

# P0-PROJEKT

RoboCup



<b>Titel</b>	P0 – Projekt RoboCup
<b>Projektperiode</b>	2. September 2015 – 18. September 2015
<b>Semester Gruppe</b>	1.semester på ITC B306
<b>Projektgruppens deltagere</b>	Jeppe Øland Laurids Thormann Sebastian Damsgaard Jonas Alsen
<b>Vejleder</b>	Rasmus Løvenstein Olsen
<b>Sidetæl</b>	20
<b>Bilag</b>	2

### Synopsis

Denne P0-rapport indeholder en analyse af en bane, som en robot skal kunne gennemføre, samt en analyse af hvordan dette skal kunne lade sig gøre.

Problemformuleringen for projektet lyder:

”Hvordan kommer en robot bedst muligt igennem banen, med flest mulige point?”

I analyserne afklares der hvilke krav banen sætter, til robotens design og hvordan robotten skal programmeres. Der findes også ud af hvilke materialer af motorer der skal bruges og hvilken taktik der skal lægges for at fuldføre banen bedst muligt.

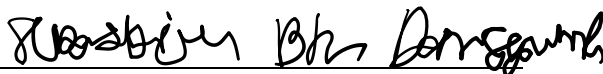
Banen har et pointsystem, som der ønskes udnyttet bedst muligt. Dette sætter taktikken at komme først i mål, som så gør at ikke alle ”forhindringer” skal laves. Heraf kommer der krav til robotens design. Ved tests bliver robotten lavet om flere gange, da der ses fordele ved dette.



Laurids Thormann



Jeppe Øland



Sebastian Damsgaard



Jonas Alsen

## Indholdsfortegnelse

Indledning .....	1
Problemformulering.....	1
Baggrund og analyse af banen .....	2
Starten.....	2
Første flaskeopsamling .....	2
Vippen .....	2
Fire linjer .....	2
Målskiven .....	2
Flaskeforhindring og bump .....	2
Mål .....	2
Prioritering af rute .....	3
Robotdesign .....	4
Robot version 1 .....	4
Robot version 2 .....	4
Robot version 3 .....	4
Delbeskrivelse .....	5
EV3 Brick .....	5
Gyro.....	5
Test 1.....	6
Motorer.....	6
Gribeanordning.....	7
Afstandssensor.....	7
Lyssensor.....	8
Linjen.....	8
EV3 farvesensor .....	8
Farvemåling.....	8
Reflekteret lys fra egen lyskilde .....	8
Reflekteret lys fra omgivelserne .....	9
Valg af målemetode.....	9
Test af Målingstilstande .....	10
lineFollow .....	11
Test 1.....	11
Test 2.....	11
Planlægning af kørsel .....	12
Fejlhåndtering .....	12

Program.....	13
Main task.....	13
Musik task .....	14
Farvekalibrering – manualCallibColor() .....	14
CheckIfLost(float lostTimer, bool direction) .....	14
calcDistMoved() .....	14
Mission1 (M1) .....	15
Mission 6 (M6) .....	15
Mission 9 (M9) .....	16
Mission 10 (M10) .....	16
Mission 11 (M11) .....	16
Program 12 (M12).....	17
Konklusion.....	18
Bilag.....	19
Litteraturliste .....	20

## Indledning

RoboCup-projektet går ud på at konstruere og programmere en robot, til at komme igennem en given bane. Banen har et pointsystem, som skalerer efter hvor hård hver enkelt forhindring er. Point bliver givet, for alt, fra at gennemføre en forhindring, spille musik, have et funktionelt og flot design, til at komme hurtigst i mål. Til dette projekt er der udleveret et LEGO Mindstorm sæt og et programmeringsprogram, kaldet RobotC. Inkluderet i LEGO Mindstorm sættet er CPU'en til robotten, samt motorer, sensorer og div LEGO byggeklodser. Målet for gruppen er at gennemføre banen og alt efter, hvilke forhindringer der viser sig mest lukrative mht. risiko/point, vil det blive forsøgt at klare disse. Dette gøres ved at analysere banen og det udleverede materiale grundigt, for herefter at lægge den bedst mulige taktik for at vinde RoboCup 2015!

## Problemformulering

**Hvordan kan en autonom LEGO-Mindstorm robot gennemføre en AAU-RoboCup med flest mulige point.**

En 'forhindrings'-bane står klar på Campus og den skal klares med flest mulige point på bundlinjen. Banen skal køres igennem, automatisk, af en LEGO-Mindstorm robot, som skal forprogrammeres i programmet RobotC.

Projektet munder ud i en opgave, som skal indeholde div. tanke-processer mhb. at løse opgaven så godt som muligt.

## Baggrund og analyse af banen

For at danne et overblik over banen, deles den ind i forskellige sektioner. Det er derfor muligt at fokusere på én sektion ad gangen, både med henhold til design af robotten, pointfordeling og selve softwaren. På baggrund af banes analyse kan der sammensættes en plan for, hvilke færdigheder robotten skal have. Der vil så kunne tages stilling til hvilken rute der bedst kan betale sig på baggrund af tid, point og sværhedsgrad.

### Starten

Her skal robotten kunne følge linjen, skifte over til en anden linje og tilbage igen.

### Første flaskeopsamling

Her er der placeret en flaske ned ad en side linje. Derfor skal robotten kunne navigere ned ad den side linje. Robotten skal herefter kunne samle flasken op og sætte den ned bag et markeret punkt. Det er også muligt at skubbe flasken, dog for færre point. Efter flaskeopgaven skal robotten køre tilbage til den første linje og fortsætte til næste del.

### Vippen

Her er der to muligheder robotten kan tage. Den ene mulighed går uden om den placerede vippe og her skal robotten kunne følge en linje, dog med et sving. Den anden mulighed går hen over en vippe. Vippen har en lille kant og robotten skal derfor være i stand til at kunne køre hen over den og få sig placeret midt på vippen. Vippen ville være den optimale rute pointmæssigt, men det er risikabelt da robotten kan vælte ned fra vippen.

### Fire linjer

Der er fire parallelle linjer, hvor det er den tredje af dem som fortsætter. Robotten skal derfor være i stand til at finde den rigtige linje og følge den.

### Målskiven

Her skal der placeres en flaske så tæt på midten af en målskive som muligt. Robotten skal derfor være i stand til at kunne navigere sig frem til flasken og samle den op eller skubbe den. Herefter skal robotten kunne placere flasken så tæt på midten af målskiven som muligt og så finde tilbage til linjen igen.

### Flaskeforhindring og bump

Her er der placeret to flasker og et bump. Robotten skal kunne identificere flasken og køre rundt om den. Herefter skal den kunne køre hen over et bump, eller udenom ved hjælp af to plader placeret i en 90 graders vinkel, her er der ikke nogen linje at følge. Efter det skal robotten identificere den anden flaske og køre den modsatte vej rundt om, end den gjorde ved den første.

