

Figur 1: Første robot

Figur 2: Differentiale og rotationssensor

1 Første design

1.1 Første design af robotten

I det første design af robotten havde vi nogle krav vil ville have opfyldt: robotten skulle køre meget præcis og tage højde for den resterende upræcise bevægelse ved at justere for den.

1.1.1 Sensorer

Omgivelserne skulle detekteres ved brug af to lyssensorer og en rotationssensor. Ideen var at lyssensoren skulle sidde uden for gadeelementerne for at detektere features, se figur 1, mens rotationssensoren skulle bruges til at beregne positionen indenfor et enkelt gadeelement.

1.1.2 Akuatorer

Robotten skulle fremdrives af to motorer, et per hjul. Da vi kun havde en rotationssensor til rådighed, samtidigt med at vi ville måle, hvor langt motorerne havde bevæget robotten blev nødt til at finde en løsning. Et differentiale, hvor de to motorer sættes på hver sin akse, tilsluttes rotationssensoren. Drejer enten den højre motor, den venstre motor eller begge sammen

vil rotationssensoren ligeledes dreje, se figur 2. Med præcision som et af de store krav har vi valgt at gear hjulene meget ned i forhold til rotationssensoren, for at få en højre opløsning.

Figur 3: Initiale gadeelementer

1.2 Første design af byen / gaderne

Vi ville prøve at bruge de Legos egne gadeelementer, men de viste sig at detektion af farverne på disse plader var alt for usikker, idet råværdierne for grøn og grå lå for tæt op ad hinanden. Vi valgte derfor at designe vores egne gadeelementer, sort på hvid baggrund, og bruge de originale lego plader som forbillede. De fire elementer er et sving, et T-kryds, et almindeligt kryds og et lige stykke vej, se figur 3.

Figur 4: Hit record eksempel

1.3 Detektion af features

Med de fysiske modeller på plads var det nu muligt at diskutere, hvorledes de enkelte features skulle detekteres. De informationer vi havde til rådighed var distance vha. rotationssensoren og information af over hvilken farve lyssensorer lå. Ideen var at bruge hit records for at detektere gadeelementtyper.

1.3.1 Hit records

Vores hit record er en datastruktur som forbinder distancer med farveskift på sensorer. Ud fra disse records vil det så være muligt at konkludere type og orientering af et gadeelement. For at anskueliggøre metoden vises et eksempel for et T-kryds.

Hit record for T-kryds I dette eksempel kører robotten ind på et T-kryds fra den ene hovedgren, sådan at bigrenen er på højre side, se figur 4. Robotten vil nu køre lige ud. Når den højre sensor (rød) bevæges fra hvidt til sort, et farve skift, registreres et hit på højre side med den aktuelle distance. Det andet hit bliver registret på samme måde. Har robotten kørt en tilpas distance, udføres et venstre sving, i hvilket hit 3 og 4 optages. Ud fra de givne hit records kan nu typen og orientering bestemmes entydigt.

1.3.2 Invariant og justering

Vores detektionsmetode er meget afhængig af den præcise position af robotten samt den orientering når et hit record set skal optages. For at det lykkedes antages følgende invariant. Hver gang robot skal til at

Algorithm 1 Followline programstump

```
38     while(Running)
39     {
40         // Reading sensors
41         right = LightSenseRight();
42         left = LightSenseLeft();
43
44         // Too much right?
45         if(right == WhiteColor) PortA(
46             Float);
47         else PortA(OnPos);
48
49         // Too much left?
50         if(left == WhiteColor) PortC(
51             Float);
52         else PortC(OnPos);
53     }
```

udføre detektionsrutinen befinder sig robotten præcis i midten af et gadeelementes grene og orienteret parallelt med denne gren. Dette kræver at robotten skal rejsteres efter at den har kørt i et stykke tid. En sådan justering vil kunne blive foretaget på et T-kryds- og almindeligt krydsgadeelement. Ideen er at justere robotten indtil begge sensor ligger på en sort/hvid kant.

