Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Tobias Palmer

Zustandserkennung und Schätzung des aktuellen Szenarios anhand von Sensordaten einer smarten Tasse

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Tobias Palmer

Zustandserkennung und Schätzung des aktuellen Szenarios anhand von Sensordaten einer smarten Tasse

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

Erstprüfer: Prof. Dr. rer. nat. Thomas Lehmann
Zweitprüfer: Prof. Dr.-Ing. Andreas Meisel

Abgabedatum: 15/09/2017

Zusammenfassung

Tobias Palmer

Thema der Bachelorthesis

Zustandserkennung und Schätzung des aktuellen Szenarios anhand von Sensordaten einer smarten Tasse

Stichworte

Ubiquitous Computing, Zustandserkennung, smarte Tasse

Kurzzusammenfassung

Ziel dieser Arbeit ist es, eine Zustandserkennung durch Auswertung der aktuellen Sensordaten innerhalb einer smarten Tasse zu realisieren. Hierfür sollen die erkannten Gesten kategorisiert und anschließend einem möglichen Szenario, in welchem sich die Tasse aktuell befindet könnte, zugeordnet werden. Je nach Situation ist es dann Möglich im Rahmen dieser Arbeit relevante Sensordaten und Sendeintervalle auszuwählen. Dies führt zur Reduzierung von Masse und Datenrate der gesendeten Daten und verbessert die Betriebszeit der Tasse. Für die Entwicklung der Algorithmen wird eine experimentelle und eine modellbasierte Herangehensweise gewählt

Tobias Palmer

Title of the paper

State recognition inside a smart mug to predict the possible scenarios. Based on sensor data.

Keywords

Ubiquitous Computing, State recognition, smart mug

Abstract

This Bachelor thesis aims to implement a state recognition inside a smart mug to predict the possible scenarios. When the probable scenario is identified, the result and accumulated data can be send to the back end for further processing. The scenario recognition will recognise everyday gestures like drinking or transport processes. The preselection of relevant data inside the mug reduces data traffic and hence extending battery life. A model based and an experimental approach will be used for algorithm development.

Inhalt

Abkürzungsverzeichnis

BF ..... *Bodyframe*
CSTI ..... *Creativ Space for Technical Innovations*
CVS ..... *Character-Separated Values*
DBSCAN ..... *Density-Based Spatial Clustering of Applications with Noise*
IMU ..... *Inertial Measurement Unit*
KDD ..... *Knowledge Discovery in Databases*
KS ..... *Koordinatensystem*
Tessa ..... *TEst Safty Security Analysis*
WF ..... *Worldframe*

Abbildungs- und Tabellenverzeichnis

1 Einleitung

Der Creative Space for Technical Innovations (CSTI) der HAW Hamburg bietet Raum und Ressourcen für innovative Ideenentwicklung und deren Umsetzung als Prototyp. Dabei liegt ein Hauptaugenmerk des CSTI auf der interdisziplinären Realisierung von Konzepten und Forschungsansätzen im Bereich Human-Computer-Interaction und Smart Systems [1].
In enger Zusammenarbeit mit dem CSTI entwickelte Joschka Sondhof im Rahmen seiner Studienarbeit den Prototyp einer smarten Tasse. Er nannte Sie Mate-Mug, nach dem englischen Wort für Kumpel. Er legte damit einen Grundstein für vielfältige Weiterentwicklungsmöglichkeiten, zum Beispiel im Bereich des Ubiquitous Computing, der selbstauskunftsfähigen Objekte, der Einbindung in die virtuelle oder erweiterte Realität oder der Datenfusion mit anderen vernetzten Gegenständen zur Kontexterfassung [2].
Diese Bachelorarbeit schließt im Bereich Selbstauskunftsfähigkeit an Sondhofs Studienarbeit an, liegt thematisch nah bei dem populären Thema Ubiquitous Computing, und orientiert sich am aktuellen Zeitgeist.

1.1 Motivation

Aufgrund immer kleiner, günstiger und leistungsstärker werdenden Mikrocontrollern und der sich stetig verbessernden kabellosen Internet- und Datenverbindung verschwinden Computer mehr und mehr aus dem Raum der bewussten Wahrnehmung. Das Ubiquitous Computing ist stark auf dem Vormarsch. Dabei zeichnet sich folgende Tendenz ab: Statt großer multifunktionaler Alleskönner werden es immer mehr und immer kleinere Computer werden, die auf das Erledigen weniger, besonderer Aufgaben spezialisiert sind [3].
Viele kleine unauffällige Computer können dem Benutzer helfen, Alltagsprobleme besser zu bewerkstelligen [4]. Einer dieser kleinen Computer befindet sich in der Mate-Mug.
Eine selbstauskunftsfähige smarte Tasse ermöglicht das kontrollieren der Flüssigkeitsaufnahme und kann helfen stets ausreichend hydriert zu sein. Die Tasse löst das Problem des vergessenen kalten Kaffees, da sie auf sich aufmerksam machen kann, bevor der Inhalt komplett ausgekühlt ist. Genauso kann die Tasse vor dem Verbrühen an zu heißen Getränken schützen, indem sie visuell vor der zu heißen Temperatur warnt.
Auch im Bereich der Alten- und Krankenpflege wäre eine smarte Tasse hilfreich. Wenn Patienten in Pflegeheimen Hilfe benötigen, müssen sie auf einen Knopf drücken, um per Freisprecheinrichtung mit einer Pflegekraft verbunden zu werden. Dies ist aber zum Beispiel nach einem Sturz nicht immer möglich. So muss der Patient gegebenenfalls mit Schmerzen warten, bis ihn eine Pflegekraft bei einem Rundgang entdeckt. Wenn die smarte Tasse erkennt das sie fällt, könnte sie automatisch eine Verbindung zum Pflegepersonal aufbauen und so für schnelle Hilfe sorgen wo sie nötig ist.

Gesundheitliche Verschlechterungen werden bei Patienten in vielen Fällen nur bei regulären Pflichtuntersuchungen festgestellt. Oft wird im Zuge dieser Untersuchung die Medikation ergänzt oder umgestellt. Eine Anpassung der medikamentösen Behandlung an den Gesundheitszustand des Menschen erhöht seine Lebensqualität. Verschlechterungen lassen sich häufig anhand der Veränderungen von Bewegungsabläufen erkennen. Solche Veränderungen, beispielsweise zitternde Hände oder ein verändertes Gangbild, könnte die Tasse wahrnehmen. So wäre der Indikator für eine außerplanmäßige Untersuchung vorhanden.
Die Behandlung von Schlaganfallpatienten kann durch Gestenüberwachung, welche unter anderem mit einer smarten Tasse erfolgt, verbessert werden. Aktuelle Forschungsergebnisse dazu stehen in [5]. In der Informatik wird der Begriff Geste gewöhnlich im Zusammenhang mit Gestensteuerung genutzt und beschreibt dann bestimmte Handhaltungen oder Handbewegungen, zum Steuern eines Computers [6]. In dieser Arbeit wird der Begriff Geste allerdings als definierte Bewegung des Körpers gesehen welche Informationen enthält [7].
Auch die anderen aufgeführten Beispiele lassen sich anhand von Gesten beziehungsweise Bewegungsabläufen erkennen. Ein großer Fokus dieser Arbeit liegt deshalb auf der Auswertung der Bewegungsabläufe mit Hilfe der Inertial Measurement Unit (IMU).

1.2 Problembeschreibung

Die von Joschka Sondhof in seiner Studienarbeit entwickelte Mate-Mug ist mit ihrer verbauten Sensorik in der Lage, die auf die Tasse wirkende Beschleunigung, die Drehrate, das Erdmagnetfeld, die Temperatur, den Luftdruck, die Lichtintensität, das eigene Gewicht und den Sound der Umgebung zu erfassen. Außerdem kann sie über NFC, RFID, WLAN und Bluetooth mit der Umgebung kommunizieren. Das in der Tasse implementierte System arbeitet auf einem Arduino MKR1000 und kann für knapp drei Stunden betrieben werden.
Zum jetzigen Zeitpunkt können die aktuellen Messwerte per Bluetooth oder WLAN übertragen werden. Dabei nimmt die Tasse zyklisch die aktuellen Sensorwerte auf. Diese könnten dann über das Backend des CSTI verarbeitet werden, um komplexe Aussagen über Situation und Kontext zu machen.
Die Tasse sendet also Sensorwerte, die den der Zustand der Tasse bestimmbar machen, sie selbst kann jedoch den eigenen Kontext nicht wahrnehmen und kommunizieren.
Damit die Tasse in die Lage versetzt wird, ihren eigenen Zustand wahrzunehmen ergeben sich für diese Arbeit folgende Fragestellungen:

- Wo ist die Tasse im Einsatz, was für Szenarien ergeben sich daraus, und in welchen Zuständen befindet sich die Tasse in diesen Szenarien? Mit welchen Merkmalen lassen sich die Zustände charakterisieren, und wie können die Merkmale detektiert werden?
- Sind die erkannten Merkmale von Nutzer zu Nutzer unterschiedlich?

-Kann der Benutzer der Tasse anhand der Merkmale erkannt werden?

- Wie ist die Genauigkeit des erkannten Zustands und wie groß die Wahrscheinlichkeit des richtig zugewiesenen Szenarios?

Der Fokus dieser Arbeit liegt hier auf der Zustandserkennung.

1.3 Zielsetzung und Abgrenzung

Aus den zuvor erwähnten Fragestellungen und dem genannten Fokus ergibt sich das explizite Ziel dieser Arbeit, eine Zustandserkennung durch Auswertung der aktuellen Sensordaten innerhalb einer smarten Tasse zu realisieren. Hierfür sollen die erkannten Gesten kategorisiert und anschließend dem Szenario zugeordnet werden können, in welchem sich die Tasse höchstwahrscheinlich befindet. Je nach Situation können dann im Rahmen dieser Arbeit relevante Sensordaten und Sendeintervalle ausgewählt werden. Dies führt zu einer Reduzierung von Masse und Datenrate der gesendeten Daten und verbessert die Betriebszeit der Tasse.

1.4 Vorgehensweise

Vorbereitend zur eigentlichen Planung und Konzeption der Zustandserkennung müssen einige grundlegende Themen wie Wissensentdeckung in Daten und Positionsbestimmung im dreidimensionalen Raum betrachtet werden.

Dabei werden zum einen zwei bekannte Clusterverfahren vorgestellt welche für die Auswertung der Sensorwerte wichtig sind. Zum anderen werden zwei Arten der Positionsbestimmung beleuchtet die im späteren Verlauf gegenübergestellt werden. Diese Erörterung der Grundlagen erfolgt in Kapitel 2. Das anschließende dritte Kapitel befasst sich mit der Planung der Zustandserfassung. Dazu werden als erstes die nötigen Zustände definiert, bevor überlegt wird, wie und mit welchen Versuchen beziehungsweise Modellen diese erkannt werden können. Außerdem folgt die Entwicklung und Implementierung der Modelle und die schon erwähnte Gegenüberstellung der Positionsverfahren in Kapitel 3. Die Durchführung und Auswertung der Versuche ist in Kapitel 4 zu finden. Im Zuge der Planung fiel das Problem einer zu geringen Datenrate auf. Mögliche Lösungsansätze zur Erhöhung der Datenrate stehen in Kapitel 5. Das abschließende Kapitel 6 enthält eine Zusammenfassung der Erkenntnisse, beleuchtet offen gebliebene Fragen und gibt einen Ausblick auf mögliche Fortsetzungen der selbstauskunftsfähigen Tasse.

2 Grundlagen

Um eine Zustandserkennung durch Auswertung der aktuellen Sensordaten überhaupt möglich zu machen, müssen zunächst Daten in Datenbanken erfasst werden, um in ihnen neue Informationen zu entdecken. Die für die Informationsfindung nötigen Schritte werden unter Kapitel 2.1 vorgestellt, bevor in Kapitel 2.2 der wichtigste Teilschritt genauer betrachtet wird.
Um anhand der Sensordaten die Orientierung der Tasse bestimmen zu können, muss ein Verfahren der Positionsbestimmung im dreidimensionalen Raum durchgeführt werden. Zwei mögliche Verfahren werden in Kapitel 2.3 vorgestellt.

2.1 Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD), Englisch für "Wissensentdeckung in Datenbanken", wird häufig auch gleichbedeutend mit dem Begriff Data-Mining verwendet. Data-Mining ist jedoch nur ein Teilschritt der KDD [8]. Die komplette KDD beinhaltet fünf Schritte, welche in folgender Sequenz durchgeführt werden: Fokussieren, Vorverarbeitung, Transformation, Data-Mining und Interpretation [9]. Diese Schrittkette dient dazu, bisher unbekanntes und für die gegebene Anwendung potentiell nützliches Wissen (semi-) automatisch aus Datenmengen zu erheben [9]. Die einzelnen Schritte der KDD sind in Abbildung 1 dargestellt und werden nachfolgend erläutert.

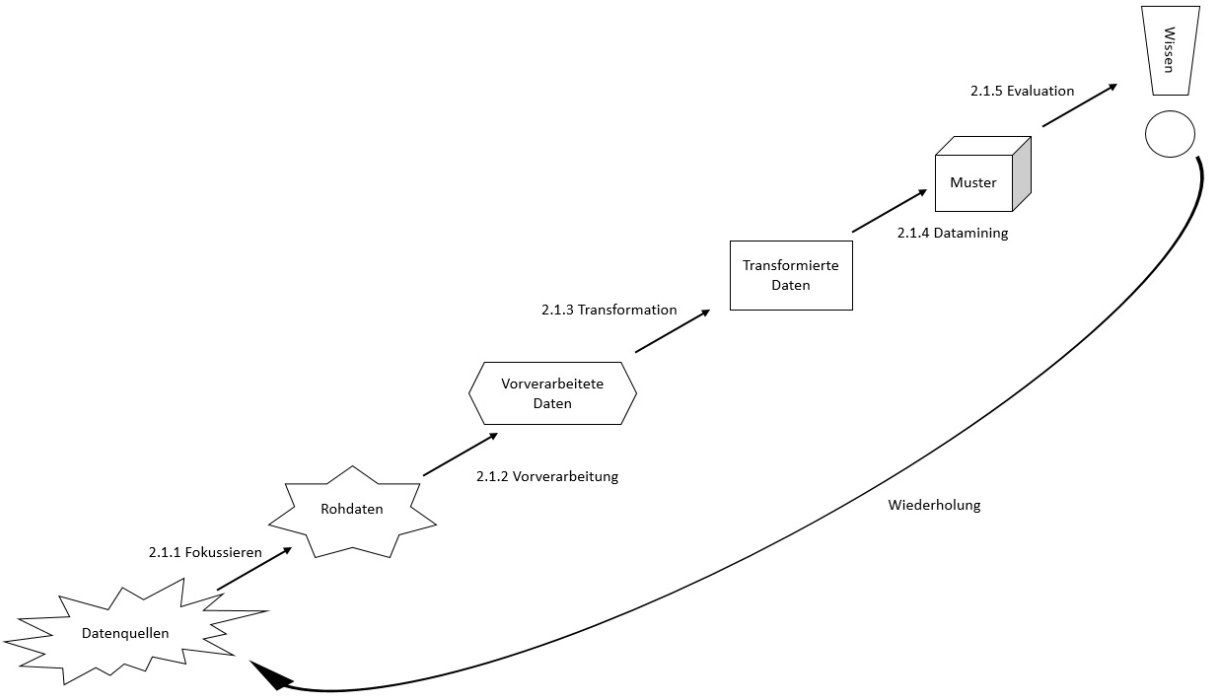

Abbildung 1: Schrittkette der KDD, nach [9 S.2]

2.1.1 Fokussieren

Je nach Anwendung müssen die Daten ausgewählt werden mit denen das KDD gespeist werden soll. Es muss festgelegt werden, ob diese direkt von den einzelnen Quellen kommen oder vorher in einem Zwischenschritt gebündelt werden.
Wenn die Datenquellen gewählt sind, muss über die Art der Speicherung entschieden werden. Wichtig ist dabei die Festlegung von Datenformat und Art der Speicherung. Daten können zum Beispiel als Binärdaten in einer Datenbank oder als lesbare Textdatei gespeichert werden. Auch dies muss stets in Abhängigkeit von Anwendung, Beschaffenheit und Weiterverarbeitung entschieden werden.

