Final Presentation

Hardware/Software Co-Design with a Lego Car

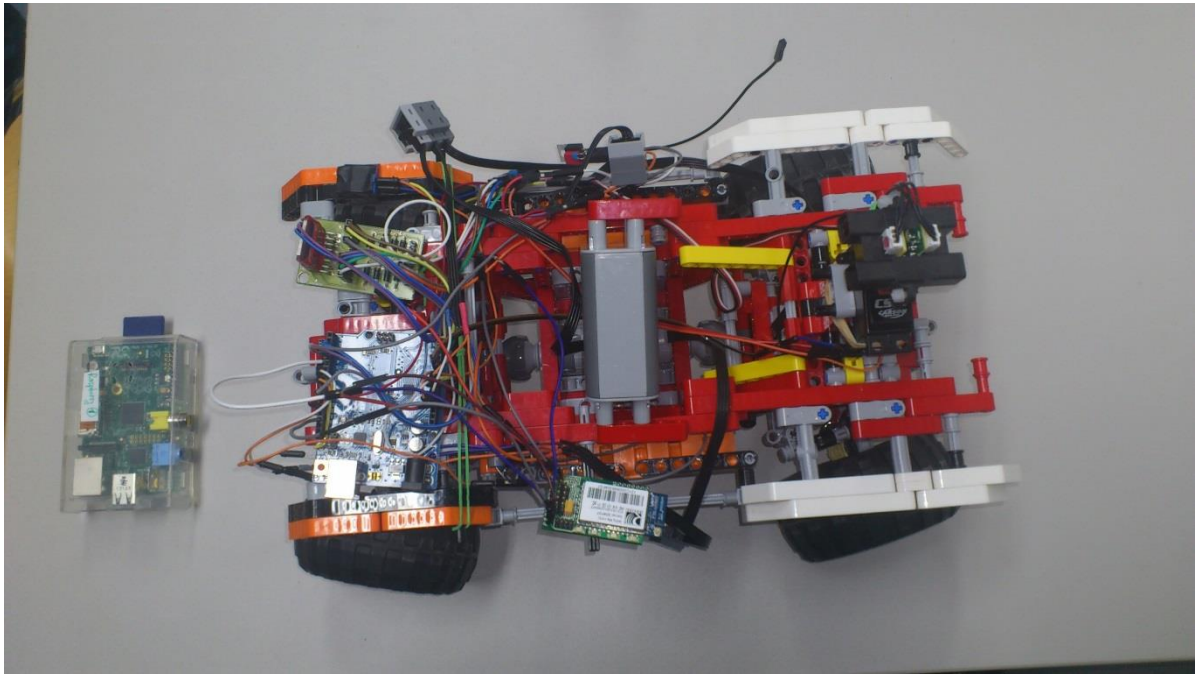
Group: Poorman‘s Laser

Daniel Collaziol
Juri Kuhn
Josef Stark
Thomas Sennebogen
Viet Tiep Do

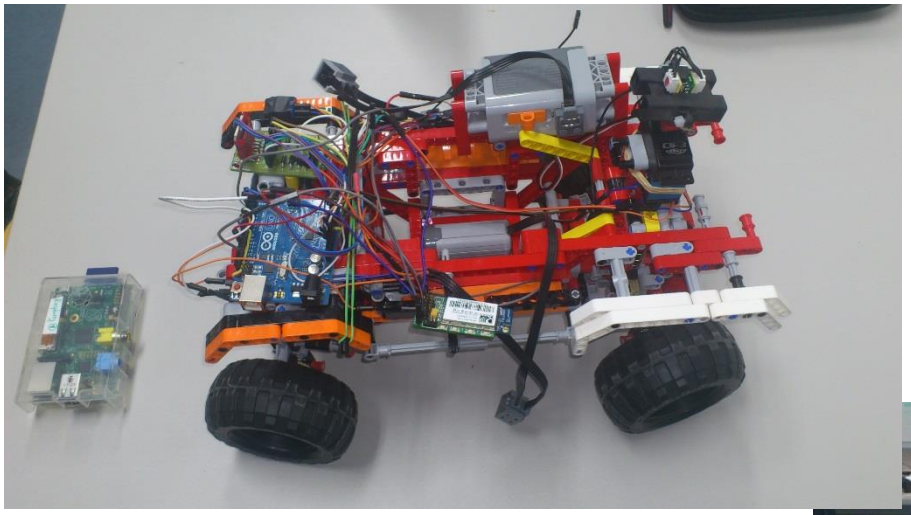
Presentation Highlights

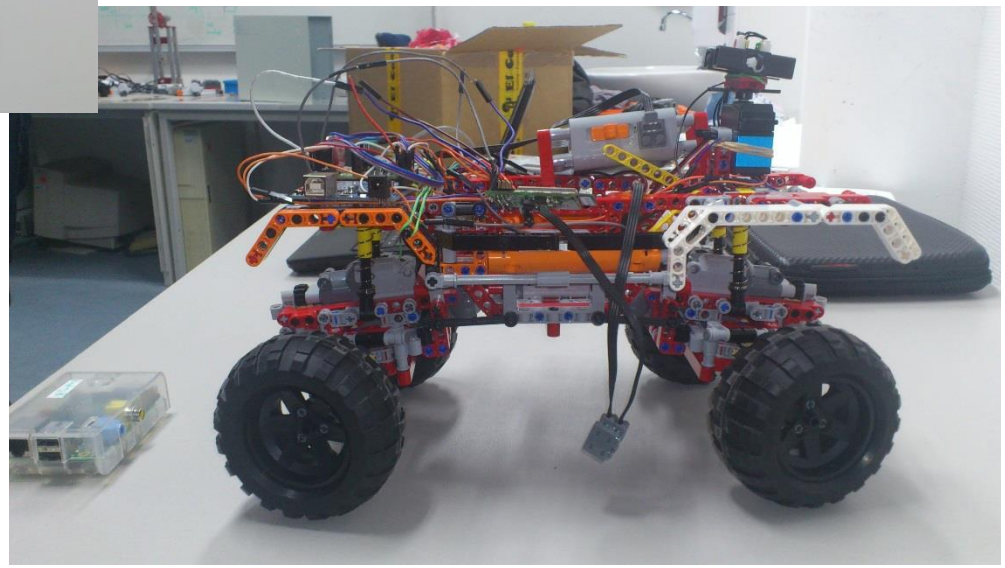
- Introduction
- Progress until midterm presentation
- Objectives after midterm presentation
- Known & Remaining problems
- Solutions and Improvements
- Results
- Experience
- Conclusion
- Additional information & documentation

