Design of Embedded Systems (DES)

Jozef Hooman

http://www.cs.ru.nl/~hooman/DES/

Course 7
15 October 2014
Planning

• **Today**: discuss #11 (clock demo) + first assignment with small Lego Rover using LeJOS & Eclipse

• **22 Oct**: discuss Rover results + Bluetooth extension
  NOTE: in **HG 00.071**

• **29 Oct & 5 Nov**: Autumn break
  when needed: **finalize Xenomai part**

• **12 Nov**: discuss Rover results + DSL/Eclipse explanation
  NOTE: in **HG 01.028**
Global Planning

**Course**
- Scheduling Theory
- TIMES exercises

**Instruction**
- Xenomai Exercises

**Group work**
- Exercises with Xenomai and TIMES

**Aim:**
Learn and try new techniques to raise level of abstraction

**15 Oct**

**demo 14 January 2015**

- Feedback on models and implementation
- Modeling and implementation of Lego Mars Rover
Generations of programming languages

**1st generation**
Machine instructions

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>100011</td>
<td>00011</td>
<td>01000</td>
<td>00000 00001 000100</td>
</tr>
</tbody>
</table>

**2nd generation**
Assembly code

```
add eax, edx
shr eax, 8
```

**3rd generation**
High-level programming languages (Fortran, Cobol, C, Java, ..)

```
if (frameSize.height > screenSize.height) {
    frameSize.height -= screenSize.height;
}
if (frameSize.width > screenSize.width) {
    frameSize.width -= screenSize.width;
}
```

**4th generation**
Higher abstraction & statement power

![Diagram showing code generation flow from assembler to compiler to code generator.](Image)
Domain Specific Languages

Domain Specific Language (DSL)

• Next generation of programming using
  – dedicated language for a certain domain
  – dedicated transformations to code
    (and models, documentation, …)

To be able to develop DSL and transformations, start bottom-up to obtain:

• Knowledge of the domain (Lego NXT robots)
• Experience with coding (patterns) for Lego NXT robots
Programming Lego NXT using LeJOS

• Installation
  – USB driver, Java, NXJ, Eclipse

• Configuration small NXT Rover

• Programming
  – motors
  – sensors

Note: there are 4 “new” PCs in the lab, which can also be used to install it all
Installation Driver & Java

• Install USB driver, so called **NXT Fantom Driver**
  

  Choose NXT Fantom Download

• **32 bit** JDK for Java, e.g., jdk-8u20-windows-i586.exe
  

  Currently:
  

  Accept license agreement and select download
  
  (e.g., jdk-8u20-windows-i586.exe)
Installation NXJ


  For Windows, use … *win32_setup.exe* for automatic installation on Windows
  ( e.g., to  C:\Program Files (x86)\leJOS NXJ )
Installation Eclipse

• Download 32 bit version of Eclipse see
  https://www.eclipse.org/downloads/
  Choose 32 bit version of
  Eclipse IDE for Java and DSL Developers

[Place short cut to eclipse.exe on Desktop]
Specify Java version for Eclipse

When there are multiple versions of Java, put the proper Java version in `eclipse.ini`.

With a 32 bit JDK in

C:\Program Files (x86)\Java\jdk1.8.0_20

add 2 lines to `eclipse.ini` as follows:

openFile
--launcher.appendVmargs
-vm

C:\Program Files (x86)\Java\jre1.8.0_20\bin\javaw.exe

-vmargs
-Dosgi.requiredJavaVersion=1.6
-Xms128m
-Xmx1024m

Note: might also be useful for other versions of Eclipse
Install Eclipse plugin

• Create an empty directory, e.g. *LeJOSprograms*
• Start Eclipse with *eclipse.exe* and use newly created directory as workspace  [close Welcome screen]
• Install NXJ plugin
  Help > Install New Software
  Work with:  http://www.lejos.org/tools/eclipse/plugin/nxj/
• Restart when this is requested
• See Help > Help Contents > leJOS NXJ

See [http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm](http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm) for screenshots
Ignore section on creating own PC project
Configure plugin

- Window > Preferences > leJOS NXJ
- check/change directory of NXJ_HOME
- Defaults for run mode & debug mode:
  - not: Run program after upload
  - check Link Verbose

Note: leJOS NXJ firmware has already been uploaded to Lego Rovers
Create new leJOS project:
File > New > Project > LeJOS: LeJos NXT Project
give name and Finish

Right click on “src” folder > New > Class
→ Try Hello World example of:
Creating your own NXT project
http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm#8
Upload program to NXT brick

Right-click on program name or in editor; select Run As > LeJOS NXT Program
Info buttons & examples

