Robotics and Autonomous Systems

Lecture 3: Programming robots

Richard Williams

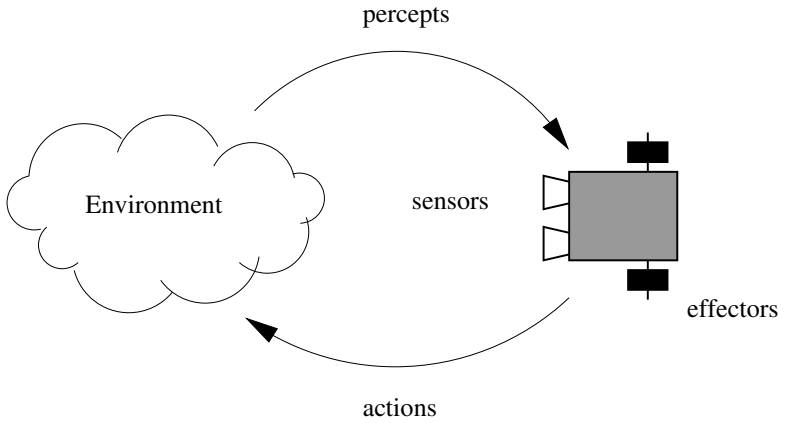
Department of Computer Science
University of Liverpool

UNIVERSITY OF LIVERPOOL

Today

- Before the labs start on Monday, we will look a bit at programming robots.
- There will be three bits
  - Programming robots in the abstract.
  - The NXT
  - LeJOS
- The NXT is the robot we will use for the labs and the projects.
- LeJOS is the language we will use to program the NXT.

The scenario (again)



- Here's one view of the basic program loop.

Basic control loop

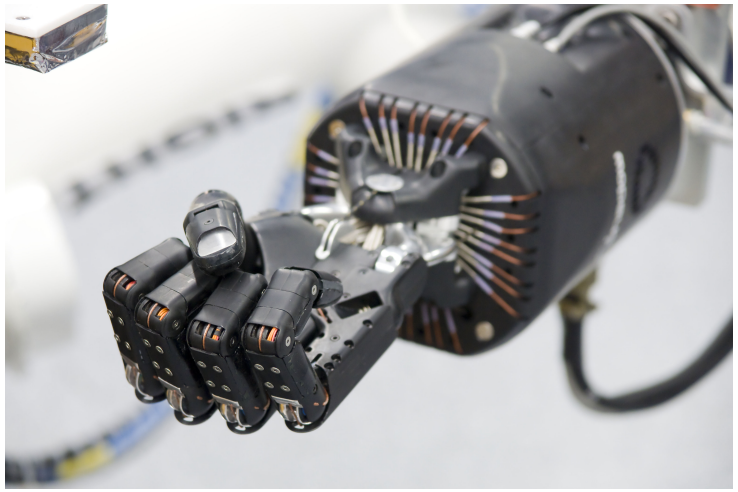
- Here it is again.

```
while(true){
  read sensors
  update internal datastructures
  make decisions
  set power on motors
}
```

- Even robots with arms and legs move them by turning motors on and off.

Basic control loop

Basic control loop



Basic control loop

- We'll get into more detail about how exactly sensors work later in the course.
- For now, just think of them as generating a value.

Basic control loop

- We won't get into more detail about how motors work.
- Just think of a motor command as setting the power on the motors.
- Setting the voltage on a particular port.

Remember timescales

- What will this:

```
while(true){
  switch motors on
  :
  do stuff
  :
  switch motors off
}
```

make the robot do?

Remember timescales

- Even a slow processor runs a lot faster than the robot.
- This:

```
while(true){
  switch motors on
  :
  do stuff
  :
  switch motors off
}
```

- No time for the robot to move.

Remember timescales

- Instead you need:

```
while(true){
  read sensors
  :
  do stuff
  :
  if sensors say "this"{
    switch motors on
  }
  if sensors say "that"{
    turn motors off
  }
}
```

where `this` and `that` are suitably complementary.

Remember control flow

- what will this:

```
while(true){
  read sensors
  if sensors say "this"{
    switch motors on
  }
  :
  wait
  :
  if sensors say "that"{
    turn motors off
  }
}
```

make the robot do?

Remember control flow

- You have to read the sensors for their state to change.

```
while(true){
  read sensors
  if sensors say "this"{
    switch motors on
  }
  :
  wait
  :
  if sensors say "that"{
    turn motors off
  }
}
```

will typically make the robot crash.

