

Inhaltsverzeichnis

1 Einleitung	1
2 Fahrzeugmodifikation	2
2.1 Originalfahrzeug	2
2.2 Neue Komponenten	4
2.3 Ansteuerung der Fahrzeugmotoren	8
2.4 Installation der Scheinwerfer und Blinker	11
2.5 Installation der Ultraschallsensoren	15
2.6 Geräteinstallationen	16
3 Einrichtung des Raspberry Pis	20
3.1 Externe Software	20
3.1.1 Motion (Webcam Live-Übertragung)	20
3.1.2 OMXPlayer	20
3.2 Programmierung	21
3.2.1 Einleitung	21
3.2.2 Setup	21
3.2.3 Herstellung der Socket-Verbindung zum Steuergerät	23
3.2.4 Empfang und Umsetzung der Befehle von der Android App	23
3.2.5 Auswertung der Sensordaten und eventuelles Abbremsen des Fahrzeugs	24
3.2.6 Übertragung des aktuellen Status an die Android App	24
3.3 Kompilieren und Starten des Programms	25
4 Android App Entwicklung	26
4.1 Programmierung	26
4.2 Webcam Live-Übertragung	27
4.3 Spracherkennung	28
4.4 Benutzeroberfläche	29
4.4.1 Hochformat	29
4.4.2 Querformat	30
5 Anhang	33
5.1 Quellcode des Raspberry Pi Programms	33
5.2 Quellcode der Android App	47

1 Einleitung

Der Funktionsumfang herkömmlicher fernsteuerbarer Modellfahrzeuge beschränkt sich oft auf das Ausführen einfacher Steuerbefehle. Diese sehen meist eine Vorwärts- oder Rückwärtsbewegung, kombiniert mit einem Einschlagen der Lenkachse nach rechts oder links, vor. Die Steuerung erfolgt dabei mit einer Fernbedienung, welche die Steuerbefehle über elektromagnetische Wellen, oder seltener Infrarotstrahlung, an das Fahrzeug überträgt. Die Kommunikation findet dabei unidirektional statt, die Fernbedienung erhält vom Fahrzeug also keinerlei Rückmeldungen.

In diesem Projekt soll die Funktionalität eines fernsteuerbaren Fahrzeugs erweitert und modifiziert werden. Als Basis dient dazu der Einplatinencomputer Raspberry Pi, der sich aufgrund der umfangreichen Funktionen und des geringen Strombedarfs gut als Steuerzentrale für das Fahrzeug eignet. Um die Kommunikationsmöglichkeiten deutlich zu erweitern, soll die Steuerung über eine Smartphone-App erfolgen, die die Fernbedienung vollständig ersetzt. Einige Steuerbefehle können dabei auch per Sprachbefehl ausgelöst werden. Die App soll außerdem genutzt werden, um dem Bediener Rückmeldungen des Fahrzeugs darzustellen. Zentraler Bestandteil dieser Rückmeldungen soll das Bild einer im Fahrzeug installierten Webcam sein.

Das Fahrzeug soll außerdem in der Lage sein, einige Situationen selbst zu erkennen und darauf entsprechend zu reagieren. So soll die Kollision mit einem Hindernis rechtzeitig erkannt und verhindert werden, außerdem sollen die Scheinwerfer des Fahrzeugs automatisch eingeschaltet werden, wenn es die Lichtsituation erforderlich macht. Die Möglichkeiten des Raspberry Pi erlauben außerdem das Hinzufügen einiger weiterer Funktionen. Über einen mobilen Lautsprecher können Warngeräusche wiedergegeben werden, Kurvenfahrten werden durch Blinker signalisiert und eine impulsartige Ansteuerung der Fahrzeugmotoren erlaubt eine Justierung der Fahrzeuggeschwindigkeit.

2 Fahrzeugmodifikation

In diesem Kapitel werden die Umbaumaßnahmen beschrieben, die zur Erweiterung des Funktionsumfangs am Fahrzeug getroffen werden. Diese sind stark von der Wahl des zu modifizierenden Fahrzeugs abhängig, deshalb wird zuerst auf das für dieses Projekt gewählte Fahrzeug eingegangen. Die anschließend getroffenen Maßnahmen betreffen zunächst die Ansteuerung der Fahrzeugmotoren über den Raspberry Pi, weiterhin wird der Einbau von Ultraschall- und Lichtsensoren sowie Leuchtdioden für Frontscheinwerfer und Blinker besprochen. Die weiteren Erläuterungen betreffen die Installation der Kamera, die Platzierung des Lautsprechers und des Raspberry Pi sowie das Anbringen einer mobilen Stromversorgung.

2.1 Originalfahrzeug

Bei der Wahl des Ausgangsfahrzeugs ist darauf zu achten, dass die Größe des Fahrzeugs ausreicht, um die hinzuzufügenden Komponenten aufzunehmen. Außerdem sollte die Motorleistung groß genug sein, um auch die hinzugefügte Masse noch mit angemessenen Fahreigenschaften bewegen zu können. Die Wahl fällt für dieses Projekt auf das Modell einer Sattelschlepper-



Abbildung 1: Fahrzeug im Originalzustand.

Zugmaschine des Herstellers Wader Toys (Modellbezeichnung: "Wozniak"), siehe Abbildung 1. Freilich handelt es sich bei diesem Fahrzeug um Spielzeug für Kinder, doch durch die großzügigen Ausmaße von $17 \cdot 45 \cdot 20$ cm und die hohe Motorleistung bietet es eine gute Basis für das Hinzufügen weiterer

Komponenten. Da der Aufbau komplett aus Kunststoff besteht, ermöglicht es auch eine leichtgängige Bearbeitung. Einige Komponenten können zudem im Führerhaus untergebracht werden.

Das Fahrzeug ist in seinem Ausgangszustand in der Lage, Vorwärts- und Rückwärtsbewegungen (jeweils mit maximalem Motordrehmoment) auszuführen. Außerdem kann die Position der Lenkachse verändert werden, dabei stehen die Positionen links, geradeaus und rechts zur Verfügung. Die Steuerung dieser Funktionen erfolgt mit der mitgelieferten Funkfernbedienung (Abbildung 2). Für die Vor- und Rückwärtsbewegung wird die vordere



Abbildung 2: Funkfernbedienung.

der beiden Hinterachsen angetrieben, an deren Rädern für bessere Haftung schmale Gummireifen angebracht sind. Solche sind auch an den Rädern der Lenkachse zu finden.

Die Stromversorgung des Fahrzeugs erfolgt über einen 9,6V-Akku mit einer Kapazität von 750mAh, der im Unterbau des Fahrzeugs (schwarzer Kunststoff) hinter einer Klappe platziert ist. In diesem Unterbau, der sich fast über die gesamte Länge des Fahrzeugs erstreckt, sind sämtliche für die Fortbewegung benötigten Bauteile, darunter die Motoren und die Steuerplatine, enthalten. Wird der Akku entfernt, lassen sich alle Schrauben lösen, die den Unterbau mit dem Aufbau verbinden. Nach der Entfernung des Aufbaus ist die Abdeckung der Steuerplatine zugänglich, deren Klammern sich, zum Beispiel durch Anheben mit einem Schraubenzieher, öffnen lassen. Die Steuerplatine kann dann herausgehoben werden. Abbildung 3 zeigt den geöffneten Unterbau.

Das Fahrzeug verfügt ansonsten über keine steuerbaren Bauteile. Bei den Scheinwerfern an der Fahrzeugfront handelt es sich lediglich Aufkleber.

Als rein ästhetische Modifikation wird vor dem Hinzufügen neuer Komponenten eine Lackierung des Fahrzeugs in den Farben der Ruhr-Universität Bochum vorgenommen. Dazu wird Sprühfarbe genutzt, einige Details werden mit Filz- und Lackstiften hinzugefügt.

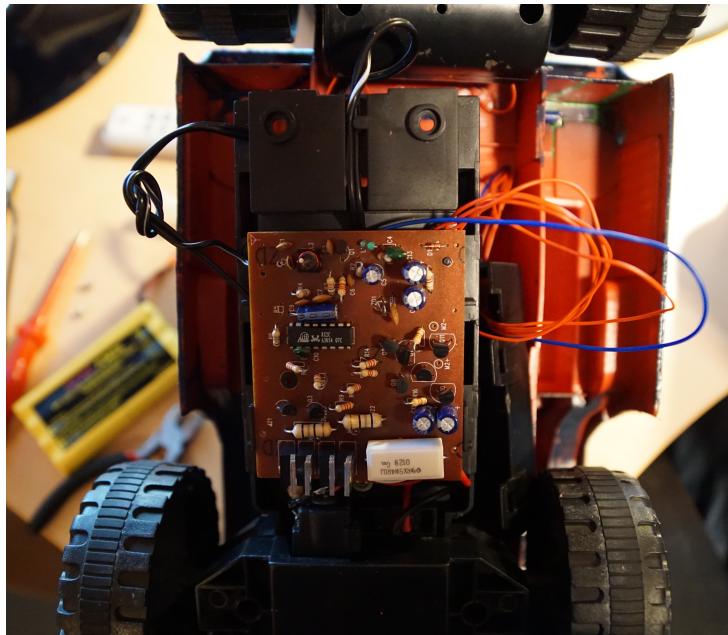


Abbildung 3: Geöffneter Unterbau mit freigelegter Steuerplatine.

2.2 Neue Komponenten

Im folgenden werden sämtliche Bauteile, Geräte und Materialien aufgeführt, die in diesem Projekt Verwendung finden und nicht bereits Teil des Fahrzeugs sind.

Raspberry Pi 2 Zentraler Bestandteil dieses Projekts ist der Einplatinencomputer Raspberry Pi 2 Model B (siehe Abbildung 4), der die Ansteuerung der Motoren und Leuchtdioden, die Auswertung der Sensordaten und die Kommunikation mit dem Smartphone übernimmt. Er kann über ein USB-Kabel mit Strom versorgt werden und hat bei 5V Versorgungsspannung einen maximalen Ausgangsstrom von 800mA. Für dieses Projekt wird er in einem passenden Gehäuse betrieben.

WLAN-Adapter Da der Raspberry Pi standardmäßig nur über eine kabelgebundene Netzwerkschnittstelle (LAN) verfügt, wird ein WLAN-Adapter mit USB-Schnittstelle benötigt. Für dieses Projekt wird ein EDIMAX EW-7811UN eingesetzt, der sich durch eine kompakte Bauform auszeichnet (siehe Abbildung 5).



Abbildung 4: Raspberry 2 Model B.



Abbildung 5: WLAN-Adapter.

Steckbrett Bei diesem Projekt werden später einige Signalleitungen mit der frei programmierbaren Schnittstelle (auch GPIO, *General Purpose Input/Output*, genannt) des Raspberry Pi verbunden. Für die Ansteuerung der Leuchtdioden müssen außerdem Widerstände in Reihe geschaltet werden. Es bietet sich daher an, ein kleines Steckbrett (Abbildung 6) als Anlaufstelle für alle Leitungen zu verwenden.

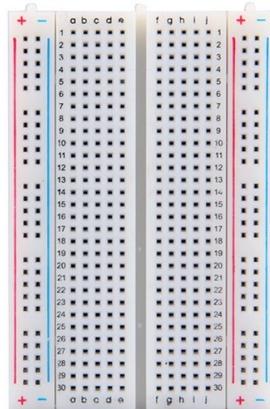


Abbildung 6: Steckbrett.

T-Kupplung und Flachbandkabel Wird auf das Steckbrett eine T-Kupplung aufgesteckt und über das Flachbandkabel mit der GPIO-Schnittstelle des Raspberry Pi verbunden, können die Pins der Schnittstelle leicht über das Steckbrett zugänglich gemacht werden. Abbildung 7 zeigt die Einzelteile der T-Kupplung (links) und das Flachbandkabel.



Abbildung 7: T-Kupplung und Flachbandkabel.

Ultraschallsensoren Für dieses Projekt werden vier Ultraschallsensoren des Typs HC-SR04 (siehe Abbildung 8) für Abstandsmessungen genutzt. Dieser Typ bietet laut Datenblatt einen Erfassungsbereich von 2 - 450 cm bei einem maximalen Winkel von 15°. Die Ultraschallsensoren benötigen jeweils eine Versorgungsspannung von 5V (Gleichspannung) und zwei Signalleitungen für Trigger- und Echosignale.

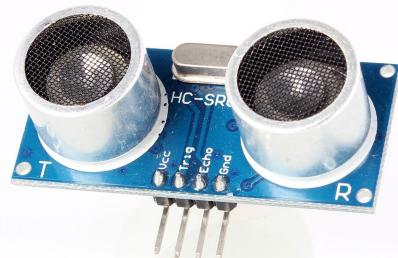


Abbildung 8: Ultraschallsensor (Typ HC-SR04).

Leuchtdioden Das Fahrzeug wird mit vier weißen (als Frontscheinwerfer) und zwei gelben (als Blinker) Leuchtdioden ausgestattet. Es handelt sich dabei um handelsübliche Leuchtdioden mit ungefähren Durchlassspannungen von $U = 3,5V$ (weiß) bzw. $U = 2,2V$ (gelb).

Elektrische Widerstände Es werden drei ohmsche Widerstände benötigt, davon zwei 330Ω - und ein $10k\Omega$ -Widerstand.

Fotowiderstand Zur Erkennung der Lichtsituation wird ein Fotowiderstand, wie in Abbildung 9 abgebildet, verwendet.

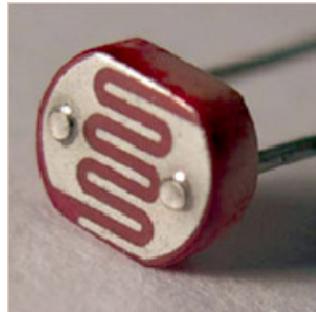


Abbildung 9: Fotowiderstand.

Webcam Für dieses Projekt wird eine QuickCam S5500 (Abbildung 10) des Herstellers Logitech verwendet, die über die USB-Schnittstelle an den Raspberry Pi angeschlossen werden kann.



Abbildung 10: Webcam.

Mobiler Lautsprecher Um Warntöne und andere Geräusche wiederzugeben, wird ein mobiler Lautsprecher des Herstellers Beats Electronics (Modell: Beatbox Mini, siehe Abbildung 11) eingesetzt. Dieser Lautsprecher verfügt über einen eigenen Akku und kann über einen 3,5mm-Klinkenstecker mit

dem Audioausgang des Raspberry Pi verbunden werden. Prinzipiell eignet sich dieser Lautsprecher auch für die Musikwiedergabe.



Abbildung 11: Mobiler Lautsprecher.

Externer Zusatzakku Für die Energieversorgung des Raspberry Pi wird ein externer Zusatzakku (auch Powerbank genannt) eingesetzt. In diesem Fall handelt es sich um ein Powerpack T10400 des Herstellers PNY Technologies (siehe Abbildung 12). Dieses Modell hat eine Kapazität von 10,4Ah und kann bei 5V einen maximalen Ausgangsstrom von 2,4A liefern.



Abbildung 12: Externer Zusatzakku.

Weiteres Die Verbindungen zwischen Sensoren, Leuchtdioden, Steuerplatine und Steckbrett werden über Steckbrett-Steckbrücken oder einfache isolierte Kupferkabel realisiert. Für die Befestigung Letzterer wird ein Lötkolben verwendet. Weitere verwendete Werkzeuge sind Schraubenzieher, ein Akkuschrauber mit Bohraufsatz und eine Rundfeile. Außerdem werden eine Heißklebepistole und ein Multimeter genutzt. Für die Befestigung des externen Zusatzakkus werden zudem Klettverschluss-Kabelbinder verwendet.

2.3 Ansteuerung der Fahrzeugmotoren

Für eine Ansteuerung der Fahrzeugmotoren über die GPIO-Schnittstelle des Raspberry Pi sind zwei verschiedene Methoden denkbar: Zunächst wäre es

möglich, eine eigene Verstärkerschaltung zu bauen oder einen vorgefertigten Motortreiber zu verwenden. Diese Schaltungen müssen dann jeweils für ein Treiben des für den Motorbetrieb benötigten Stroms ausgelegt sein. Die zweite Möglichkeit besteht darin, die bereits vorhandene Steuerplatine des Fahrzeugs zu untersuchen und gezielt die Eingänge der vorhandenen Verstärkerschaltungen anzusteuern. Dabei muss beachtet werden, dass der Raspberry Pi über seine Ausgänge nur einen begrenzten Strom von etwa 16 mA treiben kann, eine direkte Ansteuerung der Motoren über diese Ausgänge kommt daher nicht in Frage (der Raspberry Pi könnte dadurch sogar beschädigt werden).

Die Entscheidung fällt für dieses Projekt auf die zweite Methode, da diese einen geringeren Aufwand erscheinen lässt.

Analyse der Steuerplatine Wie im Abschnitt “Originalfahrzeug” beschrieben, wird zunächst der Unterbau vom Aufbau getrennt und Zugang zur Steuerplatine geschaffen. Eine erste Sichtung der Platine ergibt, dass sich die Bauteile grob drei verschiedenen Gruppen zuordnen lassen (Abbildung 13): Der rechte Teil, zu dem einige Kondensatoren gehören, ist vermutlich dem

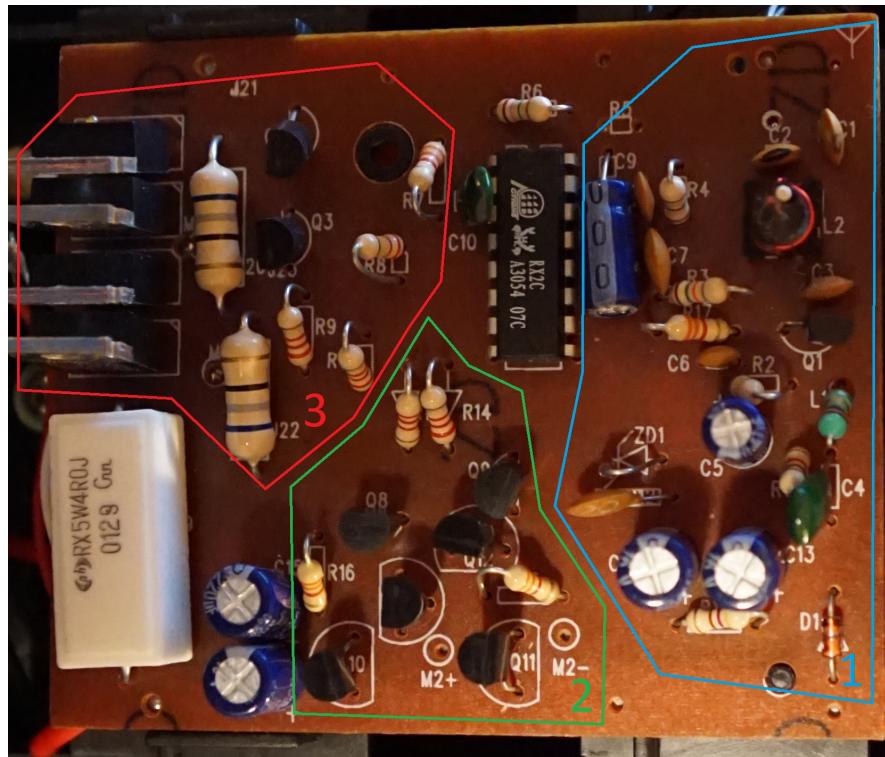


Abbildung 13: Grobe Zuordnung der Funktionsgruppen auf der Steuerplatine.

Empfang des Funksignals zuzuordnen, da hier auch eine Antenne angebracht ist. Im unteren Bereich der Platine ist an den mit M2+ und M2- beschrifteten Stellen auf der Unterseite eine Leitung angebracht, die zum Stellmotor für die Lenkachse führt. Die Transistoren in der Nähe dieser Stellen sind also der Verstärkung des Lenksignals zuzuordnen. Bei der Untersuchung der dritten Gruppe fallen die großen Leistungstransistoren auf. Dieser Teil scheint verantwortlich für den Betrieb des Fahrmotors zu sein, der offensichtlich die größte Leistungsaufnahme zeigt.

Zentrales Bindeglied der drei Gruppen ist der Controller-Chip in der Mitte der Platine. Eine Ansteuerung der Transistoren erscheint also an dieser Stelle sinnvoll, zumal dieser Chip vermutlich nicht in der Lage ist, große Ströme zu treiben.

Als nächstes muss nun festgestellt werden, welche Anschlüsse des Controller-Chips jeweils welche Funktionen steuern. Dazu wird die Platine in Betrieb genommen, in dem der Akku wieder angeschlossen wird. Anschließend wird mit einem Multimeter das Potential der Controller-Anschlüsse gegenüber dem negativen Versorgungspotential (das im Übrigen auf der Platine als Ground, also Masse, gekennzeichnet ist) gemessen. Dann werden mit der Funkfernbedienung Steuerbefehle gesendet und die Anschlüsse auf Potentialänderungen untersucht. Auf diese Weise können die vier einer Motorsteuerung zugehörigen Anschlüsse identifiziert werden.

Verbindung mit den Ausgängen des Raspberry Pi Die einer Motorsteuerung zugehörigen Lötstellen auf der Unterseite der Platine werden mit dem Lötkolben erhitzt und jeweils ein Kabel angebracht. Ein weiteres Kabel wird auf diese Weise am Massepotential der Platine angebracht. Abbildung 14 zeigt die Unterseite der Steuerplatine mit den neu angebrachten Leitungen, die jeweils durch einen Kreis hervorgehoben werden.

