HAW Hamburg
Fakultät DMI

Dokumentation
Projekt

Free Ride

---

durchgeführt im:	Sommersemester 2018	
Fach:	IT-S / Mobile Systeme	
bei:	Prof. Dr. Edeler Prof. Dr. Plaß	
Teilnehmer:	Max Hammer	2300038
	Martin Nick	2293440
	Waldemar Goßmann	2285410
	Karl-Peter Blynow	2156277

Inhaltsverzeichnis

1.Konzept

1.1 Projektziel

Bau eines ferngesteuerten Autos mit verschiedenen Modi. Im Discovery Mode kann man mit Hilfe einer auf dem Wagen montierten Kamera, auf Erkundungstour gehen und die Welt aus einem anderen Blickwinkel sehen. Gesteuert wird durch eine App auf dem Handy. Die Kamera kann im Discovery Mode frei durch die Bewegungssteuerung des Handys bewegt werden. Eine in 2-Achsen bewegliche Kamera gibt einem die Möglichkeit, seine gesamte Umgebung zu erfassen.
Im Track Mode wird die Kamera in eine frontalen Ansicht fixiert. Ein besonderes Erlebnis ist die realitätsnahe Lenkung, bei dem das Handy zum Lenkrad wird. Durch Neigung des Handys nach rechts oder links, lenkt das Auto entsprechend. Die Geschwindigkeit wird weiterhin über Tasteneingaben kontrolliert. Durch die Kompaktheit des kleinen Flitzers kommt man in die entlegensten Ecken. Der Track Mode weckt in Jung und Alt das Rennfahrerherz. Mit Hilfe von 3 Toren steckt man zunächst eine Strecke ab. Alle Möglichkeiten stehen einem offen und man kann sich überall seine Traumstrecke zusammenstellen und dann gegeneinander um die Bestzeit kämpfen. In einer bestimmten Reihenfolge müssen nun die Tore durchfahren werden. Die Zeitnahme startet beim ersten Tor und die Runde endet beim Dritten. Beim Durchfahren des mittleren Tores wird eines Sektorzeite genommen, angezeigt und gespeichert. Auch die Rundenzeiten werden gespeichert und Bestenlisten erstellt. Werde der schnellste und schlage deine Mitstreiter!

1.2 Anforderungsanalyse

1. Steuerung des Autos

Durch Ansteuerung der vier Motoren kann das Auto vorwärts fahren und nach links und rechts Lenken. Die Bedienung erfolgt mittels einer Verbindung zwischen Back- und Frontend. Die Steuerdaten des Handys werden auf den Raspberry Pi übertragen und dort auf die Motoren umgesetzt.

2. Videoübertragung

Die Kamera liefert dem Raspberry Pi ein Livebild. Dieses wird per WLAN über einen Broker an die Handyapp gestreamt.

3. Kamerasteuerung

Bewegung der Kamera mit Hilfe des gyroskopischen Sensors vom Handy.

4. Anfertigung einer Kamerahalterung

Bewegungsmöglichkeit der Kamera auf der horizontalen um ca. 360° und auf der vertikalen Achse um ca. 90°

5. Fertigung von 3 Toren

Die Tore müssen mit verschiedenen LEDs ausgestattet sein, welche auf den seitlich in die Mitte des Tores strahlen und autark mit Strom versorgt werden.

6. Implementierung des Farbsensors

Ein CMOS-Farbsensor sendet RGB-Werte digital an den Raspberry Pi.
Abhängig von dem Wert wird das Start, Ziel oder der Checkpoint erkannt

7. Zeitnahme und Übertragung

Die Farbsensoren werden durch das emittierte Licht der LED-Tore getriggert und setzen so die Sektor- und Rundenzeiten. Diese werden an die App übermittelt und dort verarbeitet und gespeichert.

