

1. Teambesprechung

Zeitplanung

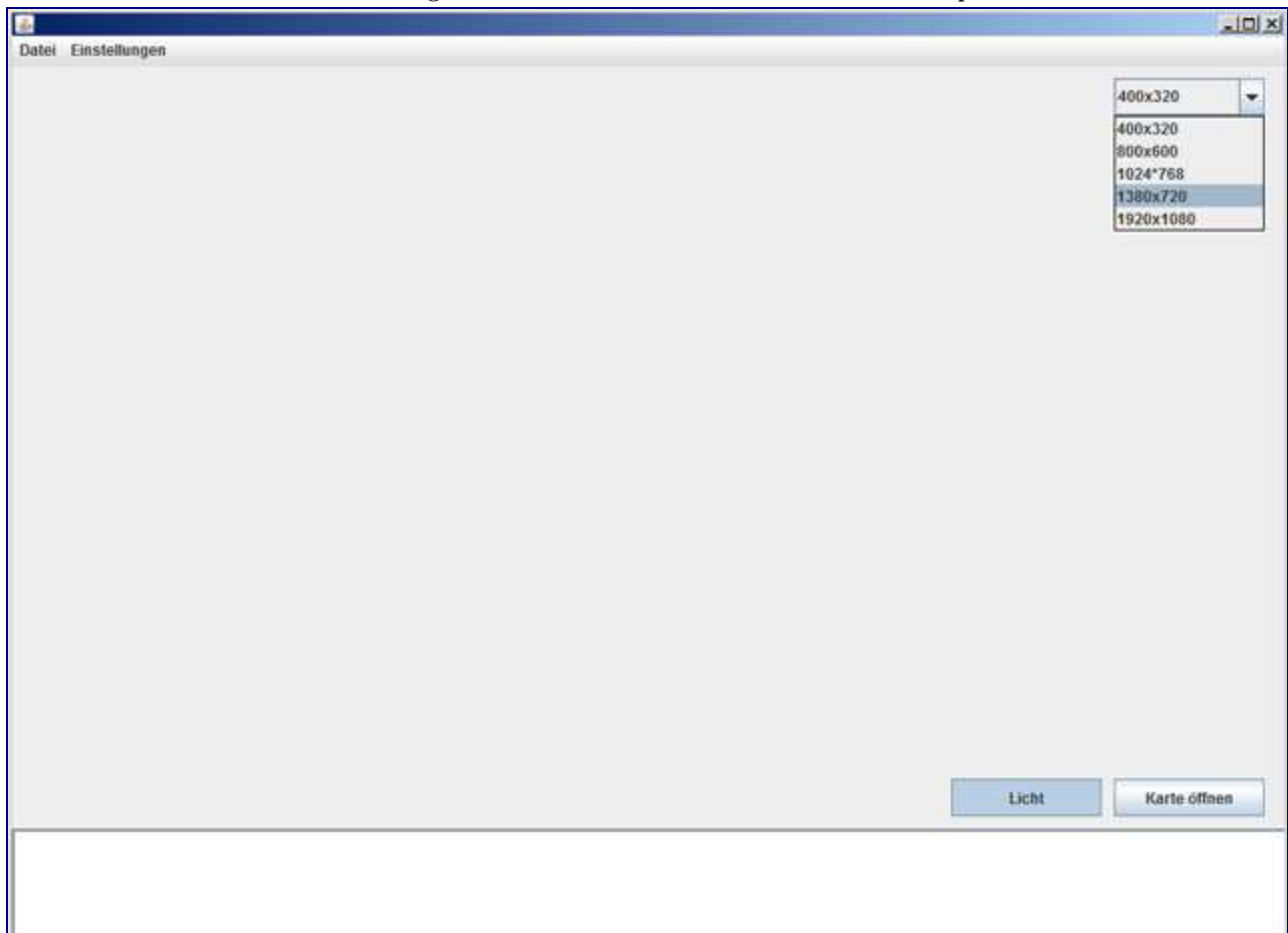
	Grundstruktur	Verbindung/Camera		Sound/Steuerung	Vorlauf	
	1. Woche	2. Woche	3. Woche	4. Woche	5. Woche	20.6
Benjamin B.	GUI Android	Connection	Kamera Android/Java	Soundrecording	Fehlerkorrektur/ Wunschkriterien	Vortrag
Benjamin L.	GUI Java/Log	Packagedata	Carcontroller	Arduino		
Felix	Fassade Controller	Packagedata	Carcontroller	Arduino		
Paul	Android Log	Kamera Android/Java		Soundrecording		
Sven	Android Fassade	Packagedata	Infobox	Arduino		
Christian	Android Fassade	GPS	Infobox	Soundrecording		
Lars	GUI Java/Log	Kamera Android/Java		Springer		
Robin	Netzwerk	Connection	Kamera Android/Java			

Aufbau Grundstruktur

Aufbau der Fassaden

GUI Java

Grundversion der grafischen Oberfläche steht mit den grundlegenden Einstellmöglichkeiten. Zudem wird an dieser weiterhin gearbeitet und alle neuen Funktionen implementiert.



