Embedded Systems - Embodied Agents, Digital Control in a Physical World (Q3+4) (10 ECTS)

Objectives of the course

The participants will after the course have a basis for understanding ***embedded systems***, especially robots, and practical experience with ***physical construction and programming of embedded systems***, especially robots

Lectures (2 h/week), lab sessions (3 h/week)

Lectures
Time: Thurdays at 12.15-14.00,
Location: Store Aud, it-Huset

Lab Sessions
Location: Zuse, Åbogade 40.
  Class A:
  Time: Thurdays at 14.15-17
  Class B:
  Time: Fridays at 10.15-13

Course homepage
https://bb.au.dk/webapps/portal/frameset.jsp?
tab_tab_group_id=_2_1&url=%2Fwebapps
%2Fblackboard%2Fexecute%2Flauncher%3Ftype
%3DCourse%26id%3D_33635_1%26url%3D

LEGOLab

Assessment methods: Lab notebooks and oral exam
7-scale, internal examiner

A **lab notebook from each labsession** is handed in each week (before the next week's labsession) in order to fulfill the **compulsory programme** of the course.

At the end of the course each group do **an end course project**.
Your grade for the course will be based on selected lab notebooks, your lab notebook for the end course project and the oral presentation at the end of the course.

In the lab sessions you will work in **groups of 4**. After each lab session the group should hand in a labnotebook.

During the first two lab sessions each group should enrool as groups on the course webpage.

To perform the activities in the lab sessions each group need **LEGO Mindstorms Material**.

The material can be borrowed during the course.

At the first two lab sessions material will be available for handout.

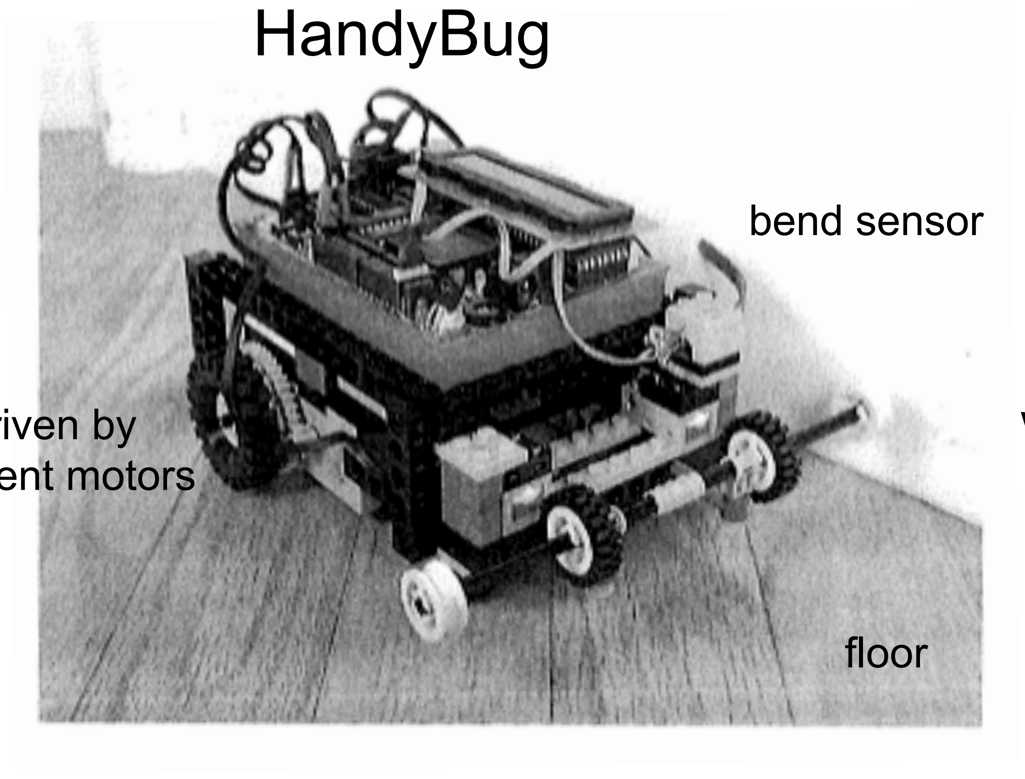
# 9797 LEGO Mindstorms Education NXT Base Set.

# UB22S USB Bluetooth Dongle
# 9833 LEGO Mindstorms transformer

An **embedded system** is a special-purpose computer system designed to perform one or a few dedicated functions.

