Kinematics, Odometry and Navigation, Differential Drive

Henning Christiansen

Adapted from slides by Mads Rosendahl, 2014; program examples revised by Ole Torp Lassen

Overview

- Week 1: How to prepare a program and have it executed on the brick
- Week 2 = now: Getting the brick to move
- Week 3–11: Getting the brick to do something interesting

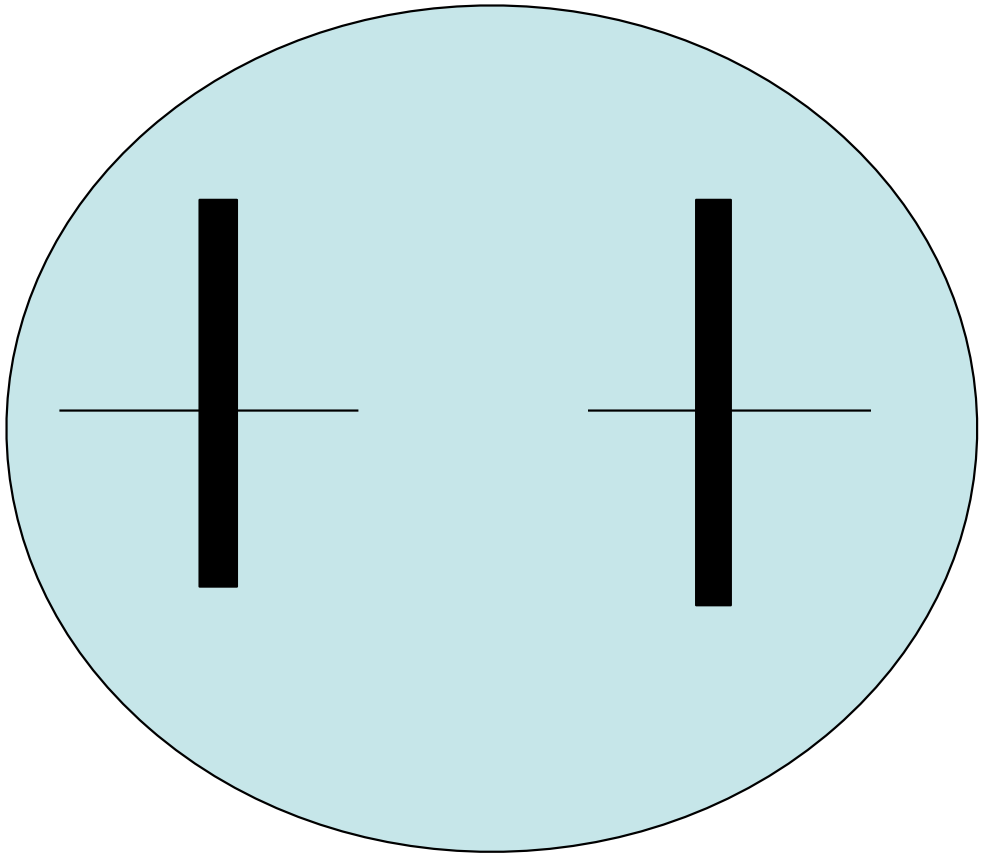
Program for today: getting the brick to move

- Differential drive
- Dead reckoning
- Navigation
- Lejos API
- Work with assignment 1

- Part of the game: Lejos is open source, changes from time to time, documentation not always the best. Only one way out: *test it and see what happens!!*