GUI Android

Anbei die fertiggestellte Oberfläche im Android.



Log Java / Android

Log Funktion vom Android komplett funktionsfähig. Währenddessen bei dem Java-Programm noch Anpassungen erfolgen bezüglich kleiner Denkfehler im Entwurf.

Aufgabenverteilung

Paul: GPS_Daten und Connection/Verbindung auslesen+ Zusammenführen Klassen

Robin: Verbindungsaufbau(Netzwerk) und Kameradaten(Aufnehmen und Senden)

Benjamin L: GUI_Java + Informationsbox mit Werten aktualisieren

Sven: Zusammenführen Klassen + JavaDoc/Doxygen belesen und Tutorial einbinden ins Trac

Benjamin B: Icon/IP-Auslesen, Eingliedern bei Problemen in Gruppen

Christian: Soundrecording

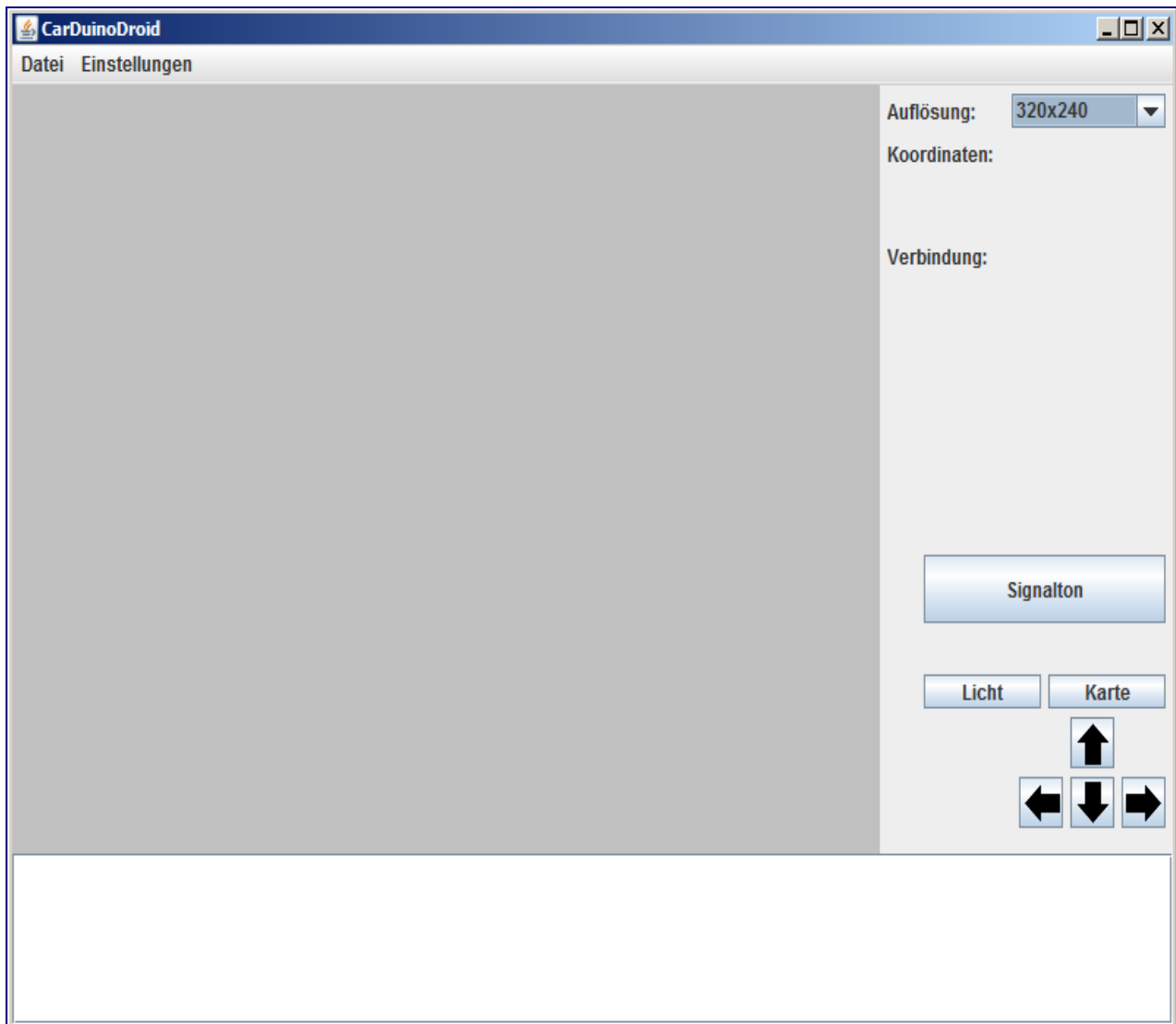
Felix: Soundpool, Controller_Java, Eingliedern bei Problemen in Gruppen