2 Followline test

Den første test, vi lavede for at teste robotens design, var en followline test. Princippet i followline er meget simpelt: robotten skal følge en streg ved brug af lys-sensorer. Koden er ligeså simpelt som selve followlineopgaven, når robotten kommer får langt til venstre bremses højre motor og vice versa. Algoritme 1 viser hovedloopet af followlineprogrammet. I linje 41 og 42 opdateres værdierne for right og left, som er af typen Color. Opdatering sker via kald til Color modulet, som aflæser råværdier af sensoren og konverter til en farve type på grundlagt af et threshold, se algoritme 3 linje 102 - 108. I linjerne 44 - 50 tændes eller slukkes output for port A og B, dvs. motorerne drejer eller hviler, at efter de opdaterede værdier for right og left.

Koden blev uploadet til RCX'en og robotten blev testkørt på en række af ligeud og sving gadeelementer. Det viste sig at followline koden som sådan virkede fint nok, men at robotten ikke kørte 100% ligeud, selvom den befandt sig indenfor vejen.

3 Nyt design

Problemet med at køre lige ud, som followlinetesten viste, som robotten havde krævede et nyt design. Hovedændringerne bestod i at vi ville tilføje en mekanisme som kunne lås og åbne differentialet for de to hjul og at vi ville lave lidt om på gadeelementerne.

3.1 Differentiale lås

Robotten kunne ikke køre 100% ligeud, hvilket skyldes at motorerne havde forskellig drejmoment, samt at friktionen i tandhjulene ikke var den samme for højre som for venstre hjul. Løsningen krævede en avanceret mekanisk differentialelås.

På figur 5 ses et 3d model af differentialelåsen. De to motorhovedakser forbindes med modsat drejningsretning i differentialet, hvilket har den effekt at selve differentialet nu kun vil dreje, når hovedakserne ikke kører med samme hastighed, eller sagt mere præcist: differentialets rotation giver udtryk for forskellen af hastigheden af det højre og venstre hjul. Sættes nu motoren, som er direkte forbundet til differentialet, til at blokere, tvinges det højre og venstre hjul til at dreje med eksakt samme hastighed. Derved kan robotten nu gå 100% lige, mens mulighed for at dreje stadig opretholdes. I starten var vi lidt bekymret om robotten havde kraft nok til at drive alle de tandhjul som vi satte på, men da vi havde gearet hjulene meget ned viste det sig ikke at være et problem. Faktisk var vores robot så kraftigt at den uden videre kunne skubbe en kontorstol!

3.2 Nye gader

Efter nærmere omtanke blev vi enig om, at det ville være bedre at have en form for skinner, som robotten kunne køre på i hver enkelt gadeelement. Skinner blev så realiseret ved at tilføje en hvid stribe på hvert enkelt gadeelement, som i midten blev brudt. Det ville nemlig skabe mulighed for at robotten kunne holde sig stabilt, dvs parallelt med gadeforløbet ved brug af followline programmet, mens brudet af striben kunne bruges i den nye detektionsmetode se sektion 3.2.1. Så denne metode var mere inspireret af princippet "sensing rather than planning". De nye gadeelementer er

Figur 5: Mekanisk Differentialelås

Figur 6: Nye gadeelementer med followline striber

afbildt i figur 6.

3.2.1 Ny detektionsmetode

Som følge af de nye gadeelementer og indførelsen af et followline program blev vi nødt til at genoverveje, hvordan robotten skulle detektere de enkelte gadeelementer. Vi nåede frem til at tre dedikerede lyssensor, som kun tog sig af denne opgave, ville ikke blot være en mulig løsning, men også den som ville kræve mindst ændring af den eksisterende kode og samtidigt sørge for at vi fik en logisk adskildelse af de to opgave / opførelser robotten havde, nemlig følg en streg og detekter et feature.

