke formål. De fire registre er:

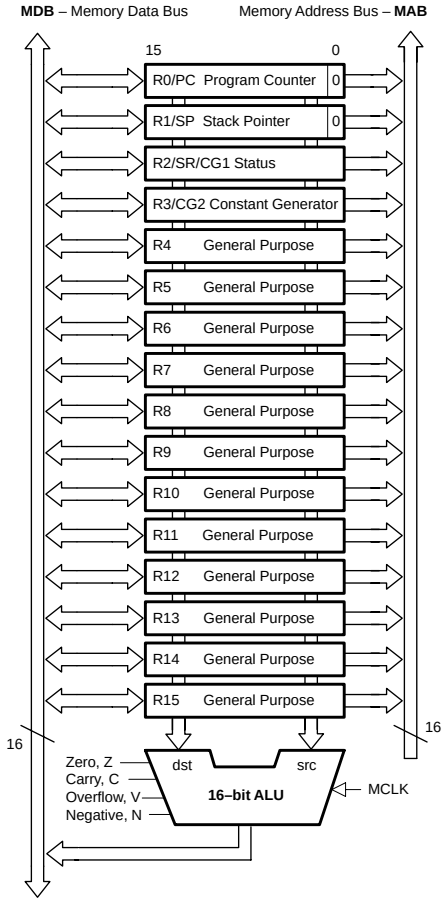
- Program Counter (PC) - Peger på den næste instruktion der skal udføres.
- Stack Pointer (SP) - Indeholder returadressen for subrutioner og interrupts.
- Status Register (SR) - Angiver statusflag for MSP'en.
- Constant Generator Register (CG) - Generer de seks mest brugte konstanter (-1, 0, 1, 2, 4 og 8). Værdien af konstanterne kan altså ses uden adgang til hukommelsen.

De generelle registre kan bruges som f.eks. data registre eller adresse pointere. Registrenes placering kan ses på figur 3.2 på modstående side. CPU'en kan tilgå hver af registrene direkte ved hjælp af dens interne bus.

CPU'en udfører operationer på følgende vis:

1. Den næste instruktion hentes fra hukommelsen.
2. Program Counteren tælles op så den peger på den næste instruktion.
3. Instruktionstypen bestemmes.
4. Hvis instruktionen bruger data fra hukommelsen findes disse.
5. Hvis der bruges data hentes disse og lægges ind i et register.
6. Instruktionen udføres.
7. Resultaterne gemmes i lageret.
8. Start forfra med næste instruktion.

[Ins04] [Tan05]

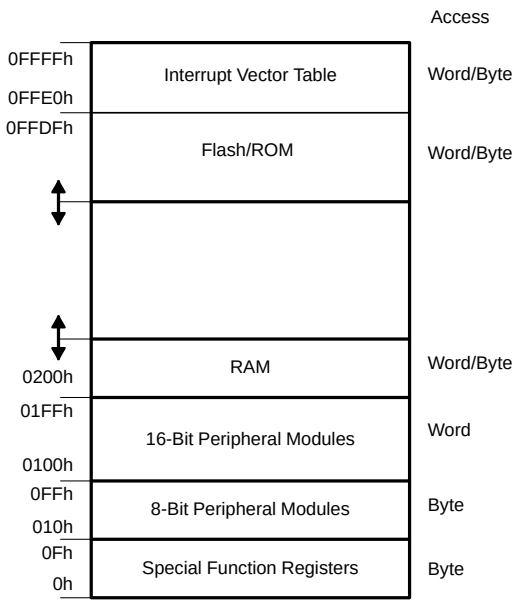


Figur 3.2: CPU'ens 16 interne registre, samt forbindelser via MAB og MDB og til CPU'ens ALU. [Ins04]

3.2 Hukommelse

MSP'ens 64 KB hukommelse er opdelt i 2 KB Random Acces Memory (RAM), 60 KB flash/Read Only Memory (flash/ROM), og 2 KB fordelt på bl.a. de ydre enheder. Fordelingen kan ses på figur 3.3 på næste side

Flash/ROM
Data der er lagret i flash/ROM er permanent, indtil det slettes eller overskrives, også selvom strømmen afbrydes, derfor er flash/ROM velegnet til programkode. Det er også muligt at anvende en del af flash/ROM lageret som RAM, hvilket betyder at der er mere RAM til rådighed. I dette projekt er der dog kun brug for de 2 KB RAM der allerede er til rådighed. Interruptvektoren ligger som de øverste 32 byte af Flash/ROM.



Figur 3.3: Hukommelsens opdeling med start- og slutadresser [Ins04]

RAM
Da RAM er hurtigere at tilgå end flash/ROM'en, bruges dette område til midlertidigt data, der ofte ændres eller flyttes rundt på. RAM kan også bruges til programkode, men da det nulstilles, når strømmen afbrydes, er det under normale situationer ikke hensigtsmæssigt.

SFR
De specielle funktions registre er delt op i: interrupt enable registre (giver mulighed for at aktivere interrupts), interrupt flag registre (giver bl.a. mulighed for at aktivere interrupt ved USART) og module enable registre (giver mulighed for at aktivere ydre enheder f.eks. USART). [Ins04]

3.3 Porte

I dette projekt skal der kobles forskellige komponenter til MSP'en. Det er ifølge kravspecifikationen afsnit 2.2.3 på side 20 bl.a. motorer, diverse sensorer og Bluetooth.

MSP'en har 48 digitale I/O ben, der alle kan bruges som digitalt input eller output. Desuden har alle benene nogle sekundære funktioner, som kan konfigureres individuelt af hinanden. De er inddelt i 6 porte á 8 ben, med betegnelse P1.x til P6.x.

Enhed	Ben	Forklaring
LCD Display	P1.0-P1.7 og P2.0-P2.2:	Styring af Display vha almindelig I/O
Joghjul	P2.3-P2.5:	Måling af interaktion vha. almindelig I/O
Omdrejningstæller	P2.6:	Måling af lifthøjde vha. almindelig I/O
Kontakter	P2.7 og P3.0:	Aktivering af kontakter vha. almindelig I/O
Seriel Forbindelse	P3.4-P3.5:	Afsending og modtagelse af seriel data hhv.
Motorer	P4.1-P4.6:	Styring af motorer vha. PWM modulation
Batteri	P6.2:	Måling af batterispænding vha. ADC
Lyssensorer	P6.3-P6.7:	Måling af lyssensorer vha. ADC

Tabel 3.1: Tabellen viser hvilke komponenter der er tilsluttet MSP'ens forskellige ben.

Tilgængelige sekundære funktioner:

P1.x / P2.x:	Timer A
P3.x:	UART/USART
P4.x:	Timer B (PWM)
P5.x:	UART/USART og ekstern clockfrekvens
P6.x:	ADC

I tabel 3.1 ses en oversigt over tilsluttede komponenter. [Ins03]

3.4 Interrupt

For at sikre at robotten altid kan sende statusmeddelelser og samtidig være i stand til at modtage nye ordrer fra lager-PC'en, bruges der interrupt. Interrupts er signaler der afbryder det kørende program i en begrænset tidsperiode og overlader kontrollen til en interrupthandler, der udfører de nødvendige handlinger eller rutiner. Efter interruptet er afviklet, overlader interrupthandleren igen kontrollen til programmet.

Et interrupt kan forårsages af forskellige input, eksempelvis en timer (se afsnit 3.5 på side 37) eller et input på en af MSP'ens interruptporte. Programmet, der bliver afbrudt, begynder igen når interruptet er afviklet, som om der ikke er sket noget, interrupts er altså transparente.

Når programmet får signal om interrupt, udføres følgende rutine:

1. Programmet færdiggøre den aktuelle instruktion.

2. Programtælleren bliver lagt i stack sammen med statusregistret, så det er muligt at returnere til den instruktion hvor programmet blev afbrudt.

3. Interruptvektoren, som er et identifikationsnummer for interruptet, bliver hentet ind i programtælleren.

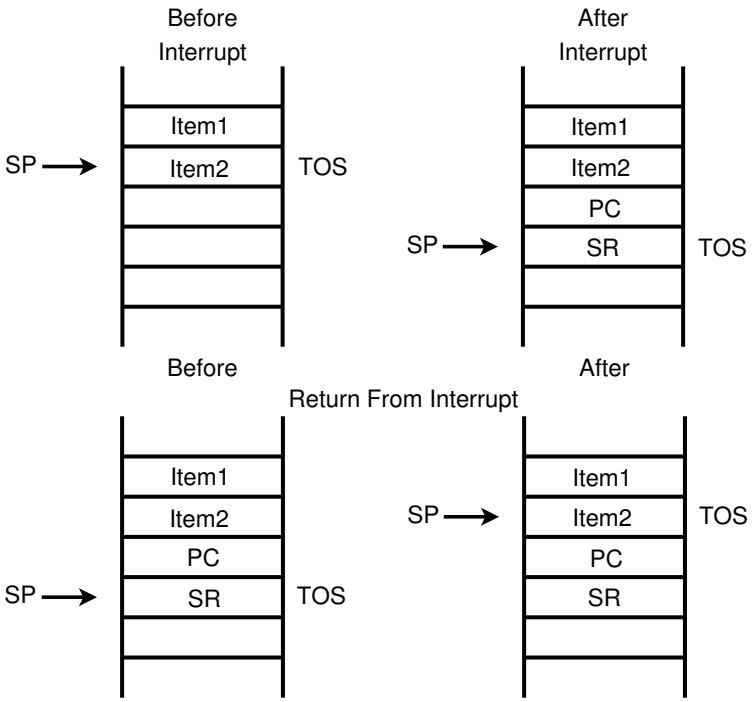
4. Programmet fortsætter med interruptrutinen.

5. Efter afslutningen af et interrupt bliver statusregistret og tidligere indstillinger hentet fra stack.

6. Programtælleren bliver hentet fra stack.

7. Programmet forsætter med den næste instruktion.

Figur 3.4 viser udseendet af stack ved interrupt.



Figur 3.4: Indhold af stack ved interrupt. Øverst: Interruptrutinen kaldes. Nederst: Interruptrutinen returnerer [Ins04]

Da et interrupt forårsages af forskellige input, kan der opstå flere interrupts til et bestemt tidpunkt. For at sikre at de vigtigste interrupts bliver udført først, sker der

en prioritering således at lavere prioriterede interrupts ikke kan afbryde et højere prioriteret. De lavere prioriterede bliver derfor først afviklet når de højere prioriterede interrupts er afviklet.

I dette projekt benyttes maskable interrupts, som kun kan aktiveres hvis global interrupt enable bit (GIE) er sat. Disse interrupts benyttes ved kommunikationen mellem MSP og PC, og ved timerstyret funktionskald. [Tan05] [Ins04]

3.5 Timere

Der findes tre forskellige timere i MSP'en: Timer A, Timer B og Watchdogtimeren. Timer A vil blive benyttet som en tidsindikator, og Timer B skal styre de 3 motorer ved at generere et "puls bredde signal" (PWM). Watchdog timeren anvendes ikke i dette projekt.

Timer A

Timer A og Timer B er næsten ens opbygget, med mange af de samme funktioner, selvom der eksisterer nogle forskelle, f.eks.:

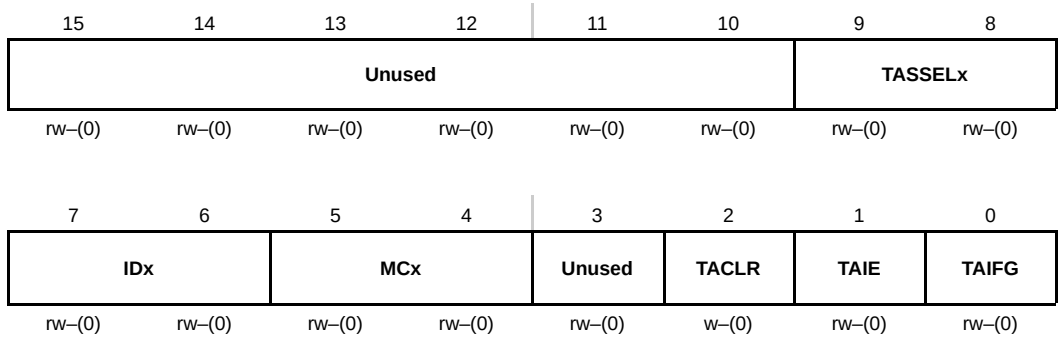
- Timer A er 16 bit, mens Timer B kan sættes til 8, 10, 12 eller 16 bit.
- Timer A har 3 capture/compare registre, hvor Timer B har 7.

Timer A's opsætning styres af TACTL (Timer A Control Register). Opbygningen af dette kan ses på figur 3.5 på den følgende side.

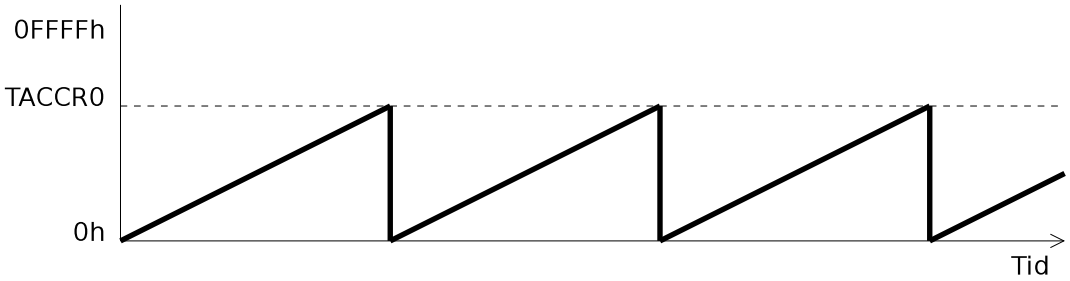
I dette projekt sættes MCx for Timer A til up mode, hvilket betyder at Timer A bliver styret af TACCTL0 (Timer A capture/compare control register 0), ved at nulstille den ved værdien TACCR0. Dette ses på figur 3.6 på næste side.

Timer B

Timer B understøtter interval-timing og PWM output. PWM vil blive brugt til at styre motorerne, hvilket bliver beskrevet i afsnit 4.2.1 på side 48. For at få Timer B til at give et PWM output skal den også sættes til *up mode*. Den eneste forskel



Figur 3.5: Timer A Control Register. TASSELx bruges til at vælge clocksource til Timer A. IDx bruges til at bestemme hvor meget clocken skal divideres med (1,2,4,8). MCx bestemmer hvilket mode Timer A skal sættes til. TACLR bruges til at nulstille timeren, counteren og divideren. TAIE og TAIFG bruges til at sætte interrupt op.[Ins04]



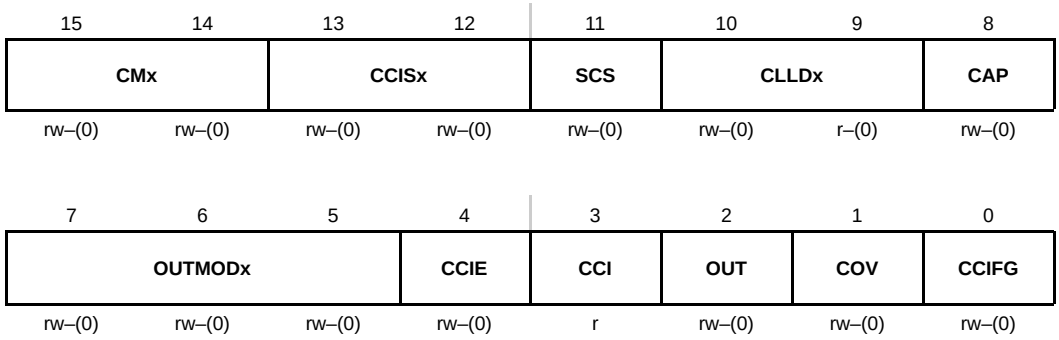
Figur 3.6: Figuren viser hvordan Timer A fungerer i up mode. Der tælles op fra 0 til TACCR0, hvorefter timeren nulstilles.[Ins04]

på Timer A og Timer B's kontrolregister, er TBCLGRPx (som kan bruges til at gruppere TBCCTLx) og CNTLx (som definerer bit-størrelsen på Timer B).

