The Juggernaut

Navn på gruppemedlemmerne:

Michael Hansen, Emil Jansen og Dennis Kjærgaard

Video af testkørsel:

Vi har valgt at lave to videoer, da vi startede med at lave robotten til otte batterier. Dette blev forhindret grundet mangel på batterier, som resulterede i at vi måtte skifte batteri type og antal.

Video 1 (Bilag1) viser den udgave vi kørte konkurrence med (seks batterier). Video 2 (Bilag 2) viser vores egentlige tænkte robot (otte batterier).

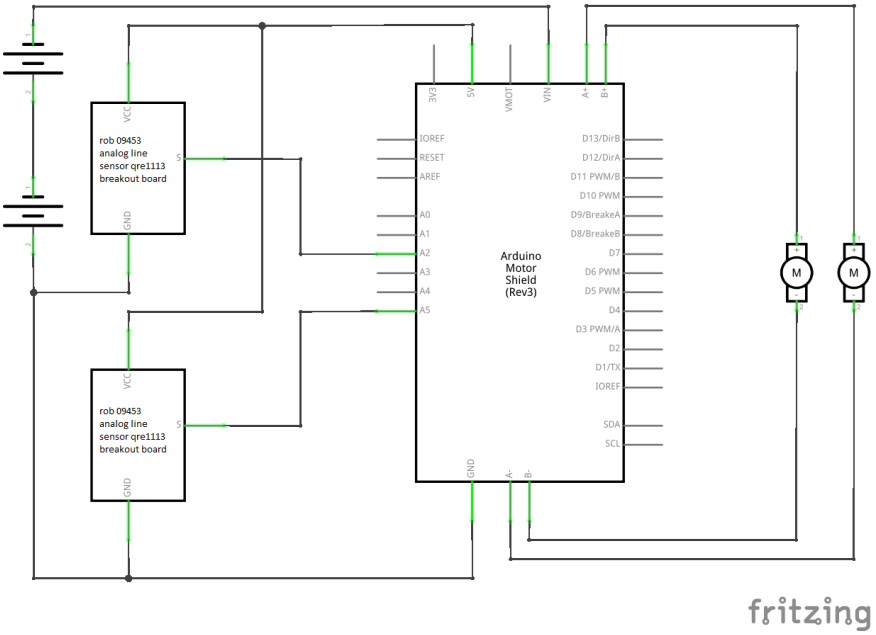
Se Bilag 1 og Bilag 2.

Bedste konkurrencetid:

45.something sekunder

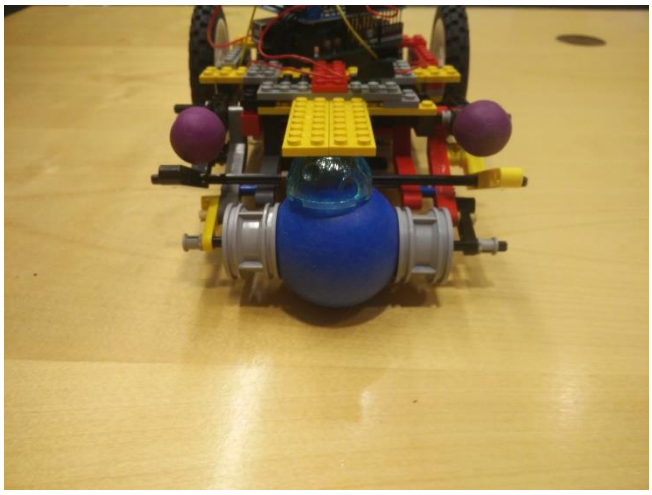
Opbygning af hardware:

Der har ikke været nogle store komplikationer i forhold til hardware opbygningen. Da vi kun har de 2 linje-sensorer på, og de er lavet til et standard 5-volts system. Dog har der været en del små komplikationer i forhold til at ledningerne ikke ville blive i arduinoen, og derved tit skulle sættes på plads. Derudover mødte vi et problem på konkurrence dagen, da der ikke var flere 1.2 Volts batterier tilbage, blev vi nødt til at skifte til 1.5 Volts batterier, og for ikke at ødelægge systemet skulle vi skifte ned til 6 batterier i stedet for 8. Hvilket gav et skift i spænding fra 9.6 Volt til 9 Volt hvilket er et fald på 6.25%. Dette gjorde at vores system i samspil med softwaren skulle optimeres en del, og det formåede vi ikke at gøre på de indlagte 45 minutter, da vi både skulle skifte batterikasse og teste, samt at motorerne ikke fik samme kraft som tidligere. Hvilket betød at robotten ikke fik præsteret som forventet til konkurrencen.