• For the buttons on the NXT brick see
  The [leJOS] NXJ Menu System
  http://www.lejos.org/nxt/nxj/tutorial/MenuSystem/MenuSystem.htm

  ➔ stop program by ENTER + ESCAPE  
  (orange and dark grey button simultaneously)

• See section:
  Importing the samples and examples into Eclipse
  http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm#2
NXT brick

Sensor ports S1, S2, S3, S4

USB to download programs on brick

Motor ports A, B, C
Small NXT Rover

S1: light sensor
S2: ultrasonic sensor
S3: touch sensor
A: left motor
B: right motor (as seen from back)
C: lamp (control as motor)
Charge NXT brick when battery low

Connect below sensor ports:
- charging: green and red light
- charged: green only
Motors

See:
http://www.lejos.org/nxt/nxj/tutorial/MotorTutorial/ControllingMotors.htm
http://www.lejos.org/nxt/nxj/api/index.html (or help in Eclipse):
Classes: Motor, MotorPort, NXTRegulatedMotor, ...

Some program fragments (just as example!):

```java
import lejos.nxt.Motor;
import lejos.robotics.RegulatedMotor;

static RegulatedMotor rightMotor = Motor.A;
static RegulatedMotor leftMotor = Motor.B;

rightMotor.setSpeed(speed);     rightMotor.forward();     rightMotor.stop();
leftMotor.setSpeed(speed);      leftMotor.forward();      leftMotor.stop();

rightMotor.rotate(-720,true);  leftMotor.rotate(-720,true);

rotate(360): returns when finished
rotate(360,true): returns immediately
```

Note: see also DifferentialPilot for control of 2-wheel robot
Lamp

Control lamp as motor

**Not** as “regulated motor”: continuous control

→ lamp always on

Use MotorPort:

- lamp on:   MotorPort.C.controlMotor(100,1);
- lamp off:  MotorPort.C.controlMotor(0,3);

orientation of light on underlying brick is important

```
void lejos.nxt.MotorPort.controlMotor(int power, int mode)
Low-level method to control a motor.

Specified by: controlMotor(...) in BasicMotorPort
Parameters:
  power  power from 0-100
  mode   defined in BasicMotorPort. 1=forward, 2=backward, 3=stop, 4=float.
See Also:
  BasicMotorPort.FORWARD
  BasicMotorPort.BACKWARD
  BasicMotorPort.FLOAT
  BasicMotorPort.STOP
```
Sensors

Program which reads and displays sensor values:

```java
static LightSensor light = new LightSensor(SensorPort.S1);
static UltrasonicSensor sonar = new UltrasonicSensor(SensorPort.S2);
static TouchSensor touch = new TouchSensor(SensorPort.S3);

public static void main(String[] args) {
    while(!Button.ESCAPE.isDown()){
        LCD.drawString("Read light value", 0, 2);
        LCD.drawInt(light.getLightValue(), 4, 0, 3);
        LCD.drawInt(light.getNormalizedLightValue(), 4, 0, 4);
        LCD.drawInt(SensorPort.S1.readRawValue(), 4, 0, 5);
        LCD.drawInt(sonar.getDistance(), 4, 0, 6);
        // next is same as getLightValue
        //LCD.drawInt(SensorPort.S1.readValue(), 4, 0, 6);
    }
}
```

just another way to stop
Other topics

- time
  
  ```java
  long startTime = 0;
  startTime = System.currentTimeMillis();
  ```

- sound
  
  ```java
  Sound.beep();
  Sound.buzz();
  ```

- LCD:
  
  http://www.lejos.org/nxt/nxj/tutorial/LCD_Sensors/LCD_Sensors.htm

- behaviours:
  
  http://www.lejos.org/nxt/nxj/tutorial/Behaviors/BehaviorProgramming.htm
Subsumption architecture

http://en.wikipedia.org/wiki/Subsumption_architecture

Classical approach by Brookes et al for autonomous robots

higher level layers subsume the roles of lower level layers when the sensory information determines it

NOTE: many of the “samples” follow this pattern
Subsumption architecture in LeJOS

• Define collection of behaviours, each behaviour implements three methods:
  – `takeControl()`: true if behaviour should become active
  – `action()`: executed when behaviour becomes active
  – `suppress()`: immediately terminates execution of `action`

• Use instance of Arbitrator, with array of behaviours – in the order of increasing priority; when started:
  – executes `action` of highest prio behaviour for which `takeControl` is true
  – this is repeated, calling `suppress` to stop active behaviour
Assignment 22 October 2014

- Develop a LeJOS program for small NXT robot which
  - stays within black border
  - detects collisions with object (*use bottle*)
  - tries to avoid collisions with object
  - indicates status by lamp and sound

→ try to use program structure that could be generated

Mail source files + short description before Tuesday 21 October 18:00