Life's too short
for the wrong job!
jobsintown.de
47

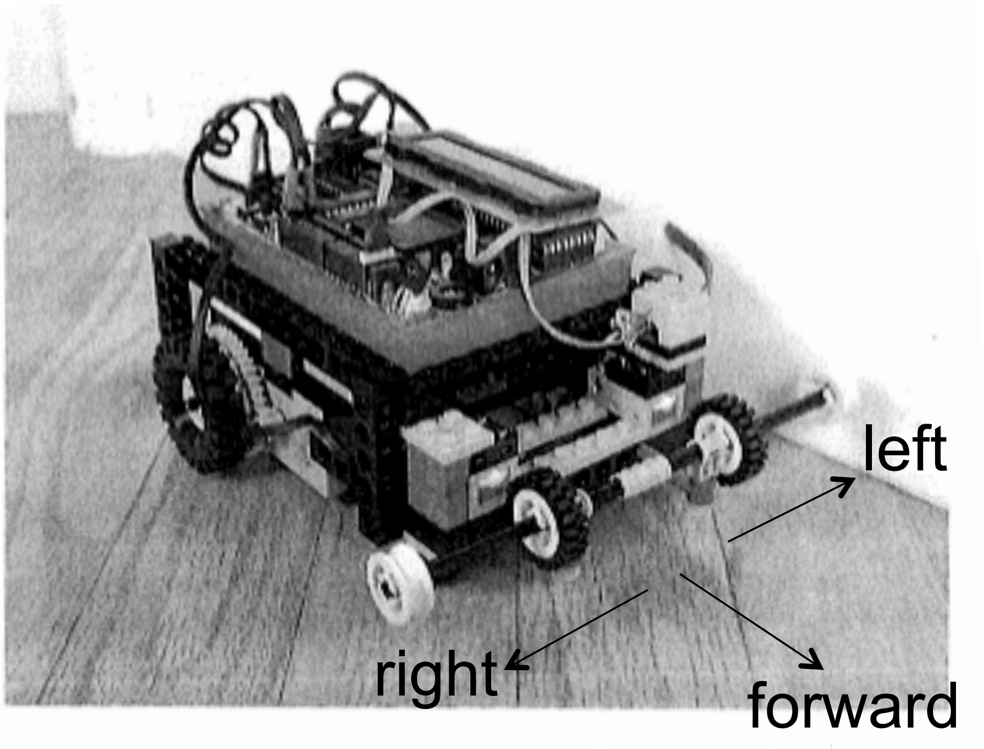
Fred Martin, Chapter 5

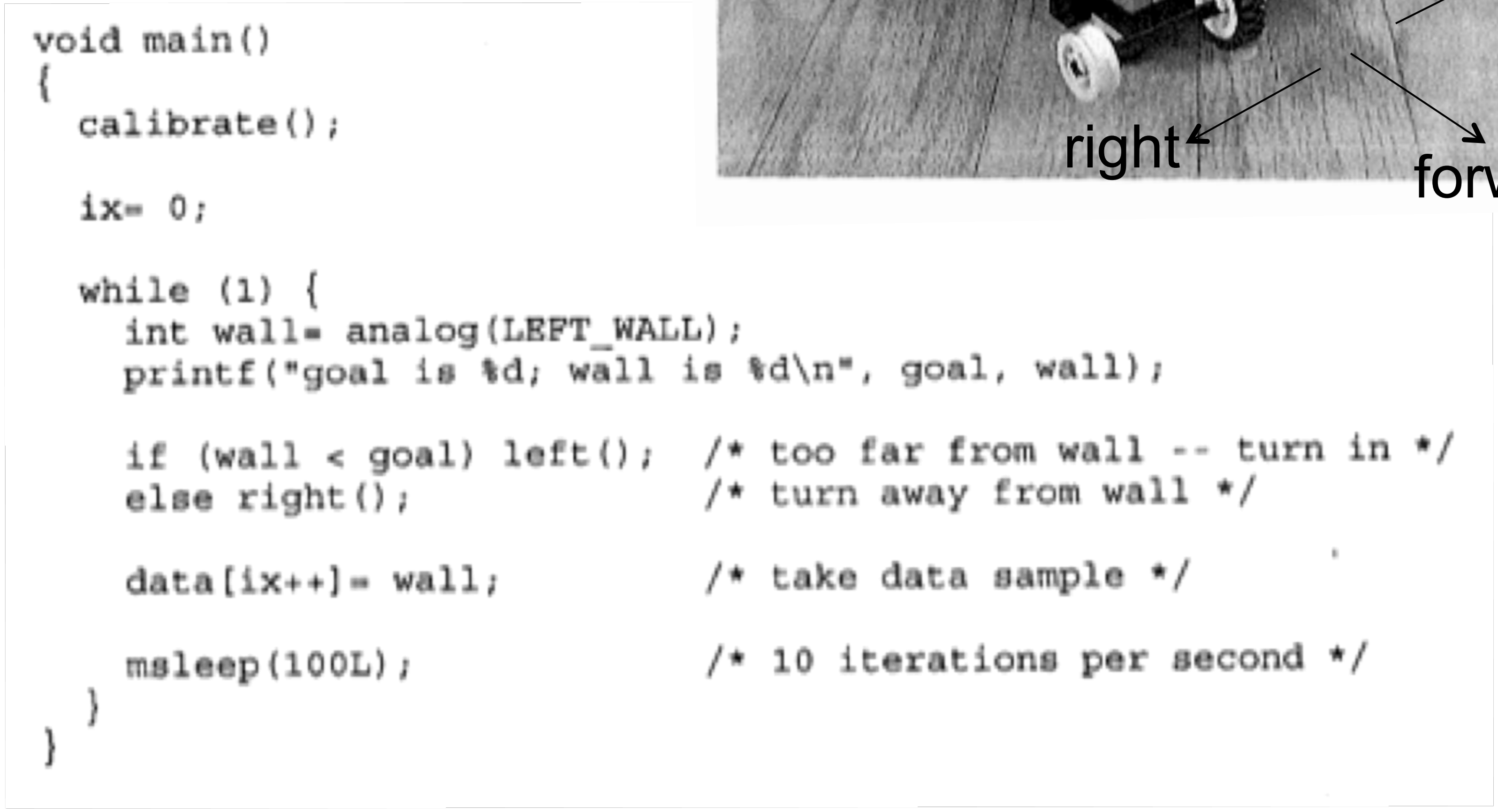


bend sensor value:

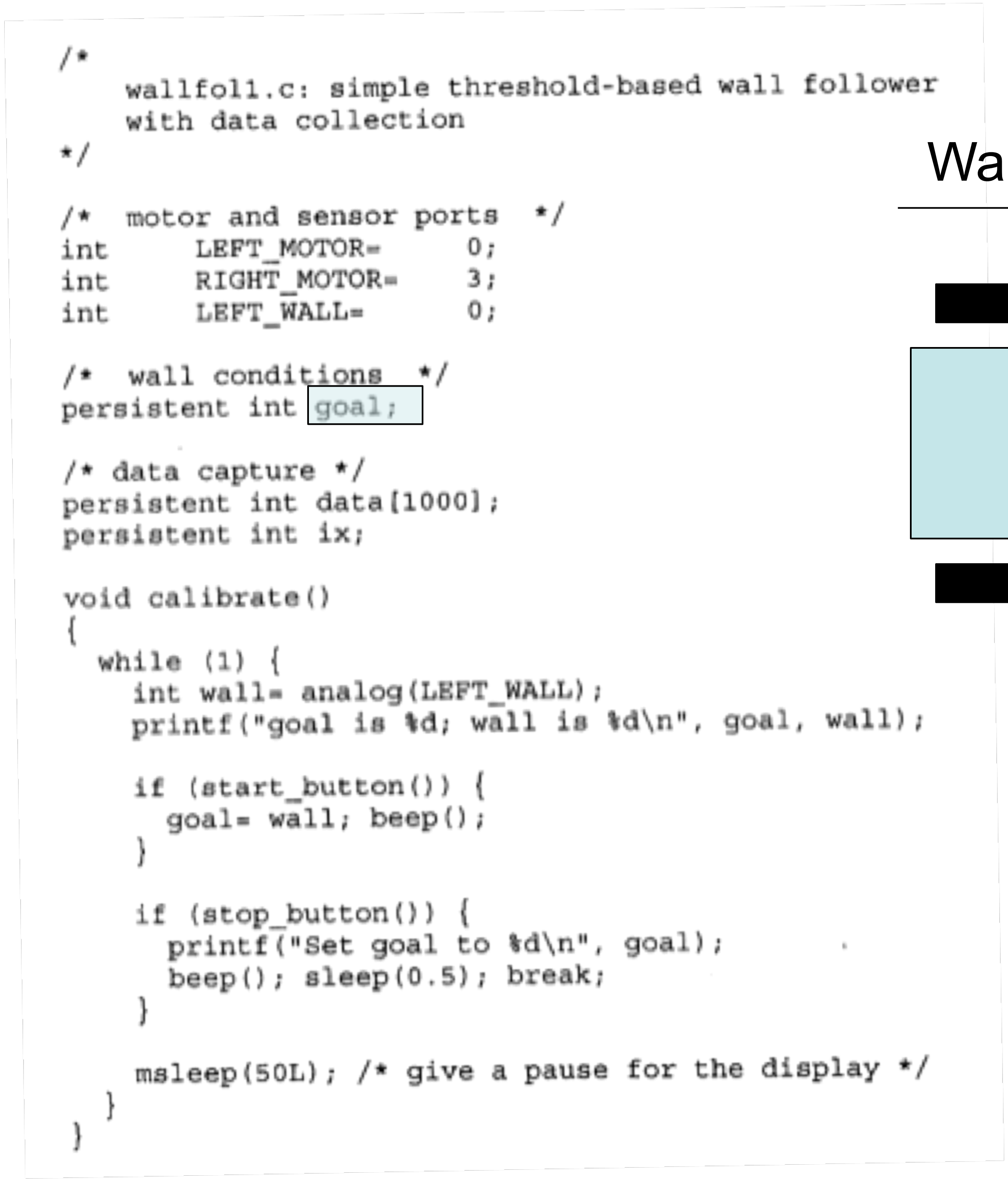
high value - close to wall

low value - away from wall

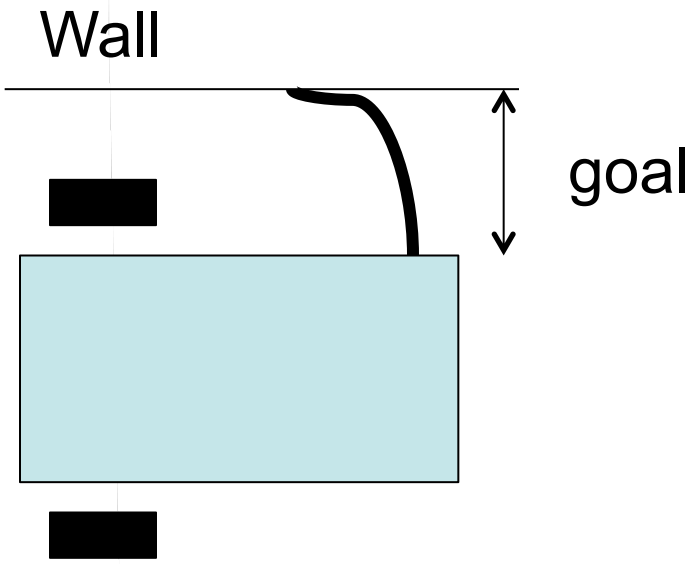


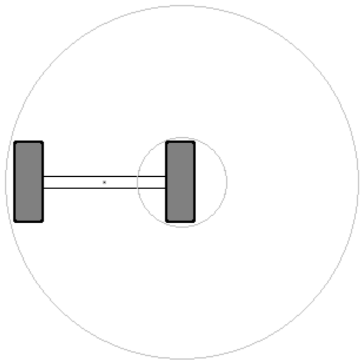


```
void main()
{
  calibrate();

  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 */
  }
}
```

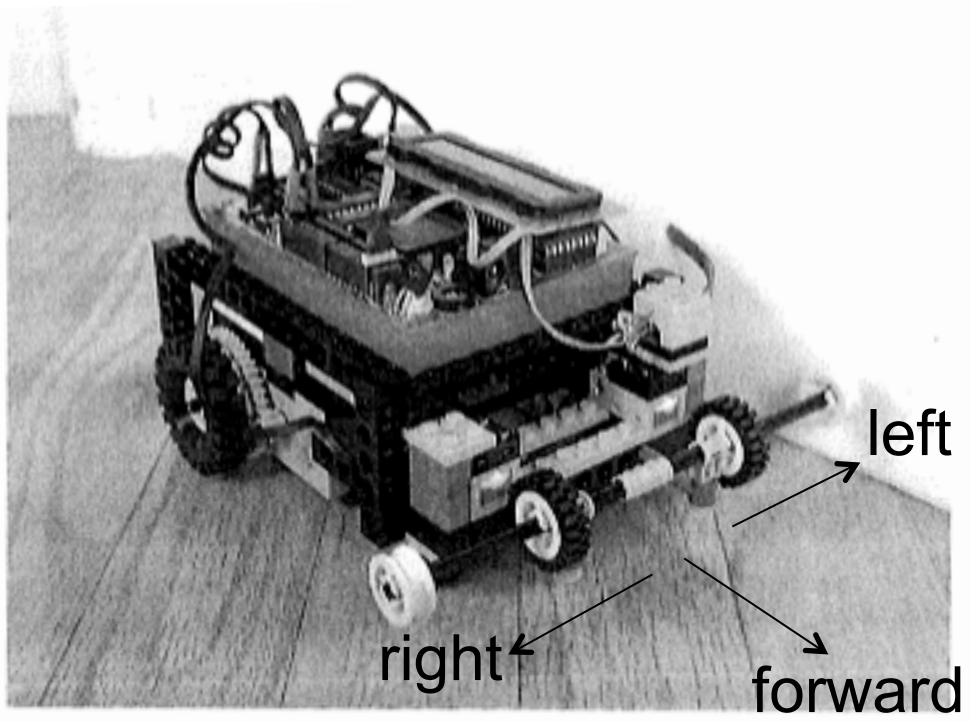


```
/*
    wallfol1.c: simple threshold-based wall follower
    with data collection
*/

/*  motor and sensor ports  */
int      LEFT_MOTOR=      0;
int      RIGHT_MOTOR=     3;
int      LEFT_WALL=       0;

/*  wall conditions  */
persistent int goal;

/* data capture */
persistent int data[1000];
persistent int ix;

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

    if (start_button()) {
      goal= wall; beep();
    }

    if (stop_button()) {
      printf("Set goal to %d\n", goal);
      beep(); sleep(0.5); break;
    }

    msleep(50L); /* give a pause for the display */
  }
}
```



