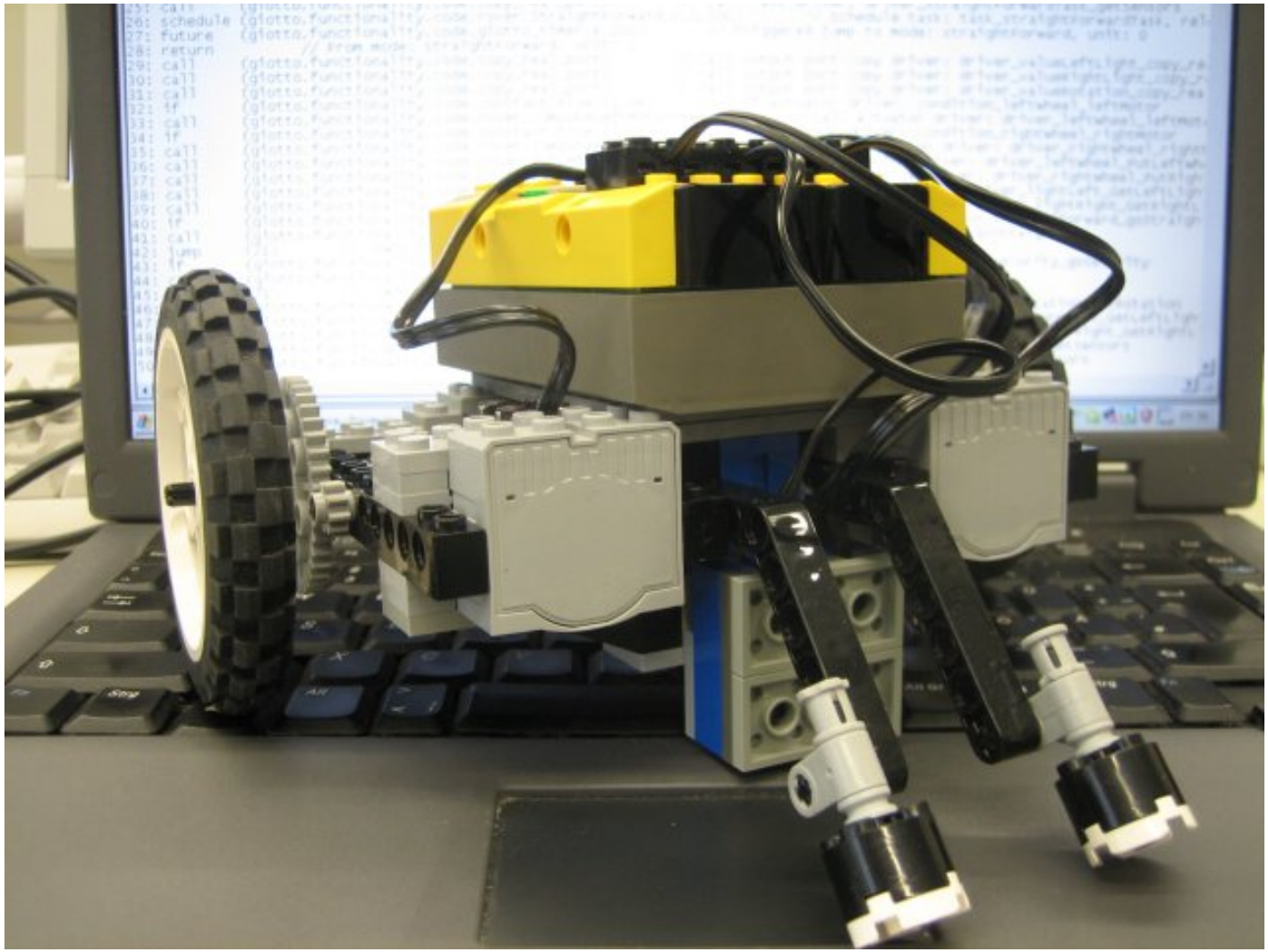
ROVER

Bernhard Mühlbacher, Hannes Payer

University of Salzburg

February 8, 2007



Ideas

- Java on Lego Mindstorm
- leJOS firmware on Lego Mindstorms RCX brick
- Direction invariant pathfinder
- Communication
- Following master rover
- Giotto on leJOS

Hardware

- Lightsensor
- Rotationsensor
- Differential
- infrared device
- smart steering
- Motor
- RCX

Problems

- Motor:
  - different speed (left, right) => Solution: differential gear
  - different speed of two rovers => Solution: something like ultrasonic
  - speed depends on battery state
  - => unable to keep 2 RCX's synchronized
- Lightsensor: light intensity
- 32kb memory

leJOS

- `http://lejos.sourceforge.net`
- leJOS is replacement firmware for the Lego Mindstorms RCX brick - a JVM that fits within the 32kb on the RCX.
- slim java API
- easy to handle
- Eclipse PlugIn