Die Kabel werden anschließend durch ein Loch auf der Unterseite des Führerhauses geführt, das gegebenenfalls noch mit dem Akkuschrauber vergrößert werden muss. Die Kabelenden werden mit dem Stift einer Steckbrücke versehen und auf dem Steckbrett einem der ansteuerbaren GPIO-Ausgänge zugeordnet (zuvor wurde das Steckbrett über eine T-Kupplung und ein Flachbandkabel mit der GPIO-Schnittstelle des Raspberry Pi verbunden). Das Kabelende des Massepotentials wird auf einen negativen Anschluss der Versorgungsspannung des Raspberry Pi auf dem Steckbrett gesteckt. Die Zuordnung der jeweiligen Funktionen erfolgt später im Programmcode des Raspberry Pi (siehe Abschnitt 3.2.2).

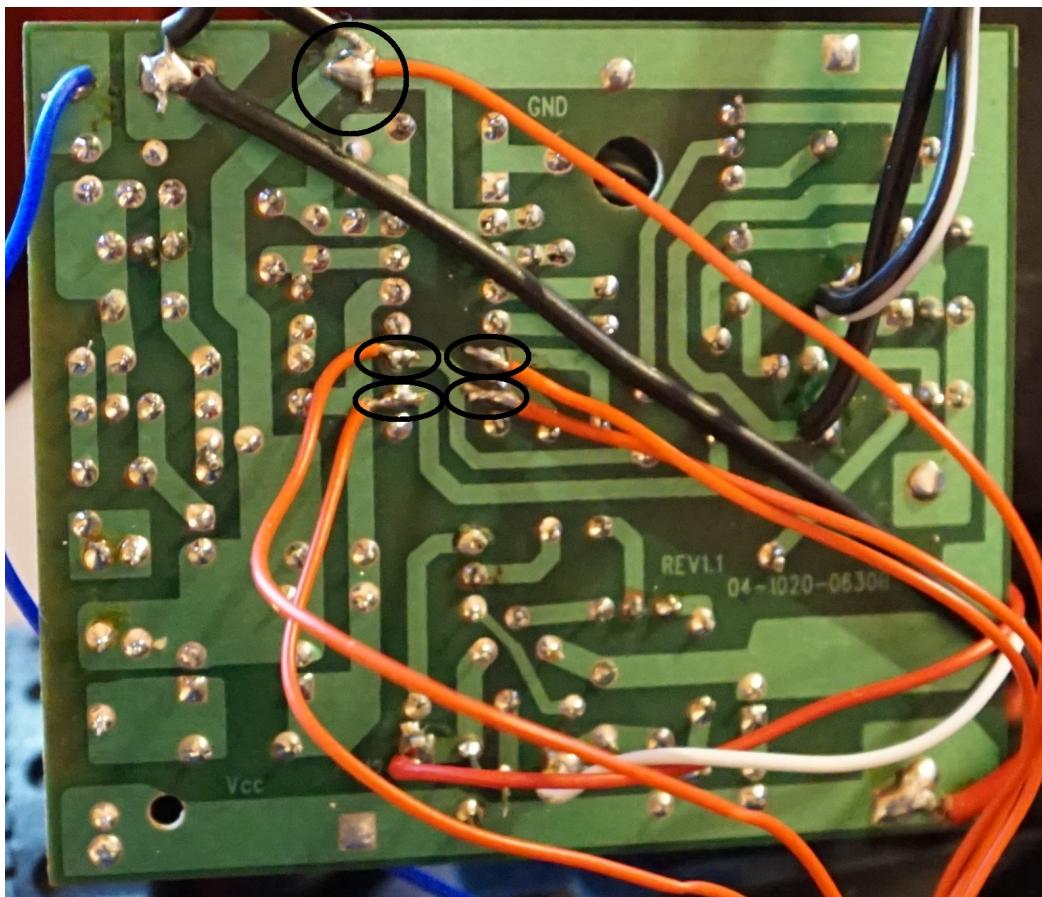


Abbildung 14: Unterseite der Steuerplatine mit den neu angebrachten Leitungen.

2.4 Installation der Scheinwerfer und Blinker

Das Fahrzeug soll mit funktionsfähigen Frontscheinwerfern und Blinkern ausgestattet werden. Aufgrund der hohen Lichtausbeute bei geringen Strömen und kompakten Ausmaßen eignen sich hierfür gut Leuchtdioden (LEDs). Das Fahrzeug verfügt an der Front über jeweils zwei Aussparungen an jeder Seite, die ursprünglich mit einem Aufkleber die Attrappe eines Scheinwerfers darstellen sollten. In diesem Projekt sollen diese Scheinwerfer mit einer Funktion versehen werden. Die Scheinwerfer sollen außerdem abhängig von der Lichtsituation automatisch eingeschaltet werden. Hierfür bietet sich der Einsatz eines Fotowiderstands an, der auf dem Dach des Führerhauses angebracht werden soll, da hier eine gute Erfassung der Lichtsituation möglich ist. Weiterhin sollen Kurvenfahrten mit Blinkern angezeigt werden. Diese Blinker sollen in der Stoßstange angebracht werden.

Einbau der Frontscheinwerfer Zunächst wird, wie in Abschnitt 2.1 beschrieben, der Unterbau entfernt. Mit einem 6mm-Bohrer (entspricht ungefähr dem Durchmesser der Leuchtdioden) wird dann in jede der vier Aussparungen der Scheinwerferattrappen ein möglichst mittig platziertes Loch gebohrt. Meist ist im Anschluss eine Entgratung erforderlich. Bevor nun die vier weißen Leuchtdioden in diese Löcher geschoben werden, werden die Leuchtdioden verkabelt. Es ist vorgesehen, die Dioden parallel zu schalten. Dies ist nicht unbedingt ein ideales Verfahren, da die Dioden theoretisch unterschiedlich belastet werden können. Eine Reihenschaltung ist jedoch nicht möglich, da dann die Durchlassspannungen (ca. 3,5V pro weißer LED) addiert werden und ein Betrieb nur bei ungefähr 14V Versorgungsspannung möglich wäre. Bei Parallelschaltung genügen hingegen die 5V, die der Raspberry Pi an seinen Ausgängen liefern kann.

Bei der Verkabelung werden die Spannungs- und Massepotentiale zu gemeinsamen Knoten zusammengeführt, von denen dann jeweils eine weitere Leitung durch ein Loch im Führerhaus zum Steckbrett geführt wird. Um eine feste Arretierung an den Anschlüssen der Leuchtdioden und den Knotenpunkten zu erreichen, wird hier jeweils verlötet. Auf dem Steckbrett wird die eine Masseleitung mit dem negativen Versorgungspotential des Raspberry Pi verbunden. Zu der anderen Leitung, die das Versorgungspotential der Dioden führt, wird ein 330Ω -Widerstand in Reihe geschaltet, um Strom und Spannung zu begrenzen. Dieser Widerstand wird dann wiederum mit einem steuerbaren Ausgang des Raspberry Pi verbunden.

Nachdem die LEDs durch die Löcher geschoben wurden, werden sie (auf der Innenseite des Aufbaus) mit etwas Klebstoff aus der Heißklebepistole fixiert.

Einbau der Blinker Die Blinker werden auf gleiche Weise wie die Frontscheinwerfer eingebaut, als Platz dafür dienen diesmal die seitlichen Flanken der Stoßstange. Für die Blinker werden gelbe Leuchtdioden verwendet, jeweils eine auf jeder Seite. Damit die Blinker separat angesteuert werden können, müssen beide positiven Anschlüsse mit einem eigenen GPIO-Anschluss (und natürlich jeweils einem Serienwiderstand, hier 330Ω) verbunden werden. Dadurch entfällt auch das bei den Frontscheinwerfern auftretende Verschaltungsproblem, da keine Parallelschaltung eingesetzt werden muss. Als Massepotential kann der bereits vorhandene Knotenpunkt der Frontscheinwerfer genutzt werden. Abbildung 15 zeigt Frontscheinwerfer und Blinker auf der rechten Fahrzeugseite, in Abbildung 16 wird die Funktion demonstriert (weitere Komponenten wurden hier bereits eingebaut).

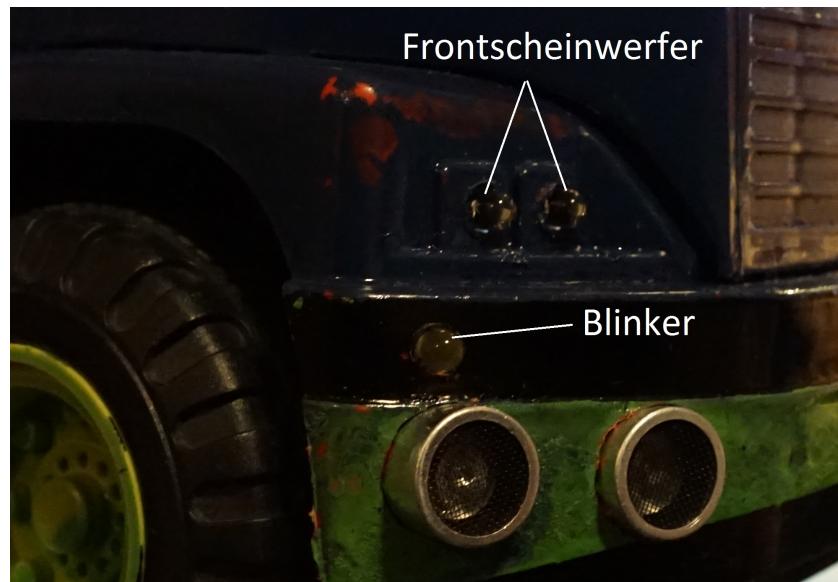
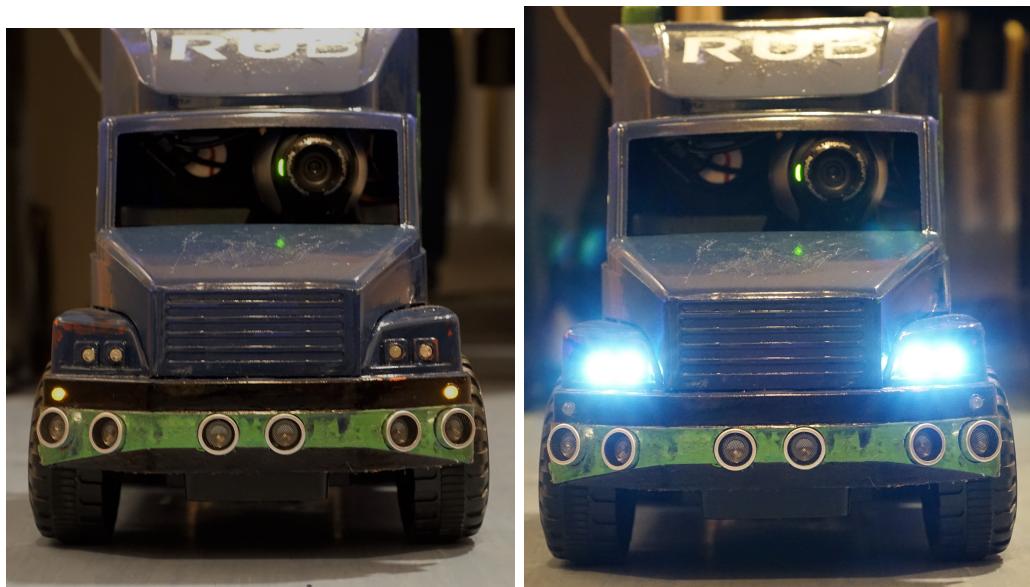


Abbildung 15: Leuchtdioden auf der rechten Fahrzeugseite.



(a) Fahrzeugfront mit eingeschalteten Blinkern.
(b) Fahrzeugfront mit eingeschalteten Frontscheinwerfern.

Abbildung 16: Demonstration der Scheinwerfer und Blinker.

Einbau des Fotowiderstands Je mehr Licht auf einen Fotowiderstand fällt, desto kleiner wird sein elektrischer Widerstand. Diese Eigenschaft soll

genutzt werden, um mit Hilfe des Raspberry Pi zu erkennen, ob es die Lichtsituation erforderlich macht, die Frontscheinwerfer einzuschalten. Der Raspberry Pi verfügt allerdings nicht über einen Analog-Digital-Wandler, es ist also (ohne zusätzliche Bauteile) nicht möglich, die Lichtintensität als absoluten Wert zu messen. Eine Unterscheidung zwischen zwei Zuständen (hell und dunkel) ist allerdings möglich. Der Raspberry Pi registriert dann eine Spannung an einem seiner GPIO-Anschlüsse, sobald ein bestimmter Schwellwert überschritten wird. Dieser Schwellwert lässt sich nach unten, also in Richtung einer geringeren Lichtintensität, korrigieren, in dem ein weiterer elektrischer Widerstand in Reihe zum Fotowiderstand geschaltet wird. Bevor nun auf diese Weise eine Justierung erfolgt, erfolgt der Einbau in das Dach des Führerhauses. In diesem Fall wird mit einem 5mm-Bohrer ein Loch in die Nähe der stilisierten Schornsteine gebohrt. Der Fotowiderstand wird an eine zweipolige Steckbrett-Steckbrücke angeschlossen und in das Loch gesteckt. Hier wird er dann mit der Heißklelepistole fixiert, der Klebstoff wird hierbei auch über die Oberfläche des Fotowiderstands verteilt, um einen Schutz vor äußeren Einflüssen zu erreichen. Zu den Anschlüssen der Steckbrücke wird nun ein Widerstand in Reihe geschaltet, der so gewählt wurde, dass der Schwellwert für die Lichterkennung auf einem angemessenen Niveau liegt. Beides wird dann mit einem GPIO-Anschluss und einem positiven 5V-Versorgungspotential auf dem Steckbrett verbunden. Abbildung 17 zeigt den in das Fahrzeugdach eingeklebten Fotowiderstand.



Abbildung 17: Fotowiderstand, ins Fahrzeugdach eingeklebt.

2.5 Installation der Ultraschallsensoren

Ein formuliertes Ziel ist es, Kollisionen des Fahrzeugs mit Hindernissen zu vermeiden. Um diese Hindernisse erkennen zu können, werden Ultraschallsensoren eingesetzt. Mit diesen können Hindernisse in einem bestimmten Winkel vor ihnen erkannt werden, dieser Winkel ist jedoch meist relativ klein (in diesem Fall nicht mehr als 15°). Um einen größeren Bereich, vor allem bei Kurvenfahrten, abdecken zu können, empfiehlt sich die Verwendung mehrerer Ultraschallsensoren. Der Aufbau der Fahrzeugfront macht dies relativ gut möglich, da hier eine breite Stoßstange vorhanden ist. Generell empfiehlt sich ein Einbau der Sensoren in geringem Abstand zum Boden, um auch niedrige Hindernisse erkennen zu können. Am Heck des Fahrzeugs gestaltet sich der Einbau jedoch schwieriger, da dieses von den großen Reifen dominiert wird und in der Mitte eine Art Anhängerkupplung den Platz versperrt. Es ist daher geplant, die Fahrzeugfront mit drei Ultraschallsensoren auszustatten, während am Heck nur ein einzelner angebracht wird. Da das Fahrzeug vermutlich die meisten Bewegungen in Vorwärtsrichtung ausführen wird, erscheint dies vertretbar. Abbildung 18 zeigt schematisch die durch die Sensoren abgedeckten Bereiche.

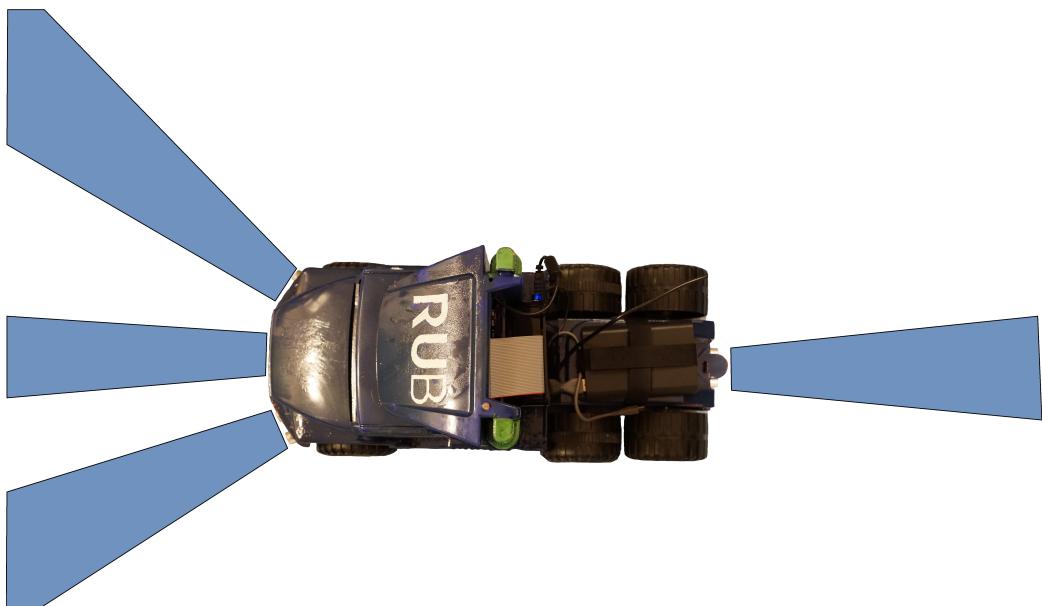


Abbildung 18: Schematische Darstellung der Abdeckung durch die Ultraschallsensoren.

Einbau Frontsensoren Die drei Ultraschallsensoren an der Fahrzeugfront werden so platziert, dass der mittlere Sensor genau geradeaus und die anderen Sensoren etwas seitlich ausgerichtet sind. Mit einem 10mm-Bohrer werden dann in die Stoßstange insgesamt sechs Löcher für die Schallwandler gebohrt, die anschließend mit der Rundfeile so weit vergrößert werden, dass die Schallwandler gerade hindurchgesteckt werden können. Dadurch wird gewährleistet, dass die Wandler später nicht mehr verrutschen. Für die seitlichen Sensoren muss gewährleistet sein, dass das sie nicht mit den Rädern der Lenkachse kollidieren, wenn diese eingeschlagen wird. Gegebenenfalls muss dann noch mit der Rundfeile auf der Innenseite der Stoßstange etwas Platz geschaffen werden, sodass die Sensoren weiter nach vorne rutschen können. An die Anschlüsse für die Stromversorgung der Sensoren (Vcc) werden Kabel angelötet, die auf einen gemeinsamen Knoten geführt werden. Dieser wird wiederum mit der 5V-Versorgungsspannung des Raspberry Pi auf dem Steckbrett verbunden, die Leitung wird wieder durch das Führerhaus geführt. Für das Massepotential (Gnd) kann der bereits beim Einbau der Leuchtdioden entstandene Knotenpunkt genutzt werden. Die beiden Signalanschlüsse Trig und Echo werden für jeden Sensor über jeweils eine zweiseitige Steckbrücke auf jeweils zwei GPIO-Anschlüsse auf dem Steckbrett (ebenfalls durch das Führerhaus) geführt.

Nach dem Anschluss sämtlicher Kabel werden die Sensoren mit der Heißklebefistole fixiert. Abbildung 19 zeigt die Leitungsverläufe.

Einbau Hecksensor Der einzelne Sensor am Fahzeugheck wird so platziert, dass er bei Rückwärtsfahrt geradeaus ausgerichtet ist. Da am Fahrzeug der Platz etwas begrenzt ist, wird der Sensor unterhalb des kleinen Vorbau platziert. Mit der Rundfeile werden dazu leichte Einbuchtungen für die Schallwandler geschaffen. Dabei ist jedoch zu beachten, dass die Schallwandler nicht auf den kleinen Vorbau gerichtet sind (und diesen eventuell als Hindernis erkennen). Der Anschluss erfolgt wie bei den Frontsensoren, die Kabel werden dabei durch den Zwischenraum zwischen Fahrzeugmotor und Aufbau geführt. Abbildung 20 zeigt das Steckbrett mit allen angeschlossenen Kabeln. Bei der Fixierung sollte nicht an Klebstoff gespart werden, um einen festen Halt zu garantieren.

2.6 Geräteinstallationen

Zu den weiteren Teilen des Projekts gehören eine Webcam und ein Lautsprecher, außerdem muss für den Raspberry Pi eine mobile Stromversorgung geschaffen und das Steckbrett untergebracht werden. Webcam, Lautsprecher und Steckbrett sollen dabei im Führerhaus untergebracht werden.

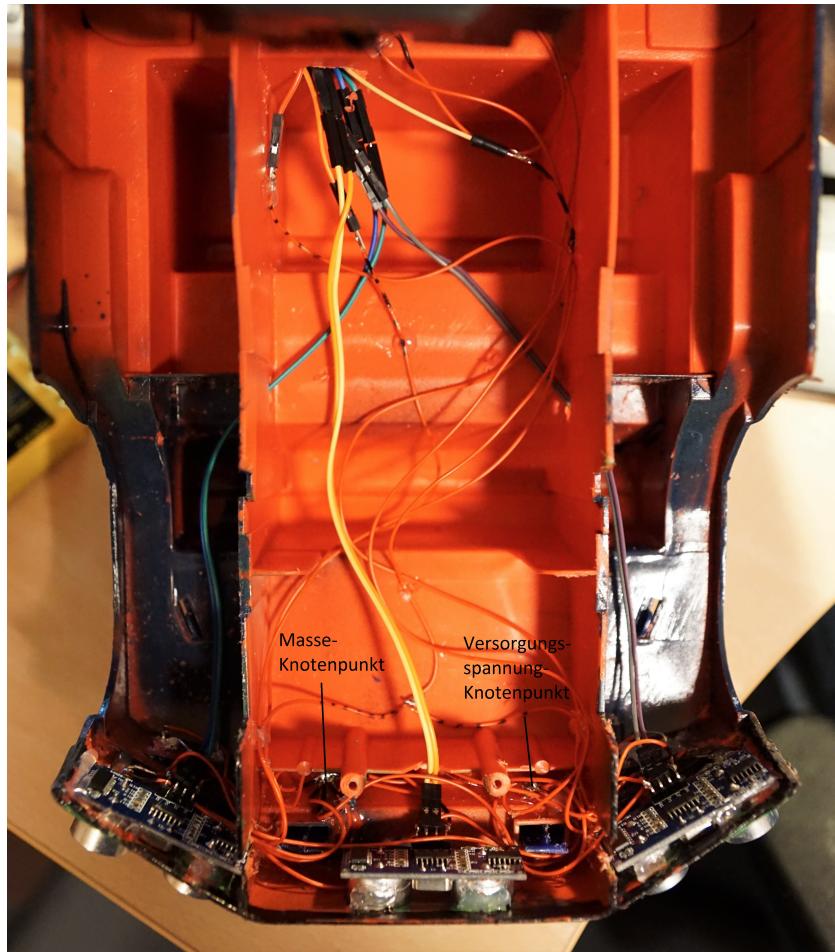


Abbildung 19: Kabelverläufe und Knotenpunkte.

