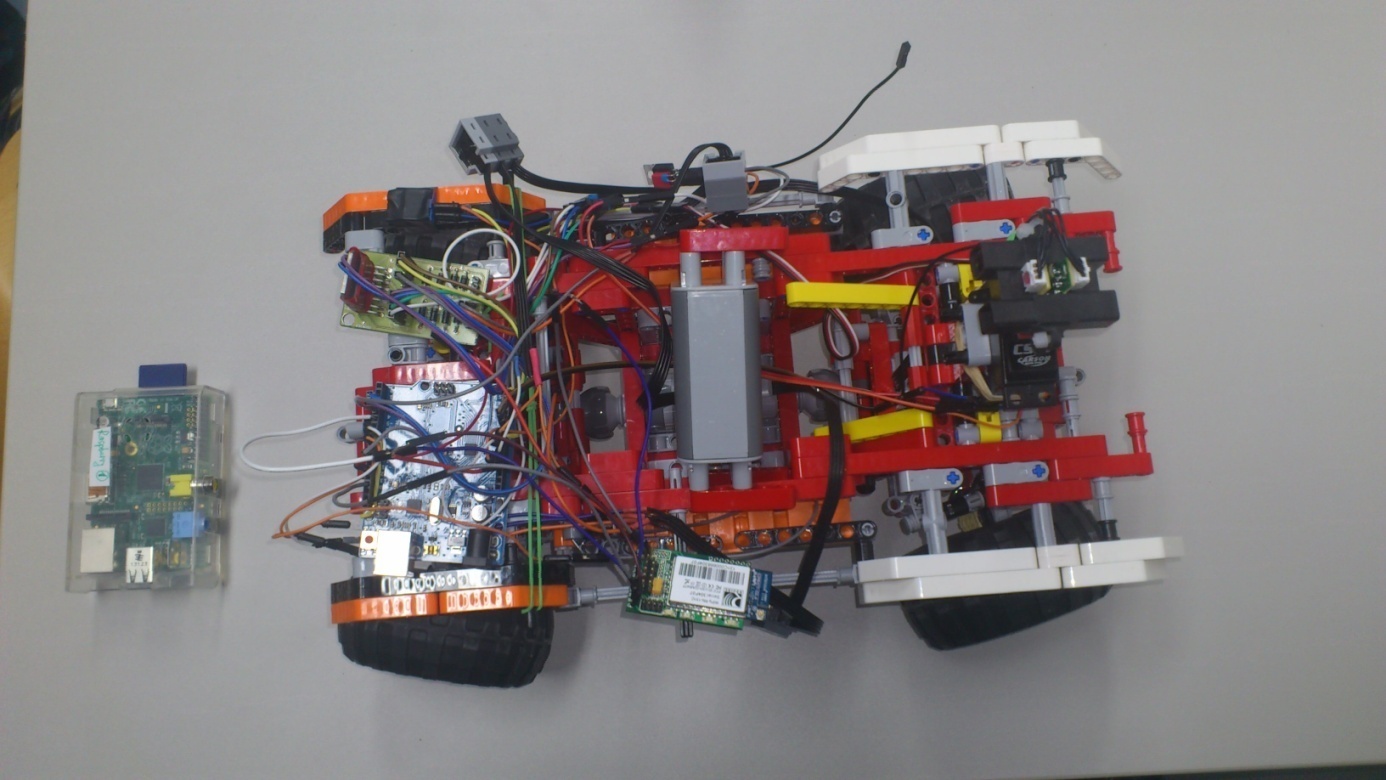


Fakultät für Informatik

Lehrstuhl Robotic und Embedded System

**HW/SW Co-Design with a Lego Car**

**Final Project Documentation – Team Poorman’s Laser**



**Daniel Collazion – Juri Kuhn – Josef Stark – Thomas Sennebogen – Viet Tiep Do**

TABLE OF CONTENT

[Overview 3](#_Toc383448457)

[1. Initial situation 4](#_Toc383448458)

[2. Wireless communication 5](#_Toc383448459)

[2.1 The Android app 5](#_Toc383448460)

[2.2 Raspberry Pi 7](#_Toc383448461)

[3. Added Features 8](#_Toc383448462)

[4. Path planing 9](#_Toc383448463)

[5. Conclusion 10](#_Toc383448464)

[6. Appendix 11](#_Toc383448465)

# Overview

Objective:

The Objective was to build an car that was able to drive autonomously and could be controlled wirelessly. We were given the accomplishments of our predecessing group, which included a lego car with two steering servo motors, two driving motors, an arduino board, a raspberry pi embedded system, a WiFly serial module for wireless communication, two infrared distance sensors mounted together on a servo motor and the required hardware connections between all these components. We also got the sourcecode for a would-be autonomous path planning system. Due to the fact that the previous group did not implement any self-localization mechanism, they tried to implement a simple algorithm which avoids obstacles but follows no special path. It was supposed to scan the environment while standing still, then drive around obstacles until the end of the scanned area is reached, just to stop and scan the environment again, repeating this over and over. Unfortunately, they used ROSserial as library for communication between the raspberry and the arduino board. As the implementation of Rosserial was/is not yet in a usable state on the raspberry, they were not able to test their code. All of this can be read in their documentation.

Goals:

A) Enable basic collision avoidance without path planning

B) Enable collision avoidance with path planning and wireless control via WIFI over an android app.

The beginnings:

We thought, that it would make much sense to build on the work of the previous group, so that we would not have to reimplement everything. So we pulled a current ROSserial release, compiled everything and tried everything out, hoping that the current ROSserial implementation would allow our car to drive. But as - even after trying out many dirty hacks - it was not possible to get it to work, we decided that it would not make much sense to put more energy into this unpromising task of reviving the old code, considering that both a) reimplementing a basic collision avoidance algorithm with a different protocol and b) implementing a simple binary protocol via serial pipes did seem not that difficult. We also estimated, that even in the worst case, the arduino board alone (without help from the raspberry), would provide enough computing power (16 MHz) and memory (2 KiB RAM, 16 KiB ROM) for such a basic task, and also that the scanning could be done while driving, since a full spin of the IR-sensors only needs soundsoviel seconds, while they provide reliable range of 20 to about 90 centimeters and the car is driving with a maximum speed of only soundsoviel m/s. Only using the arduino for this first goal, we could avoid the hassle of having to enable communication between arduino and raspberry. (After the first goal was reached, we however enabled such a protocol since we then used the raspberry for path planning and wireless communication).

# Initial situation

# Wireless communication

# The Android app

In order to control the car remotely using wireless communication we come to the idea of an Android app – name Poorman’s App. As the main purpose is providing a platform which is able to send simple command to control the car rather than developing a fancy or sophisticated one, we did not focus on the design of the app. The app has very simple and basic components that guaranties the usability and still cover all the functions needed to control the lego car. A screenshot of the main page of Poorman’s App can be seen below.

|  |
| --- |
| Unbenannt.jpg  Picture 1: Screenshot of the Android app´ |

In the Speed Adjustment area, user can adjust the car’s speed he want. Speed is an important parameter in Path-planning mode. Notice that any speed-value smaller than 64 will be ignored. The maximal value is 255.

The app supports two modes:

- Basic mode: User can drive the car forwards or backwards by pressing and holding Forwards- or Backwards-Button. Leaving the buttons make the car stop. The buttons Turn Left, Straight and Turn Right steer the wheel-position to the left, straight or to the right correspondently. That means user should set the steering firstly and then drive the car backwards or forwards as changing the direction when the car is driving is not allowed. Besides, there are only three wheel-positions given: straight, 45° degree to the left and 45° degree to the right.

- Path-finding mode: The Lego car is also able to drive to a given position, which can be specified by giving its coordination. User type the x-value and y-value in the text box and press Send-button to trigger the command. As the car receives this command, it will consider its actual position as null-position und drive itself to the desired position. Based on the car speed, curve-radius of its wheel we develop an simple algorithm that automatically calculate the car’s driving path to the reachable given position (See chapter 4 for further information). The most important parameter that decides the accuracy is the speed and to keep the car in constant speed enough power should be provided.

The communication between the app and lego car bases on the technology of socket. A socket address is the combination of an IP address and a port number, much like one end of a telephone connection is the combination of a phone number and a particular extension. We connect a Wifi-usb adapter, which provides a local wifi network, to the Raspberry Pi. A server is setup on the Raspberry, opens a port for communication. The app is basically a client and it can send to or receives data from the server. More details in coding can be found in the appendix.

|  |
| --- |
| scheme.png  Picture 2: Server-client principle |

# Raspberry Pi

Text

# Added Features

Text

# Path planing

Text

# Conclusion

Text

# Appendix

**Arduino Code**

Code here

**Raspberry Code**

Code here

**Android Code**

In order to install the PoormanApp we need several files which can be found in the project folder. Below only the content of three important files are shown. Those are the MainActivity.java, activity\_main.xml and the Manifest.xml. The MainActivity.java contains the core function of the app, which handle the input data provides by the users and send this information to Rasperry Pi. The activity\_main.xml is simple the design of the app and the Manifest.xml file presents the important configuration.