Timer B bliver nulstillet af TBCCR0, ligesom Timer A bliver nulstillet af TACCR0. Herefter kan Timer B bruges til at generere et PWM output på port P4.1–P4.6, via. regesterne TBCCTL1–TBCCTL6 og TBCCR1–TBCCR6. På figur 3.7 på næste side kan indholdet af TBCCTLx registret ses. [Ins04]

3.6 Seriel kommunikation

Kommunikation mellem MSP'en og eksterne enheder består af en seriel forbindelse. En seriel forbindelse betyder, en sekventiel dataoverførsel bit-by-bit. På MSP'en



Figur 3.7: TBCCTLx registret. CAP vælger mellem capture eller compare mode og OUTMODx bestemmer hvilket output mode TBCCTLx sættes til.[Ins04]

findes der to USART interfaces, hver med to porte til henholdsvis URXD (receive) og UTXD (transmit).

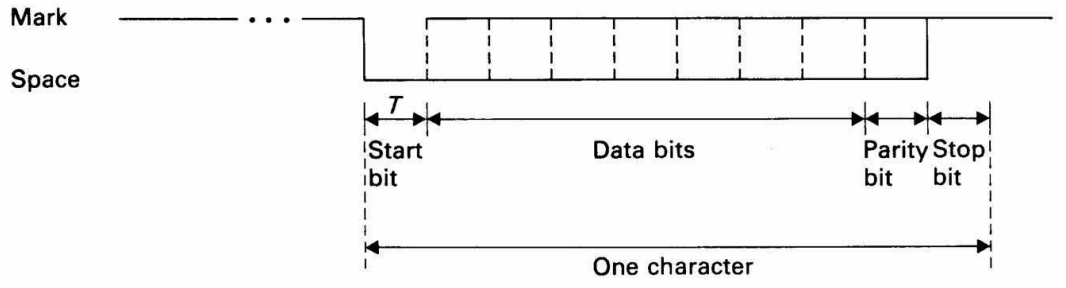
3.6.1 USART

USART Universal Synchronous-Asynchronous Receiver/Transmitter standarden angiver hvorledes den serielle kommunikation foregår. Transmissionen kan foregå asynkront (UART) og synkront (USRT), men standarden er den samme.

Ved starten af en transmission sendes et startbit, hvorefter 7 eller 8 databit sendes. Efter transmissionen af disse databit, sendes et eventuel paritetsbit, hvorefter der sluttes af med et eller to stopbit. Paritetsbittet gør det muligt at gensende informationer ved fejl. Dette sker ved at der tælles, hvor mange høje bit der har været i transmissionen. Herefter sættes paritetsbittet til høj/lav alt efter om der ønskes lige eller ulige paritet. Eksempelvis vil lige paritet med 8 databit som 11011010 få påhæftet et paritetsbit på 1, så der ialt er et et lige antal høje bit [Cle97]. En skematisk tegning over transmissionen af en byte kan ses på figur 3.8 på den følgende side

Før transmissionen, specificeres følgende for MSP'en og den eksterne enhed:

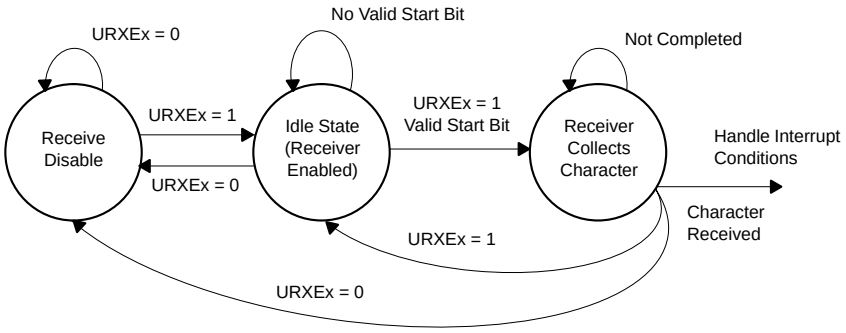
- Antal databit: 7 eller 8.
- Paritetsbit: intet, lige eller ulige.
- Antal stopbit: 1 eller 2.
- Baud rate, som er et udtryk for signalskift i sekundet.



Figur 3.8: Afsendelse af en karakter med et startbit, 7 databit, 1 paritetbit og 1 stop-bit. [Cle97]

I dette projekt vil kommunikationen foregå over USART0, som skal sættes til UART-mode (der gør det muligt at sende og modtage data uafhængigt af hinanden), en baudrate på 19200, 8 databit, 1 stopbit og ingen paritetscheck. Kommunikation mellem MSP'en og lager-PC'en, vil foregå via en trådløs bluetooth forbindelse.

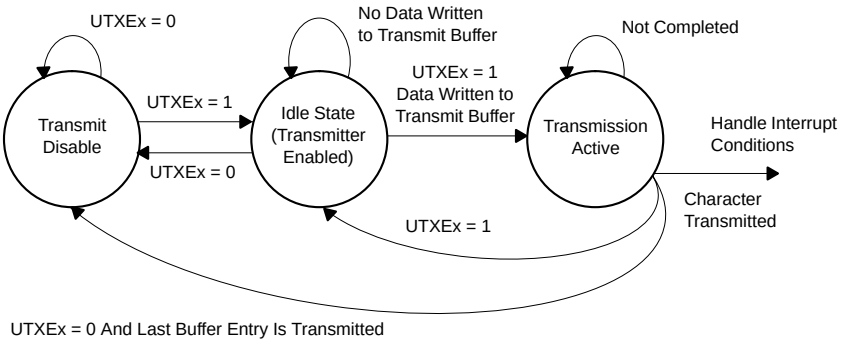
Til at indlede en transmission, findes der i MSP'en to bit der skal sættes for at kunne sende og modtage. For at kunne modtage sættes bittet URXEx høj (USART receive enable), se figur 3.9.



Figur 3.9: USART recieve enable bittets funktion. [Ins04]

For at kunne sende data sættes bittet UTXEx høj (USART transmit enable), se figur 3.10 på modstående side.

Hver gang der er modtaget eller sendt en karakter, genereres et interrupt, såfremt SFR er sat op til dette. [Ins04]



Figur 3.10: USART transmit enable bittets funktion. [Ins04]

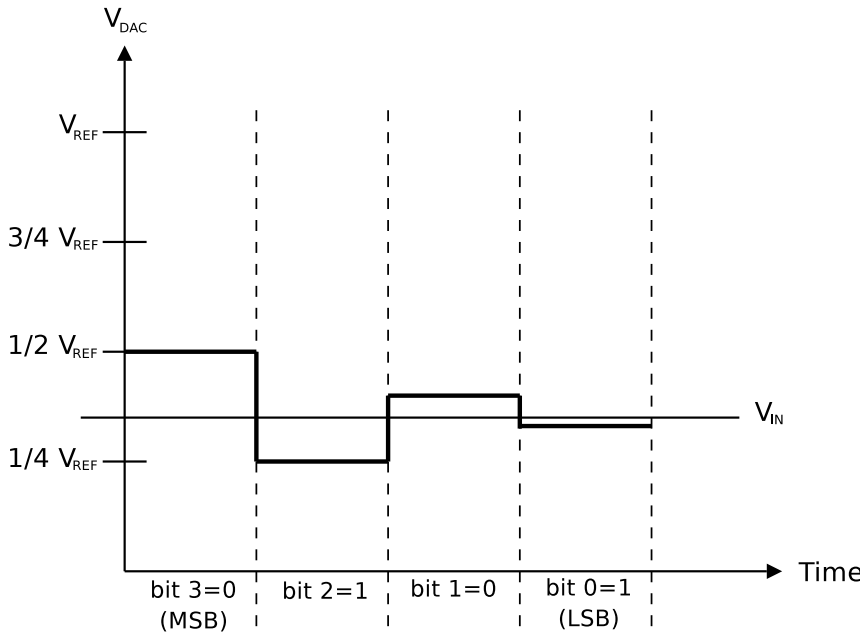
3.7 Analog Digital Konvertering

For at vide hvilken farve lyssensorerne registrerer, samt kunne overvåge batteriniveauet, er det nødvendigt at konvertere de analoge værdier om til digitale, der kan bruges i MSP'en. Dette gøres med en analog til digital converter (ADC). Den ADC der er implementeret i MSP'en er en 12 bit converter, hvilket vil sige at den bruger 12 bit til at beskrive den analoge spænding. Den har et måleområde fra 0-2.5 V, og har dermed en præcision på:

$$\frac{2,5V}{2^{12}} = 0,61mV$$

Den algoritme ADC'en bruger for at finde det binære tal kaldes SAR (Successive Approximation Register). Det fungerer ved at der i ADC'en også sidder en digital til analog converter (DAC). På figur 3.11 på næste side ses et eksempel på en sådan konvertering med en 4 bit converter. Først bliver det mest betydende bit (MSB) sat til 1, dette bliver så konverteret til et analogt signal af DAC'en og dette signal sammenlignes med det analoge inputsignal. Hvis det originale signal er lavere end det, det sammenlignes med bliver MSB sat til 0, ellers bliver det på 1. Dette gøres med alle bit ned til det mindst betydende bit (LSB).

Med MSP'ens ADC er det muligt at sample på 8 af MSP'ens porte enten én gang eller gentagende. Der kan styres vha. f.eks. Timer A eller Timer B hvor tit der skal samples. Samplingerne bliver gemt i op til 16 registre. Hvis der kun samples på et mindre antal signaler, f.eks. 6, og der er valgt at sample gentagende, sættes et EOS (end-of-sequence) bit i det sidste hukommelsesregister. [Ins04] [MAX01]



Figur 3.11: Konvertering i en 4 bit SAR ADC. Konverteringen bliver mere præcis jo flere bit der er til rådighed. [MAX01]

Kapitel 4

Hardware

I dette kapitel beskrives hardwaren, som benyttes til lagerrobotten. Først vil lysesensorerne blive beskrevet, derefter trip-tælleren på gaffeltårnet og til sidst hardwaren til motorstyringen.

Desuden henvises der til bilag B på side 90, hvor de problemer der har været i opbygningen af hardwaren bliver beskrevet.

Lagerrobotten forsynes af et 11,2 V LI-Poly batteri. MSP'en og mange af de andre komponenter kan ikke tåle så høj en spænding, derfor anvendes der 2 lineære strømforsyninger til at generere 3,3 V og 5 V ud fra batterispændingen. Dette kan ses i tabel 4.1. Strømforsyningerne benyttes til at forsyne komponenterne med deres optimale spænding.

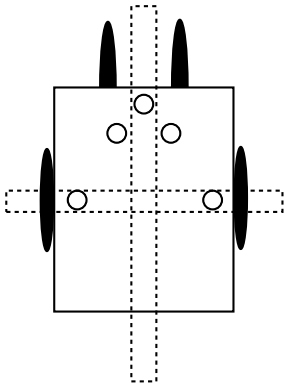
Parameter	**Spændingsforsyning**
Batteri	12 V
$+5V$	5 V
$+3,3V$	3,3 V
GND	0 V

Tabel 4.1: Strømforsyning til Lagerrobotten

4.1 Sensorer

4.1.1 Lyssensorer

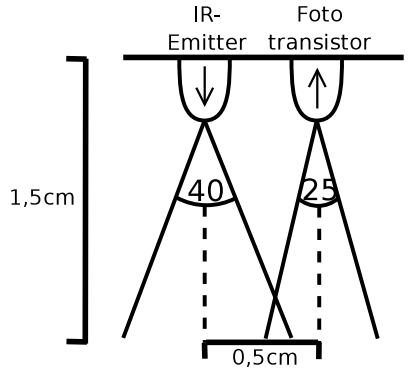
Lagerrobotten skal kunne navigere ved hjælp af streger på gulvet. Steder hvor robotten skal køre langsomt, som f.eks. lige før en palle eller ved enden af en gang skal stregerne være af anden farve, da lagerrobotten så kan programmeres til automatisk at sætte farten ned her. Der benyttes i alt 5 sensorer, 3 til at køre efter stregen foran på lagerrobotten og en på hver side til at registrere streger, der går på tværs af køreretningen. Sensorernes placering på lagerrobotten kan ses på figur 4.1. Sensorerne skal kunne registrere forskel på tre forskellige farver, ligesom de også skal fungere uafhængigt af anden belysning i lagerhallen. Dette gøres ved at benytte infrarøde dioder og fototransistorer.



Figur 4.1: Lyssensorernes placering på lagerrobotten. 3 front-sensorer og 2 side-sensorer.

En lyssensor bygges altså op af en infrarød diode og en fototransistor. Til at regulere hvor meget strøm der går igennem dem, påmonteres der tre modstande, se figur 4.3 på side 46. Når fototransistoren bliver belyst har den en meget lille modstandsværdi, og når den ikke bliver belyst vil modstandsværdien i stedet meget stor.

Der benyttes en SFH487-2 som diode og en SFH309FA som fototransistor. SFH-309FA'en har et dagslysfilter, der filtrerer synligt lys fra, hvilket betyder at den ikke påvirkes i nævneværdig grad af anden belysning i lagerhallen. Den infrarøde diode skal sidde så den lyser på gulvet, den mængde lys der reflekteres, opfanges af fototransistoren. Derfor er vinklen mellem dioden og fototransistoren vigtig for præcisionen. Der er en lille spredningsvinkel på både dioden og fototransistoren, så derfor er der ikke en direkte belysning. Placeringen kan ses på figur 4.2 på modstående side. [Sie99]



Figur 4.2: Fototransistorens og IR-Emitterens placering i forhold til hinanden

Ved at ændre modstanden (R1), ændres strømmen der løber igennem dioden. Denne modstand skal justeres, så dioden udsender nok lys til at fototransistoren kan registrere forskel på tre forskellige farver. Ifølge databladet for SFH487 må dioden maksimum trække en strøm på 100 *mA*. Da der over dioden er et spændingsfald på 1,5 *V*, skal der minimum sættes en modstand i serie af denne størrelse:

$$\frac{5V - 1,5V}{100mA} = 35\Omega$$

Ved denne modstand udsender dioden den største mængde infrarøde lys, men da fototransistoren vil blive overbelyst, vælges en større modstand. [Sie]

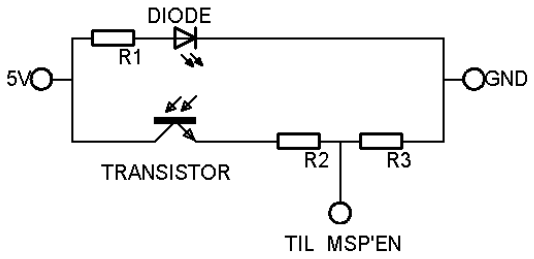
Til fototransistoren er der tilsluttet to serielt forbundne modstande (R2+R3), da der ønskes en spændingsdeling af 5 *V* til maksimum 2,5 *V*. Denne spænding er det højeste ADC'en kan måle. Fototransistoren fungerer som en NPN-transistor, der ved en lille strøm (belysning) på "basis" benet gradvist vil åbne for strømmen igennem transistoren. [Sie99]

For at finde de tre bedst egnede farver og modstande til formålet, er der udført et forsøg, hvor spændingsfaldet over fototransistoren er målt ved forskellige modstande og farver. Dette er gjort for at finde de spændingsværdier for farverne, der giver størst mulige spændingsintervaller. Resultaterne fra forsøget kan ses i appendiks C på side 92.

De modstandsværdier, hvor spredningen er størst, når R1 er 700 Ω, og R2 og R3 hver er på 40 *k*Ω. I tabel 4.2 på den følgende side ses der målingerne for de forskellige farver ved disse modstande. Da der ønskes så stor en spredning mellem farverne som muligt, blev sort og hvid valgt som de første. Da der skal bruges tre farver, vælges blå da den ligger nærmest middelværdien. På denne måde gives der det størst mulige spændings interval.

R1 [Ω]	R2+R3 [$k\Omega$]	Gul [V]	Sort [V]	Grøn [V]	Rød [V]	Blå [V]	Hvid [V]
700	80	2,17	0,58	1,87	1,87	1,37	2,39

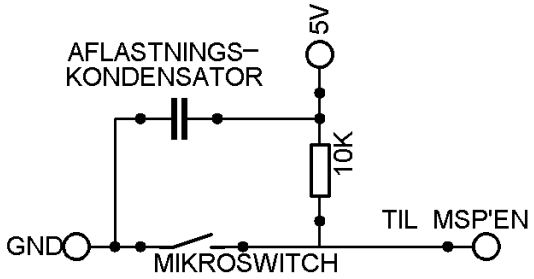
Tabel 4.2: Måleresultater, ved modstands værdierne R1 = 700 Ω, R2 = R3 = 40 kΩ



Figur 4.3: Kredsløbet for en lyssensor. R1 = 700 Ω, R2 = R3 = 40 kΩ.