Remember control flow

- The robot won't change internal state in a wait.

```
while(true){
  read sensors
  if sensors say "this"{
    switch motors on
  }
  :
  wait
  :
  read sensors
  if sensors say "that"{
    turn motors off
  }
}
```

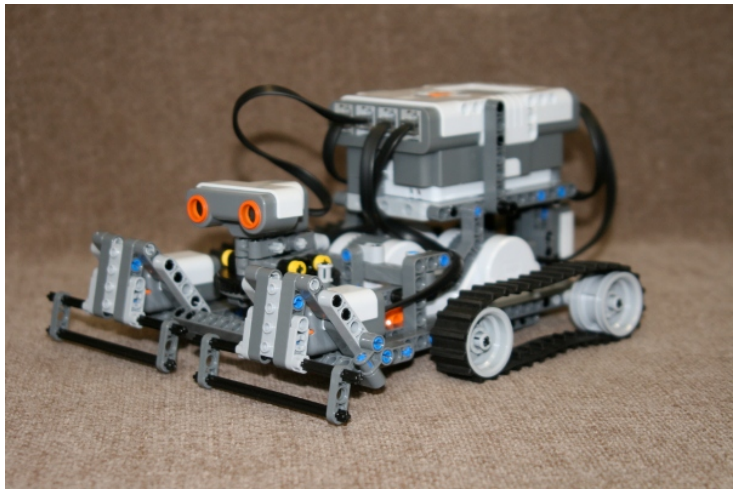

Remember control flow

- If you need to wait, use a counter.

```
while(true){
  read sensors
  if sensors say "this"{
    switch motors on
    set counter to 0
  }
  :
  if counter > limit{
    turn motors off
  }
  :
  increment counter
}
```

- More elegant variations exist using the various clock functions.

Our robot



The brain/heart

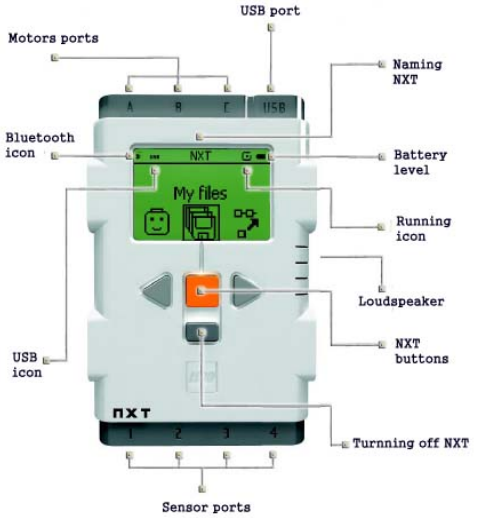
The brain/heart

- The brick is a small computer
  powered by a Li-Ion rechargeable battery
  intended for robot control.
- 32-bit ARM7 processor, running at 48 MHz
- 256 Kbytes non-volatile flash storage
  for storing programs
- 64 Kbytes RAM
  for runtime memory
- Speaker

The brain/heart

- If that seems under-powered, bear in mind it is a lot more capable than its predecessor.

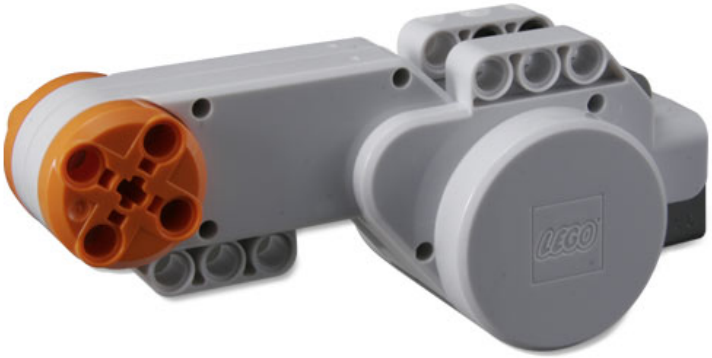
The brain/heart schematic



The brain/heart schematic

- 3 ports (A, B and C) for actuators/motors.
- 4 ports (1 to 4) for sensors.
- LCD display panel (monochrome, 100x64)
- 4 input buttons (left, right, center, escape)
- 1 USB port (for downloading programs on the brick)
- Bluetooth connection (for communication with computer)