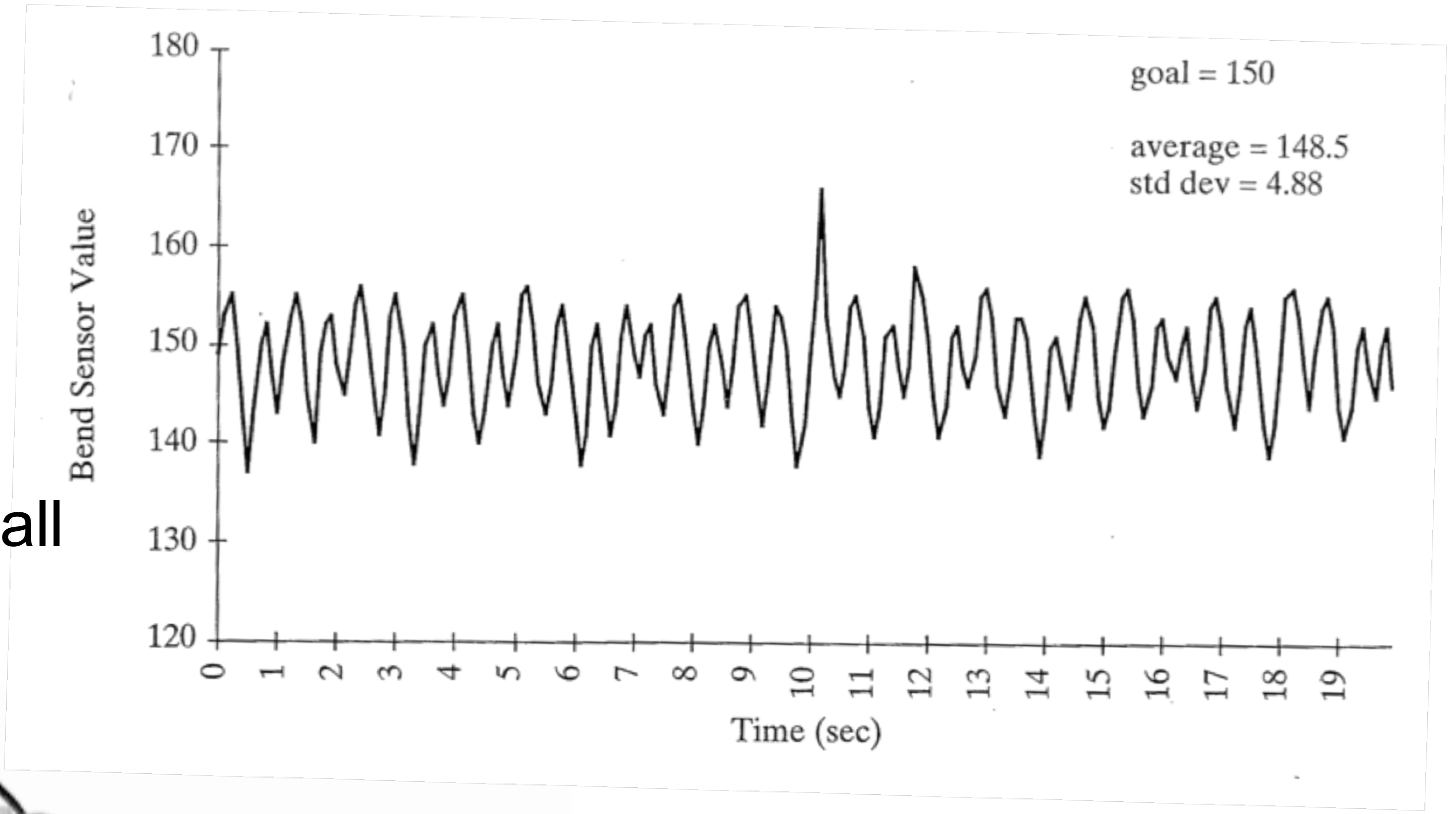


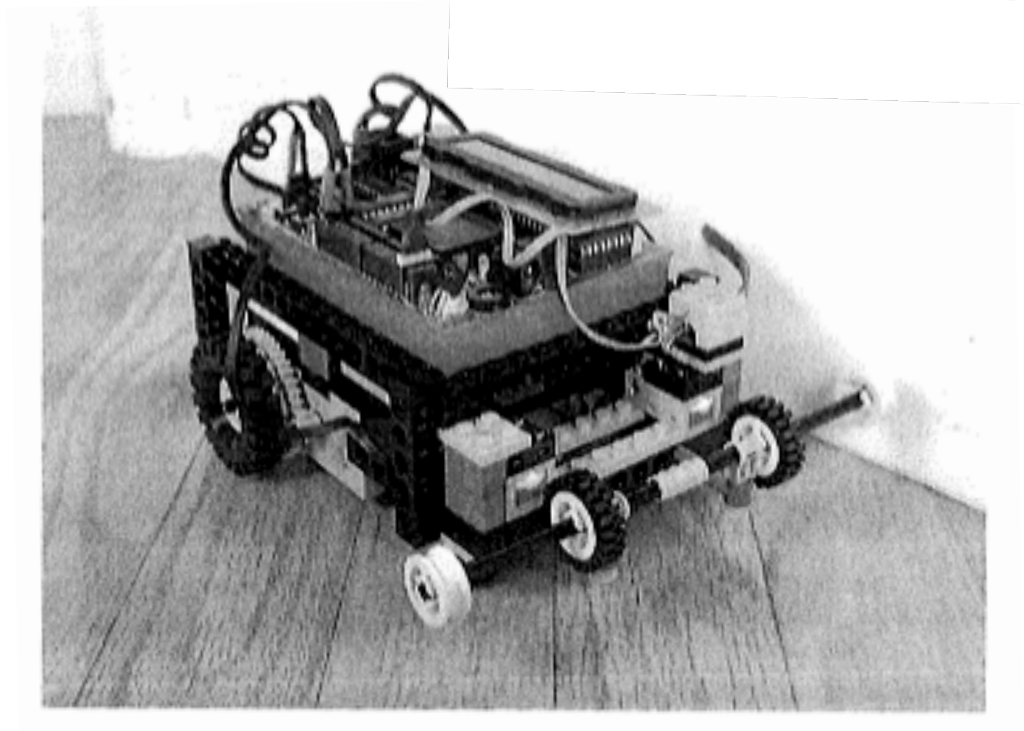
```
void left()
{
  motor(RIGHT_MOTOR, 100);
  motor(LEFT_MOTOR, 0);
}

void right()
{
  motor(LEFT_MOTOR, 100);
  motor(RIGHT_MOTOR, 0);
}
```