2.1.2 Vorverarbeitung

Die ausgewählten Daten können Fehler und Inkonsistenzen aufgrund von Mess- und Sensorfehlern enthalten. Sie liegen unter Umständen in einem Format vor, welches die weiteren Schritte der KKD erschwert. Wie stark die Rohdaten bearbeitet werden müssen, ist von ihrer Qualität und dem Format ihrer Einspeisung abhängig.
Sollten Rohdaten aufgrund der Fehler unbrauchbar sein oder sogar ganz fehlen, sollten die fehlerhaften Informationen falls möglich aus den anderen Daten berechnet werden, genaue Default-Werte als Alternative definiert, oder der Datensatz ganz verworfen werden.
Sollte die Vorverarbeitung nicht gründlich genug durchgeführt werden, kann es zu fehlerhaften Mustererkennung beziehungsweise übertrainierten Mustern kommen. Ist dies der Fall, entspricht das erkannte Muster nicht mehr dem allgemeinen sondern einem speziellen Muster, bei welchem die gleichen Fehler auch in den Ausgangsdaten vorhanden sind [8].

Daher umfasst die Vorbereitung der Vorverarbeitung in vielen KDD-Projekten einen Großteil des Gesamtaufwands [8].

2.1.3 Transformation

Um den Rechenaufwand für das Data-Mining möglichst gering zu halten, müssen die vorverarbeiteten Daten in eine für das Ziel der KDD geeignete Repräsentation transformiert werden. Denn häufig sind nicht alle Attribute eines Datensatzes für das aktuelle Ziel relevant.
Dies geschieht normalerweise durch Attributzusammenfassung beziehungsweise -selektion und die Diskretisierung von Attributen. Die Transformation kann automatisiert oder manuell durchgeführt werden, dies wird in [9] weiter ausgeführt.

2.1.4 Data-Mining

In diesem Schritt werden mit Hilfe von Algorithmen Muster in den Sensordaten entdeckt. Aus diesem Grund werden, wie schon erwähnt, Data-Mining und KDD oft synonym verwendet.
Die wichtigsten dafür genutzten Verfahren werden in Abbildung 2 verdeutlicht und lauten:

Clustering: Mit Hilfe des Clusterings sollen Daten in Cluster (Gruppen) zusammengeschlossen werden, so das Objekte in einem Cluster möglichst ähnlich sind, sich aber von Objekten in einem anderen Cluster möglichst stark unterscheiden. Ausreißer, Daten welche zu keinem Cluster gehören, können hierdurch auch identifiziert werden.
Klassifikation: Durch Trainingsobjekte mit Attributwerten, die bereits Teil einer Klasse sind, werden Funktionen angelernt. Diese angelernten Funktionen sollen zukünftig Objekte aufgrund ihrer Attributwerte in Klassen aufteilen.
Assoziationsregeln: Wenn eine Datenbank mit geregelten Abläufen vorhanden ist, können Assoziationsregeln häufig auftretende, starke Zusammenhänge innerhalb der Abläufe beschreiben (WENN A UND B DANN C).
Generalisierung: Das Ziel der Generalisierung ist eine kompakte Darstellung der vorhandenen Datenmenge. Dies wird durch Verallgemeinerung der Attributwerte und somit durch Reduzierung der Datensätze erreicht.

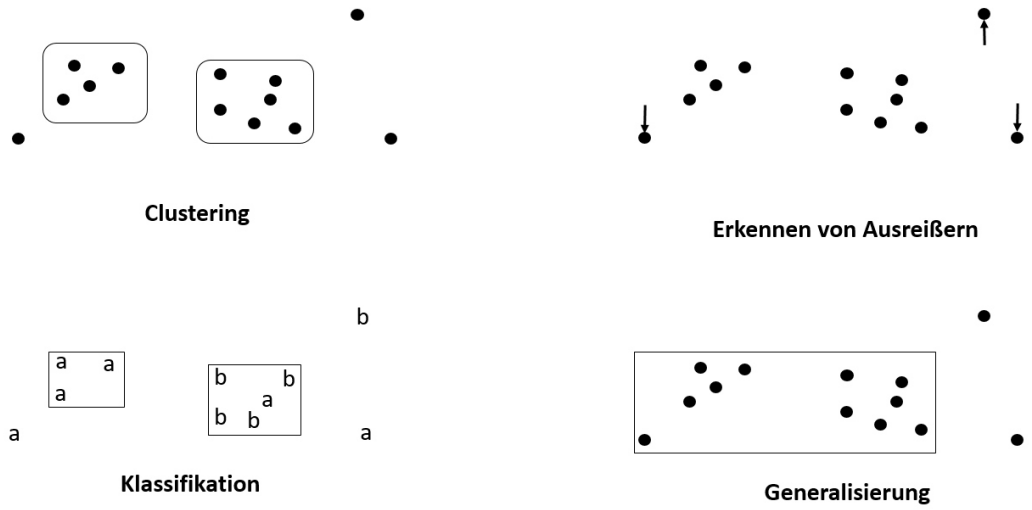

Abbildung 2: Die wichtigsten Data-Mining-Aufgaben, nach [9 S.5]

Data dredging

Data dredging, englisch für Datenbaggern, bezeichnet die fehlerhafte Anwendung von Data-Mining. Wer lang genug in Daten sucht, findet Muster die statistisch relevant erscheinen, es aber nicht sind [8]. Aus diesem Grund ist es wichtig die verwendeten Algorithmen mit Sorgfalt auszuwählen und die Data-Mining-Ergebnisse zu hinterfragen.

2.1.5 Evaluation

Als letzter Schritt der KDD muss das gefundene Muster im Hinblick auf die definierten Ziele überprüft werden [9]. Sollten diese Ziele nicht erreicht sein, kann eine neue Iteration der KDD initiiert werden. Diese neue Iteration kann bei jedem beliebigen Teil der Schrittkette starten. So ist es zum Beispiel möglich, die Algorithmen des Data-Mining mit veränderten Parametern noch einmal durchzuführen oder die Vorverarbeitung zu verfeinern. Zur Überprüfung können bekannte Stichproben mit dem Muster verglichen werden. Umso weniger Abweichungen zwischen den bekannten Proben und dem erkannten Muster existieren, umso besser ist das Ergebnis des KDD-Prozesses. Ist die Evaluation erfolgreich, kann das gefundene Wissen dokumentiert und in das System integriert werden. So dient es als Ausgangspunkt für neue KDD-Prozesse.

2.2 Clusterbildung

In Kapitel 2.1 wurden die verschiedenen Verfahren des Data-Mining und das Ziel des Clustering bereits vorgestellt. In diesem Abschnitt soll das Clustering, auf Deutsch Clusterbildung, genauer betrachtet werden. Hierfür werden erst die Begriffe Cluster und Distanz erklärt, bevor die häufig verwendeten partitionierenden Clusterverfahren k-Means und DBSCAN vorgestellt werden.

2.2.1 Cluster

Laut Quelle [9] enthält ein Cluster einen oder mehrere Datensätze. Für jeden einzelnen Datensatz des Clusters wird die Distanz zu allen anderen Datensätzen berechnet. Das Clusterzentrum ist der Datensatz mit der kleinsten Distanz zu den restlichen Datensätzen des Clusters. Damit ein neuer Datensatz einem Cluster hinzugefügt wird, wird die Distanz zum aktuellen Clusterzentrum berechnet. Unterschreitet die Distanz einen festzulegenden Schwellenwert, wird der Datensatz hinzugefügt und das Clusterzentrum neu berechnet.

2.2.2 Distanz

Die Distanz zweier Datensätze definiert ihre (Un-)Ähnlichkeit. Ihre Berechnung hängt stark vom Datentyp der einzelnen Datensätze ab. Bei numerischen Sensordaten kann zum Beispiel die nachfolgend erklärte Euklidische Distanz berechnet werden. Die Verfahren für andere Datentypen, beispielsweise Texte und endliche Mengen, werden in [9] beschrieben. Bei endlichen Mengen kann zum Beispiel die Distanz durch die Anzahl der überlappenden Elemente bestimmt werden. (1) zeigt die Euklidischen Distanzfunktion $dist(x, y)$ für die Datensätze $x = (x_1, \dots, x_d)$ mit numerischen Datenwerten x_i.

$$dist(x,y) \quad = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} \quad (1)$$

2.2.3 k-Means

Das k-Means-Verfahren [9] teilt die Datensätze in k-Cluster auf. Die k-Clusterzentren werden hierfür im ersten Durchlauf zufällig verteilt. Anschließend werden die Datensätze mithilfe einer Distanzfunktion den Clusterzentren zugeordnet, zu welchen die Distanz am geringsten ist. Danach werden die Clusterzentren neu berechnet, indem der Schwerpunkt des jeweiligen Clusters gebildet wird. Nun kann die Zuteilung der Datensätze erneut erfolgen. Dies geschieht bis zu dem Zeitpunkt, an dem entweder die Anzahl der gewünschten Wiederholungen erreicht ist oder sich die Clusterzentren nicht mehr verändern. Nicht mehr verändern kann auch bedeuten, dass die Veränderung einen Schwellenwert unterschreitet. Die einzelnen Schritte und ihre Auswirkungen werden in Abbildung 3 für k=2 verdeutlicht. Step 1, 3 und 5 entsprechen dabei dem Zuweisen der Datensätze zu einem Clusterzentrum, Step 2, 4 und 6 der Neuberechnung des Clusterzentrums.

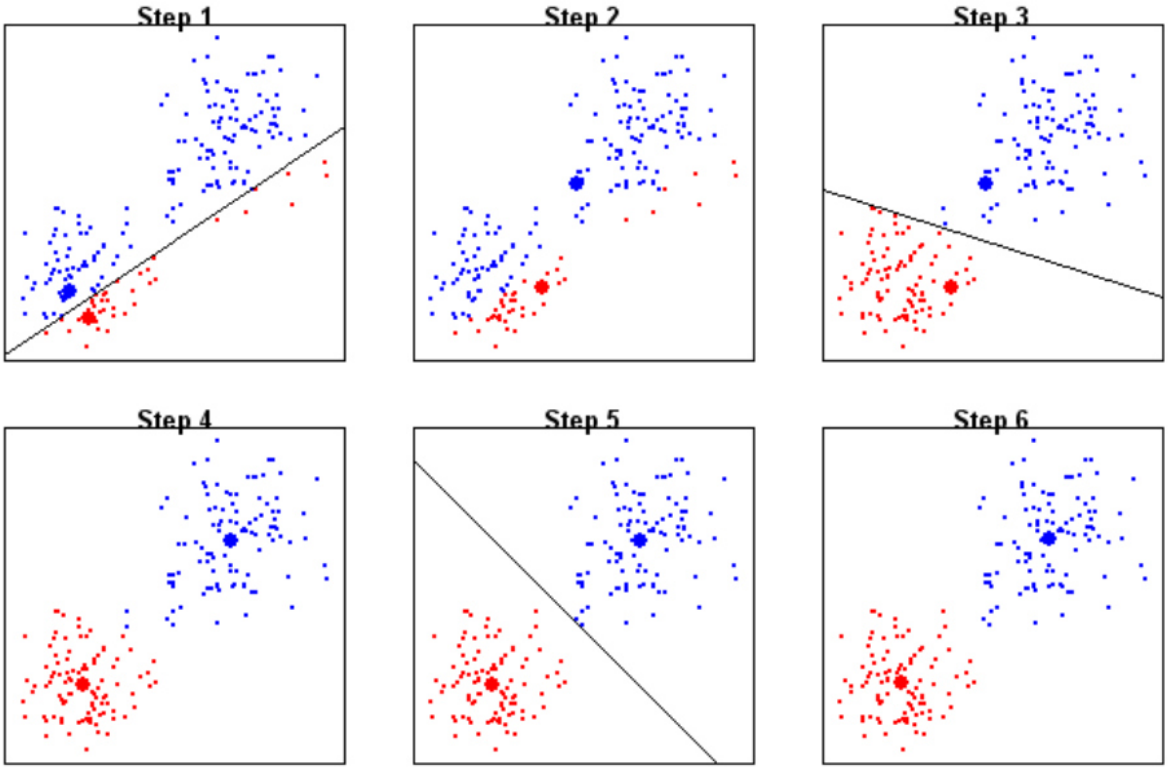

Abbildung 3: k-Means Schritte, aus [10]

2.2.4 DBSCAN

Das DBSCAN-Verfahren (Density-Based Spatial Clustering of Applications with Noise, in etwa dichtebasierte räumliche Clusteranalyse mit Rauschen) [9] teilt Datensätze in drei Kategorien ein: Kernobjekte, dichte-erreichbare Objekte und Rauschobjekte. Dies geschieht mit Hilfe von ε, der maximalen Distanz zwischen den Objekten damit das Objekt noch als erreichbar gilt, und $minPts$, der Anzahl der mindestens erreichbaren Nachbarn damit ein Objekt als dicht, also als Kernobjekt gilt.

Zusammenhängende Kernobjekte bilden ein Cluster. Hat ein Objekt nicht genügend erreichbare Kernobjekte gilt es als dichteerreichbares Objekt. Rauschobjekte sind Objekte welche weder dicht noch dichteerreichbare Objekte sind.