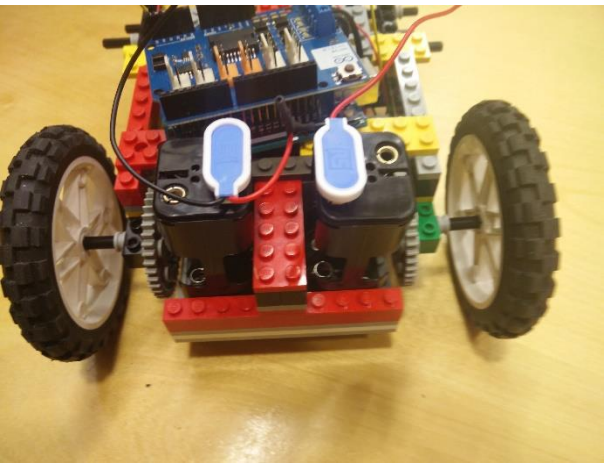


Opbygning af robot i Lego:

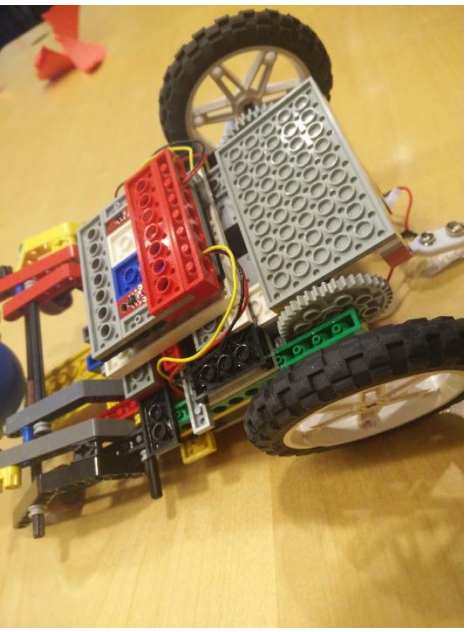
Vi har valgt at bruge to baghjul, der sidder rimeligt tæt sammen, og størst muligt. For at skabe størst mulig afstand pr. omdrejning fra motoren. Grunden til at afstanden mellem motorerne er så lille, er for at skabe god stabilitet.



Foran på robotten har vi valgt at lave et kugleleje bestående af to hjulkapsler, til at holde en plastik bold på plads i siderne, og en 'kuppel' øverst. Kuplens funktion er at sørge for at vægten af robotten, ikke ville resultere i at bolden ville falde ud. Denne opsætning er finjusteret så bolden kan køre rundt men samtidig holdes på plads.



Batterierne har vi valgt at placere bagerst og tættest på hjulene, for at holde vægten tættest på baghjulene og derved skabe bedre greb på banden. Placeringen af batterierne er placeret så vidt muligt samlet ved midten, igen for at skabe bedre stabilitet.



De to linje sensorer er placeret nederst på robotten, bag midten tæt på hjulene. Dette er gjort for at lettere kunne justere robotten når denne skal dreje, så der kommer mindst mulige sving ved hver læsning af input fra linje sensorerne.