Lars: Package (Java auseinandernehmen)-Informationen + Android-Arduino (Übertragung/Befehle)

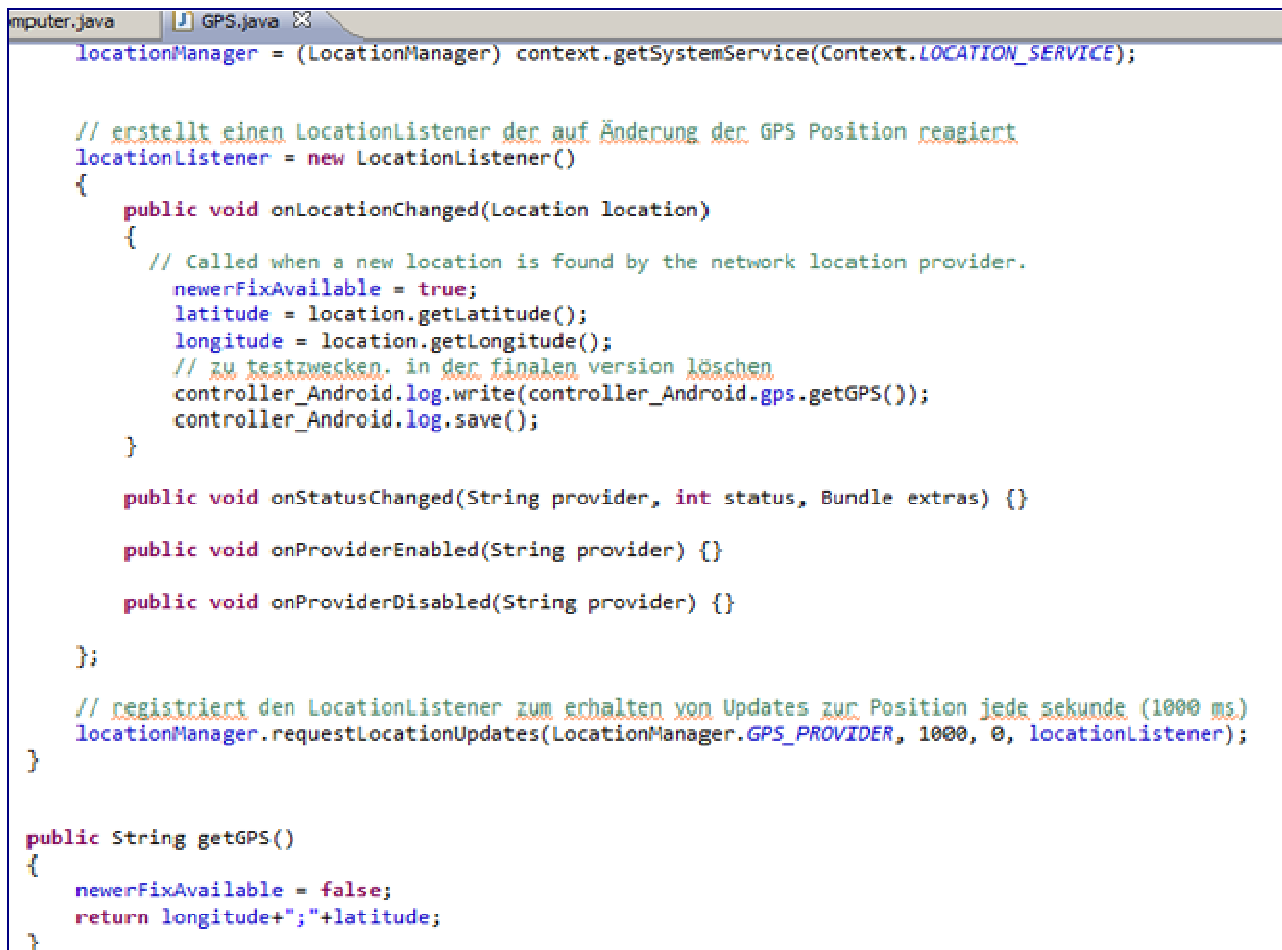
Für alle Threads auslesen für Oberfläche Java und JavaDoc