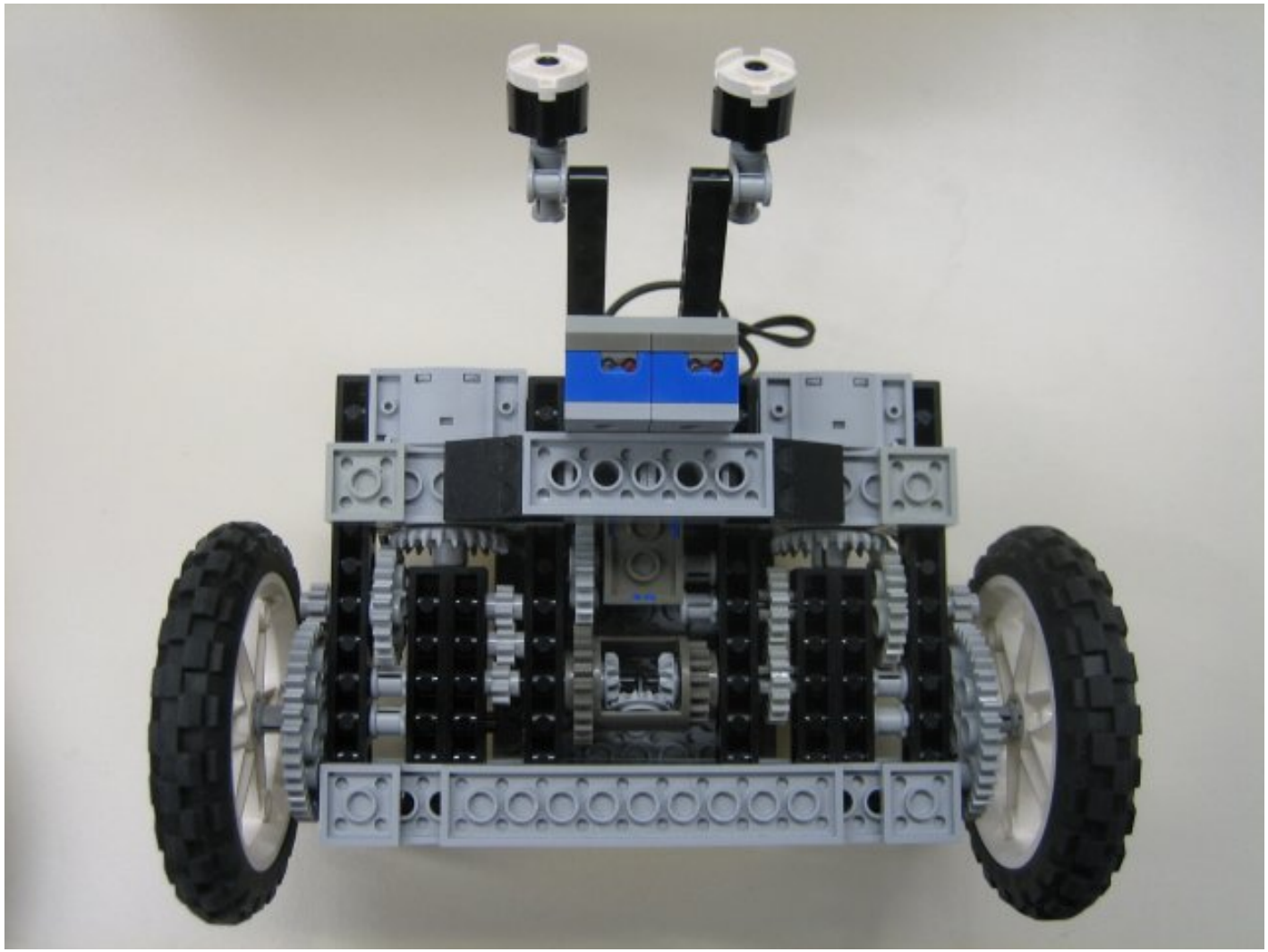
Problems

- no fileIO
- LCD display bug
- too slim java API (string concatenation, limited type casts, no generics, no reflection, ...)
- motor speed configurable with integers (0-7)
- less documentation

