Marc Ilsø Poulsen, Peter Bartram Hansen og Bjarne Antonsen 11/01-2011
dLego: Projekt Session 4 Rapport
Varighed: 7½ time

Formål

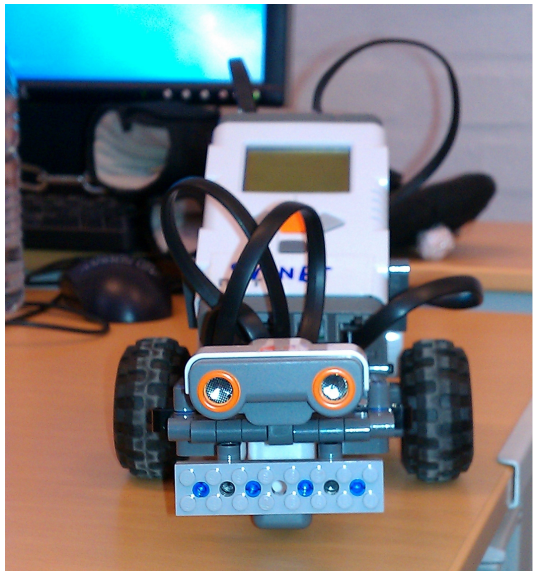
At skaffe endnu en NXT-enhed med tilhørende sensorer, så vi kan lave to ens robotter og forsøge at implementere kommunikation mellem de to NXT-enheder, så de kan sende information om deres tilstand til hinanden.

Plan

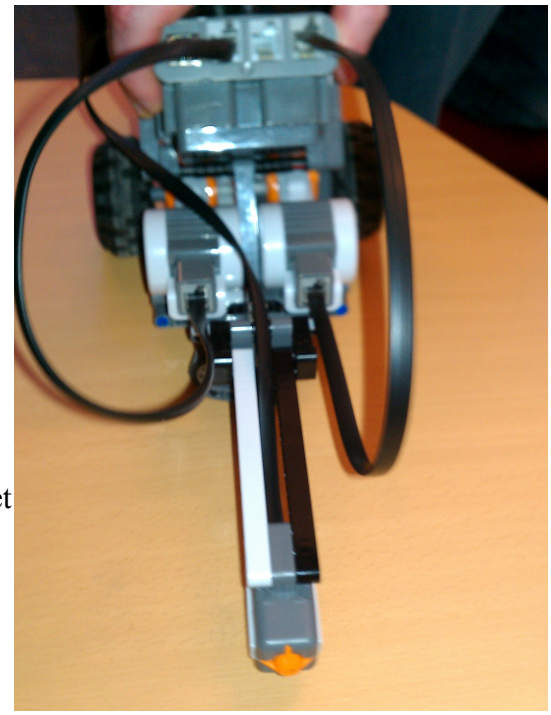
På dette tidspunkt i projektet var det allervigtigste at skaffe en yderligere NXT-enhed, så det havde førsteprioritet denne dag. Derefter skulle vi med den nyligt anskaffede NXT-enhed konstruere endnu en robot, som skulle være fuldstændig magen til vores i forvejen eksisterende robot. Vi besluttede os for at anvende Bluetooth til kommunikation imellem de to robotter, da Bluetooth var understøttet af NXT'erne og derfor virkede mest oplagt. Implementeringen af denne kommunikation skulle vi se på, efter at vi havde bygget de to robotter færdige. Endelig skulle vi diskutere, hvordan vores checkpoints skulle udformes.

Resultater

Efter at have fået skaffet endnu en NXT-enhed gik vi i gang med at konstruere en robot, som var magen til den, som vi allerede havde bygget. Dette voldte os ikke de store problemer, men vi havde nogle overvejelser omkring, hvordan vi kunne sørge for, at UltraSonicSensoren ikke detekterede den anden robot som en forhindring, der skulle undviges. Vi havde nemlig i forvejen monteret en UltraSonicSensor forrest på robotten, da dette modsat det gængse Wall-E design gav mulighed for også at detektere objekter, som var tæt på overfladen, hvorpå robotten kørte.