8. Verschiedene Modi

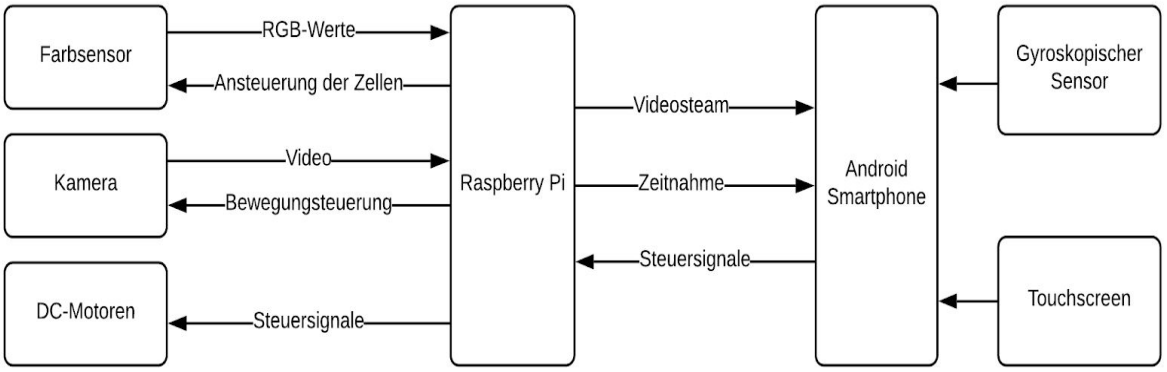
Im Discovery Mode ist die Kamera frei steuerbar und das Auto kann per Tasten bewegt werden. Im Trackmode ist die Kamera fest und der gyroskopische Sensor übernimmt die Steuerung des Autos.

1.3 Technische Rahmenbedingungen

Auf der Basis der bereits vorhandenen Fahrzeug-Grundlage wird ein Raspberry Pi montiert. Dieser verfügt über eine WLAN Verbindung und hat Linux als Betriebssystem. Die Programme die auf dem Raspberry laufen werden in Python geschrieben.
Auf dem Fahrzeug werden außerdem eine Kamera zur Bildübertragung, 2 Motoren zur Bewegung der Kamera, 4 Motoren für Lenkung und Antrieb, sowie ein Farbsensor montiert.
Die Programmierung der App wird mithilfe von Android-Studio in Kotlin auf Windows Computern stattfinden. Die App wird auf Android-Smartphones mit WLAN laufen.
Die Tore bestehen aus Farb-LEDs sowie Knopfzellen zur Stromversorgung.

1.4 Technisches Konzept

Eingaben und Bewegungsbefehle erfolgen mithilfe des im Smartphone vorhandenen Gyroskops und des Touchscreens. Die Steuersignale werden von der selbstgeschriebenen App über das lokale WLAN an einen MQTT Broker gesendet und von dort an den Raspberry Pi übertragen. Dieser sendet auf die gleiche Weise die Aufnahmen der Kamera über einen mjpeg-Stream und sonstige benötigte Informationen. Die Verarbeitung der Daten erfolgt großteilig in der App.
Die 3D-gedruckte Kamera-Aufhängung ist steuerbar von 2 Motoren.
Ansteuern der Motoren für Räder und Kamera folgt auf das, über WLAN an den Raspberry Pi übermittelte, Signal.
Der Farbsensor reagiert auf die LED-Beleuchtung in den Toren, durch Messen des Farbtons als RGB-Wert.
Das Leiten der Beleuchtung von oben auf den Sensor wird durch die Form der Tore gegeben.
Als Reaktion auf die Beleuchtung durch die LEDs wird die Signal ausgegeben, welches in der App entsprechende Start- und Zielzeiten stoppt.



1.5 Bedienkonzept

Das Auto wird durch eine App auf dem Smartphone gesteuert. Ein Button wird die Möglichkeit bieten zwischen Discovery Mode und Track Mode zu wechseln. Im Discovery Mode lässt sich das Auto mit Buttons, und die sich auf dem Auto befindende Kamera durch Neigen und Drehen des Smartphones steuern (Gyroscope). Mit einem zusätzlichen Button kann außerdem ein Screenshot gemacht werden. Im Track Mode, in dem das Auto durch die Rennstrecke gesteuert werden soll, lässt es sich durch drehen des Smartphones lenken und die Kamera wird in einer frontalen Ansicht fixiert. Mit einem Button/Regler wird dabei beschleunigt. In einer Ecke des Bildschirms wird hierbei die aktuelle Zeit zu sehen sein.