*• MainActivity.java*

package com.example.poormanapp;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.net.Socket;

import java.net.UnknownHostException;

import android.app.Activity;

import android.graphics.Color;

import android.graphics.PorterDuff;

import android.graphics.drawable.ClipDrawable;

import android.graphics.drawable.Drawable;

import android.graphics.drawable.LayerDrawable;

import android.os.Bundle;

import android.view.MotionEvent;

import android.view.View;

import android.view.View.OnClickListener;

import android.view.animation.Animation;

import android.view.animation.AnimationUtils;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ProgressBar;

import android.widget.SeekBar;

import android.widget.SeekBar.OnSeekBarChangeListener;

import android.widget.TextView;

public class MainActivity extends Activity {

Button mainButton;

Button leftButton;

Button rightButton;

Button driveButton;

Button backButton;

Button straightButton;

Button sendButton;

Animation buttonEffect;

Drawable d;

public static final int ServerPort = 4359; //ServerPort is free to choose

public static final String ServerIP = "10.0.0.1"; //ServerIP should be changed according to device

Socket socket;

PrintWriter printWriter;

DataOutputStream DOS;

String xWert;

String yWert;

EditText xInput;

EditText yInput;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

final TextView reading = (TextView) findViewById(R.id.reading);

SeekBar seekBar = (SeekBar)findViewById(R.id.seekbar);

seekBar.setMax(255);

xInput = (EditText) findViewById(R.id.xvalue);

yInput = (EditText) findViewById(R.id.yvalue);

if (ServerIP != null)

reading.setText("ServerIP: "+ServerIP);

try {

socket = new Socket(ServerIP, ServerPort);

printWriter = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));

if (socket.isConnected())

reading.setText("Connection succeeds!");

else

reading.setText("Connection fails!");

seekBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

reading.setText("Speed "+progress+ " cm/s");

if(progress <= 64){

seekBar.setBackgroundColor(Color.RED);

}else{

seekBar.setBackgroundColor(Color.GREEN);

}

printWriter.print("Speed " + Integer.toString(progress));

printWriter.flush();

}

});

buttonEffect = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.button\_effect);

mainButton = (Button) findViewById(R.id.button\_main);

mainButton.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View v){

mainButton.startAnimation(buttonEffect);

printWriter.print("Exit");

printWriter.flush();

try {

socket.close();

} catch (IOException e) {

e.printStackTrace();

}

finish();

}

});

straightButton = (Button) findViewById(R.id.button\_straight);

leftButton = (Button) findViewById(R.id.button\_left);

rightButton = (Button) findViewById(R.id.button\_right);

driveButton = (Button) findViewById(R.id.button\_forward);

backButton = (Button) findViewById(R.id.button\_backward);

sendButton = (Button) findViewById(R.id.button\_send);

d = (Drawable) mainButton.getBackground();

sendButton.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View v){

sendButton.startAnimation(buttonEffect);

xWert = xInput.getText().toString();

yWert = yInput.getText().toString();

String destination = "Destination " + (xWert.equals("") ? "0" : xWert) + " " + (yWert.equals("") ? "0" : yWert);

printWriter.print(destination);

printWriter.flush();

}

});

leftButton.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View v){

leftButton.setBackgroundColor(Color.CYAN);

rightButton.setBackgroundDrawable(d);

straightButton.setBackgroundDrawable(d);

printWriter.print("Left");

printWriter.flush();

}

});

straightButton.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View v){

straightButton.setBackgroundColor(Color.RED);

rightButton.setBackgroundDrawable(d);

leftButton.setBackgroundDrawable(d);

printWriter.print("Straight");

printWriter.flush();

}

});

rightButton.setOnClickListener(new OnClickListener(){

@Override

public void onClick(View v){

rightButton.setBackgroundColor(Color.GREEN);

leftButton.setBackgroundDrawable(d);

straightButton.setBackgroundDrawable(d);

printWriter.print("Right");

printWriter.flush();

}

});

driveButton.setOnTouchListener(new View.OnTouchListener() {

@Override

public boolean onTouch(View v, MotionEvent event) {

if(event.getAction() == MotionEvent.ACTION\_DOWN) {

driveButton.setBackgroundColor(Color.BLUE);

printWriter.print("Forwards");

printWriter.flush();

