

Dokumentation des Unipraktikums

Manuel Hinz

October 1, 2018

13.15 uhr ab 12 in den raum

Contents

1	Einleitung	3
1.1	Motivation	3
1.2	Zielsetzung	3
2	Plannung	4
2.1	Ausblick	4
2.2	Programmierung	4
2.2.1	Einleitung	4
2.2.2	Python	4
2.2.3	Programmstruktur	4
2.3	Technologien	5
2.3.1	Github	5
2.3.2	Slack	5
2.3.3	L ^A T _E X	5
2.3.4	Hilfsprogramme?	5
2.4	Elektrotechnik	5
3	Realisierung	6
3.1	Probleme	6
3.1.1	Gelste Probleme	6
3.1.2	Ungelste Probleme	6
3.2	Teamdynamik	6
4	Anhang	7
4.1	Quellen	7
4.1.1	Elektrotechnik	7
4.1.2	RaspberryPi	7
4.1.3	GrovePi	7

Chapter 1

Einleitung

1.1 Motivation

Im Moment sind selbstfahrende Autos so ziemlich das technische Thema. Universitäten rund um den Globus forschen an verschiedensten Technologien, welche das selbstfahrende Auto braucht. Ein Autohersteller nach dem anderen kauft entweder Startups zu dem Thema auf, oder investiert in eine eigene Forschungsabteilung. Als sich nun die Möglichkeit eines Praktikums im Rahmen des Elektrotechnikleistungskurses an der Bergischen Universität Wuppertal anbot, fiel die Wahl des Themas leicht. Dass dieses Praktikum nicht nur eine gute Möglichkeit ist um den aktuellen technischen Entwicklungen nachzueifern, zeigt sich schon an den verschiedenen Aufgaben die zu erledigen sind. Im Laufe des Projektes müssen nämlich Motoren mithilfe von H-Brücken und Pulsweitenmodulation gesteuert, ein Programm entworfen und ein Raspberry Pi in Betrieb genommen werden. Neben dem technischen Wissen, was erlernt und angewendet werden muss (und sich praktischerweise relativ gut mit dem Abiturstoff deckt), wird hier auch ein Einblick in den Universitätsalltag erlangt.

1.2 Zielsetzung

Es gibt viele mögliche Aufgaben, die man im Rahmen des Themas lösen könnte. Besonders interessant sind hier vor allem das Finden von einem und das Einparken auf einem Parkplatz, sowie das Teilnehmen am Straßenverkehr. Letzteres entfällt wegen der Schwierigkeit des Simulierens von anderem Verkehr. Deshalb ist die hier gewählte Aufgabe das Finden und Befahren einer Parklücke in einem straßenähnlichen Setting. Genauer gesagt wird das Auto eine Strecke entlangfahren und mit Hilfe von Sensoren eine Parklücke in die es hineinpasst identifizieren und dann dort einparken. Dies wird innerhalb von 16 90-minütigen Einheiten über einen Zeitraum von sieben Wochen. Ein weiteres Ziel ist es das für die Vollendung des Projektes nötige Wissen für alle Gruppenmitglieder verständlich zu bearbeiten und die Gruppe damit für das Abitur und weitere Projekte vorzubereiten.

Chapter 2

Plannung

2.1 Ausblick

2.2 Programmierung

2.2.1 Einleitung

Bei der Programmierung des Projekts haben wir einen objektorientierten Lösungsansatz gewählt. Dies haben wir einerseits wegen der Thematisierung der Objekt-Orientierten-Programmierung (OOP) im Informatikunterricht und andererseits wegen dem einfachen Bezug dem man hier zwischen realen Objekten (Sensor, Motor) und zugehörigen Klassen herstellen kann. Außerdem ist die OOP einer der Paradigmen, welche in der Realität benutzt werden.

2.2.2 Python

Als Programmiersprache haben wir Python gewählt, da wir diese Sprache im Informatikunterricht verwenden, sie für Laien leicht verständlich ist und wir mehrere Personen in der Gruppe bzw. dem nahen Umfeld haben, welche diese Sprache bereits gut kennen. Außerdem unterstützt Python die OOP, wodurch die Sprache auch unsere Paradigmenwahl zulässt. Da sie eine beliebte und weitverbreitete Sprache ist unterstützt sowohl das aktuelle Image das wir auf dem Pi haben, als auch der Hersteller unsere Sensoren die Sprache. Auch kann man sagen, dass die Sprachwahl es uns ermöglicht Code, welcher schon früher von einzelnen Mitgliedern geschrieben wurde, wieder zu benutzen. Ein Nachteil ist hier, dass Python nicht alle Konzepte der OOP unterstützt (zum Beispiel Abstrakte Klassen, Interfaces/Traits).