Abbildung 4 illustriert das DBSCAN-Verfahren mit $minPts = 4$. Punkte bei A sind Kernobjekte. Punkte B und C sind dichteerreichbare Objekte und gehören so auch zum gleichen Cluster. Punkt R ist durch keinen Nachbarn erreichbar und ist somit Rauschen.

Bei der Positionsbestimmung im dreidimensionalen Raum gibt es zum einen ein Koordinatensystem (KS) welches der Referenz dient, den Worldframe (WF). Zum anderen ein körpereigenes KS welches die Orientierung des Körpers darstellt, den Bodyframe (BF). Dies wird in Abbildung 5 verdeutlicht.

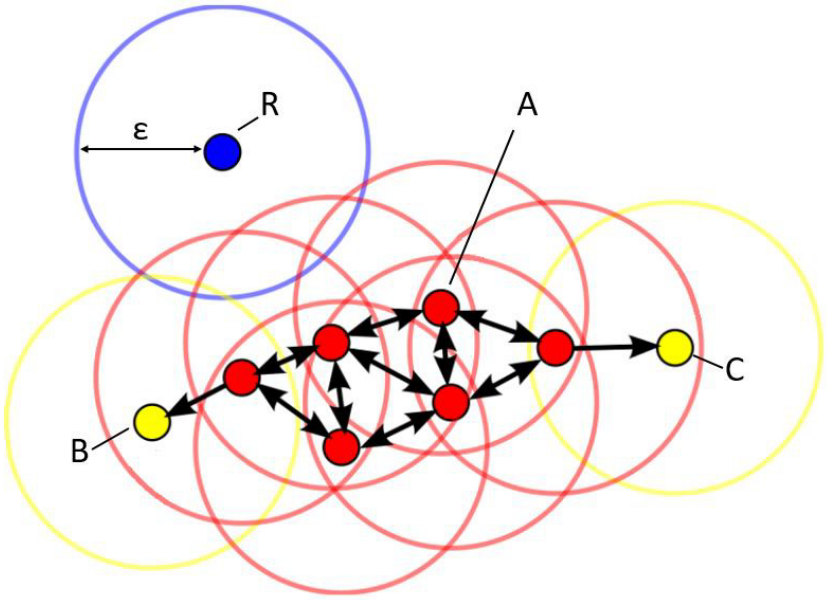

Abbildung 4: DBSCAN mit minPts=3 oder 4, nach [11]

2.3 Positionsbestimmung im dreidimensionalen Raum

Die Ausrichtung des Körpers und dessen Position in Bezug auf den WF wird durch eine Transformation beschrieben. Diese Transformation überführt das WF-KS in das BF-KS. Die Transformation enthält dabei Informationen über die Rotation und die Translation des BF in Bezug zum WF.

In dieser Arbeit ist vor allem die Ausrichtung der Tasse im Raum wichtig, weniger ihre Verschiebung in diesem. Aus diesem Grund wird nachfolgend auf die Rotation genauer eingegangen, die Translation nicht weiter betrachtet. Das Euler-Verfahren kann als Standardverfahren der Winkelerkennung angesehen werden und wird daher nur rudimentär vorgestellt. Die Winkeldarstellung in Quaternionen-Form wird etwas ausführlicher beschrieben. Zur umfangreicheren Einarbeitung empfehle ich einen Blick in [12] für das Euler-Verfahren oder [13] für Quaternionen.

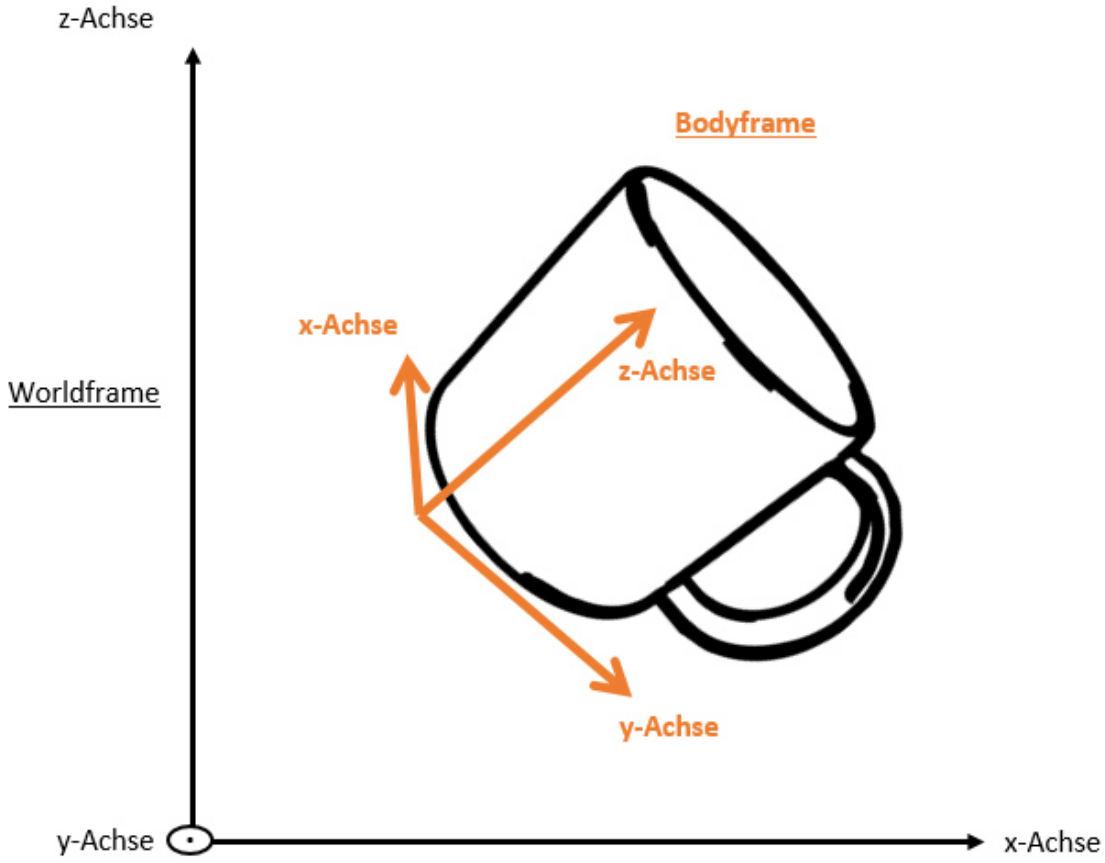

Abbildung 5: World- & Bodyframe

2.3.1 Eulerverfahren

Um eine beliebige Drehung im Eulerverfahren darzustellen, wird diese in drei Drehwinkel (Eulerwinkel) zerlegt. Diese drei Drehungen zeigt Tabelle 1.

Tabelle 1: Eulerverfahren, Winkel und Drehung

Winkelbezeichnung	Formelzeichen	Drehung um
Roll-Winkel	Φ	x-Achse
Pitch- / Nick-Winkel	Θ	y-Achse
Yaw-/ Gier-Winkel	Ψ	z-Achse

Drehsysteme

Damit die Rotationen eindeutig sind wurden verschieden Drehsysteme beziehungsweise Konventionen definiert, welche die Reihenfolge der vorgenommenen Rotationen bestimmt. Die Notation dieser Konventionen erfolgt mit Hilfe der Nennung der Achsenreinfolge um die gedreht wird. Zum Beispiel das zyx-Drehsystem, bei dem das KS wird zuerst um die z-Achse, danach um die einmal rotierte y-Achse und am Ende um die bereits zweimal rotierte x-Achse rotiert.

Für die Drehung $R(zyx, \psi, \theta, \phi)$ lässt sich folgende Rotationsmatrix (2) herleiten [12]:

$$R(zyx,\psi,\theta,\phi) = \begin{pmatrix} C\psi C\theta & C\psi S\theta S\phi - S\psi C\phi & C\psi S\theta C\phi + S\psi S\phi \\ S\psi C\theta & S\psi S\theta S\phi + C\psi C\phi & S\psi S\theta C\phi - C\psi S\phi \\ -S\theta & C\theta S\phi & C\theta C\phi \end{pmatrix} \quad \begin{matrix} mit\ C = cos \\ \\ mit\ S = sin \end{matrix} \quad (2)$$

2.3.2 Quaternionen

Quaternionen ($\mathbb{H}$) lassen sich mit imaginären Zahlen vergleichen. Der Unterschied ist, dass Quaternionen statt einer imaginären Zahlenkomponente (i), drei (i, j und k) besitzen. Dadurch erhält eine Quaternion einen vierdimensionalen Vektorraum, welcher aus einer skalaren Komponente und einem dreidimensionalen vektoriellen Imaginär-Teil besteht. Die üblichen Schreibweisen stehen in (3) bis (5).

$$q \equiv w + ix + jy + kz \quad ,w,x,y,z \in \mathbb{R} \quad (3)$$

$$\equiv [w,(x,y,z)] \quad ,w,x,y,z \in \mathbb{R} \quad (4)$$

$$\equiv [w,v] \quad ,w \in \mathbb{R}, v \in \mathbb{R}^3 \quad (5)$$

Das Verhalten der imaginären Komponenten ist denen der imaginären Zahlen ähnlich. Die Quadrate der Imaginär-Teile ergeben minus eins: $i^2 = j^2 = k^2 = -1$. Zusätzlich gibt es aufgrund der nicht kommutativen Multiplikation folgende Regeln: $i \cdot j = k,\ j \cdot k = i,\ k \cdot i = j$ und $j \cdot k = -k$, welche sich in $i \cdot j \cdot k = -1$ zusammenfassen lassen.

Rechenverfahren

Die Addition von zwei Quaternionen wird komponentenweise durchgeführt (6). Die Multiplikation zweier Quaternionen ist definiert durch (7). Dies ist eine Kombination aus Skalar und Kreuzprodukt. Die Definition des zu q konjugierten Quaternion q^* steht in (8). Der Betrag einer Quaternion ist die Wurzel aus der Summe der einzelnen Quadrate (9). Dabei gilt q als Einheitsquaternion ($\mathbb{H}_1$) wenn gilt: $\big||q|\big| = 1$. Die Inverse einer Quaternion steht in (10). Aus ihr und aus (8) wird ersichtlich, dass bei Einheitsquaternionen die Inverse und die Konjugierte gleich sind (11).

$$q + q' = w + w', v + v' \tag{6}$$

$$q \cdot q' = ww' - vv', wv' + w'v + v \times v' \tag{7}$$

$$q^* = [w, -v] \tag{8}$$

$$\big||q|\big| = \sqrt{w^2 + x^2 + y^2 + z^2} \tag{9}$$

$$q^{-1} = \frac{q^*}{||q||^2} \tag{10}$$

$$q^{-1} = q^* \qquad q \in \mathbb{H}_1 \tag{11}$$

Rotation

Für die Berechnung von Rotation im dreidimensionalen Raum ist die Polarform der Quaternionen (12) wichtig.

Im Falle einer Einheitsquaternion kann dies zu (13) vereinfacht werden, dabei ist n ein Vektor der Länge Eins.

Die eigentliche Rotation (R) erfolgt mit Hilfe von Quaternionen-Multiplikation und ermöglicht eine Drehung des Punktes (p) um die Achse des Vektors q mit dem Winkel von 2θ (14). Dies wird in [13] bewiesen. Im Falle eines Einheitsvektors kann statt der Konjugierten auch die Inverse der Quaternion verwendet werden (15). Rotationen R_1 und R_2, welche durch die Quaternionen q_1 und q_2 dargestellt werden, können durch Multiplikation verkettet werden.

$$q = \big||q|\big| \cdot \cos(\theta) + i \cdot \sin(\theta) + j \cdot \sin(\theta) + k \cdot \sin(\theta) \tag{12}$$

$$q = [\cos(\theta), n \sin(\theta) \tag{13}$$

$$R = q \cdot p \cdot q^* \tag{14}$$

$$R_{12} = q_1 \cdot q_2 \tag{15}$$

Für ein Quaternion $q = [w, (x, y, z)]$ ergibt sich die Rotationsmatrix (16) [13].

$$R(q) = \begin{pmatrix} 1-2(y^2+z^2) & 2(xy-sz) & 2(xz+sy) \\ 2(xy+sz) & 1-2(x^2+z^2) & 2(yz-sx) \\ 2(xz-sy) & 2(yz+sx) & 1-2(x^2+y^2) \end{pmatrix} \quad (16)$$

3 Planung der Zustandserfassung

Die Zustände, welche im Rahmen dieser Arbeit bestimmt werden sollen, wurden als erstes definiert. Dafür wurden zum einen die Szenarien betrachtet, welche in der vorangegangen Arbeit [2] definiert wurden, zum anderen im Rahmen dieser Arbeit überlegt, welchen Situationen eine Tasse im Alltag jeden Tag aufs Neue ausgesetzt ist. Bevor diese Experimente geplant, beziehungsweise die Modelle implementiert werden konnten, wurde überlegt, was die einzelnen Zustände charakterisiert und wie diese Charakteristika erkannt werden können. Die daraus resultierenden Zustände erhielten eine Einteilung in Modell und Experiment, je nachdem ob ein Zustand durch ein Modell vereinfacht werden kann oder in einem Experiment erfasst werden muss.

3.1 Zustände

Joschka Sondhof macht sich in seiner Studienarbeit [2 S.35] Gedanken über Szenarien welche mithilfe seiner Tasse: Mate-Mug detektiert werden sollten. Ein Auszug derselben wurde in Tabelle 2 übernommen. Auf diese aufbauend wurden in dieser Arbeit Alltagszenarien und repräsentative Zustände gesucht. Die gewählten Zustände (Abbildung 6) werden abgeleitet aus Momentaufnahmen und Veränderungen der Sensorwerte. Sie enthalten Informationen über den Bewegungs- und Inhaltszustand sowie die Benutzungsintensität, dargestellt durch die Aktivität der Tasse.

Tabelle 2: Mögliche Szenarien

Szenarien
Füllstand erkennen
Temperatur der Flüssigkeit erkennen
Bewegung wahrnehmen
Anzahl der Füllungen erfassen
Trinkvorgang erkennen
Gestensteuerung
Umgebungshelligkeit z.B. für Abschalten wahrnehmen

Anschließend wurde überlegt welche Sensorwerte und Sensorwertveränderungen einzelne Zustände charakterisieren, wie die Zustände aufeinander wirken und wie diese Charakteristika und Wechselwirkungen am besten erkannt werden können. Je nachdem ob die Zustände vorhersagbar sind um sie in einem Modell abzubilden, oder ob zugehörige Sensorwerte in einem Experiment erfasst werden müssen. Daraus resultierte die Einteilung der Zustände in zwei Kategorien, „Experiment“ und „Modell“. In Abbildung 6 wird die Kategorie durch eine hinterlegte Wolke oder Viereck dargestellt.