4.1.2 Tryksensoren

Selve tryksensoren der er placeret på gaflen, er en mikroswitch (en lille kontakt). For at tilslutte den til MSP'en, kobles nogle komponenter dertil, for at sikre at signalet fra kontakten enten er højt eller lavt, og ikke påvirkes af støj. Måden kontakten er koblet op på kan ses på figur 4.4.



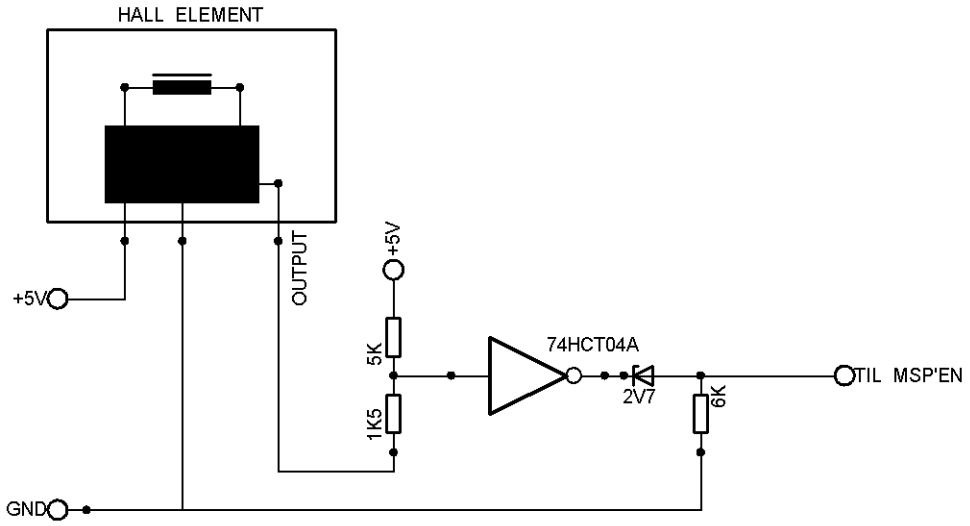
Figur 4.4: Kredsløbet for tryksensoren

4.1.3 Triptæller

For at få gaflen til at kunne løfte og stille paller i to højder, er det nødvendigt at vide hvor gaflen befinder sig, se kravspecifikationen afsnit 2.2 på side 16. Måden det kan gøres på er ved enten, at sætte nogle sensorer i forskellige højder på gaffeltårnet så MSP'en ved hvor højt gaflen er. En anden måde at finde gaflens position på er

ved, at måle antallet af omdrejninger på motoren der trækker gaflen og på den måde beregne hvor gaflen befinder sig.

Det er valgt at måle antal omdrejninger på motoren i stedet for sensorer på bestemte højder på gaffeltårnet, Da det vil gøre løsningen mere fleksibel. Til at måle omdrejningerne, er der brugt et HALL element, model TLE4905, der fungerer ved at sætte udgangen til GND, når en magnet nærmer sig elementet. I HALL elementet er der indbygget en schmitttrigger, så der ikke kommer støj i tidsrummet hvor outputtet åbner og lukker men kun reelle pulser. For at gøre målingen detaljeret er der monteret 4 magneter på et tandhjul, så HALL elementet giver 4 pulser for hver omgang motoren drejer. Opkoblingen af dette kan ses på figur 4.5.



Figur 4.5: Tilslutning af triptælleren til MSP'en: Fra HALL elementet sidder to serielt koblede modstande på 1,5 kΩ og 5 kΩ. Spændingen mellem de to modstande sendes igennem en inverter, der så giver 5 V når HALL elementet er åbent. Da HALL elementet kræver mere end 3,3 V forsynes med 5 V, Det bliver derfor nødvendigt med begrænse spændingen for at den ikke overstiger MSP'ens maksimale indgangsspænding. Derfor sidder der fra inverteren en 2,7 V zener diode som sænker spændingen til 3 V på udgangen når HALL elementet åbner og 0 V når elementet er lukket. Desuden sidder der en 6 kΩ modstand som der sikre at MSP'en ikke bliver overbelastet.

For at kende positionen af gaflen benyttes en software variabel, der tælles op eller ned, ud fra hvilken retning motoren kører, hver gang der modtages en puls fra HALL elementet. Det er desuden nødvendigt at kende udgangspositionen for gaflen for, at sikre at variablen har den rigtige start værdi. Derfor er der monteret en mikroswitch i bunden af gaffeltårnet, således at variablen nulstilles, når mikroswitchen trykkes.

For at tilslutte mikroswitchen til MSP'en skal den tilsluttes på samme måde som tryksensoren på gaflen, i afsnit 4.1.2 på side 46.

Med den valgte gearing, mellem motoren og gaflen, giver det 120 pulser fordelt på 27 *cm* højde. Det betyder at gaflens position kendes med en præcision på ca. 2,3 *mm*, når der tages højde for slør i LEGO tandhjulene. [Sie97]

4.2 Motorstyring

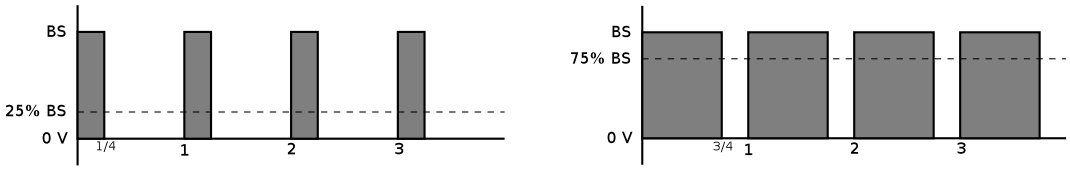
For at få lagerrobotten til at bevæge sig rundt og transportere paller på et lager, skal den drives af nogle motorer. Motorerne på robotten skal kunne køre frem og tilbage samt hæve og sænke gaflen. Det betyder at alle motorerne skal kunne køre i begge retninger, derfor skal det være muligt at bytte rundt på polerne. Da lagerrobotten i dette projekt bygges af LEGO benyttes LEGOs egne motorer af model 43362, fordi disse passer efter LEGOs mål og er nemme at montere på robotten.

Selve MSP'ens maksimale spænding ud på portene er 3,3 *V*, og må maksimalt belastes med 6 *mA* pr. ben. Derfor er det nødvendigt at forstærke signalet fra MSP'en før det kan bruges til at styre motorerne med. En mulighed for at gøre dette kunne være at forstærke et analog signal fra MSP'en, vha. en operationsforstærker, men da der ikke er nogle analog udgange fra MSP'en vil dette ikke være muligt. Det er derfor valgt at bruge et PWM signal via Timer B, til at styre de 3 LEGO motorer. [Hur03][Ins03, s. 21]

4.2.1 Puls Bredde Modulation (PWM)

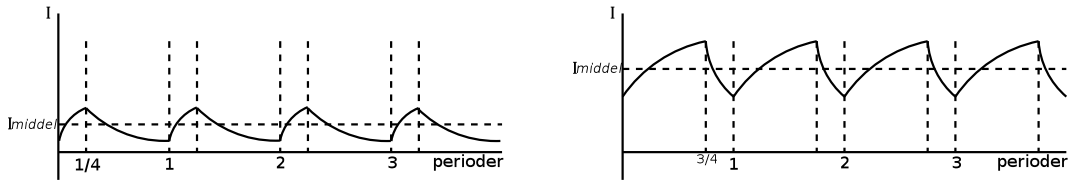
MSP'en er udstyret med syv PWM udgange (P4.0-P4.6), der i dette projekt er valgt til at styre de tre motorer, der er monteret på lagerrobotten. Grunden til at disse kan forsynes med et PWM signal er at, motorerne internt er opbygget som en spole, og derfor kan udnytte op- og afladningstiden for denne. Det ønskes at forsyne motorerne med den fulde batterispænding.

Ved at styre motorerne med et PWM signal, ønskes det at de forsynes enten med batterispændingen eller GND i et bestemt tidsinterval. PWM fungere ved at åbne og lukke for en spændingen i intervaller, dette er illustreret på figur 4.6 på næste side. Tidsperioden holdes konstant, og ved at ændre tiden hvori spændingen er høj, vil middelværdien blive ændret.



Figur 4.6: PWM signal ved henholdsvis 25% *og* 75% *effekt*

Da LEGO motoren er opbygget af en spole vil den, når PWM spændingen går fra lav til høj, begynde at trække en strøm. Strømforbruget vil stige i det interval hvor spændingen er høj. Når spændingen går fra høj til lav, vil spolen i motoren forsøge, at holde spændingen. Strømmen vil derfor begynde at falde. Når motorerne pulseres med et PWM signal, vil strømmen gennem motoren komme til at svinge omkring et middelniveau, i forhold til tidsintervallet på PWM signalet. Dette er illustreret på figur 4.7.

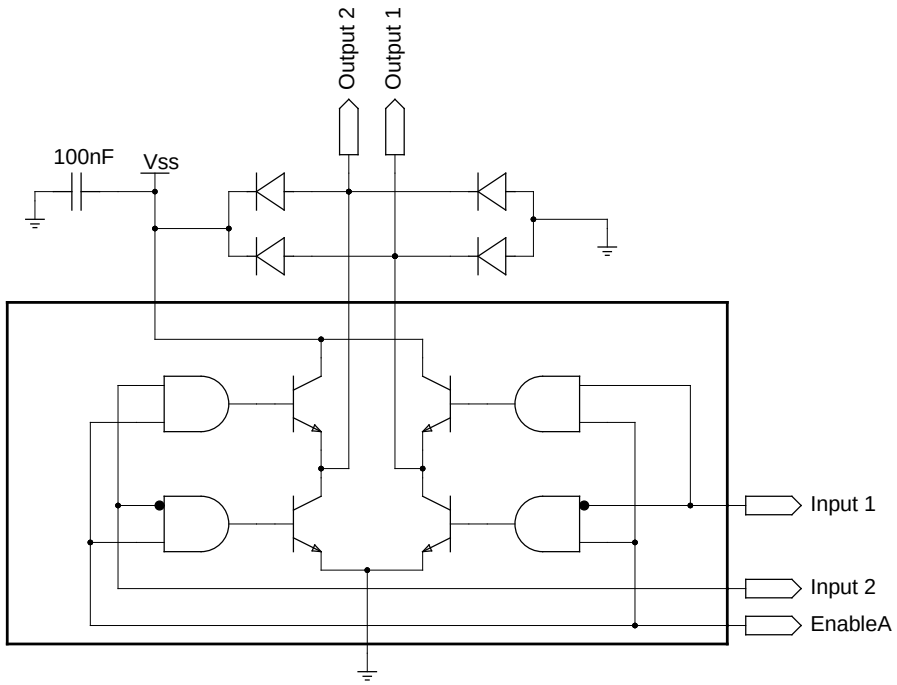


Figur 4.7: Middelstrøm gennem motoren ved et PWM tidsinterval på henholdsvis 25% *og* 75% *af perioden*

Motorerne vil fungere som et lavpasfilter i sig selv, derfor vil det ikke være nødvendigt at behandle PWM signalet, for at få motorerne til at køre jævnt. De kan derfor kobles direkte på H-broen.

H-broen er en IC der kan levere en høj spænding eller GND på udgangene, og kan håndtere at der bliver trukket en stor strøm, ud fra et logisk højt/lavt signal. H-broen der bliver brugt, er en L298 der indeholder to uafhængige identiske kredse til styring af en motor. H-broen kan klare en spænding på op til 46 V og en peakstrøm på 3 A. Det er en logisk digital kreds der opfatter et input som lavt hvis spændingen er <1,5 V og højt når spændingen er >2,3V. Disse værdier passer med hvad MSP'en kan give som output og de to komponenter kan derfor kobles direkte sammen uden problemer.

På figur 4.8 på næste side kan der ses et simpelt diagram af den ene halvdel over den interne opbygning af en H-bro. Den er symmetrisk opbygget, og den anden halvdel vil derfor fungere på samme måde. [ele00]

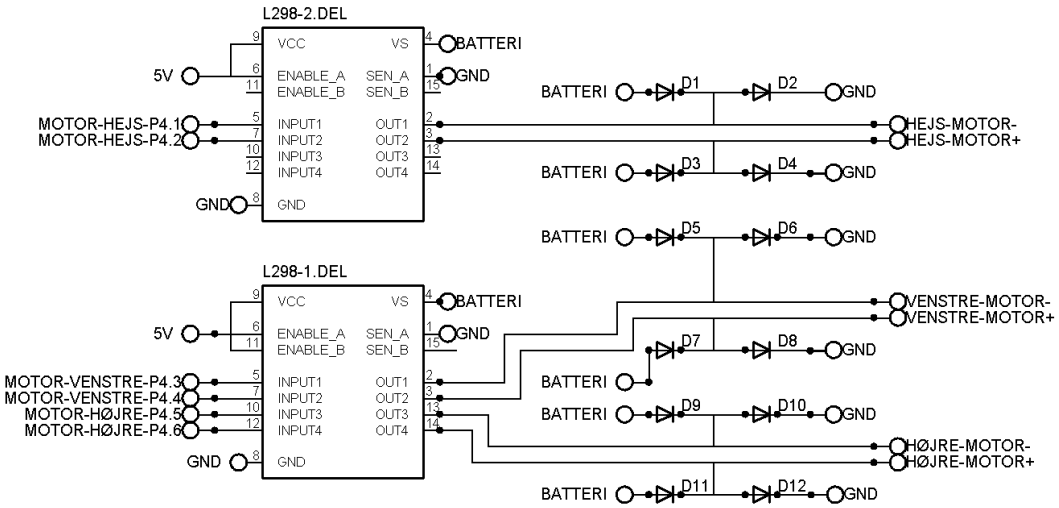


Figur 4.8: Et simpelt diagram over den ene halvdel af en H-bro. På input 1 kobles PWM signal 1 (PWM1), og PWM signal 2 (PWM2) kobles på input 2. Enable skal sættes til +5 V for at aktivere H-broen. Vss sættes til batteriet. Ved et lavt signal på PWM1 vil output 1 være GND og ved et højt signal vil den være batterispændingen. Der er koblet en række hurtig reagerende effekt dioder på udgangene for at undgå induktionsstrøm fra motorene. H-broens logiske del, som f.eks. AND-gates, skal forsynes med 5 V, men det er ikke vist på dette diagram

4.2.2 Tilslutning af H-broen

H-broen's tilslutning til MSP'en er vist på figur 4.9 på modstående side, og der kan på tabel 4.3 på næste side ses hvad de forskellige betegnelser står for. I MSP'en bruges PWM delen af P4.0 internt, til at styre Timer B, derfor bruges output P4.1–P4.6 til at styre H-broerne. Der bruges to PWM output pr. motor, og derfor er der brug for to H-broer for at kunne styre alle motorerne.

Nu er LEGO motorerne forbundet til MSP'en, via de to H-broer, og kan styres ved at generere et PWM output på port P4.1–P4.6. Det skal dog være sådan at P4.1 skal være slukket hvis P4.2 kører, da de er forbundet til motorerne parvis. Ellers vil det resultere i at motoren vil bremse. [Sem97]



Figur 4.9: Tilslutning af H-bro til MSP'en

Parameter	**Spænding**
V_s	Batterispænding
V_{ss}	$5V$
GND	$0V$
Enable A & B	V_{ss}
Current Sensing A & B	0V

Tabel 4.3: Strømforsyning til H-bro

Kapitel 5

Programdesign: Mikroprocessor

Ifølge kravene fra kravspecifikationen til mikroprocessoren, skal der udvikles et program, der gør det muligt at behandle input fra det hardware, som er beskrevet i afsnit 4 på side 43 og meddelelser fra lager-PC'en. For at overskueliggøre programmets struktur og opbygning, vælges det at inddele det i forskellige moduler i en lagdelt struktur. Dette kan ses på figur 5.1 på næste side, hvor hver boks betegner et modul. Derudover gør denne metode det let for programmøren at opbygge en bestemt funktionalitet eller modul. F.eks. behøver processerne ikke vide hvad der foregår i hardwaren, men kun reagere på input fra driverne. Dette design forudsætter således at driverne skal skrives før de overliggende lag kan fungere.