2.2.3 Programmstruktur

Jedes große Bauteil bekommt eine Klasse zugewiesen. Die wichtigen Bauteile sind hier: die Sensoren, die Motoren, die Motortreiber und der Raspberrypi.

Für die Sensoren haben wir drei Klassen: Einmal eine abstrakte Klasse Sensor, welche definiert, dass jeder Sensor ein Signal auslesen(`get_value`) und die Bezeichnung des Pins ausgeben(`get_location`) kann. Wir haben dann zwei Klassen, namentlich `LightSensor` und `DistanceSensor`, welche von Sensor erben, und jeweils die Methode zum Signalauslesen, sowie den Konstruktor beschreiben.

2.3 Technologien

2.3.1 Github

Die Gruppe benutzt Github als unser Versionskontrollsystem. Github bietet sich hier an, weil einerseits git auf dem Raspberrypi ist, wodurch man immer die aktuellste Version des Programmes auf dem Pi haben kann. Außerdem erleichtert Github das gleichzeitige Arbeiten, sodass mehrere Leute gleichzeitig programmieren, und die Programme schlussendlich hochladen können. Ein weiteres Plus ist, dass Github eine große Anzahl an letzten Versionen speichert, wodurch auch spät erkannte Fehler leicht überwunden werden können. Als letztes Argument für diese Technologie ist zu sagen, dass sie quasi als Dokumentation für alles was auf Github hochgeladen wird dienen kann. Da man jeden Schritt nachvollziehen kann.

2.3.2 Slack

2.3.3 L^AT_EX

Für die Tagesdokumentationen und die Hauptdokumentation wird L^AT_EX verwendet. Dies ist einerseits eine Referenz auf die echte Welt, in welcher L^AT_EX oft (im wissenschaftlichen und im privaten) benutzt wird. Andererseits wollen wir von den Vorteilen profitieren. Die Vorteile liegen darin, dass man einfacher alle Kapitel, Sektionen und Formeln variabel gleichzeitig ändern kann. Außerdem ist L^AT_EX eine Alternative zu Word, wobei man viele der blöden Probleme von Word umgeht. Beispiele wären hier zum Beispiel Verschiebung von Text bei dem Einfügen von Bildern, welches man in L^AT_EX deutlicher definieren muss.

2.3.4 Hilfsprogramme?

2.4 Elektrotechnik

Chapter 3

Realisierung

3.1 Probleme

3.1.1 Gelste Probleme

3.1.2 Ungelste Probleme

3.2 Teamdynamik

Chapter 4

Anhang

4.1 Quellen

4.1.1 Elektrotechnik

- http://cdn-reichelt.de/documents/datenblatt/A100/BC546_48-CDIL.pdf
- <http://anleitung.joy-it.net/wp-content/uploads/2017/06/SBC-MotoDriver2-Anleitung.pdf>
- http://cdn-reichelt.de/documents/datenblatt/A300/COM_MOTOR_RAD_DB-DE.pdf

4.1.2 RaspberryPi

- https://www.terraelectronica.ru/pdf/show?pdf_file=%252Fds%252Fpdf%252FT%252FTechicRP3.pdf
- http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_GPIO_C.html
- <https://help.github.com/enterprise/2.14/user/articles/generating-a-new-ssh-key-and-add>
- <https://devmarketer.io/learn/set-ssh-key-github/>

4.1.3 GrovePi

- http://cdn-reichelt.de/documents/datenblatt/A300/GRV_IR_DISTANCE_AL-EN.pdf
- http://www.mouser.com/catalog/specsheets/Seeed_101020022.pdf
- <https://github.com/DexterInd/GrovePi/blob/master/Hardware/GrovePi%2B%20v3.0.pdf>

- <https://github.com/DexterInd/GrovePi/blob/master/Hardware/GrovePi%20Graphical%20Datasheet.jpg>