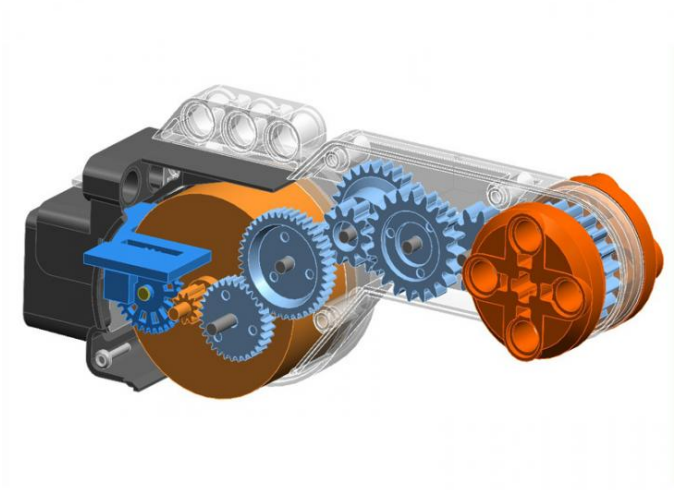
Motors



Motors

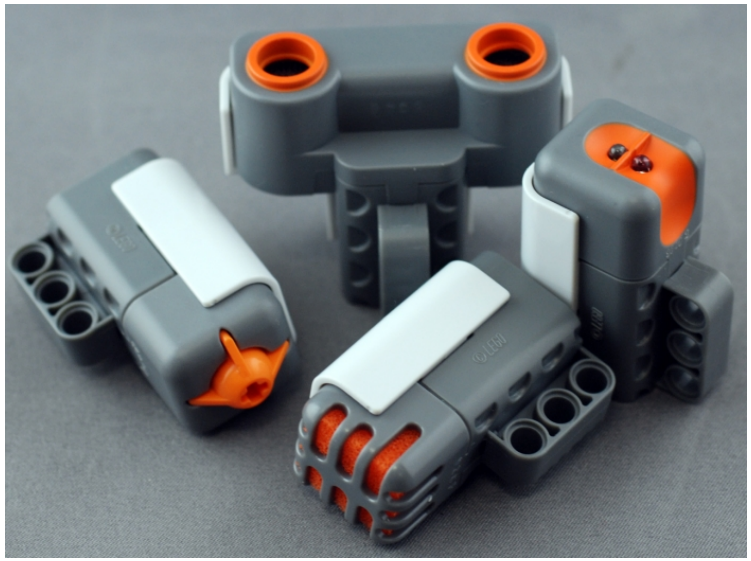
- These are the basic actuators of the NXT kit.
- These are servomotors
  - Motor plus feedback
  - A sensor measures information about rotation and supplies it back to the brick.
  - The motor can be controlled very precisely.
- We will use these to provide locomotion.
- Other uses involve robotic arms, or active sensor support

Motors



- Lots of gears in there also.

Sensors



Touch

Touch

- Detects when the orange button is pressed.
- Returns a Boolean
  - TRUE = sensor is pressed
  - FALSE = sensor is not pressed
- Mechanically the sensors work better if there is a bumper that presses the sensor.
- Our robot has two bumpers — left and right.

Sonar/Ultrasound

Sonar/Ultrasound

- Standard range sensor.
  - We will talk more about how sonar work in a later lecture.
- Reports distance to object.
- Works for objects between 5 and 255cm
  - Precision of $\pm$3cm
- Objects beyond 255cm away report 255.
- Sensitive to the kind of objects it is detecting and subject to errors due to specular reflection.

Light/Colour

Light/Colour

- Emits light and measures the reflection.
- Analyses the colour of the reflection.
- Can distinguish between a number of different colours.

Overall

- Might not seem much, BUT:
  - Has pretty much the same capabilities as any mobile robot.
  - Can do all the mobile robot things I mentioned last time.
- Just as fun and frustrating.

LeJOS

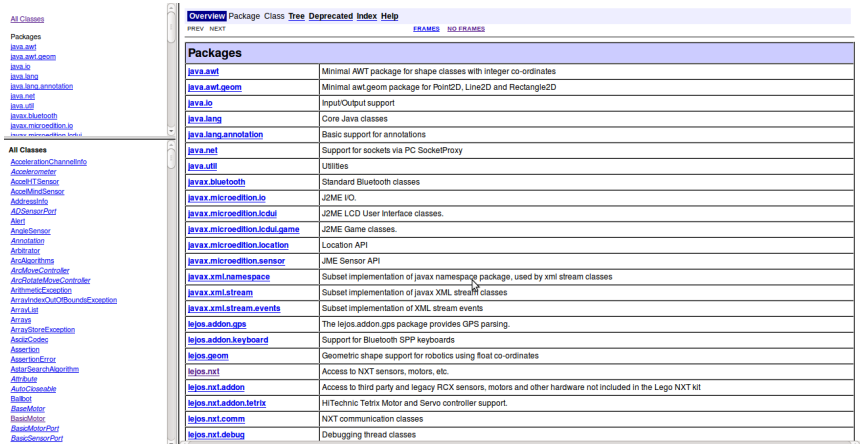
- Lego Java Operating Systems

- Spanish pronounciation:
  - Lay-Hoss
- A small JVM is downloaded onto the brick and allows Java programs to be executed.
- Some standard Java things are missing.
  - Original LeJOS VM was called “Tiny VM”