Til denne opdeling opstilles et regelsæt, der gør det muligt, at abstrahere fra de dele som ikke grænser op til det enkelte modul. Disse regler lyder som følger:

1. Et hvert modul må ikke tilgå andre moduler i samme lag.

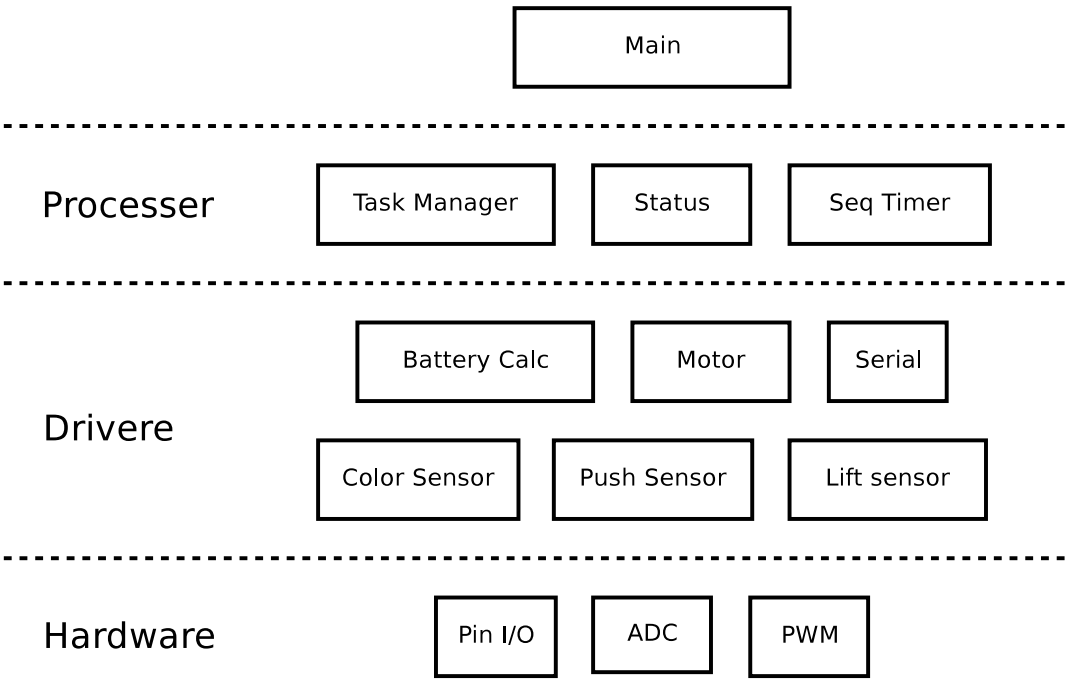
2. Et hvert modul har grænseflader mod et eller flere moduler i laget over og under.

3. Alle moduler kan tilgå og/eller ændre globale variabler.

4. Modulerne må ikke tilgå og/eller ændre i lokale variabler i andre moduler.

5. Main er det eneste modul som må benytte uendelige løkker.

6. Ingen moduler må bruge CPU-tid på at vente på input eller andre hændelser.

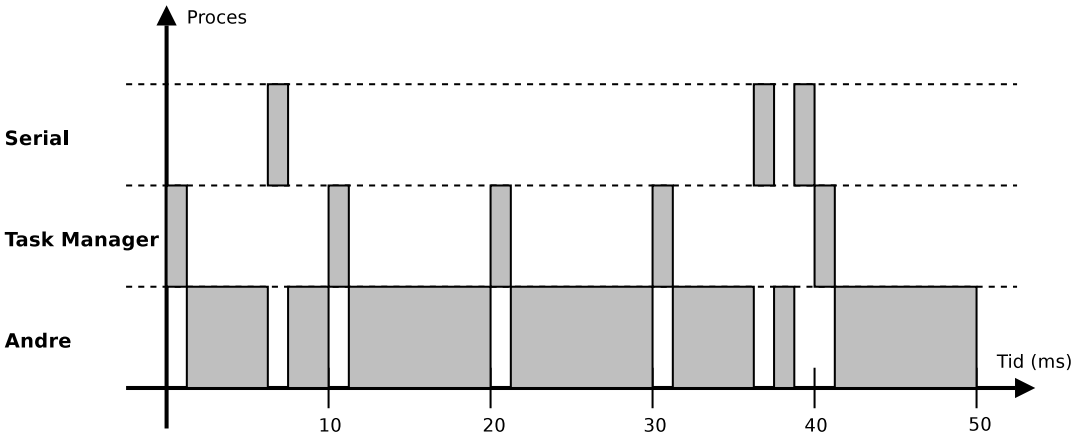


Figur 5.1: Programmets opdeling i moduler. Øverst ses main-løkken, derunder processer, drivere og hardware. Modulerne kaldes fra laget ovenover.

5.1 Procesdesign

Da der ikke er mulighed for, at afvikle parallelle processer på MSP'en, skal der afsættes CPU-tid til et modul ad gangen. Der kan afsættes CPU-tid på bestemte tidspunkter, med bestemte intervaller eller når der er behov for det. På den måde kan der skiftes mellem de enkelte processer, så det udadtil virker som om de kører parallelt. Figur 5.2 på den følgende side viser hvordan dette kan gøres.

Da den serielle kommunikation skal afvikles præcis på det tidspunkt hvor der kommer et input fra PC'en, styres dette af interrupt og får dermed høj prioritet. Robottens fremdrift og styring, som håndteres under Task Manager, foregår relativt langsomt i forhold til MSP'ens arbejdshastighed. Derfor er der kun behov for at reagere på nye hændelser med et bestemt interval. Den resterende CPU-tid bruges på at køre andre rutiner som ikke er tidskritiske, men som skal køres så meget som muligt. F.eks. kan der ligge data i modtagebufferen som skal behandles.



Figur 5.2: Figuren viser afviklingen og prioriteringen af de forskellige processer i programmet, men angiver ikke hvor meget CPU-tid de bruger. Øverst ses den serielle kommunikation som er interruptstyret og er derfor højt prioriteret, i midten ses modulet "Task Manager"som kaldes med 10 ms interval og nederst ses andre prosesser.

5.2 Moduldesign

I dette afsnit vil de enkelte moduler fra figur 5.1 på foregående side blive gennemgået som en modulspecifikation over alle moduler. Denne beskriver grænsefladerne til modulet, input, output og hvad de skal udføre.

5.2.1 Task Manager

Task Manager er en avanceret proces, som skal håndtere robottens instruktioner og kontrollere om de bliver udført tilfredsstillende. Processen skal kaldes hvert 10. ms, da det er hurtigt nok til at robotten kan reagere på de forskellige sensorer.

Task Manager indeholder følgende funktioner:

- Kontrollér om instruktionen er korrekt.
- Tilføj instruktionen til en kø.
- Kald det modul der skal udføre instruktionen.
- Håndtere returneret værdi.

- Nulstil instruktionskø.

Da der findes 4 forskellige instruktioner, opdeles processen i følgende undermoduler:

- forward.
- back.
- turn.
- pallet,get/put.

Når Task Manager bliver kaldt, kontrolleres der om tekststrengen, som er modtaget fra lager-PC'en er korrekt. Hvis dette ikke er tilfældet, sendes der besked til Lager-PC'ren om at instruktionen ikke er gyldig.

Task Manager skal udfra tekststrengen bestemme hvilket modul der skal udføre instruktionen, og dette modul skal kaldes. Modulerne skal have en pointer til tekststrengen, så de selv kan finde de oplysninger der er nødvendige for at udføre instruktionen.

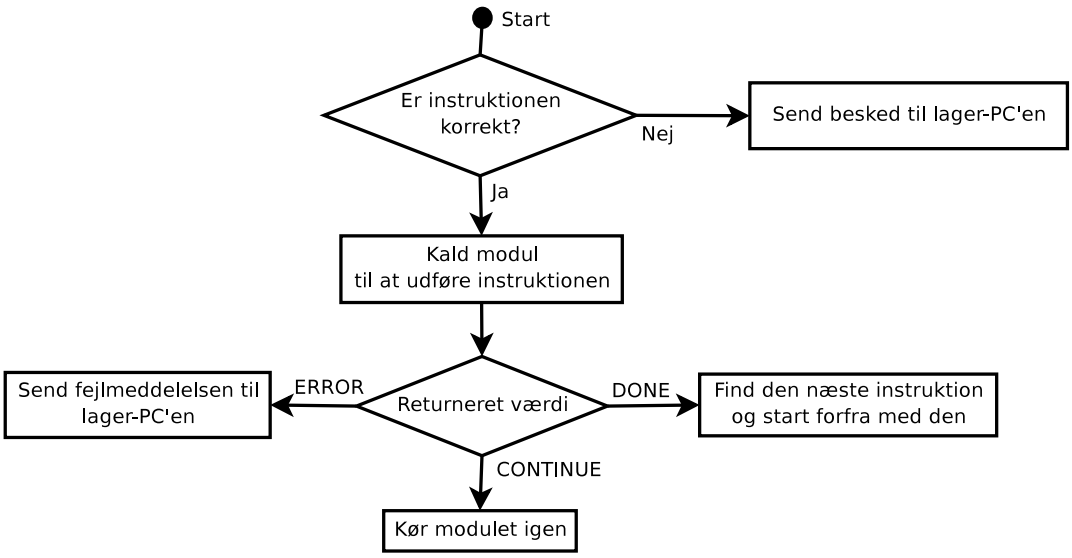
Undermodulet der bliver kaldt, returnerer enten DONE, ERROR eller CONTINUE. Hvis der returneres værdien DONE, skal modulet Task Manager finde den næste instruktion i køen og starte forfra. Hvis det modul der skal udføre intruktionen derimod returnerer ERROR, skal Task Manager sende besked om dette til lager-PC'en så brugeren kan informeres om at der er opstået en fejl. Hvis modulet returnerer CONTINUE skal det køres igen.

Grænseflader
Task Manager tilføjer de modtagne tekststrenge til en kø i mikroprocessoren. Når modulet henter den næste tekststreng kaldes det et undermodul til at udføre instruktionen.

Input
Task Manager får en værdi returneret fra undermodulerne når de er kørt igennem.

Output
Modulet kalder et undermodul med en pointer til tekststrengen som argument.

Figur 5.3 på den følgende side viser afviklingen af modulet Task Manager.



Figur 5.3: Figuren viser et flowdiagram for afviklingen af Task Manager

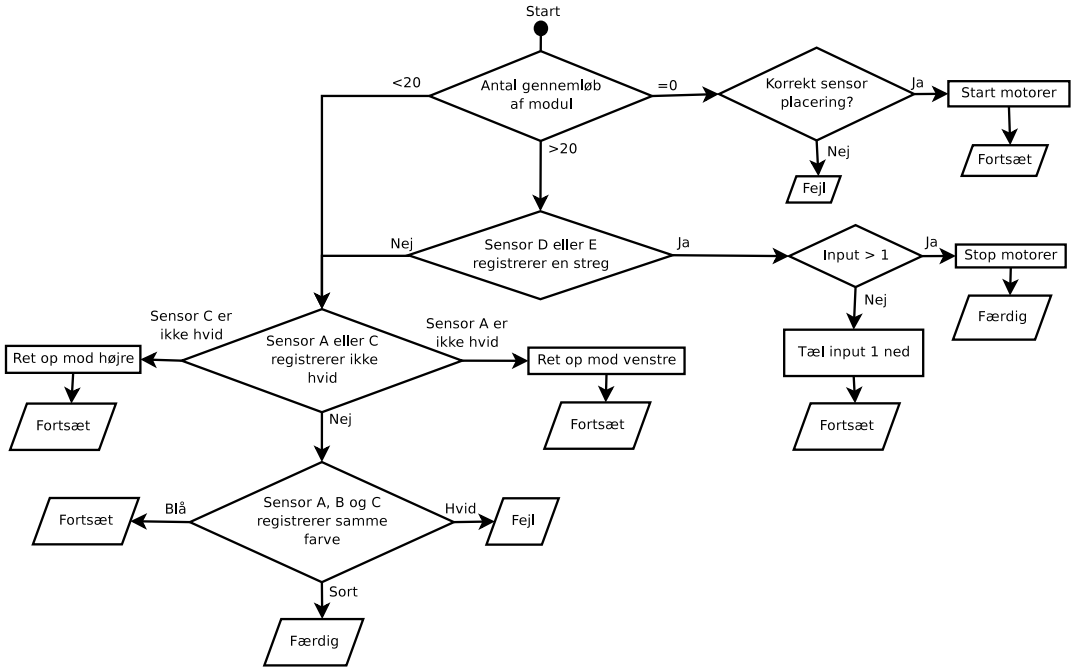
5.2.2 Forward

Dette modul kaldes når robotten skal køre lige ud, enten til et kryds eller ind til en reol.

Figuren 5.4 på næste side viser hvordan forward-modulet skal bygges op omkring en masse spørgsmål, der så fører hen til den korrekte handling. Der holdes styr på hvor mange gange modulet er gennemløbet, da det ikke er de samme ting der skal tjekkes for hver gang.

Første gang modulet gennemgås, kontrolleres det om sensorerne er placeret korrekt, dvs. om den forreste sensor (B) er på en streg. Hvis det er tilfældet startes motoren. Fra anden gang og op til 20 gange tjekkes kun om lagerrobotten følger stregen. Det gøres ved at tjekke om én af de to forreste sensorer (A og C) ikke er hvide. Hvis det er tilfældet, bestemmes hvilken retning der skal rettes op alt efter hvilken sensor der ikke registrerede hvid.

Når modulet er gået igennem 20 gange, tjekkes der også for om en eller begge sid-desensorer (C og D) registrerer en streg. Hvis de gør tjekkes der om robotten skal stoppe eller forsætte til endnu en streg. Hvis inputtet var 1 stoppes motorerne, ellers tælles inputtet én ned og lagerrobotten forsætter. Da lagerrobotten ikke kun skal kunne stoppe ved et kryds, men også på vej ind efter en palle, tjekkes der også om de forreste sensorer (A, B og C) registrerer den samme farve. Hvis det er sort stoppes robotten, hvis det er blå ignoreres det, og hvis det er hvid stoppes robotten og der



Figur 5.4: Figuren viser flowet i modulet forward

returneres at der er sket en fejl.

Grænseflader
For at vide hvilke farver de forskellige sensorer registrerer bruges driveren “Color Sensor” og for at styre motorerne bruges driveren “Motor”.

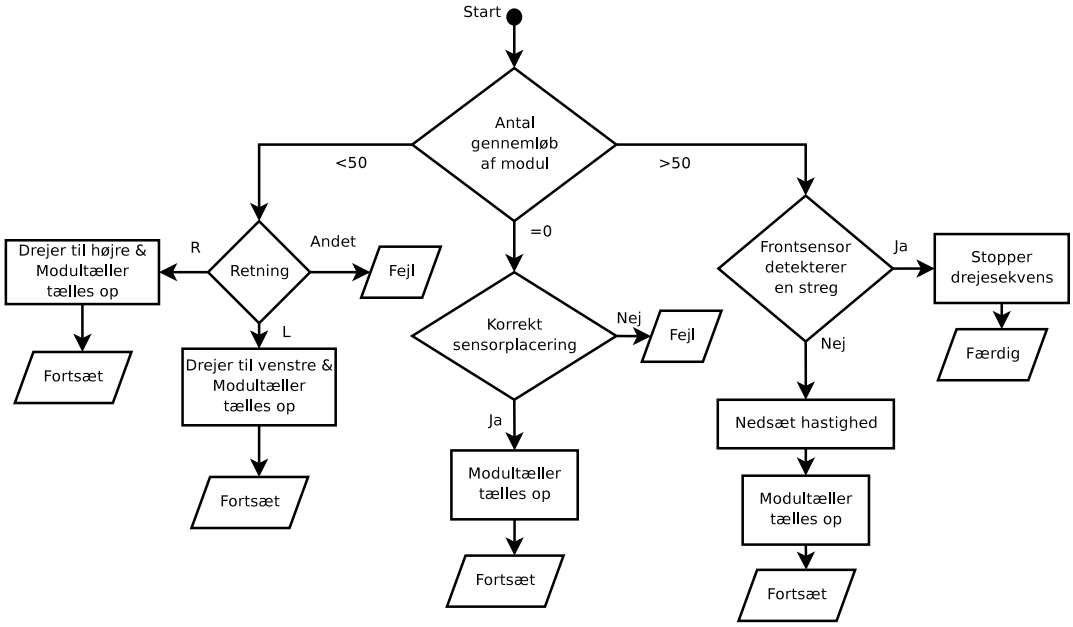
Køremodulet kaldes af Task Manager.