Differential drive

- Two independently powered wheels



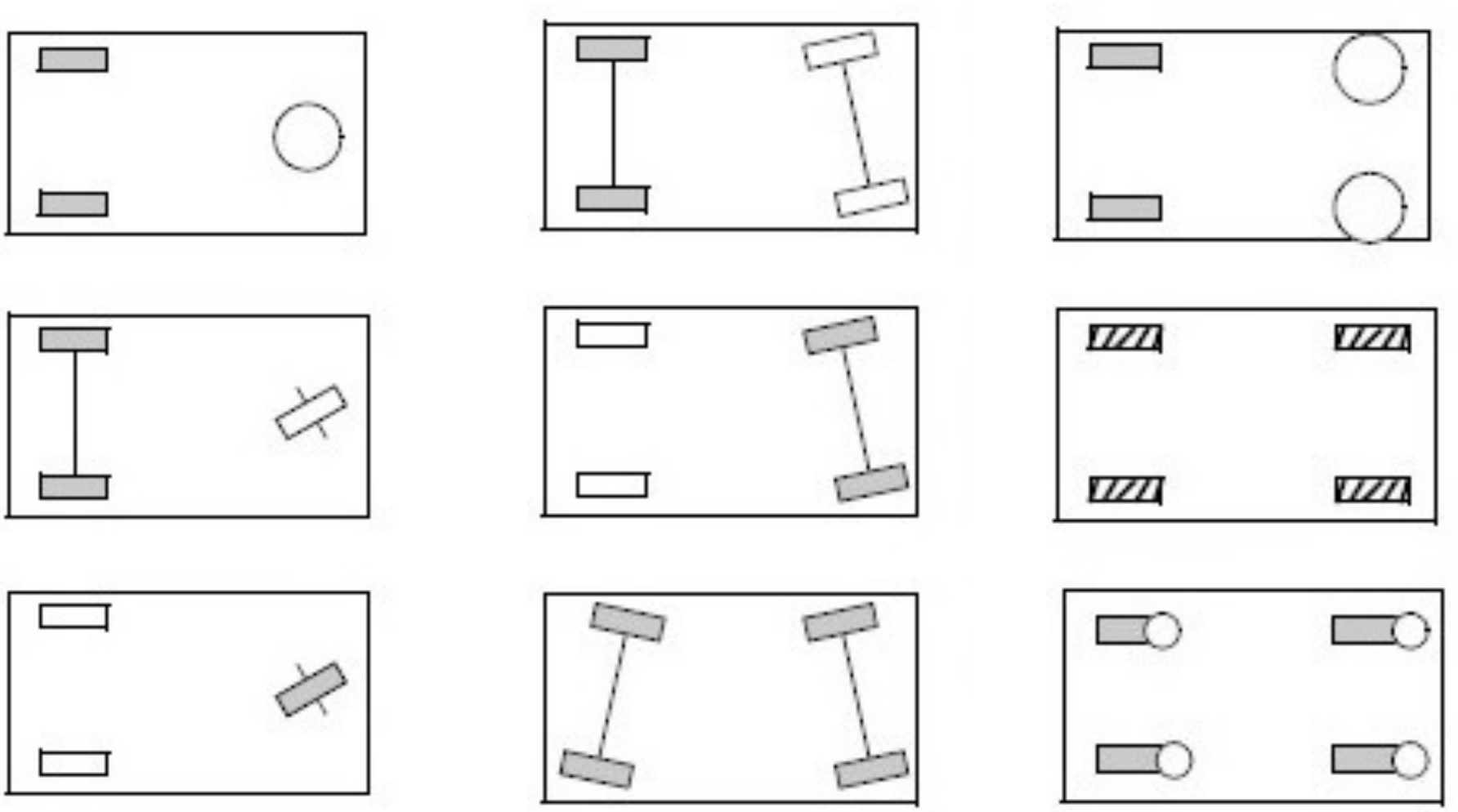
Other wheel configurations

- How many wheels for static stability?

- Front wheels steering, rear wheels for power?
- Three wheels or four wheels? (or two or one or *n*??)

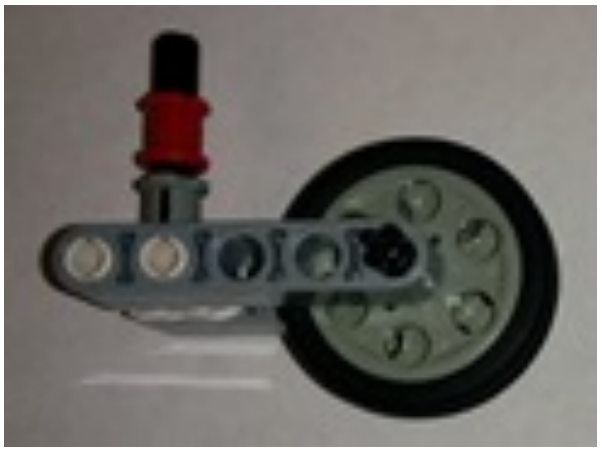
- Usual wheel configurations will require some wheels to slip (a little bit)
- Ackerman steering – discussed in the book, but we will skip it