Opbygning af program:

Programmet er opbygget vha. en masse konstanter og funktioner som styrer robottens køre egenskaber. Konstanterne styrer forskellige ting, som input af motorerne og sensorerne, samt at de styrer kraften til motorerne og finfølingen af grænsen mellem sort og hvid.
Funktionerne er defineret som Stop, Lige ud, venstre drejning og højre drejning.
Derudover er den sidste del hardcoded ind så robotten følger et bestemt mønster, og ikke detekterer noget.

```
const int
PWM_A   = 3, DIR_A   = 12, BRAKE_A = 9,
PWM_B   = 11, BRAKE_B = 8, DIR_B   = 13,
ldrPin1 = A2, ldrPin2 = A5,
forskel = 300, spd = 255, x = 50;
int
rate1, rate2;
void setup() {
  pinMode(DIR_A, OUTPUT);    // Direction pin on channel A
  pinMode(BRAKE_A, OUTPUT);  // Brake pin on channel A
  pinMode(DIR_B, OUTPUT);    // Direction pin on channel A
  pinMode(BRAKE_B, OUTPUT);  // Brake pin on channel A

  void LigeUd(){
    digitalWrite(DIR_A, HIGH); // setting direction to LOW the motor will spin backward
    digitalWrite(BRAKE_A, LOW);
    analogWrite(PWM_A, spd-x); //Set the speed of the motor, 255 is the maximum value
    digitalWrite(DIR_B, HIGH);
    digitalWrite(BRAKE_B, LOW);
    analogWrite(PWM_B, spd);
  }
```

```
  void DrejVenstre(){
    digitalWrite(DIR_A, HIGH);
    digitalWrite(BRAKE_A, LOW);
    analogWrite(PWM_A, (spd));
    digitalWrite(DIR_B, LOW);
    digitalWrite(BRAKE_B, HIGH);
    analogWrite(PWM_B, 0);
  }
  void DrejHojre(){
    digitalWrite(DIR_A, LOW);
    digitalWrite(BRAKE_A, LOW);
    analogWrite(PWM_A, x);
    digitalWrite(DIR_B, HIGH);
    digitalWrite(BRAKE_B, LOW);
    analogWrite(PWM_B, (spd));
  }
```

```
  void Stop(){
    digitalWrite(DIR_A, HIGH);
    digitalWrite(BRAKE_A, LOW);
    analogWrite(PWM_A, 0);
    digitalWrite(DIR_B, HIGH);
    digitalWrite(BRAKE_B, LOW);
    analogWrite(PWM_B, 0);
  }
}
```

```
void loop() {
  rate1 = analogRead(ldrPin1);
  rate2 = analogRead(ldrPin2);
    if( abs(rate1 - rate2) < forskel){
      LigeUd();
    }
    else if(rate2 > (rate1 + forskel) ){
      DrejHojre();
      delay(150);
    }
    else if(rate1 > (rate2 + forskel)){
      DrejVenstre();
      delay(150);
    }
  else{
  }
```

```
  if(rate1> 800 && rate2 > 800){
    Stop();
    delay(100);
    LigeUd();
    delay(600);
    DrejVenstre();
    delay(1200);
    LigeUd();
    delay(2900);
    DrejHojre();
    delay(1375);
    LigeUd();
    delay(1150);
    DrejHojre();
    delay(1375);
    LigeUd();
    delay(5000);
    Stop();
    delay(10000);
  }
}
```

Der bliver brugt rates til at tjekke om det er hvidt eller sort. Hvor forskellen i værdien på en rate går fra ca. 200(hvid) til ca. 900(sort). Programmet tjekker efter om den rammer en sort linje, og derefter kører den så den hardcodede del. Da den ene motor er lidt svagere end den anden, bruger vi en x-værdi til at udligne det. Vi valgte at hardcode den sidste del da vi var under tidspres, samt at de grupper der brugte afstandssensorer fortalte at det krævede en del ekstra arbejde. Dette udgjorde at valget faldt som det gjorde. Robotten virker da også, men vil nok få problemer hvis banen bliver ændret en smule, eller hvis motorkraften falder eller stiger.

Hvordan er samspillet mellem mekanik, elektronik og software?:

Med mekanik menes der hjul, gear, kuglelejer og selve opbygningen af bilen. Med elektronik menes der, batterier, arduino og sensorer. Med software menes der arduino programmet. Samspillet mellem disse er ret simpel og opstilles bedst som en step-by-step liste.