Input
Modulet får en pointer til en tekststreng, hvor første karakter er et tal mellem 1 og 9, som input. Dette tal afgør hvor mange kryds robotten skal registrerer før den standser. Hvis robotten kører til en reol stopper den altid og bruger altså ikke inputtet.

Output
Når modulet køres returnerer den enten CONTINUE hvis instruktionen endnu ikke er udført, ERROR, hvis der er registreret uventede farver fra sensorerne eller DONE hvis opgaven er udført.

5.2.3 Turn

Dette modul skal sørge for at robotten kan dreje 90 grader omkring sin egen akse. Dette skal gøres ved hjælp af input som kommer fra Task Manager.



Figur 5.5: Figuren viser flowet i drejemodulet

På figuren 5.5 ses hvordan drejemodulet skal opbygges. Selve modulet bygges op omkring en tæller, som holder styr på hvor mange gange modulet er gennemløbet. Således vil modulet første gang det køres detektere om lyssensorernes værdier passer med det forventede. Det forventes at den forreste midtersensor og en af sidesensorerne står på en streg.

Fra andet gennemløb af modulet tjekker modulet for argument og begynder at dreje til den side argumentet bestemmer. Efter 50 gennemløb forventes det at robotten er nået ud over den streg der drejes fra. Derfor begynder modulet at tjekke for streg på frontsensoren. Hvis der detekteres en steg stoppes drejesekvensen og modulet har udført instruktionen.

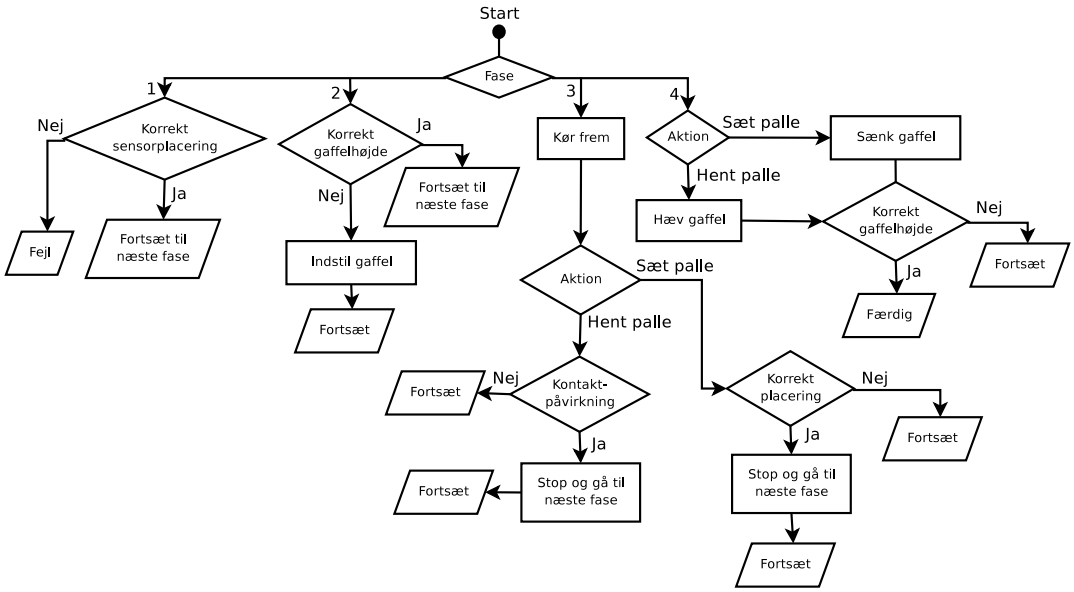
Grænseflader For at hente status på de enkelte lyssensorer benyttes driveren “Color Sensor”, og for at sætte hastigheden på motoren når robotten drejer benyttes driveren “Motor”.

Input For at modulet kan se hvilken vej der skal drejes, gives henholdsvis argumenterne R og L som højre og venstre.

Output Modulet returnerer ligesom “Forward” en værdi til Task Manager. CONTINUE, hvis instruktionerne ikke er udført, ERROR, hvis der er opstået en fejl eller DONE hvis instruktionen er udført.

5.2.4 Pallet,get/put

Dette modul skal kunne styre gaflen på robotten, således at den kan samle paller op og sætte dem ned igen i forskellige højder. Ud fra de input der kommer fra Task Manager.



Figur 5.6: Flowet i modulet

På figur 5.6, vises hvordan modulet skal fungere. Selve modulet vil være bygget op omkring 4 forskellige faser, der vil blive styret ud fra de input der kommer fra sensorerne og kontakterne på robotten. Når modulet bliver kaldt skal det igennem alle 4 faser før instruktionen er udført.

Ved første gennemløb af modulet vil 1. fase blive kaldt. Denne fase vil kontrollere om robotten står korrekt på gulvet, hvis den gør det vil 2. fase vil blive kaldt.

I 2. fase vil gaflens højde blive indstillet, således at hvis en palle skal hentes vil gaflen være i en højde mellem etagen og pallen der skal hentes. Hvis pallen derimod skal stilles på etagen vil gaflen være hævet en smule, således pallen ikke rører etagen

når robotten kører frem. 2. fase vil blive gennemløbet indtil gaflen står i den rigtige højde og 3. fase vil blive kaldt.

I 3. fase vil robotten begynde at køre frem mod lagerreolen. Hvis den skal hente en palle vil den stoppe når tryksensoren bliver påvirket af pallen. Hvis pallen skal stilles, er kontakten tryksensoren allerede påvirket og derfor kan den ikke bruges til at bedømme afstanden til lagerreolen. Derfor vil robotten i stedet styre efter en streg på gulvet, og stoppe så snart stregen indikere at robotten står ved lagerreolen. Når robotten igen står stille vil modulet gå videre til 4. fase.

I 4. fase vil robotten stå hvor pallen skal stilles eller løftes fra. Hvis den skal stilles i reolen, vil gaflen sænke sig så pallen er fri, og hvis pallen skal tages fra reolen vil gaflen blive hævet en smule, så pallen er løftet fri af reolen.

Grænseflader
I faserne skal enten gaffelpositionen ændres, eller robottens placering. Til dette bruges driveren “Motor”. I 1. og 3. fase bliver der hente informationer fra lyssensorerne via driveren “Color Sensor”. I 2. og 4. fase skal gaflens højde bestemmes, til dette bruges driveren “Lift Sensor”. I 3. fase bruges tryksensoren med driveren “Push Sensor”.

Input
Modulet skal modtage to argumenter. Første argument skal enten være “G” eller “P” for henholdsvis at hente og stille pallen. Andet argument skal være et karakteren A eller B, der angiver højden hvor pallen skal stilles eller hentes.

Output
Ved hver gennemløb af modulet vil det, ligesom “Forward” og “Turn” returnere en værdi. CONTINUE, hvis modulet skal køres igen, ERROR hvis der er opstået en fejl eller DONE når instruktionerne er udført.

5.2.5 Status

Dette modul skal stå for at hente status på de enkelte drivere og sende status videre til lager-pc’en. Modulet skal kaldes men et interval på ca. 1 sekund.

Grænseflader
De enkelte drivere der hentes status fra er:

- Lyssensordriveren (Color Sensor).
- Batteridriveren (Battry Calc).

- Gaffeldriveren (Lift Sensor).

Input
Modulet modtager en værdi fra hver af driverne. “Color Sensor” returnerer en farve fra hver sensor (A–E), “Battery Calc” og “Lift Sensor” returnerer et tal.

Output
Status sendes til Pc’en med følgende format:

`$status,A:`*x*`,B:`*x*`,C:`*x*`,D:`*x*`,E:`*x*`,BATT:`*x*`,STEP:`*x*

Hvor *x* er forskellige værdier.

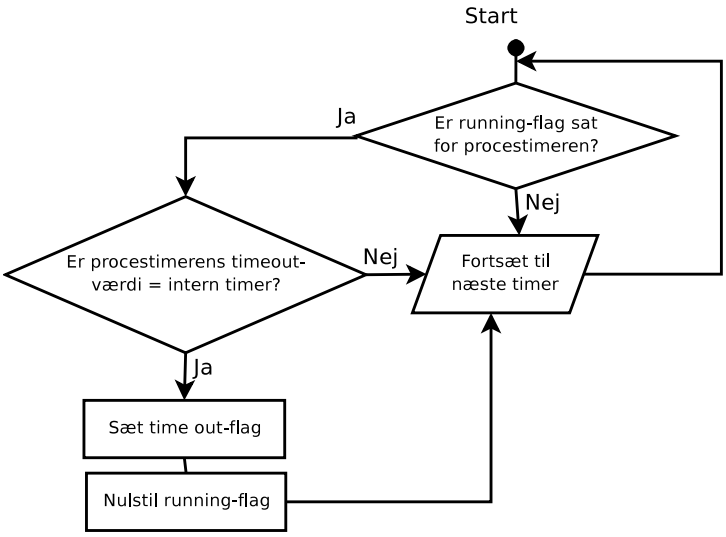
5.2.6 Seq Timer

Sequence Timer er en proces som tæller op på de interne timere, som bestemmer hvornår de intervalstyrede processer skal køre. Optællingen kaldes med interrupt fra Timer A hvert *ms*. “Main” kalder, for hvert gennemløb, modulets time-out-funktion for hver interval-styret proces og kan derudfra bestemme om det er tid til køre processen.

Modulet er bygget op omkring en fast struktur. Denne struktur består af en time-out værdi, som skal indeholde intervallet for den enkelte timer og 2 flag, som angiver om timeren er i gang eller udløbet. Der skal opbygges følgende 3 funktioner for at kunne styre denne proces:

- En managerfunktion til at overvåge timerne og afgøre hvilke timere der er udløbet.
- En startfunktion som starter timeren op igen med en given udløbstid, som angives i *ms*.
- En time-out-funktion, der angiver om den enkelte procestimer er udløbet.

Denne opdeling sørger for, at overlade proceskaldene til main for at følge det opstillede programdesign. Managerfunktionen skal køre som det første og “Main” kan derefter benytte time-out-funktionen til bestemme hvilke procestimere der er udløbet, og derfor hvilke processer der skal køres. Figur 5.7 på den følgende side viser flowet i managerfunktionen. Udgangspunktet er at alle timere er startet med startfunktionen. Når en procestimer er udløbet, skal den startes op igen med startfunktionen. Dette gøres typisk i sammenhæng med at der konstateres at timeren er udløbet.



Figur 5.7: Sequence timerens managerfunktion. Denne funktion vil køre fra den første procestimer til den sidste. Den interne timer bliver talt 1 op ms.

For at managerfunktion kan styre alle procestimere med et givet tidsinterval, er det vigtigt at den ved hvor langt tid der er gået. Det er i denne sammenhæng at Timer A's interruptfunktion skal bruges. Timer A sættes op til at generere en interrupt hvert *ms*. Dette gøres ved at sætte Timer A op til følgende:

Timer A's control register (TACTL) sættes TACTL = 0x02D0. Det vil sige at Timer A registrene vil blive sat op til følgende:

- At kører på SMCLK clock.
- Clocken divideres med 8.
- Timer A sættes til op mode. Timeren tæller op til TACCR0.
- Ingen reset.
- Ingen interrupt fra Timer A, det skal generes i TACCTL0.

TACTL kan ses på figur 3.5 på side 38.

TACCTL0 skal generere et interrupt når værdien skrevet i TACCR0 er den samme som Timer A. Dette gøres ved at sætte TACCTL0 = 0x0010. Derved vil TACCTL0 styre Timer A vha. TACCR0. Dette kan ses på figur 3.6 på side 38.

For at bestemme det interval der ønskes interrupt ved, skal der bruges følgende formel:

$$\text{TACCR0} = (\text{PERIOD} * \text{XT2})/8.0$$

PERIODE er den ønskede periode i sekunder og XT2 er 7372800Hz, som er frekvensen på den eksterne krystal. Da der ønskes en periode på 1 *ms* (0,001 sek.) vil resultatet være.

$$0,001 * \frac{7372800}{8} = 921,6$$

Derfor sættes TACCR0 = 922.

Grænseflader
Sequence Timer bruger Timer A for at tælle op på procestimere.

Input
Startfunktionen skal kende hvilken procestimer tiden skal sættes på og hvad tidsintervallet er. Time-out-funktionen får input om hvilken procestimer, time out flaget skal kontrolleres på.

Output
Managerfunktionen sætter 2 flag, der afgør om timerne er i gang eller udløbet og time-out-funktionen returnerer om en procestimer er udløbet.

5.2.7 Battery Calc

Batteriets spænding bliver givet som input til ADC'en i MSP'en og returnerer den digitale værdi med en præcision på 1 decimal. Batteriets spænding skal gennem en spændingsdeling på 1:10, da MSP'en kun kan måle værdier mellem 0 og 2.5 V og batteriets spænding minimum skal være 9 V.

Modulet "Battery Calc" skal hente batteriets værdi fra ADC'en. Da det er en 12 bit ADC, ligger værdien mellem 0 og 4096. For at få spændingen bruges der følgende formel:

$$\frac{2,5}{4096} * 10 * \text{DV} = \text{BS}mV$$

DV er den digitale værdi fra ADC'en og BS er batteriets spænding.

Grænseflader
Modulet "Status" henter batteriets spænding, gennem denne driver, for at sende det til lager-PC'en.

Input

"Battery Calc" får en værdi fra ADC modulet som det omregner til batteriets spænding.

Output
Modulet giver batteriets spænding som output.

5.2.8 Motor

Denne driver styrer de tre motorers fart og omdrejningsretning. Motorerne skal have et PWM signal, så hardwaren skal først sættes op så den kan generere dette. Det er valgt at Timer B bruges til at generere PWM signalet, derfor skal den sættes op til følgende:

Timer B control register (TBCTL) sættes til 0x0250. Derved sættes Timer B op til følgende:

- Hver TBCCTLx loader uafhængig af hinanden.
- En størelse på 16 bit.
- At kører på SMCLK clockken.
- Clockken divideres med 4.
- Up mode. Timeren tæller op til TBCCR0.
- Der genereres ingen interrupt.

Tidsperioden bestemmes ved at sætte TBCCR0 = 99. Det giver Timer B en tidsperiode på

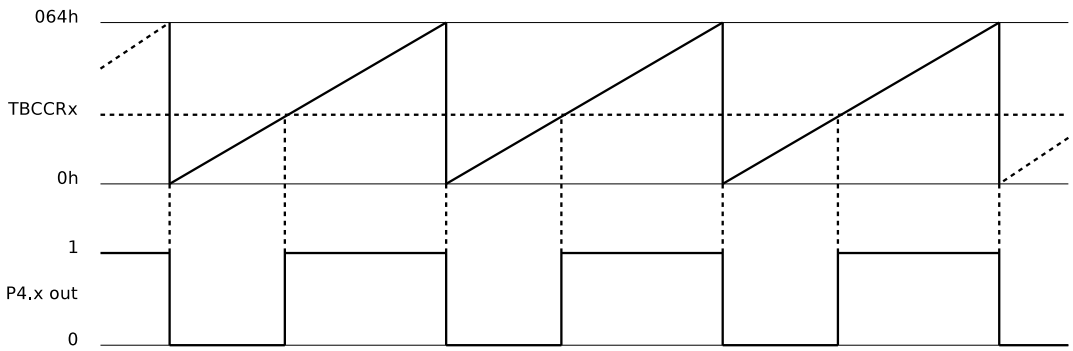
$$\frac{7372800Hz/4}{99} = 18618Hz.$$

Alle TBCCTLx sættes ens, det er kun værdien i TBCCRx der ændres for at give forskellige PWM output.

```
TBCCTLx = 0x0060;
```

TBCCTLx bliver sat til at køre Compare mode og Set/Reset. På figur 5.8 på næste side kan det ses hvordan der ved hjælp af TBCCRx kan genereres frekvensmodulering på udgangen P4.x.