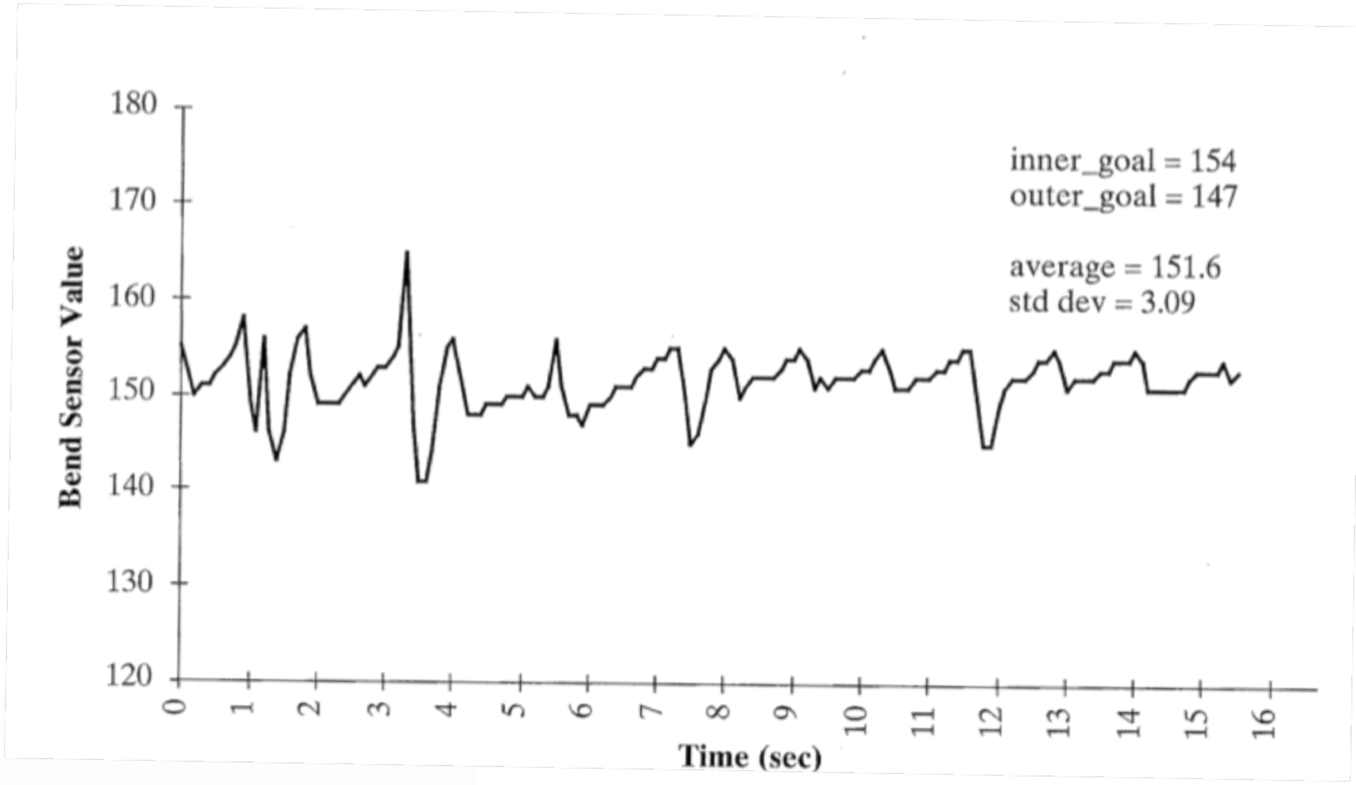


Close to wall

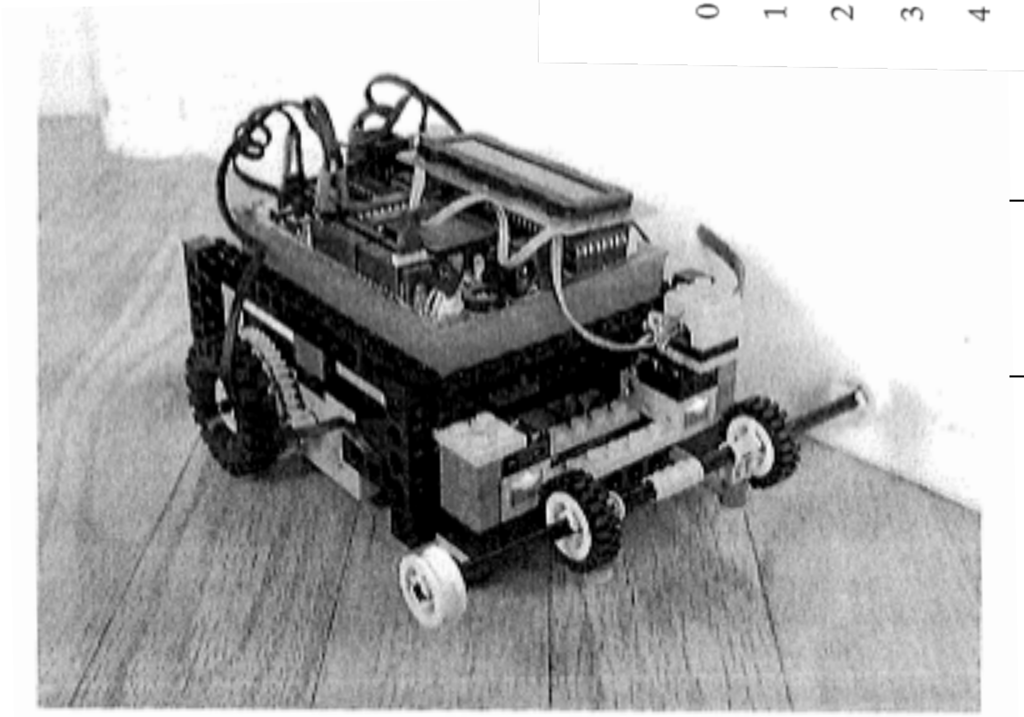
Away from wall

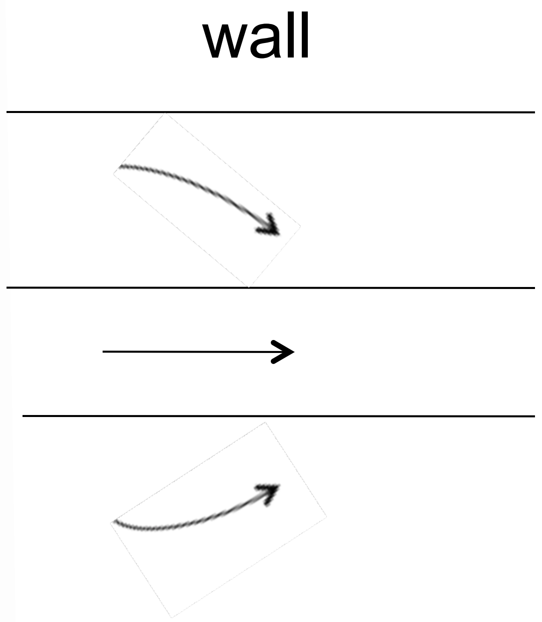






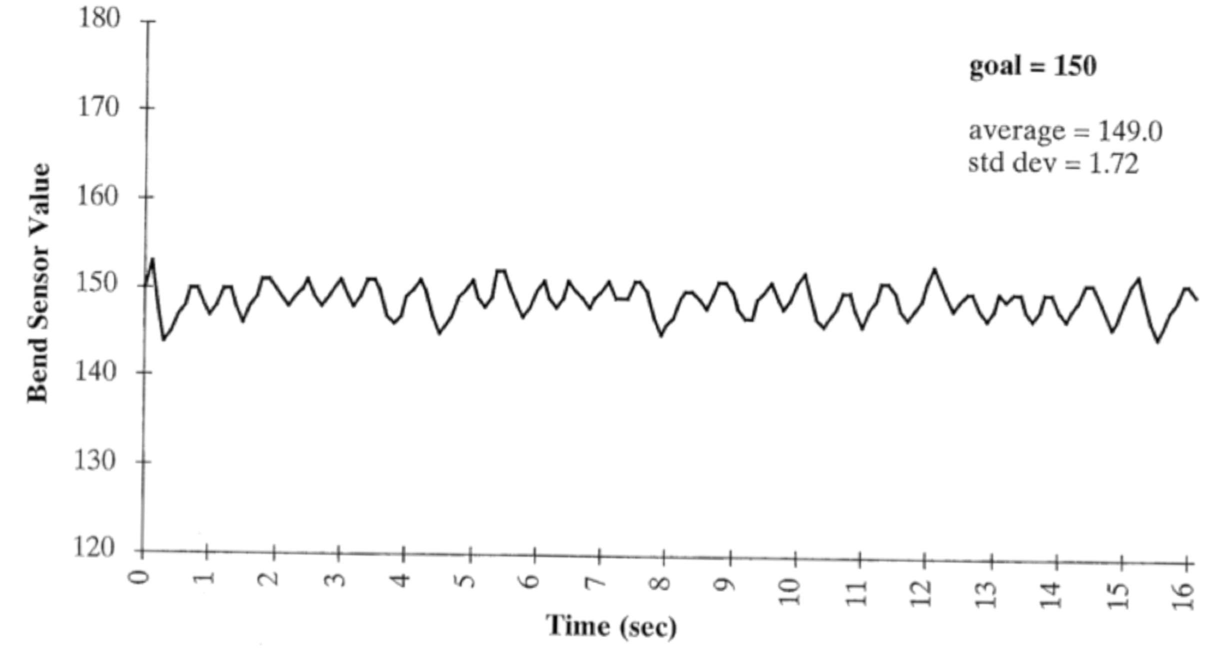
Bend Sensor Value
Time (sec)
inner_goal = 154
outer_goal = 147
average = 151.6
std dev = 3.09