So kann zum Beispiel der Füllstand der Tasse über den Kippwinkel des Trinkvorganges bestimmt werden, was sich in einem Modell beschreiben lässt. Wohingegen die vorhandenen Beschleunigungen welche beim Tragen auftreten, zum einen vom Füllstand der Tasse und zum anderen von der tragenden Person abhängen, und deswegen experimentell erfasst werden müssen. Vorhandene Modelle müssen gegebenenfalls durch Experimente validiert werden.

Die Aktivität ergibt sich aus der Häufigkeit der erkannten Bewegungen innerhalb eines definierten Zeitraumes. Zum einen kann durch sie festgestellt werden, wann die Tasse vergessen wurde, zum anderen kann die „Aktivität“ in Kombination mit den anderen Zuständen zur Identifikation komplexerer Szenarien dienen.

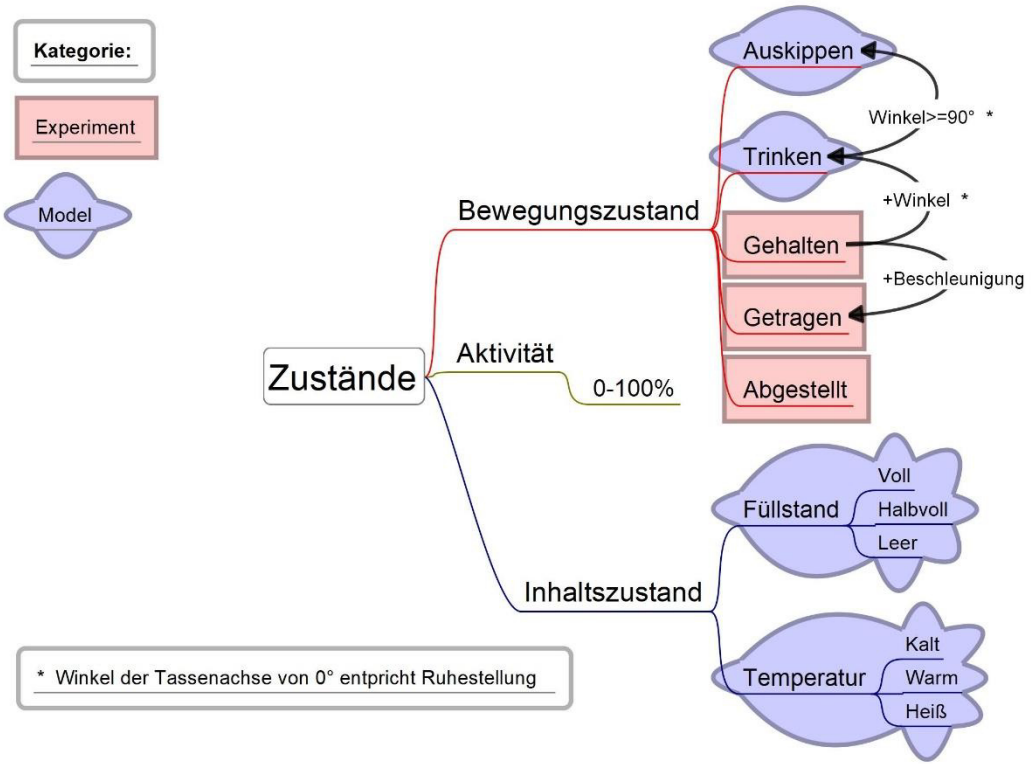

Abbildung 6: Gewählte Zustände

3.2 Zustandserkennung

Wie in 3.1 erwähnt, wurden Charakteristika der Zustände gesucht. Auf diese Charakteristika der Bewegungs- und Inhaltszustände soll nachfolgend genauer eingegangen werden.

3.2.1 Bewegungszustände

Die Bewegungszustände „Gehalten“ und „Abgestellt“ sollten nach der ersten Überlegung mit Hilfe des Gewichtssensors erkannt werden. Aufgrund der Weiterentwicklung der Tasse rückte jedoch die Erkennung der Zustände mithilfe des Beschleunigungssensors mehr und mehr in den Fokus. Der Nachteil des taktilen Gewichtsensors ist, dass dieser stets beweglichen Kontakt zum Untergrund haben muss um die Veränderung der wirkenden Kraft zu erkennen. Dieser bewegliche Kontakt zum Untergrund muss entweder feuchtigkeitsundurchlässig konstruieren sein, sonst ist er aufgrund der Beweglichkeit ein Schwachpunkt des Gehäuses. Hinzu kommt, dass ein taktiler Gewichtssensor im Vergleich zum Beschleunigungssensor sehr träge reagiert. Ein Beschleunigungssensor ist in der Lage, den beim Abstellen der Tasse entstehenden Schock zu erkennen. Das Aufnehmen der Tasse sollte durch die veränderte Beschleunigung auf der Z-Achse erkennbar sein und der Zustand „Gehalten“ aufgrund des bis jetzt ausgebliebenen Schocks beim Abstellen.

Der Zustand „Getragen“ ist auf das Erkennen des Geradeausgehens ausgelegt, denn im Alltag läuft der Mensch am häufigsten direkt auf sein Ziel zu. Dies sollte in Abhängigkeit von Person und Füllstand zu einem charakteristisch oszillierenden Muster in den Beschleunigungswerten der x- oder y-Achse führen (Abbildung 7), welches erkannt werden kann. Dieses charakteristische Muster entsteht durch die nicht lineare Geschwindigkeit mit der ein Mensch geht [14].

Mit Hilfe der IMU lässt sich der aktuelle Kippwinkel der Tasse bestimmen. Wird ein Kippwinkel detektiert, kann bis zu einem bestimmten Winkel von einem Trinkvorgang ausgegangen werden. Bei Überschreitung des Winkels wird die Tasse wahrscheinlich ausgekippt. Die Zustände „Trinken“ und „Auskippen“ haben eine Änderung des Füllstands zu folge. Der Füllstand wird im nächsten Abschnitt genauer betrachtet.

Tabelle 3 zeigt den Zusammenhang von Winkel, Zustand und Füllstand. Dieser Zusammenhang wird in Abbildung 8 illustriert.

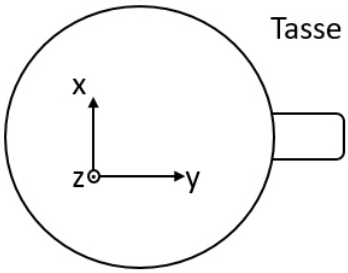

Abbildung 7: Tasse Draufsicht, Ausrichtung des BF

Tabelle 3: Zusammenhang Winkel-Zustand-Füllstand

Winkel	**Zustand**	**Füllstand**
<23°	Trinken	Voll
23°-45°		Halbvoll
46°-90°		Halbleer
>90°	Auskippen	Leer

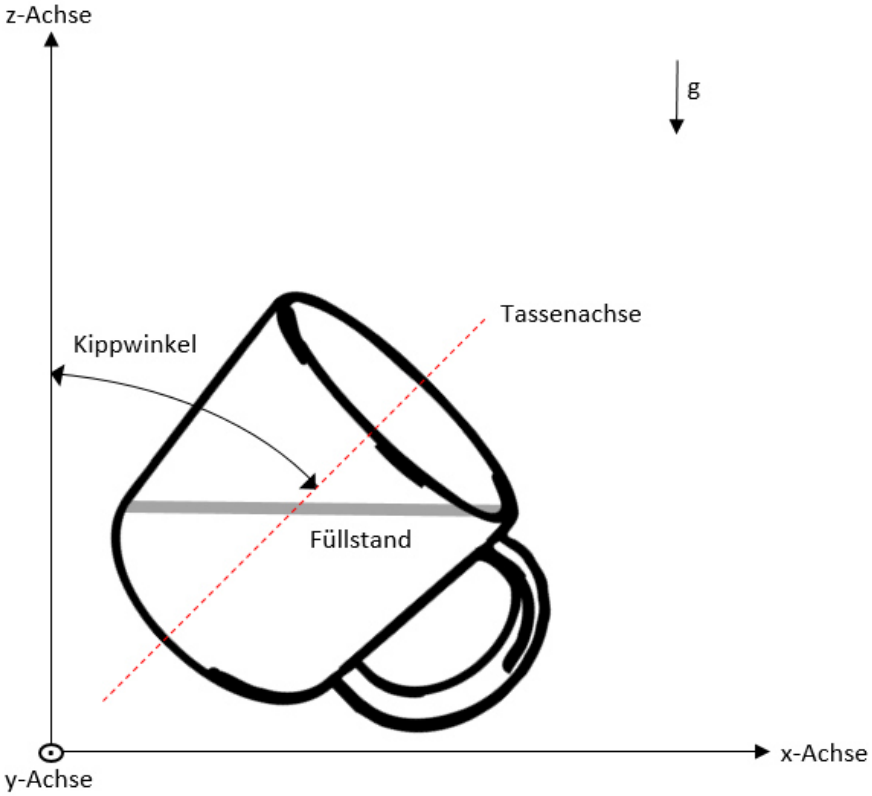

Abbildung 8: Illustration Tassen-Kippwinkel

3.2.2 Inhaltszustände

Herr Sondhoff sah für die Bestimmung des Füllstands den Gewichtssensor vor. So implementierte er in seiner Bibliothek (yMateMug_Library) eine Funktion, welche anhand der Form der Tasse und der Dichte der Flüssigkeit den aktuellen Füllstand berechnet. Im Unterkapitel 3.2.1 Bewegungszustände wurde bereits auf die Nachteile des Gewichtsensors sowie den Zusammenhang von Kippwinkel und Füllstand eingegangen. Tabelle 3 zeigt alle gängigen Entleerungswege der Tasse. Das Erkennen der erneuten Befüllung stellt sich etwas komplizierter da, wenn auf den Gewichtssensor verzichtet werden soll.

Wenn davon ausgegangen wird, dass eine Tasse nur mit Heiß- oder Kaltgetränken befüllt wird, kann das Befüllen aufgrund des Temperaturwechsels erkannt werden. Hat die eingefüllte Flüssigkeit in etwa Raumtemperatur, entfällt diese Methode. Dann kann das Befüllen und der neue Füllstand aber durch den beim nächsten Trinkvorgang erzeugten Kippwinkel erkannt werden. Im Internet lassen sich zur Zeit Tassen [15, 16] finden, welche versprechen, das Heißgetränk möglichst lange auf perfekter Temperatur zu halten. In diesen Fällen befindet sich die perfekte Trinktemperatur bei 50 bis 65 °C. Dies wurde bei der Auswahl der Temperaturbereiche berücksichtigt, siehe Tabelle 4.

Tabelle 4: Temperatur und Bereich

Temperatur	Bereich
<40°C	Kalt
40-60°C	Warm
>60°C	Heiß

3.2.3 Aktivität

Auf den Internetseiten [15, 16] befinden sich Temperatur-Zeit-Graphen. Sie zeigen, dass eine mit ~90°C heißer Flüssigkeit befüllte Keramiktasse nach circa 60 Minuten fast komplett ausgekühlt ist. Nach etwa 15 Minuten erreicht die Temperatur 60°C, den warmen Temperaturbereich (Tabelle 4). Daher und da sich 15 Minuten einfach zu vollen Stunden addieren lassen, wurde bei der Aktivität ein Bereich von 15 Minuten gewählt. Mit der Konsequenz, dass die Aktivität von 100% nach 15 Minuten ohne erkannte Zustandsänderung bei 0% ist (Tabelle 5).

Tabelle 5: Aktivität

Prozent	Zeit
1 %	9 sec
0% nach	15 min / 900 sec

3.3 Versuchsplanung

Nachdem im vorherigen Kapitel die Charakteristik der Zustände bestimmt wurde, kann nun die Datenverarbeitung gewählt und der Ablauf der Experimente geplant werden.

3.3.1 Vorüberlegungen

Auch wenn die Tasse als kabelloses Objekt konzeptioniert wurde und die kabellose Kommunikation als finales Ziel angesehen wird, wurde im Rahmen dieser Arbeit auf die Verbindung per Serial-Port zurückgegriffen. Dies hat den Vorteil, dass die Kommunikation in Mikrocontrollern zum Standard gehört und Fehler bei der Einbindung von zusätzlichen Bibliotheken beziehungsweise der Einrichtung von Netzwerken ausgeschlossen werden können. Außerdem ist so die Spannungsversorgung der Tasse stets gesichert und es muss keine Rücksicht auf die Akkulaufzeit von derzeit circa drei Stunden [2] genommen werden.
Dies ist sowohl bei der Wiederholung von Experimenten vorteilhaft als auch bei später geplanten Langzeittests. Die durch das USB-Kabel beeinträchtigte Dynamik der Tasse kann minimiert werden, wenn das USB-Kabel über den Arm der Versuchsperson und den Tassenhenkel geführt wird. Trotzdem muss die veränderte Dynamik unter Umständen zu einem späteren Zeitpunkt berücksichtigt und gegebenenfalls eliminiert werden. Das Mitführen eines Laptops ist zu Zeiten immer kompakter werdender Technik ein zu vernachlässigender Umstand.

Alle gesendeten Daten sollen im CVS-Format in einer Textdatei gespeichert werden. Für eine bessere Dokumentation und vereinfachte Auswertung der Daten werden die Experimente per Foto oder Video dokumentiert. Damit ein Synchronisieren zwischen Sensordaten und Video möglich ist, muss ein Synchronisationssignal, ähnlich der Filmklappe, implementiert werden. In allen Experimenten befindet sich der Henkel der Tasse über der Querstrebe des Bodenträgers auf welchem sich die IMU befindet. Außerdem ist der Henkel auf der Seite des USB-Kabels. Dadurch wird sichergestellt, dass die Ausrichtung der Tasse und der Sensoren in allen Versuchen identisch bleibt und Abbildung 7 entspricht.

3.3.2 Programm Datacollect

Das Programm Datacollect wird auf den Mikrocontroller der Tasse geladen. In jedem Programmzyklus aktualisiert es die Sensordaten und sendet diese per Serial-Port an den angeschlossenen Computer.

Zu Beginn wartet das Programm in einer while-Schleife darauf, dass der Lichtsensor Helligkeit ausgesetzt wird und sendet bis dahin ein Wartesignal. Dies ermöglicht ein Synchronisieren von Video- und Datenaufnahme, da sich der Zeitpunkt des ersten gesendeten Datensatzes und die Entfernung der Bodenabdeckung genau feststellen lassen.