Nach jedem erfolgreich durchfahrenen Tor wird auf dem Display angezeigt, wieviel schneller oder langsamer man war, im Vergleich zum letzten Fahrer/der aktuellen Bestzeit und sollte man einen neuen Rekord aufstellen, darf man sich am Ende in die Bestenliste eintragen, die genau wie die Screenshot-Sammlung aus dem Hauptmenü aufgerufen werden kann.

1.6 Zeitplan

Konzepterstellung(7-8h pro Person):

Erstellung des konkreten Konzeptes und Planung des Prototypen.

24.04. Konzeptabgabe

Erstellung Prototyp (20-25h pro Person):

Grundlegende Funktionen umsetzen und technisch funktionierenden Protoypen erstellen.

29.05. Vorstellung Prototyp

Umsetzung/Feinschliff (15-20h pro Person):

Design einfügen. Sämtliche geplante Funktionen umsetzen. Eventuell zusätzliche Funktionen hinzufügen.

19.06. Generalprobe

Fehlerbeseitigung + Verschönerung (15h pro Person):

Einwandfreie Funktion sicherstellen. Feinschliff. Eventuell zusätzliche Funktionen hinzufügen.

12.07. Präsentation

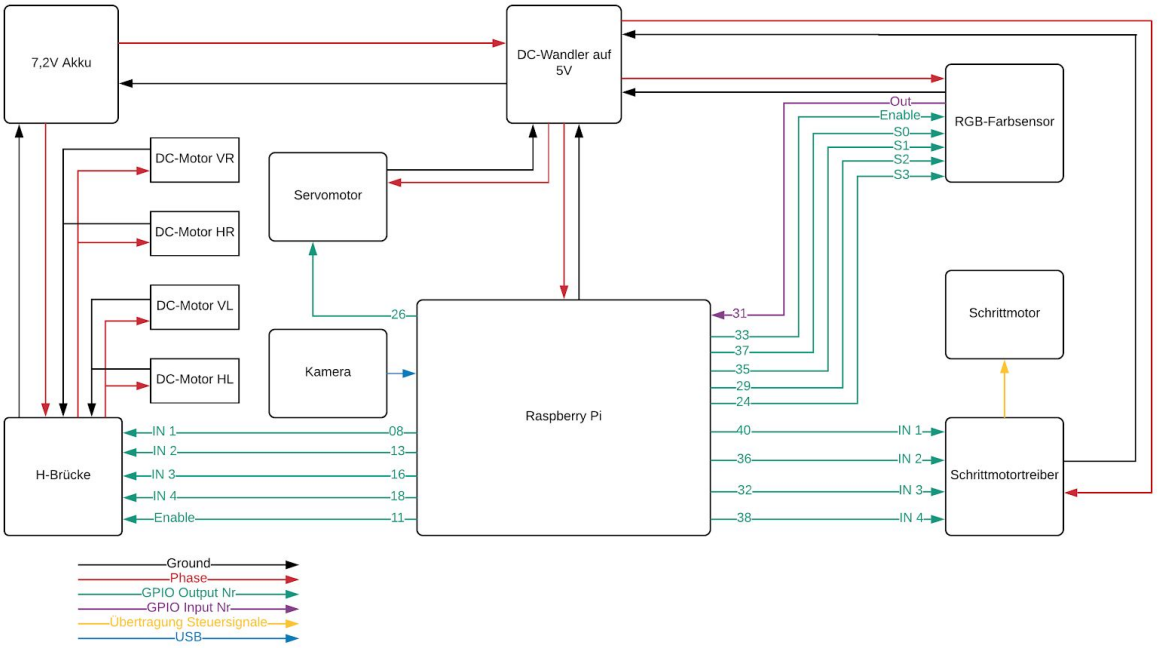
Dokumentation 5h pro Person:

Erstellung und Zusammenfassung der Dokumentation aus den bereits protokollierten Arbeitsschritten.