Wheel configurations



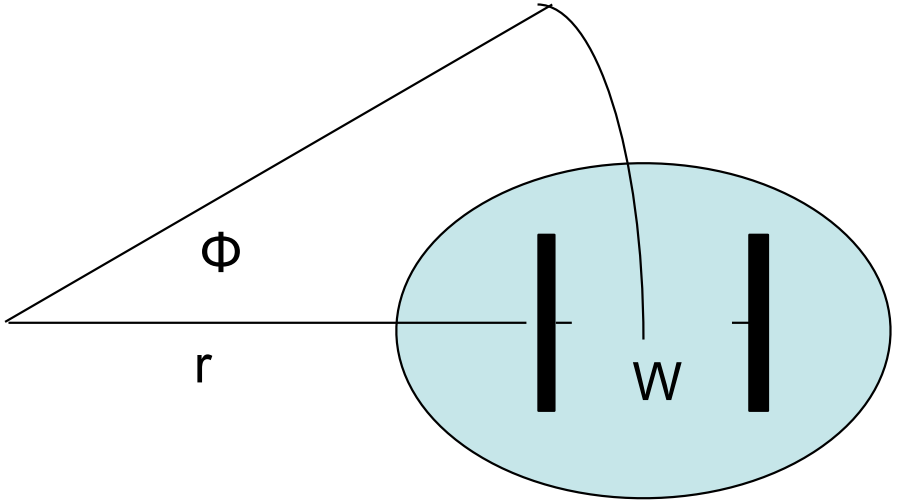
Caster wheel or contact point

Wheel should no be directly under axis



Contact point can slide over floor

Differential drive



w: wheel distance

d: wheel diameter

Φ: turning angle

ω_L, ω_T: speed left and right wheel (angle/time)

r: turning radius (inner wheel), time t

Relationship between the variables

$$r \cdot \Phi = d/2 \cdot \omega_L \cdot t$$

$$(r+w) \cdot \Phi = d/2 \cdot \omega_R \cdot t$$

Example

Given wheel distance w = 14cm, wheel diameter d = 55mm

Set inner wheel speed ω_L =180 degrees per second.

With time for turn: t = 2 seconds, turn Φ = 45 degrees, which speed to set for outer wheel, ω_R?

$\omega_R = \omega_L + 2 \cdot w \cdot \Phi/(d \cdot t) = 180 + 114 = 294$dps

Turn in 3 seconds: $\omega_R = 256$

Turn in 1 second: $\omega_R = 409$

Another example: one full circle of my robot:

(d:44mm, w: 11cm, ω_L = 180/s, t=10)

$\omega_R = 180+2 \cdot 11 \cdot 360/ (4.4 \cdot 10) = 360$dps

$r = d/2 \cdot 1 \cdot \omega_L \cdot t/\Phi = 11/2 \cdot 180 \cdot 10/360 = 27,5$

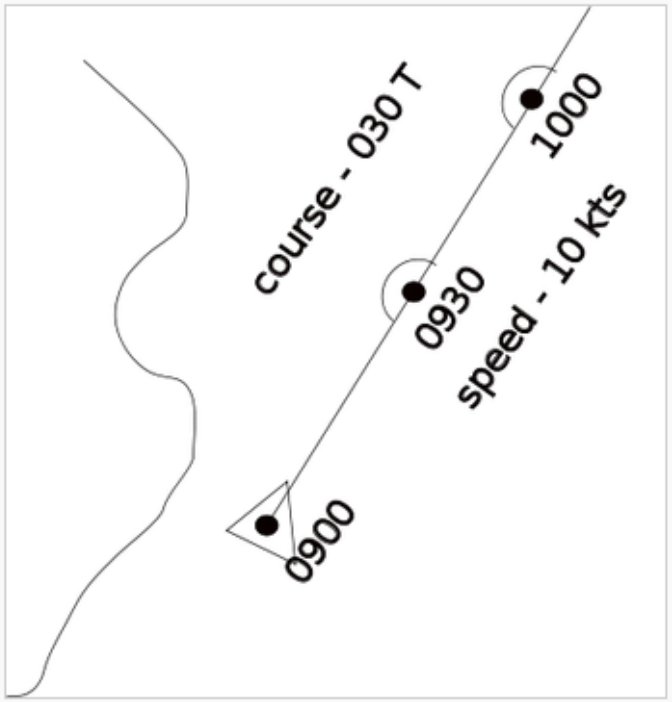
One circle of my robot

```
import lejos.hardware.motor.Motor;
public class TestCircle{
  public static void main(String[] args){
    Motor.A.setSpeed(180);
    Motor.B.setSpeed(360);
    Motor.A.forward();
    Motor.B.forward();
    try{Thread.sleep(10000);}catch(Exception e){}
    Motor.A.stop();
    Motor.B.stop();
  }
}
```

(we return to the magic class Motor later)

Dead reckoning – known from ship navigation (1:1)

Estimate your position by calculations from 1) *last known position*, 2) *speed* and 3) *heading* [think on being in a heavy fog]



The navigator plots their 9am position, indicated by the triangle, and, using their course and speed, estimates their own position at 9:30am and 10am.