1. Introduction



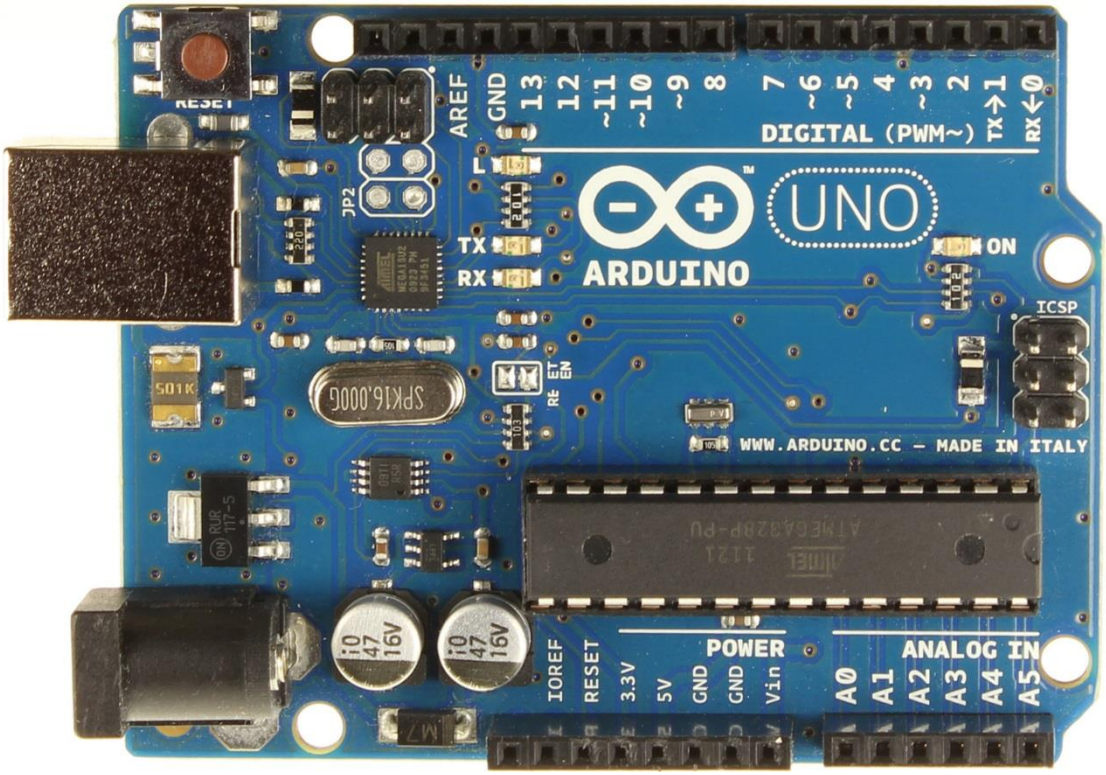
Initial Components
Lego Car
Arduino Board
Raspberry Pi
Engines
Infrared Sensor
Battery, Cables,H-Bridge..
Wifi-Module(which didnt work...)

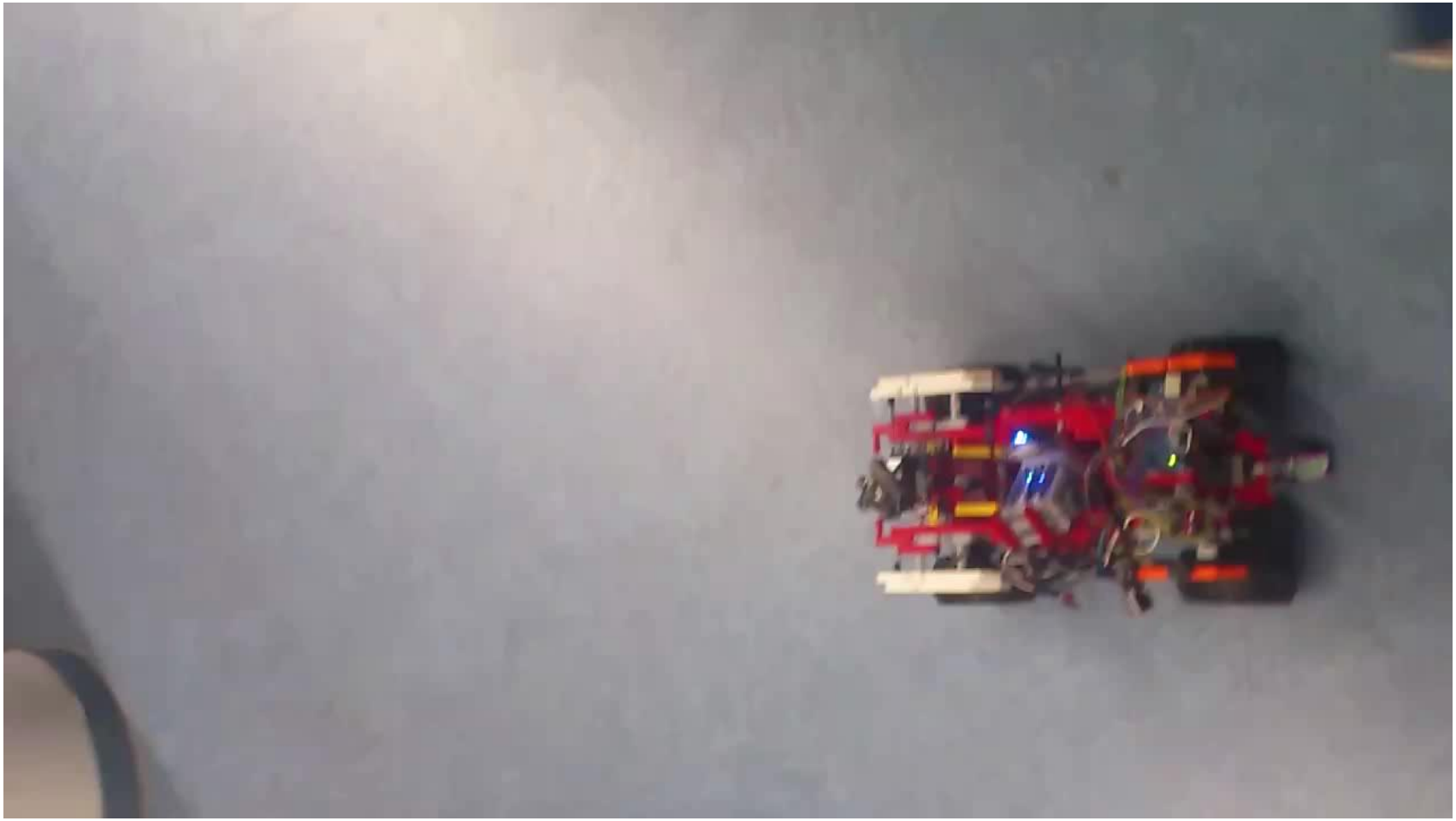




2. Progress until Midterm presentation

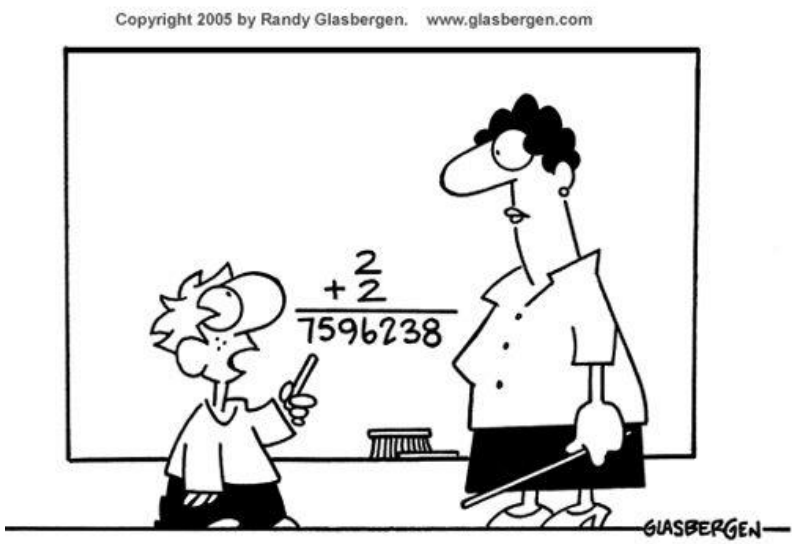
- Development of a basic collision avoidance system
- The C.A. algorithm runs entirely on Arduino
- No path planning yet!