Driveren bygges op med 3 switch cases, en for hver motor. Motorerne skal kunne køre både frem og tilbage, derfor konstrueres følgende cases:



Figur 5.8: Eksempel på hvordan der ved hjælp af TBCCRx kan styres tidsperioden for P4.x. Hver gang at værdien i TBCCRx er lig med Timer B, sættes porten høj og den nulstilles så igen sammen med Timer B. Derved dannes et PWM signal.

- Hvis motoren skal køre tilbage, bruges hastigheder mellem 0 og -100. Her sættes det ene TBCCR register (F.eks. TBCCR1) = 100 for slukke motoren og det andet (F.eks. TBCCR2) sættes = (100 - (hastigheden*-1)). Dette får motoren til at køre tilbage med den værdi der er angivet som argument.

- Hvis motoren skal køre frem, bruges hastigheder mellem 0 og 100. Her sættes det ene TBCCR register (F.eks. TBCCR2) = 100 for at holde det slukket og det andet (F.eks. TBCCR1) sættes = (100 - hastigheden). Dette får motoren til at køre frem med den værdi der angives som hastighedsargument.

- For hastighedsværdi på 0 eller større end 100 slukkes begge motorer ved at sætte registrene til 100.

Grænseflader
PWM-signalet bliver genereret på P4.x. De moduler der skal udføre instruktionerne fra lager-PC'en, kalder driveren for at styre motorernes hastighed.

Input
Driveren skal modtage et motornummer og et hastighedsargument mellem -100 og 100. For at lette tilgangen fra andre moduler defineres motornumrene som right, left og lift.

Output
Driveren genererer et PWM-signal ud fra inputtet, så motorerne køre med den rigtige hastighed.

5.2.9 Color Sensor

Værdien fra lyssensorerne bliver givet som input til ADC'en, som så beregner den digitale værdi. Dette modul skal ud fra de værdier ADC'en returnerer, bestemme hvilken farve de enkelte lyssensorer registrerer.

De enkelte farver tildeles et interval indenfor de 12 bit, som ADC-konverteringen giver mulighed for. ADC'en returnerer værdier mellem 0–4096, hvor 0 svarer til 0 V og 4096 til 2,5 V.

Sort tildeles et interval på 0-2000.

Blå tildeles et interval på 2000-3500.

Hvid tildeles et interval på 3500-4096.

Grænseflader
Modulerne der skal udføre instruktionerne sendt fra lager-PC'en, henter lyssensorens værdi fra denne driver for at kunne navigere rundt på lageret. Driveren henter lyssensorernes værdi fra ADC'en.

input
Color Sensor tager en af de 5 lyssensorer (A–E) som input.

Output
Den værdi der hentes fra ADC'en sammenlignes med intervallerne og der returneres bogstaverne K, B eller W for henholdsvis sort, blå og hvid.

5.2.10 Push Sensor

Dette modul skal give en høj eller lav værdi alt efter om tryksensoren er aktiveret eller ej.

Grænseflader
Driveren bruges af modulet "Pallet,get/put", så robotten ved om der en palle på gaflen.

Input
Driveren får et input fra tryksensoren.

Output
"Pallet,get/put" returnerer en høj eller lav værdi.

5.2.11 Lift Sensor

For at driveren er i stand til at bruge triptælleren, skal den kende gaffelmotorens rotationsretning. Derudfra skal der tælles op eller ned, alt efter hvilken retning gaflen er på vej i. Triptælleren giver fire pulser for hver motoromgang. Driveren skal kaldes mindst $4 pulser/omdrejning * 360 rpm_{max} = 1440$ gange i minuttet, for at tælle samtlige pulser der kommer, når motorerne kører med fuld fart. [Hur03]

Grænseflader
Driveren bruges af "Pallet,get/put", så robotten kan sætte pallen i den rigtige højde.

Input
Driveren får pulser fra triptælleren som input.

Output
Driveren returnerer en værdi mellem 0 og 130, afhængig af hvor højt gaflen er oppe på gaffeltårnet. Værdien er nul når gaflen er ved gulvet, hvilket trykknappen under gaflen indikere når dens signal bliver lavt.

5.2.12 Serial

Denne driver skal sende og modtage data via USART0. For at driveren kan fungere uden at bruge meget CPU-tid skal den deles op i en række mindre funktioner som hver udføre en del af opgaven. Driveren er delt op i en funktion til modtagelse og en til afsendelse. De har følgende opgaver:

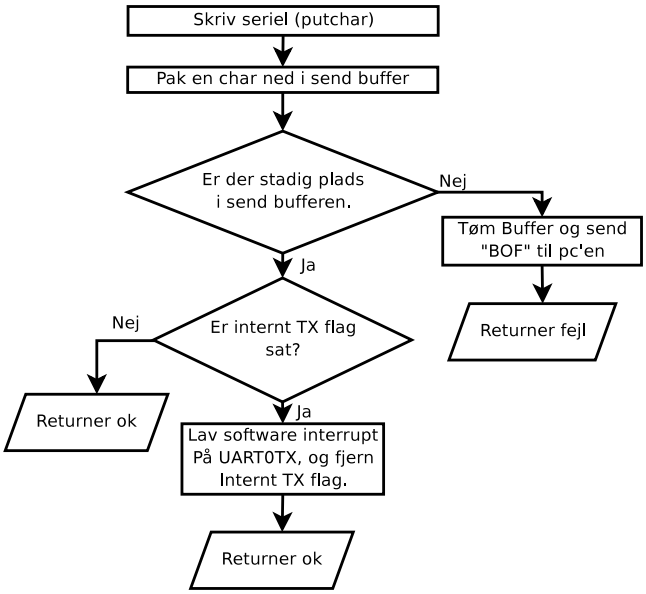
- Afsendelse af data.
  - Pakning af data til afsendelse.
  - Afsendelse af data fra sendebuffer på USART0, ved interrupt på USART0 transmit.
- Modtagelse af data.
  - Udlæsning af data fra USART0 til minimodtagebuffer, ved interrupt på USART0 receive.
  - Tolkning af modtaget data.

Pakning af data til afsendelse
Denne funktionalitet skriver det data programmet sender, ned i en buffer, der derefter

sendes videre via hardwaren gennem interrupt på UTXIFG0. Det er en modifikation af C-funktionen "putchar()", som bruges af "printf()", således at det er muligt, at sende data ved at kalde:

```
printf("tekst til afsending")
```

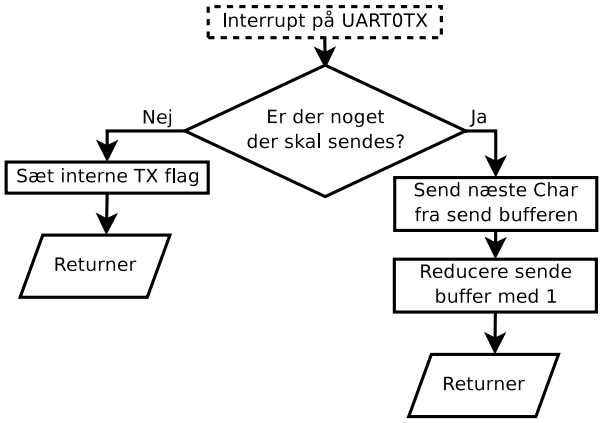
Bufferens størrelse ved afsendelse defineres til 128 Bytes. Denne størrelse er bestemt fordi data ofte kommer i "bursts" og ikke i en jævn strøm. Derfor er det vigtigt at der kan lagres en stor mængde data, der så kan behandles senere. Flowet for funktionen kan ses på figur 5.9.



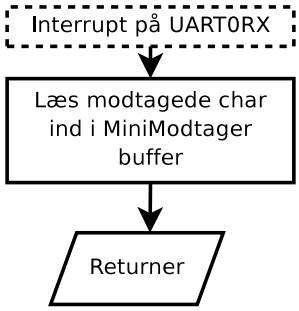
Figur 5.9: Flowdiagram for "putchar" der er en del af driveren til den serielle kommunikation

Afsendelse af data fra sendebuffer på USART0
Denne funktionalitet sender data placeret i sendebufferen ud på USART0. I tilfælde af at der genereres interrupt, men ikke er noget data at sende, er det vigtigt at kunne holde styr på om der har været et interrupt (dvs. om USART0 er klar til at afsende data), da det ikke bliver sat igen før der har været afsendt data. Til dette er det valgt at anvende et internt TX-flag, der således holder styr på dette. Figur 5.10 på næste side viser afsendelse af data på USART0.

Udlæsning af data fra USART0 til minimodtagebuffer
Denne funktionalitet bliver kørt hver gang, der modtages data via den serielle forbindelse, og der derfor genereres interrupt på USART0RX. Hovedopgaven er at kopiere det modtagende data over i en minimodtagebuffer, og derved gøre plads



Figur 5.10: Flowdiagram for interrupt på UTXIFG0, der skriver data fra sendebufferen ud på USART0



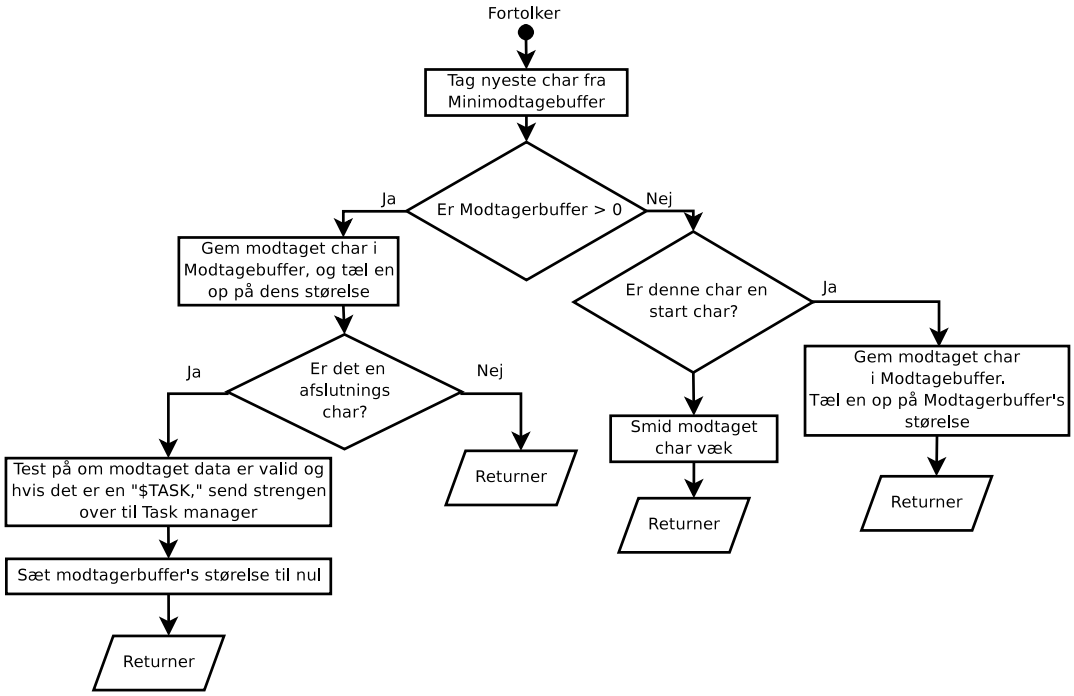
Figur 5.11: Flowdiagram for interrupt på URXIFG0, der skriver data fra sendebufferen ud på USART0

til modtagelse af næste data byte. Flowet for funktionen er vist på figur 5.11.

Det er vigtigt at denne funktion er hurtig, da den køres under en interruptrutine, og derfor afbryder andre funktioner, som f.eks. en analog måling eller tidsafhængige operationer.

Tolkning af modtaget data
Denne funktionalitet læser data fra minimodtagebufferen indtil der modtages en startkarakter ("$"). Flowet for udførelsen af funktionen kan ses på figur 5.12 på næste side.

For at kunne anvende disse funktioner, er det nødvendigt at konfigurere den serielle forbindelse som følgende:



Figur 5.12: Funktion henter data fra minimodtagebufferen og tester om det er gyldigt. Hvis det modtagne data er gyldigt afgøres hvilket modul det skal sendes videre til.

- `U0CTL &=~SWRST;` genererer et software reset, for at sikre at alt er nulstillet.
- `U0CTL = 0x50;` indstiller USART0 til at køre med 8 databit, 1 stopbit og ingen paritetscheck.
- `U0TCTL = 0x30;` sætter USART0 til at køre på SMCLK.
- `U0BR0 = BAU_R0_19200;` og `U0BR1 = BAU_R1_19200;` sætter USART0 til en baudrate på 19200bps, udregnet ved SMCLK/19200.
- `U0MCTL = 0x00;` sikrer at USART0 kører uden modulation.
- `ME1 |= UTXE0+URXE0;` aktiverer transmit og receive på USART0.
- `P3SEL |= BIT4+BIT5;` sætter pin P3.4 og P3.5 til at være forbundet til USART0.
- `P3DIR |= BIT4;` sætter pin P3.4 til udgang.
- `IE1 |= (URXIE0 + UTXIE0);` aktiverer interrupt på USART0 transmit og receive.

Grænseflader
Driveren bliver brugt når der skal sendes eller modtages data på den serielle forbindelse på USART0.

Input
Driveren tager det data der bliver sendt til MSP'en på USART0 som input.

Output
Det data driveren modtager bliver lagt i en kø på MSP'en, hvis det er korrekt, så det kan hentes af andre.

5.2.13 ADC

Dette modul er et hardware-modul der skal aflæse analoge spændingsværdier på ADC portene og konvertere dem til digitale værdier som MSP'en kan arbejde med.

Der sættes 8 porte op til ADC, som skal samples konstant i forhold til en 2,5 V referencespænding. Hver gang der er samples gemmes resultatet midlertidigt i registerene ADC12MEMx. For at lette tilgangen til ADC konverteringen defineres sigende navne for de enkelte ADC porte.

ADC konverteringen initieres ved at:

- Sætte alle 8 af P6's input-ben høje ved `P6 = 0xFF`.
- Sætte ADC kontrol register 0 til at køre med samtidig konvertering, referencespænding 2,5 V samt tænde ACD12 modulet ved `ADC12CTL0 = 0x11F0`.
- Sætte ADC kontrol register 1 til at gemme de enkelte resultater i ADC12MEM0–ADC12MEM7, sample i forhold til intern timer, bruge en clockdivider på 4 og bruge den interne ADC12 oscillator clock, samt konstant at sample på alle kanaler. Dettes gøres ved at sætte `ADC12CTL1 = 0x0266`.
- Vælge input kanalerne A0-A7 fra P6, sætte de enkelte kanaler til en referencespænding på VR+ = Vref, VR- = AVss. For første kanal sættes `ADC12MCTL0 = 0x10` hvor bit 0-3 er input kanalen. Bit 0-3 sættes én højere for hver efterfølgende kanal. Det sidste register får endvidere sat et EOS bit, for at vise at det er sidste register i samplingen.

Til slut "OR'es" ADC kontrol register 0 med 0x0003 for at sikre at samplingen startes. Modulet tager et kanalnummer (som er defineret til et navn) som argument og returnerer den værdi, som står i registret for dette kanalnummer.

Grænseflader
ADC bruges af “Color Sensor” og “Battery Calc”

Input
Modulet tager et kanalnummer (som er defineret til et navn) som argument

Output
Modulet returnere den digitale værdi, som står i registret for kanalnummeret.

