# Einführung

Bachelorprojekt eingereicht im Rahmen der Bachelorprüfung

im Studiengang Mechatronik am Departement Fahrzeugtechnik und Flugzeugbau der Fakultät Technik und Informatik der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer: Prof. Dr. Rasmus Rettig

Abgabedatum: 18. Juni 2021 <-Aktuelles Datum

# 3.

Wichtige Anforderungen dafür sind auch, neben einer benutzerfreundlichen Bedienung, ~~ist auch~~ eine Konstruktion, ~~die~~ welche den Einsatz im Straßenverkehr ermöglicht.

# 4.1

Nach einer Anfrage bei der Firma „item Industrietechnik GmbH“ nach Nutensteinen mit entsprechendem 1/4“ Innengewinde, war bekannt, dass sich derartige Produkte auf dem Markt nicht ohne Weiteres besorgen lassen.

Der Tausch des Stativs durch ein alternatives, ebenfalls zur Verfügung stehendes,

# 4.2

Nach sehr kurzer Diskussion war beschlossen, dass mittels CAD-Technologie ~~ein~~ Gehäuse konstruiert und ~~im vorhandenen~~ sie mit dem im Labor vorhandenen 3D-Drucker gefertigt werden sollen (Abbildung 2).

In einer ~~zweiten~~ dritten Überarbeitung sollten diese beiden Punkte beachtet werden.

Nach der Neukostruktion (von was??) und der Ergänzung einer Stützkonstruktion für den Raspberry Pi 4, war es möglich in der Installationsbox alle Bauteile zu platzieren, ohne eine erneute Neuanschaﬀung zu tätigen. (Abbildung 6)

# 4.4

Um die Mikrofone entlang einer Linie positionieren zu können, wurde ein ~~1,1 m langes~~ vorhandenes Aluminiump ~~item-P~~roﬁl von item aus der Serie 6 in den Dimensionen 1100x30x30 mm eingesetzt, auf ~~dem~~ welchem die Gehäuse der Mikrofone nun mit Nutensteinen in deﬁnierte~~m~~n ~~A~~Äbstanden montiert werden können.

# 5 Programmierung

~~5.0.1~~ Beweisführung <- Wo ist 5.0.2?

# 5.0.1

Abbildung 7 ist zu klein

Um die Komplexität des Systems möglichst gering halten zu können, wird die Kameraansteuerung direkt in dem Hauptprogramm mithilfe einer Wrapper-Klasse realisiert.

# 5.1

Abbildung 8 ist zu klein

# 5.1.1

Nach dem Programmaufruf wird die DOA Konﬁguration… <-Was ist DOA?

Abbildung 9 ist zu klein

# 5.1.3

…sie dann mithilfe der *fastAddValue* Methode im Signed UInt32 Format in den Speicher.

Am Ende dieses Satzes fehlt der Punkt am Ende, und warum ist *fastAddValue* im Dokument kursiv formatiert

# 5.1.6

Die checkForBang Methode ist eine der wichtigsten im gesamten Programmablauf…

Warum ist die Methode eine der wichtigsten? Das geht aus dem Text nicht hervor…

Abbildung 10 ist zu klein

# 5.1.7

Diese Methode ist ~~nun~~ dafür verantwortlich…

# 5.1.8.

Die Klasse cameraControl benutzt ~~nun~~…..

# 5.2

Betrachten wir als Beispiel zwei reellwertige Funktionen f und g, die sich nur durch eine unbekannte Verschiebung entlang der x-Achse unterscheiden ~~. Man kann~~ , kann man die Kreuzkorrelation verwenden~~, um herauszuﬁnden,~~ um wie viel g entlang der x-Achse verschoben werden muss, damit es mit f identisch wird.

~~In unserem~~ Im vorliegenden Fall werden….

~~Dann~~ Damit wird in der xcorr Methode die maximale Latenz….

# 6.1

Abbildung 11 ist zu klein

# 6.2

Abbildung 12 ist zu klein

Die Erwartungshaltung an die Ergebnisse ~~war~~ ist ein sauberer Ausschlag…

Lediglich die Aufzeichnungen aus Mikrofon 1 ~~entsprachen~~ entsprechen den Erwartungen(Ausschlag bei ca. 40.000 Samples), während die restlichen Mikrofone deutliches Rauschen ~~aufwiesen~~ aufweisen.