Einbau Steckbrett Das Steckbrett stellt durch die mittlerweile große Zahl an aufgesteckten Kabeln eines der sperrigsten Teile dar, daher empfiehlt es sich, dieses zuerst im Führerhaus unterzubringen. Es wird dazu durch das hintere Fenster in die Fahrerkabine geführt und dann senkrecht in den Spalt zwischen Fahrersitz und Rückwand des Führerhauses gesteckt. So bleibt genug Platz für die weiteren Geräte. Beim Hereinführen muss darauf geachtet werden, dass sich keines der aufgesteckten Kabel löst. Das Flachbandkabel wird so weit in die Fahrerkabine geführt, dass der Raspberry Pi, der auf der Rückseite des Führerhauses unter dem Fenster platziert wird, daran hängt. Später wird es dann durch den Fuß der Webcam fixiert.

Einbau Webcam Die Webcam soll Bilder aus der “Fahrerperspektive”, also aus dem inneren des Führerhauses, aufnehmen. Dazu wird sie durch das

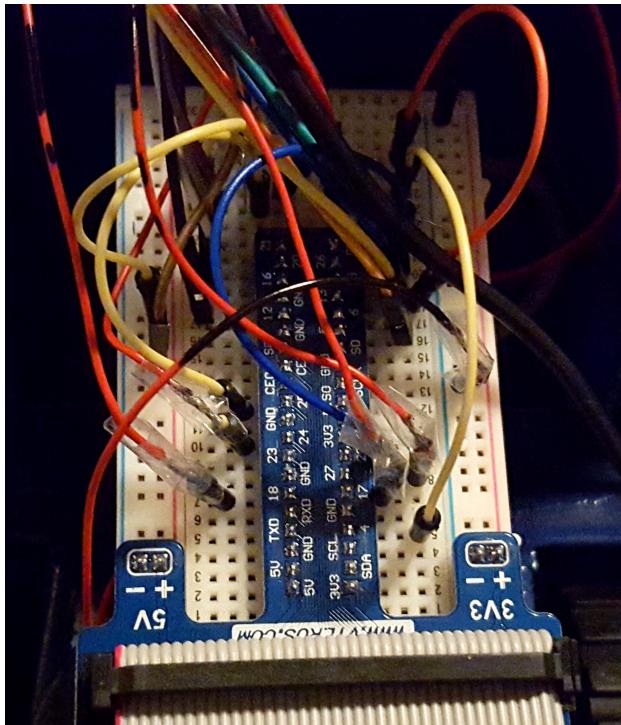


Abbildung 20: Steckbrett mit allen aufgesteckten Leitungen.

vordere Fenster in die Fahrerkabine geführt. Die hier verwendete Logitech-Kamera verfügt über einen verformbaren Standfuß, mit dem sie sich im Führerhaus fixieren lässt. Da der Fuß relativ lang ist, kann er auch genutzt werden, um das Flachbandkabel zu arretieren. Das USB-Kabel der Webcam wird durch das hintere Fenster geführt und am Raspberry Pi angeschlossen.

Einbau Lautsprecher Der Lautsprecher wird ebenfalls durch das vordere Fenster geführt und im hinteren Teil der Fahrerkabine, neben dem Steckbrett, platziert. Dabei wird er so gedreht, dass die Bedienelemente später durch das hintere Fenster erreichbar sind. Das Audiokabel wird dann ebenfalls durch das hintere Fenster geführt und an den Audioausgang des Raspberry Pi angeschlossen. Abbildung 21 zeigt die Platzierung von Lautsprecher und Webcam.

Einbau Stromversorgung Der Raspberry Pi benötigt für den Betrieb eine Spannung von 5V, wie sie beispielsweise über eine USB-Schnittstelle bereitgestellt werden kann. Daher lässt sich auch ein externer Zusatzakku (Powerbank) nutzen, der normalerweise beim Aufladen von Smartphones und Tablets Verwendung findet. Am Fahrzeug soll der Zusatzakku auf der Hal-



Abbildung 21: Webcam und Lautsprecher im Führerhaus.

tevorrichtung platziert werden, die normalerweise den Auflieger hält. Dort wird er mit Klettverschluss-Kabelbindern fixiert. Der Ladeausgang wird anschließend über ein entsprechendes Kabel mit dem Raspberry Pi verbunden. Letzterer wird damit schlussendlich nur durch Kabel gehalten (Abbildung 22), dies soll für dieses Projekt jedoch genügen. Es erlaubt außerdem eine schnelle Entnahme des Raspberry Pi zu Wartungszwecken.

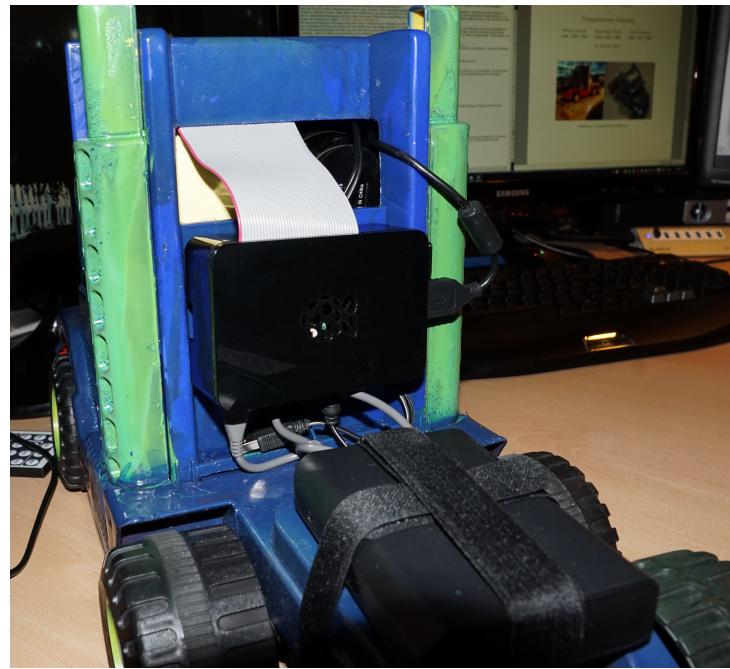


Abbildung 22: Rückansicht des Fahrzeugs mit montiertem Raspberry Pi und Zusatzakku.

3 Einrichtung des Raspberry Pis

Der Raspberry Pi hat zum einen die Aufgabe die von der Android App empfangenen Befehle umzusetzen indem dieser die Motoren oder LEDs über die GPIO-Ports ansteuert und zum anderen soll dieser die Fahrt überwachen und in gefährlichen Situationen eingreifen. Dabei werden die Sensordaten von den vier angebrachten Entfernungssensoren und dem Lichtsensor erfasst und ausgewertet. So soll etwa gewährleistet werden, dass das Fahrzeug vor Hindernissen abbremst oder das Vorderlicht (hier die selbst angebrachten LEDs) bei Dunkelheit eingeschaltet wird. Außerdem sollen Hub- und Warngeräusche ausgegeben werden.

3.1 Externe Software

3.1.1 Motion (Webcam Live-Übertragung)

Um das Bild der Webcam in der Smartphone-App anzeigen zu können, muss auf dem Raspberry Pi zunächst die eine Software installiert werden. Für dieses Projekt wird die motion¹ Software verwendet. Sie überträgt ein Video der angeschlossenen Webcam, indem sie in kurzen Abständen Bilder im JPEG-Datenformat auf einem lokalen Speicher aktualisiert. Für dieses Projekt wird die motion Software so konfiguriert, dass über die IP-Adresse des Raspberry Pi und einen eingestellten Port (hier Port 8081) auf diese Bilder zugegriffen werden kann.

3.1.2 OMXPlayer

Um die Audiodateien für die Hub- und Warngeräusche abzuspielen wird der OMXPlayer² verwendet, welcher speziell für den Raspberry Pi entwickelt worden ist und vom Programm direkt gestartet werden kann.

¹motion: <http://pimylifeup.com/raspberry-pi-webcam-server/>

²OMXPlayer: <https://github.com/popcornmix/omxplayer>

3.2 Programmierung

3.2.1 Einleitung

Die Programmierung der Software wird in der Programmiersprache C durchgeführt, wobei manche verwendeten libraries auf C++ basieren. Dabei wird g++ als Compiler und VIM³ als Editor verwendet, welcher eine sehr praktische Bedienung aufweist und sich aufgrund eines enormen Funktionsumfangs hervorragend für die Programmierung im Terminal eignet. Die Programmierung und Kompilierung im Terminal hat den Vorteil, dass sämtliche Arbeitsschritte direkt auf dem Raspberry Pi über SSH erfolgen können und kein ständiger Upload bei Programmänderungen von Nöten ist. Außerdem ist diese Methode unabhängig von dem Computer, sodass die Arbeit von beliebigen Geräten aus stattfinden kann ohne immer alle Dateien transferieren zu müssen. Lediglich ein SSH-Client ist notwendig, wobei hier „PUTTY“ verwendet wird.

Auf dem Raspberry Pi ist das Betriebssystem Raspbian⁴ installiert. Dabei handelt es sich um eine für den Raspberry Pi optimierte Debian-Distribution. Gearbeitet wird in dem Verzeichnis /home/pi/rc-control/.

Der vollständige Programmcode ist im Anhang ab Seite 33 aufgeführt.
Das Programm lässt sich in fünf Teile unterteilen:

1. Setup
2. Herstellung der Socket-Verbindung zum Steuergerät
3. Empfang und Umsetzung der Befehle
4. Auswertung der Sensordaten und eventuelles Abbremsen des Fahrzeugs
5. Übertragung des aktuellen Status an die Android App im JSON-Format

3.2.2 Setup

Zuerst werden in einer Headerdatei (hier: constants.h - Seite 33) alle Konstanten wie etwa die Pin-Belegung oder die Zahlenwerte zu den einzelnen Befehlen festgelegt, sodass sich diese innerhalb dieser Datei leicht ändern lassen ohne immer den gesamten Code durchgehen zu müssen. Eine Übersicht der meisten Befehl-Zahlen-Paare ist in Tabelle 2 dargestellt bzw. ist aus dem Quellcode auf Seite 33 zu entnehmen. Anschließend wird in einer

³VIM: <http://www.vim.org/>

⁴Raspbian: <https://www.raspbian.org/>

weiteren Headerdatei eine Struktur (struct) erzeugt, in welcher alle wichtigen Daten zum aktuellen Zustand festgehalten werden. (Seite 34) Dazu gehören unter anderem die Werte für die aktuelle Geschwindigkeit oder die zuletzt gemessenen Sensordaten. Anschließend werden zu Beginn des Programms die einzelnen GPIO-Ports des Raspberry Pis jeweils in den Ausgangs- (z.B. die Ports für die Motorsteuerung) bzw. Eingangsmodus (z.B. die Ports an denen die Sensordaten erfasst werden müssen) versetzt und Anfangszustände, wie etwa die Startgeschwindigkeit, festgelegt. (Seite 37 Zeilen: 30-90) Für das Arbeiten mit den GPIO-Ports wird die library WiringPi⁵ genutzt. Dank der am Anfang erzeugten Headerdatei constants.h, werden für die einzelnen Pins die dort definierten Bezeichnungen verwendet, was das Arbeiten im Code deutlich erleichtert. So wird etwa anstatt von

```
pinMode(5, OUTPUT);
```

folgende Anweisung benutzt:

```
pinMode(PIN_LEFT, OUTPUT);
```

Die Vollständige GPIO-Pin-Zuordnung ist in Tabelle 1 dargestellt:

GPIO-Pin	Zuordnung
0	Motor: Rückwärts
1	Motor: Vorwärts
2	Lenkung: Rechts
5	Lenkung: Links
3	Photowiderstand
29	Entfernungssensor vorne-rechts: ECHO
28	Entfernungssensor vorne-rechts: TRIGGER
21	Entfernungssensor vorne: ECHO
22	Entfernungssensor vorne: TRIGGER
25	Entfernungssensor vorne-links: ECHO
24	Entfernungssensor vorne-links: TRIGGER
27	Entfernungssensor hinten: ECHO
26	Entfernungssensor hinten: TRIGGER
6	LEDs vorne
7	Blinker rechts
23	Blinker links

Tabelle 1: Zuordnung der einzelnen GPIO-Pins

⁵WiringPi: <http://wiringpi.com/>

3.2.3 Herstellung der Socket-Verbindung zum Steuergerät

Zum Herstellen der Socket Verbindung zum Steuergerät (hier: Android Smartphone mit selbstgeschriebener App) ist die sys/socket.h⁶ Headerdatei zu importieren. Anschließend wird ein Server-Socket erstellt, welcher nach der Durchführung des Setups auf die Verbindung eines Clients (Hier das Smartphone) wartet. (Seite 37 Zeilen: 110-136)

3.2.4 Empfang und Umsetzung der Befehle von der Android App

Für alle nun verwendeten Funktionen, welche die Steuerung des Fahrzeugs betreffen wird der Übersicht halber eine neue Datei „functions.c“ (Seite 38) erstellt und in die Hauptdatei importiert.

Nach der erfolgreichen Verbindung zwischen Smartphone und Raspberry Pi über den Socket, wird ein Thread (unter Verwendung von pthread.h⁷) gestartet, welcher auf Anweisungen vom Smartphone wartet um diese umzusetzen. Der vom Smartphone übertragene Befehl entspricht dabei einer Zahl zwischen 0 und 255 (1 Byte), dabei wurde in der Headerdatei „constants.h“ festgelegt welche Zahl welchem Befehl entspricht. (Tabelle 2 auf Seite 27 bzw. Quelltext auf Seite 33). Zur besseren Separation werden hier Anweisungen, welche sich nicht auf die Geschwindigkeit oder die Socket-Verbindung beziehen Zahlen in dem Zahlenraum zwischen 0 und 70 zugeordnet.

Diese Befehle, wie etwa das Schalten der LEDs oder die Lenkung, werden direkt umgesetzt und im Status-struct Objekt „currentStatus“ für eventuelle Abfragen hinterlegt. Somit lässt sich z.B. zu jeder Zeit herausfinden ob das Licht gerade eingeschaltet ist. 77 entspricht dem Beenden der Socket-Verbindung und Nummern zwischen 110 und 120 geben die gewünschte Geschwindigkeit an, wobei 110 für den fünften Rückwärtsgang und 120 für den fünften Vorwärtsgang steht. Zum Anhalten des Fahrzeugs wird somit vom Smartphone der Wert 115 an den Raspberry Pi versendet und im struct Objekt „currentStatus“ gespeichert.

Ein weiterer Thread (hier: „speedRegulatedDrivingThread“) liest den zuletzt empfangenen Geschwindigkeitswert aus dem Objekt aus und steuert die Motoren entsprechend an. Dafür wird für jede Geschwindigkeit eine bestimmte Ansteuerungs- (Pin aktiv) und Freilaufzeit (Pin nicht aktiv (GND)) festgelegt und in einer Schleife umgesetzt. (Ab Seite 38 Zeilen: 282-314)

Damit durch die einzelnen Verweilzeiten das gesamte Programm nicht blo-

⁶sys/socket.h: <http://pubs.opengroup.org/onlinepubs/7908799/xns/syssocket.h.html>

⁷POSIX Threads Programming (pthread): <https://computing.llnl.gov/tutorials/pthreads/>

ckiert wird ist es unumgänglich dies in einem zusätzlichen Thread durchzuführen. Hinzuzufügen ist, dass bei der Rückwärtsfahrt und dem Hupbefehl eine Audiodatei über den oben erwähnten Omxplayer abgespielt wird. Dieser Player wird über einen direkten Shell-Befehl ausgeführt und gegebenenfalls beendet.

Im Fall eines Lenkmanövers wird der Thread „blinkingThread“ (Ab Seite 38 Zeilen: 28-55) gestartet, in welchem die Lenkrichtung abgefragt und der entsprechende Blinker in einem bestimmten Zeitintervall angesteuert wird.

3.2.5 Auswertung der Sensordaten und eventuelles Abbremsen des Fahrzeugs

In einem weiteren Thread (hier: „securityCheckThread“) werden in einem in der „constants.h“ festgelegten Intervall (hier 200 ms) sämtliche Sensordaten ausgelesen (Ab Seite 38 Zeilen: 258-517). Der Übersicht halber wird jedem Entfernungssensor ein struct-Objekt vom Typ „ultrasensor“ (Seite 34) zugeordnet. Dieser enthält stets die zuletzt gemessenen Daten. Nach dem Auslesen der Entfernungssensordaten wird anschließend geprüft ob sich ein Hindernis in Fahrtrichtung befindet. Wenn dies der Fall ist, wird beim Unterschreiten eines bestimmten Abstandswertes, welcher von der aktuellen Geschwindigkeit abhängig und ebenfalls in der „constants.h“ festgelegt ist, das Fahrzeug gestoppt. Dabei werden die Motoren für eine bestimmte Zeit in entgegengesetzte Richtung angesteuert um das Fahrzeug aktiv abzubremsen. Solange sich das Fahrzeug in der Gefahrensituation befindet wird im currentStatus-Objekt die Variable „caution“ auf true gehalten und ein Warngeräusch über den Omxplayer abgespielt um den Benutzer darauf hinzuweisen.(Ab Seite 38 Zeilen: 57-109)

Außerdem wird bei eingeschalteter Lichtautomatik (ebenfalls aus dem currentStatus-Objekt abfragbar) überprüft ob der Photowiderstand Licht detektiert und die vorderen LEDs dementsprechend angesteuert.

3.2.6 Übertragung des aktuellen Status an die Android App

Nach der Auswertung der Sensordaten werden in einem weiteren Thread „sendCurrentStatusThread“ die gemessenen Daten und die aktuellen Zustandsdaten (aus dem currentStatus-Objekt) im JSON-Format an die Android App versendet.(Ab Seite 38 Zeilen: 635-665) Dafür wird die cJSON-Library⁸ verwendet. Der zusätzliche Thread ist notwendig, da sonst das Versenden der Daten, etwa bei einer schlechten Verbindung, das Programm blockieren könnte. Im Gegensatz zum Empfang der Befehle von der Android-

⁸cJSON: <https://sourceforge.net/projects/cjson/>

App, wo aus Sicherheitsgründen der Sendevorgang sehr schnell von statten gehen muss, wird bei dem Senden des aktuellen Zustands keine so schnelle Übertragung und Verarbeitung aller Daten erwartet, sodass es keine Probleme bereitet ein etwas größeres Daten-Paket zu verschicken. Hier wird das JSON-Format verwendet, da es zum einen für Menschen sehr leicht zu lesen ist und zum anderen dieses von vielen Programmen eingelesen werden kann, da es sich dabei um einen offenen Standard handelt.

3.3 Kompilieren und Starten des Programms

Das Programm wird mit dem Befehl

```
g++ server.c cJSON/cJSON.c -o Server -lwiringPi -pthread
```

kompiliert. Dabei werden dem g++-Compiler die zu kompilierenden Code-Dateinamen und die benutzten libararies zum linken übergeben.

Das starten des Programms geschieht anschließend über den Befehl:

```
sudo ./home/pi/rc-control/Server
```

Die Super-User Rechte werden bei der Ausführung des Programms benötigt um die GPIO-Ports am Raspberry Pi schalten zu können.

4 Android App Entwicklung

Dieses Kapitel befasst sich mit der im Laufe des Projekts erstellten Android Anwendung, die der Interaktion mit dem Fahrzeug und der Darstellung von Sensordaten dient.

Hauptsächlich sind die Aufgaben der Anwendung neben der Fahrzeugsteuerung vor allem die Darstellung der Webcam-Live Übertragung und die Sprachsteuerung des Fahrzeugs. Der Begriff der Fahrzeugsteuerung beinhaltet hier neben "geradeaus/rückwärts" und "links/rechts" ebenfalls die Steuerung des Lichts, des Sounds und der Gänge.

Die Anwendung erhält das Live-Bild der Webcam und die gemessenen Abstände der Ultraschallsensoren als Dateneingaben um entsprechende Visualisierungen vornehmen zu können.

4.1 Programmierung

Die Programmierung der Android Anwendung erfolgt in JAVA in der Entwicklungsumgebung Android Studio. Die Verbindung zum Raspberry Pi ist über ein Socket⁹ realisiert. Dazu wird die IP-Adresse des Raspberry Pi und ein festgelegter Port benötigt.

Die IP-Adresse wird intern, in Kombination mit dem Port 1337, für die Verbindung zum Programm auf dem Raspberry Pi verwendet. Auf dem Port 8081 findet, mit der selben IP-Adresse, dabei gleichzeitig die Live-Übertragung der Webcam statt. Die beiden Ports sind im Code festgelegt.