### Mål

På mållinjen gælder det om at placere sig så tæt på midten som muligt. Robotten skal derfor være i stand til at finde midtpunktet og stoppe på det.



## Prioritering af rute

Ruten er blevet gennemgået og en plan af prioriteringer er blevet fremstillet.

Batteri-porte: Hver gennemført port giver 5 point og hvis man ser bort fra 'Vippen' og dens 2 porte, har banen i alt 12 porte, hvilket er 60 point. Da vi kommer igennem alle 12 porte, ved at følge banens grå linje og da første prioritering er at gennemføre banen, er det et mål at samle 60 point her.

Første forhindring – Brudt streg: Den brudte streg giver 10 point, samt 5 point for en batteri-port der er placeret midt på den brudte streg. Dette ses som en relativ nem forhindring og bør være 15 lette point at samle.

Anden forhindring – Flyt flaske: Her vælges først at skubbe flasken over strengen og herefter at lade robotten snøre om sin egen akse og smide flasken fra sig, så hårdt som muligt, da der gives 20 bonuspoint for at have den mest destruktive robot. Det ses som en fordel ikke at løfte flasken, hvis disse 20 bonuspoint skal opnås, da der kun er udleveret en løftemotor, hvilket indebærer at løft og opsamling skal ske i en bevægelse.

Tredje forhindring – Vippen: Vippen ignoreres, da det vurderes, at det ikke er risikoen værd at gå efter de 30 point, når den første prioritering er at gennemføre hele banen. Hvis robotten vælter giver det 20 minuspoint og chancen for at gennemføre reduceres væsentligt.

Fjerde forhindring – Parallelle streger: Denne forhindring er nødvendig at klare for at kunne gennemføre banen og 10 point samles her.

Femte forhindring – Flaske i målskive: Denne forhindring er blevet nedprioriteret, da det overordnede mål er at komme igennem banen, men vil blive kigget på såfremt det overordnede mål nås.

Sjette forhindring – Rundt om flaske: Denne forhindring ligger ligesom den brudte streg og de parallelle streger på ruten og 10 point samles.

Syvende forhindring – Bump/hjørne: Grundet robotens design vil det ikke være muligt at gennemføre bumpet. Hjørnet er den oplagte mulighed for 25 point.

Ottende forhindring – Rundt om flaske (modsat første flaske): Denne forhindring ligger ligesom den brudte streg og de parallelle streger på ruten og 10 point samles.

Niende forhindring – Landingsbanen: Prikken over i'et er at samle 50 point, hvilket bliver prioriteret højt, da banen her er gennemført og der ikke eksisterer nogen risiko for at misse eventuelle foranliggende point.

Ud over de ovenstående 9 forhindringer er der bonuspoint at skrabes sammen. Der gives op til 50 point for at spille musik på turen, hvilket der vil blive lagt kræfter i at opnå. Derudover kan 50 point samles for hurtigste robot gennem banen, hvilket ikke er et mål i sig selv. Dog vil der løbene blive optimeret på koden til robotens lineFollower, hvilket evt. kan resultere i en hurtig omgangstid.

## Robotdesign

Der har været mange overvejelser angående robotens design undervejs. Det har ledt til flere forskellige versioner. Her beskrives de forskellige versioner af robotten og forskellene mellem dem.

### Robot version 1

Version 1 af robotten er opbygget af en simpel ramme, der sidder rundt om EV3 Bricken. På siden sidder der to motorer, som hvert har et hjul monteret. Robotten er 22cm bred, 19cm lang og 8cm høj.

Da der er to motorer er det muligt at dreje robotten ved at køre hjulene i hver deres retning, eller lade det ene hjul køre hurtigere end det andet. Bag på robotten er monteret et kugleleje, der fungerer som det tredje hjul. Selvom robotten har 3 hjul er den stadig meget stabil, og den vil ikke være i stand til at vælte under normale forhold.

For at navigere robotten er der monteret en farvesensor foran, så robotten er i stand til at aflæse banens linje. Under EV3 Bricken er der monteret en gyro, så robotten er i stand til at måle sin rotation.

Placeringen af farvesensoren udgør et problem, da den ligger midt i omdrejningspunktet. Hvis lyssensoren placeres i robotens omdrejningspunkt, vil koden for `checkIfLost`<sup>1</sup> ikke fungerer efter hensigten, da lyssensorens relative position ikke vil ændres mærkbart. Dette gør at robotten har utrolig svært ved at lokalisere linjen, så hvis robotten kommer for langt væk fra linjen har den ikke mulighed for at finde tilbage igen. Det optimale er at robotten kan scanne dens omgivelser i en bred bue.

### Robot version 2

Version 2 af robotten er opbygget efter at skulle have et centreret omdrejningspunkt, for at farvesensoren kan scanne optimalt. Derfor er de to hjul placeret direkte på midten af robotten.

Et kugleleje er placeret foran, da det er der, der bliver lagt mest vægt på. Bagpå er der placeret en lille stang der skal forhindre robotten i at tippe bagover. Farvesensoren er stadig placeret foran, men på grund af den nye hjulplacering kan den scanne effektivt.

På grund af placeringen af hjulene er denne version effektiv til at dreje, og der skal derfor ikke bruges nær så meget kraft. Den kan derfor også scanne et større område, og kan derfor finde tilbage på linjen, hvis den kører forkert.

### Robot version 3

3. version er en direkte opgradering af version 2. Her er der placeret et kugleleje bagpå i stedet for et støtteben. Der er også placeret en anordning foran, som skal kunne gribe fat i en flaske. Denne anordning bruger den lille motor. Farvesensoren er placeret tættere på jorden, da test har afsløret problemer med at aflæse refleksioner optimalt, hvis den placeres for højt. Farvesensoren er placeret 1 cm over overfladen på den færdige robot.



Figure 1 Version 1

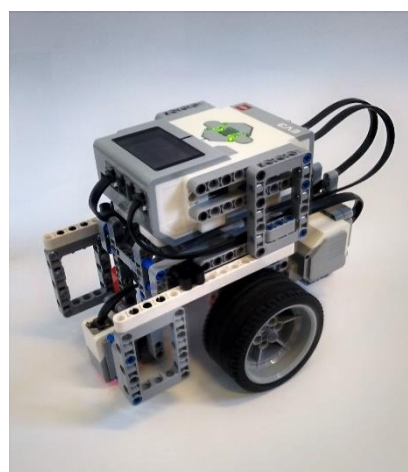


Figure 2 Version 2

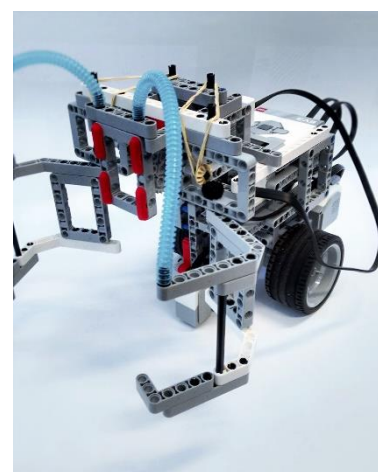


Figure 3 Version 3

<sup>1</sup> Beskrivelse af `checkIfLost` kan findes på side 14



## Delbeskrivelse

### EV3 Brick

EV3 Bricken er en programmerbar computer som kan bruges til en lang række forskellige opgaver. EV3 er den 3. generation af LEGO Mindstorm brick og er grundlaget for hele opgaven.

