Marc Ilsø Poulsen, Peter Bartram Hansen og Bjarne Antonsen 10/1-2011
dLego: Projekt Session 3 Rapport
Varighed: 5 timer

Formål:

Formålet for i dag var som det første, at vi skulle finde frem til fejlen i subsumption arkitekturen, som bevirkede, at fjernstyrings-klassen RemoteControl.java ikke blev korrekt undertrykt af de andre opførselslag.

Udover dette skulle vi efterhånden til at lægge os helt fast på det endelige formål med vores projekt, da vi i det oprindelige udkast for vores projekt kun havde en meget vag idé til, hvad det overordnede mål for vores robot-leg skulle være.

Plan:

Vi skal have løst fejlen i subsumption implementationen, hvilket vi ikke håber vil tage alt for lang tid. Derefter skal vi diskutere og lægge os fast på det endelige mål for vores projekt og så arbejde videre med det.

Resultater:

Det viste sig, at der var en ganske simpel grund til, hvorfor vi stadigvæk kunne fjernstyre robotten på de tidspunkter, hvor fjernstyringen ellers burde være undertrykt af AvoidFront laget. Ved modtagelse af styrings-kommandoer fra GUI'en i RemoteControl klassen, så tilgik vi motorerne direkte vha. TachoPilot klassen:

(RemoteControl.java)

```
case FORWARD:
      pilot.forward();
      break;
case BACKWARD:
      pilot.backward();
      break;
case LEFT:
      pilot.rotate(99999, true);
      break;
case RIGHT:
      pilot.rotate(-99999, true);
      break;
```

Ved at gøre dette bliver der ikke tjekket, om RemoteControl rent faktisk har lov til at tilgå motorerne eller om klassen er blevet undertrykt af AvoidFront laget. Ved modtagelse af styrings-kommandoer havde vi derfor blot glemt at kalde de rigtige metoder, som befandt sig i superklassen til opførselslagene, Behavior:

(Behavior.java)

```
    // Access to motors
    public void forward()
    {
      if (!isSuppressed()) pilot.forward();
    }

    public void backward()
    {
      if (!isSuppressed()) pilot.backward();
    }

    public void stopBehavior()
    {
      if (!isSuppressed()) pilot.stop();
    }

    public void rotate(int degree, boolean immediateReturn) {
      if (!isSuppressed()) pilot.rotate(degree, immediateReturn);
    }
```

Herigennem fik vi sikret os at fjernstyringslaget ikke havde adgang til motorerne når laget var blevet undertrykt.

Endeligt formål for vores projekt - NXTCart:

I vores oprindelige forslag til, hvad vores projekt skulle gå ud på havde vi tænkt på at lade to fjernstyrede robotter dyste om at indsamle flest mulige objekter. På nuværende tidspunkt i processen tænkte vi dog, at det ville være sjovere at anvende de opførselslag vi indtil videre havde lavet i et lille racerløb imellem de to fjernstyrede robotter.

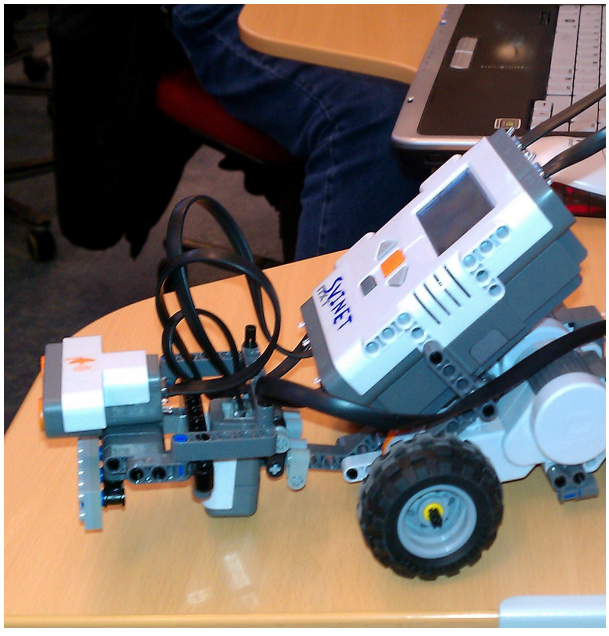
Her ville vi fortsat anvende vores AvoidFront lag til at sørge for, at robotterne ville undgå at styre ind i objekter foran dem. TouchBehavior laget ville vi anvende som et middel for den ene robot til midlertidigt at suspendere fjernstyringen af den anden robot. Dette skulle blot foregå ved at touch sensoren blev placeret et sted på robotterne, hvor det ville være muligt for den anden robot at køre

ind i og trykke på den.

Kapløbet imellem de to robotter skulle bestå af en række checkpoints, som robotterne skulle køre igennem for at nå i mål. Dette forestillede vi os kunne fungere ved, at vi skulle have et antal farvede stykker A4-ark, hvortil vi ville kunne anvende lyssensoren til at detektere, når robotten kørte henover et af disse checkpoints. Vi kunne her tænke os, at det skulle være muligt for robotterne at kommunikere direkte med hinanden, så når en af robotterne kom til et checkpoint skulle den meddele det til den anden robot. Dette ønskede vi at anvende til at tildele en midlertidig fordel tilden førende robot i løbet ved at fjernstyringen af den anden robot skulle være spejlvendt i eksempelvis fem sekunder.