Sämtliche zu verschickenden Daten werden über den OutputStream der Socket-Verbindung gesendet. Empfangen werden Daten über den InputStream der Socket-Verbindung. Da empfangene Daten ständig aktualisiert werden müssen, läuft der InputStream in einer Endlosschleife eines separaten Threads, parallel zum Thread der Benutzeroberfläche. Nach dem selben Prinzip läuft das Senden von Befehlen über den OutputStream in einem separaten Thread. In diesem wird je nach Szenario ein anderer Zahlenwert über die Socket-Verbindung gesendet (Eine vollständige Auflistung der Zahlenwerte (IDs) und ihrer Bedeutungen findet sich in Tabelle 2. Der Thread terminiert, wenn das Senden eines Befehls abgeschlossen ist. Zum Senden wird somit immer, wenn auch nur kurzzeitig, ein neuer Thread gestartet.

Wird die Socket-Verbindung z.B. vorhersehbar vom Klienten geschlossen, so wird zuvor die ID für "CLOSESOCKET" gesendet. Darauf kann das Programm auf dem Raspberry Pi dann reagieren. Bei dieser ID werden z.B. die Motoren sinnvollerweise ausgeschaltet.

⁹Socket: <https://docs.oracle.com/javase/tutorial/networking/sockets/>

Name	ID	Bedeutung
CLOSESOCKET	77	Schließe die Socket-Verbindung
GO_1	116	Fahre im ersten Gang vorwärts
GO_2	117	Fahre im zweiten Gang vorwärts
GO_3	118	Fahre im dritten Gang vorwärts
GO_4	119	Fahre im vierten Gang vorwärts
GO_5	120	Fahre im fünften Gang vorwärts
BACK_1	114	Fahre im ersten Gang rückwärts
BACK_2	113	Fahre im zweiten Gang rückwärts
BACK_3	112	Fahre im dritten Gang rückwärts
BACK_4	111	Fahre im vierten Gang rückwärts
BACK_5	110	Fahre im fünften Gang rückwärts
STOP	115	Hält Fahrzeug an
LEFT	11	Schlage Räder nach links ein
RIGHT	12	Schlage Räder nach rechts ein
STRAIGHT	10	Stelle Räder gerade
LIGHTS_OFF	18	Schalte Licht aus
LIGHTS_ON	19	Schalte Licht an
LIGHTS_AUTO_OFF	20	Schalte Automatikmodus an
LIGHTS_AUTO_ON	21	Schalte Automatikmodus aus
HORN	25	Gib Sound der Hupe wieder
MUSIC_ON	30	Gib Musiktitel wieder
SOUND_OFF	31	Beende Soundwiedergabe

Tabelle 2: Auflistung der Zahlenwerte die über die Socket-Verbindung gesendet werden

Weiterer Code der Anwendung dient der Fehlerbehandlung, der internen Persistenz von Daten, der Initialisierung von Objekten...etc. Auf eine nähere Beschreibung dieser Komponenten wird an dieser Stelle verzichtet.

4.2 Webcam Live-Übertragung

Die Bildübertragung der Webcam zählt aus Sicht der Android Anwendung zur Dateneingabe. Die Funktionsweise der auf dem Raspberry Pi installierten motion Software wurde in Kapitel 3.1.1 behandelt. Da der JPEG-Stream im lokalen Netzwerk über die IP-Adresse des Raspberry Pi und den Port 8081 sichtbar ist, kann er im Code per HTTP-Anfrage abgegriffen werden. Zur Verarbeitung und Visualisierung des Streams werden Teile von externem

JAVA Code verwendet¹⁰. Die übernommene Klasse "MjpegInputStream" kümmert sich um die Verarbeitung des Streams. Die Klasse "MjpegView" stellt ein benutzerdefiniertes View-Objekt bereit, das problemlos in die eigene Anwendungsstruktur eingebaut werden kann. Nachdem im eigenen Code nun die HTTP-Anfrage erfolgreich ist, kann der JPEG-Stream der motion Software der MjpegInputStream-Klasse als Inputstream übergeben werden. Letztlich muss im eigenen Code einem MjpegView-Objekt eine Referenz der MjpegInputStream-Klasse als Quelle zugewiesen werden. Nun kann das MjpegView-Objekt zur Darstellung des Live-Streams verwendet werden.
(Ist die Webcam korrekt am Raspberry Pi angeschlossen, wird in der Anwendung das Bild angezeigt. Läuft zwar der Server, aber die Webcam ist nicht mit dem Raspberry Pi verbunden, wird ein graues Bild angezeigt.)
Da der Live-Stream zur Laufzeit durchgängig aktiv sein soll, erfolgt die Verarbeitung in einem separaten Thread, parallel zum Thread der Benutzeroberfläche.

4.3 Spracherkennung

Die Spracherkennung erfolgt vollständig offline über eine externe library¹¹ die auf C-Code basiert. Auf der Grundlage dieser library wird eine SpeechRecognition-Klasse erstellt, die sich um die Verarbeitung von Sprachsignalen kümmert. Die Funktionsweise der library sieht vor, mögliche Begriffe vorher in einer Datei festzulegen. Wird ein Sprachsignal aufgenommen und soll verarbeitet werden, so wird aus diesen Begriffen derjenige ausgewählt, der der Aufnahme am Ähnlichsten ist. Im eigenen Code muss ein Objekt der SpeechRecognition-Klasse erstellt werden. Dieses nimmt über das Handymikrofon, solange der Mikrofonknopf gedrückt ist, ein Sprachsignal auf. Die verwendete library hatte im Test immer wieder Probleme mit gleichklingenden Wörtern und undeutlicher Aussprache. Um dieses Problem einzuschränken, werden zwei Wortgruppen erstellt. Die Wörter in der einen Datei sind Motorbefehle, während die in der Anderen für Licht und Sound zuständig sind. Der Benutzer kann über das Handymikrofon entweder die Motoren oder Licht und Sound steuern. Die Motoren lassen sich über die folgenden Wörter steuern: "left, right, stand, go, back, line up". Das Licht und der Sound lassen sich dagegen über: "beep, on, off, auto" steuern. Die Sprachbefehle und ihre Aktionen sind in Tabelle 3 aufgeführt.

¹⁰neuralassembly: <https://bitbucket.org/neuralassembly/simplempegview/src/6a5cf5bd8f64?at=master>

¹¹pocketsphinx: <https://github.com/cmusphinx/pocketsphinx-android-demo>

Befehl	Aktion
left	Stelle die Räder nach links
right	Stelle die Räder nach rechts
stand	Halte an
go	Fahre vorwärts
back	Fahre rückwärts
line up	Stelle die Räder gerade
beep	Drücke die Hupe
on	Schalte das Licht an
off	Schalte das Licht aus
auto	Stelle auf Lichtautomatik

Tabelle 3: Sprachbefehle

4.4 Benutzeroberfläche

Die Benutzeroberfläche der Anwendung kann in Hoch- und Querformat des Smartphones unterteilt werden. Der Hochformat dient u.a. der Einstellung der IP-Adresse, wogegen im Querformat die eigentliche Interaktion mit dem Fahrzeug stattfindet.

4.4.1 Hochformat

Beim Start der Anwendung gelangt der Benutzer immer in den Hochformat. Da sich die IP-Adresse des Raspberry Pi je nach Netzwerk ändern kann, ist es erforderlich, diese über die Benutzeroberfläche einzugeben (siehe Abbildung 23). Wurde eine ungültige IP-Adresse eingegeben, oder ist der Raspberry Pi offline, so wird dem Benutzer ein Network Error Dialog angezeigt (siehe Abbildung 24).

Ist der Verbindungsauflauf zum Raspberry Pi erfolgreich, so besteht die Möglichkeit, die Verbindung neu aufzubauen. Dem Benutzer ist dafür das Symbol mit den zwei Pfeilen am oberen Bildschirmrand gegeben. Diese Funktion kann z.B. genutzt werden, wenn die Latenz zu groß ist, oder das Smartphone im Debug-Modus an einem Computer angeschlossen ist und dadurch langsam reagiert.

Die drei Punkte am oberen Bildschirmrand beinhalten zwei Optionen: "Change URL" (Ruft den Dialog aus Abbildung 23 auf) und "Debug" (Debug-Modus. Näheres hierzu im Kapitel 4.4.2 Querformat).

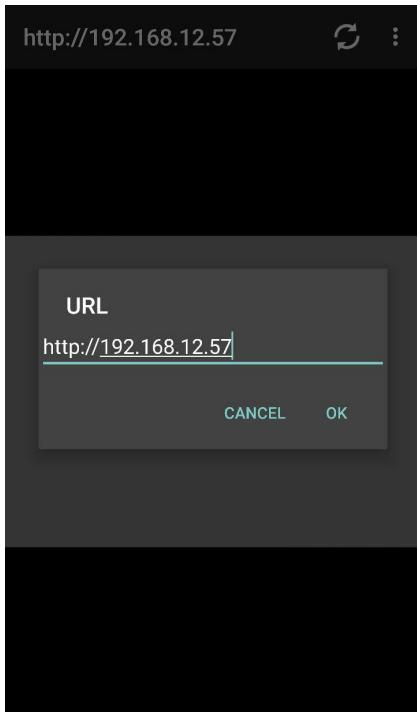


Abbildung 23: Dialog zur Eingabe der IP-Adresse

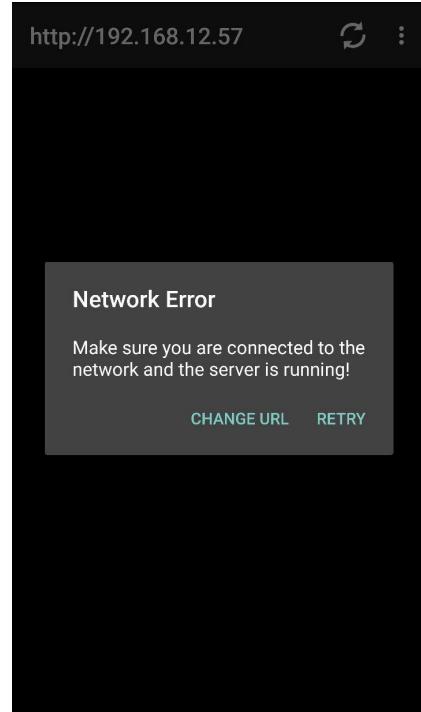


Abbildung 24: Network Error Dialog

4.4.2 Querformat

Eine Ansicht der Anwendung im Querformat ist in Abbildung 25 zu sehen. Die Motoren des Fahrzeugs werden über die Steuerpfeile in den unteren beiden Ecken gesteuert. Unten links für die Fahrt "geradeaus/rückwärts" und unten rechts für die Lenkung "link/rechts". Das Fahrzeug fährt geradeaus oder rückwärts, bzw. lässt die Räder in die gewählte Richtung links oder rechts eingeschlagen, solange der entsprechende Pfeil gedrückt ist.

Am unteren Bildschirmrand befindet sich die Ganganzeige. Die Anzeige besteht aus fünf Knöpfen, wobei die Nummer des aktuell eingelegten Gangs hervorgehoben ist. Je höher der Gang, desto schneller fährt das Fahrzeug. Zu Beginn ist immer der erste Gang eingelegt. Der Benutzer kann jederzeit in einen beliebigen Gang wechseln, indem er auf einen der vier anderen Gänge tippt. Die fünf Gänge gelten jeweils für die Vorwärts- und Rückwärtssfahrt. Oben rechts in der Ansicht befindet sich das Mikrofonsymbol für die Spracherkennung. Hält es der Benutzer gedrückt, nimmt das Mikrofon des Smartphones ein akustisches Signal auf, bis das Symbol losgelassen wird. Mit einem Schaltknopf unter dem Mikrofonsymbol kann zwischen Sprachbefehlen für die

Fahrt oder Sprachbefehlen für Licht und Hupe umgeschaltet werden. Von der Anwendung können somit immer nur die Befehle der eingestellten Kategorie erkannt werden. (Eine Erläuterung hierzu findet sich in Kapitel 4.3 Spracherkennung)

Oben links in der Ansicht sind drei weitere Symbole abgebildet. Jedes der drei Symbole ist ebenfalls ein Knopf und kann gedrückt werden. Das linke Symbol stellt die aktuelle Lichteinstellung dar. Es gibt drei Einstellungen: "off, on, auto". Bei "off" ist das Licht am Auto ausgeschaltet, bei "on" ist es eingeschaltet und bei "auto" ist es im Automatikmodus (siehe Kapitel 2.4 Installation der Scheinwerfer und Blinker). Das Symbol ändert dem Zustand entsprechend seine Darstellung. In der Mitte befindet sich die Hupe und rechts daneben ein Musiksymbol. Wird einer dieser Knöpfe gedrückt, wird ein akustisches Signal über den Lautsprecher(siehe Kapitel 2.2 Neue Komponenten). Die Hupe gibt einen kurzen Ton aus, der Musikknopf startet die Wiedergabe eines Musiktitels.

Das rote Warndreieck in der Mitte der Ansicht wird angezeigt, sobald das Programm auf dem Raspberry Pi meldet, dass das Fahrzeug zu nah an einem Hindernis steht oder sich auf dieses zubewegt. Das Warndreieck ist im Gegensatz zu allen anderen Symbolen in der Ansicht kein Knopf mit dem der Benutzer interagieren kann. Es verschwindet nur wieder, wenn sich das Fahrzeug nicht mehr in unmittelbarer Nähe eines Hindernisses befindet.

Der Debug-Modus der Anwendung ist in Abbildung 26 abgebildet. Der einzige Unterschied zur normalen Ansicht ist der weiße Text in der unteren Hälfte des Bildschirms. Angezeigt werden u.a. die Abstände, die die Ultraschallsensoren melden und die aktuelle Lichteinstellung. Die Informationen stammen vom Inputstream der Socket-Verbindung, also direkt vom laufenden Programm des Raspberry Pi. Nützlich sind diese Informationen, falls ein Fehler oder ein unerwartetes Verhalten auftritt. Zwischen einem Fehler im Code des auf dem Raspberry Pi laufenden Programms und einem Fehler im Code der Android Anwendung kann so leichter differenziert werden. Bei korrekter Integration der Symbole der Benutzeroberfläche stimmen die angezeigten Symbole stets mit den Informationen aus dem Debug-Modus überein.

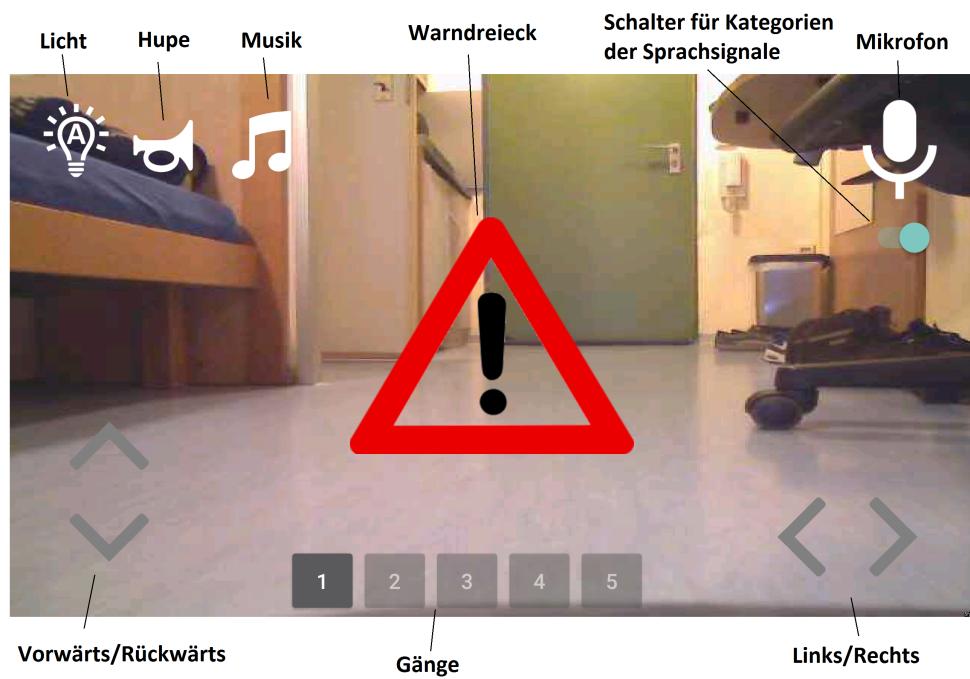


Abbildung 25: Screenshot der Anwendung im Querformat mit Beschreibung der Elemente

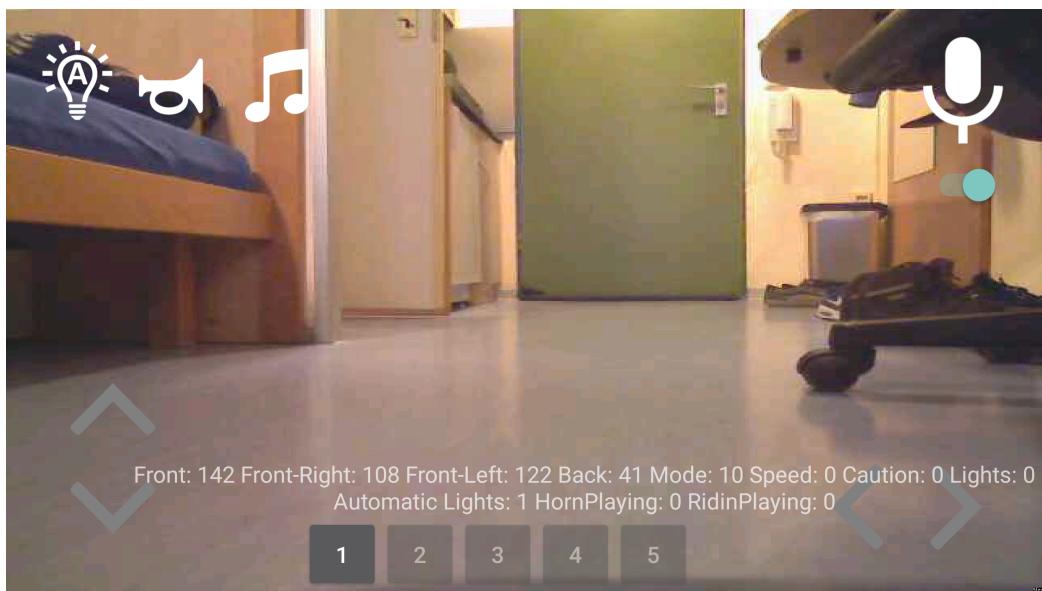


Abbildung 26: Screenshot der Anwendung im Querformat im Debug-Modus

5 Anhang

5.1 Quellcode des Raspberry Pi Programms

C-Code/constants.h

```
1 #define DEBUG true
2
3 #define NUMBEROFULTRASENSORS 4
4 #define CLOSESOCKET 77
5
6 #define FRONTRIGHTSENSOR 0
7 #define PIN_ECHO_1 29
8 #define PIN_TRIGGER_1 28
9
10 #define FRONTSENSOR 1
11 #define PIN_ECHO_2 21
12 #define PIN_TRIGGER_2 22
13
14 #define FRONTLEFTSENSOR 2
15 #define PIN_ECHO_3 25
16 #define PIN_TRIGGER_3 24
17
18 #define BACKSENSOR 3
19 #define PIN_ECHO_4 27
20 #define PIN_TRIGGER_4 26
21
22 #define PIN_LIGHTSENSOR 3
23 #define PIN_FRONTLIGHT 6
24
25 #define PIN_LEFT 5
26 #define PIN_RIGHT 2
27 #define PIN_BACKWARD 0
28 #define PIN_FORWARD 1
29
30 #define PIN_BLINKER_LEFT 23
31 #define PIN_BLINKER_RIGHT 7
32
33 #define BLINKERMS 700
34
35 #define STRAIGHT 10
36 #define LEFT 11
37 #define RIGHT 12
38
39 #define LIGHTS_OFF 18
40 #define LIGHTS_ON 19
41 #define LIGHTS_AUTO_OFF 20
42 #define LIGHTS_AUTO_ON 21
43
44 #define HORN 25
45
46 #define RIDIN_DIRTY 30
47 #define MUSIC_OFF 31
48
49
50 #define MINSPEEDBACK 110
51 #define MAXSPEEDFORWARD 120
52
53 #define CAUTIONFRONT_CM 10
54 #define CAUTIONFRONTLEFT_CM 5
55 #define CAUTIONFRONTRIGHT_CM 5
56 #define CAUTIONBACK_CM 10
57
58 #define CAUTIONFACTOR_1 2
59 #define CAUTIONFACTOR_2 3
60 #define CAUTIONFACTOR_3 4
61 #define CAUTIONFACTOR_4 5
62 #define CAUTIONFACTOR_5 5
63
64
65 #define POSITIVERESPONSE 1
66 #define NEGATIVERESPONSE 0
67
68 #define REFRESHMS 200
```

C-Code/status.h

```
1 struct status {
2     struct ultrasensor *ultrasensors [NUMBEROFULTRASENSORS];
3     int mode;
4     int speed;
5     int newsockFD;
6     bool connected;
7     bool caution;
8     bool lightsOn;
9     bool lightsAutoOn;
10    bool horn;
11    bool hornPlaying;
12    bool soundPlaying;
13    bool ridin;
14    bool ridinPlaying;
15 };
```

C-Code/ultrasensor.h

```
1 #include <wiringPi.h>
2
3 struct ultrasensor {
4     uint8_t pin_ECHO;
5     uint8_t pin_TRIGGER;
6     volatile int lastCM;
7 };
8
9 int getCM(ultrasensor * usens) {
10
11     //Send trig pulse
12     digitalWrite(usens->pin_TRIGGER, HIGH);
13     delayMicroseconds(20);
14     digitalWrite(usens->pin_TRIGGER, LOW);
15
16     //Wait for echo start
17     long startTime1 = micros();
18     while(digitalRead(usens->pin_ECHO) == LOW) {
19         if((micros() - startTime1) > 20000)
20         {
21             printf("Ultrasensor Timeout\n");
22             usens->lastCM = -1;
23             return -1;
24         }
25     }
26
27     //Wait for echo end
28     long startTime = micros();
29     while(digitalRead(usens->pin_ECHO) == HIGH) {
30         if((micros() - startTime) > 20000)
31         {
32             // printf("Ultrasensor Timeout\n");
33             usens->lastCM = -1;
34             return -1;
35         }
36     }
37     long travelTime = micros() - startTime;
38
39     //Get distance in cm
40     int distance = travelTime / 58;
41     usens->lastCM = distance;
42     return distance;
43 }
```