Koden til EV3 Bricken er i sproget C og den kan programmeres i en lang række forskellige programmer. Lige fra den basale software løsning produceret af LEGO til et mere avanceret program som RobotC. Til dette projekt koder vi i RobotC.

### EV3 Specifikationer

Operativsystem	Linux
Processor	300 MHz ARM9
Flash Hukommelse	16 MB
RAM	64 MB
Skærmopløsning	178x128 Sort og hvid
Tilslutningsmuligheder	USB 2.0, USB 1.1, Bluetooth
Porte	4 Motorporte (Output) (A,B,C,D) 4 Sensorporte (Input) (1,2,3,4)
Strøm	6 AA Batterier Alkaline eller Lithium Ion
Tilslutningskabler (?)	RJ12 – 6P6C (Telefonkabel)

### Gyro

Gyrosensoren måler robottens rotationsbevægelse og kan bruges til at orientere sig.

I mission 1 (M1), skal robotten skifte linje 2 gange. Her skal gyroen ved kontakt med en sort streg(missionsmarkør), skifte til X antal grader, således at robotten styrer mod den næste streg.

Ved kontakt med en sort streg skal robotten stoppe. Gyroen nulstilles, sådan at den står på 0 grader i forhold til den sorte streg. Herefter skal gyroen vide om den skal indstilles i minus eller plus grader, så robotten bevæger sig i den rigtige retning.

Der nulstilles og bestemmes retning afhængig af plus- eller minus grader:

```
void rotate(int degrees) {
    resetGyro(Gyro);

    if(degrees > 0) {
        setMotorSpeed(LeftMotor,10);
        setMotorSpeed(RightMotor,-10);
    }
    else {
        setMotorSpeed(LeftMotor,-10);
        setMotorSpeed(RightMotor,10);
    }

    while(abs(getGyroDegrees(Gyro)) <= abs(degrees)) {
    }
    setMotorSpeed(LeftMotor,0);
    setMotorSpeed(RightMotor,0);
}
```

Figure 4 Gyrokode

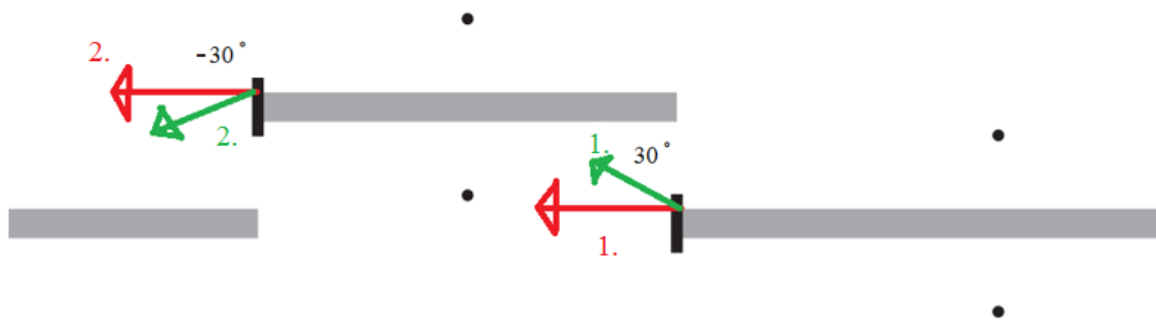


Figure 5 Illustration af M1 til M2

Ved første missionsmarkør (M1) stopper robotten og nulstiller gyroen. Fra denne position(rød pil1) roterer robotten 30 grader(grøn pil1), her ved robotten at den skal rotere til højre, da graderne er større end 0(figur 4, markeret med rød).

Anden missionsmarkør (M2). Gyroen nulstiller igen og der kommer en ny position(rød pil2). Fra denne position skal robotten rotere -30 grader(grøn pil2). Det kommer af koden, at når graderne er mindre end 0, roteres der mod uret(figur 4, markeret med grøn).

### Test 1

Første har vist mindre problemer. Det første problem var, at sensoren som skulle detekttere missionsmarkøren, ignorerede den. Sensoren er derfor sat længere ned mod overfladen, hvilket løste problemet. Næste problem var at hjulene roterede for hurtigt, hvilket gjorde at gyroen ikke kunne følge med. Motorkraften ved rotation blev sat ned og det hele virkede.

Ved testene blev der testet med forskellige vinkler. Robotten startede med at dreje 45 grader ved første linjeskift, hvilket gjorde at robotten ramte en batteriport. Herefter justeres vinklen til 30 grader, hvilket fungerer perfekt.

### Motorer

De to store stepmotorer er stærkere end den lille motor, men har derimod en lidt større responstid. De er beregnet til at køre og styre robotten. Disse motorer har en indbygget rotationssensor. De kan køre samtidig, med stor præcision hvilket gør at de nemt kan køre ligeud. Den lille motor er hurtigere og velegnet som gribeanordning.

	Lille motor	Stor Motor
RPM	240-250 rpm	160-170 rpm
Stå moment	8 Ncm	40 Ncm
Kører moment	12 Ncm	20 Ncm

## Gribeanordning

Som udgangspunkt skal robotten kunne dreje hurtigt omkring sin egen omdrejningsakse, hvilket betyder at gribeanordningsaggregatet skal agere så tæt på robotens centrum så muligt, af hensyn til den øgede centrifugalkraft, som denne kommer til at skabe. Dette har bl.a. betydet at gribeanordningen blokerer for en del af displayet på EV3 Bricken, hvilket er et kompromis.

Da gribeanordningen skal kunne kaste flasken fra sig med så stor kraft som muligt, er det vigtigt, at gribeanordningen spænder flasken stabilt fast, så flasken ikke ryster, samtidig med at den kan slippe flasken så hurtigt som muligt, hvilket betyder, at gribeanordningen ikke må holde for meget omkring flasken. Dette er, som vist på figur 6 og figur 7, blevet opnået ved at klemme flasken fra hver side med de sorte vinger og med så lille en anordning som muligt, sørge for at gribeanordningen ikke lader flasken undslippe.

## Afstandssensor

Afstandssensoren, som er en ultralydssensor, har til formål at opdage flaskerne og er blevet placeret bag ved gribeanordningen. Hvis den er placeret over gribeanordningen vil den være placeret for højt, til at kunne registrere flasken og hvis den er placeret foran gribeanordningen, vil der ikke være plads til, at gribeanordningen vil kunne holde om flasken.