In der Aufzeichnung von Mikrofon 3 war selbst das Event (erwartet bei ca. 40.000 Samples) nicht zu erkennen, während auf den anderen Kanälen noch Andeutungen eines stärker verrauschten Bereiches zu erkennen ~~waren~~ ist. Dieses Verhalten ~~hätte~~ hat seine Ursache beim Raspberry Pi ~~haben können~~. Dieser muss für die Taktung der Mikrofone einen SPI-Takt über das gesamte Array betreiben können.

Durch ein Fachgespräch mit Max Weltz ~~schlossen wir auf~~  werden folgende mögliche Ursachen in den Fokus gerückt.

Um eine qualitative Ursachensuche vornehmen zu können, ~~beschlossen wir~~ werden die aufgezählten Ursachen separat ~~zu Untersuchen~~ untersucht.

# 6.3

~~Weil~~ Da für den Mikrofontest keine eigenen Kabel verwendet ~~wurden~~ werden,…

Dafür ~~wurden~~ werden diese mit einem Mulitmeter erst auf Durchgang….

# 6.4

~~In einer zweiten Iteration wurde das System umkonﬁguriert, sodass jetzt mit einem SPI-Takt von 3,9 MHz und nicht wie ursprünglich mit 7,8 MHz gearbeitet wird.~~

In einer zweiten Iteration soll das System so angepasst werden, dass der SPI-Takt von den ursprünglichen 7,8 MHz auf 3,9 MHz reduziert wird.

Für diese ~~Konﬁguration wurde~~ Anpassung wird die gleiche Testmethode verwendet. ~~und es ist zu erkennen,~~ Wie vermutet ist ~~dass~~ die Messungen nun deutlich weniger verrauscht ~~waren~~.

Die Messungen ~~waren~~ sind somit immer noch zu stark verrauscht…

Eine reine Reduzierung des ~~verwendeten~~ SPI-Takts hat das Problem ~~also noch~~ nicht vollständig ~~gelöst~~ lösen können.

Abbildung 13 ist zu klein

# 6.5

In der nächsten Anpassung ~~wurde~~ werden die Kabel~~länge~~ der Mikrofone~~kabel~~ auf ~~nun~~ ca. 25 cm reduziert…

Die nun durchgeführte Messung ~~wies~~ weist das gewünschte, rauschfreie Verhalten der Messwerte, auf allen Kanälen, auf. Die Kabel ~~wurden~~ sind somit, in Bezug auf die Datenübertragung und Funktion, hinreichend ~~auf die Funktion~~ überprüft.

Abbildung 14 ist zu klein

# 6.6

Im nächsten Testschritt ~~wurde~~ wird das…

Das bedeutet das System ~~wurde~~ wird…

Zunächst wurde wird…

Die Aufgezeichneten Werte der Messungen, also 205000 Messwerte pro Mikrofon, ~~wurden~~

werden….

Die Lokalisierung ergab folgendes Bild: <-in der Doku ist dort kein Bild eingefügt!

Das Ergebnis der Berechnung stimmte ebenfalls mit den Erwartungen überein, da das Geräusch ebenfalls ungefähr frontal vor dem System erzeugt ~~wurde~~ wird.

Abbildung 15 ist zu klein

Abbildung 16 ist zu klein

# 6.7

Im nächsten Test ~~wurde~~ wird…

Auch hier ~~wurde~~ wird…

In diesem Test ~~wurde~~ wird…

Das bedeutet, dass das System ~~lief~~ endlos läuft (bis es durch den Nutzer abgebrochen ~~wurde~~ wird), während~~e~~ ~~durch~~ das Programm fortlaufend~~e~~ Messdaten ausgewertet ~~wurden~~.

Erwartet wurde eine Benachrichtigung des Nutzers über ein erkanntes Event mit des zugehörigen Lokalisierungsergebnis über die Systemkonsole mit einer zugehörigen .png-Datei, auf der das erkannte Segment, farblich markiert wurde.

Erwartet wird, dass der Nutzer eine Benachrichtigung, über ein erkanntes Event, mit dem dazugehörigen Lokalisierungsergebnis, über die Systemkonsole, mit einer zugehörigen .png-Datei, auf der das erkannte Segment, welches farblich markiert wird, erhält.