LeJOS

- However LeJOS has lots of useful robot-specific stuff.
- Java-based, so OO.
- The robot-specific things are implemented as classes + objects.
- Classes to represent, for example:
  - Buttons
  - Motors
  - Sensors
- Access to these devices is then through method calls.

LeJOS



All Classes

Packages
java.awt
java.awt.geom
java.io
java.lang
java.lang.annotation
java.net
java.util
javax.bluetooth
javax.microedition.io

All Classes
AccelerationChannelInfo
Accelerometer
AccelHTSensor
AccelMindSensor
AddressInfo
ADSensorPort
Alert
AngleSensor
Annotation
Arbitrator
ArcAlgorithms
ArcMoveController
ArcRotateMoveController
ArithmeticException
ArrayIndexOutOfBoundsException
ArrayList
Arrays
ArrayStoreException
AsciiCodes
Assertion
AssertionError
AstarSearchAlgorithm
Attribute
AutoCloseable
Ballbot
BaseMotor
BasicMotor
BasicMotorPort
BasicSensorPort

Overview Package Class Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Packages	
java.awt	Minimal AWT package for shape classes with integer co-ordinates
java.awt.geom	Minimal awt.geom package for Point2D, Line2D and Rectangle2D
java.io	Input/Output support
java.lang	Core Java classes
java.lang.annotation	Basic support for annotations
java.net	Support for sockets via PC SocketProxy
java.util	Utilities
javax.bluetooth	Standard Bluetooth classes
javax.microedition.io	J2ME I/O.
javax.microedition.lcdui	J2ME LCD User Interface classes.
javax.microedition.lcdui.game	J2ME Game classes.
javax.microedition.location	Location API
javax.microedition.sensor	JME Sensor API
javax.xml.namespace	Subset implementation of javax namespace package, used by xml stream classes
javax.xml.stream	Subset implementation of javax XML stream classes
javax.xml.stream.events	Subset implementation of XML stream events
lejos.addon.gps	The lejos.addon.gps package provides GPS parsing.
lejos.addon.keyboard	Support for Bluetooth SPP keyboards
lejos.geom	Geometric shape support for robotics using float co-ordinates
lejos.nxt	Access to NXT sensors, motors, etc.
lejos.nxt.addon	Access to third party and legacy RCX sensors, motors and other hardware not included in the Lego NXT kit
lejos.nxt.addon.tetrix	HiTechnic Tetrix Motor and Servo controller support.
lejos.nxt.comm	NXT communication classes
lejos.nxt.debug	Debugging thread classes

LeJOS

- In many ways LeJOS is just like other robot programming languages.
- Provides function calls that interface with with the robot hardware.
- Means no robot programming language can be completely general
  - Aspects of the language are specific to the hardware.
- LeJOS for the NXT is different to LeJOS for the RCX.
- Couldn’t use LeJOS to control the TurtleBot.

HelloWorld

```
import lejos.nxt.Button;

public class HelloWorld {
  public static void main (String[] args) {
    System.out.println("Hello World");
    Button.waitForAnyPress();
  }
}
```

- Looks pretty standard.

HelloWorld

- In fact, the completely standard program runs:

```
public class HelloWorld {
  public static void main (String[] args) {
    System.out.println("Hello World");
  }
}
```

- But you won't see the message.
- Why?

HelloWorld

- In fact, the completely standard program runs:

```
public class HelloWorld {
  public static void main (String[] args) {
    System.out.println("Hello World");
  }
}
```

- The message will flick by too fast to see.
- That `Button.waitForAnyPress()` is what holds up the previous program.