Sensoren generer lydbølger, som den kan aflæse ekko af. Fra disse ekko bølger, kan afstanden til et objekt måles.

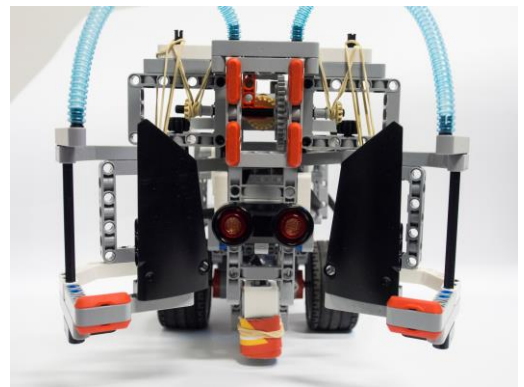


Figure 6 Billede af gribeanordning forfra

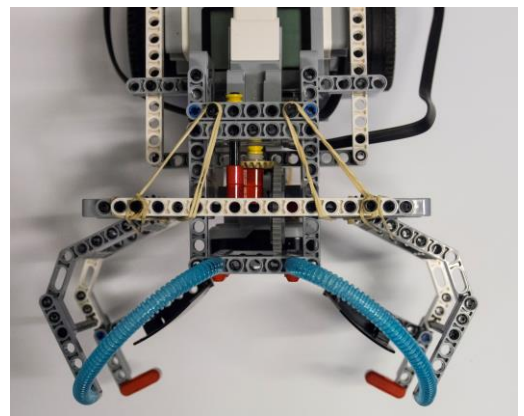


Figure 7 Billede af gribeanordning oppefra

## Lyssensor

### Linjen

Ruten robotten skal gennemføre, er indikeret med en grå streg på en hvid baggrund. Stregen er 5 cm bred og på udvalgte steder afbrudt, af et tværgående sort rektangel. Rektanglet er 10 cm x 2 cm og indikerer begyndelsen på en opgave, som robotten skal udføre. Linjen og missionsmarkørene er derfor visuelle fingerpeg og kan med fordel aflæses med en fotoelektrisk sensor, som den inkluderede farve sensor, i et LEGO mindstorms EV3 kit. Farvesensoren bruges til at skelne mellem de tre nuancer, som banen er indikeret med.

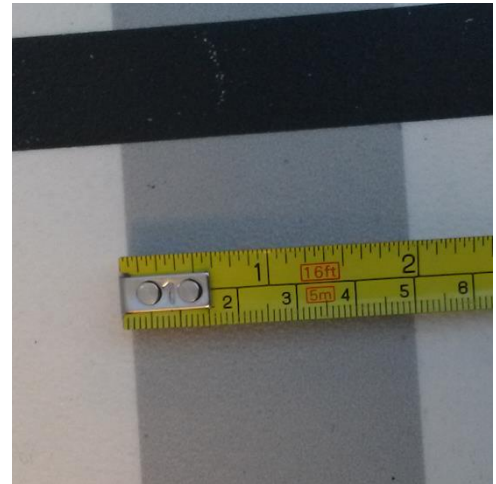


Figure 8 - Grå linje på hvid baggrund med sort missionmarkør.

### EV3 farvesensor

Farvesensormodulet består af en fotoelektrisk farvesensor og en RGB<sup>2</sup> lysdiode. Modulet har tre forskellige måletilstande.

#### Farvemåling

I denne tilstand belyser sensoren emnet, med hvidt lys fra sensorens indbyggede lyskilde. Derpå måles hvilke farver, der reflekteres tilbage til sensoren, som processerer måleresultaterne og klargør dem til afsendelse over I2C<sup>3</sup> protokollen.

I RobotC kan den målte farve præsenteres i to forskellige formater:

1. Som RGB værdi. I dette format vil hver af de tre farver repræsenteret, få en værdi mellem 0 og 255.
2. Som hue. I dette format bliver farven udtrykt ved en talværdi mellem 0 og 360.

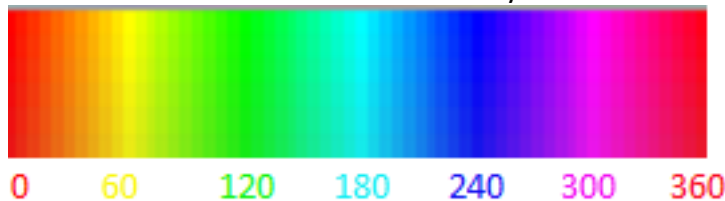


Figure 9 - Farvespektre som hue værdier

#### Reflekteret lys fra egen lyskilde

I denne tilstand belyser sensormodulet emnet med et rødt lys. Derpå måles hvor meget lys, der reflekteres tilbage fra emnet i det givne farvespektrum. Fordelen ved at belyse og måle med en grundfarve frem for f.eks. hvidt lys er, at målingen kan være mere resistent over for forstyrrelser. Den målte værdi er et tal mellem 0 og 100, hvor 0 er mindst refleksion og 100 er maks refleksion.

---

<sup>2</sup> Red Green Blue

<sup>3</sup> Inter-Integrated-Circuit

### Reflekteret lys fra omgivelserne

Den sidste tilstand måler sensormodulet, hvor meget af det omkringværende lys, der tilbagekastes fra emnet. Det vil sige at sensoren ikke leverer noget lys, fra sin egen lyskilde og er 100% afhængig af omkringværende lyskilder.

### Valg af målemetode

For at udvælge den bedst mulige målemetode, til at løse opgaven, er farvesensoren blevet testet på to forskellige overflader, med forskellige farver og nuancer. Den bedste metode er den, hvor de målte data klartest viser forskel på banens markeringer:

Gulvet i gangarealet uden for grupperum B306, har påtegnet en blå figur meget lig med den endelige bane, robotten skal køre på. Gulvet er gradieret sort og grå. Farverne på gulvet er meget anderledes, end de på den endelige bane og kontrasten mellem baggrund og figuren, der skal følges er ringe. Det sidstnævnte er særdeles interessant, da det sætter farvesensoren på hård prøve. Hvis den vil være i stand, til at skelne mellem kontrasterne vil den sandsynligvis have gode odds, for at skelne mellem den skarpere kontrast på den rigtige bane.



Figure 10 - Gulvet uden for grupperum B306

Den opstillede testbane udskrevet på A4 papirark. Denne banes farver og udformning er meget tæt på den endelige bane. Med andre ord er testbanen det tætteste på virkeligheden der kan kommes.