2. Teambesprechung

GUI Java



GPS-/Verbindungsdaten auslesen

A screenshot of a Java IDE window titled 'GPS.java'. The code is in German and implements a LocationListener to read GPS data. It includes methods for onLocationChanged, onStatusChanged, onProviderEnabled, and onProviderDisabled. It also has a getGPS() method that returns a string with longitude and latitude. The code uses the LocationManager class and the Context.LOCATION_SERVICE constant. It also uses the Log class for logging and the save() method of a controller. The code is as follows:

```
import android.location.LocationManager;
import android.location.Location;
import android.os.Bundle;

public class GPS {
    private LocationManager locationManager;
    private boolean newerFixAvailable = false;
    private double latitude;
    private double longitude;
    private Controller controller;

    public GPS(Context context, Controller controller) {
        locationManager = (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);

        // erstellt einen LocationListener der auf Änderung der GPS Position reagiert
        locationListener = new LocationListener()
        {
            public void onLocationChanged(Location location)
            {
                // Called when a new location is found by the network location provider.
                newerFixAvailable = true;
                latitude = location.getLatitude();
                longitude = location.getLongitude();
                // zu testzwecken. in der finalen version löschen
                controller.log.write(controller.gps.getGPS());
                controller.log.save();
            }

            public void onStatusChanged(String provider, int status, Bundle extras) {}

            public void onProviderEnabled(String provider) {}

            public void onProviderDisabled(String provider) {}
        };

        // registriert den LocationListener zum erhalten von Updates zur Position jede sekunde (1000 ms)
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 0, locationListener);
    }

    public String getGPS()
    {
        newerFixAvailable = false;
        return longitude+" "+latitude;
    }
}
```