Der Versuch ~~hat gezeigt~~ zeigt, dass das System den erhöhten Pegel eines auftretenden Events sehr gut ~~erkennen kann~~ erkennt. Zwar ~~kam~~ kommt es immer wieder zu variierenden Verzögerungen in der Anzeige auf der Konsole, aber ~~es wurden~~ dennoch werden aufeinanderfolgende Events korrekt hintereinander aufgelistet.

Der hierzu verwendete Schwellenwert, ab dem das System auslösen soll, ~~musste~~ muss…

Wie im obigen Bild aufgezeigt, ~~konnte~~ kann

Bei wiederholter Testdurchführung ~~kam~~ kommt es allerdings auch immer wieder zu Messungen, die nicht den Erwartungen ~~entsprochen haben~~ entsprechen.

Obwohl sich der Ursprungsort des Events ungefähr in der Bildmitte ~~befand, wurde~~ befindet, wird…

Durch die Fehllokalisierung ~~trat~~ tritt darüber hinaus auch der Fall auf, dass das Event zwar korrekt erkannt ~~wurde~~ wird, aber aufgrund des fehlerhaft berechneten Ursprungswinkel ~~wurde~~ wird das Event als außerhalb des Bildbereiches bewertet und keine Beweisführung durchgeführt.

~~Dadurch waren wir gezwungen~~ Die hat zu Folge, dass die Lokalisierung mit einer deutlich reduzierten Menge an Messwerten ~~durchzuführen~~ durchgeführt wird.

Leider war es ~~uns~~ nicht möglich ein Testverfahren zu entwickeln, mit welchem sich die Genauigkeit der Lokalisierung qualitativ bewerten lässt, um so ein Minimum an benötigten Messdaten zu ermitteln ~~zu können~~. Die verwendete Buﬀergröße ~~wurde~~ wird lediglich durch ~~Tests~~ ~~ermittelt~~ durch die Testergebnisse bestimmt.

# 7.

Für Systeme, wie sie in der Geschwindigkeitsüberwachung eingesetzt werden, ist die Minimierung der Totzeit sehr wichtig. Sie beschreibt die Zeit, in ~~der~~ welcher das System nicht in der Lage ist, ein neues Event zu erkennen, ~~weil~~ da es z.B. noch mit der Datenauswertung oder Datenverarbeitung beschäftigt ist.

Eine Messung dieser Zeiten ~~war~~ ist nur systemintern, mit Software-Timer, möglich, welche ~~die~~ in diesem Fall ~~allerdings~~ hinreichend genau ~~waren~~ sind.

Die Tests ~~haben gezeigt~~ zeigen,…

Zu Testzwecken ~~wurde~~ wird diese um ein Vielfaches…

# 8.

Zunächst ~~müssen~~ muss sich der „Raspberry Pi“ und das Zweitgerät in einem gemeinsamen Netzwerk beﬁnden.

Besteht eine gültige SSH-Verbindung, ~~so~~ gibt es ~~nun~~ zwei Betriebsvarianten.

Zu Testzwecken wurde das System ~~z.B.~~ so eingestellt, dass es in einem ruhigen Umfeld ein Klatschen erkennt.

Der Prozess wird korrekt beendet, wenn zuerst das Hauptprogramm in der IDE und anschließend das Skript beendet ~~werden~~ wird.

Das Skript muss sich jedoch im gleichen Verzeichnis wie die von der IDE ausführbar kompilierte Version des Hauptprogrammes und das Skript zum ~~starten~~ Starten der Pipeline befinden.

Je nach Konﬁguration des Hauptprogramms läuft der Prozess ~~nun wieder unendlich,~~ endlos.

~~in~~ In diesem Fall können aber alle Unterprogramme mit einem „stg+C“ Kommando gleichzeitig beendet werden.

So lassen sich Kopier- und Bearbeitungsvorgänge, sowie die Sicherung und Sichtung der ~~B~~bearbeiteten Beweisbilder direkt über das verwendete, zweite Endgerät ~~durchgeführt werden~~ durchführen.

# 9.

Dieses Verhalten konnte auf die Position des Arrays zum Wind zurückgeführt werden.

An besonders windigen Tagen hat dieser in den Mikrofongehäusen immer wieder Störgeräusche verursacht, ~~die~~ welche bereits ~~ausreichend waren~~ ausgereicht haben, um die Pegelerkennung auszulösen.