20.07. Abgabe Dokumentation

2.Technische Umsetzung

2.1 Verschaltung und Aufbau



Die konkrete Verschaltung der einzelnen Bauteile des Modellautos ist dem Blockschaltbild (siehe oben) zu übernehmen und wird im Folgenden grundsätzlich erläutert. Zunächst muss erwähnt werden, dass die Leiter nicht direkt an die Raspberry Pi GPIO-Pins angelegt wurden. Um eine möglichst “cleanes” Erscheinungsbild des RC-Cars zu erreichen, wurde eine Platine angefertigt. Hier gehen zunächst alle Raspberry Pi Pins über einen Flachbandstecker ein und werden von dort über feste Lötstellen und angebrachte Stecker weitergeleitet. Auch der parallele Anschluss der Versorgungsspannung für den Raspberry und die verschiedenen Anbauteile ist hier untergebracht. Auch hier kommen Steckverbindungen mit 2 Pins (5V/Ground) zum Einsatz um größtmögliche Flexibilität zu schaffen. Die eigentliche Stromversorgung übernimmt ein 7,2 Volt Akkumulator. Jedoch läuft lediglich die H-Brücke, welche für die Ansteuerung der einzelnen DC-Motoren zuständig ist, mit 7,2 V und ist dementsprechend direkt an den Akku angeschlossen. Alle anderen Teile wie der Raspberry, der Schritt- und der Servomotor, sowie der Farbsensor werden mit 5V versorgt, welche durch einen Spannungswandler bereitgestellt werden. Dieser wandelt die 7,2 V Spannung des Akkus auf 5V.
Alle Steuersignale zu und von dem Sensor und zu der H-Brücke und dem Servomotor sind über Stecker realisiert. Die Ansteuerung des Schrittmotortreibers sind fest auf der Platine verlötet. Ähnlich der H-Brücke setzt dieser die eingehenden Steuersignale auf seine 4 Motoren um. Konsequenterweise finden sich für die Verschaltung hier 2 Kabelbäume um ein aufgeräumtes und sauberes Erscheinungsbild der Elektronik zu erzielen.

Als Letztes muss noch auf die Platzierung des Farbsensors eingegangen werden. Dieser findet sich vorne rechts, zur Seite ausgerichtet und durch eine Verschalung von Tageslicht geschützt um eine korrekte Zeitnahme, ausgelöst durch die LED's an den einzelnen Toren, durchführen zu können.

2.2 Kameramount

Das Ziel war die Erstellung einer Kamerahalterung, welche in der Horizontalen als auch in der Vertikalen bewegbar ist. Für die Konstruktion wurde das Programm Tinkercad benutzt und das erstellte Modell (siehe STL Datei) später mit einem 3D-Drucker ausgedruckt. Im Gesamten besteht es aus 5 Teilen. Die Bodenplatte bildet den Auflagepunkt mit dem Modellauto. Eine Einkerbung für die Ausgangswelle des verwendeten Schrittmotors sorgt für die Möglichkeit die darüber befindliche Motorenaufnahme in der Horizontalen zu bewegen. Aufgenommen wird diese mit einem pressfit-eingepassten Kugellager. Durch die Verwendung eines Kugellagers wird eine möglichst reibungsarme und somit flüssige Rotation der Kamera gewährleistet. Für die horizontale Bewegung wird ein Servomotor verwendet. Dieser ist auch auf der Kameraplatte angeschraubt und sorgt über eine Verbindung mit der oberen Kamerabrücke für den vertikalen Tilt der Kamera. Auch hier kommen 2 Kugellager zum Einsatz. Die Kamerabrücke wird über 2 Seitenarme mit der Motoraufnahme verschraubt. Erst so ist es möglich diese kugelgelagert einzufassen. Die eigentliche Kameraaufnahme besteht aus einem Kugelkopf, in welchen sie eingesteckt wird.
Insgesamt ergibt sich eine durch die benötigten Kabel für die Verbindung des Schrittmotors mit dem Motortreiber, die Verbindung des Servomotors und der Kamera mit dem Raspberry Pi begrenzte horizontale Bewegung von ungefähr 270°. In der Vertikalen lässt sich die Kamera um ungefähr 75° bewegen.

