Pi-lot – Technische Dokumentation

Von:

Bager Bingöl (2239338)

Andre Till (2239431)

Philipp Dreyer (2249310)

Frederik Gutbrod (2248407)

Betreuer:

Prof. Dr. Andreas Plaß

Prof. Dr. Torsten Edeler

19. Juli 18

Mobile Systeme / IT-Systeme

Studiengang Media Systems (B.Sc.)

———————————————————————————————

Hochschule für angewandte Wissenschaften Hamburg /
Hamburg University of Applied Sciences

Department Medientechnik
Fakultät Design, Medien und Information

1 Vorwort

Das Projekt Pi-lot entstand im Rahmen der Veranstaltungen Mobile System und IT-Systeme an der Hochschule für angewandte Wissenschaften Hamburg.

Ziel des Kurses war es ein mobile VirtualReality-Anwendung zu entwickeln. Umgesetzt werden sollte dies, mit Hilfe eines Raspberry Pi's und einer mobilen Autoplattform.

Unser Ziel war es ein ferngesteuertes Auto zu entwickeln, welches die geforderten Bedingungen erfüllt und zusätzlich ein angenehmeres Fahrerlebnis bietet.

Wir haben uns aufgrund des schlechten Lenkverhaltens, gegen den gegebenen Bausatz der Hochschule entschieden. Deswegen haben wir uns das Ziel gesetzt, eine eigene mobile Autoplattform, nach unseren Bedürfnissen, zu entwickeln. Da sich unser Team gleichverteilt aus MT- und MS-Studenten zusammensetzt, konnten wir die Aufgaben entsprechend unserer Fähigkeiten gut verteilen.

2 Technische Umsetzung

Das Projekt besteht aus dem Wagen und der dazugehörigen Software. Im Folgenden wird die Realisierung unseres Projektes erläutert.

2.1 Wagen

Der Wagen besteht aus drei Hauptkomponenten. Der Karosserie, die sich mit dem Grundgerüst des Wagens beschäftigt, die Lenkung und dem Antrieb.

2.1.1 Karosserie

Das Grundgerüst des Wagens besteht aus zwei rechteckigen Sperrholzplatten, die mit mehreren 3D-Gedruckten Abstandshalter, von fünf cm Länge, auseinander gehalten werden. Dies bietet uns die Möglichkeit, die Anordnung unserer Bauteile selbst festzulegen und nicht durch einen Bausatz limitiert zu sein. Im Laufe des Projektes war es von Vorteil, Holz als Material zu wählen. Änderungsmaßnahmen an anderen Materialien hätte deutlich mehr Arbeitszeit in Anspruch genommen.

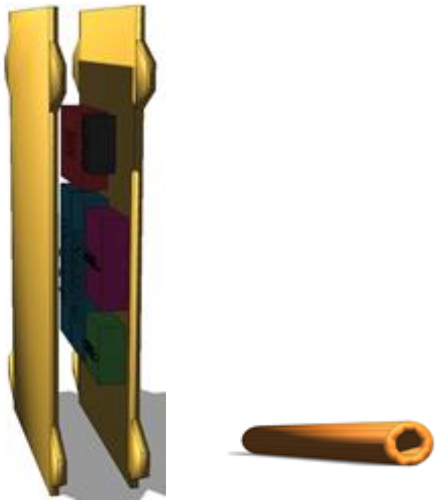

Abbildung 1: Abstandshalter

2.1.2 Lenkung

Die Lenkung wurde über eine Umlenkung an einem 90° Winkel gelöst. Zwischen Servoarm und der Umlenkung befinden sich gegenläufige Gewindestangen zur Feinjustierung der Lenkung. Jedes Rad ist einzeln aufgehängt, gefedert und läuft auf einem Kugellager.

Abbildung 2: Foto der Lenkung

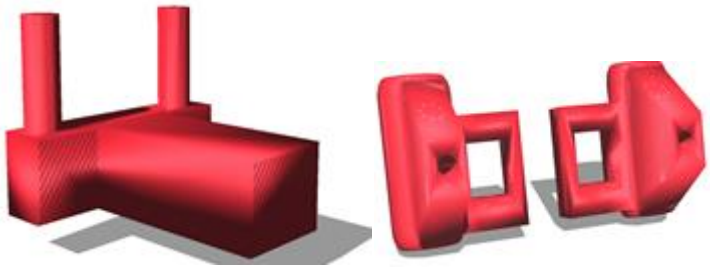

Abbildung 3: Lenkachsenteile

2.1.3 Antrieb

Der Antrieb des Wagens besteht aus einem „Automodell Brushed Elektromotor“ von Modelcraft, mit 12292 Umdrehungen die Minute. Die Hinterachse ist über eine Übersetzung (4:1) mit dem Motor verbunden. Die Teile für die Übersetzung kommen aus dem Modellbauhandel, so auch die Motorbefestigung. Um das Ritzel an der Achse zu befestigen, mussten wir ein Teil designen, dass das Ritzel an der gewünschten Position hält (Abbildung 4). Die Umsetzung wurde mit Hilfe der Software Tinkercad und dem von der Hochschule zur Verfügung gestellten 3D-Drucker realisiert. Die Antriebsritzel-Achsbefestigung wurde hierbei mit drei m3 Schrauben auf dem Achsritzel befestigt. Das modifizierte Ritzel wurde mit der Achse verbunden. Dies geschah durch ein Loch, das in die Achse gebohrt wurde. Außerdem haben wir einen Fahrtenregler integriert (Graupner V40R). Dieser regelt die Spannung am Motor und somit die Drehzahl durch Pulsweitenmodulation.