Kapitel 6

Programdesign: Lager-PC

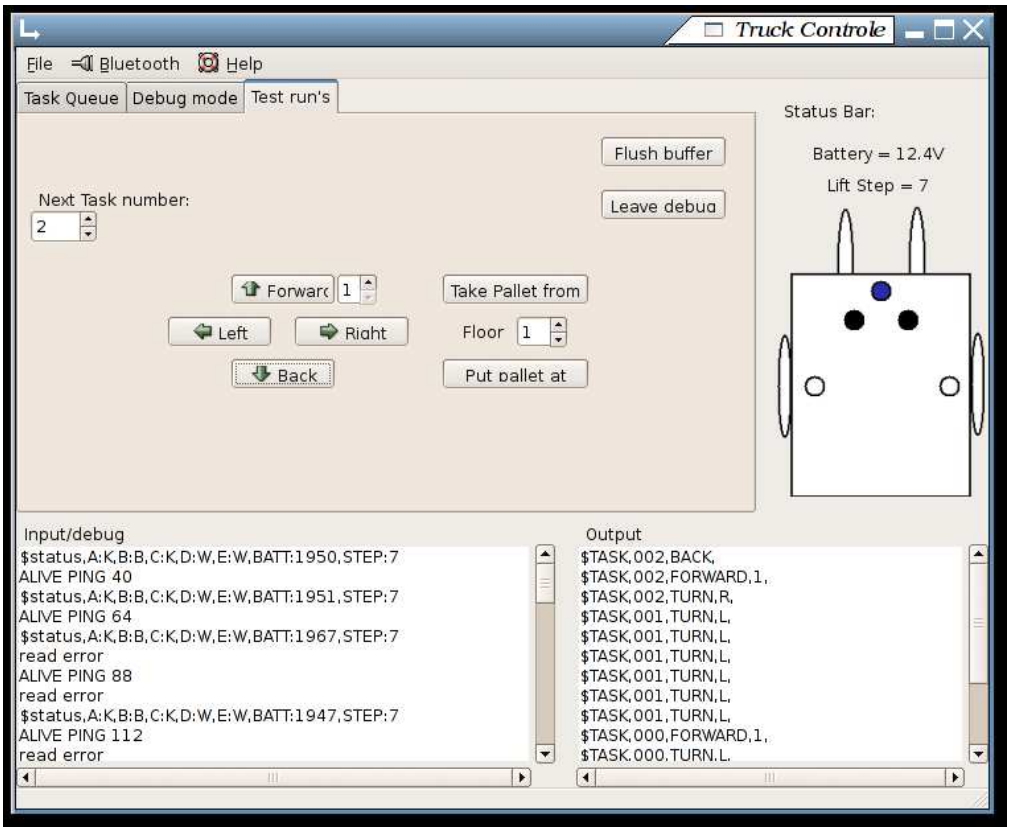
Eftersom der er lagt fokus på mikroprocessoren i dette projektforløb, er der valgt kun at give et hurtigt overblik over softwaren til lager-PC'en. I dette afsnit bliver der kort gennemgået hvilke værktøjer der er blevet anvendt, hvilke overvejelser der har lagt til grund for softwaren og hvordan det hele hænger sammen. På figur 6.1 på den følgende side kan ses et udkast til det færdige PC program for lager-PC'en.

6.1 Værktøjer

- GUI
  For det første skal der anvendes et GUI, ved hvilken det er muligt at se status, indtaste opgaver, reagere på input osv. På dette område faldt valget på GTK ud af mange muligheder, bl.a. GTK, QT, POSIX, NCurses.

- Opbygning af GUI
  Der anvendt Glade og LibGlade, til hhv. at opbygge brugerfladen og til at *parse* den dynamisk ved kørsel.

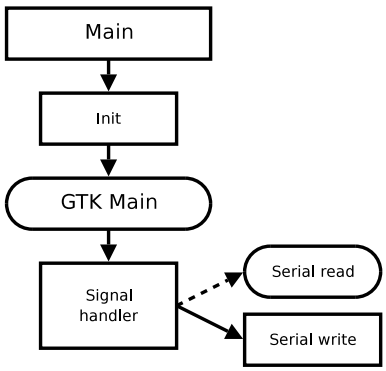
- Tråde
  Fordi det er nødvendigt at anvende tråde til bl.a. den serielle kommunikation, er der anvendt pthreads og GDK, der muliggør oprettelse og adskillelse af tråde. En tråd er en proces der kan køre selvstændigt og uafhængigt af resten af programmet.



Figur 6.1: PC programmet, hvor de enkelte værdier for status kan ses. Forneden ses input- og outputvinduet.

6.2 Seriel kommunikation

For at kunne kommunikere uden væsentlige forsinkelser, er det vigtigt, at den data der udveksles, opfanges med det samme. Derfor er det nødvendigt at have en proces til at læse data ud fra den serielle forbindelse konstant. Problemet ved dette er at alt andet kode ikke bliver afviklet i mellemtiden. Løsningen på problemet er brugen af tråde, for ikke softwaremæssigt, at skulle tage højde for veksling af ressourcerne imellem kommunikationen og resten af programmet. Der oprettes to tråde; en til GTK main der er hovedløkken i programmet, og en til aflæsning af data på den serielle forbindelse. Det overordnede flow i programmet, kan ses på figur 6.2 på næste side.



Figur 6.2: Figuren viser det overordnede flow i PC softwaren. I trin 3 bliver der startet en tråd til `GTK main`. Denne tråd er en løkke der kører kontinuerligt og styrer alle opgaver med hensyn til brugerinterfacet. Til læsning af den serielle data der ankommer til lager-PC'en, bliver `Serial read` startet i en ny tråd, der også kører kontinuerligt.

6.3 Lagring af opgaver

For at kunne lægge opgaver i kø, til udførsel på et senere tidspunkt, er det også vigtigt at have et lager til disse. En oplagt løsning er at anvende en database til formålet. Dernæst kan der tilføjes opgaver og hentes opgaver, i den rækkefølge og til den tid, de skal dekodes til simple instruktioner og sendes. Der anvendes en MySQL database til formålet.

6.4 Navigation

For ikke at lægge navigationsalgoritmerne og ressourcerne over på mikroprocessoren bliver dette håndteret på PC siden. Begrundelsen er at der er en meget begrænsede mængde lagerplads til rådighed på mikroprocessoren. Derudover vil kompleksiteten bliver højere og mulighederne begrænsede ved udbygning af disse algoritmer.

Navigationen skal fungere ved at lageret er opbygget modulært, således at pallepladserne er grupperede og unikt navngivet (eksempelvis niveau 2, hylde 17, række J, område 12, lager 1, osv.). Der er herefter to umiddelbare metoder til at finde vej. Den mest simple af metoderne, er at lade alle pallepladserne, samt vejen til dem (i simple instruktioner) fra et givent udgangspunkt, stå i en database og derfra trække oplysninger ud. Denne måde vil medføre et stort arbejde ved udbygn-

ing/ombygning af lageret, samt meget spildkørsel ved at der hele tiden skal returneres til udgangspunktet.

På den anden metode vil det ikke være nødvendigt at returnere til udgangspunktet, da det allerede er kendt ud fra navnet på den nuværende position. Herved er det muligt at nøjes med at returnere til den mindste fællesnævner (niveau, hylde, række, område osv.) og derfra finde vejen. Derudover skal der kun stå i databasen hvorledes robotten kommer fra en vilkårlig gruppe til gruppen over denne, samt alle undergrupper.

Kapitel 7

Accepttest konklusion

I dette afsnit vil der, udfra accepttesten afsnit 2.3 på side 22 og udførelsen af den, bilag A på side 85, blive diskuteret hvorlangt udviklingen af systemet er nået. Denne gennemgang vil omhandle Lager-PC'en, MSP'en og hardwaren, i nævnte den rækkefølge.

Lager-PC
Softwaren til lager-PC'en er en test version på nuværende tidspunkt. De grundlæggende funktioner, som at kunne generere forståelig data til MSP'en samt at sende og modtage data via den serielle kommunikation, er blevet udviklet. Desuden kan lager-PC'en afkode statusstrengen som MSP'en sender en gang i sekundet til brugbare informationer for brugeren. Muligheden for, at en bruger kan tilføje og slette hele opgaver, samt muligheden for at tilføje enkelte opgaver er endnu ikke udviket. Den videre udvikling af denne software vil være, at implementere opgavehåndtering og tilføje et billede af lageret, så softwaren bliver brugervenlig.

MSP'en
Softwaren til MSP'en kan kontrollere robotten, samt kommunikere via en seriel bluetooth forbindelse. Desuden fungere MSP'ens styring og indhentning af data fra den tilsluttede hardware, og den udsender fejlbesked, når den indhenter data fra sensorerne, som ikke stemmer overens med de ønskede data.

Softwaren er ikke helt optimeret til styringen af robotten, hvilket kan få den til at køre ud af kurs og hjælp fra brugeren er nødvendigt. Selve styringen, der får robotten til at følge stregerne på gulvet virker effektivt. Det kan ses på videooptagelser af robotten på den vedlagte CD. MSP'en kan tabe data i den serielle modtagelsen, når at bufferen bliver overfyldt . Håndteringen af paller fungerer efter de grundlæggende ideer, der mangler dog noget sikkerhedskontrol, så robotten ikke bare kører frem i

uendelighed hvis der ingen palle er på gaflen. Robottens håndteringen af paller kan ligeledes ses på videooptagelser på den vedlagte CD.

Hardware

Lyssensorerne registrerer de forskellige farver og signalet til MSP'en overholder alle krav. Bredden af linierne på testbanen er ikke optimal i forhold til afstanden mellem lyssensorerne monteret på robotten, så robotten kan nogle steder, at køre meget skævt i forhold til linien, før den begynder at rette op. Det kan resultere i at robotten mister orienteringen.Problem er størst i kryds og når pallerne skal stilles eller hentes, altså i skiftet mellem opgaver. Triptælleren og tryksensorerne, til styringen af gaflen, fungerer efter hensigten og overholder alle kravene for signal til MSP'en. Triptællerens præcisionen af højden på gaflen stemmer ikke helt overens med virkelig, set i forhold til det slør der er i gearingen mellem motoren og gaffeltrækket.Det batteri der bruges til at forsyne robotten, overholder de krav der er stillet til det, dog kommer der ingen alarm når batterispændingen er blevet lav.

Kapitel 8

Konklusion

I dette projekt var målet at udvikle et styresystem til en autonom lagerrobot, som skal udfylde samme funktion som en manuelt styret truck. Dette system skulle udvikles til mikroprocessoren MSP430F149. For at teste dette system, skulle der udvikles en prototype af en lagerrobot samt software til en lager-PC, som skulle gøre det muligt for brugeren at give systemet opgaver. For at få lagerrobotten til at udføre disse opgaver, skulle den være udstyret med forskellige sensorer, motorer og andre komponenter.

Derfor blev der i indledningen, kapitel 1 på side 7, opstillet følgende initierende problem: “Hvordan konstrueres og programmeres et system til en lagerrobot, så opgaverne på et lager kan automatiseres?”

Det ønskede system opdeles i følgende dele:

- Software til lager-PC
- Software til mikroprocessoren
- Prototype af lagerrobot med tilhørende hardware

I udviklingsfasen er der taget udgangspunkt i SPU-modellen, for at opnå en struktur i dokumentationen af systemet. På baggrund af temaet for Datateknik 3. semester, blev der lagt vægt på at udvikle softwaren til mikroprocessoren.

For hver del er der blevet opstillet krav som det færdige system skal overholde. Lager-PC’en skal kunne sende opgaver til mikroprocessoren opdelt i simple instruktioner og modtage forskellige statusmeddelelser. Mikroprocessoren skal få lagerrobotten til at udføre disse opgaver, som består i at flytter paller fra et sted på lageret

til et andet. Det skal gøres ved at få robotten til at følge en streg på gulvet, ved at benytte lyssensorer.

Den udviklede software til mikroprocessoren, er blevet opdelt i processer og drivere, for lette implementation af de funktionaliteter som blev opstillet i kravspecifikationen. Gennem en accepttest er der blevet testet om de opstillede krav bliver overholdt, og det konkluderes at de grundlæggende krav er opfyldt. Dette uddybes i forbindelse med perspektiveringen, kapitel 9 på næste side.

Da der bliver lagt vægt på software til mikroprocessoren, er den ønskede software til lager-PC'en ikke færdigudviklet. Det er derfor ikke muligt at tilføje en opgave ved at specificere pallens placering, destination og prioritet, men kun at lægge instruktioner i programmets database. Derudover mangler der at blive implementeret et kort over lageret som kan vise hvor lagerrobotten befinder sig.

Under konstruktionen at prototypen til lagerrobotten, har der været problemer med hardwarens stabilitet, bl.a. løse forbindelser og støj fra motorstyringen. Disse problemer er blevet løst tilfredstillende, og der er opnået et stabilt kredsløb på robotten.

Samlet kan det konkluderes at det er muligt at konstruere og programmere et system til en lagerrobot, så opgaverne på et lager kan automatiseres. Dette kan gøres ved at programmere en mikroprocessor til at styre lagerrobotten og udvikle et PC-program så brugeren kan overvåge robotten og give den opgaver.

Kapitel 9

Perspektivering

Lagerrobot

I projektet er kravene fra kravspecifikation tildels opfyldt. De dele af kravspecifikationen der ikke er blevet opfyldt er f.eks. kommunikationen. Det er muligt at sende instruktioner til lagerrobotten. men det sikres ikke i alle tilfælde at den behandler instruktionen rigtigt. Ved f.eks. “FORWARD” kontrolleres det ikke om der kommer et parameter med. Denne del kan udbygges så lagerrobotten kontrollerer om den modtager korrekte tekststrenge. Der mangler ligeledes fejlhåndtering f.eks. hvis lagerrobotten mister forbindelsen til lager-PC’en.

Der mangler også at blive opfyldt nogle af kravene i hardwaredelen. I den endelige udgave som skal bygges på en lagertruck, skal det sikres bedre at der ikke vil opstå problemer med sensorer eller bluetooth. I forbindelse med bluetooth ville det i dette tilfælde sandsynligvis være en mere holdbar løsning at bruge trådløst netværk, hvor man kan sætte flere hotspots op, der kan sikres at der altid vil være forbindelse til lagerrobotten.

Hvis man vælger at bruge trådløst netværk vil det også ændre en del på komminikations siden, da det kører via TCP-IP. Grundlæggende for alle løsninger er at de skal fungere vha. radiobølger. Inden for dette område er der mange muligheder, f.eks. GSM netværk, som de fleste mobiltelefoner bruger i dag.

En anden del at hardwaren som kunne forbedres er, som nævnt, sensorene. Hvis denne løsning skulle bruges på en lagertruck, skulle den udbygges med et kamara der kan bruges hvis robotten kommer uden for stregerne.

En anden forbedring kunne også være at den kunne registre hvis der er forhindringer

i vejen for den, f.eks. mennesker eller paller. Dette vil kræve at der laves en hardwareløsning, der kan måle afstanden til objekter foran robotten. Det er også teknisk muligt vha. kraftige lyssensorer, men de vil kunne dække et lille område foran robotten, derfor kunne dette evt. også løses med et kamera der tager billeder af hvad der forgår foran robotten.

Der er mange måder at forbedre systemet og løse de forskellige problemer på. Hvilken løsning der vælges afhænger af hvilket miljø robotten skal fungere i. Hvis det er et større lager er kravene sikkert højere end i et lille lager.

PC Software

Den PC Software, der er blevet udviklet i dette projekt, opfylder meget få af de krav der blev opstillet i kravspecifikation. Det ville derfor være det område der først og fremmest skulle udvikles.

For at kunne opfylde kravspecifikation skal der tilføjes en database til programmet, og en algoritme, der kan udregne hvilke instruktioner, der får lagerrobotten til at udføre opgaven korrekt. Derudover mangler muligheden for at vise hvor lagerrobotten befinder sig på et kort. Der mangler generelt en brugertest på lager-PC'ens software, så den kan forbedres i forhold til brugervenligheden.

Hvis dette udbygges vil kravspecifikation være opfyldt, men der vil stadig være funktioner som mangler hvis systemet skal kunne bruges tilfredsstillende. Funktioner der kan tilføjes er f.eks. de følgende:

- At styre robotten manuelt. Til dette skal der laves en mobil enhed, så operatøren kan tage det med når robotten skal styres manuelt.
- Bedre håndtering af fejl, så operatøren nemmere kan udbedre fejlen.
- En toturial til PC programmet, så brugeren nemt kan lære funktionerne i programmet.

Litteraturliste

[BS02] Stephen Biering-Sørensen. *Struktureret Program-Udvikling*. Ingeniøren|bøger, 1 edition, 2002. ISBN: 87-571-1046-8.

[Bør99] Kim Sejr Børsen. *Robotter hindrer jobflugt og udflytning*. `http://www.proinvent.dk/medieomtale/1999/artikel990708_2.asp`, 1999. Hentet d. 30. nov. 2005.