}

else if(event.getAction() == MotionEvent.ACTION\_UP)

{

driveButton.setBackgroundDrawable(d);

printWriter.print("Stop");

printWriter.flush();

}

return true;

}

});

backButton.setOnTouchListener(new View.OnTouchListener() {

@Override

public boolean onTouch(View v, MotionEvent event) {

if(event.getAction() == MotionEvent.ACTION\_DOWN){

backButton.setBackgroundColor(Color.YELLOW);

printWriter.print("Backwards");

printWriter.flush();

}

else if(event.getAction() == MotionEvent.ACTION\_UP)

{

backButton.setBackgroundDrawable(d);

printWriter.print("Stop");

printWriter.flush();

}

return true;

}

});

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

public void setProgressBarColor(ProgressBar progressBar, int newColor){

LayerDrawable ld = (LayerDrawable) progressBar.getProgressDrawable();

ClipDrawable d1 = (ClipDrawable) ld.findDrawableByLayerId(R.id.progressshape);

d1.setColorFilter(newColor, PorterDuff.Mode.SRC\_IN);

}

}

*• activity\_main.xml*

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

tools:context=".MainActivity" >

<TextView

android:id="@+id/textView2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="@dimen/font\_size"

android:layout\_marginTop="10dp"

android:layout\_centerHorizontal="true"

android:text="@string/SpeedSetting" />

<SeekBar

android:id="@+id/seekbar"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentRight="true"

android:layout\_below="@+id/textView2"

android:layout\_margin="1dp" />

<TextView

android:id="@+id/reading"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/seekbar"

android:layout\_centerHorizontal="true" />

<TextView

android:id="@+id/textView3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="@dimen/font\_size"

android:layout\_marginTop="100dp"

android:layout\_centerHorizontal="true"

android:text="@string/Destination" />

<EditText

android:id="@+id/xvalue"

android:layout\_width="50dp"

android:layout\_height="40dp"

android:layout\_marginTop="135dp"

android:layout\_centerHorizontal="true"

android:layout\_alignParentLeft="true"

android:layout\_marginLeft="70dp"

android:inputType="numberDecimal|numberSigned"/>

<EditText

android:id="@+id/yvalue"

android:layout\_width="50dp"

android:layout\_height="40dp"

android:layout\_marginTop="135dp"

android:layout\_centerHorizontal="true"

android:layout\_alignParentLeft="true"

android:layout\_marginLeft="185dp"

android:inputType="numberDecimal|numberSigned"/>

<TextView

android:id="@+id/xValue"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="150dp"

android:layout\_centerHorizontal="true"

android:layout\_alignParentLeft="true"

android:layout\_marginLeft="20dp"

android:text="@string/xValue" />

<TextView

android:id="@+id/yValue"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="150dp"

android:layout\_centerHorizontal="true"

android:layout\_alignParentRight="true"

android:layout\_marginRight="135dp"

android:text="@string/yValue" />

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="@dimen/font\_size"

android:layout\_marginTop="185dp"

android:layout\_centerHorizontal="true"

android:text="@string/LazyLegocar\_Control" />

<Button

android:id="@+id/button\_send"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_alignParentRight="true"

android:layout\_marginRight="10dp"

android:layout\_marginTop="135dp"

android:text="@string/Send" />

<Button

android:id="@+id/button\_forward"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="220dp"

android:text="@string/Forwards" />

<Button

android:id="@+id/button\_backward"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="350dp"

android:text="@string/Backwards" />

<Button

android:id="@+id/button\_right"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_alignParentRight="true"

android:layout\_marginTop="285dp"

android:text="@string/Right" />

<Button

android:id="@+id/button\_straight"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="285dp"

android:text="@string/Straight" />

<Button

android:id="@+id/button\_left"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_marginTop="285dp"

android:text="@string/Left" />

<Button

android:id="@+id/button\_main"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="430dp"

android:text="@string/Exit" />

</RelativeLayout>

*• Manifest.xml*

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.poormanapp"

android:versionCode="1"

android:versionName="1.0" >

<uses-sdk

android:minSdkVersion="8"

android:targetSdkVersion="18" />

<uses-permission android:name="android.permission.INTERNET" >

</uses-permission>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" >

</uses-permission>

<application

android:allowBackup="true"

android:debuggable="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:screenOrientation="portrait"

android:name="com.example.poormanapp.MainActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

**Fotos, Videos, Presentation, Repository and further information can be found on this site:**

**www.github.com/jmstark/lazylegocar**