Wenn bei Programmstart der Boden der Tasse abgedeckt ist, können die restlichen Versuchsvorbereitungen in Ruhe getroffen werden. Wird zu einem beliebigen Zeitpunkt die Abdeckung entfernt, startet das Senden der Daten. Das Setzten eines Merkers verhindert die versehentliche Rückkehr in die while-Schleife.

Da sich das wiederholte Aufrufen der Serial.print-Funktion negativ auf die Prozesszeit auswirken kann, werden die Daten der Sensoren in einem String gespeichert und nicht einzeln per Serial-Port gesendet. Dabei sind die einzelnen Sensordaten mit einem Semikolon getrennt. Bevor der aktuelle Sensordaten-String übermittelt wird, wird ein interner Zeitstempel gesetzt. Dieser Zeitstempel besteht aus den seit Programmstart vergangenen Millisekunden.

Um im Versuchsablauf mögliche Probleme schneller zu erkennen, wurde ein visuelles Feedback programmiert. Solange sich das Programm in der Schleife der Sensordatenübermittlung befindet, blinken die LEDs Rot.

3.3.3 Tragen

Ziel des Experiments Tragen ist es, in den aufgenommenen Daten Merkmale des geradlinigen Gehens mit einer Tasse zu erkennen.

Die Art und Weise wie eine Tasse getragen wird, hängt stark vom Füllstand der Tasse und ihrem Träger ab. Bei der Beobachtung von Studenten in der Cafeteria der HAW Hamburg im Berliner Tor 7 fiel auf, dass die volle Tasse mehrheitlich auf Bauch- bis Brusthöhe vor dem Körper getragen wurde.

Daher wurde der Focus auf die Erkennung dieses speziellen Szenarios gelegt, das Tragen der Tasse vor dem Körper auf Brusthöhe. Außerdem wurde die Tasse stets mit der rechten Hand am Henkel angefasst. Diese immer gleiche Haltung der Tasse vereinfacht die Identifikation von Mustern. Auf der anderen Seite führen diese Einschränkungen dazu, dass andere Formen des Tassentragens wie zum Beispiel das lose Tragen der leeren Tasse am Henkel in einem separaten Versuch erfasst werden müssen.

Das Experiment selbst besteht darin, die Tasse in oben beschriebener Position zu halten und nach dem Starten des Messlaufs fünf Sekunden lang mit gemäßigtem Schritt geradeaus zu

gehen. Dadurch sollte eine wechselnde oszillierende Beschleunigung in Gehrichtung erkennbar sein.

Wenn erste Muster erkannt wurden, kann die Haltung der Tasse variiert werden. So sollte sich die zuvor beschriebene Beschleunigung unabhängig von der Griffposition in anderen Achsen wiederfinden lassen.

Durch Figure 3 in [14] lässt sich die zu erwartende Beschleunigung im Gehen im Vorfeld abschätzen. Die „stable phase“ zeigt eine maximale Veränderung der Geschwindigkeit von circa $v_1 = 1{,}45\frac{m}{s}$ bei etwa $t_1 = 3{,}5\ s$. zu ungefähr $v_2 = 0{,}6\frac{m}{s}$ bei rund $t_2 = 4{,}25\ s$. Hieraus lässt sich eine Verzögerung von $a = -0{,}13g$ berechnen (17).

$$a = \frac{\Delta v}{\Delta t} = \frac{-0{,}85\frac{m}{s}}{0{,}75s} = -1{,}3\frac{m}{s^2} = -0{,}13g \quad (17)$$

3.3.4 Abstellen

Das Abstellen soll durch Erkennen des Abstell-Schocks identifiziert werden. Das dies möglich ist zeigt ein erfolgreicher Test, welcher ohne Spezifikation, mit Hilfe eines Android Smartphones und der App Vibrometer durchgeführt wurde. Dieser Test kann aber mit Hilfe eines beliebigen Smartphones und einer der zahlreich verfügbaren Seismographen-Apps wiederholt werden. Die Tasse soll wiederholt aufgenommen und abgestellt werden, wobei die Art der Haltung variieren sollte um allgemein gültige Muster zu erkennen. Ansonsten kann es dazu kommen, dass nur der spezifische Fall „Tasse am Henkel anfassen und abstellen“ erkannt wird. Damit die Tasse nicht ungewohnt hart aufgesetzt wird, wird das Experiment mit einer gefüllten Tasse durchgeführt.

3.3.5 Aufnehmen

Da Aufnehmen das Gegenstück zu Abstellen ist, kann das Muster „Aufnehmen“ auch mit dem in 3.3.4 beschreiben Experiment untersucht werden. Wenn ich eine Tasse von der Tischplatte zum Trinken anhebe, hebe ich sie um etwa 50 cm und tue dies in etwa einer Sekunde. Unter der Annahme, dass dies eine gleichmäßig beschleunigte Bewegung ist, kann ein erwarteter Beschleunigungswert von $0{,}1g$ approximiert werden (18).

$$a = 2{\cdot}s{\cdot}t^2 = 2{\cdot}0{,}5{\cdot}1^2 = 1\frac{m}{s^2} = 0{,}1g \quad (18)$$

3.4 Modellentwicklung und -implementierung

Im folgenden Kapitel sollen zuerst die Modellentwicklung und die dafür nötigen Gedankengänge betrachtet werden, bevor auf die Umsetzung der entwickelten Modelle eingegangen wird.

3.4.1 Vorüberlegungen

Die implementierten Modelle sollen in die vorhandene Programmstruktur „mateMugSys“, welche durch Joschka Sondhof [2] vorgegeben wurde, integriert werden.
Aufgrund der begrenzten Rechenleistung des Arduino muss auf eine ressourcenschonende Implementierung geachtet werden.
Geschieht dies nicht, besteht die Gefahr, dass durch zu lange Bearbeitung beziehungsweise Berechnung eines Modells, der Programmfluss negativ beeinträchtigt wird. Zu diesen Beeinträchtigungen könnte das Verschleppen von Sensorwerten zählen. Dies würde zu einer Fehlinterpretation der Experimente und Fehlern in den Modellen führen.
Wie in 3.2 beschrieben hängen die Zustände „Trinken“, „Auskippen“ und „Füllstand“ alle mit dem Kippwinkel der Tasse zusammen. Hinzu kommt, dass der Kippwinkel aufgrund der vielen Einflussgrößen und seiner Dynamik komplizierter zu programmieren ist, als zum Beispiel das Modell der Temperaturänderung. Aus diesen Gründen wurde die Implementierung der Winkelerkennung priorisiert.

3.4.2 Winkel-Erkennung

Der Neigungswinkel der Tasse soll unabhängig von Griffposition und Kipprichtung ressourcenschonend erkannt werden.

Gewähltes Verfahren

In Kapitel 2.3 wurden zwei gängige Winkelerkennungsverfahren für den dreidimensionalen Raum vorgestellt. Für die Winkel-Erkennung in dieser Arbeit wurde das Verfahren der Quaternionen verwendet. Der Vorteil der Quaternionen liegt vor allem darin, dass die Konkatenation (Verkettung) von Rotationen und somit die Transformation zwischen BF und WF effizienter ist.

„Statt der bei 3x3-Matritzen notwendigen 27 Multiplikationen und 18 Additionen lässt sich der Rechenaufwand bei Quaternionen mit dem entsprechenden Algorithmus auf 8 Multiplikationen und 4 Divisionen beschränken“ [13 S.11].

Hinzu kommt die Eindeutigkeit der Rotationen und das daraus resultierende Wegfallen der Konventionen. Außerdem erfolgt die Rotation direkt über die gewünschte Drehachse und das

Problem des Gimbal Lock entfällt. Gimbal Lock beschreibt das Problem, dass bei dem Eulerverfahren zwei Achsen zusammenfallen können.
Ein bekannter Nachteil von Quaternionen ist, dass sie keine Informationen über die Translation enthalten [13]. Dieser Nachteil entfällt aufgrund der Problemstellung.
Um die Rechenzeit zu verbessern wurde zur Quaternion-Integration die Kleinwinkelnäherung verwendet, wodurch auf aufwendige Sinus-, Cosinus- und Wurzelfunktionen verzichtet werden konnte. Das gewählte Integrationsverfahren benötigt nur acht Multiplikationen, drei Additionen und drei Subtraktionen und wird in [17] ausführlich beschrieben.

Filter

Da eine Tasse vor allem stationär am Arbeits- beziehungsweise Essplatz genutzt wird, sie dabei mehrfach abgesetzt wird und die Tasse im Allgemeinen nicht über einen langen Zeitraum ununterbrochen in Bewegung ist, wurde von der Implementierung eines komplexen Kalman- beziehungsweise Mahony-/Madgwick-Filters abgesehen.
Der nach [18] implementierte einfache Komplementärfilter verzichtet auf eine Yaw-Drift-Kompensation, da das Magnetometer nicht in den Filter einbezogen wird. Dies wirkt sich zugunsten der benötigten Ressourcen aus.

Programmintegration

Das Modell wurde in die vorhandene Programmstruktur integriert indem der Klasse sensorIMU (Abbildung 9) die nötigen Funktionen (Tabelle 6) hinzugefügt wurden.

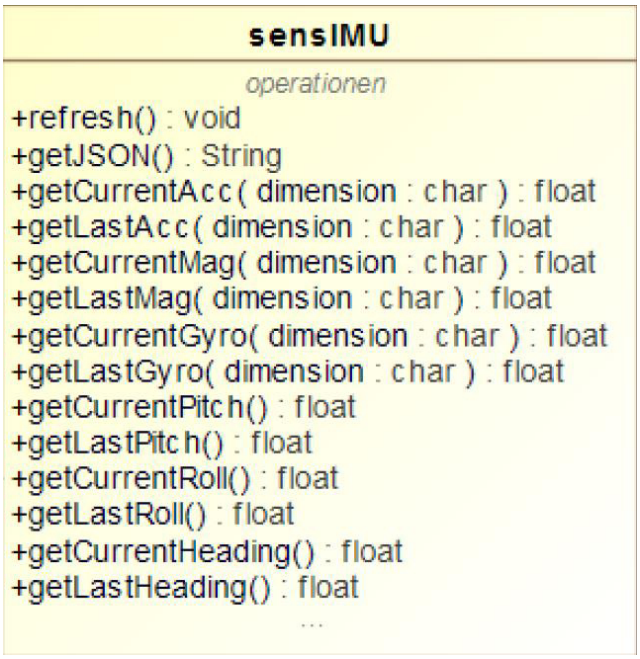

Abbildung 9: Funktionen der Klasse sensorIMU, aus [2 S.21]

Tabelle 6: Integrierte Funktionen zur Winkelerkennung

Operationen	**Liefert**
+getCurrentQuatVec(dimension : char) : float	den vektoriellen Imaginär-Teil der aktuellen/letzten Positions-Quaternions
+getLastQuatVec(dimension : char) : float	
+getCurrentQuatW() : float	skalare Komponente der aktuellen/letzten Positions-Quaternions
+getLastQuatW() : float	
+getCurrentRotAngle() : float	aktuellen/letzten Winkel zwischen Tassen-Achse und xy-Ebene
+getLastRotAngle() : float	

Da das JSON-Format in dieser Arbeit nicht genutzt wurde, wurde auf eine Integration der neu implementierten Funktionen in die getJSON()-Funktion verzichtet.
Die skalare Komponente bei der Winkelerkennung nicht von Belang und wurden zur Vollständigkeit halber erst im Programm Datacollect2_1 implementiert.

Richtungsbestimmung

Nachdem die Sensordaten aktualisiert wurden, wird die Drehrate in rad/s umgewandelt und gemeinsam mit der Beschleunigung, gemessen in g, in Vektorform transformiert.
Anschließend müssen im Bezug auf den BF aufgenommene Sensordaten in den WF rotiert werden. Dies geschieht durch Konkatenation der Quaternion, welche die aktuelle erwartete Richtung enthält, mit dem Vektor der aktuellen Beschleunigungswerte.
Indem das Kreuzprodukt des rotierten Beschleunigungsvektors mit einem perfekten vertikalen Vektor (0, 0, 1) berechnet wird, kann der Sensorfehler geschätzt werden. Dieser geschätzte Fehler berücksichtigt keine Abweichungen, welche Aufgrund nicht-gravitiativen Beschleunigungen, wie Vibrationen, hervorgerufen wurden. Anschließend muss der Fehlervektor wieder zurück in den BF rotiert werden. Dies geschieht wieder durch Konkatenation mit dem Quaternion welches die erwartete Richtung enthält. Um die Sensordaten zu berichtigen werden der Fehlervektor und der Vektor der aktuellen Sensorwerte addiert. Im Anschluss daran wird mit Hilfe der Kleinwinkelnäherung über Δt (Zeit pro Programmdurchlauf) ein inkrementeller Rotationsquaternion erstellt. Hierdurch haben kurzzeitige Störungen einen minimalen Einfluss auf die erwartete Richtung, während Kräfte wie Gravitation auf lange Sicht gegen Sensordrift und Messfehler wirken.
Die so erzeugte Rotationsquaternion wird mit der Quaternion der aktuellen erwarteten Richtung konkateniert und ergibt die erwartete Richtung für den nächsten Programmdurchlauf vor.

Winkelberechnung

Die Oberfläche von Flüssigkeiten richtet sich ohne Aufbringung von Radialkräften, aufgrund der Erdanziehungskraft, im Ruhezustand immer parallel zur xy-Ebene des WF aus.
Daher ist es ausreichend den Schnittwinkel zwischen Normalvektor der xy-Ebene und Rotationsachse der Tasse zu ermitteln. Der Winkel der Tasse um ihre Achse (w) ist hierfür unwichtig.
Um den Schnittwinkel zu berechnen wird zuerst aus der Quaternion mit der aktuell erwarteten Richtung der Rotationsvektor gebildet.
Danach erfolgt die Berechnung der nötigen Komponenten um den Schnittwinkel zu berechnen (19). Der Betrag des Skalarprodukts aus Rotationsvektor $\vec{r}$ und Normalvektor $\vec{n}$ wird durch das Produkt der Beträge geteilt und ergibt so $\sin \propto$.

$$\propto \quad = \sin^{-1}\left(\frac{|\vec{r} \cdot \vec{n}|}{|\vec{r}| \cdot |\vec{n}|}\right) \quad (19)$$

4 Experimente zur Zustandsidentifikation

Nachdem in Kapitel 3 die Zustände sowie die durchzuführenden Experimente und deren Ziele definiert wurden, kann nun die praktische Umsetzung angegangen werden. Eine Validierung der Sensordaten wurde in der vorangegangenen Arbeit [2] nicht durchgeführt. Deswegen müssen die gesendeten Sensorwerte überprüft werden.
Anschließend werden die nötigen Versuchsvorbereitungen beschrieben, bevor die Versuche selbst und deren Ergebnisse erörtert werden.
Bereits bei ersten Tests des Datacollect-Programms (3.3.2) wurde eine geringe Datenrate von drei Hertz festgestellt. Weil zu erwarten war, dass aus der geringen Datenrate Abtastprobleme resultieren, wurden umgehend Lösungen erarbeitet. In Kapitel 5 wird darauf genauer eingegangen.