C-Code/server.c

```
1 #include <stdio.h>
2 #include <sys/types.h>
3 #include <sys/socket.h>
4 #include <netinet/in.h>
5 #include <netdb.h>
6 #include <string.h>
7 #include <stdlib.h>
8 #include <unistd.h>
9 #include <errno.h>
10 #include <wiringPi.h>
11 #include <pthread.h>
12 #include <ao/ao.h>
13 #include <mpg123.h>
14 #include "constants.h"
15 #include "status.h"
16 #include "ultrasensor.h"
17 #include "cJSON/cJSON.h"
18
19 #define BITS 8
20
21 pthread_mutex_t mutex;
22 pthread_mutex_t modeMutex;
23 pthread_mutex_t speedMutex;
24
25 struct status *currentStatus;
26
27 #include "functions.c"
28
29
30 void setup()
31 {
32     currentStatus = (struct status*) malloc(sizeof(struct status));
33     currentStatus->mode = STRAIGHT;
34     currentStatus->speed = 0;
35     currentStatus->connected = false;
36     currentStatus->caution = false;
37     currentStatus->ridin = false;
38     currentStatus->horn = false;
39     currentStatus->lightsOn = false;
40     currentStatus->lightsAutoOn = true;
41
42     struct ultrasensor **ultrasensors = currentStatus->ultrasensors;
43
44     wiringPiSetup();
45
46     for(int i = 0; i < NUMBEROFULTRASENSORS; i++)
47     {
48         ultrasensors[i] = (struct ultrasensor*) malloc(sizeof(struct ultrasensor));
49     }
50
51
52     ultrasensors[FRONTRIGHTSENSOR]->pin_ECHO = PIN_ECHO_1;
53     ultrasensors[FRONTRIGHTSENSOR]->pin_TRIGGER = PIN_TRIGGER_1;
54
55     ultrasensors[FRONTSensor]->pin_ECHO = PIN_ECHO_2;
56     ultrasensors[FRONTSensor]->pin_TRIGGER = PIN_TRIGGER_2;
57
58     ultrasensors[FRONTLEFTSENSOR]->pin_ECHO = PIN_ECHO_3;
59     ultrasensors[FRONTLEFTSENSOR]->pin_TRIGGER = PIN_TRIGGER_3;
60
61     ultrasensors[BACKSENSOR]->pin_ECHO = PIN_ECHO_4;
62     ultrasensors[BACKSENSOR]->pin_TRIGGER = PIN_TRIGGER_4;
63
64     for(int i = 0; i < NUMBEROFULTRASENSORS; i++)
65     {
66         pinMode(ultrasensors[i]->pin_TRIGGER, OUTPUT);
67         pinMode(ultrasensors[i]->pin_ECHO, INPUT);
68         digitalWrite(ultrasensors[i]->pin_TRIGGER, LOW);
69     }
70
71     pinMode(PIN_LEFT, OUTPUT);
72     pinMode(PIN_RIGHT, OUTPUT);
73     pinMode(PIN_BACKWARD, OUTPUT);
74     pinMode(PIN_FORWARD, OUTPUT);
75     pinMode(PIN_LIGHTSENSOR, INPUT);
76     pinMode(PIN_FRONTLIGHT, OUTPUT);
77
78     pinMode(PIN_BLINKER_LEFT, OUTPUT);
79     pinMode(PIN_BLINKER_RIGHT, OUTPUT);
80
81     digitalWrite(PIN_LEFT, LOW);
82     digitalWrite(PIN_RIGHT, LOW);
```

```

83     digitalWrite(PIN_FORWARD, LOW);
84     digitalWrite(PIN_BACKWARD, LOW);
85     digitalWrite(PIN_FRONLIGHT, LOW);
86     digitalWrite(PIN_BLINKER_LEFT, LOW);
87     digitalWrite(PIN_BLINKER_RIGHT, LOW);
88
89     delay(30);
90 }
91
92 void error( char *msg ) {
93     perror( msg );
94     exit(1);
95 }
96
97 int main(int argc, char *argv[]) {
98     setup();
99
100    pthread_t receiveCommandThread;
101    pthread_t sendCurrentStatsThread;
102    pthread_t speedRegulatedDrivingThread;
103
104
105    pthread_mutex_init (&mutex, NULL);
106    pthread_mutex_init (&modeMutex, NULL);
107    pthread_mutex_init (&speedMutex, NULL);
108
109
110    int sockfd, newsockfd, portno = 1337, clilen;
111    char buffer[256];
112    struct sockaddr_in serv_addr, cli_addr;
113
114
115    printf( "using port %d\n", portno );
116
117    sockfd = socket(AF_INET, SOCK_STREAM, 0);
118    if (sockfd < 0)
119        error( const_cast<char *>("ERROR opening socket") );
120    bzero((char *)&serv_addr, sizeof(serv_addr));
121
122    serv_addr.sin_family = AF_INET;
123    serv_addr.sin_addr.s_addr = INADDR_ANY;
124    serv_addr.sin_port = htons( portno );
125    if (bind(sockfd, (struct sockaddr *)&serv_addr,
126             sizeof(serv_addr)) < 0)
127        error( const_cast<char *>("ERROR on binding") );
128    listen(sockfd, 5);
129    clilen = sizeof(cli_addr);
130
131    //--- infinite wait on a connection ---
132    while ( 1 ) {
133        printf( "waiting for new client...\n" );
134        if ( ( newsockfd = accept4( sockfd, (struct sockaddr *) &cli_addr, (socklen_t *)
135            &clilen, SOCK_NONBLOCK) ) < 0 )
136            error( const_cast<char *>("ERROR on accept") );
137        printf( "opened new communication with client\n" );
138
139        currentStatus->newsockFD = newsockfd;
140        currentStatus->connected = true;
141        ultrasensor **ultrasensors = currentStatus->ultrasensors;
142
143        pthread_create(&receiveCommandThread, NULL, receiveCommand,(void *) currentStatus);
144        pthread_create(&speedRegulatedDrivingThread, NULL, speedRegulatedDriving,(void *)
145                      currentStatus);
146
147        digitalWrite(PIN_BLINKER_RIGHT, HIGH);
148        digitalWrite(PIN_BLINKER_LEFT, HIGH);
149        usleep(1000 * 1000);
150        digitalWrite(PIN_BLINKER_RIGHT, LOW);
151        digitalWrite(PIN_BLINKER_LEFT, LOW);
152
153        while( currentStatus->connected )
154        {
155
156            for(int i = 0; i < NUMBEROFULTRASENSORS; i++)
157            {
158                getCM(ultrasensors[i]);
159            }
160            pthread_create(&sendCurrentStatsThread, NULL, sendCurrentStats, (void *)
161                           currentStatus);
162            securityCheck((void *) currentStatus);
163            pthread_join(sendCurrentStatsThread, NULL);

```

```
163     usleep (REFRESHMS*1000) ;
164
165 }
166 close( newsockfd ) ;
167 currentStatus->newsockFD = 0;
168 }
169 printf( "END" );
170 return 0;
171 }
172 }
```

C-Code/functions.c

```

1 pthread_t makeSoundThread;
2
3 void sendData(int sockfd, char *json)
4 {
5     int n;
6     if (write(sockfd, json, strlen(json)) < 0 && errno == EAGAIN)
7     {
8         if (DEBUG)
9         {
10             printf("Resource unavailable\n");
11         }
12     }
13 }
14
15 int getData(int sockfd)
16 {
17     uint8_t data;
18
19     while (read(sockfd, &data, sizeof(uint8_t)) < 0 && errno == EAGAIN)
20     ;
21     if (DEBUG)
22     {
23         printf("Got: %d\n", data);
24     }
25     return data;
26 }
27
28 void *blinking(void *vargp)
29 {
30
31     struct status *currentStatus = (status*) vargp;
32     if (DEBUG)
33     {
34         printf("Start Blinking in Mode: %d\n", currentStatus->mode);
35     }
36     while (currentStatus->mode == 11)
37     {
38         digitalWrite(PIN_BLINKER_LEFT, HIGH);
39         usleep(BLINKERMS * 1000);
40         digitalWrite(PIN_BLINKER_LEFT, LOW);
41         usleep(BLINKERMS * 1000);
42     }
43
44     while (currentStatus->mode == 12)
45     {
46         digitalWrite(PIN_BLINKER_RIGHT, HIGH);
47         usleep(BLINKERMS * 1000);
48         digitalWrite(PIN_BLINKER_RIGHT, LOW);
49         usleep(BLINKERMS * 1000);
50     }
51     if (DEBUG)
52     {
53         printf("Stop Blinking in Mode: %d\n", currentStatus->mode);
54     }
55 }
56
57 void *makeSound(void *vargp)
58 {
59
60     struct status *currentStatus = (status*) vargp;
61     char *name[] =
62     { "back.mp3", "horn.mp3", "caution.mp3", "ridin.mp3" };
63     int i = -1;
64
65     if (currentStatus->speed < 0)
66     {
67         i = 0;
68     }
69     if (currentStatus->caution)
70     {
71         i = 2;
72     }
73     if (currentStatus->horn)
74     {
75         i = 1;
76         currentStatus->horn = false;
77         currentStatus->hornPlaying = true;
78     }
79     if (currentStatus->ridin)
80     {
81         i = 3;
82         currentStatus->ridin = false;

```

```

83     currentStatus->ridinPlaying = true;
84 }
85 if (i != -1)
86 {
87     currentStatus->soundPlaying = true;
88     if (DEBUG)
89     {
90         printf("Start playing of %s\n", name[i]);
91     }
92     int pid;
93     pid=fork();
94     if (pid==0)
95     {
96         execvp("/usr/bin/omxplayer", " ", name[i], NULL);
97         currentStatus->ridinPlaying = false;
98         currentStatus->hornPlaying = false;
99         currentStatus->soundPlaying = false;
100        _exit(0);
101    }
102 else
103 {
104     wait();
105 }
106 }
107 }
108 }
109 }
110 void playSound()
111 {
112     currentStatus->ridinPlaying = false;
113     currentStatus->hornPlaying = false;
114     currentStatus->soundPlaying = false;
115
116     system("killall omxplayer.bin");
117     pthread_join(makeSoundThread, NULL);
118     pthread_create(&makeSoundThread, NULL, makeSound, (void *) currentStatus);
119 }
120
121
122 int action(int a)
123 {
124     int respond;
125     if (DEBUG)
126     {
127         printf("Action function received: %d\n", a);
128     }
129
130     pthread_t blinkingThread;
131     if (a < 70)
132     {
133         switch (a)
134         {
135             case STRAIGHT:
136                 digitalWrite(PIN_LEFT, LOW);
137                 digitalWrite(PIN_RIGHT, LOW);
138                 pthread_cancel(blinkingThread);
139                 digitalWrite(PIN_BLINKER_LEFT, LOW);
140                 digitalWrite(PIN_BLINKER_RIGHT, LOW);
141                 respond = POSITIVERESPONSE;
142             break;
143             case LEFT:
144                 digitalWrite(PIN_RIGHT, LOW);
145                 digitalWrite(PIN_LEFT, HIGH);
146                 respond = POSITIVERESPONSE;
147                 pthread_create(&blinkingThread, NULL, blinking,
148                               (void *) currentStatus);
149             break;
150             case RIGHT:
151                 digitalWrite(PIN_LEFT, LOW);
152                 digitalWrite(PIN_RIGHT, HIGH);
153                 respond = POSITIVERESPONSE;
154                 pthread_create(&blinkingThread, NULL, blinking,
155                               (void *) currentStatus);
156             break;
157             case LIGHTS_ON:
158                 digitalWrite(PIN_FRONTLIGHT, HIGH);
159                 currentStatus->lightsOn = true;
160             break;
161             case LIGHTS_OFF:
162                 digitalWrite(PIN_FRONTLIGHT, LOW);
163                 currentStatus->lightsOn = false;
164             break;
165         }
166     }
167 }

```

```

166     case HORN:
167         currentStatus->horn = true;
168         playSound();
169         break;
170     case RIDIN_DIRTY:
171         currentStatus->ridin = true;
172         playSound();
173         break;
174     case LIGHTS_AUTO_ON:
175         currentStatus->lightsAutoOn = true;
176         break;
177     case LIGHTS_AUTO_OFF:
178         currentStatus->lightsAutoOn = false;
179         break;
180     case MUSIC_OFF:
181         currentStatus->ridinPlaying = false;
182         currentStatus->hornPlaying = false;
183         currentStatus->soundPlaying = false;
184         system("killall omxplayer.bin");
185         break;
186     default:
187         respond = NEGATIVERESPONSE;
188         if(DEBUG)
189         {
190             printf("Doesn't find action for: %d\n", a);
191         }
192     }
193 }
194
195 if (respond == POSITIVERESPONSE)
196 {
197
198     pthread_mutex_lock(&modeMutex);
199     currentStatus->mode = a;
200     pthread_mutex_unlock(&modeMutex);
201 }
202 else if (a >= MINSPEEDBACK && a <= MAXSPEEDFORWARD)
203 {
204     pthread_mutex_lock(&speedMutex);
205     currentStatus->speed = a - ((MINSPEEDBACK + MAXSPEEDFORWARD) >> 1);
206     pthread_mutex_unlock(&speedMutex);
207     respond == POSITIVERESPONSE;
208     if(DEBUG)
209     {
210         printf("Action: New Speed: %d\n", currentStatus->speed);
211     }
212
213     if (currentStatus->speed < 0)
214     {
215         playSound();
216     }
217     else
218     {
219         currentStatus->soundPlaying = false;
220         system("killall omxplayer.bin");
221     }
222 }
223 else if (a == CLOSESOCKET)
224 {
225     if(DEBUG)
226     {
227         printf("Action: Closing Socket\n");
228     }
229     currentStatus->mode = STRAIGHT;
230     currentStatus->lightsOn = false;
231     digitalWrite(PIN_BACKWARD, LOW);
232     digitalWrite(PIN_FORWARD, LOW);
233
234     pthread_cancel(makeSoundThread);
235     digitalWrite(PIN_LEFT, LOW);
236     digitalWrite(PIN_RIGHT, LOW);
237     pthread_cancel(blinkingThread);
238     digitalWrite(PIN_BLINKER_LEFT, LOW);
239     digitalWrite(PIN_BLINKER_RIGHT, LOW);
240     digitalWrite(PIN_FRONTLIGHT, LOW);
241     currentStatus->lightsAutoOn = false;
242     currentStatus->speed = 0;
243     currentStatus->connected = false;
244 }
245 else
246 {
247     respond == NEGATIVERESPONSE;
248 }

```

```

249     if (DEBUG)
250     {
251         printf("Action done\n");
252     }
253
254     return respond;
255 }
256
257
258 void *securityCheck(void *vargp)
259 {
260     struct status *currentStatus = (status*) vargp;
261     ultrasensor **ultrasensors = currentStatus->ultrasensors;
262
263     int frontCM = ultrasensors[FRONTSENSOR]->lastCM;
264     int frontLeftCM = ultrasensors[FRONTLEFTSENSOR]->lastCM;
265     int frontRightCM = ultrasensors[FRONTRIGHTSENSOR]->lastCM;
266     int backCM = ultrasensors[BACKSENSOR]->lastCM;
267
268     double cautionFactors[] =
269     { CAUTIONFACTOR_1, CAUTIONFACTOR_2, CAUTIONFACTOR_3, CAUTIONFACTOR_4,
270       CAUTIONFACTOR_5 };
271
272     int frontCautionCM = CAUTIONFRONT_CM
273     * cautionFactors[abs(currentStatus->speed) - 1];
274     int frontLeftCautionCM = CAUTIONFRONTEFT_CM
275     * cautionFactors[abs(currentStatus->speed) - 1];
276     int frontRightCautionCM = CAUTIONFRONTRIGHT_CM
277     * cautionFactors[abs(currentStatus->speed) - 1];
278     int backCautionCM = CAUTIONBACK_CM
279     * cautionFactors[abs(currentStatus->speed) - 1];
280
281     if (currentStatus->speed != 0 && frontCM != -1 && frontLeftCM != -1 && frontRightCM
282         != -1
283         && backCM != -1)
284     {
285         currentStatus->caution = false;
286
287         switch (currentStatus->speed)
288         {
289             case -5:
290                 if (backCM < backCautionCM)
291                 {
292                     if (DEBUG)
293                     {
294                         printf("Detect obstacle: Back\n");
295                     }
296                     pthread_mutex_lock(&speedMutex);
297                     currentStatus->speed = 5;
298                     currentStatus->caution = true;
299                     playSound();
300                     usleep(350 * 1000);
301                     currentStatus->speed = 0;
302                     pthread_mutex_unlock(&speedMutex);
303                     if (DEBUG)
304                     {
305                         printf("Emergency Stop succeed\n");
306                     }
307                 }
308                 break;
309             case -4:
310                 if (backCM < backCautionCM)
311                 {
312                     if (DEBUG)
313                     {
314                         printf("Detect obstacle: Back\n");
315                     }
316                     pthread_mutex_lock(&speedMutex);
317                     currentStatus->speed = 5;
318                     currentStatus->caution = true;
319                     playSound();
320                     usleep(290 * 1000);
321                     currentStatus->speed = 0;
322                     pthread_mutex_unlock(&speedMutex);
323                     if (DEBUG)
324                     {
325                         printf("Emergency Stop succeed\n");
326                     }
327                 }
328                 break;
329             case -3:
330                 if (backCM < backCautionCM)

```

```

331     {
332         if (DEBUG)
333         {
334             printf("Detect obstacle: Back\n");
335         }
336         pthread_mutex_lock(&speedMutex);
337         currentStatus->speed = 5;
338         currentStatus->caution = true;
339         playSound();
340         usleep(200 * 1000);
341         currentStatus->speed = 0;
342         pthread_mutex_unlock(&speedMutex);
343         if (DEBUG)
344         {
345             printf("Emergency Stop succeed\n");
346         }
347     }
348     break;
349 case -2:
350     if (backCM < backCautionCM)
351     {
352         if (DEBUG)
353         {
354             printf("Detect obstacle: Back\n");
355         }
356         pthread_mutex_lock(&speedMutex);
357         currentStatus->speed = 5;
358         currentStatus->caution = true;
359         playSound();
360         usleep(150 * 1000);
361         currentStatus->speed = 0;
362         pthread_mutex_unlock(&speedMutex);
363         if (DEBUG)
364         {
365             printf("Emergency Stop succeed\n");
366         }
367     }
368     break;
369 case -1:
370     if (backCM < backCautionCM)
371     {
372         if (DEBUG)
373         {
374             printf("Detect obstacle: Back\n");
375         }
376         pthread_mutex_lock(&speedMutex);
377         currentStatus->speed = 5;
378         currentStatus->caution = true;
379         playSound();
380         usleep(100 * 1000);
381         currentStatus->speed = 0;
382         pthread_mutex_unlock(&speedMutex);
383         if (DEBUG)
384         {
385             printf("Emergency Stop succeed\n");
386         }
387     }
388     break;
389 case 1:
390     if (frontCM < frontCautionCM || frontLeftCM < frontLeftCautionCM
391         || frontRightCM < frontRightCautionCM)
392     {
393         if (DEBUG)
394         {
395             printf("Detect obstacle: Front\n");
396         }
397         pthread_mutex_lock(&speedMutex);
398         currentStatus->speed = -5;
399         currentStatus->caution = true;
400         playSound();
401         usleep(100 * 1000);
402         currentStatus->speed = 0;
403         pthread_mutex_unlock(&speedMutex);
404         if (DEBUG)
405         {
406             printf("Emergency Stop succeed\n");
407         }
408     }
409     break;
410 }
411
412 }
413

```

```

414     case 2:
415         if (frontCM < frontCautionCM || frontLeftCM < frontLeftCautionCM
416             || frontRightCM < frontRightCautionCM)
417         {
418             if (DEBUG)
419             {
420                 printf("Detect obstacle: Front\n");
421             }
422             pthread_mutex_lock(&speedMutex);
423             currentStatus->speed = -5;
424             currentStatus->caution = true;
425             playSound();
426             usleep(150 * 1000);
427             currentStatus->speed = 0;
428             pthread_mutex_unlock(&speedMutex);
429             if (DEBUG)
430             {
431                 printf("Emergency Stop succeed\n");
432             }
433         }
434         break;
435     case 3:
436         if (frontCM < frontCautionCM || frontLeftCM < frontLeftCautionCM
437             || frontRightCM < frontRightCautionCM)
438         {
439             if (DEBUG)
440             {
441                 printf("Detect obstacle: Front\n");
442             }
443             pthread_mutex_lock(&speedMutex);
444             currentStatus->speed = -5;
445             currentStatus->caution = true;
446             playSound();
447             usleep(200 * 1000);
448             currentStatus->speed = 0;
449             pthread_mutex_unlock(&speedMutex);
450             if (DEBUG)
451             {
452                 printf("Emergency Stop succeed\n");
453             }
454         }
455         break;
456     case 4:
457         if (frontCM < frontCautionCM || frontLeftCM < frontLeftCautionCM
458             || frontRightCM < frontRightCautionCM)
459         {
460             if (DEBUG)
461             {
462                 printf("Detect obstacle: Front\n");
463             }
464             pthread_mutex_lock(&speedMutex);
465             currentStatus->speed = -5;
466             currentStatus->caution = true;
467             playSound();
468             usleep(290 * 1000);
469             currentStatus->speed = 0;
470             pthread_mutex_unlock(&speedMutex);
471             if (DEBUG)
472             {
473                 printf("Emergency Stop succeed\n");
474             }
475         }
476         break;
477     case 5:
478         if (frontCM < frontCautionCM || frontLeftCM < frontLeftCautionCM
479             || frontRightCM < frontRightCautionCM)
480         {
481             if (DEBUG)
482             {
483                 printf("Detect obstacle: Front\n");
484             }
485             pthread_mutex_lock(&speedMutex);
486             currentStatus->speed = -5;
487             currentStatus->caution = true;
488             playSound();
489             usleep(350 * 1000);
490             currentStatus->speed = 0;
491             pthread_mutex_unlock(&speedMutex);
492             if (DEBUG)
493             {
494