Denne konstruktion kunne dog muligvis give problemer, da robotterne skulle være i stand til at køre ind i hinanden og dermed sætte TouchBehavior adfærdsmønsteret i gang, men hvis UltraSonicSensorens sampling gav en afstand, som var lav nok, ville AvoidFront adfærdsmønsteret jo forhindre, at den ene robot kunne komme tæt nok på for at trykke på den anden robots TouchSensor. Placeringen af UltraSonicSensoren var et muligt problem i denne sammenhæng, men ved at montere TouchSensoren meget lavt på robotten (se billede til højre) og sørge for, at vi havde en konstruktion, som fik placeret denne et godt stykke bagved selve NXT-enheden, undgik vi, at afstanden fra UltraSonicSensoren på den ene robot til den anden robot kunne blive for lav.



Vi havde indtil dette punkt ikke haft mulighed for at teste, hvorvidt denne konstruktion kunne give problemer, da vi først nu havde skaffet endnu en NXT-enhed, så dette satte vi os for at teste ved på helt enkel vis at lade den ene robot køre op i den anden. Heldigvis viste vores test, at den ene robot sagtens kunne køre helt op bagi den anden robot, uden at den målte afstand kom under vores threshold, så konstruktionen var brugbar til formålet.

Da vores konstruktion af de to robotter var færdig, var det tid til at kigge nærmere på implementeringen af Bluetooth-kommunikation mellem de to robotter. Vores idé med denne kommunikation var, at den ene robot skulle sende en besked til den anden robot, når den var kommet til et checkpoint. Hvis det modtagne checkpoint så var højere end det checkpoint, som man selv var nået til, så skulle robotten have en straf, eftersom den var bagefter i kapløbet. Her havde vi i forvejen lagt op til, at robotten skulle have spejlvendt styring som straf i et kort interval.

Vi forsøgte at implementere Bluetooth-kommunikation mellem de to NXT-enheder ved igen at anvende klassen lejos.nxt.comm.Bluetooth. Idéen var her, at en NXT-enhed kunne agere server og

den anden enhed kunne være klient, sådan at den ene NXT-enhed ville vente på en indadgående forbindelse, mens den anden ville forsøge at oprette en udadgående forbindelse. Vores LightCheckPoint klasse blev udvidet og vi introducerede en ny klasse kaldet BTReceiveThread, som på en NXT-enhed skulle have ansvar for at læse indadgående kommandoer fra en anden NXT. De to klassers implementering kom til at se således ud:

```
(LightCheckPoint.java)

public class LightCheckPoint implements Runnable {


      public LightCheckPoint(RemoteControl rc) {

            ...

            LCD.drawString("Press LEFT to", 0, 0);
            LCD.drawString("be receiver,", 0, 1);
            LCD.drawString("RIGHT to be", 0, 2);
            LCD.drawString("sender!", 0, 3);

            while (!(Button.RIGHT.isPressed() || Button.LEFT.isPressed())) {}

            LCD.clearDisplay();

            if (Button.RIGHT.isPressed()) {
                  LCD.drawString("Connecting...", 0, 0);
                  btc = Bluetooth.connect(Bluetooth.getKnownDevice("Svinet"));
            }

            else if (Button.LEFT.isPressed()) {
                  LCD.drawString("Waiting...", 0, 0);
                  btc = Bluetooth.waitForConnection();
            }

            LCD.drawString("Connected to NXT", 0, 2);

            input = btc.openDataInputStream();
            output = btc.openDataOutputStream();

            BTReceiveThread btr = new BTReceiveThread(this, input);
            Thread receive = new Thread(btr);
            receive.setDaemon(true);
            receive.start();

            try {
                  Thread.sleep(1000);
            } catch (InterruptedException e) {}
      }
      ...
}
```