2.3 Programmierung Raspberry Pi

Zur Ansteuerung der verschieden Bauteile, zum Auslesen des Farbsensors und zur Übermittlung der verschiedenen Daten und Steuersignale zwischen der App und dem Raspberry Pi müssen verschiedene Programme ausgeführt werden, welche dann parallel laufen. Für die Kommunikation zwischen der APP und dem Raspberry Pi wird ein MQTT Broker genutzt. Dieser wird beidseitig über das lokale WLAN angesprochen.

2.3.1 Kamera

“cam.py” wird verwendet um die Kamera zu steuern. Der Schrittmotor steuert die Horizontale und die Richtung wird bestimmt durch die Reihenfolge der Ansteuerungen der Motoren im Schrittmotor(Z 29-39 & 50-68).Der Servomotor steuert die Vertikale und die Richtung wird bestimmt durch die Länge des gegebenen Impulses(Z 135-159).
Über den MQTT-Server wird ein Steuersignal empfangen und dann ausgeführt. Das Signal enthält einen Wert für die Horizontale und die Vertikale und ist Teil des Steuersignale welches auch für die Ansteuerung der Räder verwendet wird.
Die Kamera muss anfangs mittig ausgerichtet werden. Dann wird die Differenz aus dem derzeitigen Wert und dem Wert des Steuersignals genommen und die Kamera um diese Anzahl an Schritten gedreht(Z 87-159).
Beim Beenden des Programms oder beim Schalten in den Racing Modus in der App fährt der Kamerakopf wieder in Startposition(Z 161-260).
Um die Startposition einzustellen, können die “config” Programme verwendet werden.

Das Video Bild wird mit Hilfe des Unterprogramms “Motion” als MJPEG-Stream über das lokale Netzwerk an die App übertragen (siehe 2.4.2).

2.3.2 Farbsensor

“sen.py” steuert den Farbsensor. Generell besteht dieser aus verschiedenen Photozellen, welche jeweils mit Rot, Blau, Grün und Allpassfiltern belegt sind. Zur Auslesung muss jeweils die gefordertete Zelle angesprochen werden (CMOS). Um die zu verarbeitenden Daten zu reduzieren, sendet der Sensor nur einen Wert, wenn das geforderte Tor durchfahren wird(Z 96-117).
Es wird dafür der blaue und rote Sensor ausgelesen(Z 27-50 / 77 / 87 / 89) . Dann wird die Differenz genommen und nur bei einem großen Unterschied zwischen den zwei Sensoren wird angenommen, dass ein Tor passiert wurde(Z 75 / 96).
Nach einem Sendevorgang wird das Programm pausiert um zu verhindern, dass das Signal mehrmals gesendet wird.
Die Sensoren geben abhängig von der einstrahlenden Energie eine Frequenz an 1 und 0 heraus. Zum Auslesen dieser Frequenz wird die Anzahl der Flanken des Signals über die Zeit gemessen(Z 53-61).

2.3.3 Motorensteuerung

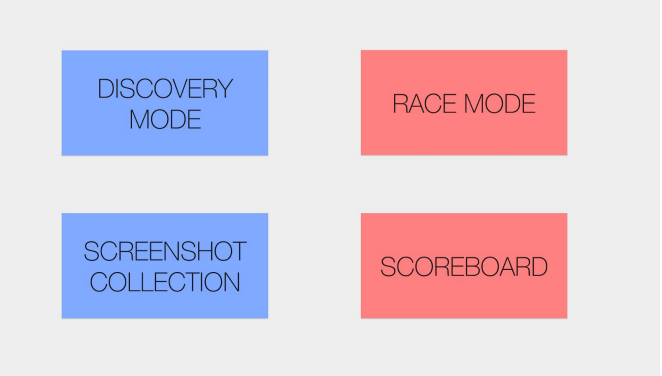
“eng.py” steuert die Motoren an. Über mqtt wird ein Steuersignal empfangen, das die Geschwindigkeit und die Richtung enthält(Z 14 /20).
Die Gleichstrommotoren werden dann über Pulscodemodulation gesteuert(Z 43-156) . Aus der Geschwindigkeit und dem Takt wird die Zeit berechnet, in der die Motoren angesteuert