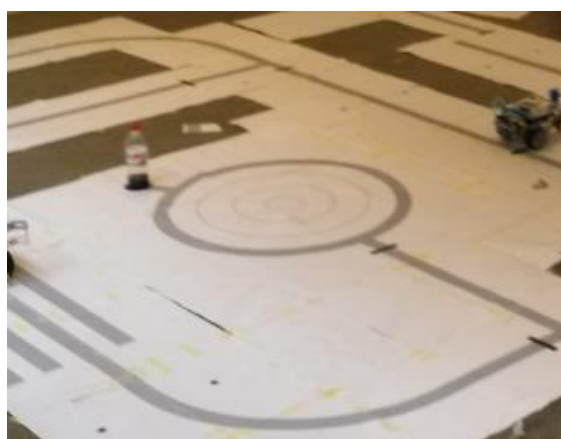


Figure 11 - Testbane af A4 papir



### Test af Målingstilstande

Farvemålingstilstanden er først testet på gulvet i gangarealet og derefter på testbanen. Sensoren holdes vinkelret på overfladen, i en afstand af 5mm over overfladen. Afstanden er valgt på baggrund, af den afstand LEGO foreslår i EV3-user guide<sup>4</sup> og fordi det er den korteste afstand, der passer med monteringshuller i LEGO klodserne. Farven udlæses som hue værdi.

De andre målemetoder testes med samme fremgangsmåde, dog udlæses værdierne, som mængden af reflekteret lys på en skala fra 0 til 100.

### Måleresultater

	Farverefleksion (hue)	Refleksion fra eget lys (0-100)	Refleksion fra omgivelser (0-100)
Gangareal Grå overflade	60	20	1
Gangareal Blå streg	190	10	1
Testbane Hvid overflade	62	85	7
Testbane Grå streg	64	48	4

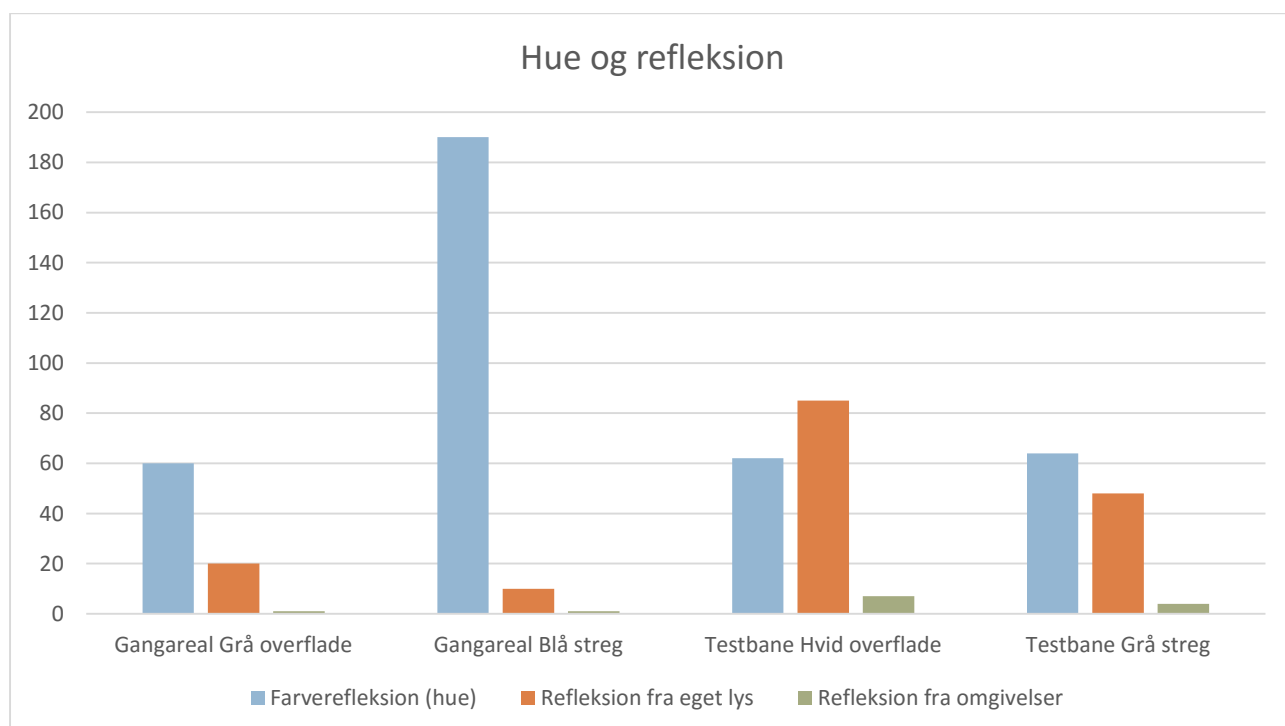


Figure 12 – Grafisk oversig over forskellige måledata

<sup>4</sup> Side 12 LEGO Mindstorms EV3 User guide:

### *Konklusion på resultater fra test af farvesensor*

Formålet af de foregående test, er at finde den sensor indstilling, som bedst egner sig til, at differentiere mellem banens hvide baggrund og de grå streger. Det vil sige at de målinger, som giver den største numeriske forskel mellem en streg og en baggrundsfarve, vil blive valgt som den bedste.

Farverefleksionen viser sig at være meget god, til at se forskel på gangarealets blå og grå farver. Dog er der meget ringe forskel på testbanens hvide og grå.

Refleksion fra eget lys giver en markant mindre forskel, på gangarealets blå og grå farver, men på testbanen ser resultatet meget bedre ud end farverefleksion.

Til sidst vises resultaterne for refleksion fra omgivelserne. Disse resultater viser sig at have så lille en forskel mellem linje og baggrund, at det ikke er nemt at differentiere mellem dem.

Baseret på resultaterne i foregående afsnit, besluttes det at bruge refleksion fra eget lys til den fremtidige konstruktion af robotten. Denne målemetode giver den bedste differentiering, mellem den grå streg og hvide bane. Desuden er metoden meget resistent over for udefrakommende lys, så længe lyset ikke er rødt.

## lineFollow

For at robotten skal kunne kende forskel på den grå linje, missionsmarkøren og den hvide baggrundsfarve, skal den kalibreres. For begge disse test bliver der kalibreret hver gang inden testen går i gang.

### Test 1

Robotten finder den grå linje og følger den højre kant. Robotten skifter selv retning, når den enten kommer ud i det hvide område eller ind på den grå streg. Den følger kanten ad den grå linje.

I denne test opstod der flere problemer. Det første var at hastigheden var for høj. Dette gjorde at responstiden mellem sensor og motor var for langsom. Sensoren kom for langt væk fra linjen, så den konstant søgte frem og tilbage. Dette resulterede i en oscillerende bevægelse, som medførte en nedsat fremgang.