[Cle97] A. Clements. *Microprocessor Systems Design*. Thomson, 3. udgave 1997 edition, 1997. ISBN: 0-534-94822-7.

[ele00] ST Micro electronics. *L298 Dual Full-Bridge Driver*. http://www.st.com, 2000.

[Hur03] Philippe Hurbain. *Lego 9V Technic Motors compared characteristics*. `http://www.philohome.com/motors/motorcomp.htm`, 2003. Hentet d. 30. nov. 2005.

[Ins03] Texas Instruments. *Data Sheet for MSP430x13x, ,MSP430x14x Mixed Signal Microcontroller*. http://www.ti.com, 2003.

[Ins04] Texas Instruments. Msp430x1xx family user's guide. http://www.ti.com, 2004.

[MAX01] DALLAS Semiconductor MAXIM. *Understanding SAR ADCs*. `http://www.maxim-ic.com/appnotes.cfm/appnote_number/1080`, 2001. Hentet d. 2. dec. 2005.

[Nag03] Chris Nagy. *Embedded Systems Design using the TI MSP430 Series*. Newnes, 2003. ISBN: 0-7506-7623-X.

[Sem97] Philips Semiconductors. *Datasheet - BYV28 series - Ultra fast low-loss controlled avalanche rectifiers*. http://www.philips.com, 1997.

[Sie] Siemens. *GaAIAs Infrared Emitter - SFH487*. http://www.alldatasheet.com. Hentet d. 9. dec. 2005.

[Sie97] Siemens. *Special Economic Hall-Effect IC TLE4905*. http://www.alldatasheet.com, 1997. Hentet d. 21. okt. 2005.

[Sie99] Siemens. *NPN-Silizium-Fototransistor - SFH309*. http://www.alldatasheet.com, 1999. Hentet d. 9. dec. 2005.

[Swi04] Sebastian Swiatecki. Robotterne er blevet flere og billigere. *Ingeniøren*, 2004. Skrevet d. 22. okt. 2004.

[Tan05] Andrew S. Tanenbaum. *Structured Computer Organization*. Pearson Prentice Hall, 5 edition, 2005. ISBN: 0-13-148521-0.

Appendiks A

Udførelse af accepttest

I dette afsnit udføres accepttesten på baggrund af accepttestspecifikationen i afsnit 2.3 på side 22. Testen er stillet op således, at der er en lige linie mellem specifikationen og testen, i form af de enkelte testpunkter. De enkelte test er udført 10 gange i træk, for at få et pålideligt resultat.

Lager-PC

Tilføj opgave

1. PC programmet testes for om der kan tilføjes en ny opgave.
    - Da denne funktion ikke er udviklet i den nuværende version af PC programmet, er det ikke muligt at tilføje en ny opgave.
2. PC programmet testes for om det muligt at fjerne en opgave fra databasen.
    - Da denne funktion ikke er udviklet i den nuværende version af PC programmet, er det ikke muligt at tilføje en ny opgave.
3. PC programmet testes for om det kan vise status i afvikling af opgaver.
    - Vinduet til opgaveafvikling findes i programmet, men opgavehåndteringen er ikke færdiggjort.

Send opgave

1. Det verificeres at PC programmet kan konstruere ruten mellem nuværende og kommende placering.

   (a) PC programmet kan endnu ikke konstruere ruten, da vejfindingsalgoritmen ikke er udviklet.

2. Hver gyldig instruktion med en gyldig parameter og instruktionsnummer, sendes til robotten. Det verificeres at robotten godkender instruktionen.

   (a) Ved at give robotten en instruktion med parameter gennem vinduet "Test run's" (f.eks "Forward" med parametren 2), sendes strengen: `$TASK,-001,FORWARD,2,` , hvilket kan ses i outputvinduet. I inputvinduet ses det at robotten returnerer "Task added".

3. PC programmet testes for om det modtager en fejl fra robotten, hvis der sendes en ugyldig instruktion med ugyldig parameter eller instruktionsnummer.

   (a) Ved at sende en ugyldig instruktion igennem et terminalvindue til robotten som f.eks. `$abc123` , returnerer robotten "Task not valid".

4. Fra terminalvinduet sendes der en tilfældig tekststreng, uden "$", til lagerrobotten. De kontrolleres at der ikke modtages svar og robotten ikke reagerer.

   (a) Ved at sende en tilfældig tekststreng igennem et terminalvindue til robotten som f.eks. `abc123`, kommer der intet svar fra robotten og den reagerer ikke fysisk.

Vis status

1. Under udførelse af instruktioner med robotten, kontrolleres det at det er muligt at se robottens status.

   (a) I PC programmet kontrolleres de viste værdier i "Status bar". Det kontrolleres at lyssensorerne, batterispænding og gaflens position stemmer overens med de faktiske værdier som robotten har. Se figur 6.1 under afsnit 6 på side 73 om programdesign af lager-PC'en .

2. I PC programmet kontrolleres om robottens position på kortet passer med robottens faktiske position.

   (a) Da denne funktion ikke er udviklet i den nuværende version af PC programmet, er det ikke muligt at se robottens position.

Alarmér

1. Det kontrolleres at PC programmet reagerer korrekt hvis der modtages en alarm. Hvis robotten placeres udenfor stregen eller skubbes af stregen, kontrolleres det, at:

   (a) Der modtages en fejl fra robotten.

   - Hvis robotten skubbes af stregen, så der alle de forreste sensorer er hvide, sendes en fejl fra robotten. Fejlen "ERROR nr x" ses i inputvinduet. Der er endnu ikke udviklet en rød alarmknap i programmet.

   (b) Robotten stopper.

   - Hvis robotten skubbes af stregen, stopper robotten den instruktion den er i gang med.

   (c) Robotten slår over til manuel styring.

   - Robotten sender "Enter debug" ved fejl, dette ses i inputvinduet.

Mikroprocessoren

Udfør opgave

1. Det verificeres at robotten kan modtage op til 20 instruktioner.

   (a) Igennem PC programmet sendes 20 instruktioner til robotten. Robotten returnerer for hver instruktion "Task added". Hvis alle 20 instruktioner sendes på en gang returnerer robotten "Buffer overflow" da der kun er plads til 128 karakterer i bufferen, som ikke tømmes med samme hastighed som der sendes data.

2. Det verificeres at robotten udfører det forventede ved hver instruktion.

   (a) Ved 20 instruktioner kan robotten ved forsøg i gennemsnit gennemføre 7 instruktioner inden der sker fejl. Disse fejl skyldes at sensorerne ikke har den rigtige værdi, ved start af en ny instruktion. Det ses endvidere i inputvinduet at der modtages 'Task done" samtidig med at robotten har udført en instruktion. Se nærmere på videoen på den medfølgende CD.

Overvåg sensorer

1. Der kontrolleres om sensorerne fungerer korrekt. Ud fra testbeskrivelsen i afsnit 2.3 på side 22 Overvåg sensorer 1. a–g er følgende testet:

(a) Robotten kan køre frem på en streg og korrigere efter den. Se evt. videoklip på den medfølgende CD.

(b) Robotten kan stoppe i et kryds og returnere ”Task done”.

(c) Der ses ud fra “Status bar” i PC programmet at lyssensorerne har værdier for de tre farver. Det ses at disse værdier passer med de faktiske farver under sensorerne.

(d) Når robotten kommer til en advarselsfarve er det testet at robotten stopper og sender “Task done”.

(e) Igennem en softwaretæller tælles de enkelte tegn i 1 minut. Ved at gøre dette ses det at der samples på lyssensorerne 100 gange i sekundet.

(f) Højden på gaflen har i ét tilfælde en værdi på “Lift = 7” i “Status bar” i PC programmet. Dette svarer ca. til den faktiske højde, da et step svarer til 2,3 *mm* og gaflen er målt til at have en højde på 1,5 *cm*.

(g) Det kan ses på videoen på den medfølgende CD, at robotten stopper med at køre frem når tryksensoren detektere en palle.

Send status

1. Der kontrolleres at status sendes korrekt.

   (a) Igennem en softwaretæller tælles status i 5 minutter. Ved at gøre dette ses det at status bliver sendt 1 gang i sekundet.

   (b) Ved at sende en gyldig instruktion til robotten, returnerer denne de forventede resultater (“Task added og Task done”).

   (c) Når der opstår en fejl returnerer robotten “ERROR nr. in task nr.” som kan ses i inputvinduet.

Hardware

1. Det kontrolleres at hardwaren fungerer korrekt.

   (a) Ved at holde lyssensorerne op imod dagslys ses det at værdien svinger +-300 *bit*, som svarer til +-0,24 V.

   (b) Lyssensorerne giver en værdi på mellem 0,12 V og 2,4 V. Tryksensoren giver en værdi på mellem 0,02 V og 3,26 V. Triptælleren giver en værdi på mellem 0,02 V og 3,13 V.

   (c) Ud fra målinger giver triptælleren 1 puls / 2,3 *mm*.

(d) Ved måling af spændingen på batteriet ses det at spændingen er mellem 10 V og 12 V.

(e) Ved at give motorerne en spænding på minimum 1 V ses det at de kan trække robotten og gaflen.

Appendiks B

Støjproblemer på komponenterne

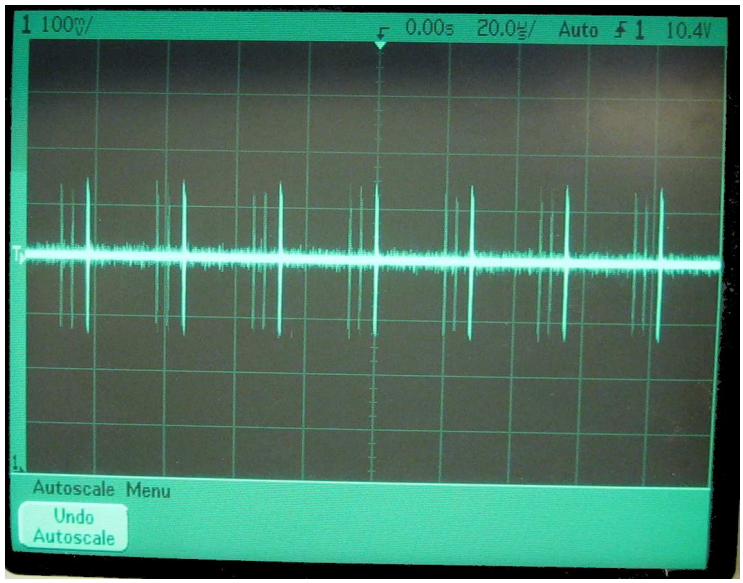
Dette afsnit indeholder en gennemgang af hardwaren til lagerrobotten. Formålet med afsnittet er at nævne de problemer der har været i opbygningen af hardwaren.

Det første der blev tilsluttet MSP'en var Bluetooth enheden, der blev ikke konstateret nogen fejl ved tilslutningen, men at bluetooth'en brugte meget strøm. For at lette debugningen af softwaren til MSP'en, blev der tilsluttet et to linjers display, hvor det var muligt at udlæse variabler i realtid, samt en drejeknap til at styre displayet.

Derefter blev elektronikken til motorstyringen tilsluttet, hvorefter der begyndte der, at opstå enkelte fejl, der enten fik MSP'en til at fryse, displayet til at fejle eller bluetooth enheden til af miste forbindelsen. Fejlene kom desuden hvis der blev holdt på motorerne når de kørte eller bare dreje dem rundt med håndkræft.

For at løse problemet blev der monteret nogle aflastningskondensatorer til at dæmpe støjen fra motorerne, hvilket hjalp lidt på problemet, uden dog helt at løse det. Efter en gennemgang af de komponenter der blev brugt, samt diverse målinger af spændinger og strømme, viste det sig at problemet lå i, at ledningerne brugt til systemet var tynde wrap ledninger. Når bluetooth enheden og motorerne trak strøm, begyndte wrap ledningernes modstand derfor at have betydning. Der blev målt et spændingsfald på mere et volt over en wrap ledning på 7 cm. Ledningerne blev udskiftet til nogle tykkere, de steder hvor der sad strømkrævende komponenter og der blev monteret effektdioder på h-broen for at undgå støj fra motorerne. Støjen blev herefter så lille, at den ikke længere påvirkede komponenterne, selv ved forsøg på at fremprovokere det. hvilket kan ses på figur B.1 på næste side

Lyssensorerne blev derefter monteret på robotten. Disse er mere følsomme overfor støj da, deres arbejdsområde er på 2,5 V, og deres signal til MSP'en er analogt. Derfor var det nødvendigt også her at montere aflastningskondensatorer.



Figur B.1: Støj målt hvor batteriet er tilsluttet. Støjen er målt efter monteringen at tykkere ledninger, kondensatorer og dioder. Som det kan ses kommer en puls på knap 150 mV hvert 28 µS . Denne støj kommer fra h-broen, og ændre sig alt afhængig af hvordan PWM pulsen ser ud.

Systemet blev testet for støj igen da alle komponenterne var tilsluttet, og det blev konstateret at påvirkningen var så lille at den var uden betydning.

Appendiks C

Målejournal for lyssensorer

I dette appendiks vil måleopstillingen og selve forsøget blive beskrevet og kommenteret.

Formål

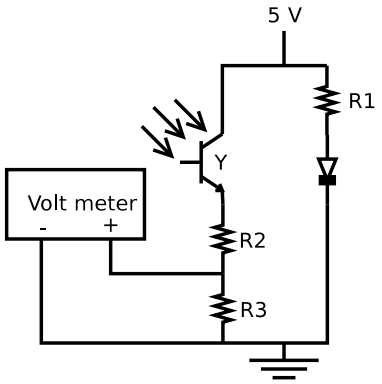
Formålet med denne måling er at undersøge hvilke farver der kan skelnes fra hinanden ved hjælp af refleksion af infrarødt lys. Desuden ønskes der fundet modstandsværdier, som giver den størst mulige spændingsforskel mellem farverne.

Apparaturliste

- Strømforsyning indstillet på 5V (AUC LBNR 08167 Inst. 8 B1-141-D-5).
- Digitalt voltmeter indstillet på et måleområde på 2-20V DC (Meterman 5XP).
- Et stykke hvidt papir med 5 forskellige farver isoleringstape.
- Testprint med IR-emitter (SFH487-2), fototransistor (SFH309FA) og to variable modstande af størrelsen $1k\Omega$ (R1) og $100k\Omega$ (R2+R3) Se måleopstillingen på figur C.1 på modstående side.
- Testbil bygget af Lego med testprintet monteres således at IR-emitteren og fototransistoren har en afstand på 1,5 cm til papiret.

Måleopstilling

På figur C.1 ses måleopstillingen.



Figur C.1: Billede af måleopstilling

Målebeskrivelse

Testbilen køres hen over en af farverne, således at IR-emitteren og fototransistoren befinder sig lige oven over den valgte farve. Herefter aflæses spændingen på voltmeteret. For hver enkelt farve gentages målingen med nye modstandsværdier.

Målinger

De målte spændinger kan ses i skemaet.

Evaluering af målinger

I forsøget kan der have været tale om følgende fejlkilder.

Fejlkilder
Da det var svært præcis at se om IR-emitteren fototransistoren var oven over farverne, kan det være en fejlkilde at denne indstilling ikke har været præcis. Vi vil dog vurdere, at denne fejlkilde er ubetydelig, da resultaterne stemmer godt overens med hinanden.

R1 [Ω]	R2+R3 [kΩ]	Gul [V]	Sort [V]	Grøn [V]	Rød [V]	Blå [V]	Hvid [V]
200	40	2,42	1,08	2,41	2,41	2,37	2,42
500	40	1,62	0,41	1,30	1,41	0,99	2,03
700	40	1,13	0,30	0,88	0,94	0,68	1,38
200	60	2,42	1,29	2,42	2,42	2,41	2,43
500	60	2,29	0,60	1,90	1,97	1,44	2,40
700	60	1,60	0,42	1,32	1,40	1,10	2,04
200	80	2,43	1,72	2,43	2,43	2,42	2,44
500	80	2,40	0,82	2,38	2,39	1,88	2,42
700	80	2,17	0,58	1,87	1,87	1,37	2,39

Tabel C.1: Måleresultater. Det ses at ved lave modstandsværdier for dioden vil den målte spænding ikke være nævneværdig forskellig. For fototransistoren ses det også at en høj modstandsværdi giver størst spredning for spændingsværdierne.