werden müssen, und die Zeit in der sie nicht angesteuert werden müssen. Aus der Geschwindigkeit und der Richtung wird berechnet, wie lange die gewünschte Seite im Verhältnis zur Geschwindigkeit angesteuert werden muss um in die gewünschte Richtung zu fahren(Z 21-30).

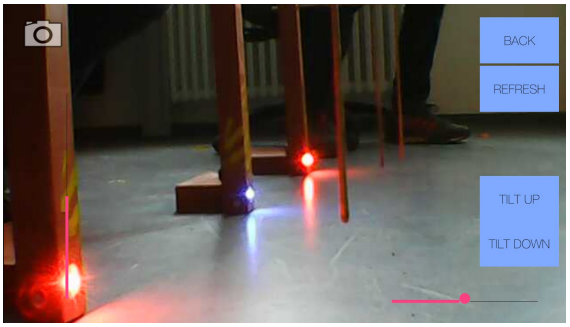
2.4 Android Anwendung

2.4.1 FreeRide App

Beim Starten der App öffnet sich das Hauptmenü in dem einem mit vier Buttons die verschiedenen Funktionen der Anwendung zur Verfügung stehen.



Mit dem ersten Button lässt sich der Discovery Mode starten (MainActivity.kt, Zeile: 241-269). In diesem Modus lässt sich die Kamera durch seitliches Kippen des Handys frei nach links und rechts schwenken (MainActivity.kt, Zeile: 77-104) und mit den Tilt-Buttons nach oben und unten bewegen (MainActivity.kt, Zeile: 400-418).
Das Auto steuert man in diesem Modus mit zwei Schiebereglern. Der linke Schieberegler regelt die Geschwindigkeit des Autos (MainActivity.kt, Zeile: 540-557) und der in der rechten unteren Ecke die Lenkrichtung (MainActivity.kt, Zeile: 558-575). Oben rechts befinden sich die Buttons um zurück ins Hauptmenü zu gelangen (MainActivity.kt, Zeile: 366-398) und um den Livestream neu zu laden (MainActivity.kt, Zeile: 473-478). Zu guter Letzt befindet sich oben links noch das Symbol um einen Screenshot aufzunehmen (MainActivity.kt, Zeile: 607-658). Alle Buttons wurden hierbei im hellblauen Stil gehalten, der den Discovery Mode verkörpern soll.



Im gleichen Stil befindet sich im Hauptmenü gleich unter dem Discovery Mode die dazugehörige Screenshot-Sammlung (MainActivity.kt, Zeile: 297-334). In der Screenshot-Sammlung lässt sich eine Galerie der gespeicherten Bilder ansehen. Mit zwei Buttons am unteren Rand kann man das nächste oder vorherige Bild anzeigen (MainActivity.kt, Zeile: 420-446) und auch der Button um ins Hauptmenü zurückzukehren ist wieder vorhanden.

Als Alternative zum Discovery Mode findet man im Hauptmenü auf der rechten Seite in rot gehalten den Race Mode (MainActivity.kt, Zeile: 271-295). Der Race Mode unterscheidet sich visuell lediglich dadurch, dass Screenshot- und Kamerabewegungs-Buttons fehlen und die Lenkung des Autos nun über das seitliche Kippen des Geräts erfolgt (MainActivity.kt, Zeile: 77-95). Die Kamera ist in diesem Modus fest ausgerichtet.

