Lab Session 3

Test of the Sound Sensor

Til den første opgave skulle vi montere lydsensoren på vores sædvanlige robot. I første omgang læste vi ikke opgaven på ugesedlen grundigt nok, så vi antog at linket til SonicSensorTest.java var det program, som vi skulle bruge. Efter lidt forvirring indså vi dog hurtigt at programmet var det samme program som ved lab session 2, og at opgaven i stedet bad os om selv at tilrette programmet til at anvende lydsensoren. Dette var hurtigt gjort (skifte fra at bruge UltrasonicSensor klassen til SoundSensor klassen). Ved at lave forskellige lyde i form af klap og råb under kørsel af programmet, så kunne vi aflæse forskellige sample værdier på displayet som forventet.

Data logger

De to udleverede programmer, DataLogger.java og SoundSampling.java, blev benyttet til en simpel indsamling af data fra lydsensorens målte værdier. Ved at anvende "nxjbrowse" kunne vi hente de loggede data ned som en .txt fil. Nedenfor følger uddrag af denne fil:

"86,87,87,88,89,89,90,91,91,92,92,93,93,92,88,80,76,68,62,55,

52,47,44,40,36,32,30,28,26,24,22,20,19,17,17,15,13,12,12,11,

10,10,9,8,8,8,8,8,7,6,6,6,6,5,5,5,4,4,4,4,

4,4,4,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,

3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3, [...]"

Vi kunne ikke umiddelbart finde et program til plotning af disse data på en graf...

Sound Controlled Car

Udførslen af de to programmer, SoundCtrCar.java og Car.java, viste at vi kunne få robotten til at skifte mellem at køre fremad, til højre, til venstre og til sidst stoppe helt, hvorefter den igen kørte fremad. Skiftene mellem disse skete hver gang vi foretog en lyd (f.eks. klap), som var højt nok. Af koden kan det ses hvordan programmet virker. Den primære part af programmet er denne while løkke:

```
while (! Button.ESCAPE.isPressed())
{
    waitForLoudSound();
    LCD.drawString("Forward ",0,1);
    Car.forward(100, 100);

    waitForLoudSound();
    LCD.drawString("Right ",0,1);
    Car.forward(100, 0);

    waitForLoudSound();
    LCD.drawString("Left ",0,1);
    Car.forward(0, 100);
```

```
    waitForLoudSound();
    LCD.drawString("Stop ",0,1);
    Car.stop();
}
```

Programmet indeholder desuden en variabel, "soundThreshold", som er sat til 90. Denne bruges i waitForLoudSound, så den busy-waiter så længe lydsensoren ikke har aflæst en lyd med værdi større end 90.

En ulempe ved programmet er at den eneste måde at få programmet til at stoppe er ved at man skal holde ESCAPE knappen nede i den sidste del af while-løkken. Dette kunne vi gøre smartere ved at tilføje en ButtonListener, som lyttede på et ESCAPE.isPressed event ved at "override" buttonPressed metoden.

Clap Controlled Car

Til denne opgave modificerede vi programmet fra første delopgave, så vi i programmet anvender en liste, hvor vi ved hver sampling tilføjer den målte værdi til listen. For at listen ikke bliver enorm, så sørger vi for at kun gemme de seneste 56 sample værdier i denne liste således at vi jvf. Sivan Toledo's definition på klap-detektion har lige præcis nok målinger til at vi kan detektere et klap.

Uddrag af koden fra vores ClapDetector.java klasse:

```
static List<Integer> data = new ArrayList<Integer>();


private static boolean detectClap()
{
    if (data.get(0) < 50)
    {
        for (int i = 1; i < 6; i++)
        {
            if (data.get(i) > 85)
   {
                for (int j = i+1; j < i+51; j++)
        {
            if (data.get(j) < 50) return true;
        }
            }
        }
    }
    return false;
}
```

Algoritmen virker OK til at detektere klap, men det er dog også forholdvis let at "snyde" den ved at lave andre høje og korte lyde. Man skal desuden klappe helt tæt på sensoren før den reagerer på klappet (Men det er sandsynligvis sensorens skyld).