4.1 Datenvalidierung

Joschka Sondhof führte zwar für alle Sensoren erfolgreiche Funktionstests durch, allerdings teilte er mir im Gespräch mit, dass die Validierung der einzelnen Sensorwerte den zeitlichen Rahmen seiner Arbeit gesprengt hätte.
Um sinnvolle Werte bei den Experimenten zu gewährleisten, erfolgte eine Datenüberprüfung für jeden einzelnen Sensor. Dies geschah zum Beispiel durch Nachmessen mit externen Prüfmitteln wie dem Thermometer, Lehren, zum Beispiel einem Winkelmesser und, kritischer Beobachtung.

Folgende Auffälligkeiten wurden festgestellt:

A) Der Gewichtsensor liefert mit Hilfe der Weight_we Funktion Zahlenwerte (Tabelle 7), welche nicht dem Gewicht in mg entsprechen, sich aber erwartungsgemäß ändern. Die restlichen Funktionen für das Volumen der enthaltenen Flüssigkeit und den Füllstand der Tasse liefern keine sinnvollen Daten.
B) Die IMU gibt für Pitch und Roll den gleichen Sensorwert aus.

Tabelle 7: Zahlenwerte der Weight_we Gewichtssensorfunktion

Tassenzustand	Zahlenwerte
Gehalten	$-9 \cdot 10^5\ bis\ -14 \cdot 10^5$
Leer	$-3{,}4 \cdot 10^5\ bis\ -2 \cdot 10^5$
Voll	$2 \cdot 10^5\ bis\ 3 \cdot 10^5$

Da die Zahlenwerte im Fall A, falls nötig Auskunft über das momentane Gewicht der Tasse geben können und in Abschnitt 3.2.1 bereits auf die Nachteile des Gewichtssensors eingegangen wurde, wird das fehlerhafte Verhalten des Gewichtssensors im Rahmen dieser Arbeit nicht berichtigt.

Für Fehler B konnte das falsche Zuweisen von Sensorwerten in der implementierten Updateschleife identifiziert werden. Um den Fehler zu finden erfolgte zuerst eine Funktionsüberprüfung des Sensors mit Hilfe von Beispielsoftware. Anschließend wurde die Implementierung und die darin enthaltenen Berechnungen überprüft. Der Fehler konnte in der Updateschleife gefunden und berichtigt werden.

4.2 Versuchsvorbereitung

Zu Beginn jedes Versuchs musste der Boden der Tasse mit Hilfe eines Pappdeckels abgedeckt werden um den Lichtsensor vor Helligkeit zu schützen. Das Hochladen des Programms Datacollect2_0 auf den Mikrocontroller der Tasse erfolgte mit Hilfe der Arduino IDE (Abschnitt 7.2.1). Zum Speichern der über den Serial-Port empfangenen Daten wurde das Programm „Real Term: Serial Capture Program“ (Abschnitt 7.2.2) verwendet.

Vor jedem Versuch erfolgte die Ausrichtung einer Videokamera auf die Teststrecke beziehungsweise auf die Tasse, inklusive manueller Auslösung der Aufnahme.

4.3 Versuchsdurchführung

Die Ergebnisse wurden direkt nach jedem Experiment kurz evaluiert, gegebenenfalls in ihrer Durchführung verändert und wiederholt. Dies wird im Rahmen der jeweiligen Experimente genauer beschrieben. Die ausführliche Evaluation erfolgt in Abschnitt 4.4.3.

Das aufgezeichnete Video sowie die Rohdaten erhielten einen eindeutigen Dateinamen in folgender Form: Y=Jahr, M=Monat, D=Tag, h=Stunde, m=Minute, YYYYMMDDhhmm_Versuchsversion&Name. Hier ein Beispiel für das erste Experiment „Abstellen“: 201708151010_1AbstellenVoll.

4.3.1 Abstellen Voll

Wie in 3.3.4 und 3.3.5 beschrieben soll der Schock des Abstellens und die Beschleunigung beim Aufnehmen erkannt werden.

201708151010_1AbstellenVoll

Die mit Wasser gefüllte Tasse wurde siebenmal abgestellt und aufgenommen. Die Tasse wurde dabei unterschiedlich gehalten und auf eine Höhe von etwa zehn Zentimeter angehoben.

Evaluation

Anders als in Abschnitt 3.3.5 beschrieben wurde die Tasse nicht auf 50 cm angehoben. Die Wiederholung der Bewegung führte zu einem unübersichtlichen Datenbild im Plot „Beschleunigung z-Achse und Zeit“ in dem kein expliziter Schock zu erkennen ist (siehe Abbildung 10). Aus diesem Grund wurde das Experiment 4.3.2 durchgeführt.

4.3.2 Aufnehmen

Ziel des Experiments „Aufnehmen“ ist es, anhand der Sensorwerte den Zustandswechsel von „Abgestellt“ zu „Gehalten“ zu erkennen.

201708161315_1Aufnhemen

Die Tasse wurde für zehn Sekunden auf dem Tisch abgestellt, um ausreichend Messwerten für den Zustand „Abgestellt“ zu liefern.
Anschließend wurde die Tasse ohne explizite Haltung gegriffen und um circa 50 cm angehoben. Die Tasse war bei der Versuchsdurchführung leer.

Evaluation

Der Zustandswechsel ist im Plot „Beschleunigung z-Achse und Zeit“ zu erkennen.

4.3.3 Tragen

Die wechselnde Beschleunigung beim Gehen soll wie in 3.3.3 beschrieben mit diesem Experiment erfasst werden.

201708151450_1Tragen

Die Tasse wurde wie beschrieben mit der rechten Hand am Henkel im leeren Zustand gehalten. Nach dem Messstart wurde direkt mit dem Gehen begonnen. Für acht Sekunden wurde geradeaus gegangen, bevor die Messwertaufnahme beendet wurde.

Evaluation

Bei diesem Versuch wurde die Kamera falsch ausgerichtet, wodurch das Synchronisationssignal nicht von der Kamera erfasst werden konnte. Des Weiteren konnte im Plot „Beschleunigung xy-Achse und Zeit“ kein deutliches Muster erkannt werden.

201708161635_2Tragen

In diesem Versuch wurde direkt in Richtung Kamera gegangen und die Tasse mit der rechten Hand am Henkel, im leeren Zustand, gehalten.
Um deutliche Pole im Plot zu erzeugen erfolgte nach dem Messstart ein Stillstand von zehn Sekunden. Darauf folgte eine Gangphase von fünf Sekunden.

Evaluation

Da bei Messstart der geplante Stillstand kurzzeitig vergessen wurde und in den Rohdaten ein Fehler gefunden wurde, erfolgte eine Wiederholung des Versuchs. Der Fehler ereignete sich nach zwei erfolgreich gesendeten Datensätzen und zeigte sich darin, dass erneut Platzhalter gesendet wurde.

201708161650_2Tragen

Die Tasse wurde wie im vorherigen Experiment gehalten. Das Experiment bestand aus einem Stillstand von 25 Sekunden, gefolgt von zehn Sekunden Gehen.

Evaluation

Der Zustandswechsel ist im Plot „Beschleunigung x-Achse und Zeit“ zu erkennen.

4.4 Datenverarbeitung

Bevor auf die Visualisierung der in Kapitel 4.3 gesammelten Daten eingegangen wird, erfolgt eine Betrachtung der angewendeten Verarbeitungsschritte der Daten.

4.4.1 KDD

In Kapitel 2.1 wurden die einzelnen Schritte der KDD erklärt. Aus diesem Grund wird hier nur deren Anwendung beschrieben.

Fokussieren

Die Sensordaten der Tassen werden in numerischer Form in einer Textdatei zwischengespeichert. Aus dieser Textdatei können dann je nach betrachtetem Szenario beliebige Sensordaten für die KDD ausgewählt werden.

Vorverarbeitung

Da die Messfehler der Rohdaten bereits durch die vorhandenen Funktionen der verwendeten Sensorbibliotheken ausgebessert werden und es beim Erkennen vieler Zustände um das Bemerken von Tendenzen und Differenzen geht, nicht um die Feststellung eines expliziten Messwertes, wurde auf eine explizite Vorverarbeitung verzichtet.

Transformation

Die Transformation erfolgt zum einen in Form der Berechnung der Tassenposition und zum anderen durch manuelle Auswahl der zu evaluierenden Sensorwerte im Matlabskript. Das Matlabskript wird im nachfolgenden Kapitel genauer beschrieben.

Data-Mining

In dieser Arbeit wurden unterschiedliche Verfahren des Data-Mining angewendet.
Zum einen werden die Temperatur- und Winkelwerte wie in Kapitel 3.2 beschrieben generalisiert.
Zum anderen wird das k-Means-Clusterverfahren angewendet (Abschnitt 4.4.2 & 3). Das k-Means-Verfahren besticht durch seine einfache Implementierungsweise. Mit ihm ist es möglich, ohne großen Aufwand Messwerte zu visualisieren um einen Überblick zu erhalten.

Evaluation

Die Daten wurden mit Hilfe von Matlab-Plots visualisiert. Dies wird in Abschnitt 4.4.3. ausgeführt.

4.4.2 Matlabskript

Die aufgenommenen und in einer Textdatei gespeicherten Werte werden mit einem Matlabskript (CVS_Read_DC20.m) verarbeitet. Damit dies funktioniert, müssen alle gesendeten Wartesignale („x“) aus der Datei gelöscht werden, sodass in der ersten Zeile der Datei der erste gesendete Messwert steht. Das Skript liest Daten einer Textdatei ein, welche sich im Matlab Work-Verzeichnis befindet. Die genaue Struktur der eingelesenen Daten befindet sich im Anhang 7.1.1.. Der externe Zeitstempel wird aus dem UNIX-Format in ein von Menschen lesbares Format gebracht. Der interne Zeitstempel startet bei Null und wird in Millisekunden angegeben.
Die Sensordaten werden Variablen zugewiesen. Dabei gibt es eine Variable, welche alle Sensorwerte enthält. Ansonsten enthält jede Variable alle Daten einer Sensor-Klasse.
Anschließend werden aus den Variablen welche die Sensoren vertreten einzelne Datenreihen ausgewählt, um diese je nach Bedarf zu zwei- oder dreidimensionalen Clusterdaten zusammen zusetzten.
Die Matlab-Funktion „kmeans“ erstellt k-Cluster welche zusammen mit den Clusterdaten durch die Funktion PlotCluster [19] visualisiert werden.

4.4.3 Plots

Zur Auswertung der Messwerte wurden diese grafisch dargestellt. Dabei entspricht ein Punkt einem Datenpunkt und ein Viereck einem Clusterzentrum.

Abstellen voll

Abbildung 10 zeigt die im Versuch 4.3.1 aufgenommenen Beschleunigungswerte der z-Achse in g über die Zeit in Millisekunden. Es wurden sieben Cluster gewählt um das siebenmalige Abstellen besser zu visualisieren. Da bekannt ist, welche Bewegungen vollführt wurden, lassen sich Muster erahnen. Für validierte Aussagen ist die Datendichte von drei Hertz allerdings zu gering.

Aufnehmen

Der Plot in Abbildung 11 zeigt die Beschleunigungswerte der z-Achse in g über die Zeit in Millisekunden. Die zehn Sekunden andauernde Ruhephase ist deutlich zu erkennen. Ein nachvollziehbarer Sensorwertverlauf des Aufnehmens lässt sich in diesem Plot nicht feststellen.

Tragen

Bei den beiden „Tragen“-Experimenten war die x-Achse in Gehrichtung ausgerichtet. Daher werden hier die Plots der Beschleunigungswerte der x-Achse in g über die Zeit in Millisekunden betrachtet.

Abbildung 12 visualisiert das unübersichtliche Ergebnis des Versuchs „1Tragen“ (304.3.3). Rückschlüsse lässt dieser Plot nicht zu.

In Abbildung 13 ist der Stillstand von 25 Sekunden eindeutig abgebildet.

Um allerdings Merkmale in den Sensorwerten zu erkennen muss die Datenrate erhöht werden.