Der letzte Button des Hauptmenüs bringt einen zur Bestenliste (MainActivity.kt, Zeile: 336-364). Sollte man im Race Mode einen neuen Rekord aufstellen und es unter die besten fünf Rennfahrer schaffen wird man hier unter dem eingegebenen Namen gespeichert (MainActivity.kt, Zeile: 747-823). Außer dem Back-Button steht einem hier die Möglichkeit zur Verfügung alle Rekorde zu löschen um eine neue Rennstrecke aufzubauen (MainActivity.kt, Zeile: 455-471).

2.4.2 FreeRide Quellcode

Die für das Projekt relevanten Quellcode-Dateien sind *MainActivity.kt* für den Ablauf der Funktionen (Mosy_FreeRide/app/src/main/java/com/eldinox/freerideprototyp3/MainActivity.kt) und *activity_main.xml* für das Layout der App (…/main/res/layout/activity_main.xml).

MainActivity.kt
Importe: Zeile 10-36
Initialisierung der Variablen: Zeile 38-61
Funktion onSensorChanged(): Zeile 77-110
Verbindung zum MQTT Broker: Zeile 112-141
Funktion publish(): Zeile 143-147
Initialisierung der Grafikelemente: Zeile 174-199
Initialaufruf des Hauptmenüs: Zeile 208-237
ClickListener der Buttons: Zeile 240-479
Konfiguration der WebView: Zeile 481-493
Timer und Handler: Zeile 495-537
ChangeListener der Schieberegler: Zeile 539-576
Verbindungsaufruf zum Broker: Zeile 578-604
Funktion captureScreenshot(): Zeile 607-658
Empfangen von Nachrichten des Brokers: Zeile 670-705
Funktion readHighscores(): Zeile 707-732
Funktion writeHighscores(): Zeile 733-745
Funktion compairTime(): Zeile 747-823
Funktion setName(): Zeile 825-856

Zur Übertragung haben wir den MQTT Broker von HiveMQ verwendet. Um zu verstehen wie die Verbindung von einem Android Gerät mit dem Broker funktioniert, haben wir uns am Tutorial (https://www.hivemq.com/blog/mqtt-client-library-enyclopedia-paho-android-service) von Sandro

Kock und an einem Youtube-Tutorial (https://www.youtube.com/watch?v=BAkGm02WBc0&t=1000s) von Toeshiro Novisu orientiert. Der Python-Code basiert auf der Vorlage aus den Folien von Prof. Dr. Plaß und zum Anzeigen des MJPEG-Streams benutzten wir Motion (https://willy-tech.de/motion-fur-den-raspberry-pi/) und die Übertragung über das lokale WLAN an die Webview der Smartphone-App.

3. Fazit

Zum Abschluss des Projekts muss man konsultieren, dass wir die uns zu Beginn gesetzten Ziele erreicht haben. Der Weg dorthin war jedoch teilweise sehr langwierig und stellte uns vor einige unerwartete Probleme, die es zu lösen galt. Dies erforderte letztlich auch geringfügige Anpassung des ursprünglichen Konzepts. Teilweise lag dies an nicht zu umgehenden Umständen (siehe Farbsensor) und bei der Steuerung zum Beispiel an Anforderungen an die Bedienbarkeit. Das Hauptziel, ein ferngesteuertes Auto zu bauen, mit dem man eine menge Spaß haben kann, wurde jedoch mehr als erreicht und gerade diese vielen kleinen, immer wieder auftauchenden Schwierigkeiten und meist unerwarteten Probleme, machten es zu einem extrem lehrreichen Projekt.

Anhang

im GIT Hub befinden sich:

Quellcode Pythonprogramme:
https://github.com/Eldinox/Mosy_FreeRide_python
Quellcode Smartphoneapp:
https://github.com/Eldinox/Mosy_FreeRide_android
Blockschaltbild RC-Modellauto:
https://github.com/Eldinox/Mosy_FreeRide_python/blob/master/Blockschaltbild%20ITS.png
STL 3D-Modell Kameramount:
https://github.com/Eldinox/Mosy_FreeRide_python/blob/master/Kamerakopf%20und%20Arme%20korrekt.stl
Poster für Ausstellung:
https://github.com/Eldinox/Mosy_FreeRide_python/blob/master/Mosy_FreeRide_Plakat.pdf