- Errors considered in ship/ flight navigation:
  - drift (can be corrected for if estimation is known)

Illustration: https://en.wikipedia.org/wiki/Dead_reckoning

Dead reckoning – known from ship navigation (2:2)

- Useful for short distances, but errors accumulate.
- Some internal errors:
  - Latency, delay due to acceleration
  - Saturation, max speed reached
  - Backlash difference between motor/wheel move

Odometry

- *"Odometry is the use of data from motion sensors to estimate change in position over time"*
- LeJOS supports this, enhanced by the motors' built-in *tachometer*
- Tachometer ?????
  - A motor is also sensor
  - You can ask it for how many steps (degrees, ... whatever) it actually did rotate
    - **`Motor.A.getTachoCount()`**

Classes that make the robot move – at different levels of abstraction

The most primitive level: class Motor

- `Motor.A.setSpeed(),Motor.B.start(), Motor.C.stop(),Motor.A.getTachoCount()`

Steering: class MovePilot

- Does all the calculations of speeds etc. for the different wheels
  - `.setAngularSpeed(45), .setLinearSpeed(20),`
  - `.travel(100), etc.`
- To do that, it requires definitions of a "Chassis" (details later)

The ultimate odometry level: class Navigator

The Motor class

http://www.lejos.org/ev3/docs/lejos/hardware/motor/Motor.html:

- Motor class contains 3 instances of regulated motors.

Field Summary

Fields

Modifier and Type	Field and Description
`static NXTRegulatedMotor`	**A** Motor A.
`static NXTRegulatedMotor`	**B** Motor B.
`static NXTRegulatedMotor`	**C** Motor C.
`static NXTRegulatedMotor`	**D** Motor D.

- Not documented, but it may be hypothesized that it check at runtime, which actual motors are attached

MovePilot. Part 1: defining it

```
import ...  (see all imported lib's in source code on moodle);

public class TestPilot{
  public static void main(String[] args){
    Wheel wheelL = WheeledChassis.modelWheel
                                (Motor.A, 43.2).offset(-51);
    Wheel wheelR = WheeledChassis.modelWheel
                                (Motor.B, 43.2).offset(51);

    Chassis chassis = new WheeledChassis
                (new Wheel[] { wheelL, wheelR },
                 WheeledChassis.TYPE_DIFFERENTIAL);

    MovePilot pilot = new MovePilot(chassis);
```

(continued)

MovePilot. Part 2: using it

```
    pilot.setAngularSpeed(45);  // degrees per sec
    pilot.setLinearSpeed(20);   // cm per sec
    pilot.travel(100);          // cm
    pilot.rotate(-30);          // degrees
    pilot.travel(-100);         // move backward
    pilot.arc(100,40);          // radius, degree
    pilot.travel(-80);          // move backward
    pilot.rotate(-13);          // degree
    pilot.stop();
  }
}
```

Warnings

- Distance units sometimes mm and sometimes cm (sic!!!)
  - (do not always trust API nor your teachers' examples)
- Angles in degrees (and Java's Math. class uses radians!!!)

Navigator. Part 1: defining it

```
import ...  (see all imported lib's in source code on moodle);

public class TestNavigator{
  public static void main(String[] args){
    Wheel wheelL = WheeledChassis.modelWheel(Motor.A, 43.2).offset(-51);
    Wheel wheelR = WheeledChassis.modelWheel(Motor.B, 43.2).offset(51);
    Chassis chassis = new WheeledChassis(new Wheel[] { wheelL, wheelR },
                                         WheeledChassis.TYPE_DIFFERENTIAL);
    MovePilot pilot = new MovePilot(chassis);
    pilot.setAngularSpeed(450);// degree per sec
    pilot.setLinearSpeed(40);  // mm per sec

    Navigator robot = new Navigator(pilot);
```

Navigator. Part 2: using it

```
...

    Navigator robot = new Navigator(pilot);


    robot.goTo(100,100);
    robot.goTo(0,0);
    while(robot.isMoving()) Sound.pause(500);
    robot.rotateTo(0);
}   }
```

Notes:

1. class Navigator can do a lot more; see http://www.lejos.org/ev3/docs/lejos/robotics/navigation/Navigator.html
2. Documentation does not say anything about how the coordinate system is defined. Most likely guess:
   According to robots location and heading when **`new Navigator`** is done