1) Så snart der er strøm på systemet, starter softwaren.
2) Softwaren modtager analog signal fra sensorerne under bilen.
3) Alt efter hvilket signal der bliver modtaget, sætter softwaren motorerne igang. Hvis begge sensorer viser hvidt, kører bilen ligeud. Hvis én af sensorerne viser sort, retter bilen sig selv ind på sporet.
4) Motorerne får gearingen, og derved også hjulene, til at køre rundt.
5) Bilens nye position giver nyt signal til softwaren.
6) Steps 2 - 5 bliver gentaget indtil begge sensorer står over sort tape, hvorefter koden kører en 'hardcoded' del, som styrer robotten uden om vægen.

Dette konkluderer samspillet mellem de mekaniske og elektroniske dele og softwaren.

Placeringen af sensorerne er også velovervejet. Den step-by-step liste virker kun fordi alle delene er placeret hvor de er placeret, med den gearing der er på bilen. Hvis sensorerne kommer for langt frem på robotten kan de ikke nå at registrere linjen under robotten når den drejer, fordi omdrejningspunktet på robotten ligger på det hjul der bremser. Jo længere væk fra omdrejningspunktet sensorerne kommer, jo hurtigere flytter de sig.
Dette sammenspil mellem mekanik og elektronik/software er meget interessant, og var noget vi overså i de første udkast af robotten. Da vi satte sensorerne længere tilbage på robotten blev det lige pludselig lettere at styre den rundt i svingene, og derved øge hastigheden.

En anden ting vi har overvejet en stor del er hvordan robotten skulle dreje når en af sensorerne er over den sorte streg. Vi prøvede nogle forskellige metoder, men endte ud med at stoppe den ene motor, og lade den anden køre. Vi forsøgte også at skrue ned for hastigheden og endda køre bagud, men begge metoder viste sig at være for upræcise og blev erstattet af den nuværende. Der var dog nogle komplikationer på vejen, da en af vores motorer kører hurtigere end den anden. Dette er der også taget hensyn til i koden, så robotten kan tage svingene som den skal og køre ligeud uden at halte til den ene side.

En tredje ting vi blev opmærksomme på i de indledende faser af portføjlen er, at stabilitet er vigtigt robotten. Samspillet mellem mekanik, elektronik og software er meget delikat, og derfor gør stabilitet en verden til forskel når der skal testes, bygges og finjusteres. Vi havde problemer med at sensorerne rykkede sig en lille smule for hver gang vi samlede robotten op, og i sidste ende gjorde det at robotten ikke kunne bruge de konstanter vi havde sat i koden.

Konklusion:

Man kan konkludere, at samarbejdet mellem software og hardware kan være meget tidskrævende og svært at fejlfinde på. Dette konkluderes ud fra de utallige finjusteringer der har skulle til for at opnå et godt resultat, og at det ikke er alle fejl mellem disse der kan forklares.

Her kan også konkluderes, problemstillingen i at skulle fremstille en autonom robot der skulle kunne gennemfører en pre-opstillet bane. Er gennem ført ved at konstruere en Lego robot bestående af to hjul, et hjemmelavet kugleleje og to linjesensorer, styret af en arduino. Dette har medført en god forståelse af autonome robotter og kredsløb.

Perspektivering og forbedringer:

Vi kan se ud fra test at de otte batterier der giver 9.6V giver hurtigere resultater end de seks batterier der giver 9.0V. Forskellen er ca. 7 sekunder (Som kan ses i e to videoer) på den pre-byggede bane.
Det tog lang tid at fremtænke et kugleleje, tid som kunne være blevet brugt bedre.

Hvis vi skulle lave dette projekt med den viden vi har nu, ville vi bygge robotten anderledes. Vi ville sørge for at robotten blev mindre, og mere robust i sin opbygning, da vi har haft store problemer med stabiliteten, og derved også resultaterne. Hvis vi havde haft et mere stabilt grundlag, ville vi kunne bruge mere af tiden på at optimere koden og tilføje flere sensorer på robotten, så vi ikke blev nød til at 'hardcode' robotten til at køre rundt om vægen. Bedre gearing kunne også være en ting som vi ville prøve at forbedre hvis vi skulle lave projektet nu.