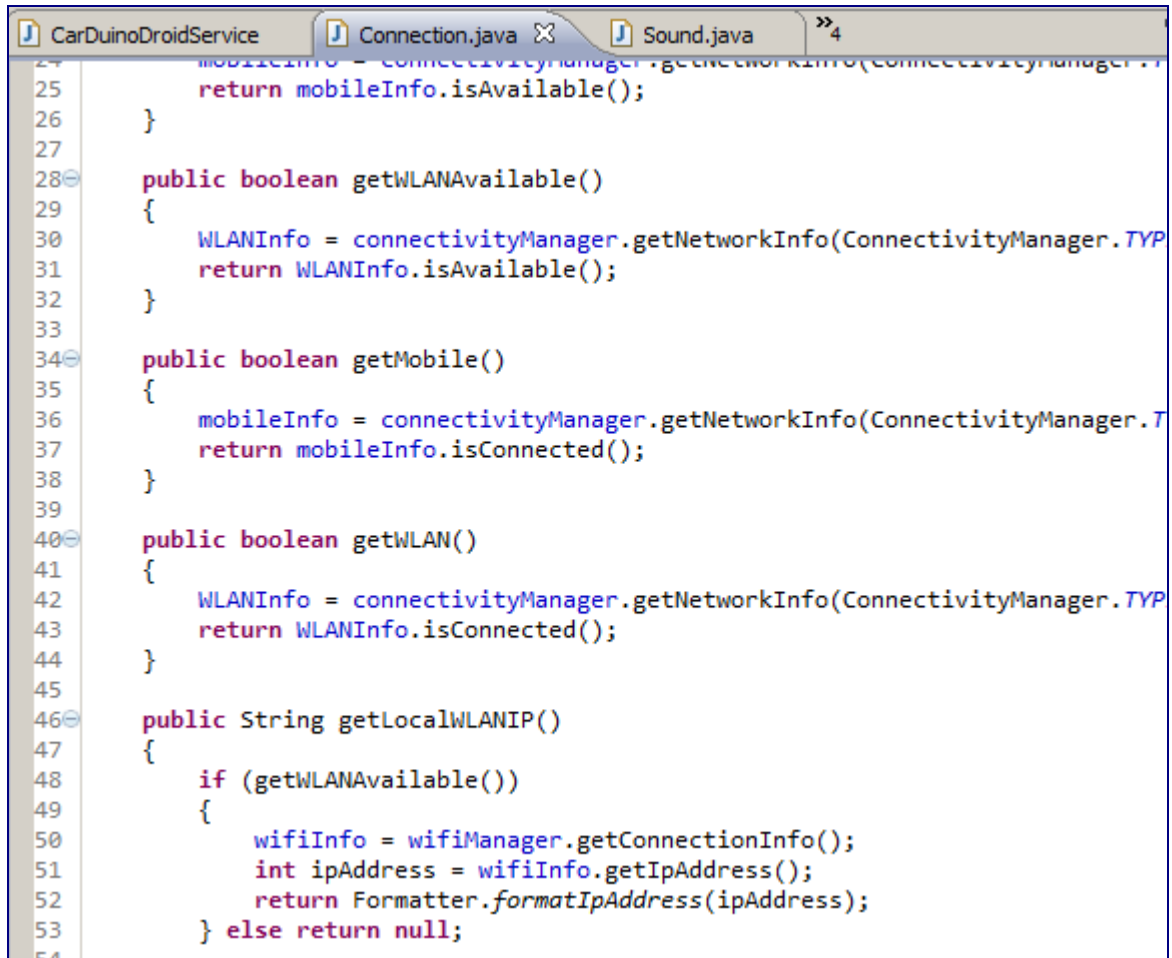
Verbindungsaufbau

Das Senden von Informationen und Daten ist fertiggestellt. Die ersten Testwerte werden im Zusammenspiel zwischen Lars und Robin erstellt und ausgetauscht.

Bildübertragung

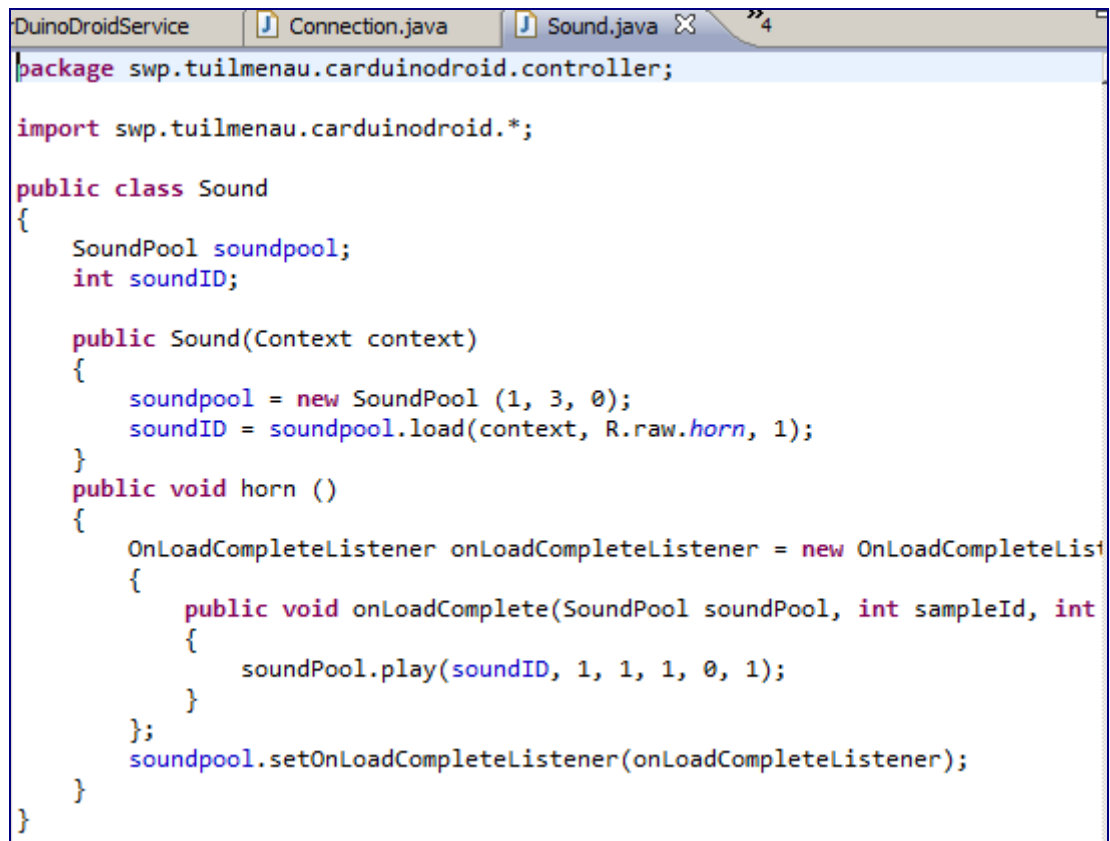
Das große Problem des Projektes, da die Übertragung bisher immer mit starker Verzögerung behaftet ist. Testprogramme klappen nur in geringen Prozentzahlen, wobei auch hier ein Delay bis teilweise 10s vorhanden sind (je nach Umsetzungsart).