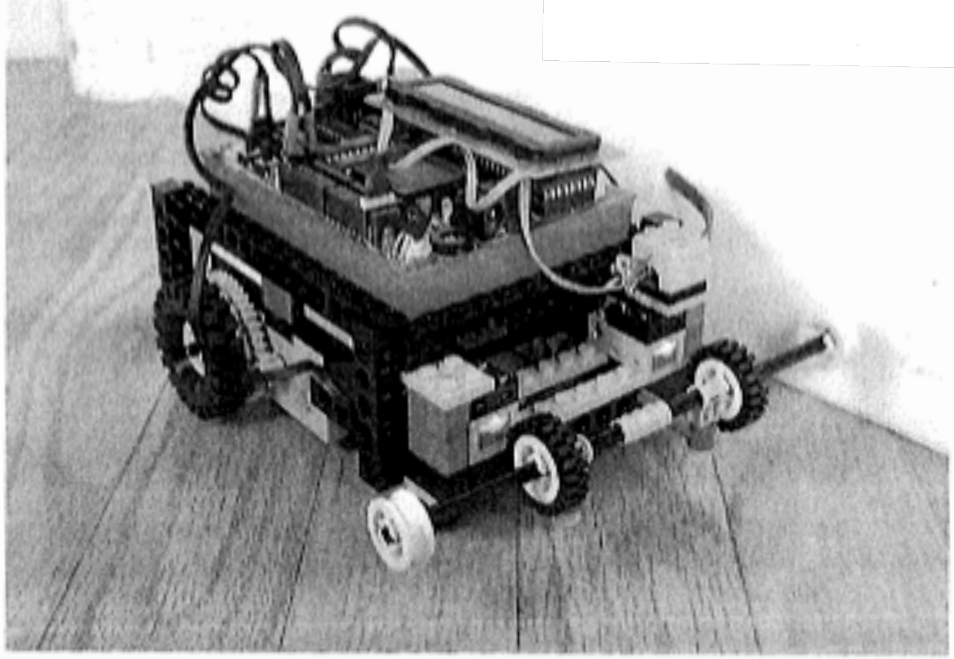


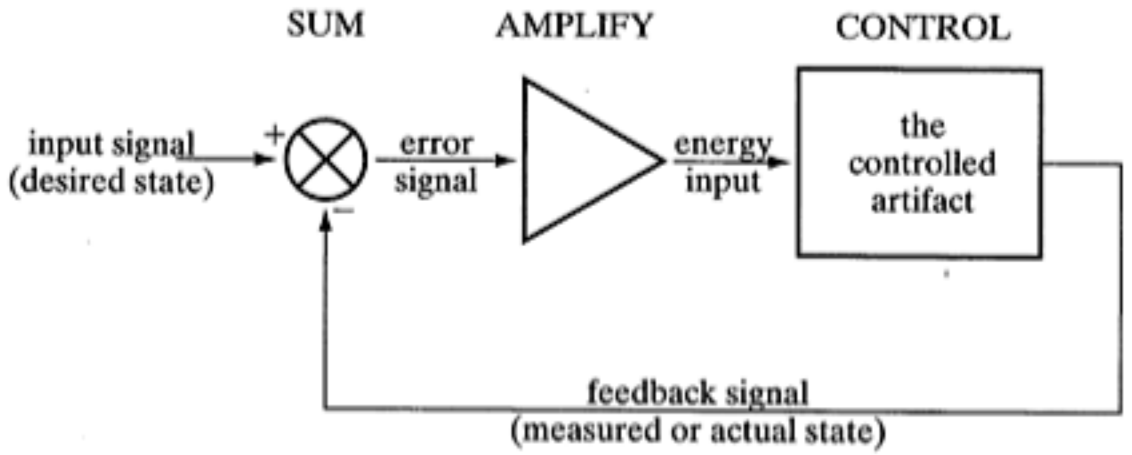


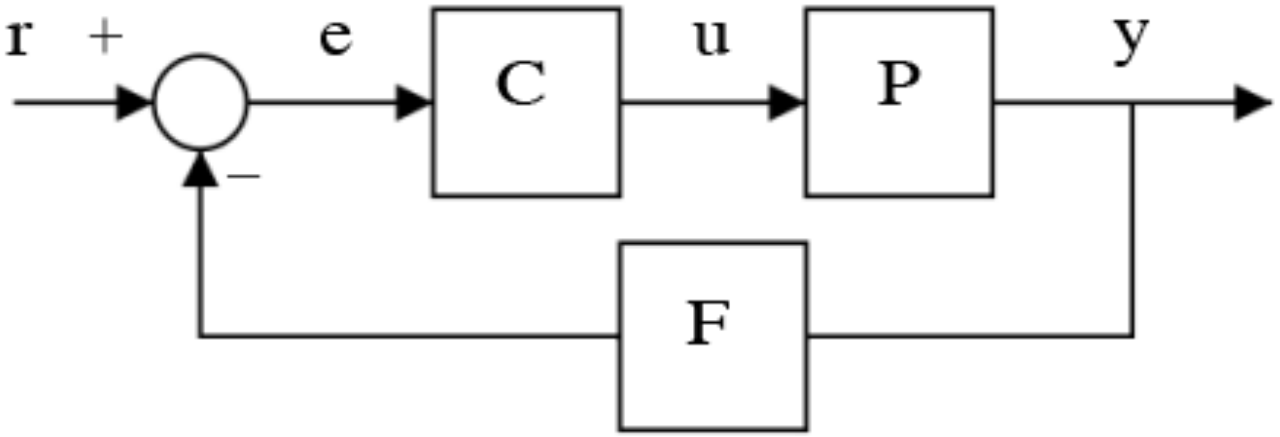
wall
inner
outer

```
void left()
{
  motor(RIGHT_MOTOR, 100);
  motor(LEFT_MOTOR, 50);
}

void right()
{
  motor(LEFT_MOTOR, 100);
  motor(RIGHT_MOTOR, 50);
}
```






SUM
AMPLIFY
CONTROL
input signal
(desired state)
+
−
error
signal
energy
input
the
controlled
artifact
feedback signal
(measured or actual state)



r	reference value
e	error
C	controller
u	output from controller to control P
P	plant, controlled system
y	state of P
F	feedback to controller

LEGO® MINDSTORMS™ NXT

LEJOS
Java for LEGO Mindstorms

LEGO mindstorms

- Home
- NXT Brick
  - leJOS NXJ
    - API
    - PC API
    - Tutorial
    - Downloads
- RCX Brick
  - leJOS RCX
    - API
    - Tutorial
    - Downloads
    - FAQ
- Forum
- Books
- Links
- Contact

SOURCEFORGE.net

java.net MEMBER

JVM NXT Brick, Icommand technology, ...

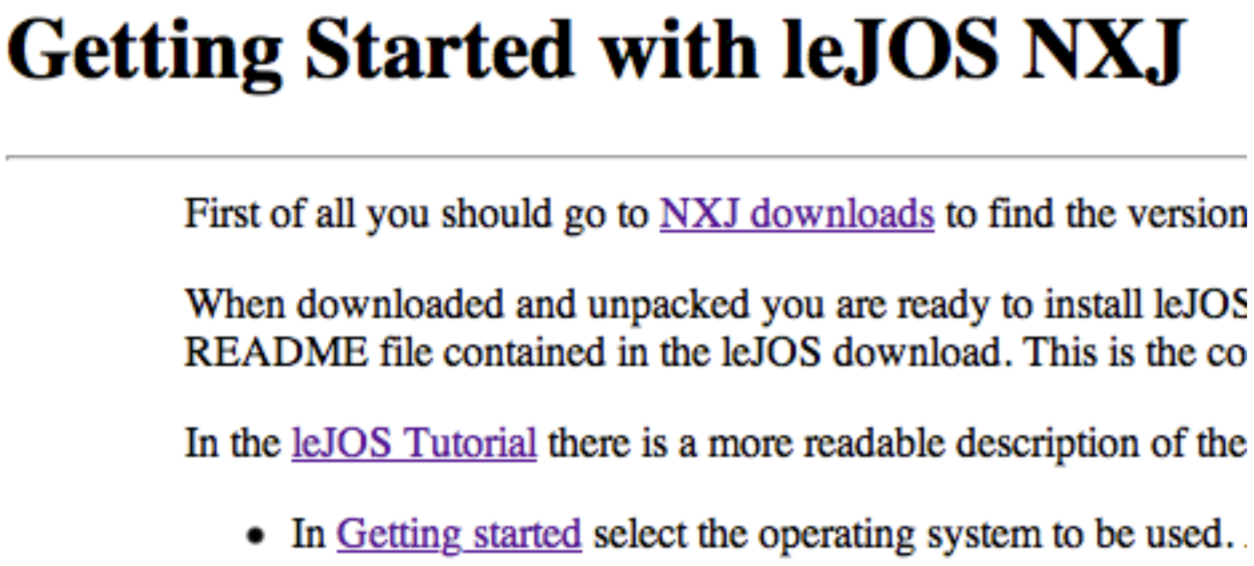
GO

JVM RCX Brick

GO

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.



Getting Started with leJOS NXJ

First of all you should go to NXJ downloads to find the version

When downloaded and unpacked you are ready to install leJOS
README file contained in the leJOS download. This is the co

In the leJOS Tutorial there is a more readable description of the

- In Getting started select the operating system to be used.

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.

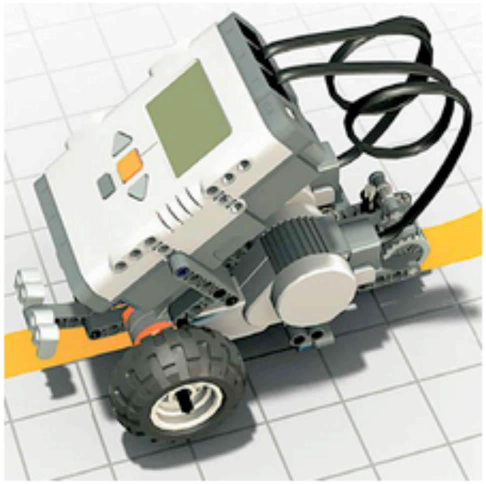


Figure 1 The 9797 LEGO car with two motors.