De tre sensorer skulle anbringes således, at de ville befinde sig over de mulige followlinestriber, når robotten befandt sig i midten af et gadeelement, dvs når de to sensor som blev brugt af followlineopførelsen

4.2 Ekstra lyssensor og tidsdrevne rutiner

I en de næste test fandt vi ud af at vores ide at udelukkende bruge rotationssensor til at navigere rundt i gaderne gjorde, at en upræcis drejning ville i værste fald blive mere upræcis som robotten kørte rundt i gadenettet. Vores oprindelige håb, om at followline programmet ville udligne de upræcise drej, var blevet knust. Igen skulle vi have brugt "sensing rather than planning" paradigmet. Valget faldt så på at droppe rotationssensorene til fordel for en ekstra lyssensor, som vi anbragt lige i midt foran followline sensorerne. Ideen var nu den, at vi i et sving ville dreje indtil denne sensor havde målt råværdier for farverne hvid sort hvid og så et lille stykke længere. Sagt med andre ord ønskede vi robotten at dreje væk fra den aktuelle followlinestribe indtil den var drejet ind på en ny followlinestribe og så lige et stykke længere så den atter befandt sig midt over striben. Da denne process vil være uafhængigt af foregående drejninger vil fænomenet at robotens orientering bliver værre med tiden forsvinde, hvilket også viste sig i efterfølgende tests. Da begge rotationssensorene blev fjernet benyttede vi i stedet for tid som mål for hvor langt robotten skulle dreje, køre ligeud etc. Det havde så den ulempe at roboten blev afhængig af batteriniveauet, men en tilpasning af rotationstider tog typisk ikke længere en 1 -2 minutter, og holdt så omkring 3 - 4 timer.

Den endelige robot kan ses på figur 9.

5 Programmelt

I denne sektion beskrives det program som udføres af køreenheden.

5.1 Programstruktur

Programmet består af en række moduler, som hver har deres ansvarsområder, og der eksekveres flere processer på RCXenheden for at opnå en bedre struktur og fleksibel kode. De moduler som ligger i køreenheden er

RobotRTS i en modificeret udgave, som har til opgave at switche mellem processer samt at kalde

Figur 9: Endelig robot

opdatering i sensor modulet, **SensorControl**.

SensorControl er ansvarlig for at opdatere interne variabler som distance og farverne målt af højre, venstre og den centrale lyssensor.

Color modulet tager sig af kalibration af lyssensorerne, samt at fortolke råværdi på lyssensorer som farver.

Navigation er hovedkomponenten af robotten. Det er her information fra de enkelte moduler samles og bruges til at udføre followline såvel som at navigere på features.

På figur 10 ses en skematisk illustration af hvorledes moduler er sat sammen.

5.1.1 SensorControl modul

Som nævnt forinden er det i **SensorControl** modulet variablerne fra sensing af omgivelserne bliver opdateret. I algoritme 2 ses opdateringsmetoden af **SensorControl**-modulet. I linjerne 27 - 30 aflæses de fire sensor ved kald til **Color** modulet, men de ny værdier bliver ikke sat med det samme, som de iagttages i linjerne 34 - 62. Derimod bliver en præcisionstæller talt ned med en hver kan en af aktuelle værdierne er

forskellig fra de globale værdier. Først når tælleren har en værdi på 0 assignes den aktuelle værdi. Hermed opnår vi en mere stabil aflæsning i områder hvor værdierne fluktuerer, dvs. i kantområder. Desuden holdes to andre variabler opdateret, EndOffRoad og OnRoad, som hhv. angiver om robotten befinder sig med begge sensorer på followlinestriben eller modsat, se linje 64 og 65.

5.1.2 Color modul

Color modulet er ansvarlig for mapping af råværdier til farveværdier, 0 for sort og 1 for hvid. I kalibrationsmetoden måles en serie af værdier per farve per sensor. Middelværdien af råværdierne for hvid og sort trækkes så fra hinanden deles med to og adderes med den mindste af de to middelværdier, se algoritme 3 linje 88 - 93. Denne ny værdi bliver så brugt som skillepunkt mellem sort og hvid, dvs. en råværdi som er større interpreteres som sort og eller som hvid, se linje 102 - 104.