### Test 2

Der blev testet på en lignende måde. Denne gang skulle farvesensoren, når den kom for langt fra linjen, måle sig ind til midten. Robotten roterer ud og finder venstre kant og roterer herefter tilbage og finder højre kant. Her måler den så rotationsværdien imellem de to kanter og halverer værdien. Herefter drejer robotten indtil at gyroen har opnået den værdi, den lige har fundet. Herefter skulle den køre ligeud indtil den kom ud til enden af kanten igen. Dette var sat i et loop og kørte rigtig fint. Robotten kørte hurtigere gennem banen med denne metode, men det viste sig at være mindre sikkert. Den mistede tit orientering, da robottens hjul ofte endte uden for strengen hvilket medførte konstant søgning efter linjens kanter. Samtidig havde den svært ved at backtracke. Så når den først mistede kontakt med linjen var den helt ude, hvorimod robotten i den anden test, sagtens kan finde tilbage og fortsætte korrekt. Det baner vejen for taktikken i 1. test. Hellere komme sikkert igennem løbet, end at komme hurtigt igennem med en stor chance for at fejle.

## Planlægning af kørsel

Den overordnede strategi er at få robotten hele vejen gennem banen uden kritiske fejl. Der er derfor lagt mest vægt på at få en pålidelig robot til de mere simple opgaver. Derfor er de mere teknisk krævende opgaver nedprioriteret, da det er her, der er størst risiko for fejl. Vippen er den første opgave som er valgt fra, da der er for stor risiko for at robotten kan vælte og falde ned. Da vippen er så tidligt på banen, vil det kunne ødelægge chancen for at gennemføre resten. Derefter er målskiven blevet valgt fra, da der er for stor chance, for at robotten kan køre forkert; enten ved at sidde fast i indersiden af cirklen, eller vælge den forkerte vej rundt.

For at danne et bedre overblik over banen henvises der til baneoversigten i billaget. Her er hver port, missionsmarkør og flaske nummereret, så det er nemmere at overskue.<sup>5</sup>

Strategien for at komme igennem banen bliver fordelt imellem de to missionsmarkører (M1-M12.)

M1-M2	Robotten skal finde M1, dreje med uret og derefter kører fremad indtil den lokaliserer næste linje. Denne passeres indtil modsatte kant er registreret. Herfra skal den følge linjen frem til M2.
M2-M3	Når M2 er identificeret drejer robotten mod uret og kører fremad indtil den rammer den originale linje. Herfra følger den linjen hen til M3.
M3-M4	Her fortsætter robotten ad den originale linje frem til M4. Opsamling af F1 er nedprioriteret.
M4-M6	Her skal robotten følge svinget rundt og ende ved M6. Vippen (M5) er blevet fravalgt da den udgør for stor en risiko.
M6-M7	Når M6 er identificeret skal robotten dreje mod uret. Herefter skal robotten lokalisere den tredje linje og fortsætte frem til M7.
M7-M9	Her springes M8 over på grund af tidsbegrænsninger med henhold til kodning. Robotten kører derfor ligeud til M9.
M9-M10	Når M9 er identificeret, drejer robotten med uret. Herefter bruges encoderne i motorerne til at måle en afstand. Når den ønskede afstand er målt drejer robotten mod uret og finder tilbage til linjen. Herefter tager den svinget rundt og når til M10.
M10-M11	Samme fremgangsmåde som i M9-M10, bortset fra at der her er et delay fra M10, da det ikke er muligt at dreje med det samme pga. Robottens størrelse.
M11-M12	Samme fremgangsmåde som i M9-M10, med undtagelse af at robotten her starter med at dreje mod uret.
M12	Når M12 er identificeret, kører robotten en given afstand som svarer til midten af pladen. Herefter stopper den og banen er gennemført.

## Fejlhåndtering

Der er lavet en enkelt procedure til fejlhåndtering. Når robotten ikke kan lokalisere den ønskede farveforskel, begynder den at dreje rundt om sig selv i samme retning, som den havde før den mistede fokus. Når den så finder den ønskede farveforskel fortsætter den ad linjen.

---

<sup>5</sup>Nummereret kort. Se Bilag (side 19)

## Program

Programmet består af to tasks: main task og musik task.

Derudover er lavet adskillige hjælpefunktioner der primært kaldes fra main task.

### Main task

I main task ligger den basale navigations kode, der gør robotten i stand til at følge en linje og detektere missionsmarkører. Hele navigationskoden ligger i et uendeligt loop, der kaldes efter musik task'en er startet og farvesensoren er blevet kalibreret. Når navigationskoden begynder forventer programmet at robotten er placeret max 7,5 cm til højre for linjen, da dette er robottens rotationsradius.

Selve navigationsloopet er opdelt i 5 faser:

1. Initialisering af reflektionsmåling og nulstilling af timer1
2. Gå ind i loop der kører så længe reflektionsværdien er højere end skilleværdien. I loopet sættes de to hovedmotorer til at køre fremad med asynkron hastighed, hvor den højre motor kører hurtigst. Dette får robotten til at køre fremad samtidigt med at den drejer til venstre. Den egentlige effekt af loopet er at robotten vil køre fremad imens den langsomt søger ind mod stregen. Hver gang loopet kører læses der også fra farvesensoren om reflektionen ændrer sig. Det sidste der bliver gjort inden loopet gentages er at kalde checkIfLost() funktionen. Når loopet afsluttes på grund af for lav reflektionsværdi nulstilles timer1 før næste loop påbegyndes.



Figure 13 - Illustration af anbefalede robotplacering.

3. Dette loop køres så længe robotten er cirka midt på linjen. Det vil sige at så længe reflektionsværdien er  $\pm 10\%$  af skilleværdien kører robotten lige fremad. Her bruges checkIfLost funktionen ikke. Når dette loop brydes bliver timer1 igen nulstillet før det næste loop begynder.
4. Når reflektionen er mindre end skilleværdien, men højere end stopLine værdien vil robotten antage at sensoren er over den grå streg. I det tilfælde vil den køre fremad mens den drejer langsomt til højre. Ellers fungerer denne fase ligesom fase 2.
5. Den sidste fase er at tjekke efter om robottens farvesensor er over en sort missionsmarkering. Hvis dette er tilfældet vil robotten lægge én til stopLineCounts og derefter bruge denne globale variabel til at vurdere hvilken mission robotten er nået til.

### Musik task

Musik task'en består af et uendeligt loop der står for at spille musik så snart musik task'en er startet. Musik task'en vil dynamisk ændre musiknummer alt efter robottens status. Hvis robotten er i en fejltilstand, spilles der en mere dystre lyd, hvorimod der spilles en glad melodi mens robotten kører som den skal.

### Farvekalibrering – `manualCallibColor()`

For at sikre, at robotten er i stand til at navigere i forskellige lysforhold skal farvesensoren kalibreres ved hver opstart. Når kalibreringsfunktionen kaldes, bliver der skrevet simple hjælpeinstruktioner på EV3 skærmen. Først skal robotten placeres så farvesensoren kan se den hvide baggrunds overflade. Den målte refleksionsværdi vises på skærmen. Når værdien er stabil, trykkes der på enter knappen og programmet vil måle gennemsnitsværdien af overfladen over 10 samples, ved at kalde `avgReflectedLight()` funktionen. Når værdien er gemt, gentages det samme for den grå strege og de sorte missionsmarkører.