A Java Control Program: LineFollower

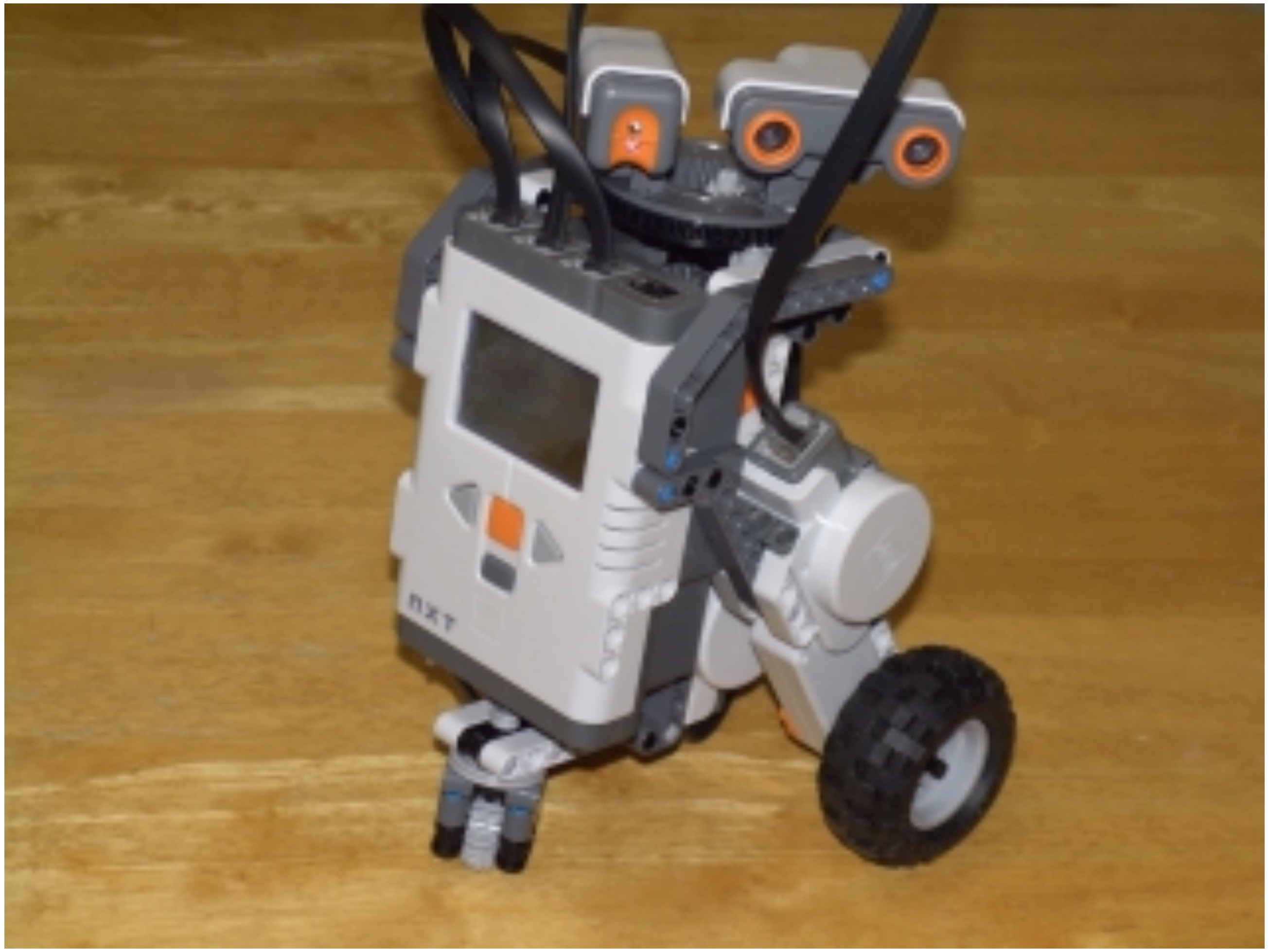
The first Java program that we are going to execute on the NXT is the following Java program that makes the LEGO car follow a black line on a white surface: (`LineFollower.java`):

Date:
Duration of activity:
Group members participating:

Furthermore, each activity should be described by:

- a **goal** (or goals) for this activity, maybe with a list of subgoals,
- a **plan** for the activity including a description of methods that lead to fulfillment of the goal(s),
- **results** obtained including descriptions of
  - experiments together with a description that other groups can use to reproduce your experiments,
  - programming attemts with program segments and links to programs,
  - output from programs,
  - measurements,
  - pictures of LEGO models,
  - problems encountered.
- a **conclusion** with a status and suggestions for what to do next.
- **references** to papers, web pages or copied material.

R2MeeToo



```
1  →   import lejos.nxt.*;
       import lejos.navigation.*;

       /**
        * Maximum LEGO NXT: Building Robots with Java Brains
        * ISBN-13: 9780973864915
        * Variant Press (C) 2007
        * 1st Edition - Chapter 2
        * R2MeToo robot
        * Platform: leJOS NXJ
        * @author Brian Bagnall
        * @version Sept-4-2007
        */
       public class R2MeToo {

2  →       static final float WHEEL_DIAM = 5.6F;
           static final float TRACK_W = 13F;
           static final int INTERVAL = 45;
           static final int SWEEP = 360 - 90;

3  →       static UltrasonicSensor us = new UltrasonicSensor(SensorPort.S1);
           static LightSensor ls = new LightSensor(SensorPort.S3, true);
           static Pilot sc = new Pilot(WHEEL_DIAM, TRACK_W, Motor.C, Motor.B, true);

           public static void main(String [] args) throws Exception {
               Motor.A.setSpeed(850);  // Head speed
               sc.setSpeed(700); // Movement speed
               sc.forward();

               while(!Button.ESCAPE.isPressed()) {

                   if(us.getDistance() < 45) {
                       sc.stop();
                       Sound.twoBeeps();
                       sc.rotate(getBestDir());
                       sc.forward();
                   }
                   Thread.sleep(200);
               }
           }

           // Rotate head and find longest direction
           public static int getBestDir() {
               ls.setFloodlight(true);
               int bestDir = 0;
               int bestDist = 0;
               for(int i=-SWEEP/2;i<SWEEP/2;i = i + INTERVAL) {
                   Motor.A.rotateTo(i * 9);  // 9 = gear ratio
                   int curDist = us.getDistance();
                   if(curDist > bestDist) bestDir = i;
                   if(curDist >  200) break;

               }
               Motor.A.rotateTo(0);
               ls.setFloodlight(false);
               return bestDir;
           }
       }
```

```java
import lejos.nxt.*;
import lejos.navigation.*;
```

```java
static final float WHEEL_DIAM = 5.6F;
static final float TRACK_W = 13F;
```