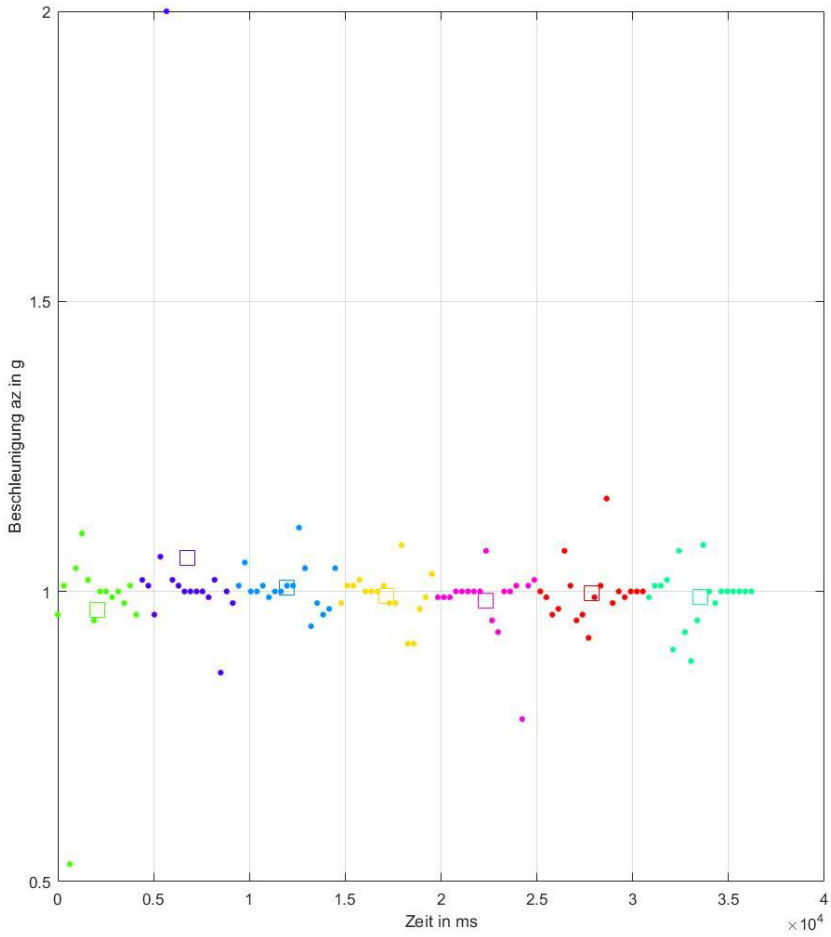

Abbildung 10: Plot Beschleunigung z-Achse und Zeit, 201708151010_1AbstellenVoll

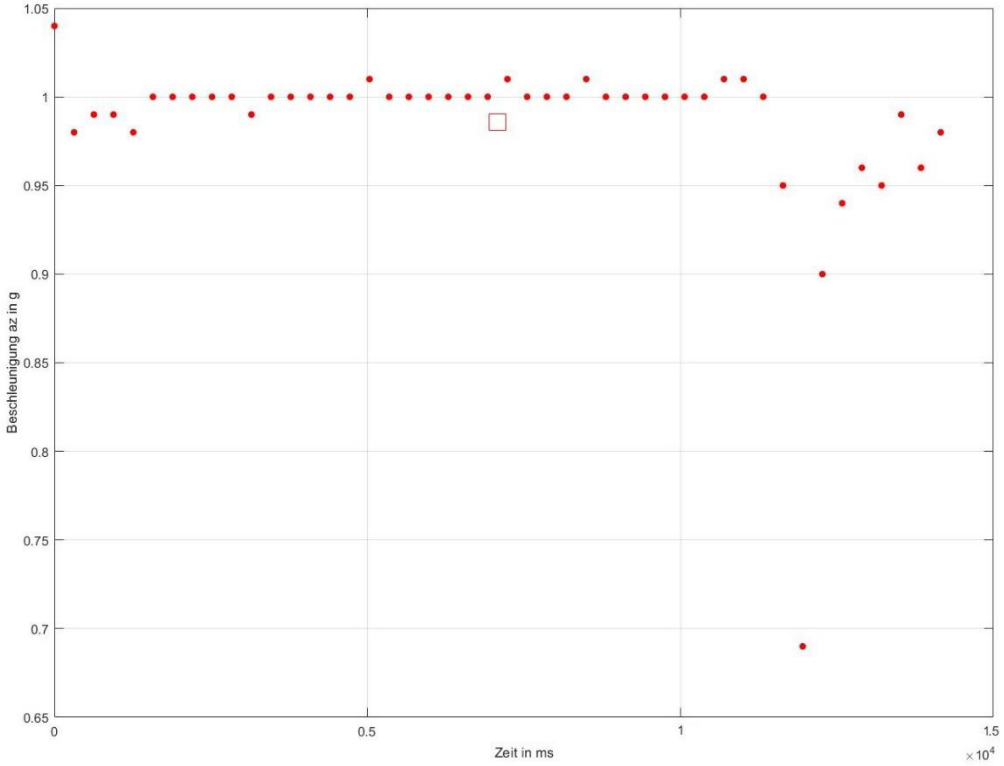

Abbildung 11: Plot Beschleunigung z-Achse und Zeit, 201708161315_1Aufnehmen

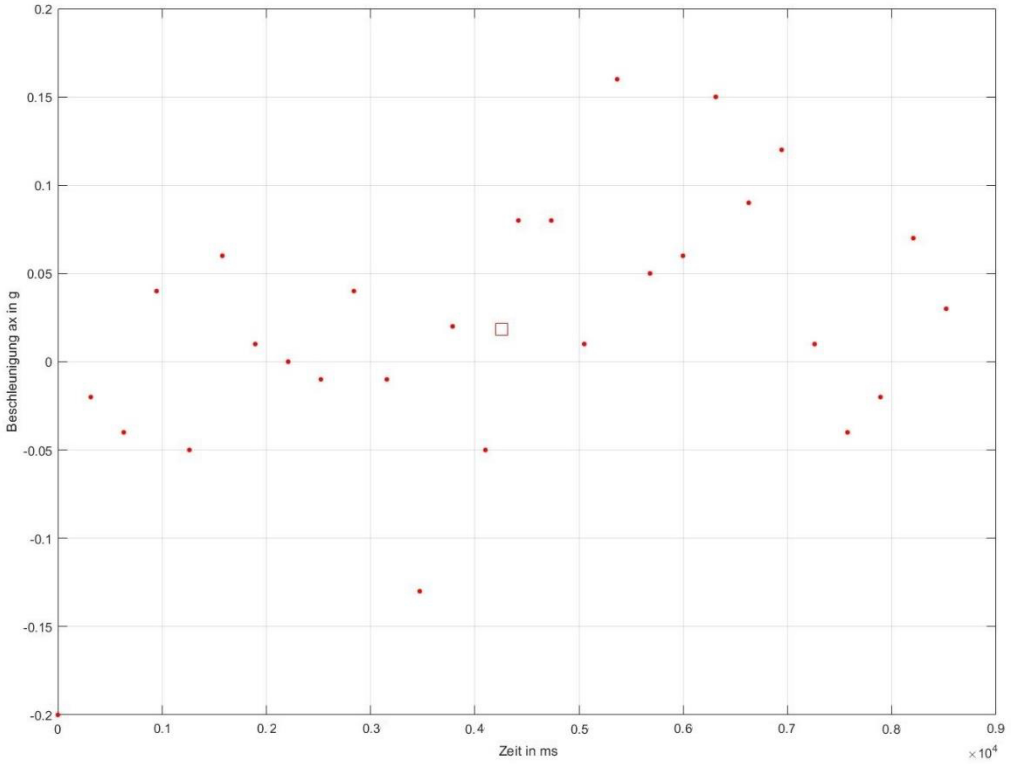

Abbildung 12: Plot Beschleunigung x-Achse und Zeit, 201708151450_1Tragen

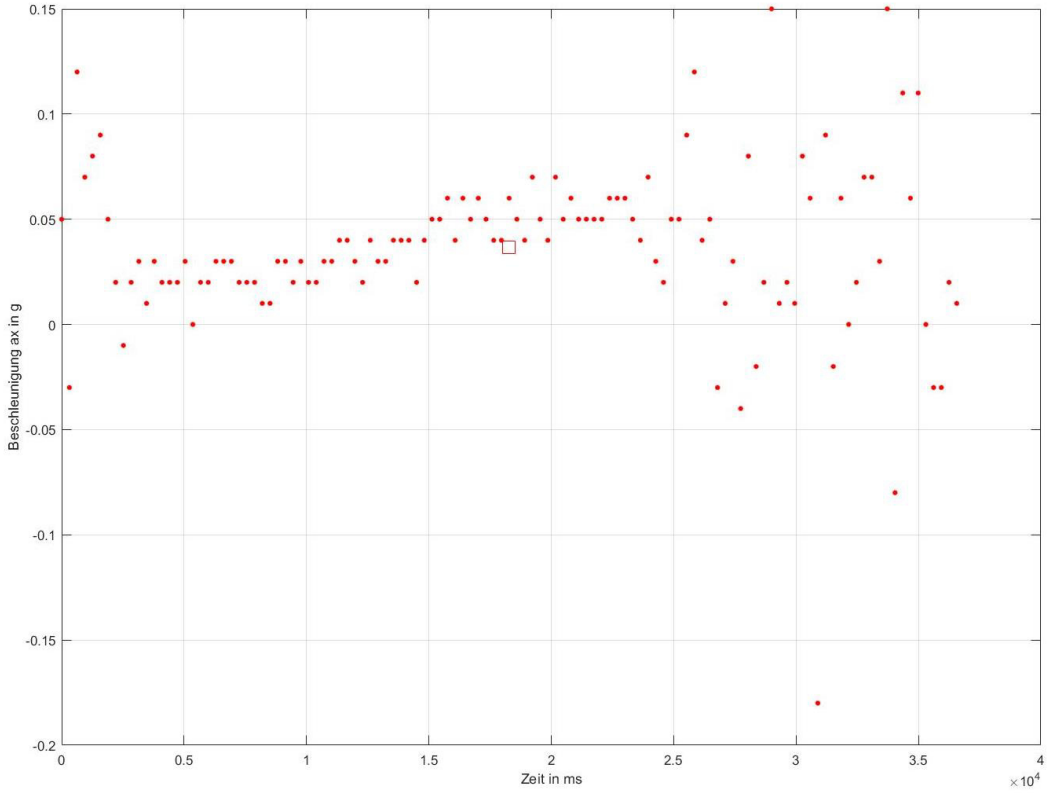

Abbildung 13: Plot Beschleunigung x-Achse und Zeit, 201708161650_2Tragen

4.5 Fazit

Eine rudimentäre Zustandserkennung wäre mit der vorhandenen Datenrate von drei Hertz möglich. So könnte zum Beispiel anhand Abbildung 13 die Zustandsveränderung von „Gehalten“ zu „Gehen“ erkannt werden.
Aufgrund der geringen Datendichte gehen allerdings viele Informationen verloren. Dadurch steigt die Wahrscheinlichkeit enorm, Zustände fehlerhaft zu interpretieren. Bevor weitere Versuche zur Zustandsfeststellung gemacht werden sollten, muss die Datenrate stark erhöht werden. Das Nyquist-Shannon-Abtasttheorem besagt, dass die Abtastfrequenz mindestens doppelt so hoch sein muss wie die Frequenz des abgetasteten Signals. Da die abzutastende Frequenz erst noch festgestellt werden muss, sollte die Datenrate für die nächsten Tests möglichst hoch ausgewählt werden.
Die Verwendung des k-Means Clusterverfahrens wäre bei den gezeigten Plots nicht notwendig gewesen, da diese immer über die Zeit geplottet wurden.
Es hilft aber dabei, verschiedene Abschnitte in den Sensorwerten leichter zu erfassen, wie zum Beispiel in Abbildung 10. Dort zeigt die farbliche Einteilung das siebenmalige Abstellen der Tasse.
Für weiterführende Betrachtungen empfiehlt sich das DBSCAN-Verfahren, da es als dichtebasiertes Verfahren relevante Datenmengen besser hervorhebt und einzelne auftretende Messfehler wie in Abbildung 11 nicht die Position des Clusterzentrums verfälschen.

5 Erhöhung der Datenrate

Wie in der Einführung zu Kapitel 4 bereits erwähnt, wurde die geringe Datenrate bereits bei ersten Programmtests des Datacollect-Programms festgestellt. Dass der Schock des Abstellens, welcher drei Millisekunden lang auftritt, mit einer Datenrate von drei Herz nicht zuverlässig erkennbar ist war abzusehen. Auch, dass das Anheben einer Tasse, was in etwa eine Sekunde dauert (Abschnitt 3.2.1), anhand von drei Messwerten schwer zu verifizieren ist war vorhersehbar. Daher wurde die Erhöhung der Datenrate priorisiert. Die angewendeten Lösungsansätze werden in dem folgenden Abschnitten vorgestellt.

5.1 Softwareseitige Optimierung

Das Hauptprogramm wurde durch Minimalisierung der Funktionsaufrufe verschlankt, siehe Abschnitt 3.3.2. Das Einlesen der Daten dauert fünf Millisekunden und das Erstellen des Strings acht Millisekunden.

Das Speichern der Daten in einem Array und anschließendes Senden des Datenarrays wurde überlegt. Dies hätte in den Zyklen in denen der Array mit Daten gefüllt wird, minimale Verbesserungen bringen können. Allerdings nur im Gegenzug für größere blinde Sendeflecken, also dem Zeitpunkt in dem der Datenarray an den PC übermittelt wird. Aufgrund dieses Nachteils wurde davon abgesehen.
In Gesprächen mit Mitgliedern der TESSA-Arbeitsgruppe der HAW wurde mir die Compileroptimierung O3 empfohlen.
O3 ist eine Optimierungsform, welche die Kompilierungszeit minimiert und die Ausführungszeit verbessert. Der Compiler der Arduino IDE wurde nach [20], in O3 geändert. Außerdem wurde die Baudrate auf $2 \cdot 10^6$ Baud erhöht, die höchste offiziell unterstütze Baudrate des Arduino. Allerdings konnte keine Verbesserung festgestellt werden.
Dass die Compileroptimierung zu keiner Veränderung führte, könnte daran liegen, dass der ausführbare Code nicht sehr lang ist und keine aufwendigen Rechenoperationen benötigt werden, welche optimiert werden könnten.

5.2 Testtasse

Da die Optimierung der Software nicht den gewünschten Erfolg brachte, erfolgte das Abwägen von Verbesserungen der Hardware. Der verwendete Arduino MKR 1000 mit einem 48MHz Singlecore Prozessor hat nur begrenzte Leistung zur Verfügung. Aus diesem Grund und um die vorhandene Programmstruktur weiterhin nutzen zu können, wurde sich für die Einbindung eines neuen Mikrocontrollers in die gleichbleibende Peripherie entschieden.

5.2.1 Aufbau

Das CSTI fragte bereits zu Beginn dieser Arbeit an, ob weiter Mate-Mugs im Rahmen dieser Arbeit erstellt werden könnten. Dies bewegte mich dazu, anhand des Prototypen von Joschka Sondhof, einen Testtassen-Klon zu bauen. Er sollte alle Sensoren des Originals enthalten, es sollte aber ein ESP 32-Developmentboard mit einem 160MHz Dualcore Prozessor verbaut werden. Die Auswahl der Mikrocontroller wird in Abschnitt 5.2.2 genauer beschrieben.
Da das ESP 32-Developmentboard bereits über Bluetooth und Wifi verfügt, mussten diese Komponenten nicht extra bestellt werden. Außerdem wurden die Aktoren minimiert. Die Anzahl der LEDs wurde auf zwei reduziert und der Vibrationsmotor ganz weggelassen. Alle für die Testtasse bestellten Komponenten stehen in Abschnitt 7.1.2.

Im Rahmen der Planung der Versuchsdokumentation gab es die Idee, für Langzeittests einen Raspberry-Pi als Datenlogger anzuschließen und gleichzeitig über eine Kamera am Pi ein Video aufzunehmen. Die Möglichkeit zu einem späteren Zeitpunkt einen Raspberry-Pi als Mikrocontroller anzuschließen wurde bei der Planung der Testtasse wieder relevant.

Um diese Option offen zu lassen, wurde der Anschluss für den Mikrocontroller außerhalb der Tasse realisiert.

Da in der Studienarbeit [2] nur ein Lageplan (Abbildung 14) der Komponenten und deren Anschluss an den Mikrocontroller (Abbildung 16), nicht aber ein Schaltplan enthalten ist, wurde anhand der Verdrahtung der Mate-Mug ein Schaltplan für die Testtasse entwickelt (Abbildung 17). Für die Schaltplanerstellung wurde das Programm Fritzing (7.2.3) verwendet.

Anhand dieses Schaltplans wurden die Komponenten zur Testtasse verlötet.