NB: a Motor is a RegulatedMotor

```
import lejos.hardware.motor.Motor;
public class TestMotor2{
  public static void main(String[] args){
    Motor.A.setAcceleration(1000);
    Motor.B.setAcceleration(1000);
    Motor.A.setSpeed(180);
    Motor.B.setSpeed(180);
    Motor.A.forward();  Motor.B.forward();
    try{Thread.sleep(1000);}catch(Exception e){}
    Motor.A.stop();     Motor.B.stop();
    Motor.A.rotate(10); Motor.B.rotate(10);
    try{Thread.sleep(1000);}catch(Exception e){}
    Motor.A.rotateTo(Motor.B.getTachoCount());
 }
}
```

Now a quick tour of useful facilites in the LeJos API

- Sound (for both testing and fun)
- Access the buttons on the EV3 brick
- Sensors

- We may not go through all; read slides afterwards and check API when you need it

Sound

```
import lejos.hardware.Sound;

public class TestSound{

  public static void main(String[] args){
    Sound.beepSequence();
    Sound.setVolume(50);
    Sound.pause(1000);
    int[] scale ={440,494,523,587,659,698,783};
    for(int i=0;i<50;i++){
      Sound.playTone(scale[random(scale.length)],500);
      Sound.pause(300); }
    Sound.setVolume(10);
  }

  static int random(int x){
    return (int) (Math.random()*x); }
}
```

Button

```
import lejos.hardware.*;

public class TestButton{
  public static void main(String[] args){
    Button.waitForAnyPress();
    Sound.beep();
    while(!Button.ESCAPE.isDown()){
      if(Button.RIGHT.isDown())
            Sound.beepSequenceUp();
      if(Button.LEFT.isDown())
            Sound.beepSequence();
      Sound.pause(300);
    }
  }
}
```

TouchSensor

```
import lejos.hardware.Button;
import lejos.hardware.Sound;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.EV3TouchSensor;
import lejos.robotics.SampleProvider;

public class TestTouch{
  public static void main(String[] args){

    EV3TouchSensor touch = new EV3TouchSensor(SensorPort.S1);
    SampleProvider touched = touch.getTouchMode();
    float[] sample = new float[touched.sampleSize()];

    Sound.setVolume(50);

    while(!Button.ESCAPE.isDown()){
          touched.fetchSample(sample,0);
          int t = (int) sample[0];
          if(t == 1)Sound.beep();
          Sound.pause(300);}
    Sound.setVolume(10);
  }
}
```

Infrared sensor

```
import lejos.hardware.*;
import lejos.hardware.port.SensorPort;
import lejos.hardware.sensor.EV3IRSensor;
import lejos.robotics.SampleProvider;

public class TestIR{
  public static void main(String[] args){
    EV3IRSensor infraRed = new EV3IRSensor(SensorPort.S2);

    SampleProvider IRdistance = infraRed.getMode("Distance");
      Sound.setVolume(50);
      float [] sample = new float[IRdistance.sampleSize()] ;

      while(!Button.ESCAPE.isDown()){
        IRdistance.fetchSample(sample,0);
        int d = (int) sample[0];
        if(d<100) Sound.playTone(2000-d*20,200);
        Sound.pause(100);
        System.out.println("dist "+d); }
      Sound.setVolume(10);
  }
}
```

File system

```
try {
   File f= new File("MyFile1.txt");
   f.createNewFile();
   BufferedWriter out = new BufferedWriter(
           new OutputStreamWriter(new FileOutputStream(f)));


    out.write("Hello World",0,11);
    out.newLine();
    out.write("and more Hello",0,14);
    out.flush();
    out.close();


  } catch(IOException e) {}
```

Wait

```
Button.waitForAnyPress()
   //Wait for any button to be pressed


try{Thread.sleep(1000);}catch(Exception e){ }
   //wait 1000 milliseconds (or 1 second)
   //a call to sleep can be interupted from another thread


while(!Button.ENTER.isPressed()){
   // or ESCAPE, LEFT or RIGHT
   try{Thread.sleep(1000);}catch(Exception e){ }
}

lejos.util.Delay.msDelay(1000);
   //standard utility function

Sound.pause(500);
```

The rest of the day

Work with exercises (see moodle for details)

- experiment with the TestMotors2 program
- first assignment
  - Get a robot to complete a track with as much precision as possible.
  - Consider what can be done at software level to improve precision.
  - deadline September 20, 2016.