IP-Adresse auslesen

A screenshot of an IDE window titled 'CarDuinoDroidService'. It shows a Java file named 'Connection.java' with several methods. The methods are: 'getMobileInfo()' (lines 25-26), 'getWLANAvailable()' (lines 28-32), 'getMobile()' (lines 34-38), 'getWLAN()' (lines 40-44), and 'getLocalWLANIP()' (lines 46-53). The 'getLocalWLANIP()' method uses 'WIFIManager' to get connection info and 'Formatter.formatIpAddress()' to format the IP address. The IDE interface includes a tab bar at the top with 'CarDuinoDroidService', 'Connection.java', and 'Sound.java'. A line number margin is on the left, and a search icon is on the right.

```
24 mobileInfo = connectivityManager.getNetworkInfo(connectivityManager.T
25 return mobileInfo.isAvailable();
26 }
27
28 public boolean getWLANAvailable()
29 {
30     WLANInfo = connectivityManager.getNetworkInfo(connectivityManager.TYP
31     return WLANInfo.isAvailable();
32 }
33
34 public boolean getMobile()
35 {
36     mobileInfo = connectivityManager.getNetworkInfo(connectivityManager.T
37     return mobileInfo.isConnected();
38 }
39
40 public boolean getWLAN()
41 {
42     WLANInfo = connectivityManager.getNetworkInfo(connectivityManager.TYP
43     return WLANInfo.isConnected();
44 }
45
46 public String getLocalWLANIP()
47 {
48     if (getWLANAvailable())
49     {
50         wifiInfo = wifiManager.getConnectionInfo();
51         int ipAddress = wifiInfo.getIpAddress();
52         return Formatter.formatIpAddress(ipAddress);
53     } else return null;
54 }
```

Soundpool



```
package swp.tuilmenau.carduinodroid.controller;

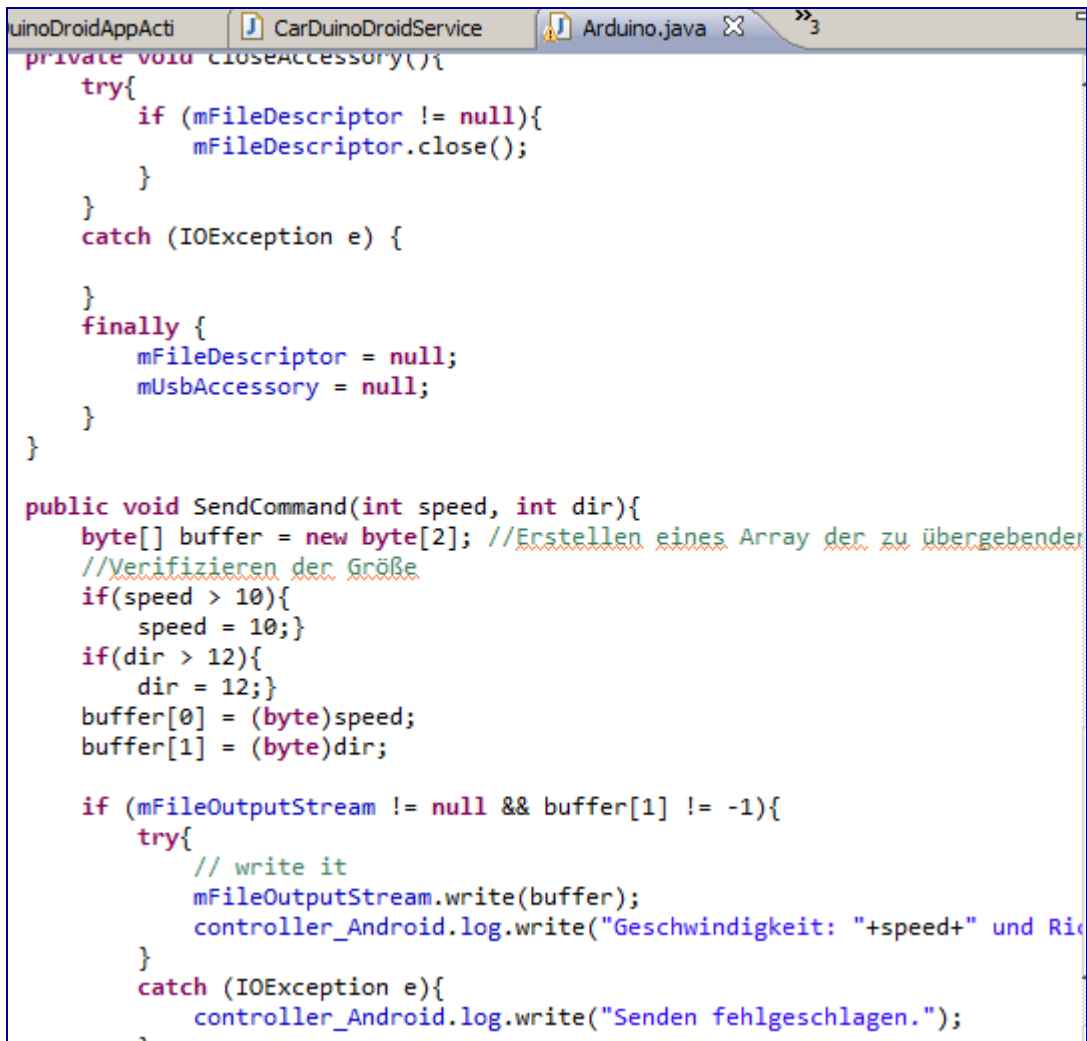
import swp.tuilmenau.carduinodroid.*;

public class Sound
{
    SoundPool soundpool;
    int soundID;

    public Sound(Context context)
    {
        soundpool = new SoundPool (1, 3, 0);
        soundID = soundpool.load(context, R.raw.horn, 1);
    }

    public void horn ()
    {
        OnLoadCompleteListener onLoadCompleteListener = new OnLoadCompleteList
        {
            public void onLoadComplete(SoundPool soundPool, int sampleId, int
            {
                soundPool.play(soundID, 1, 1, 1, 0, 1);
            }
        };
        soundpool.setOnLoadCompleteListener(onLoadCompleteListener);
    }
}
```