```

```

497         printf("Emergency Stop succeed\n");
498     }
499 }
500     break;
501 }
502 }
503 }
504 }
505 if (currentStatus->lightsAutoOn)
506 {
507     if (digitalRead(PIN_LIGHTSENSOR) == 0 && !currentStatus->lightsOn)
508     {
509         action(LIGHTS_ON);
510     } else if(digitalRead(PIN_LIGHTSENSOR) == 1 && currentStatus->lightsOn)
511     {
512         action(LIGHTS_OFF);
513     }
514 }
515 }
516 }
517 }
518 }
519 void *receiveCommand(void *vargp)
520 {
521     struct status *currentStatus = (status*) vargp;
522     ultrasensor **ultrasensors = currentStatus->ultrasensors;
523
524     while (currentStatus->connected)
525     {
526         int data = getData(currentStatus->newsockFD);
527         int response = action(data);
528     }
529 }
530 }
531 }
532 void *speedRegulatedDriving(void *vargp)
533 {
534     struct status *currentStatus = (status*) vargp;
535
536     while (currentStatus->connected)
537     {
538         while (currentStatus->speed > 0)
539         {
540             digitalWrite(PIN_FORWARD, HIGH);
541             digitalWrite(PIN_BACKWARD, LOW);
542             switch (currentStatus->speed)
543             {
544                 case 1:
545                     usleep(5 * 1000);
546                     break;
547                 case 2:
548                     usleep(5 * 1000);
549                     break;
550                 case 3:
551                     usleep(8 * 1000);
552                     break;
553                 case 4:
554                     usleep(11 * 1000);
555                     break;
556                 case 5:
557                     usleep(0);
558                     break;
559             }
560
561             if (currentStatus->speed != 5)
562             {
563                 digitalWrite(PIN_FORWARD, LOW);
564
565                 switch (currentStatus->speed)
566                 {
567                     case 1:
568                         usleep(15 * 1000);
569                         break;
570                     case 2:
571                         usleep(10 * 1000);
572                         break;
573                     case 3:
574                         usleep(10 * 1000);
575                         break;
576                     case 4:
577                         usleep(10 * 1000);
578                         break;
579                 }
580             }
581         }
582     }
583 }
584 }
```

```

580     }
581 }
582 while (currentStatus->speed < 0)
{
585     digitalWrite(PIN_FORWARD, LOW);
586     digitalWrite(PIN_BACKWARD, HIGH);
587     switch (currentStatus->speed)
588     {
589         case -1:
590             usleep(5 * 1000);
591             break;
592         case -2:
593             usleep(5 * 1000);
594             break;
595         case -3:
596             usleep(8 * 1000);
597             break;
598         case -4:
599             usleep(11 * 1000);
600             break;
601         case -5:
602             usleep(0);
603             break;
604     }
605
606     if (currentStatus->speed != -5)
607     {
608         digitalWrite(PIN_BACKWARD, LOW);
609
610         switch (currentStatus->speed)
611         {
612             case -1:
613                 usleep(15 * 1000);
614                 break;
615             case -2:
616                 usleep(10 * 1000);
617                 break;
618             case -3:
619                 usleep(10 * 1000);
620                 break;
621             case -4:
622                 usleep(10 * 1000);
623                 break;
624         }
625     }
626
627     if (currentStatus->speed == 0)
628     {
629         digitalWrite(PIN_BACKWARD, LOW);
630         digitalWrite(PIN_FORWARD, LOW);
631     }
632 }
633 }
634
635 void *sendCurrentStats(void *vargp)
636 {
637     struct status *currentStatus = (status*) vargp;
638     ultrasensor **ultrasensors = currentStatus->ultrasensors;
639     cJSON *root, *fmt;
640
641     root = cJSON_CreateObject();
642     cJSON_AddItemToObject(root, "distances", fmt = cJSON_CreateObject());
643     cJSON_AddNumberToObject(fmt, "front", ultrasensors[FRONTSENSOR]->lastCM);
644     cJSON_AddNumberToObject(fmt, "front-right",
645         ultrasensors[FRONTRIGHTSENSOR]->lastCM);
646     cJSON_AddNumberToObject(fmt, "front-left",
647         ultrasensors[FRONTELEFTSENSOR]->lastCM);
648     cJSON_AddNumberToObject(fmt, "back", ultrasensors[BACKSENSOR]->lastCM);
649     pthread_mutex_lock(&modeMutex);
650     cJSON_AddNumberToObject(root, "currentSpeed", currentStatus->speed);
651     cJSON_AddNumberToObject(root, "currentMode", currentStatus->mode);
652     cJSON_AddNumberToObject(root, "caution", currentStatus->caution);
653     cJSON_AddNumberToObject(root, "lights", currentStatus->lightsOn);
654     cJSON_AddNumberToObject(root, "lightsautomation",
655         currentStatus->lightsAutoOn);
656     cJSON_AddNumberToObject(root, "ridinPlaying", currentStatus->ridinPlaying);
657     cJSON_AddNumberToObject(root, "hornPlaying", currentStatus->hornPlaying);
658     pthread_mutex_unlock(&modeMutex);
659     char *rendered = cJSON_Print(root);
660     char newStr[strlen(rendered) + 1];
661     sprintf(newStr, "%s\n", rendered);
662     sendData(currentStatus->newsockFD, newStr);
}

```

```
663     cJSON_Delete( root );
664     free( rendered );
665 }
```

5.2 Quellcode der Android App

JAVA–Code/MMainActivity.java

```
1 package com.camera.simplemjpeg;
2
3 import android.app.AlertDialog;
4 import android.content.DialogInterface;
5 import android.content.SharedPreferences;
6 import android.content.pm.ActivityInfo;
7 import android.content.res.Configuration;
8 import android.graphics.Color;
9 import android.os.AsyncTask;
10 import android.os.Bundle;
11 import android.support.v7.app.AppCompatActivity;
12 import android.util.Log;
13 import android.view.Menu;
14 import android.view.MenuItem;
15 import android.view MotionEvent;
16 import android.view.View;
17 import android.view.Window;
18 import android.view.WindowManager;
19 import android.widget.Button;
20 import android.widget.CompoundButton;
21 import android.widget.EditText;
22 import android.widget.ImageView;
23 import android.widget.LinearLayout;
24 import android.widget.RelativeLayout;
25 import android.widget.Switch;
26 import android.widget.TextView;
27 import android.widget.Toast;
28
29 import org.apache.http.HttpResponse;
30 import org.apache.http.client.ClientProtocolException;
31 import org.apache.http.client.methods.HttpGet;
32 import org.apache.http.impl.client.DefaultHttpClient;
33 import org.json.JSONException;
34 import org.json.JSONObject;
35
36 import java.io.BufferedReader;
37 import java.io.IOException;
38 import java.io.InputStream;
39 import java.io.InputStreamReader;
40 import java.io.OutputStream;
41 import java.net.Socket;
42 import java.net.URI;
43 import java.util.Timer;
44 import java.util.TimerTask;
45
46 public class MainActivity extends AppCompatActivity {
47     public static final String TAG = "MainActivity";
48     public static boolean debug = false;
49     private static final String PREF_URL = "url";
50     private static final String URL_DEFAULT = "http://";
51     private static String url;
52     private MjpegView mv; // video-View, displays webcam-jpeg stream
53     private SpeechRecognition speechRec; // same object for speech recognition of
      driving and other
54     private boolean switchChecked; // true = microphone for driving | false =
      microphone for other
55     private boolean suspending = false;
56
57     private Socket socket;
58     private OutputStream outputStream;
59     private InputStream inputStream;
60     private BufferedReader buffer;
61
62     private Thread inputThread;
63
64     // Input received over socket
65     private int distances[] = new int[4];
66     private int mode, speed;
67     private int lights, lightsauto=1; // lights 1=on/0=off | lightsauto 1=on/0=off
68     private int caution;
69     private int musicPlaying, hornPlaying; // musicPlaying 1=on/0=off | hornPlaying
      1=on/0=off. @deprecated hornPlaying replaced by hornPlayingIntern
70     private boolean hornPlayingIntern=false;
71     private Timer hornTimer;
72     // gears
73     private int gear=1; // current gear (default=1)
74     // driving
75     private int driving=0; // -1=driving backward, 0=not driving, 1=driving forward
```

```

76
77     public void onCreate(Bundle bundle) {
78         requestWindowFeature(Window.FEATURE_NO_TITLE);
79         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FORCE_NOT_FULLSCREEN,
80             WindowManager.LayoutParams.FLAG_FORCE_NOT_FULLSCREEN);
81         super.onCreate(bundle);
82         String tmp = getPreferences(0).getString(PREF_URL, URL_DEFAULT);
83         url = tmp.replaceAll("\\s", " ");
84         setTitle(url);
85         setContentView(R.layout.main);
86         mv = (MjpegView) findViewById(R.id.mv);
87         new DoRead().execute(url);
88         hornTimer = new Timer();
89         // Pack this in xml later!
90         RelativeLayout rLayout = (RelativeLayout) findViewById(R.id.rHUDLayout);
91         for(int i=0; i<rLayout.getChildCount(); i++){
92             rLayout.getChildAt(i).setVisibility(View.INVISIBLE);
93         }
94         speechRec = new SpeechRecognition(this, getApplicationContext(), true);
95         //
96         new InitSocketThread().execute(url);
97         //
98         // new InputSocketThread().execute("");
99         //
100        initControlsListener();
101    }
102
103    public void onPause() {
104        super.onPause();
105        if (mv != null){
106            if (mv.isStreaming()) {
107                mv.stopPlayback();
108                suspending = true;
109            }
110        }
111    }
112
113    public void onResume() {
114        super.onResume();
115        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT); // always
116        start application in portrait mode!
117        if (mv != null){
118            if (suspending) {
119                new DoRead().execute(url);
120                suspending = false;
121            }
122        }
123    }
124
125    public void onDestroy() {
126        if (mv != null){
127            mv.freeCameraMemory();
128        }
129        super.onDestroy();
130        Log.e(TAG, "speechRec: " + speechRec + " speechRec.getRecognizer(): " +
131             speechRec.getRecognizer());
132        speechRec.getRecognizer().cancel();
133        speechRec.getRecognizer().shutdown();
134        closeSocketConnection();
135    }
136
137    /* Creates the menu items */
138    public boolean onCreateOptionsMenu(Menu menu) {
139        getMenuInflater().inflate(R.menu.main, menu);
140        return true;
141    }
142
143    /* Handles item selections */
144    public boolean onOptionsItemSelected(MenuItem item) {
145        // finish()
146        int id = item.getItemId();
147        if(id == R.id.changeurl) {
148            changeurlDialog();
149        } else if(id == R.id.debug){
150            if(item.isChecked()){
151                item.setChecked(false);
152                debug = false;
153            } else{
154                item.setChecked(true);
155                debug = true;
156            }
157        }
158    }

```

```

156     } else if (id == R.id.action_refresh){
157         if (mv != null) {
158             if (mv.isStreaming()) {
159                 Toast.makeText(this, "Refreshed", Toast.LENGTH_SHORT).show();
160                 closeSocketConnection();
161                 mv.stopPlayback();
162                 new DoRead().execute(url);
163                 new InitSocketThread().execute(url);
164                 new InputSocketThread().execute("");
165             }
166         }
167         return true;
168     }
169     return super.onOptionsItemSelected(item);
170 }
171
172 // change url Dialog
173 private void changeurlDialog() {
174     mv.stopPlayback();
175     final EditText editText = new EditText(MainActivity.this);
176     editText.setText(url);
177     editText.setSelection(editText.getText().length());
178     LinearLayout.LayoutParams lp = new
179         LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH_PARENT,
180             LinearLayout.LayoutParams.MATCH_PARENT);
181     editText.setLayoutParams(lp);
182
183     AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
184     builder.setTitle("URL");
185     builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
186         @Override
187         public void onClick(DialogInterface dialog, int which) {
188             closeSocketConnection();
189             mv.stopPlayback();
190             String text = editText.getText().toString();
191             if(text == null || text.equalsIgnoreCase("")){
192                 text = new String(URL_DEFAULT);
193             }
194             SharedPreferences prefs = getPreferences(0);
195             SharedPreferences.Editor editor = prefs.edit();
196             url = new String(text);
197             editor.putString(PREF_URL, url);
198             editor.commit();
199             setTitle(url);
200             try{
201                 new InitSocketThread().execute(url);
202                 new DoRead().execute(url);
203                 new InputSocketThread().execute("");
204             } catch(IllegalThreadStateException e){ // user entered correct url
205                 Log.e(TAG, "Thread already started!");
206                 e.printStackTrace();
207                 dialog.cancel();
208             }
209         }
210     }).setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
211         @Override
212         public void onClick(DialogInterface dialog, int which) {
213             dialog.cancel();
214         }
215     });
216     AlertDialog dialog = builder.create();
217     dialog.setView(editText);
218     dialog.show();
219 }
220
221 // Subclass that deals with the mjpeg streaming on a separate Thread
222 public class DoRead extends AsyncTask<String, Void, MjpegInputStream> {
223     protected MjpegInputStream doInBackground(String... url) {
224         HttpResponse res = null;
225         DefaultHttpClient httpclient = new DefaultHttpClient();
226         Log.d(TAG, "1. Sending http request");
227         Log.d(TAG, "url: "+new HttpGet(URI.create(url[0])).toString());
228         try {
229             res = httpclient.execute(new
230                 HttpGet(URI.create(url[0]+":"+Constants.PORT_MOTION))); // res.getStatusLine().getStatusCode()==401 if server requires
231                 // username:password
232             Log.d(TAG, "2. Request finished, status = " +
233                 res.getStatusLine().getStatusCode());
234             // request successfull ...
235         }
236     }

```

```

231     return new MjpegInputStream(res.getEntity().getContent());
232 } catch (ClientProtocolException e) {
233     e.printStackTrace();
234     Log.d(TAG, "Request failed -ClientProtocolException", e);
235 } catch (IOException e) { // stream service is offline / client has not
236     e.printStackTrace();
237     Log.d(TAG, "Request failed -IOException", e);
238 } catch (IllegalArgumentException e){
239     e.printStackTrace();
240     Log.d(TAG, "Host name was null");
241 }
242     return null;
243 }
244 protected void onPostExecute(final MjpegInputStream result) {
245     Log.i(TAG, "Result: " + result);
246     if(result == null){
247         runOnUiThread(new Runnable() {
248             @Override
249             public void run() {
250                 networkErrorDialog();
251             }
252         });
253     } else{
254         result.setSkip(1);
255         mv.setSource(result);
256         getSupportActionBar().show();
257         mv.setDisplayMode(MjpegView.SIZE_BEST_FIT);
258         mv.showFPS(true);
259     }
260     setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_SENSOR);
261 }
262 }
263
264 public class SocketThread extends AsyncTask<String, Void, Integer> {
265     protected Integer doInBackground(String... strings) {
266         String string = new String(strings[0]);
267         Log.e(TAG, "SocketThread starts here! String you want to send: " + string);
268         if(string!=null && !string.equalsIgnoreCase("")){
269             try{
270                 switch(string){
271                     case "go":
272                         driving = 1;
273                         switch(gear){
274                             case 2: outputStream.write(Constants.GO_2); break;
275                             case 3: outputStream.write(Constants.GO_3); break;
276                             case 4: outputStream.write(Constants.GO_4); break;
277                             case 5: outputStream.write(Constants.GO_5); break;
278                             default: outputStream.write(Constants.GO_1);
279                         }
280                         break;
281                     case "back":
282                         driving = -1;
283                         switch(gear){
284                             case 2: outputStream.write(Constants.BACK_2); break;
285                             case 3: outputStream.write(Constants.BACK_3); break;
286                             case 4: outputStream.write(Constants.BACK_4); break;
287                             case 5: outputStream.write(Constants.BACK_5); break;
288                             default: outputStream.write(Constants.BACK_1);
289                         }
290                         break;
291                     case "stand":
292                         driving = 0;
293                         outputStream.write(Constants.STOP); break;
294                     case "left": outputStream.write(Constants.LEFT); break;
295                     case "right": outputStream.write(Constants.RIGHT); break;
296                     case "line up": outputStream.write(Constants.STRAIGHT); break;
297                     case "lights on": outputStream.write(Constants.LIGHTS_ON);
298                         break;
299                     case "lights off": outputStream.write(Constants.LIGHTS_OFF);
300                         break;
301                     case "lights auto on":
302                         outputStream.write(Constants.LIGHTS_AUTO_ON); break;
303                     case "lights auto off":
304                         outputStream.write(Constants.LIGHTS_AUTO_OFF); break;
305                     // for the following commands, ui elements have to be changed
306                     // additionally (From here on are the speech-recognized words)
307                     case "beep": outputStream.write(Constants.HORN); break;
308                     case "on": outputStream.write(Constants.LIGHTS_AUTO_ON);
309                         outputStream.write(Constants.LIGHTS_ON);
310                         outputStream.flush();
311                     return 2;
312                     case "off": outputStream.write(Constants.LIGHTS_OFF);

```

```

308                     outputStream.write(Constants.LIGHTS_AUTO_OFF);
309                     outputStream.flush();
310                     return 3;
311                 case "auto": outputStream.write(Constants.LIGHTS_AUTO_ON);
312                     outputStream.flush();
313                     return 4;
314             // music
315             case "music": outputStream.write(Constants.MUSIC_ON); break;
316             // sound off
317             case "sound_off": outputStream.write(Constants.SOUND_OFF);
318                 break;
319             }
320             outputStream.flush();
321         } catch (IOException e){
322             Log.e(TAG, "Exception sending "+string+" over socket");
323             e.printStackTrace();
324             return -1;
325         }
326     }
327     return 0;
328 }
329 @Override
330 protected void onPostExecute(Integer result) {
331     switch(result){
332         case -1: Log.e(TAG, "Failed using the Socket Connection"); break;
333         case 0: Log.e(TAG, "Successfully used the Socket Connection"); break;
334         case 2: runOnUI(true, false, false); break;
335         case 3: runOnUI(false, true, false); break;
336         case 4: runOnUI(false, false, true); break;
337     }
338 }
339 public void runOnUI(final boolean on, final boolean off, final boolean auto){
340     runOnUiThread(new Runnable() {
341         @Override
342         public void run() {
343             if(on)
344                 ((ImageView) findViewById(R.id.lights)).setImageResource(R.drawable.ic_action_lights_on);
345             if(off)
346                 ((ImageView) findViewById(R.id.lights)).setImageResource(R.drawable.ic_action_lights_off);
347             if(auto)
348                 ((ImageView) findViewById(R.id.lights)).setImageResource(R.drawable.ic_action_lights_auto);
349         }
350     });
351 }
352 }
353 public class InitSocketThread extends AsyncTask<String, Void, Integer> {
354     protected Integer doInBackground(String... url) {
355         try {
356             String ip = new String(url[0].substring(7));
357             Log.e(TAG, "ip:"+ip);
358             socket = new Socket(ip, Constants.PORT_SOCKET);
359             outputStream = socket.getOutputStream();
360             inputStream = socket.getInputStream();
361             buffer = new BufferedReader(new InputStreamReader(inputStream));
362         } catch (IOException e) {
363             Log.e(TAG, "IOException Socket InitSocketThread.class ");
364             e.printStackTrace();
365             return -1;
366         }
367         return 0;
368     }
369     @Override
370     protected void onPostExecute(Integer result) {
371         if(result == 0){
372             Log.e(TAG, "InitSocketThread-onPostExecute(): Successfully initialized
373                 a Socket Connection");
374             inputThread = new Thread(new Runnable() {
375                 @Override
376                 public void run() {
377                     JSONObject json = new JSONObject();
378                     StringBuilder sb;
379                     String newLine;
380                     final ImageView fritz = ((ImageView) findViewById(R.id.fritz));
381                     final TextView debugTextView = ((TextView)
382                         findViewById(R.id.debug_textfield));
383                     try {
384                         while (true) {
385                             sb = new StringBuilder();
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2288
2289
2290
2291
2292
2293
2294
2295
2296
2296
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2
```

```

385     if((newLine = buffer.readLine()).charAt(0) == '{') {
386         sb.append(newLine);
387         while((newLine = buffer.readLine()).charAt(0) != '}'){
388             sb.append(newLine);
389         }
390         sb.append(newLine);
391     }
392     json = new JSONObject(sb.toString());
393     distances[0] =
394         json.getJSONObject("distances").getInt("front");
395     distances[1] =
396         json.getJSONObject("distances").getInt("front-right");
397     distances[2] =
398         json.getJSONObject("distances").getInt("front-left");
399     distances[3] =
400         json.getJSONObject("distances").getInt("back");
401     mode = (json.getInt("currentMode")); // left , right ,
402         line up
403     speed = (json.getInt("currentSpeed")); // [-5,...,5]
404         forward or backward
405     caution = (json.getInt("caution"));
406     lights = (json.getInt("lights"));
407     lightsauto = (json.getInt("lightsautomation"));
408     musicPlaying = (json.getInt("ridinPlaying")); // 1 if
409         currently playing
410     hornPlaying = (json.getInt("hornPlaying")); // 1 if
411         currently playing
412     Log.i(TAG, "Front: " + distances[0] + " Front-Right: "
413         + distances[1] + " Front-Left: " + distances[2] + "
414         Back: " + distances[3] + " Mode: " + mode +
415         " Speed: " + speed + " Caution: " + caution +
416         " Lights: " + lights + " Automatic Lights: " +
417         lightsauto + " HornPlaying: " + hornPlaying + "
418         RidinPlaying: " + musicPlaying);
419     runOnUiThread(new Runnable() {
420         @Override
421         public void run() {
422             if(caution==1)
423                 fritz.setVisibility(View.VISIBLE);
424             else fritz.setVisibility(View.INVISIBLE);
425             if(hornPlaying==1) ((ImageView)
426                 findViewById(R.id.horn)).setImageResource(R.drawable.ic_action_horn_hold);
427             else ((ImageView)
428                 findViewById(R.id.horn)).setImageResource(R.drawable.ic_action_horn);
429             if(musicPlaying==1)((ImageView)
430                 findViewById(R.id.music)).setImageResource(R.drawable.ic_action_music);
431             else ((ImageView)
432                 findViewById(R.id.music)).setImageResource(R.drawable.ic_action_music);
433             ImageView lightsView = (ImageView)
434                 findViewById(R.id.lights);
435             if(lightsauto==1)
436                 lightsView.setImageResource(R.drawable.ic_action_lights_auto);
437             if(lights==1 && lightsauto==0)
438                 lightsView.setImageResource(R.drawable.ic_action_lights_on);
439             if(lights==0 && lightsauto==0)
440                 lightsView.setImageResource(R.drawable.ic_action_lights_off);
441             if(debug) debugTextView.setText("Front: " +
442                 distances[0] + " Front-Right: " +
443                 distances[1] + " Front-Left: " +
444                 distances[2] + " Back: " + distances[3] + "
445                 Mode: " + mode + " Speed: " + speed + "
446                 Caution: " + caution + " Lights: " + lights +
447                 " Automatic Lights: " + lightsauto + "
448                 HornPlaying: " + hornPlaying + "
449                 RidinPlaying: " + musicPlaying);
450         }
451     });
452 }
453 } catch (IOException e) {
454     Log.e(TAG, "IOException in Socket-Input | e: "+e);
455     e.printStackTrace();
456 } catch(JSONException e) {
457     Log.e(TAG, "JSONException in Socket-Input | e: "+e);
458     e.printStackTrace();
459 } catch(NullPointerException e){
460     Log.e(TAG, "NullPointerException in Socket-Input, Server
461         closed the connection. | buffer: " + buffer);
462     e.printStackTrace();
463     runOnUiThread(new Runnable() {
464         @Override