3. Objectives after midterm presentation

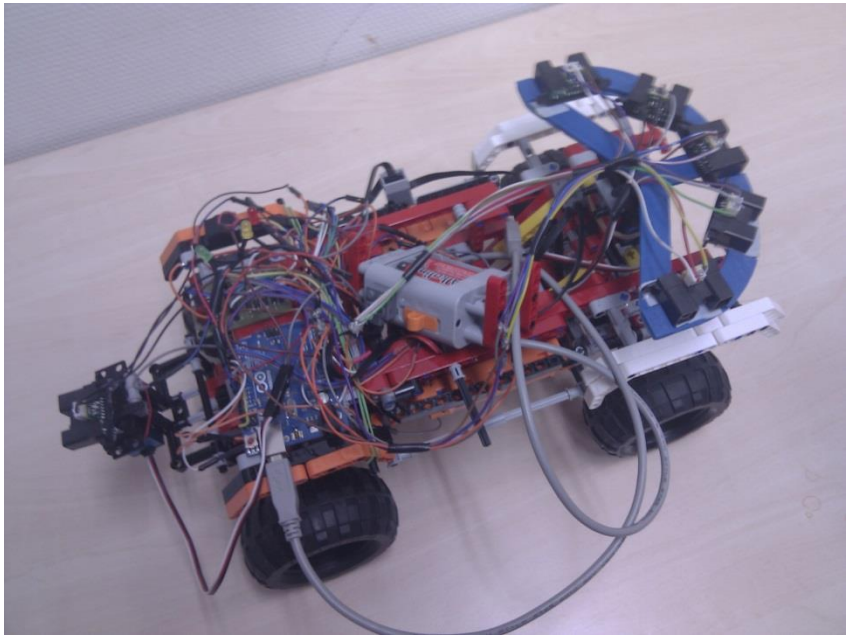
Added Features & Components:
- 5 more distance sensors
- Status LEDs
- Piezo speaker / sound feedback
- Calculations carried out by raspberrry!
- Path planning basic principles
- Remote control via Android application

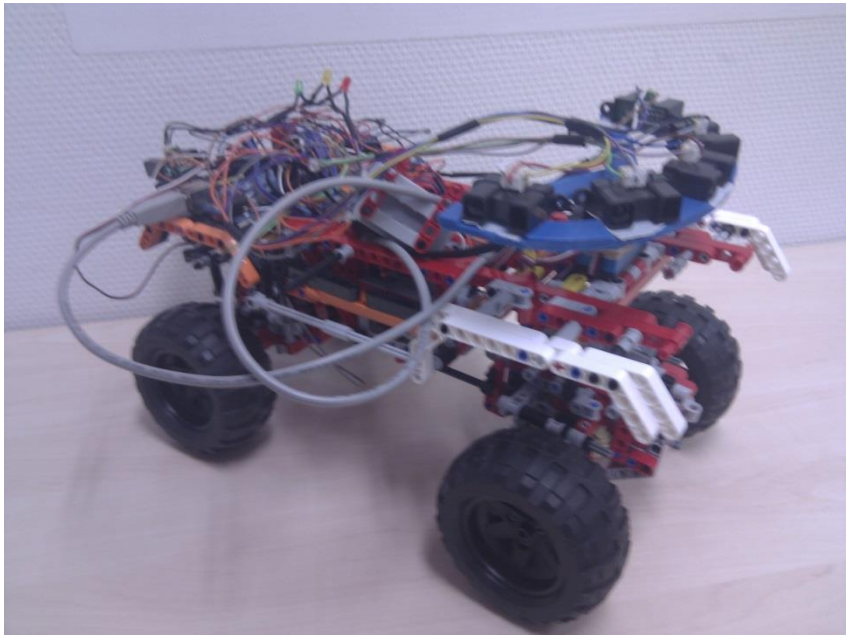


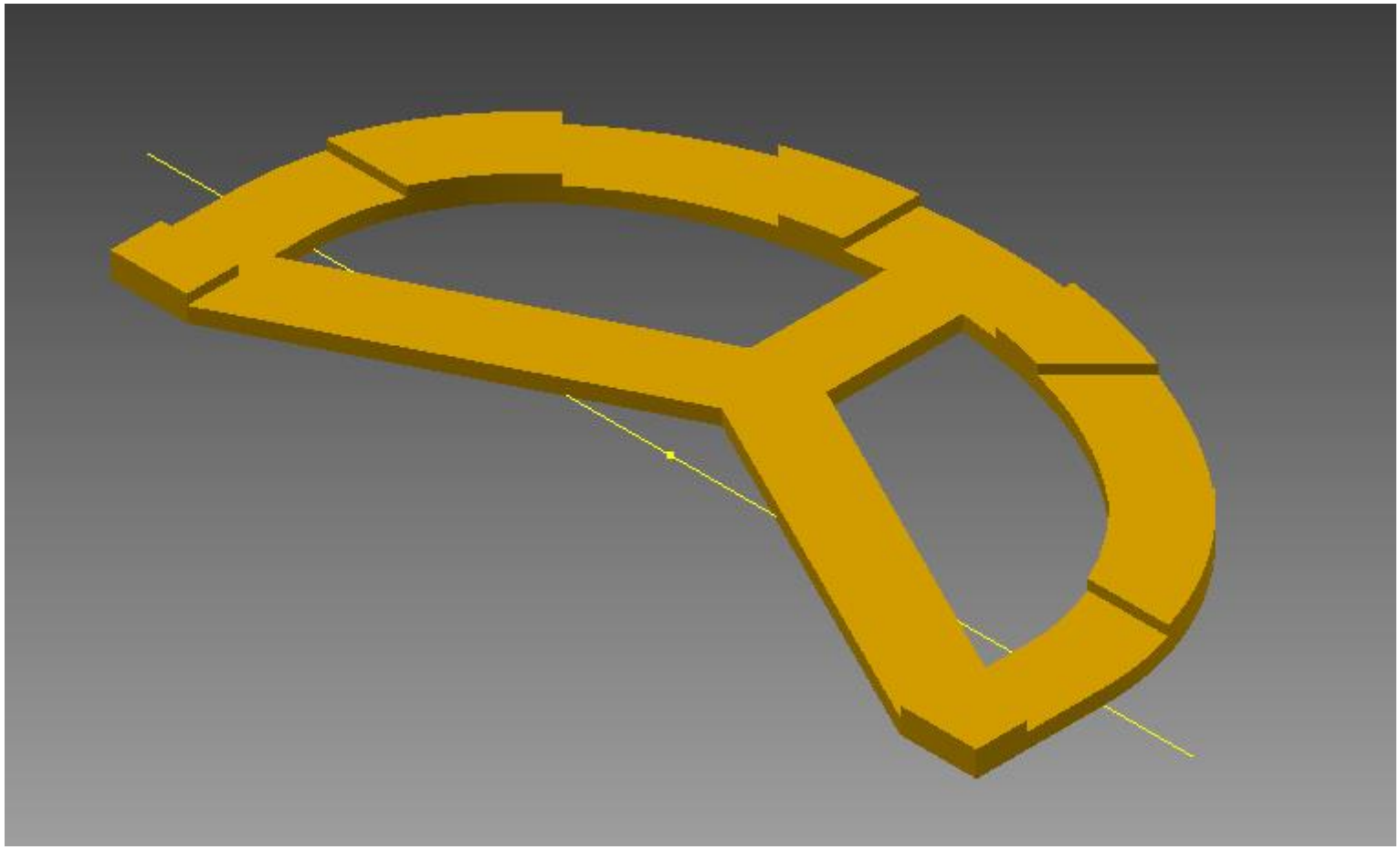
"In an increasingly complex world, sometimes old questions require new answers."