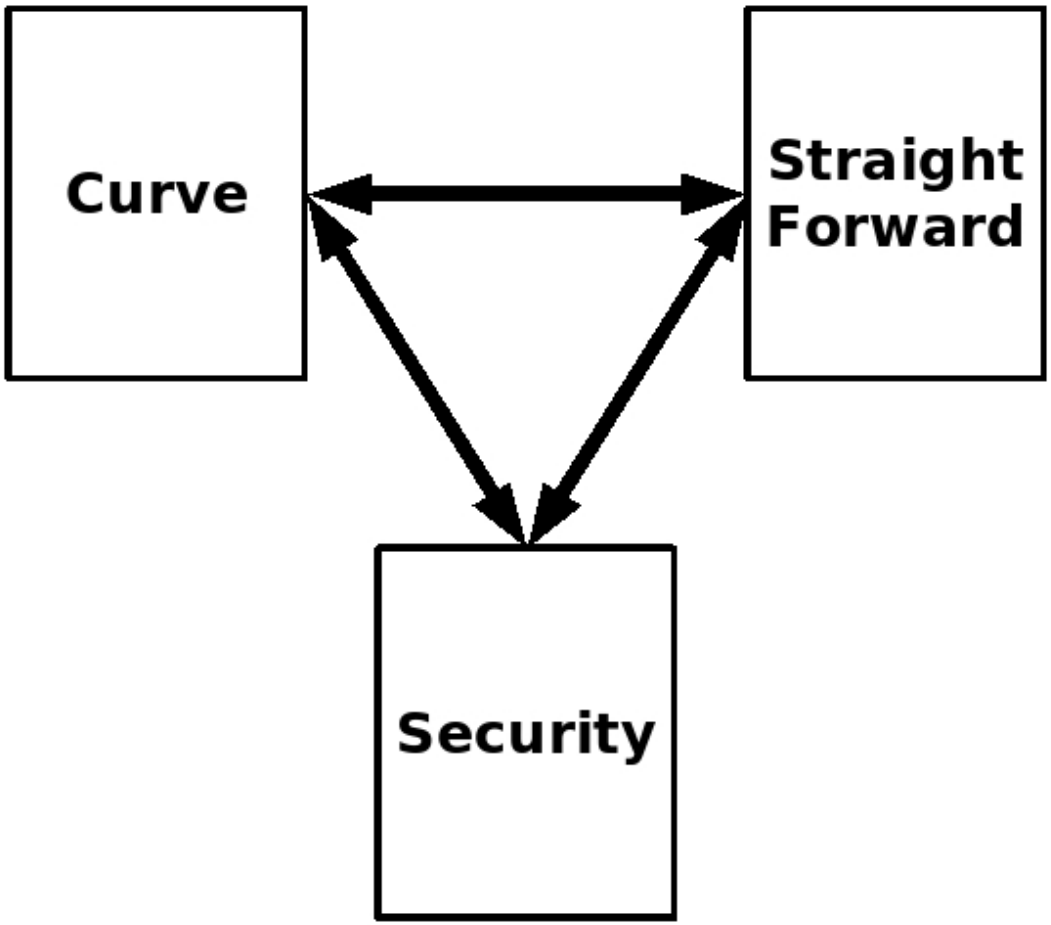
Features

- Direction invariant pathfinder
- Different speed by different color
- Communication to 2. RCX for debugging
- Giotto on RCX

Project-Description



tasks



tasks

- Straight Forward Task
  - precondition: both light sensors have to be on the line
  - motors run with same speed
  - use the rotation sensor to correct motor-speed differences
- Curve Task
  - precondition: only one light sensor is on the line
  - one motor runs with normal speed, the other one with slower speed
- Security Task
  - precondition: both light sensors are not on the line
  - use circular buffers to analyze the history

color modes

Color Ranges:

- black: sensor values $[36, 43]$
- green: sensor values $[43, 48]$

Difficulties:

- intervalls overlap
- color value depends on the room light

Implementations

- leJOS implementation
- Rover-Giotto
- Rover-VM

Toolchain

1. giotto program
2. giotto compiler
3. ECode
4. giotto-VM

Problems with giotto-VM

- less memory space
- no fileIO
- not able to load Serialized Objects
- runnable not supported
- and for sure many more

Solution: Rover-Giotto and Rover-VM

1. giotto program
2. giotto compiler extention
3. generate javafiles using templates
4. run genarated javafiles on customized giotto-rover-VM

Dynamic Class Loading

- Giotto
  - Compiler result are Serialized Objects
  - Serialized Objects are read via ObjectInputStream
  - dynamic loading of program functionality
- Rover-Giotto
  - no File IO
  - => we need a final vm image
  - => generate a handler for dynamic parts (ecode, operations, ports)

giotto compiler extention

- vision:
  - run the "show Ecode" result on a smart interpreter
  - dynamic solution
- generate OperationHandler, PortHandler, ECode

giotto demo

- giotto program
- giotto code
- templates

Rover -VM

- interpreter uses:
  - ECode.class to execute commands
  - OperationHandler.class to load registered operation objects
  - PortHandler.class to load registered port objects
- operations call the leJOS-functions

Interpreter

Interpreter Main Loop

```
Vector instruction;

while(pc < limit){
   instruction = eCode.getInstruction(pc);
   op = VMUtils.stringToInt((String)instruction.elementAt(0));

   if(op==0){ //nop
      pc++;
   }
   else if(op==1 || op==2 || op==3 || op==4){//operations
      operation = (Operation)oHandler.
                        getOperation((String)instruction.elementAt(1));
      pc = operation.op(pc, instruction);
   }
   else if(op==5){//jump
      pc = VMUtils.stringToInt((String)instruction.elementAt(1));
   }
   else if (op==6){//return
   }
}
```

Demo