Using Motors

- What happens here?

```
import lejos.nxt.Button;
import lejos.nxt.LCD;
import lejos.nxt.Motor;

public class SimpleDriver{
  public static void main(String[] args){
    System.out.println("Press any button to start");
    Button.waitForAnyPress();
    LCD.clear();
    Motor.B.forward();
    Motor.C.forward();
    System.out.println("Press any button to stop");
    Button.waitForAnyPress();
    Motor.B.stop();
    Motor.C.stop();
  }
}
```

Using Motors

- Turns the motors on and off.

```
import lejos.nxt.Button;
import lejos.nxt.LCD;
import lejos.nxt.Motor;

public class SimpleDriver{
  public static void main(String[] args){
    System.out.println("Press any button to start");
    Button.waitForAnyPress();
    LCD.clear();
    Motor.B.forward();
    Motor.C.forward();
    System.out.println("Press any button to stop");
    Button.waitForAnyPress();
    Motor.B.stop();
    Motor.C.stop();
  }
}
```


Using Motors

- What does the robot do?

```
import lejos.nxt.Button;
import lejos.nxt.LCD;
import lejos.nxt.Motor;

public class SimpleDriver{
  public static void main(String[] args){
    System.out.println("Press any button to start");
    Button.waitForAnyPress();
    LCD.clear();
    Motor.B.forward();
    Motor.C.forward();
    System.out.println("Press any button to stop");
    Button.waitForAnyPress();
    Motor.B.stop();
    Motor.C.stop();
  }
}
```


Using Motors

- Note that motors have to be connected to ports B and C for this to have any effect.
  - This is the case for our NXT robot
- Robot control programs are very sensitive not only to the robot chassis, but also to the way the robot is wired.

Using a touch sensor

- To use a touch sensor we need to introduce a couple more objects.
  - The touch sensors themselves; and
  - The sensor port they are connected to.
- Here's a program that uses a touch sensor:

Using a touch sensor

```
import lejos.nxt.Button;
import lejos.nxt.Motor;
import lejos.nxt.SensorPort;
import lejos.nxt.TouchSensor;

public class SimpleSensor{
  public static void main(String[] args){
  TouchSensor leftBump =
              new TouchSensor(SensorPort.S2);
  TouchSensor rightBump =
              new TouchSensor(SensorPort.S1);
    while (!Button.ENTER.isDown()){
      if(leftBump.isPressed()) {
        Motor.B.forward();
        Motor.C.forward();
      }
      if(rightBump.isPressed()){
        Motor.B.stop();
        Motor.C.stop();
      }
    }
  }
}
```


Using a touch sensor

- This style of program is pretty typical.
  - Matches the kind of thing we talked about before.
- A basic loop in which sensors are read and a decision made based on the sensor state.

Development Cycle

- Another difference between the programs you have written before and robot programs.
- You do the usual:
  - Write
  - Compile + Link
  - Debug

  as before.
- There is another step. What is it?

Development Cycle

- Before running the program you have to download it onto the robot.

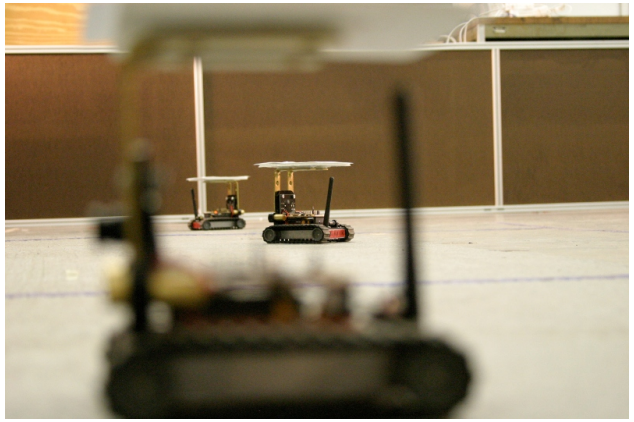
- Through USB or Bluetooth.

Development Cycle

- Same for any robot that has a separate processor.

Development Cycle

- Or at least a separate processor that runs control software



Development Cycle

- If you use Eclipse, the download step is mainly hidden.
- BUT
- The robot must be turned on.
- And the USB cable must be plugged in if you are using it.
- You may also notice that compilation produces a `.nxj` file rather than a `.class` file.

Development Cycle

- If you use don't use Eclipse, you have to run commands to:
  - Compile
  - Link
  - Download

  from the command line.

Documentation

- Website: `http://www.lejos.org/`
- API: `http://www.lejos.org/nxt/nxj/api/index.html`

Summary

- This lecture took a look at some of the issues in programming robots.
- It presented a mixture of high-level concepts and practical advice.
- The lecture also presented some information on LeJOS, which you will use to program the NXT robots.