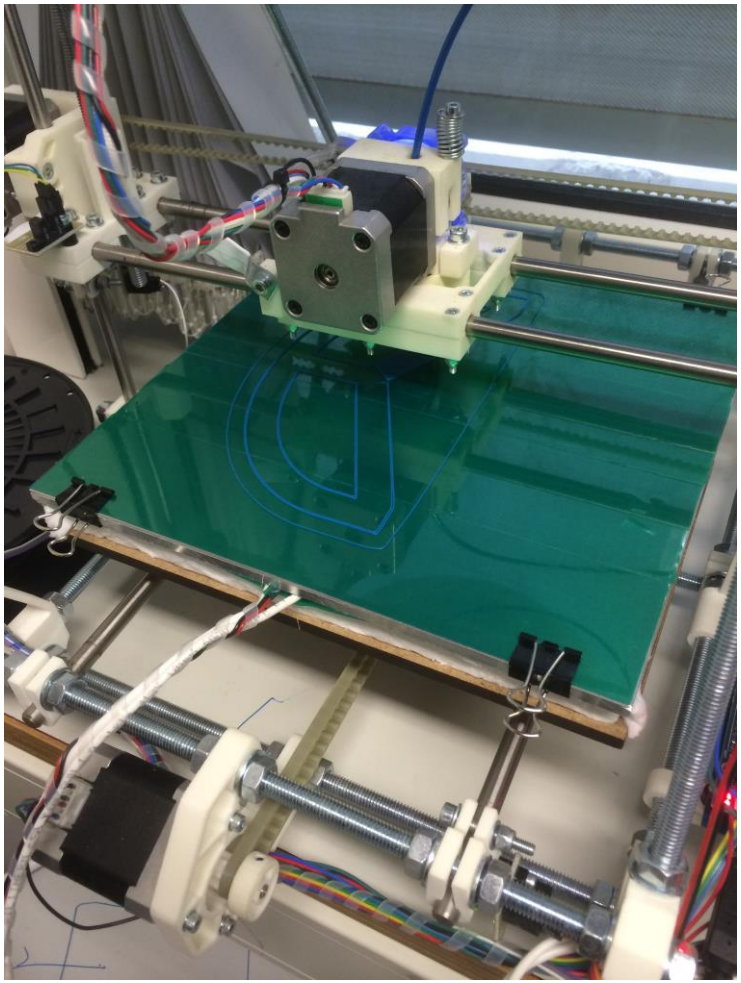
Sensors

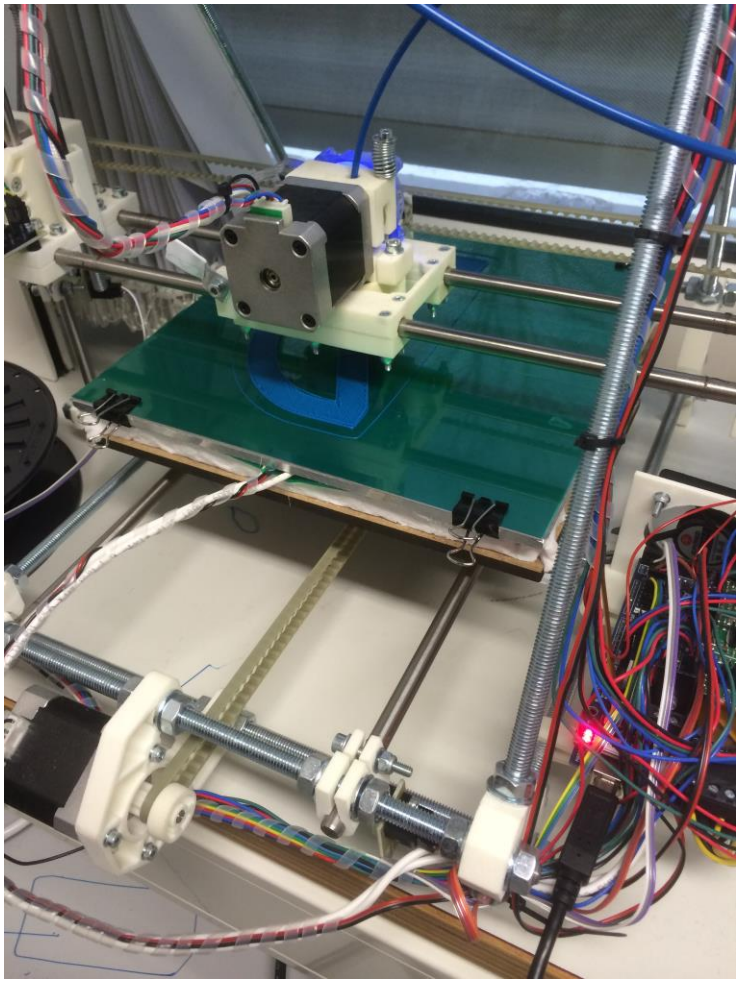
- 5 more distance sensors
- 3D printed laser to build a support for them
- 1 backwards, 5 forwards for better obstacle detection
- Movable mounted on 2 servos