(BTReceiveThread.java)

```
public class BTReceiveThread implements Runnable {

	private LightCheckPoint lcp;
	private DataInputStream input;

	public BTReceiveThread(LightCheckPoint l, DataInputStream input) {
		this.lcp = l;
		this.input = input;
	}

	@Override
	public void run() {
		while (true) {
			try {
				int message = input.readInt();

				if (message > lcp.getCheckPointsSoFar()) {
					lcp.setReversed(true);
					Thread.sleep(5000);
					lcp.setReversed(false);
				}
			} catch (Exception e) {}
		}
	}
}
```

I LightCheckPoint klassen tilførte vi en del Bluetooth-connection kode i konstruktøren. Da vi havde to NXT-enheder i spil, var en oplagt løsning at lade de to LEFT og RIGHT knapper på NXT'en styre, hvilken enhed der skulle agere server, og hvilken der skulle være klient.

BTReceiveThread klassen sørger for at læse indadgående beskeder og checker, om NXT-enheden er bagefter i kapløbet ved at sammenligne det antal checkpoints som den selv har opnået med den modtagne værdi, som er antallet af checkpoints for den anden NXT-enhed. I tilfælde af, at modtagerens antal checkpoints er det laveste, så bliver styringen spejlvendt.
Selve kernen i forsøget på at oprette Bluetooth-forbindelsen mellem de to robotter kan ses af følgende kode:

```
		if (Button.RIGHT.isPressed()) {
			LCD.drawString("Connecting...", 0, 0);
			btc = Bluetooth.connect(Bluetooth.getKnownDevice("Svinet"));
		}

		else if (Button.LEFT.isPressed()) {
			LCD.drawString("Waiting...", 0, 0);
```

```
                btc = Bluetooth.waitForConnection();
        }
```

Det er netop her, at den ene NXT-enhed op til at lytte efter en indadgående forbindelse og den anden agerer initiator og forsøger at oprette en forbindelse.
Det viste sig dog, at denne fremgangsmåde ikke var mulig, da vi fik kastet en NullPointerException ved Bluetooth.waitForConnection() kaldet. Dette undrede os meget, og vi brugte lang tid på at forsøge at regne ud, hvorfor denne exception blev kastet netop her. Efter lang tids søgen fandt vi ud af, at NXT-enhederne kun havde understøttelse for en enkelt indadgående og op til tre udadgående forbindelser, og da vi i vores program allerede havde oprettet forbindelse fra en PC til en NXT, så var der ikke flere ledige indadgående forbindelser.
Vores første idé til en løsning på dette problem var at lade NXT-enhederne være initiators i stedet for at gøre dette fra PC-siden, da vi derved ikke ville støde ind i begrænsningen om kun at have en enkelt indadgående forbindelse til rådighed på NXT-enhederne.
Denne idé var dog ikke uden problemer, da det viste sig, at LeJOS API'en ikke understøttede Bluetooth-kommunikation, hvor PC'en var server. Derfor var vi nødt til at bruge en helt anden løsning, hvis vi ville kunne kommunikere imellem NXT-enhederne.

Konklusion

Vores konstruktion af de to robotter er på nuværende tidspunkt færdig, og vi har ikke yderligere planer om at ændre på denne.
Med hensyn til at lade NXT-enhederne kommunikere direkte via. Bluetooth, så stødte vi som nævnt ind i nogle begrænsninger og måtte komme på en anden løsning. Her havde vi den idé, at vi kunne anvende de to PC'er til at sende beskederne imellem, da disse jo allerede havde forbindelse til NXT enhederne hver for sig. For at implementere denne løsning ville det kræve, at der også blev oprettet forbindelse mellem de to PC'er. En idé til en netværksarkitektur, som vi kunne arbejde videre med til næste session, kunne så eventuelt se sådan ud:

NXT 1 <---> GUI 1 <---> GUI 2 <---> NXT 2