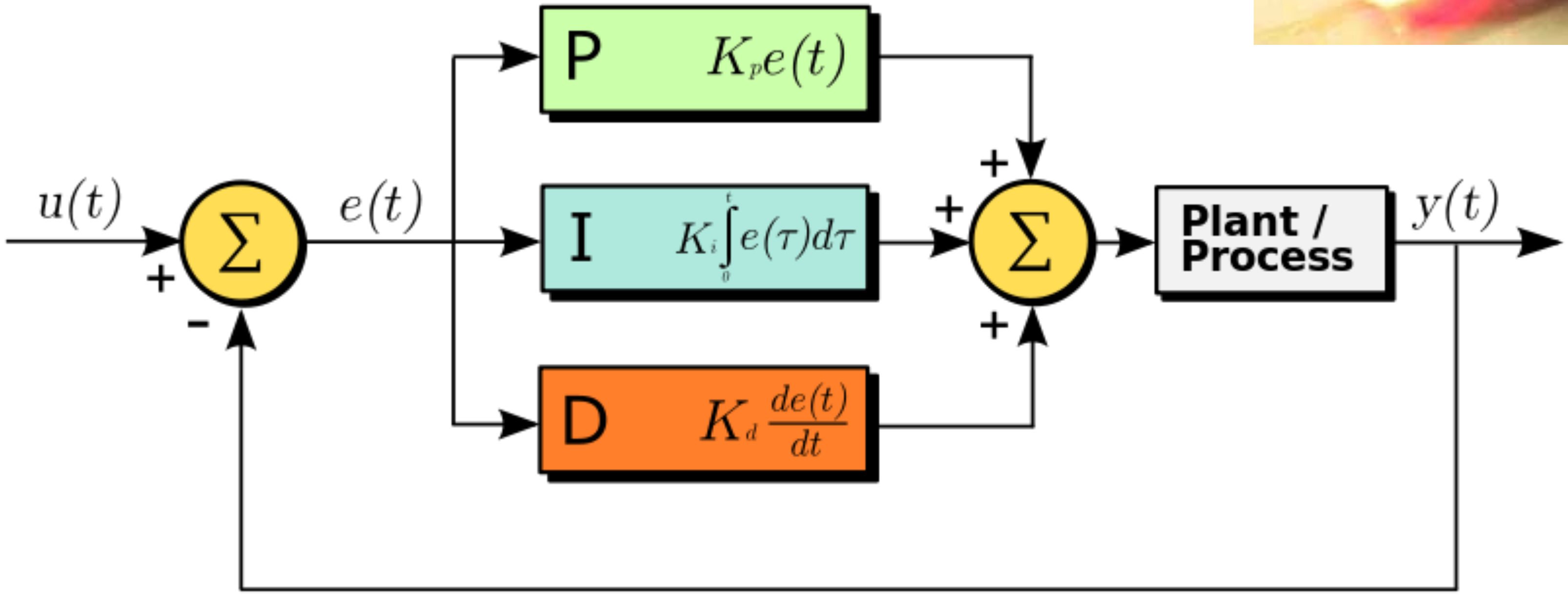
```java
static UltrasonicSensor us = new UltrasonicSensor(SensorPort.S1);
static LightSensor ls = new LightSensor(SensorPort.S3, true);
static Pilot sc = new Pilot(WHEEL_DIAM, TRACK_W, Motor.C, Motor.B, true);
```

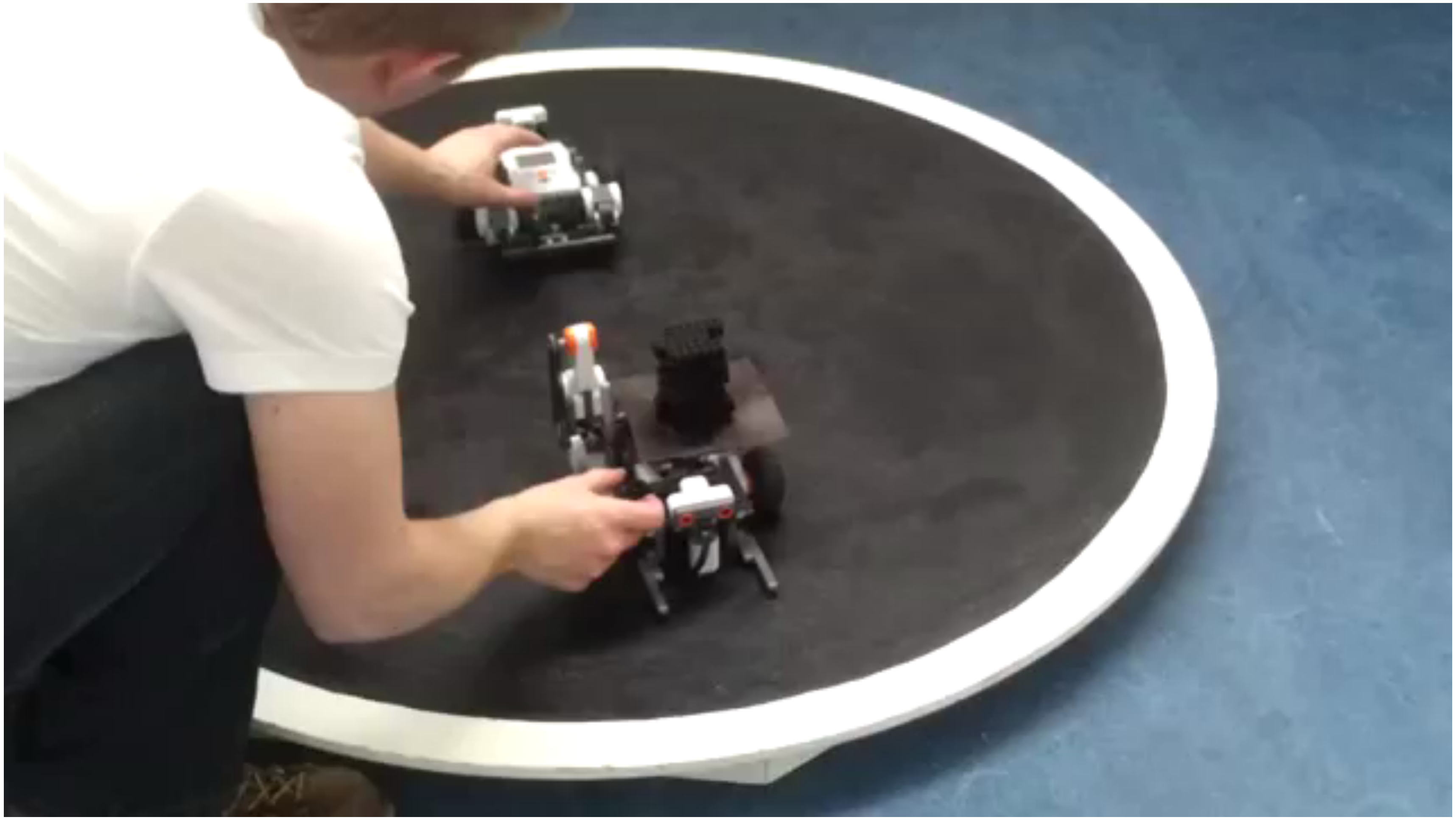
```
while(!Button.ESCAPE.isPressed()) {

    if(us.getDistance() < 45) {
        sc.stop();
        Sound.twoBeeps();
        sc.rotate(getBestDir());
        sc.forward();
    }
    Thread.sleep(200);
}
```

```
    // Rotate head and find longest direction
    public static int getBestDir() {
        ls.setFloodlight(true);
        int bestDir = 0;
        int bestDist = 0;
        for(int i=-SWEEP/2;i<SWEEP/2;i = i + INTERVAL) {
            Motor.A.rotateTo(i * 9); // 9 = gear ratio
            int curDist = us.getDistance();
            if(curDist > bestDist) bestDir = i;
            if(curDist >  200) break;

        }
        Motor.A.rotateTo(0);
        ls.setFloodlight(false);
        return bestDir;
    }
}
```

Lab session projects

PID controller





Plan for Q3

Week 5 + 6	Introduction and installation of leJOS
Week 7 to 10	Lectures and lab sessions
Week 11	Lecture and graded lab session