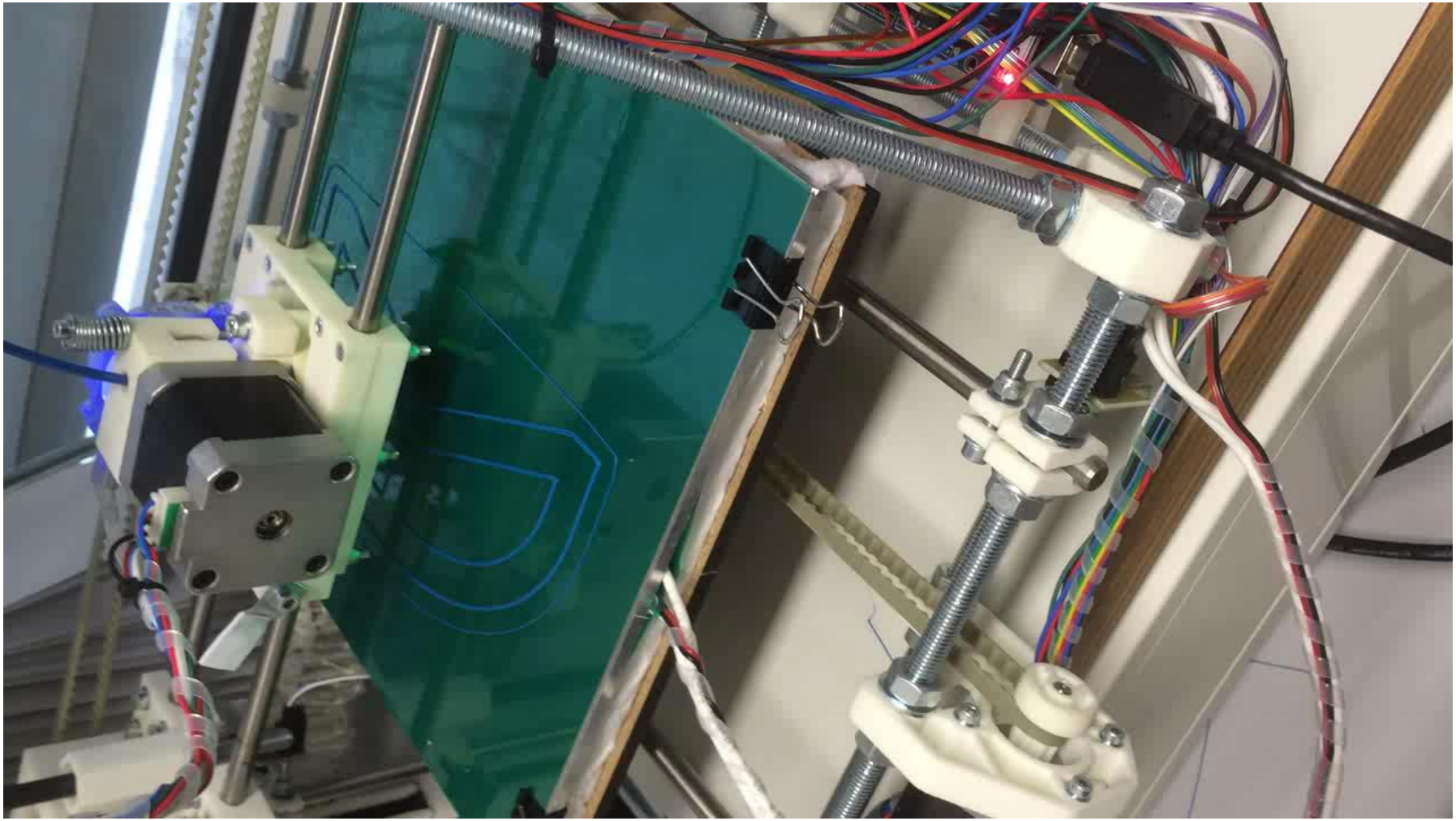








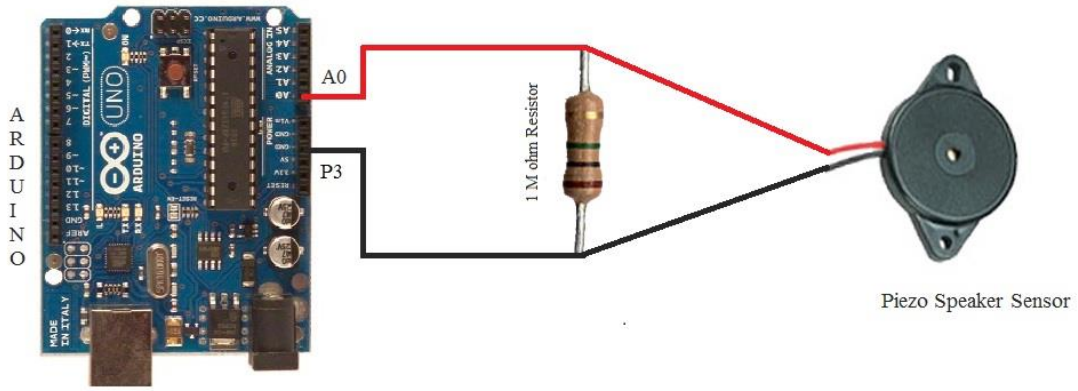


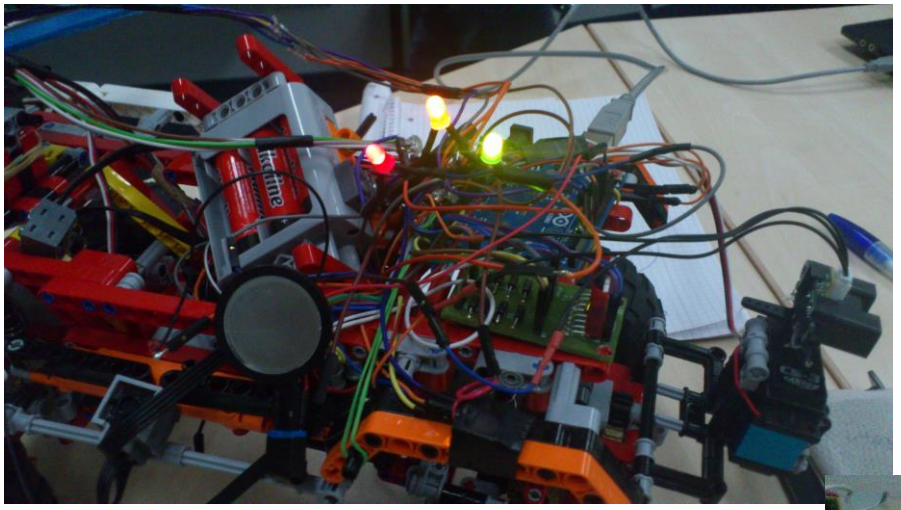


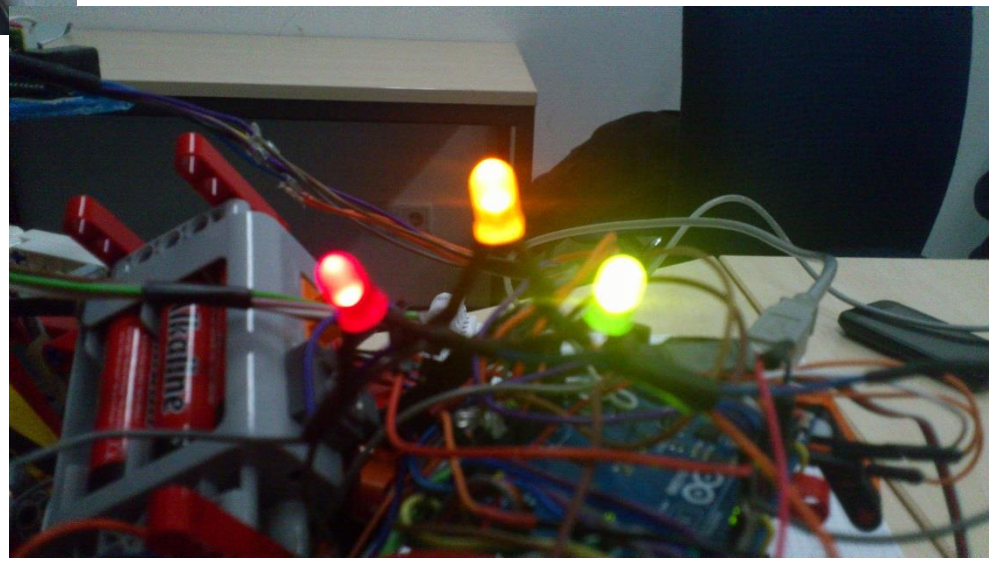
Status LEDs & Piezo speaker

Visual signal with LEDs, which give feedback for three main states:
- Green → Power source connected
- Yellow → Android‘s App connected
- Red → Path planning mode activated

