Embedded Systems - Embodied Agents, Robot Programming in Java for the NXT Mindstorms

OLE CAPRANI

Associate professor, Aarhus University
LEGO Mindstorms
LEJOS
Java for LEGO Mindstorms

Home
NXT Brick
leJOS NXJ
API
PC API
Tutorial
Downloads
FAQ

RCX Brick
leJOS RCX
API
Tutorial
Downloads
FAQ

Forum
Books
Links
Contact

SOURCEFORGE
net
java.net
MEMBER

LEJOS News:

February 06, 2012 11:30 PM
leJOS NXJ 0.9.1 is available for download. Big thanks to all leJOS developer who made that happen. This release includes many bug fixes, new sensor drivers, and even new leJOS tools such as nxjchartinglogger and nxjmapcommand. Consult the release notes included with any release for a detailed list of changes.
# LeJOS Architecture

## LeJOS System Menu
- Data Logger
- Collections/ArrayList Etc.
- Debug/Trace
- UI/Notes/Wav Audio
- Remote Commands
- Behaviors
- Navigation

## User Programs
- File System
- Java Language Support Enum/Exception/String
- Display/Sound
- Comms USB/BT RS485
- Driver Classes Motor/Sensors

##骨架构
- Loader
- Byte Code Interpreter
- Native Methods
- Garbage Collector
- Thread Scheduler
- Timers
- RS485
- PDM
- SPI
- I2C
- TWI
- USART
- USB

---

**Flash I/O**

**Flash File System**

**ATMEL I/O Controller**

**Bluetooth**
Class LCD

```java
java.lang.Object
 | lejos.nxt.LCD
```

```
public class LCD
extends Object

Text and graphics output to the LCD display.

static void `drawInt`(int i, int x, int y)
    Display an int on the LCD at specified x,y co-ordinate.

static void `drawInt`(int i, int places, int x, int y)
    Display an int on the LCD at x,y with leading spaces to occupy at least the number of characters specified by the places parameter.

static void `drawString`(String str, int x, int y)
    Display a string on the LCD at specified x,y co-ordinate.
```
Week 2

PID controllers and Embedded Java API

Articles

- PID Control.  
  Chapter 5, pp. 179-190 of  
  Fred G. Martin, 
  Robotic Explorations: A Hands-on Introduction to Engineering,  
- NXT LCD and Button.
- leJOS, Java API for LEGO Mindstorms NXT, especially the classes Battery, LCD and Button.

Further Reading

- Control System (Wikipedia).
- PID controller (Wikipedia).
- Vance J. VanDoren, Understanding PID Control.

Lab Material

- NXT Programming, Lesson 1
- Compilation and upload of Java programs for the NXT is described in the leJOS NXJ installation guide.
Control Systems
Fred Martin, Chapter 5

 HandyBug

two wheels driven by two independent motors

bend sensor

wall

floor
bend sensor value:

high value - close to wall

low value - away from wall

```c
void main()
{
    calibrate();
    ix= 0;

    while (1) {
        int wall= analog(LEFT_WALL);
        printf("goal is %d; wall is %d\n", goal, wall);

        if (wall < goal) left(); /* too far from wall -- turn in */
        else right(); /* turn away from wall */

        data[ix++]= wall; /* take data sample */
        msleep(100L); /* 10 iterations per second */
    }
}```
NXT Programming

Lesson 1

In this lesson we build a LEGO car to be controlled by the LEGO Mindstorms NXT. Then we install the leJOS Java system, [1], and use this to compile and upload a Java program to the NXT. The program will make the car follow a black line on a white surface.

The 9797 LEGO car

In the LEGO Mindstorms Education NXT Base Set 9797 there is a building instruction for a car, page 8 to page 22. Page 32 to page 34 shows how a light sensor can be added to the car. Build this car with a light sensor added.

![Figure 1](https://example.com/figure1.jpg) The 9797 LEGO car with two motors.

A Java Control Program: LineFollower

The first Java program that we are going to execute on the NXT is the following Java program that makes the LEGO car follow a black line on a white surface: [LineFollower.java](https://example.com/LineFollower.java):
LightSensor light = new LightSensor(SensorPort.S3);
final int blackWhiteThreshold = 45;

// Use the light sensor as a reflection sensor
light.setFloodlight(true);

LCD.drawInt(light.readValue(), 3, 9, 0);

if (light.readValue() > blackWhiteThreshold){
MotorPort.B.controlMotor(0, stop);
MotorPort.C.controlMotor(power, forward);

MotorPort.B.controlMotor(power, forward);
MotorPort.C.controlMotor(0, stop);
// Follow line until ESCAPE is pressed
while (! Button.ESCAPE.isPressed()){

    if (light.readValue() > blackWhiteThreshold){
        // On white, turn right
        LCD.drawString(right, 0, 1);
        MotorPort.B.controlMotor(0, stop);
        MotorPort.C.controlMotor(power, forward);
    }
    else{
        // On black, turn left
        LCD.drawString(left, 0, 1);
        MotorPort.B.controlMotor(power, forward);
        MotorPort.C.controlMotor(0, stop);
    }

    LCD.drawInt(light.readValue(), 3, 9, 0);
    LCD.refresh();
    Thread.sleep(100);
}
public void pidControl() {
    while (!Button.ESCAPE.isPressed()) {
        int normVal = ls.readNormalizedValue();

        // Proportional Error:
        int error = normVal - offset;
        // Adjust far and near light readings:
        if (error < 0) error = (int)(error * 1.8F);

        // Integral Error:
        int_error = ((int_error + error) * 2)/3;

        // Derivative Error:
        int deriv_error = error - prev_error;
        prev_error = error;

        int pid_val = (int)(KP * error + KI * int_error + KD * deriv_error);

        if (pid_val > 100)
            pid_val = 100;
        if (pid_val < -100)
            pid_val = -100;

        // Power derived from PID value:
        int power = Math.abs(pid_val);
        power = 55 + (power * 45) / 100; // NORMALIZE POWER
        Motor.B.setPower(power);
        Motor.C.setPower(power);

        if (pid_val > 0) {
            Motor.B.forward();
            Motor.C.forward();
        } else {
            Motor.B.backward();
            Motor.C.backward();
        }
    }
}
Alishan train track
analog quantity, e.g. light

digital quantity, e.g. light %

stimuli → sensor → interface → control program

response

close control system

control program → interface → actuator

digital quantity, e.g. power %
analog quantity, e.g. pulse train
Behavior of a robot depends on

1. Environment
2. Physical robot
3. Control program

stimuli → sensor → interface → control program

Response

control program → interface → actuator

digital quantity, e.g. power %
analog quantity, e.g. pulse train

digital quantity, e.g. light %
analog quantity, e.g. light
Embodied Agents

Articles

Industrial robot

Robot baby seal Paro

Sequential strategy

Reactive strategy
Ghost control program

- StayOnWhite
- Avoid

black/white sensors

touch sensor
Motivation Networks – A Biological Model for Autonomous Agent Control
End course projects
```python
def snowflake(size, level):
    3.times {
        side(size, level)
        rt 120
    }

def side(size, level)
    if (level == 0)
        fd size
        return
    side(size/3, level-1)
    lt 60
    side(size/3, level-1)
    rt 120
    side(size/3, level-1)
    lt 60
    side(size/3, level-1)

clean()
lt 30
setpos(0, -100)
snowflake(250, 4)
```
IDEAL AND REAL SYSTEMS: A Study of Notions of Control in Undergraduates Who Design Robots

FRED G. MARTIN

Epistemology and Learning Group
Learning and Common Sense Section
The Media Laboratory
Massachusetts Institute of Technology

Teaching powerful ideas with autonomous mobile robots

Rolf Pfeifer
AI Lab, Department of Computer Science
University of Zurich, Switzerland
pfeifer@ifi.unizh.ch