Arduino-Befehle senden



```
private void closeAccessory(){
    try{
        if (mFileDescriptor != null){
            mFileDescriptor.close();
        }
    }
    catch (IOException e) {
    }
    finally {
        mFileDescriptor = null;
        mUsbAccessory = null;
    }
}

public void SendCommand(int speed, int dir){
    byte[] buffer = new byte[2]; //Erstellen eines Array der zu übergebenden
    //Verifizieren der Größe
    if(speed > 10){
        speed = 10;}
    if(dir > 12){
        dir = 12;}
    buffer[0] = (byte)speed;
    buffer[1] = (byte)dir;

    if (mFileOutputStream != null && buffer[1] != -1){
        try{
            // write it
            mFileOutputStream.write(buffer);
            controller_Android.log.write("Geschwindigkeit: "+speed+" und Ric
        }
        catch (IOException e){
            controller_Android.log.write("Senden fehlgeschlagen.");
        }
    }
}
```

Festlegung auf JavaDoc

Aufgabenverteilung

Benjamin L: GUI_Java mit Implementierung Methoden

Robin: Verbindungsaufbau(Netzwerk) und Kameradaten(Aufnehmen und Senden) sowie Android App Rest

Lars: Sound Output + Camera Settings + Car Controller Java

Christian: Soundrecording + Paketdata Java

Sven: Entwurfsphase Trac fertigstellen + GPS Map plus Model GPS (gpx-Format)

Paul: Anpassen Arduino Service + Arduino Rest

Benjamin B: Springer

Felix: Springer

3. Teambesprechung

Sound Output

Klasse mit Methoden wurde mit dem ActionListener verknüpft und kann vollständig benutzt werden. Log Funktion ist zudem implementiert, die sich in die Aktion "Sendeanforderung" und die Prüfung einer erfolgreichen Aktion gliedert.

Camera Setting

Alle notwendigen Methoden (Licht, Kameraauflösung und Kameratyp) wurden implementiert und von der Oberfläche wird der Status ausgelesen. Die Log Funktion ist zudem implementiert, die sich in die Aktion "Sendeanforderung" und die Prüfung einer erfolgreichen Aktion gliedert.

Soundrecording

Das Starten und Stoppen wird als Signal aufbereitet sowie als Methoden zur Verfügung gestellt. Die Log Funktion ist zudem implementiert, welche sich in die Aktion "Sendeanforderung" und die Prüfung einer erfolgreichen Aktion gliedert.

Packagedata

Das Gliedern der empfangenen Message über das Netzwerk wird anhand der ";"-Trennung gewährleistet und in einem Array gespeichert.