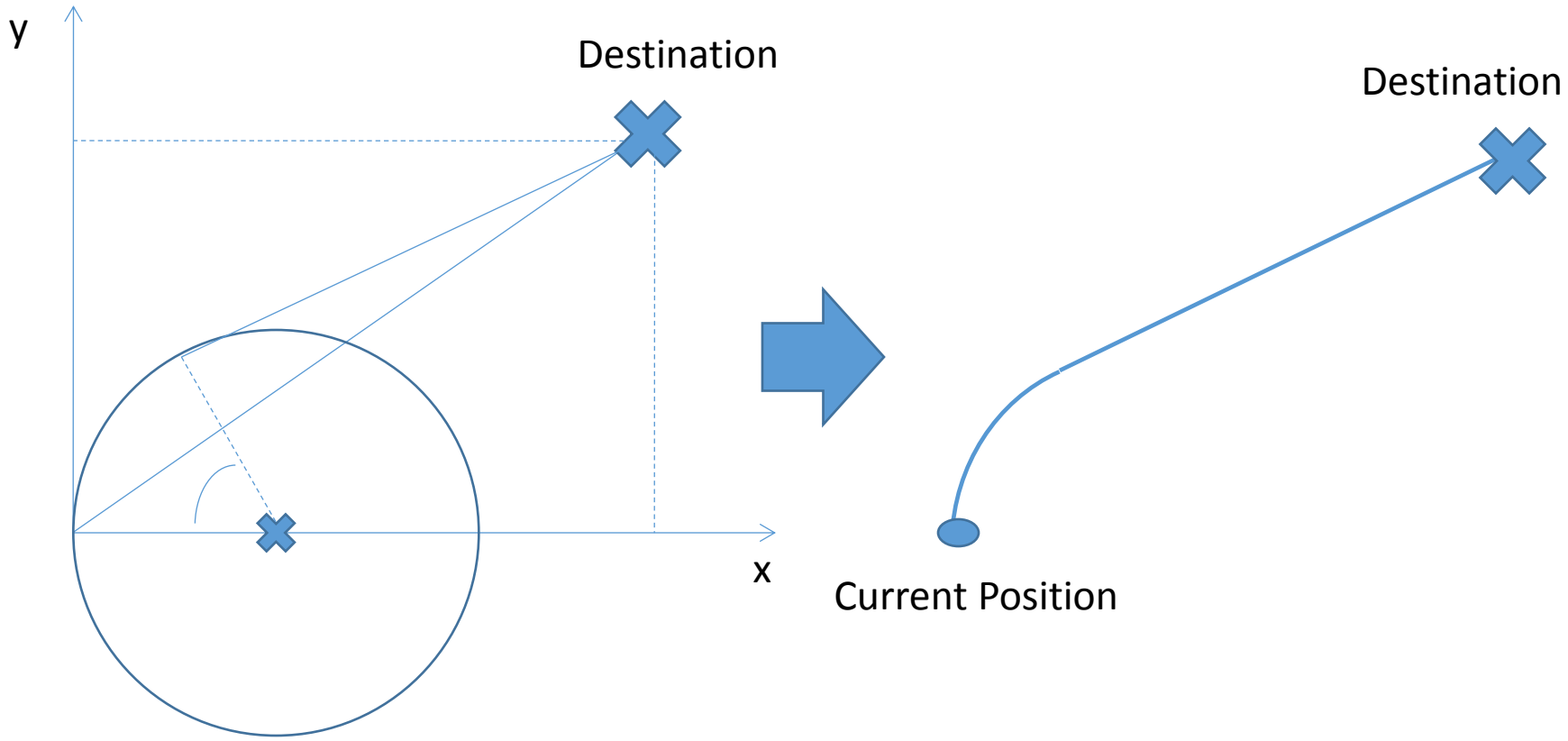
Design to attach the Piezo knock sensor with Arduino





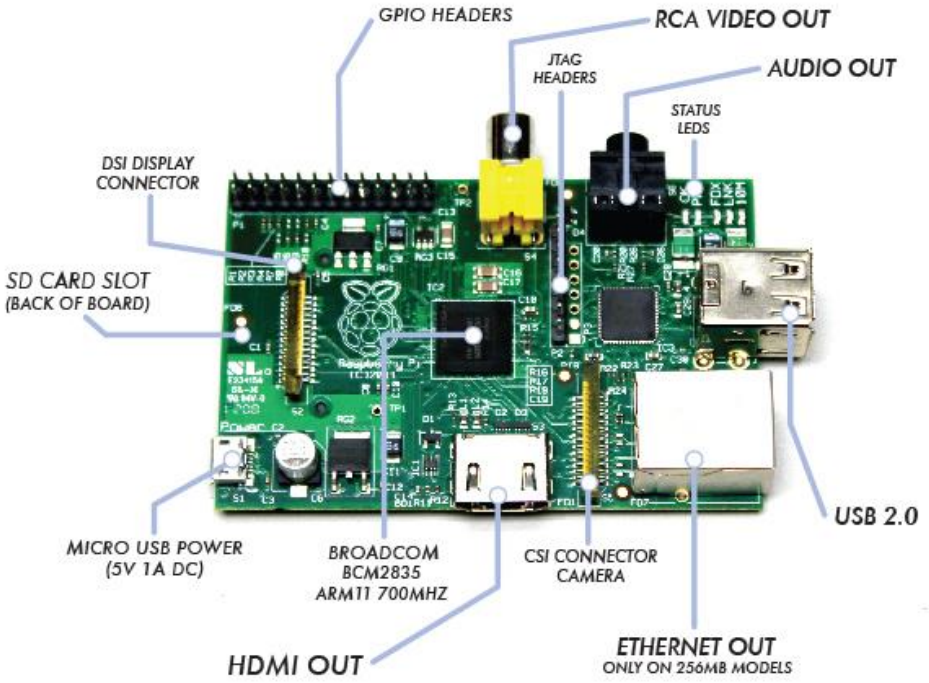


Path Finding



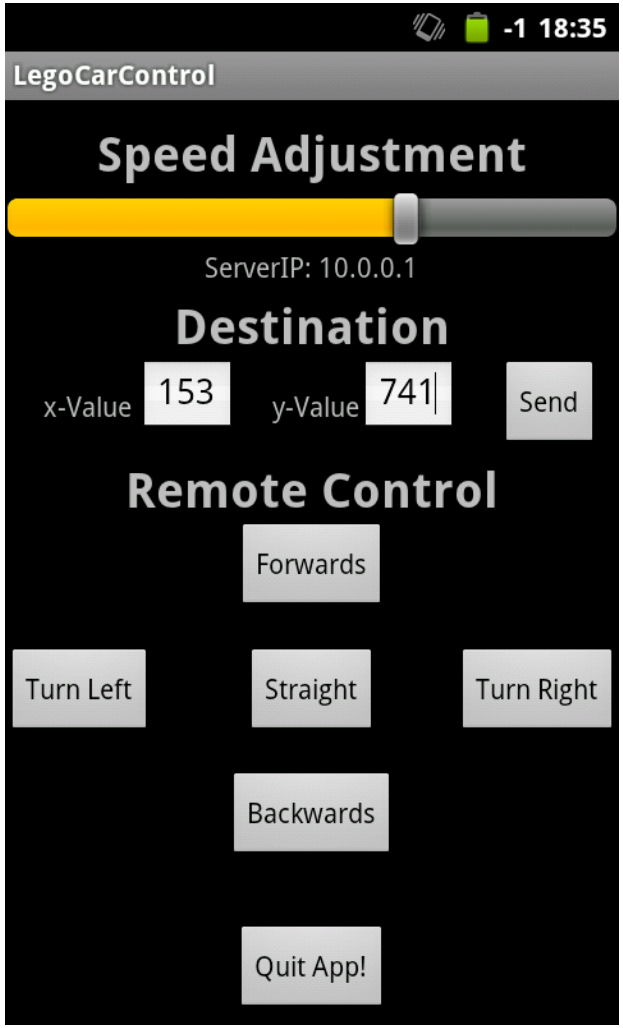
Algorithm‘s overall performance improvement

Path planning requires more memory
and better CPU throughput
→ Main program now running on Raspbe
→ Arduino acts as a simple intermediator between raw HW and Raspberry
→ Communication via binary numbers



Android App

→ Supports two modes
→ Direct control
→ Path planning via cartesian coordinates
→ Path planning: Position given, speed adjustment
→ A simple client/server Java App