Um sicher zu gehen, dass der Fahrtenregler (ESC) für unsere Zwecke geeignet ist mussten zwei Kriterien erfüllt sein.

1. Die durchschnittliche Stromaufnahme des Motors muss kleiner sein, als die zulässige Dauerbelastung des Reglers.
2. Die maximale Stromaufnahme des Motors (Blockierstrom) muss kleiner sein, als die zulässige kurzzeitige Belastbarkeit des Reglers.

Durch Herstellerangaben und die Formel für die Leistungsberechnung

$$P = U * I \text{ bzw. } I = \frac{P}{U}$$

konnten wir beide Kriterien überprüfen.

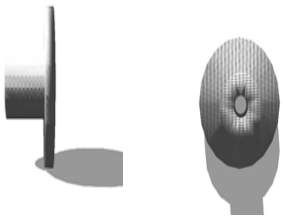

Abbildung 4: Antriebsritzel-Achsbefestigung

Abbildung 5: Foto Antriebsachse

2.1.4 Kamerahalterung

Die Kamerahalterung besteht aus einem auf einen vertikalen Servomotor montierte Halterung, auf der eine fünf Megapixel CMOS Kamera mit Weitwinkelobjektiv befestigt ist.

Abbildung 6: 3D-Druck: Kamerahalterung

2.2 Software

Die Pi-lot App steuert den Wagen und zeigt den Videostream an, die RC-Anwendung auf dem Raspberry Pi (Pi-lot RX) steuert die Servos und den ESC und die Anpassung der Linux Distribution (Daemons + Hotspot) sind Themen mit denen wir uns im Folgenden näher beschäftigt haben.

2.2.1 Pi-lot App

Die Pi-lot App dient dazu, die Steuerung vom Joystick auszulesen, zu korrigieren und an den Raspberry Pi zu senden. Außerdem zeigt er den Videostream vom Wagen an.

Der Joystick, in unserem Fall ein Playstation 3 Controller, ist mittels einem USB OTG-Adapter mit dem Smartphone verbunden. Die Pi-lot App liest die vom Playstation 3 Controller gesendeten Float-Werte, die die Positionen der Tasten und des Joysticks wiederspiegeln. Diese Float-Werte, also das Verhalten der Steuerelemente, können und mussten korrigiert werden. So wurde die maximale Geschwindigkeit des Wagens halbiert, um eine längere Akkulaufzeit zu gewährleisten. Außerdem wurde die Lenkung auf 30% im Ausschlag limitiert, um den Servo vor Schäden zu schützen.

Der Servo für die horizontale Kamerasteuerung wurde aufgrund der Verkabelung um 180° versetzt eingebaut, daher wurde die Steuerung für die Kamera invertiert.

Die korrigierten Float-Werte werden in einem ByteArray konvertiert und mittels dem User Datagram Protocol (UDP) an den Raspberry Pi gesendet.

Der MJPEG-Stream, der vom Raspberry Pi gesendet wird, wird mit der App angezeigt. Die Codegrundlage hierfür basiert auf den “Mjpeg view library for Android” von Andrea Stagi. Diese wurde jedoch für unsere Bedürfnisse angepasst. Einer der Veränderungen, ist die Möglichkeit den empfangenen Videostream als Doppelbild darzustellen. Dies ermöglicht uns beim Aufsetzen einer VirtualReality-Brille aus der Ich-Perspektive (engl. First Person View) zu fahren.

2.2.2 Pi-lot RX

Die Pi-lot RX Anwendung, dient dazu die angeschlossenen Servos und den ESC zu steuern.

Der Raspberry Pi empfängt das von der App konvertierte und per UDP versendete ByteArray und wandelt diese wieder in Float-Werte zurück. Die Float-Werte werden daraufhin in Servopulssignale umgesetzt. Die Servopulssignale steuern den ESC, die Lenkung und den Servo für die Kamerahalterung. Zur Sicherheit werden bei einem Timeout, also bei ausbleiben der Steuersignale vom UDP Socket, nach 120ms alle Steuersignale auf Neutral gesetzt.

2.2.3 Raspbian Einrichtung

Um die Kommunikation zwischen dem Smartphone und dem Raspberry Pi zu ermöglichen, haben wir ein WLAN-Netzwerk auf dem Raspberry Pi eingerichtet. Damit nicht jedes Mal beim Start des Raspberry Pi‘s der Pi-lot RX und der MJPEG-Streamer manuell gestartet werden muss, haben wir Daemons hierfür eingerichtet. Diese stellen sicher, dass während des Bootvorgangs die Anwendungen ausgeführt werden und aktiv bleiben.

Der Raspberry Pi erzeugt mit der Pi-Cam einen 24FPS, 320x240 Schwarz-Weiß-Videostream. Diese Maßnahmen wurden getroffen, weil der Videostream dadurch weniger Daten pro Bild senden muss. Dies hat zur Folge, dass die Latenz des Videostreams minimiert wird.

3 Abbildungsverzeichnis

4 Quellenverzeichnis

Stagi, Andrea: Mjpeg view library for Android (2014),
URL: https://github.com/astagi/MjpegView (Stand: 19.07.2018)

rhobincu: mjpg-streamer (2007),
URL: https://github.com/jacksonliam/mjpg-streamer (Stand: 19.07.2018)

Graupner: Bedienungsanleitung Navy V40R (2010),
URL: https://www.graupner.de/mediaroot/files/2836_2840P_Regler_Purple_de_en_fr.pdf (Stand: 19.07.2018)