Når den ene robot var nået i mål, så skulle den ligeledes kommunikere dette direkte til den anden robot, så vinderen og taberen af løbet herigennem ville kunne afgøres.

Tilføjelse af lyssensor:



For at muliggøre monteringen af lyssensoren på vores robotter, så måtte vi flytte lidt rundt på placeringen af UltraSonicSensoren. Det lykkedes os at lave en fornuftig placering af begge sensorer foran på robotterne. Til højre kan ses robotternes nye front.

Vi placerede detekteringen af checkpoints i en ny klasse, LightCheckPoint. I konstruktøren sørger vi for at få kalibreret farven på et checkpoint ved at gemme en frisk samplet værdi via lyssensoren. Konstruktøren kan ses nedenfor:

(LightCheckPoint.java)

```
public LightCheckPoint(RemoteControl rc) {
	this.rc = rc;

	ls.setFloodlight(true);

	LCD.drawString("Press ENTER to", 0, 0);
	LCD.drawString("calibrate color", 0, 1);
	LCD.drawString("of checkpoints", 0, 2);
```

```
      while (!Button.ENTER.isPressed()) {
            LCD.drawString("Value: " + ls.readValue(), 0, 4);
      }

      this.checkPointValue = ls.readValue();

      LCD.clearDisplay();
```

I selve control-loopet (som kan ses nedenfor) har vi et uendeligt loop hvori vi sampler værdien af lyssensoren. Hvis vi på et tidspunkt måler en værdi som ligger tæt på den kalibrerede værdi for vores checkpoints (variablen checkPointValue), så anvender vi den måling som indikation på at robotten kan befinde sig på et checkpoint. For at være sikker på at det rent faktisk er tilfældet, så udfører vi i disse situationer op til 5 yderligere målinger med 0,2 sekunders mellemrum. Hvis alle disse målinger også ligger tæt på den kalibrerede værdi, så føler vi os sikre på, at robotten befinder sig på et checkpoint. Vi anvender en boolean variabel, consistent, til at indikere om alle målingerne ligger tæt på checkpoint værdien.

I denne første løsning gør vi ikke andet end at forøge en tællevariabel og anvende Sound klassen, så robotten giver to beeps, når vi har detekteret, at den er ved et checkpoint. Senere ønsker vi, at den skal give besked til den anden NXT enhed via Bluetooth om, at den er nået til et checkpoint.

(LightCheckPoint.java)

```
  @Override
  public void run() {
    while (true) {

      int value = ls.readValue();

      if ((value >= checkPointValue-1) && (value <= checkPointValue+1)) {

        boolean consistent = true;

        for (int i = 0; i < 5; i++) {
          try {
            Thread.sleep(200);
            if (!(value >= checkPointValue-1)&&(value <= checkPointValue+1)) {
              consistent = false;
              break;
            }
          } catch (InterruptedException e) {}
        }

        if (consistent) {
          checkPointsSoFar++;
          Sound.twoBeeps();

          //send a message to other NXT using Bluetooth
```

```
        }
        while (true) {

          LCD.drawString("On checkpoint", 0, 5);

          int test = ls.readValue();
          if ((test < checkPointValue - 1) || (test > checkPointValue + 1)) {
                LCD.drawString("Left checkpoint", 0, 5);
                break;
          } else {
                try {
                      Thread.sleep(200);
                } catch (InterruptedException e) {}
          }
        }
      }
    }
  }
```

Vi diskuterede hvorvidt målingerne med lyssensoren og detekteringen af checkpoints skulle være et opførselslag i vores subsumption arkitektur, eller om det blot skulle være en selvstændig tråd. Da det ikke ville være nødvendigt for denne tråd at tilgå motorerne, og da den ikke på noget tidspunkt skulle undertrykke nogle af de andre lag eller selv blive undertrykt af dem, så valgte vi at lade den fortsætte med at køre som en selvstændig tråd. I RCRobot sørger vi for at starte en instans af LightCheckPoint tråden.

Konklusion:

Ved afprøvning af checkpoint detekteringen observerede vi, at selvom vi har forsøgt detektere hvornår robotten igen har forladt et checkpoint, så er det let at udnytte ved at køre ind og ud af det samme checkpoint hele tiden. Dette er selvfølgelig ikke optimalt, men vi kan ikke rigtig se hvordan vi kan lukke for denne mulighed.

Hvad en robot gør, når den når til et checkpoint, skal vi have udvidet med en Bluetooth forbindelse til den anden robot, så vi gennem denne kommunikation kan holde styr på, hvor langt de to robotter er nået i løbet i forhold til hinanden.

Endelig skal vi have skaffet yderligere en NXT-enhed, da der skal indgå to robotter i spillet.