```

```

435             public void run() {
436                 networkErrorDialog();
437             }
438         });
439     }
440 }
441 });
442 inputThread.start();
443 } else{
444     Log.e(TAG, "InitSocketThread-onPostExecute(): Failed at establishing a
445             Socket Connection");
446     networkErrorDialog();
447 }
448 }
449 }
450 }
451 @Override
452 public void onConfigurationChanged(Configuration config) {
453     super.onConfigurationChanged(config);
454     Log.e(TAG, "onConfigurationChanged(): " + config.orientation);
455     RelativeLayout rLayout = (RelativeLayout) findViewById(R.id.rHUDLayout);
456     Log.e(TAG, "rLayout: " + rLayout + " Childcount: " + rLayout.getChildCount());
457     if(config.orientation == Configuration.ORIENTATION_LANDSCAPE){
458         for(int i=0; i<rLayout.getChildCount(); i++){
459             View child = rLayout.getChildAt(i);
460             if(child.getId() == R.id.fritz){
461                 continue;
462             } else if(child.getId() == R.id.debug_textfield){
463                 if(!debug) continue;
464                 else child.setVisibility(View.VISIBLE);
465             }
466             child.setVisibility(View.VISIBLE);
467         }
468         getSupportActionBar().hide();
469         getWindow().clearFlags(WindowManager.LayoutParams.FLAG_FORCE_NOT_FULLSCREEN);
470         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
471             WindowManager.LayoutParams.FLAG_FULLSCREEN);
472         mv.setDisplayMode(MjpegView.SIZE_FULLSCREEN);
473     } else{
474         for(int i=0; i<rLayout.getChildCount(); i++){
475             rLayout.getChildAt(i).setVisibility(View.INVISIBLE);
476         }
477         getSupportActionBar().show();
478         getWindow().clearFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
479         getWindow().setFlags(WindowManager.LayoutParams.FLAG_FORCE_NOT_FULLSCREEN,
480             WindowManager.LayoutParams.FLAG_FULLSCREEN);
481         mv.setDisplayMode(MjpegView.SIZE_BEST_FIT);
482     }
483 }
484 private void networkErrorDialog(){
485     AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
486     builder.setTitle("Network Error");
487     builder.setMessage("Make sure you are connected to the network and the server
488             is running!");
489     builder.setPositiveButton("Retry", new DialogInterface.OnClickListener() {
490         @Override
491         public void onClick(DialogInterface dialog, int which) {
492             closeSocketConnection();
493             if(mv!=null) mv.stopPlayback();
494             new InitSocketThread().execute(url);
495             new DoRead().execute(url);
496         }
497     }).setNegativeButton("Change url", new DialogInterface.OnClickListener() {
498         @Override
499         public void onClick(DialogInterface dialog, int which) {
500             dialog.cancel();
501             changeurlDialog();
502         }
503     });
504     AlertDialog dialog = builder.create();
505     dialog.setCancelable(false);
506     dialog.show();
507 }
508 private void initControlsListener(){
509     Log.e(TAG, "initControlsListener()");
510     // microphone button
511     final ImageView micView = ((ImageView) findViewById(R.id.microphone));
512     micView.setOnTouchListener(new View.OnTouchListener() {
513         @Override
514         public boolean onTouch(View v, MotionEvent event) {

```

```

514         switch(event.getAction()) {
515             case MotionEvent.ACTION_UP:
516                 micView.setImageResource(R.drawable.ic_action_mic_new2);
517                 speechRec.getRecognizer().stop();
518                 return true;
519             case MotionEvent.ACTION_DOWN:
520                 micView.setImageResource(R.drawable.ic_action_mic_hold_new2);
521                 speechRec.getRecognizer().startListening(SpeechRecognition.ORDER_SEARCH);
522                 return true;
523         }
524     }
525 }
526 // switch button (below microphone)
527 final Switch switch1 = ((Switch) findViewById(R.id.switch1));
528 switch1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener()
529 {
530     @Override
531     public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
532         speechRec.getRecognizer().cancel();
533         speechRec.getRecognizer().shutdown();
534         if(isChecked) {
535             switchChecked = true;
536             speechRec = new SpeechRecognition(getApplicationContext(), true);
537         } else {
538             switchChecked = false;
539             speechRec = new SpeechRecognition(getApplicationContext(), getApplicationContext(), false);
540         }
541     }
542 });
543 // light button
544 final ImageView imgView_light = (ImageView) findViewById(R.id.lights);
545 imgView_light.setOnClickListener(new View.OnClickListener() {
546     @Override
547     public void onClick(View v) {
548         if(lightsauto == 1) {
549             new SocketThread().execute("lights_auto_off");
550             new SocketThread().execute("lights_off");
551         }
552         if(lightsauto == 0 && lights == 1) {
553             new SocketThread().execute("lights_auto_on");
554         }
555         if(lightsauto == 0 && lights == 0) {
556             new SocketThread().execute("lights_on");
557         }
558     }
559 });
560 // horn button
561 ((ImageView) findViewById(R.id.horn)).setOnClickListener(new
562 View.OnClickListener() {
563     @Override
564     public void onClick(View v) {
565         if(hornPlayingIntern) {
566             hornPlayingIntern = false;
567             new SocketThread().execute("sound_off");
568             ((ImageView)
569             findViewById(R.id.horn)).setImageResource(R.drawable.ic_action_horn);
570             hornTimer.cancel();
571             hornTimer.purge();
572         } else {
573             hornPlayingIntern = true;
574             new SocketThread().execute("beep");
575             ((ImageView)
576             findViewById(R.id.horn)).setImageResource(R.drawable.ic_action_horn_hold);
577             hornTimer.schedule(new TimerTask() {
578                 @Override
579                 public void run() {
580                     if(hornPlayingIntern) {
581                         hornPlayingIntern = false;
582                         new SocketThread().execute("sound_off");
583                         runOnUiThread(new Runnable() {
584                             @Override
585                             public void run() {
586                                 ((ImageView)
587                                 findViewById(R.id.horn)).setImageResource(R.drawable.ic_action_horn);
588                             });
589                         }, 2190);
590                     }
591                 });
592             }
593         }
594     }
595 });

```

```

590
591 //      if(hornPlaying==1){
592 //          new SocketThread().execute("sound_off");
593 //      } else{
594 //          new SocketThread().execute("beep");
595 //      }
596 }
597 });
598 // music button
599 ((ImageView) findViewById(R.id.music)).setOnClickListener(new
600     View.OnClickListener() {
601         @Override
602         public void onClick(View v) {
603             if(musicPlaying==1){
604                 new SocketThread().execute("sound_off");
605             } else{
606                 new SocketThread().execute("music");
607             }
608         }
609     });
610 final int arrowColor = 0x808080;
611 // arrow buttons
612 ((ImageView) findViewById(R.id.upArrow)).setOnTouchListener(new
613     View.OnTouchListener() {
614         @Override
615         public boolean onTouch(View v, MotionEvent event) {
616             switch(event.getAction()){
617                 case MotionEvent.ACTION_UP:
618                     ((ImageView)
619                         findViewById(R.id.upArrow)).setColorFilter(arrowColor);
620                     new SocketThread().execute("stand");
621                     return true;
622                 case MotionEvent.ACTION_DOWN:
623                     ((ImageView)
624                         findViewById(R.id.upArrow)).setColorFilter(Color.BLACK);
625                     new SocketThread().execute("go");
626                     return true;
627             }
628         }
629     });
630 ((ImageView) findViewById(R.id.downArrow)).setOnTouchListener(new
631     View.OnTouchListener() {
632         @Override
633         public boolean onTouch(View v, MotionEvent event) {
634             switch(event.getAction()){
635                 case MotionEvent.ACTION_UP:
636                     ((ImageView)
637                         findViewById(R.id.downArrow)).setColorFilter(arrowColor);
638                     new SocketThread().execute("stand");
639                     return true;
640                 case MotionEvent.ACTION_DOWN:
641                     ((ImageView)
642                         findViewById(R.id.downArrow)).setColorFilter(Color.BLACK);
643                     new SocketThread().execute("back");
644                     return true;
645             }
646         }
647     });
648 ((ImageView) findViewById(R.id.leftArrow)).setOnTouchListener(new
649     View.OnTouchListener() {
650         @Override
651         public boolean onTouch(View v, MotionEvent event) {
652             switch(event.getAction()){
653                 case MotionEvent.ACTION_UP:
654                     ((ImageView)
655                         findViewById(R.id.leftArrow)).setColorFilter(arrowColor);
656                     new SocketThread().execute("line up");
657                     return true;
658                 case MotionEvent.ACTION_DOWN:
659                     ((ImageView)
660                         findViewById(R.id.leftArrow)).setColorFilter(Color.BLACK);
661                     new SocketThread().execute("left");
662                     return true;
663             }
664         }
665     });
666 ((ImageView) findViewById(R.id.rightArrow)).setOnTouchListener(new
667     View.OnTouchListener() {
668         @Override

```

```

662     public boolean onTouch(View v, MotionEvent event) {
663         switch(event.getAction()) {
664             case MotionEvent.ACTION_UP:
665                 ((ImageView)
666                     findViewById(R.id.rightArrow)).setColorFilter(arrowColor);
667                 new SocketThread().execute("line up");
668                 return true;
669             case MotionEvent.ACTION_DOWN:
670                 ((ImageView)
671                     findViewById(R.id.rightArrow)).setColorFilter(Color.BLACK);
672                 new SocketThread().execute("right");
673                 return true;
674         }
675     });
676 }
677 // gear buttons
678 final Button gearButton1 = ((Button) findViewById(R.id.gearButton1));
679 final Button gearButton2 = ((Button) findViewById(R.id.gearButton2));
680 final Button gearButton3 = ((Button) findViewById(R.id.gearButton3));
681 final Button gearButton4 = ((Button) findViewById(R.id.gearButton4));
682 final Button gearButton5 = ((Button) findViewById(R.id.gearButton5));
683
684 final float alpha = 0.4f;
685 gearButton2.setAlpha(alpha);
686 gearButton3.setAlpha(alpha);
687 gearButton4.setAlpha(alpha);
688 gearButton5.setAlpha(alpha);
689
690 gearButton1.setOnClickListener(new View.OnClickListener() {
691     @Override
692     public void onClick(View v) {
693         gear = 1;
694         gearButton1.setAlpha(1);
695         gearButton2.setAlpha(alpha);
696         gearButton3.setAlpha(alpha);
697         gearButton4.setAlpha(alpha);
698         gearButton5.setAlpha(alpha);
699         changeGearIfDriving();
700     }
701 });
702 gearButton2.setOnClickListener(new View.OnClickListener() {
703     @Override
704     public void onClick(View v) {
705         gear = 2;
706         gearButton1.setAlpha(alpha);
707         gearButton2.setAlpha(1);
708         gearButton3.setAlpha(alpha);
709         gearButton4.setAlpha(alpha);
710         gearButton5.setAlpha(alpha);
711         changeGearIfDriving();
712     }
713 });
714 gearButton3.setOnClickListener(new View.OnClickListener() {
715     @Override
716     public void onClick(View v) {
717         gear = 3;
718         gearButton1.setAlpha(alpha);
719         gearButton2.setAlpha(alpha);
720         gearButton3.setAlpha(1);
721         gearButton4.setAlpha(alpha);
722         gearButton5.setAlpha(alpha);
723         changeGearIfDriving();
724     }
725 });
726 gearButton4.setOnClickListener(new View.OnClickListener() {
727     @Override
728     public void onClick(View v) {
729         gear = 4;
730         gearButton1.setAlpha(alpha);
731         gearButton2.setAlpha(alpha);
732         gearButton3.setAlpha(alpha);
733         gearButton4.setAlpha(1);
734         gearButton5.setAlpha(alpha);
735         changeGearIfDriving();
736     }
737 });
738 gearButton5.setOnClickListener(new View.OnClickListener() {
739     @Override
740     public void onClick(View v) {
741         gear = 5;
742         gearButton1.setAlpha(alpha);

```

```

743         gearButton2.setAlpha(alpha);
744         gearButton3.setAlpha(alpha);
745         gearButton4.setAlpha(alpha);
746         gearButton5.setAlpha(1);
747         changeGearIfDriving();
748     });
749 }
750 }
751
752 private void changeGearIfDriving() {
753     if(driving==1){
754         new SocketThread().execute("go");
755     } else if(driving==-1){
756         new SocketThread().execute("back");
757     }
758 }
759
760 public boolean getSwitchChecked(){return switchChecked;}
761 private MainActivity getActivity(){return this;}
762
763
764
765 private void closeSocketConnection(){
766     if(inputThread!=null) inputThread.interrupt();
767     try {
768         if(outputStream!=null){
769             outputStream.write(Constants.CLOSESOCKET);
770             outputStream.flush();
771             outputStream.close();
772         }
773         if(socket!=null) socket.close();
774         if(inputStream!=null) inputStream.close();
775     } catch(IOException e) {
776         Log.e(TAG, "Unable to send Close Socket (77)");
777         e.printStackTrace();
778     }
779 }
780
781 public void sendOverSocket(String string){
782     Toast.makeText(getApplicationContext(), " "+string, Toast.LENGTH_SHORT).show();
783     new SocketThread().execute(string);
784 }
785 }

```

JAVA–Code/MjpegInputStream.java

```
1 package com.camera.simplempeg;
2
3 import android.graphics.Bitmap;
4 import android.graphics.BitmapFactory;
5 import android.util.Log;
6
7 import java.io.BufferedInputStream;
8 import java.io.ByteArrayInputStream;
9 import java.io.DataInputStream;
10 import java.io.IOException;
11 import java.io.InputStream;
12 import java.net.HttpURLConnection;
13 import java.net.URL;
14 import java.util.Properties;
15
16 /*
17 import java.net.URI;
18 import org.apache.http.HttpResponse;
19 import org.apache.http.client.ClientProtocolException;
20 import org.apache.http.client.methods.HttpGet;
21 import org.apache.http.impl.client.DefaultHttpClient;
22 */
23
24 public class MjpegInputStream extends DataInputStream {
25     private final byte[] SOI_MARKER = {(byte) 0xFF, (byte) 0xD8};
26     private final byte[] EOF_MARKER = {(byte) 0xFF, (byte) 0xD9};
27     private final String CONTENT_LENGTH = "Content-Length";
28     private final static int HEADER_MAX_LENGTH = 100;
29     //private final static int FRAME_MAX_LENGTH = 40000 + HEADER_MAX_LENGTH;
30     private final static int FRAME_MAX_LENGTH = 200000;
31     private int mContentSize = -1;
32     byte[] header = null;
33     byte[] frameData = null;
34     int headerLen = -1;
35     int headerLenPrev = -1;
36
37     int skip = 1;
38     int count = 0;
39
40     private static final String TAG = "MJPEG";
41     private static final boolean DEBUG = false;
42
43     static {
44         System.loadLibrary("ImageProc");
45     }
46
47     public native int pixeltobmp(byte[] jp, int l, Bitmap bmp);
48
49     public native void freeCameraMemory();
50
51     public static MjpegInputStream read(String surl) {
52         try {
53             URL url = new URL(surl);
54             HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
55             return new MjpegInputStream(urlConnection.getInputStream());
56         } catch (Exception e) {
57         }
58
59         return null;
60     }
61
62     public MjpegInputStream(InputStream in) {
63         super(new BufferedInputStream(in, FRAME_MAX_LENGTH));
64     }
65
66     private int getEndOfSequence(DataInputStream in, byte[] sequence)
67     throws IOException {
68
69         int seqIndex = 0;
70         byte c;
71         for(int i = 0; i < FRAME_MAX_LENGTH; i++) {
72             c = (byte) in.readUnsignedByte();
73             if(c == sequence[seqIndex]) {
74                 seqIndex++;
75                 if(seqIndex == sequence.length) {
76                     return i + 1;
77                 }
78             } else seqIndex = 0;
79         }
80     }
81
82 }
```

```

83         return -1;
84     }
85
86     private int getStartOfSequence(DataInputStream in, byte[] sequence)
87         throws IOException {
88         int end = getEndOfSeqeunce(in, sequence);
89         return (end < 0) ? (-1) : (end - sequence.length);
90     }
91
92     private int getEndOfSeqeunceSimplified(DataInputStream in, byte[] sequence)
93         throws IOException {
94         int startPos = mContentLength / 2;
95         int endPos = 3 * mContentLength / 2;
96
97         skipBytes(headerLen + startPos);
98
99         int seqIndex = 0;
100        byte c;
101        for(int i = 0; i < endPos - startPos; i++) {
102            c = (byte) in.readUnsignedByte();
103            if(c == sequence[seqIndex]) {
104                seqIndex++;
105            if(seqIndex == sequence.length) {
106
107                return headerLen + startPos + i + 1;
108            }
109        } else seqIndex = 0;
110    }
111
112
113    return -1;
114}
115
116
117    private int parseContentLength(byte[] headerBytes)
118        throws IOException, NumberFormatException, IllegalArgumentException {
119        ByteArrayInputStream headerIn = new ByteArrayInputStream(headerBytes);
120        Properties props = new Properties();
121        props.load(headerIn);
122        return Integer.parseInt(props.getProperty(CONTENT_LENGTH));
123    }
124
125    public Bitmap readMjpegFrame() throws IOException {
126        mark(FRAME_MAX_LENGTH);
127        int headerLen;
128        try {
129            headerLen = getStartOfSequence(this, SOI_MARKER);
130        } catch(IOException e) {
131            if(DEBUG) Log.d(TAG, "IOException in betting headerLen.");
132            reset();
133            return null;
134        }
135        reset();
136
137        if(header == null || headerLen != headerLenPrev) {
138            header = new byte[headerLen];
139            if(DEBUG) Log.d(TAG, "header renewed " + headerLenPrev + " -> " +
140            headerLen);
141        }
142        headerLenPrev = headerLen;
143        readFully(header);
144
145        int ContentLengthNew = -1;
146        try {
147            ContentLengthNew = parseContentLength(header);
148        } catch(NumberFormatException nfe) {
149            ContentLengthNew = getEndOfSeqeunceSimplified(this, EOF_MARKER);
150
151            if(ContentLengthNew < 0) {
152                if(DEBUG) Log.d(TAG, "Worst case for finding EOF_MARKER");
153                reset();
154                ContentLengthNew = getEndOfSeqeunce(this, EOF_MARKER);
155            }
156        } catch(IllegalArgumentException e) {
157            if(DEBUG) Log.d(TAG, "IllegalArgumentException in parseContentLength");
158            ContentLengthNew = getEndOfSeqeunceSimplified(this, EOF_MARKER);
159
160            if(ContentLengthNew < 0) {
161                if(DEBUG) Log.d(TAG, "Worst case for finding EOF_MARKER");
162                reset();
163                ContentLengthNew = getEndOfSeqeunce(this, EOF_MARKER);
164            }
165        } catch(IOException e) {