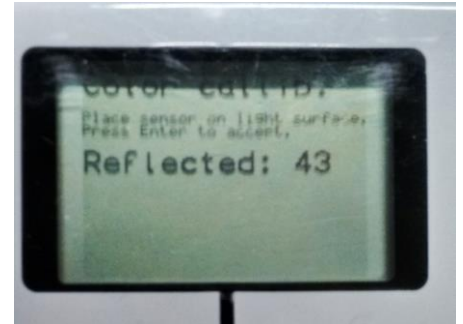


Figure 14 Kalibrerings skærm

Når alle tre nuancer er blevet opmålt, udregner programmet gennemsnitsværdien af den lyse værdi og den grå værdi. Denne værdi gemmes som en skilleværdi, der bestemmer om farvesensoren er over strege eller ej.

Værdien for den sorte missionsmarkør gemmes i en global variabel. Denne vil senere blive brugt til at bedømme hvornår sensoren er over en missionsmarkør.

### `CheckIfLost(float lostTimer, bool direction)`

Denne funktion kontrollerer om robotten har set enten strege eller baggrunden inden for de seneste 1500 ms. Hvis det ikke er tilfældet vil robotten stoppe med at køre fremad og begynde en stationær søgning efter strege eller baggrunden. I søgningstilstand vil robotten dreje rundt på stedet og fortsætte i den omdrejningsretning den kørte før de 1500 ms er udløbet.

### `calcDistMoved()`

I flere af missionerne er der brug for at vide hvor langt robotten har bevæget sig, derfor er der lavet en funktion, der kan udregne denne distance ved følgende formel:

$$\frac{\pi \cdot \text{hjul diameter}}{\text{encodePrRunde}} \cdot \text{nuværendeEncCount}$$



### Mission1 (M1)

Efter inspektion af banen, findes det fordelagtigt at få robotten til at dreje 30 grader med uret for at ramme den brudte streg og ligeledes 30 grader mod uret for at komme tilbage på linjen.

```
if(reflection <= stopLine){
    stopLineCounts++;
    switch(stopLineCounts) {
        case 1:
            mission1(30,1000);
            break;
        case 2:
            mission1(-30,1000);
            break;
    }
}
```

Når robottens farvesensor opfanger, at den mængde lys der bliver reflekteret er mindre end eller lig med stopLine variabelen, vil mission1 blive initialiseret.

### Mission 6 (M6)

Mission6 kan løses nemt ved at indse at opgaven består af det samme som mission1 gentaget to gange, hvilket betyder at koden fra mission1 blot kan genbruges. Dette er en stor fordel da der ikke skal skrives ekstra kode, og fordi den eksisterende kode er gennemtestet.

Når mission6 kaldes fra missionsvælgeren kalder mission6, mission1 to gange i træk. Først drejer robotten 20 grader mod uret, og kører fremad i 1000 ms før den begynder at søge efter den grå streg. Når denne er fundet vil den dreje 10 grader med uret og køre fremad i 500 ms, for at ignorere den streg som sensoren starter over, indtil robotten kan se den tredje grå streg. Når den tredje streg er fundet vil robotten atter påbegynde lineFollow.

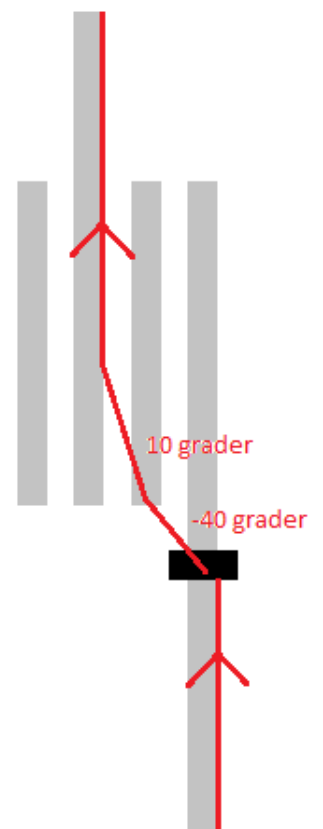


Figure 15 Mission 6

### Mission 9 (M9)

Mission 9 kan, ligesom mission 6 også genbruge en del funktioner fra tidligere og har ikke krævet noget nyt kode. Når mission 9 startes af missionsvælgeren drejer robotten 45 grader med uret, hvorefter den kører 40 cm med 15 % motor kraft. Dette gør at den kører uden om flasken med en betydelig sikkerhedsmagen. Herefter kaldes mission 2 koden som får robotten til at dreje 45 grader mod uret og derefter køre fremad mens den leder efter den grå streg. Når robotten har fundet den grå streg, fortsætter den med at følge den indtil næste missionsmarkør.

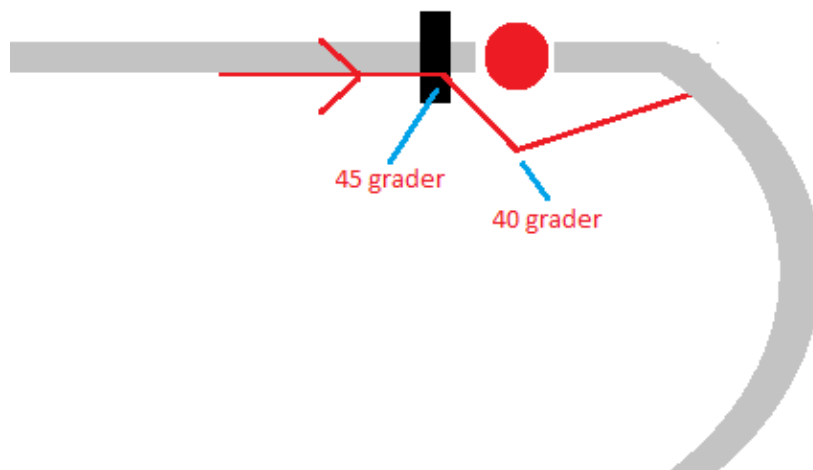


Figure 16 Mission 9

### Mission 10 (M10)

Mission 10 kaldes af missionsvælgeren. Robotten vil forsøge at undgå blokeringen og i stedet navigere gennem den rette vinkel.

Denne mission er mere kompleks end de andre og består derfor af en række af flere handlinger:

1. Robotten bevæger sig 40 cm ved 15% hastighed.
2. Derefter drejer den 30 grader mod uret.
3. Robotten bevæger sig nu mod vinkelvæggen og måler afstanden med den ultrasoniske sensor. Robotten stopper når afstanden er under 5 cm.
4. Derpå drejer robotten 55 grader med uret og kører 50 cm.
5. Det sidste step er at robotten aktiverer mission2 koden og drejer 65 grader mod uret. Derpå kører den fremad indtil den finder den grå streg.