5.1.3 Navigation modul

Navigationsmodulet er hovedmodulet, hvor al information samles og fortolkes. Det består af to processer, drive processen, som kører robotten, og en lcdprocess, der opdaterer lcddisplayet løbende med debug og anden nyttig information. Driveprocessen kan yderligere opdeles i tre, followline, kommunikation og selve navigationen mellem gade elementer.

Followlinekoden er stort set den samme, som i første test, men med nogle få ændringer, som skyldes at differentialet blive lås når robotten befinder sig indenfor followlinestriben, se linje 93 - 99 i algoritme 4. Når robotten har forladt followlinestriben, hvilket af geometriske årsager kun kan ske parallelt, bremses begge motorer og kommunikationsdelen overtager, se linje 112 - 115.

I kommunikationsdelen initieres kommunikationen af køroboten ved irtransmittering af et request. Der forventes så at den næste besked som modtages er en ny navigationsdirektiv. Koden er meget simpel og kan ses i algoritme 6.

Navigationskode består af en større switch, hvor der switches på den modtagne navigationskomman-

Figur 10: Programstruktur for køreenheden

Algorithm 2 SensorControl modul kode

```
void SensorControlUpdate( void )
{
27  currentLeft = LightSenseColor(1);
    currentMid = LightSenseColor(2);
    currentRight = LightSenseColor(3);
    // not using rotationsensors any more
32  //currentRotation = Port2Raw();

    if(currentLeft != left)
    {
37      leftprecision --;
        if (!leftprecision)
        {
            left = currentLeft;
            leftprecision = precision;
42      }
    }

    if(currentMid != mid)
    {
47      midprecision --;
        if (!midprecision)
        {
            mid = currentMid;
            midprecision = precision;
52      }
    }

    if(currentRight != right)
    {
57      rightprecision --;
        if (!rightprecision)
        {
            right = currentRight;
            rightprecision = precision;
62      }
    }

    EndOffRoad = right == BlackColor && left == BlackColor;
    OnRoad = right == WhiteColor && left == WhiteColor;

67  /* not using rotationsensors any more
        if(currentRotation != rotation)
        {
72      rotationprecision --;
            if (!rotationprecision)
            {
                rotation = currentRotation;
                distance++;
                rotationprecision = precision;
77      }
        }
    */
}
```

Algorithm 3 Color modul kode

```
void LightSenseCalibrate( void ){
76   Port1SetActive ();
   Port2SetActive ();
   Port3SetActive ();

   EstimateMinMaxValue( 1, & Color1Mid, 1);
   EstimateMinMaxValue( 3, & Color3Mid, 2);
81   EstimateMinMaxValue( 5, & Color5Mid, 3);
   EstimateMinMaxValue( 2, & Color2Mid, 1);
   EstimateMinMaxValue( 4, & Color4Mid, 2);
   EstimateMinMaxValue( 6, & Color6Mid, 3);

86   while( ! View ); while( View );

   Color1Threshold =
     Color1Mid < Color2Mid ? Color1Mid + (Color2Mid - Color1Mid) / 2 : Color2Mid + (Color1Mid - Color2Mid) / 2;
   Color2Threshold =
91   Color3Mid < Color4Mid ? Color3Mid + (Color4Mid - Color3Mid) / 2 : Color4Mid + (Color3Mid - Color4Mid) / 2;
   Color3Threshold =
     Color5Mid < Color6Mid ? Color5Mid + (Color6Mid - Color5Mid) / 2 : Color6Mid + (Color5Mid - Color6Mid) / 2;

96   lcd show int16 (Color1Threshold); lcd show digit (1); BusyPauseMS (1000);
   lcd show int16 (Color2Threshold); lcd show digit (2); BusyPauseMS (1000);
   lcd show int16 (Color3Threshold); lcd show digit (2); BusyPauseMS (1000);
}

101 Color LightSenseColor( int port )
{
  if( port == 1 && Port1Raw () > Color1Threshold) return BlackColor;
  if( port == 2 && Port2Raw () > Color2Threshold) return BlackColor;
106  if( port == 3 && Port3Raw () > Color3Threshold) return BlackColor;
  return WhiteColor;
}
```