Car Controller

Der Konstruktor mit grundlegenden Methoden wurde bereitgestellt. Umrechnung für Lenkwinkel und Geschwindigkeitsparameter wurde bisher noch nicht eingebunden. Zudem fehlt das Auslesen/Übergeben der Werte von dem GUI.

GPS Map/ Track

Nicht Fertiggestellt

Android Applikation

Große Schritte im Zusammenspiel der einzelnen Klassen sichergestellt. Umänderung von Service auf Activity bedingt durch die Kamera. Hier muss eine Activity gewährleistet sein, um Bilddaten auslesen sowie übertragen zu können. Arduino Klasse wurde dementsprechend verändert, dass die Methodik auf Activity wieder umgestellt wurde und benötigt nun einen Echtzeittest, ob übertragene Signale auf Seiten des Arduino verarbeitet werden können.

Aufgabenverteilung :

Java

- ⤴ Benjamin L.: GUI (Car_Controller Elemente auf GUI, Bildverarbeiten/-anzeigen, Feinheiten)
- ⤴ Lars: ActionListener (restliche Verknüpfung mit GUI)
- ⤴ Springer: Camera Pic weiterleiten
- ⤴ Christian: Package Data
- ⤴ Sven, Felix, Benjamin B.: GPS: Button zum Anzeigen Google Maps und *.gpx speichern
- ⤴ Robin: Kamerabilder übertragen

Android

- Robin, Paul:
 - ⤴ Kamera
 - ⤴ Controller
 - ⤴ Arduino anpassen (Rest)

4. Teambesprechung

Car_Controller mit KeyListener

Funktionierende Abfrage der Tasten über Variablen setzen/löschen gelöst, um den Mehrfachdruck zu lösen. Dabei kann einerseits das Auto per Pfeiltasten gelenkt und zu gleicher Zeit die beigefügten Gänge zur Regulierung der Geschwindigkeit genutzt werden. Auch besteht die Möglichkeit den Lenkwinkel zu steuern, um zu starke kurven bei hoher Geschwindigkeit zu vermeiden. Dieser Funktionsumfang kommt vollkommen ohne Logeinträge aus, um unnötigen Datenträffic zu vermeiden.

*.gpx-File (Wunschkriterium)

Das Speichern des Routenverlaufes einer Strecke wird dahingehend gewährleistet, dass eine demnentsprechende Datei beim Start des Java-Programms angelegt wird, die bei neuen GPS-Informationen die wichtigen Daten in den Log schreibt. Mit dem normalen Beenden des Programms wird die Datei mit den restlichen Informationen gefüllt, um auf diversen Routenprogrammen sowie Internetseiten die Strecke anzeigen zu können.

Verbinden ActionListener mit GUI

Restliche ActionListener wurden mit den Aufrufen auf der GUI verbunden. Mit dem Log konnte die Funktionalität sicher gestellt werden.

Live-Log

Die detaillierte und zeitnahe Anzeige des Live-Logs auf dem GUI versorgt den Nutzer mit ausreichend Informationen über den genutzten Funktionsumfang des Programms. Fehler in Verbindung sowie Sendeprobleme werden darin auch ausgewiesen.

JavaDoc Beispiel

Ein erstes JavaDoc Beispiel wurde in eine der zahlreichen Klassen eingebunden, um für alle anderen die nötigen Grundlagen zu schaffen, die eigenen Programmteile zu dokumentieren. Besonders die Beschreibung von Methoden und der Klasse stellen den Hauptbestandteil dar.

Sprachdateien einbindbar

Man kann per Sprachdateien die komplette Oberfläche einfach und unkompliziert ändern. Dabei müssen die Einträge nur zeilenweise übersetzt und angepasst werden. Dies gewährleistet eine internationale Nutzung des Programms

unvollständige Teile

- ⤴ Arduino-USB-Steuerung: durch fehlende Testmöglichkeit kann die Klasse mit den jeweiligen Methoden nicht in einer Live-Umgebung getestet werden
- ⤴ Bildübertragung: Probleme bezüglich der Verzögerung in vielen getesteten Programmiervarianten haben bisher die Anzeige erschwert. Weitere Methoden werden ausprobiert um das "Delay" auf ein Minimum zu reduzieren. Weiterhin noch Hardware technische Probleme bezüglich der Bilderverarbeitung (teilweise Handys zu langsam)

Aufgabenverteilung letzte Implementierungswoche

JavaDoc auskommentieren für alle. Robin/Paul:

Rest von Android Felix/Sven/Paul: Darstellung der Bilddaten auf GUI Java

Benjamin/Lars: Anpassungssachen Java

Christian: Vortrag Teile entwerfen, PDF