```

```

165         if(DEBUG) Log.d(TAG, "IOException in parseContentLength");
166         reset();
167         return null;
168     }
169     mContentLength = ContentLengthNew;
170     reset();
171
172     if(frameData == null) {
173         frameData = new byte[FRAME_MAX_LENGTH];
174         if(DEBUG) Log.d(TAG, "frameData newed cl=" + FRAME_MAX_LENGTH);
175     }
176     if(mContentLength + HEADER_MAX_LENGTH > FRAME_MAX_LENGTH) {
177         frameData = new byte[mContentLength + HEADER_MAX_LENGTH];
178         if(DEBUG) Log.d(TAG, "frameData renewed cl=" + (mContentLength +
179                         HEADER_MAX_LENGTH));
180     }
181     skipBytes(headerLen);
182
183     readFully(frameData, 0, mContentLength);
184
185     if(count++ % skip == 0) {
186         return BitmapFactory.decodeStream(new ByteArrayInputStream(frameData, 0,
187             mContentLength));
188     } else {
189         return null;
190     }
191 }
192 public int readMjpegFrame(Bitmap bmp) throws IOException {
193     mark(FRAME_MAX_LENGTH);
194     int headerLen;
195     try {
196         headerLen = getStartOfSequence(this, SOI_MARKER);
197     } catch(IOException e) {
198         if(DEBUG) Log.d(TAG, "IOException in betting headerLen.");
199         reset();
200         return -1;
201     }
202     reset();
203
204     if(header == null || headerLen != headerLenPrev) {
205         header = new byte[headerLen];
206         if(DEBUG) Log.d(TAG, "header renewed " + headerLenPrev + " -> " +
207             headerLen);
208     }
209     headerLenPrev = headerLen;
210     readFully(header);
211
212     int ContentLengthNew = -1;
213     try {
214         ContentLengthNew = parseContentLength(header);
215     } catch(NumberFormatException nfe) {
216         ContentLengthNew = getEndOfSequenceSimplified(this, EOF_MARKER);
217
218         if(ContentLengthNew < 0) {
219             if(DEBUG) Log.d(TAG, "Worst case for finding EOF_MARKER");
220             reset();
221             ContentLengthNew = getEndOfSequence(this, EOF_MARKER);
222         }
223     } catch(IllegalArgumentException e) {
224         if(DEBUG) Log.d(TAG, "IllegalArgumentException in parseContentLength");
225         ContentLengthNew = getEndOfSequenceSimplified(this, EOF_MARKER);
226
227         if(ContentLengthNew < 0) {
228             if(DEBUG) Log.d(TAG, "Worst case for finding EOF_MARKER");
229             reset();
230             ContentLengthNew = getEndOfSequence(this, EOF_MARKER);
231         }
232     } catch(IOException e) {
233         if(DEBUG) Log.d(TAG, "IOException in parseContentLength");
234         reset();
235         return -1;
236     }
237     mContentLength = ContentLengthNew;
238     reset();
239
240     if(frameData == null) {
241         frameData = new byte[FRAME_MAX_LENGTH];
242         if(DEBUG) Log.d(TAG, "frameData newed cl=" + FRAME_MAX_LENGTH);
243     }
244     if(mContentLength + HEADER_MAX_LENGTH > FRAME_MAX_LENGTH) {
245         frameData = new byte[mContentLength + HEADER_MAX_LENGTH];

```

```
245         if(DEBUG) Log.d(TAG, "frameData renewed cl=" + (mContentLength +
246             HEADER_MAX_LENGTH));
247     }
248     skipBytes(headerLen);
249     readFully(frameData, 0, mContentLength);
250     if(count++ % skip == 0) {
251         return pixelToBmp(frameData, mContentLength, bmp);
252     } else {
253         return 0;
254     }
255 }
256 public void setSkip(int s) {
257     skip = s;
258 }
259 }
```

JAVA–Code/MjpegView.java

```
1 package com.camera.simplempeg;
2
3 import android.content.Context;
4 import android.graphics.Bitmap;
5 import android.graphics.Canvas;
6 import android.graphics.Color;
7 import android.graphics.Paint;
8 import android.graphics.PorterDuff;
9 import android.graphics.PorterDuffXfermode;
10 import android.graphics.Rect;
11 import android.graphics.Typeface;
12 import android.util.AttributeSet;
13 import android.view.SurfaceHolder;
14 import android.view.SurfaceView;
15
16 import java.io.IOException;
17
18 public class MjpegView extends SurfaceView implements SurfaceHolder.Callback {
19
20     public static final String TAG = "MJPEG";
21
22     public final static int POSITION_UPPER_LEFT = 9;
23     public final static int POSITION_UPPER_RIGHT = 3;
24     public final static int POSITION_LOWER_LEFT = 12;
25     public final static int POSITION_LOWER_RIGHT = 6;
26
27     public final static int SIZE_STANDARD = 1;
28     public final static int SIZE_BEST_FIT = 4;
29     public final static int SIZE_FULLSCREEN = 8;
30
31     SurfaceHolder holder;
32     Context saved_context;
33
34     private MjpegViewThread thread;
35     private MjpegInputStream mIn = null;
36     private boolean showFps = false;
37     private boolean mRun = false;
38     private boolean surfaceDone = false;
39
40     private Paint overlayPaint;
41     private int overlayTextColor;
42     private int overlayBackgroundColor;
43     private int ovPos;
44     private int dispWidth;
45     private int dispHeight;
46     private int displayMode;
47
48     private boolean suspending = false;
49
50     private Bitmap bmp = null;
51
52     // image size
53
54     public int IMG_WIDTH = 640;
55     public int IMG_HEIGHT = 480;
56
57     public class MjpegViewThread extends Thread {
58         private SurfaceHolder mSurfaceHolder;
59         private int frameCounter = 0;
60         private long start;
61         private String fps = "00";
62
63         public MjpegViewThread(SurfaceHolder surfaceHolder, Context context) {
64             mSurfaceHolder = surfaceHolder;
65         }
66     }
67
68     private Rect destRect(int bmw, int bmh) {
69         int tempx;
70         int tempy;
71         if (displayMode == MjpegView.SIZE_STANDARD) {
72             tempx = (dispWidth / 2) - (bmw / 2);
73             tempy = (dispHeight / 2) - (bmh / 2);
74             return new Rect(tempx, tempy, bmw + tempx, bmh + tempy);
75         }
76         if (displayMode == MjpegView.SIZE_BEST_FIT) {
77             float bmasp = (float) bmw / (float) bmh;
78             bmw = dispWidth;
79             bmh = (int) (dispWidth / bmasp);
80             if (bmh > dispHeight) {
81                 bmh = dispHeight;
82                 bmw = (int) (dispHeight * bmasp);
```

```

83
84         }
85         tempx = ( dispWidth / 2) - (bmw / 2);
86         tempy = ( dispHeight / 2) - (bmh / 2);
87         return new Rect(tempx, tempy, bmw + tempx, bmh + tempy);
88     }
89     if (displayMode == MjpegView.SIZE_FULLSCREEN)
90         return new Rect(0, 0, dispWidth, dispHeight);
91     return null;
92 }
93 public void setSurfaceSize(int width, int height) {
94     synchronized (mSurfaceHolder) {
95         dispWidth = width;
96         dispHeight = height;
97     }
98 }
99
100 private Bitmap makeFpsOverlay(Paint p) {
101     Rect b = new Rect();
102     p.getTextBounds(fps, 0, fps.length(), b);
103
104 // false indentation to fix forum layout
105     Bitmap bm = Bitmap.createBitmap(b.width(), b.height(),
106                                     Bitmap.Config.ARGB_8888);
107
108     Canvas c = new Canvas(bm);
109     p.setColor(overlayBackgroundColor);
110     c.drawRect(0, 0, b.width(), b.height(), p);
111     p.setColor(overlayTextColor);
112     c.drawText(fps, -b.left, b.bottom - b.top - p.descent(), p);
113     return bm;
114 }
115 public void run() {
116     start = System.currentTimeMillis();
117     PorterDuffXfermode mode = new PorterDuffXfermode(PorterDuff.Mode.DST_OVER);
118
119     int width;
120     int height;
121     Paint p = new Paint();
122     Bitmap ovl = null;
123
124     while (mRun) {
125
126         Rect destRect = null;
127         Canvas c = null;
128
129         if (surfaceDone) {
130             try {
131                 if (bmp == null) {
132                     bmp = Bitmap.createBitmap(IMG_WIDTH, IMG_HEIGHT,
133                                         Bitmap.Config.ARGB_8888);
134                 }
135                 int ret = mIn.readMjpegFrame(bmp);
136
137 //             if (ret == -1) {
138 //                 ((MjpegActivity) saved_context).setImageError();
139 //             }
140
141             destRect = destRect(bmp.getWidth(), bmp.getHeight());
142
143             c = mSurfaceHolder.lockCanvas();
144             synchronized (mSurfaceHolder) {
145
146                 c.drawBitmap(bmp, null, destRect, p);
147
148                 if (showFps) {
149                     p.setXfermode(mode);
150                     if (ovl != null) {
151
152                         // false indentation to fix forum layout
153                         height = ((ovlPos & 1) == 1) ? destRect.top :
154                         destRect.bottom - ovl.getHeight();
155                         width = ((ovlPos & 8) == 8) ? destRect.left :
156                         destRect.right - ovl.getWidth();
157
158                         c.drawBitmap(ovl, width, height, null);
159
160                     p.setXfermode(null);
161                     frameCounter++;
162                     if ((System.currentTimeMillis() - start) >= 1000) {
163                         fps = String.valueOf(frameCounter) + "fps";
164                     }
165
166                 }
167             }
168         }
169     }
170 }

```

```

162         frameCounter = 0;
163         start = System.currentTimeMillis();
164         if (ovl != null) ovl.recycle();
165
166         ovl = makeFpsOverlay(overlayPaint);
167     }
168 }
169
170     }
171 }
172 } catch (IOException e) {
173 } finally {
174     if (c != null) mSurfaceHolder.unlockCanvasAndPost(c);
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182
183 private void init(Context context) {
184
185     //SurfaceHolder holder = getHolder();
186     holder = getHolder();
187     saved_context = context;
188     holder.addCallback(this);
189     thread = new MjpegViewThread(holder, context);
190     setFocusable(true);
191     overlayPaint = new Paint();
192     overlayPaint.setTextAlign(Paint.Align.LEFT);
193     overlayPaint.setTextSize(12);
194     overlayPaint.setTypeface(Typeface.DEFAULT);
195     overlayTextColor = Color.WHITE;
196     overlayBackgroundColor = Color.BLACK;
197     ovlPos = MjpegView.POSITION_LOWER_RIGHT;
198     displayMode = MjpegView.SIZE_STANDARD;
199     dispWidth = getWidth();
200     dispHeight = getHeight();
201 }
202
203 public void startPlayback() {
204     if (mIn != null) {
205         mRun = true;
206         if (thread == null) {
207             thread = new MjpegViewThread(holder, saved_context);
208         }
209         thread.start();
210     }
211 }
212
213 public void resumePlayback() {
214     if (suspending) {
215         if (mIn != null) {
216             mRun = true;
217             SurfaceHolder holder = getHolder();
218             holder.addCallback(this);
219             thread = new MjpegViewThread(holder, saved_context);
220             thread.start();
221             suspending = false;
222         }
223     }
224 }
225
226 public void stopPlayback() {
227     if (mRun) {
228         suspending = true;
229     }
230     mRun = false;
231     if (thread != null) {
232         boolean retry = true;
233         while (retry) {
234             try {
235                 thread.join();
236                 retry = false;
237             } catch (InterruptedException e) {
238             }
239         }
240         thread = null;
241     }
242     if (mIn != null) {
243         try {
244             mIn.close();

```

```

245         } catch (IOException e) {
246     }
247     mIn = null;
248 }
249 }
250 }
251 public void freeCameraMemory() {
252     if (mIn != null) {
253         mIn.freeCameraMemory();
254     }
255 }
256 }
257 public MjpegView(Context context, AttributeSet attrs) {
258     super(context, attrs);
259     init(context);
260 }
261 }
262 public void surfaceChanged(SurfaceHolder holder, int f, int w, int h) {
263     if (thread != null) {
264         thread.setSurfaceSize(w, h);
265     }
266 }
267 }
268 public void surfaceDestroyed(SurfaceHolder holder) {
269     surfaceDone = false;
270     stopPlayback();
271 }
272 }
273 public MjpegView(Context context) {
274     super(context);
275     init(context);
276 }
277 }
278 public void surfaceCreated(SurfaceHolder holder) {
279     surfaceDone = true;
280 }
281 }
282 public void showFps(boolean b) {
283     showFps = b;
284 }
285 }
286 public void setSource(MjpegInputStream source) {
287     mIn = source;
288     if (!suspending) {
289         startPlayback();
290     } else {
291         resumePlayback();
292     }
293 }
294 }
295 public void setOverlayPaint(Paint p) {
296     overlayPaint = p;
297 }
298 }
299 public void setOverlayTextColor(int c) {
300     overlayTextColor = c;
301 }
302 }
303 public void setOverlayBackgroundColor(int c) {
304     overlayBackgroundColor = c;
305 }
306 }
307 public void setOverlayPosition(int p) {
308     ovlPos = p;
309 }
310 }
311 public void setDisplayMode(int s) {
312     displayMode = s;
313 }
314 }
315 public void setResolution(int w, int h) {
316     IMG_WIDTH = w;
317     IMG_HEIGHT = h;
318 }
319 }
320 public boolean isStreaming() {
321     return mRun;
322 }
323 }
324 }

```

JAVA–Code/Speechrecognition.java

```
1 package com.camera.simplempeg;
2
3 import static edu.cmu.pocketsphinx.SpeechRecognizerSetup.defaultSetup;
4
5 import android.content.Context;
6 import android.os.AsyncTask;
7 import android.util.Log;
8 import edu.cmu.pocketsphinx.Assets;
9 import edu.cmu.pocketsphinx.Hypothesis;
10 import edu.cmu.pocketsphinx.SpeechRecognizer;
11
12 import java.io.File;
13 import java.io.IOException;
14
15 public class SpeechRecognition implements edu.cmu.pocketsphinx.RecognitionListener {
16     // driving
17     public static final String ORDER_SEARCH = "order";
18     public static final String TURN_LEFT = "left";
19     public static final String TURN_RIGHT = "right";
20     public static final String STOP = "stand";
21     public static final String DRIVE = "go";
22     public static final String BACK = "back";
23     public static final String STRAIGHT = "line up";
24     // other
25     public static final String HORN = "beep";
26     public static final String ON = "on";
27     public static final String OFF = "off";
28     public static final String AUTO = "auto";
29     //
30     private SpeechRecognizer recognizer;
31     private MainActivity mainActivity;
32     private boolean driving;
33
34     // the boolean 'driving' indicates, wheater the speech recognition object should be
35     // set up for "driving" or "other"
36     public SpeechRecognition(MainActivity mainActivity, final Context context, boolean
37     driving) {
38         this.mainActivity = mainActivity;
39         this.driving = driving;
40         // Recognizer initialization is a time-consuming and it involves IO,
41         // so we execute it in async task
42         new AsyncTask<Void, Void, Exception>() {
43             @Override
44             protected Exception doInBackground(Void... params) {
45                 try {
46                     Assets assets = new Assets(context);
47                     File assetDir = assets.syncAssets();
48                     setupRecognizer(assetDir);
49                 } catch (IOException e) {
50                     return e;
51                 }
52                 return null;
53             }
54             @Override
55             protected void onPostExecute(Exception result) {
56                 if (result != null) {
57                     Log.e(MainActivity.TAG, "Failed to init recognizer " + recognizer);
58                 } else {
59                     Log.e(MainActivity.TAG, "Successfully init recognizer " + recognizer);
60                     recognizer.startListening(ORDER_SEARCH);
61                 }
62             }.execute();
63     }
64
65     /** In partial result we get quick updates about current hypothesis. In
66     * keyword spotting mode we can react here, in other modes we need to wait
67     * for final result in onResult */
68     @Override
69     public void onPartialResult(Hypothesis hypothesis) {
70         if (hypothesis == null)
71             return;
72     }
73
74     /** This callback is called when we stop the recognizer */
75     @Override
76     public void onResult(Hypothesis hypothesis) {
77         ((TextView) findViewById(R.id.caption_text)).setText("");
78         if (hypothesis != null) {
```

```

79         String text = hypothesis.getHypstr();
80         if (text.equals(DRIVE)) doSomething(DRIVE);
81         else if (text.equals(TURN_LEFT)) doSomething(TURN_LEFT);
82         else if (text.equals(TURN_RIGHT)) doSomething(TURN_RIGHT);
83         else if (text.equals(STOP)) doSomething(STOP);
84         else if (text.equals(BACK)) doSomething(BACK);
85         else if (text.equals(STRAIGHT)) doSomething(STRAIGHT);
86         else if (text.equals(HORN)) doSomething(HORN);
87         else if (text.equals(ON)) doSomething(ON);
88         else if (text.equals(OFF)) doSomething(OFF);
89         else if (text.equals(AUTO)) doSomething(AUTO);
90     }
91 }
92
93 @Override
94 public void onBeginningOfSpeech() {
95 }
96
97 /** We stop recognizer here to get a final result */
98 @Override
99 public void onEndOfSpeech() {
100 }
101
102 // int as return type, connect with list!
103 private void doSomething(String searchName) {
104     recognizer.stop();
105     Log.e(MainActivity.TAG, "You said: " + searchName);
106     mainActivity.sendOverSocket(searchName);
107     mainActivity.onActivityResult(searchName);
108 }
109
110 private void setupRecognizer(File assetsDir) throws IOException {
111     // The recognizer can be configured to perform multiple searches
112     // of different kind and switch between them
113     recognizer = defaultSetup()
114         .setAcousticModel(new File(assetsDir, "en-us-ptm"))
115         .setDictionary(new File(assetsDir, "cmudict-en-us.dict"))
116         // To disable logging of raw audio comment out this call (takes a lot
117         // of space on the device)
118         .setRawLogDir(assetsDir)
119         // Threshold to tune for keyphrase to balance between false alarms and
120         // misses
121         .setKeywordThreshold(1e-45f)
122         // Use context-independent phonetic search, context-dependent is too
123         // slow for mobile
124         .setBoolean("-allphone_ci", true)
125     .getRecognizer();
126     recognizer.addListener(this);
127     // Create grammar-based search for selection
128     if(driving){
129         File orderGrammar = new File(assetsDir, "menu.gram");
130         recognizer.addGrammarSearch(ORDER_SEARCH, orderGrammar);
131     }else{
132         File orderGrammar = new File(assetsDir, "menu2.gram");
133         recognizer.addGrammarSearch(ORDER_SEARCH, orderGrammar);
134     }
135
136     @Override
137     public void onError(Exception error) {
138         Log.e(MainActivity.TAG, "error: " + error.getMessage());
139     }
140
141     @Override
142     public void onTimeout() {
143
144         public SpeechRecognizer getRecognizer() {
145             return recognizer;
146         }
147     }
148 }

```

JAVA–Code/Constants.java

```
1 package com.camera.simplejpeg;
2
3 abstract public class Constants {
4     /** Ports */
5     public static final int PORT_MOTION = 8081;
6     public static final int PORT_SOCKET = 1337;
7     /** Constants */
8     public static final int CLOSESOCKET = 77; // Client closed Socket-Connection
9     // Motor controls (the number indicates the gear)
10    // Motor 1
11    public static final int GO_1 = 116;
12    public static final int GO_2 = 117;
13    public static final int GO_3 = 118;
14    public static final int GO_4 = 119;
15    public static final int GO_5 = 120;
16    public static final int BACK_1 = 114;
17    public static final int BACK_2 = 113;
18    public static final int BACK_3 = 112;
19    public static final int BACK_4 = 111;
20    public static final int BACK_5 = 110;
21    public static final int STOP = 115; // Stops GO + BACK
22    // Motor 2
23    public static final int LEFT = 11;
24    public static final int RIGHT = 12;
25    public static final int STRAIGHT = 10; // Lines up LEFT + RIGHT
26    // Lights
27    public static final int LIGHTS_OFF = 18;
28    public static final int LIGHTS_ON = 19;
29    public static final int LIGHTS_AUTO_OFF = 20; // automatic lights
30    public static final int LIGHTS_AUTO_ON = 21;
31    // Sound
32    public static final int HORN = 25;
33    public static final int MUSIC_ON = 30;
34    public static final int SOUND_OFF = 31; // Sound off disables horn and music
35 }
```

Abbildungsverzeichnis

1	Fahrzeug im Originalzustand.	2
2	Funkfernbedienung.	3
3	Geöffneter Unterbau mit freigelegter Steuerplatine.	4
4	Raspberry 2 Model B.	5
5	WLAN-Adapter.	5
6	Steckbrett.	5
7	T-Kupplung und Flachbandkabel.	6
8	Ultraschallsensor (Typ HC-SR04).	6
9	Fotowiderstand.	7
10	Webcam.	7
11	Mobiler Lautsprecher.	8
12	Externer Zusatzakku.	8
13	Grobe Zuordnung der Funktionsgruppen auf der Steuerplatine.	9
14	Unterseite der Steuerplatine mit den neu angebrachten Leitungen.	11
15	Leuchtdioden auf der rechten Fahrzeugseite.	13
16	Demonstration der Scheinwerfer und Blinker.	13
17	Fotowiderstand, ins Fahrzeugdach eingeklebt.	14
18	Schematische Darstellung der Abdeckung durch die Ultraschall-sensoren.	15
19	Kabelverläufe und Knotenpunkte.	17
20	Steckbrett mit allen aufgesteckten Leitungen.	18
21	Webcam und Lautsprecher im Führerhaus.	19
22	Rückansicht des Fahrzeugs mit montiertem Raspberry Pi und Zusatzakku.	19
23	Dialog zur Eingabe der IP-Adresse	30
24	Network Error Dialog	30
25	Screenshot der Anwendung im Querformat mit Beschreibung der Elemente	32
26	Screenshot der Anwendung im Querformat im Debug-Modus	32

Tabellenverzeichnis

1	Zuordnung der einzelnen GPIO-Pins	22
2	Auflistung der Zahlenwerte die über die Socket-Verbindung gesendet werden	27
3	Sprachbefehle	29