Algorithm 4 Navigation modul - followlinekode

```
83   // Forward, Differential Free
   PortA (OnPos);
   PortB (Float);
   PortC (OnPos);

88   // Drive until some road is discovered
   while(EndOffRoad);

   // Follow Line
   while(!EndOffRoad)
93   {
     if(OnRoad)
     {
       // Forward, Defferential Locked
98       PortA (OnPos);
       PortB (Brake);
       PortC (OnPos);
     }
     else
     {
103       PortB (Float);

       if(left == BlackColor) PortC (Brake);
       else PortC (OnPos);

       if(right == BlackColor) PortA (Brake);
108       else PortA (OnPos);
     }
   }

113   // All Stop
   PortA (Float);
   PortB (Float);
   PortC (Float);
```

Algorithm 5 Navigation modul - kommunikationskode

```
121      // Request Command
      IRTransmit(0,ACKNOWLEDGED);
      recieve_remote_command:
126      while ((!IRReceive(&m1, &direction, &t)))
      {
          // debugging purposes
          MSCount = -1;
      }
```

do, direction, se algoritme 6. I tilfældet af et ligeud kommando, tændes begge motorer og differentialet låses, linje 130 - 133, for et tidsrum, som svarer til at robotten har nået centeret af followlinestriberne på et gadeelement, plus et tidsrum, der svarer til at køre ind på den foranliggende followlinestribe, linje 170.

Når direction er LEFT, dvs. robotten skal køre til venstre, tændes begge motoret og differentialet låses i ligeså lang tid som i en ligeud kommando, men når robotten har nået midten, dvs. det første tidsrum er over, skiftes strømmen på den venstre motor sammen med at differentialet åbnes. I denne tilstand forbliver robotten indtil der er gået en fast tid minturntime og den centrale farveværdi er hvid, linje 146, derpå ventes der et ekstra tidsrum leftturntime, linje 147. Den første tidværdi bevirker at vi mindst dreje 60 grader, dvs. den centrale lyssensor med høj sandsynlighed befinder sig på sort baggrund, mens tidsrummet leftturntime søger for at rette robotten op, således at den er parallelt med den venstre followlinestribe. Tilsidst låses differentialet og robotten kører ligesom under ligeud kommando ind på den venstre gade. Et højre sving er lige modsat og i et u-turn drejes robotten mindst 120 grader før den retter ind.

Efter dreje kommandoen er udført overtager followlinedelen igen.

Algorithm 6 Navigation modul - navigationskode

```
131 //Navigation algorithm
// Forward, Differential Locked
PortA (OnPos);
PortB (Brake);
PortC (OnPos);

136 MSCount = distance = 0;

switch(direction)
{
141 case LEFT:
while(MSCount < timetocenter);
MSCount = distance = 0;
PortB (Float);
PortA (OnNeg);
if (mid == WhiteColor) while (mid == WhiteColor);
146 while (mid == BlackColor || MSCount < minturtime);
MSCount = distance = 0;
while (MSCount < leftturtime);
MSCount = distance = 0;
PortB (Brake);
PortA (OnPos);
151 while (MSCount < timefromcentertoroad);
break;

case RIGHT:
while (MSCount < timetocenter);
156 MSCount = distance = 0;
PortB (Float);
PortC (OnNeg);
if (mid == WhiteColor) while (mid == WhiteColor);
161 while (mid == BlackColor || MSCount < minturtime);
MSCount = distance = 0;
while (MSCount < rightturtime);
MSCount = distance = 0;
PortB (Brake);
PortC (OnPos);
166 while (MSCount < timefromcentertoroad);
break;

case STRAIGHT:
while (MSCount < timetocenter + timefromcentertoroad);
171 break;

case UTURN:
PortB (Float);
PortA (OnNeg);
while (mid == BlackColor || MSCount < minturtime * 2);
176 MSCount = distance = 0;
while (MSCount < leftturtime);
PortA (OnPos);
break;

case NOTHING:
PortA (Brake);
181 PortB (Brake);
PortC (Brake);
while (!View);
goto recieve remote command;
break;

186 }
```