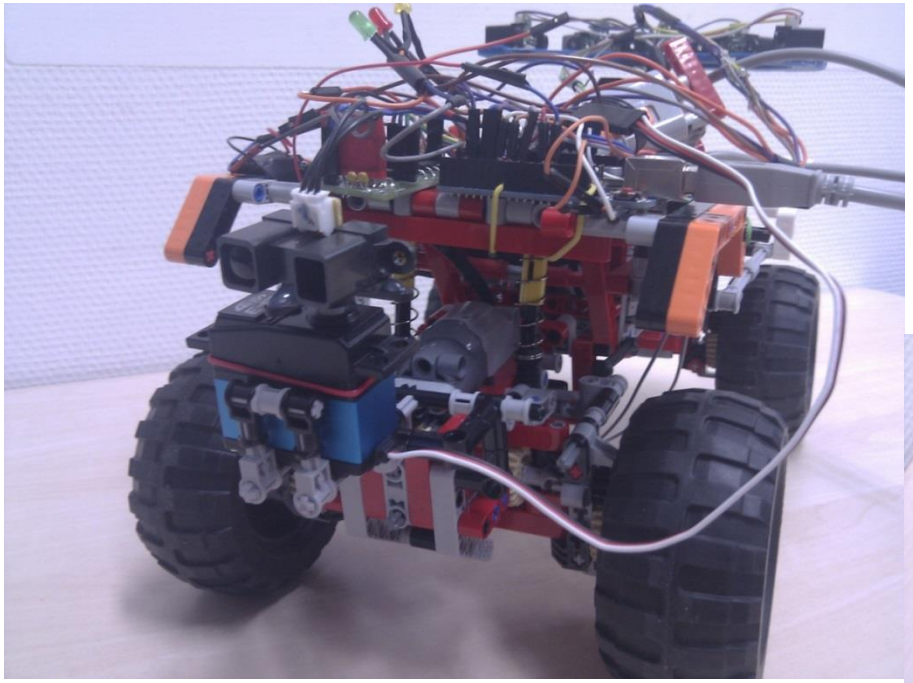
Remote control on raspberry

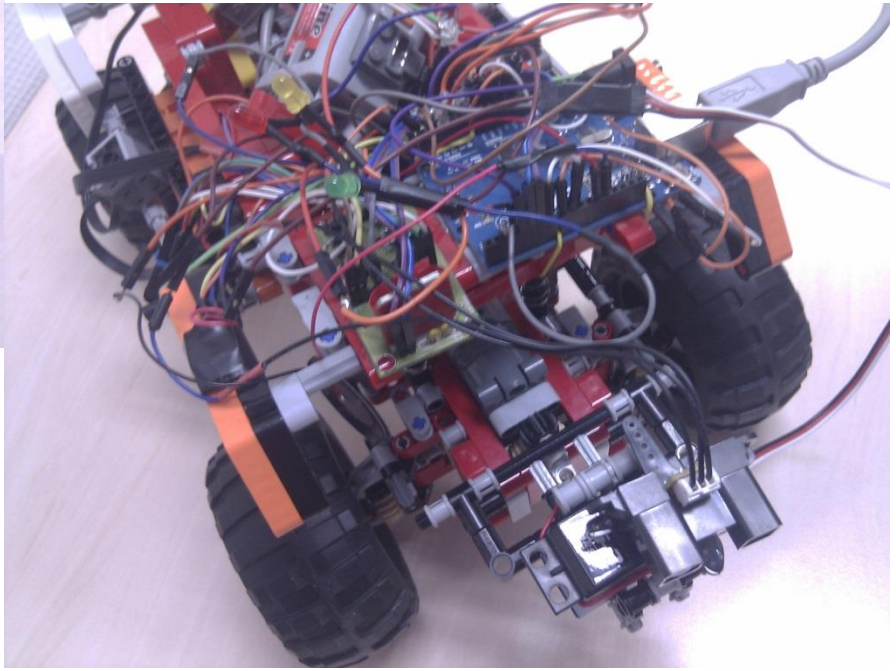
→ Standard USB WIFI Stick for wireless connectivity
→ The adapter provides an Access Point via hostapd (Linux service)
→ Complexity reduced in contrast to WiFly-Module!