Koden for de resterende missioner er endnu ikke udarbejdet, men her følger idéerne for at gennemføre M11 og M12.

### Mission 11 (M11)

Mission 11 klares ved at dreje robotten 45 grader mod uret og køre 40 cm fremad ved 15 % speed. Derpå kaldes mission1 koden med 45 grader med uret og robotten vil køre over stregen og søge den højre kant. Derpå vil robotten begynde at følge stregen indtil næste missionsmarkør.

### Program 12 (M12)

Når program 12 bliver aktiveret bevæger robotten sig langsomt fremad mens den tæller afstanden med moveDist funktionen. Robotten skal stoppe når den har tilbagelagt en afstand svarende til halvdelen af "landingsbanens" afstand. Landingsbanen er målt til at være 300 cm, hvilket betyder at robotten skal stoppe efter 150 cm.

## Konklusion

Robotten er blevet testet, omkodet og testet igen. Problemer undervejs, især med lyssensoren, er blevet forsøgt løst og robotten styrer nu, primært ved hjælp af sin lineFollower, igennem banen. På trods af, at lyssensoren blev testet, da den befandt sig 5 mm over overfladen, er den nu placeret ca. 1 cm over overfladen. Dette er hovedsageligt pga. testbanens forringede kvalitet, hvilket har medført aflæsningsproblemer.

Aflæsningsproblemerne var primært forårsaget af snavs på testbanen, samt buler og ujævnheder på testbanen.

Grundet tidspres er koden til de sidste to missionsmarkører ikke blevet konstrueret, men princippet for, hvordan robotten kan gennemføre disse, er blevet forsøgt beskrevet i rapporten.

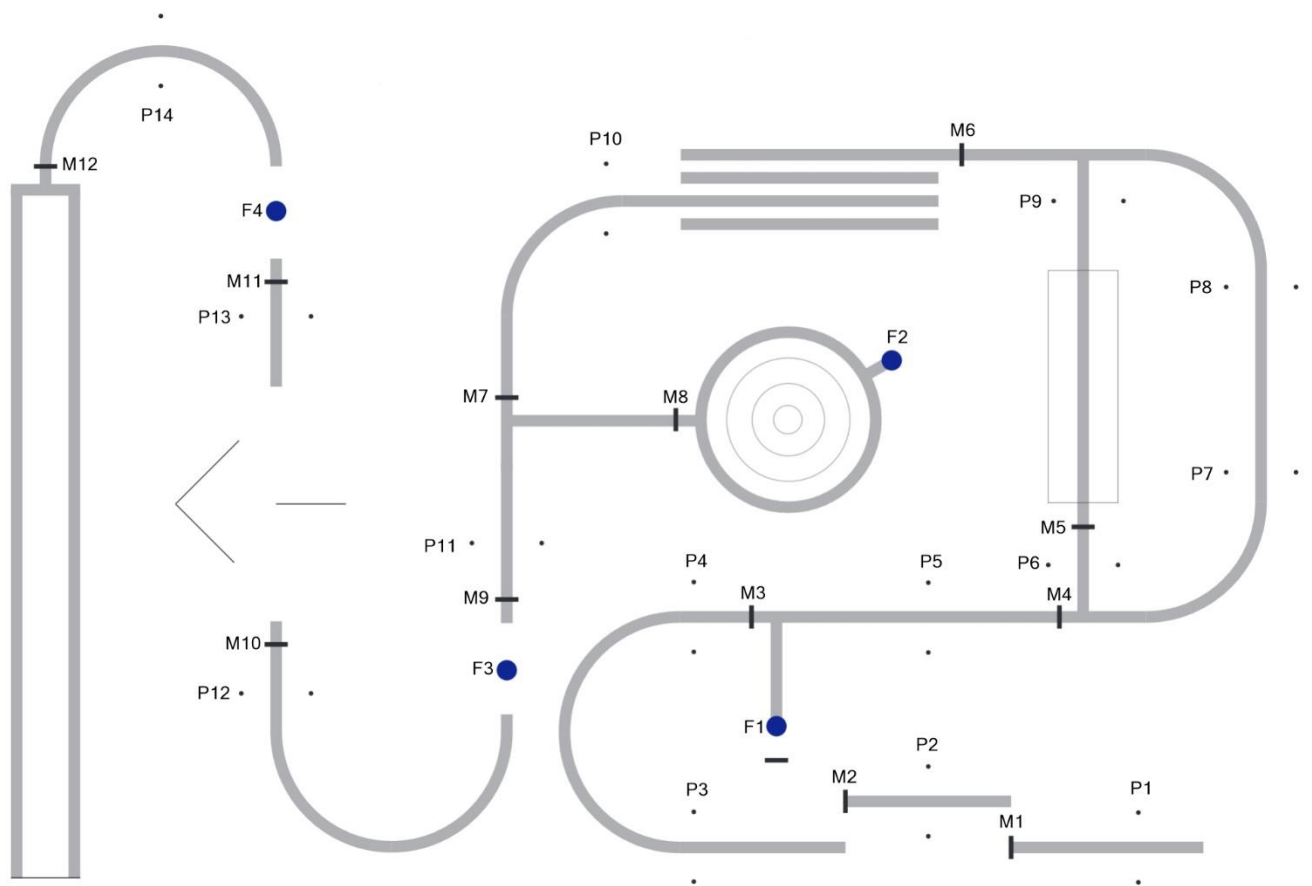
Robotten kan i skrivende stund samle ca. 100 sikre point på ruten og derudover kommer bonuspoint for musik, kreativitet og design, samt video.

Optimering af koden, samt integrering af gribeanordningen vil blive arbejdet yderligere på frem mod RoboCup 2015 og det forventes at robotten vil kunne kaste den første flaske, samt få maksimum point, for sidste forhindring.

Mange valg er blevet truffet og resultatet er en original robot, der trods sine mangler mht. de mere komplekse opgaver, kører stabilt igennem banen med mange muligheder for tilføjelser i koden, som vil gøre den mere konkurrencedygtig til RoboCup 2015.

## Bilag

### Kort over bane



### lineFollower

Koden til lineFollower er uploadet som bilag, sammen med rapporten.



## Litteraturliste

LEGO, 2013, LEGO Mindstorms EV3 User Guide

<http://cache.lego.com/r/www/r/mindstorms/-/media/franchises/mindstorms%202014/downloads/user%20guides/user%20guide%20lego%20mindstorms%20ev3%2010%20all%20enus.pdf?l.r2=-695542490>

[18/09/2015]

LEGO, 2013, LEGO Mindstorms EV3 User Guide (en anderledes version)

<http://www.nr.edu/csc200/labs-ev3/ev3-user-guide-EN.pdf>

[18/09/2015]

RobotNav, Robotnav.com

<http://www.robotnav.com/>

[18/09/2015]