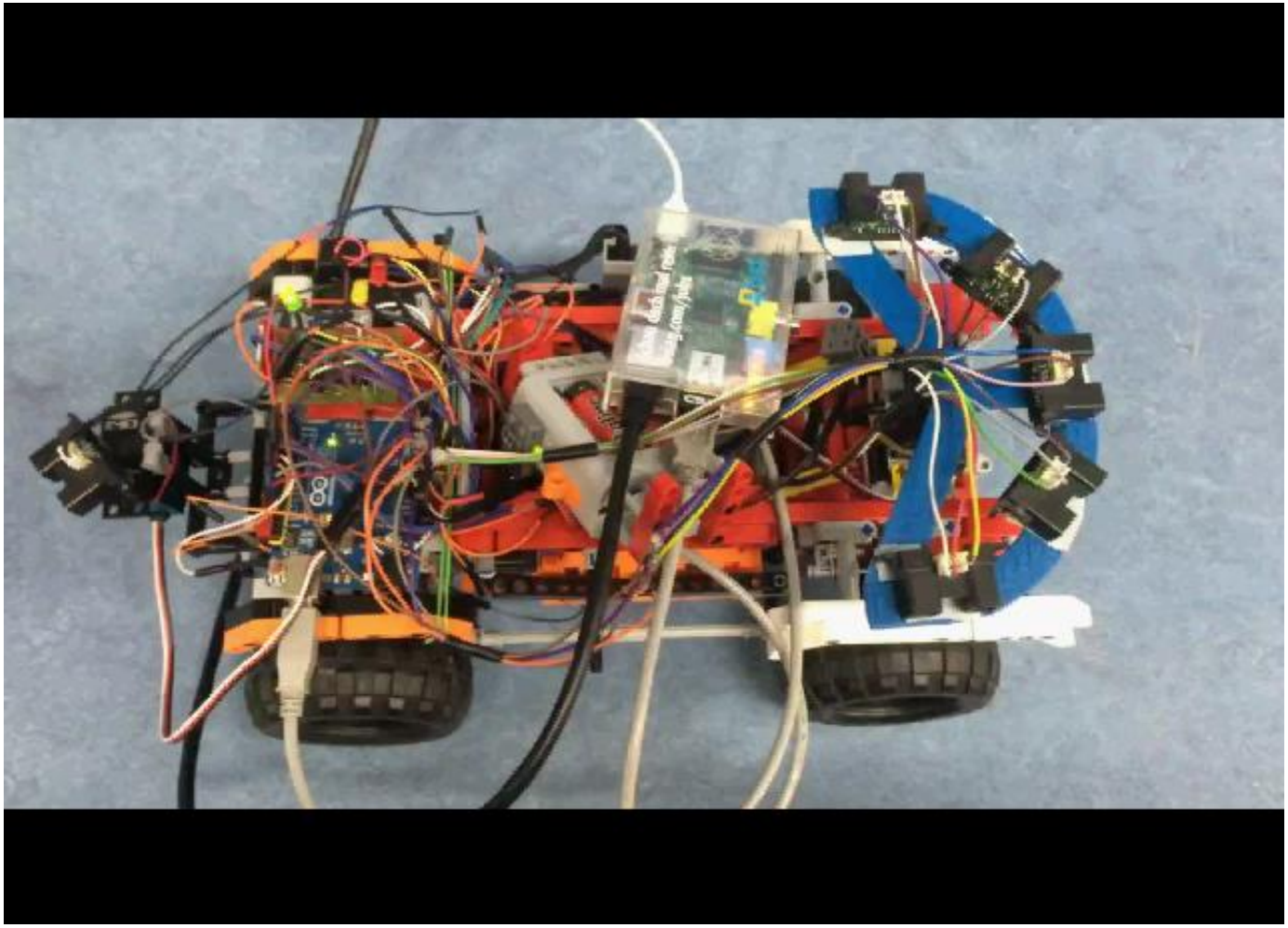
4. Known & Remaining Problems

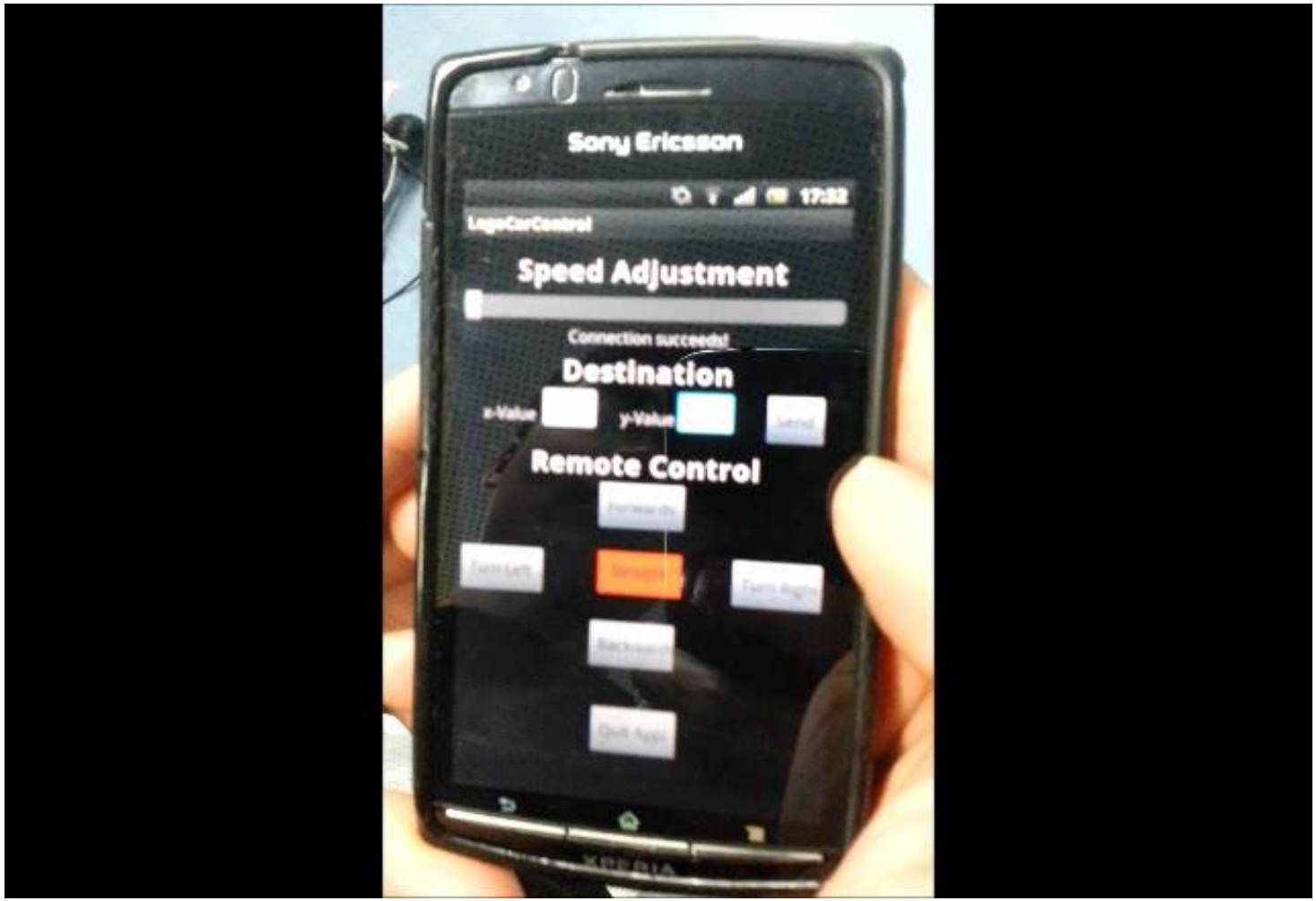
→ Wifi-Module defects, using a Wlan-Module in combination with Raspberry as subtitution
→ Sensor‘s accuracy: inconsistent values
→ Path finding
→ Wires missing, missing contact...
→ Not enough power for sensors

```
if (5 == count)
```



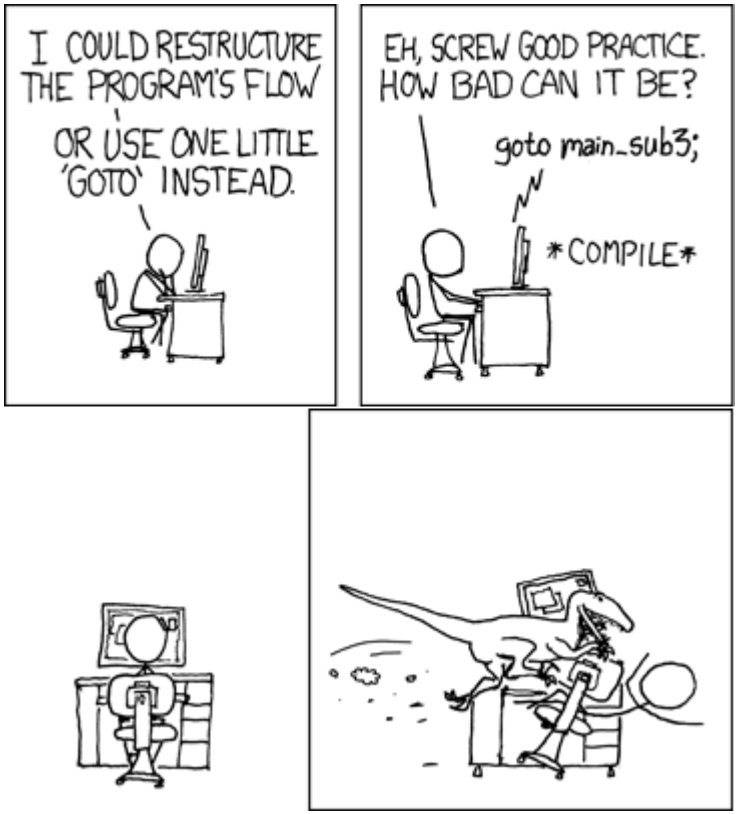





Sony Ericsson
17:32
LegoCarControl
Speed Adjustment
Connection succeeds!
Destination
x-Value
y-Value
Send
Remote Control
Forward
Turn Left
Stop
Turn Right
Backward
Quit App
XPERIA

5. Experiences & some funny facts

→ Batteries for the car, pizzas for the drivers!
→ Almost 20 batteries needed!
→ Just don‘t leave Facebook open…
→ Ok ok… debugging messages are indeed important

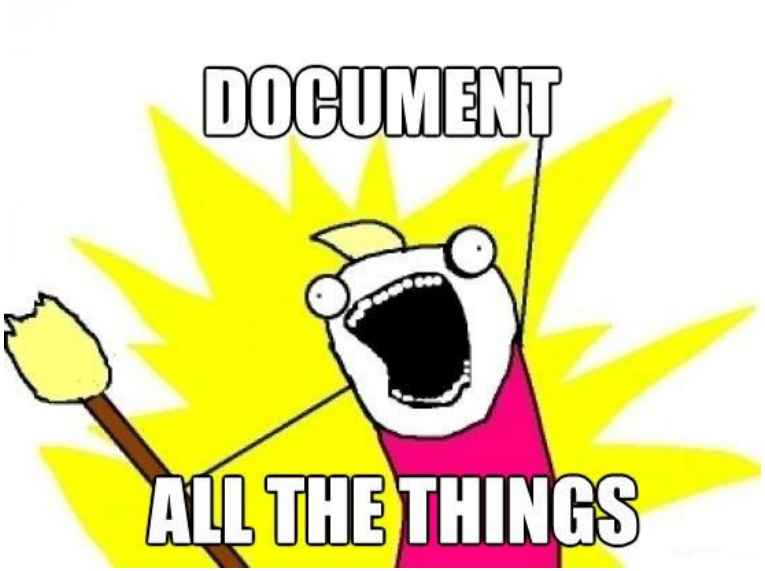


6. Conclusion and potential of development

→ Android App control using accelerometer
→ Path planing with obstacle avoidance
→ Better wires
→ More battery power
→ Microcontroller with more pins

Additional information & documentation

Check it out here:
www.github.com/jmstark/lazyLegoCar



Thank you for your attention!