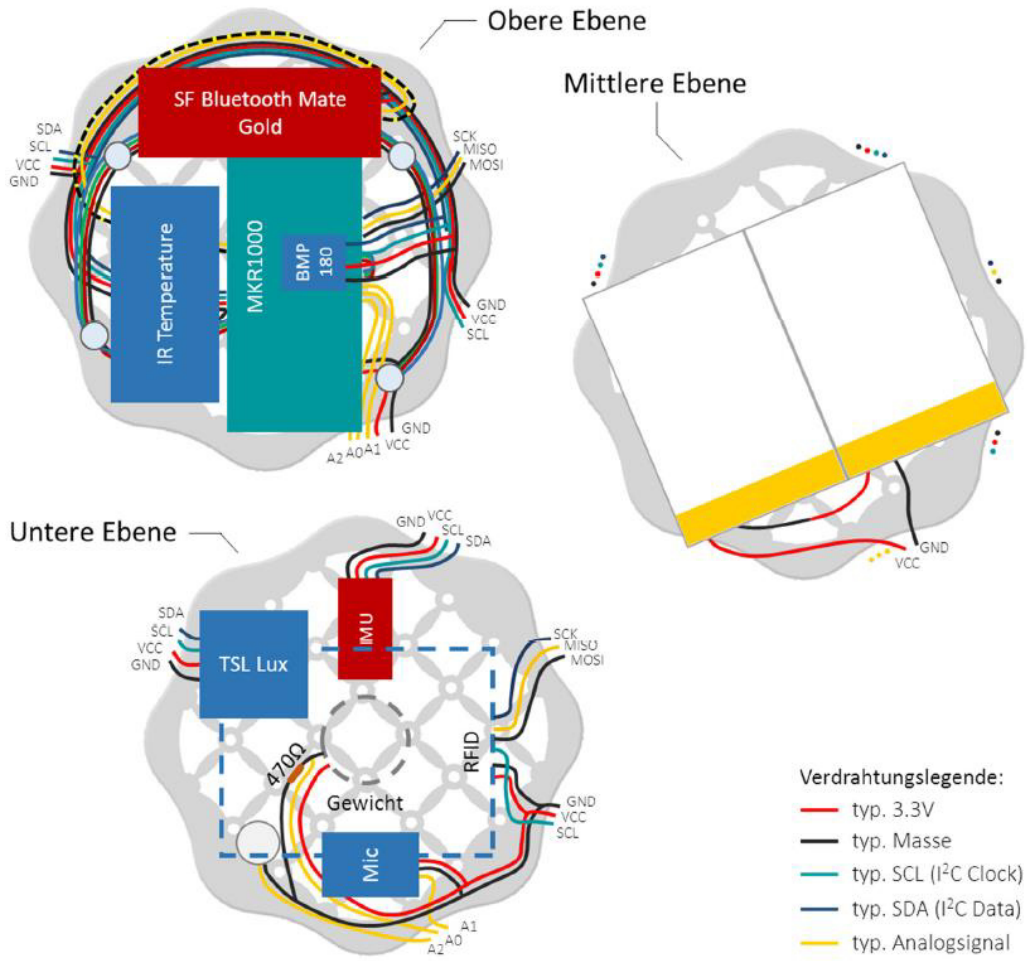

Abbildung 14: Lageplan Mate-Mug, aus [2 S.38]

5.2.2 Mikrocontroller

Um die vorhandene Programmstruktur weiterhin nutzen zu können, muss der neue Mikrocontroller Arduino-kompatibel sein.

ESP 32

Nach einer ersten Recherche war die Kompatibilität bei dem ESP 32 bis auf kleine Einschränkungen gegeben. Daher wurde die Tool-Chain für den ESP 32 nach den „Installation Instructions“ in Quelle [21] installiert. Mit Hilfe eines Beispielprogramms wurde die korrekte Einrichtung der Tool-Chain erfolgreich getestet, bevor die Implementierung der Testtasse auf dem ESP 32 getestet wurde.

Durch Umschreiben des Befehls in „Digital.Write“ konnte der nicht unterstützte Arduino-Befehl „Analog.Write“ ohne Funktionsverlust des visuellen Feedbacks behoben werden. Außerdem kam es durch die Einbeziehung der ESP 32 Core-Dateien zu Bibliotheksdoppelungen in der Ordnerstruktur, welche auch durch Verschieben in ein anderes Verzeichnis der nicht benötigten Arduino-Bibliothek gelöst werden konnte.

Leider stellte sich dann heraus, dass die Bibliotheken der verwendeten Breakoutboards stark an Arduino gebunden sind und auf dem ESP 32 die Defines der Sensor-Bibliotheken zu Fehlern führen. Bei Rücksprache mit dem CSTI erfuhr ich, dass dieses Problem dort bekannt sei. In bisherigen Projekten konnten dort manche Breakoutboards nur individuell und mit großem Aufwand auf dem ESP 32 zum Laufen gebracht werden. Aus diesem Grund wurde der ESP 32 verworfen.

Teensy 3.2

Eine Recherche nach leistungsstarken Arduino-kompatiblen Mikrocontrollern führte zum PJRC-Store und der Teensy-Familie. Da der Teensy 3.2 mit 72MHz an der Hochschule verfügbar war und nicht bestellt werden musste, wurde dieser gewählt. Mit Hilfe des Programms Teensyduino (7.2.4) wurde das Tassenprogramm ohne nötige Änderungen erfolgreich implementiert.

Tabelle 8 zeigt noch einmal die verwendeten Mikrocontroller in einer Übersicht.

Tabelle 8: Übersicht Mikrocontroller

Mikrocontroller	**Arduino MKR 1000**	**ESP 32 Dev.-Board**	**Teensy 3.2**
Prozessor	SAMD21 Cortex-M0+ 32bit low power ARM MCU, 48 MHz	Xtensa Dual-Core 32-bit LX6 microprocessors, 160 MHz	MK20DX256 32 bit ARM Cortex-M4, 72 MHz
RAM Speicher	32000	520000	65536
Flash Speicher	256000	448000	262144
Wlan Modul	Ja	Ja	Nein
Bluetooth Modul	Nein	Ja	Nein

5.3 Fazit

Eine Verbesserung der Datenrate konnte mit dem fast doppelt so leistungsstarken Teensy 3.2 nicht festgestellt werden. Da die Datenrate nicht erhöht werden konnte, wurde im Folgenden wieder mit dem originalen MKR 1000 gearbeitet, um auch dessen Wifi gegebenenfalls nutzen zu können. Abbildung 15 zeigt die Entwickelte Testtasse mit angeschlossenem Arduino.

Abbildung 15: Testtasse, fotografiert von Jessica Broscheit

6 Resümee

In dieser Bachelorarbeit wurde an einer Zustandserkennung durch Auswertung der aktuellen Sensordaten innerhalb einer smarten Tasse gearbeitet. Hierfür wurden zum einen die zu erkennenden Zustände definiert, zum anderen Charakteristika und Merkmale dieser Zustände gesucht, welche anhand von Modellen und Versuchen eindeutig identifizierbar sind. Die Details dazu befinden sich in Kapitel 3. Aufgrund einer zu geringen Datenrate konnten die in Kapitel 4 durchgeführten Experimente nicht in dem geplanten Umfang durchgeführt werden, weswegen auch die erfassten Daten und deren Auswertung nicht zu dem erwarteten Ergebnis führen. Die Versuche, das Problem der zu kleinen Datenrate zu beheben,wurden in Kapitel 5 beschrieben.

6.1 Zusammenfassung und Bewertung der Ergebnisse

Fünf Bewegungs- und sechs Inhaltszustände wurden definiert, mit denen der normale Alltag einer Tasse mit geringen Einschränkungen dargestellt werden kann.
Außerdem wurde ein Zeitraum spezifiziert, in welchem die Tasse für gewöhnlich aktiv genutzt wird. Erkennungsverfahren wurden für die gewählten Zustände festgelegt. Für diese Erkennungsverfahren konnten entsprechende Experimente und Modelle geplant und teilweise durchgeführt beziehungsweise implementiert werden.
Die erwähnten Einschränkungen ergeben sich aus den genau definierten Haltungen der Tasse und den einfachen Versuchssettings. Daraus resultiert das Erkennen der genannten Zustände, in beschriebener Form, allerdings keine allgemeingültige Zustandserkennung. Dennoch bieten die Zustände eine gute Grundlage für eine erste Zustandserfassung, welche iterativ ausgebaut werden kann.
Außerdem muss bedacht werden, dass in dieser Arbeit nicht zwischen Zustandserkennung und -schätzung differenziert wurde. So kann bei der Füllstandserkennung der aktuelle Füllstand (Tabelle 3) nur geschätzt werden. Auch die in Kapitel 3 vorgestellten Verfahren der Zustandserkennung sind bei genauerer Betrachtung nur Schätzungen, da die gewählten Merkmale auch durch andere beliebige Tassenbewegungen hervorgerufen werden könnten.
Für das Modell zur Erkennung des Trinkvorgangs wurden Funktionen geschrieben welche den Kippwinkel der Tasse unabhängig von der Griffposition erkennen. Die Funktionen wurden in die vorhandene Programmstruktur der Klasse sensorIMU integriert und können über diese aufgerufen werden. Ein Aufruf per JSON ist derzeit nicht möglich.
Die korrekte Integration und Funktion wurde getestet. Allerdings erfolgte keine Überprüfung in Form eines Experiments. Wird die Tasse gestartet, braucht die Winkelerkennung circa fünf Sekunden im Ruhezustand um zu Nullen. Anschließend erkennt sie ohne erkennbare Verzögerung den aktuellen Kippwinkel der Tasse. Wird auf das Kalibrieren verzichtet, kommt es zu Abweichungen in der Winkelbestimmung. Bei gegebenenfalls auftretenden Messfehlern, zum Beispiel durch Schütteln verursacht, kehrt der Winkel dank des Komplementärfilters bei Stillstand wieder auf Null zurück.
Von der Entwicklung und Implementierung weiterer Modelle musste aufgrund des gesteckten Zeitrahmens abgesehen werden.
Für die Datenverarbeitung wurde ein Programm geschrieben (Datacollect), welches die aktuellen Sensorwerte der Tasse aufnimmt und diese per Serial-Port an den PC sendet. Die Datenrate der gesendeten Daten beträgt drei Hertz und konnte weder durch Programm und Compiler-Optimierung noch durch einen schnelleren Mikrocontroller in einer neu entwickelten Testtasse erhöht werden.

Daher wird vermutet, dass der Grund für die niedrige Datenrate in den verwendeten Standartbibliotheken der Sensoren zu finden ist. Möglicherweise werden in einigen Bibliotheken Softwaretimer verwendet. Dies wäre eine plausible Erklärung für die langsame interne Tassenkommunikation per I2C.

Um die gesammelten Daten zu visualisieren wurde ein Matlabskript geschrieben, welches Sensorwerte aus einer Textdatei einliest (CVS-Read). Aus den Daten können mithilfe des Skriptes Messreihen beliebiger Sensoren ausgewählt werden und in zwei- oder dreidimensionalen k-Means Clustern geplottet werden.
Auch in den Plots wird deutlich, dass mit einer Datenrate von drei Hertz keine validierten Aussagen über die Bewegung des Tassenträgers möglich sind. Allerdings lassen sich in den Plots Sensorverläufe vermuten die nahelegen, dass die in dieser Arbeit gewählten Ansätze bei höherer Datenrate zum gewünschten Ergebnis geführt hätten. Im Speziellen ist hier der Plot des Versuchs 1Tragen (Abbildung 12) gemeint, in welchem man eine Sinuskurve erahnen kann.
Aufgrund der geringen Datenrate und dem vergeblichen Versuch diese zu erhöhen, konnten nicht alle in dieser Bachelorarbeit gestellten Forschungsfragen beantwortet und nicht alle gesetzten Ziele erreicht werden. Mögliche Merkmale der Bewegungen konnten zwar in Kapitel 3 definiert werden, doch war eine praktische Validierung nicht möglich.
Da die praktische Validierung scheiterte, konnten darauf aufbauende Themen wie die Benutzererkennung anhand der Merkmale oder die Genauigkeit des erkannten Zustands nicht betrachtet werden. Auch die Szenarienzuweisung beziehungswiese das selektive Senden von Daten bliebt offen.

6.2 Ausblick

Die in dieser Arbeit entworfenen Zustände und Wege diese zu identifizieren, bieten eine Basis für eine grundlegende Zustandserkennung einer smarten Tasse. Aufbauend auf dieser Grundlage können die Zustände Schritt für Schritt erweitert und verfeinert werden um immer genauer immer komplexere Zustände zu erfassen. So könnte zum Beispiel die Erfassung des Füllstands genauer werden, wenn in Kombination mit dem Kippwinkel, die Dauer des gekippten Zustands mit betrachtet wird.
Vor allem aber muss die Datenrate erhöht werden, damit anhand der Sensorwerte valide Aussagen über die Zustände getroffen werden können. Das könnte durch Optimierung der Sensorbibliotheken des vorhandenen Systems erreicht werden. Außerdem besteht die Möglichkeit einen neuen Prototyp auf Basis eines ESP32 zu entwickeln. Dies würde für ein insgesamt leistungsstärkeres System sorgen.

Die Fortsetzung des Mate-Mug-Projekts ist durch Andreas Bloch gesichert. Herr Bloch beschäftigt sich in seiner Studienarbeit „Ausarbeitung zur benutzerfreundlichen Optimierung einer kontextsensitiven Tasse“ unter anderem mit der Integrierung einer kontaktlosen Ladeeinheit. Bleibt diese beständige schrittweise Verbesserung der smarten Tasse bestehen, sind die in 1.1 beschriebenen Verbesserungen der Lebensqualität in naher Zukunft realisierbar.

7 Literaturverzeichnis

[1] CREATIVE SPACE FOR TECHNICAL INNOVATIONS: *Creative Space for Technical Innovations* : *about-csti*. Steindamm 94, 20099 Hamburg, 20.03.2017

[2] SONDHOF, Joschka: *Selbstauskunftsfähigkeit von Alltagsgegenständen* : *Studienarbeit* 2017-05-19

[3] MARK WEISER: *The Future of Ubiquitous Computing on Campus*. In: *Communications of the ACM* 1998 (01/1998), Nr. 41, S. 41–42

[4] MARK WEISER: *The Computer for the 21st Century*. In: *Scientifc American* 1991 (09/1991), September, S. 94–104

[5] HOSSEIN MOUSAVI HONDORI, MARYAM KHADEMI, CRISTINA VIDEIRA LOPES: *Monitoring Intake Gestures using Sensor Fusion for Smart Home Tele-Rehab Setting*. Huston, Texas USA. 1st Annual IEEE Healthcare Innovation Conference of the IEEE EMBS,. 2012-07-11. URL https://www.researchgate.net/profile/Hossein_Mousavi_Hondori/publication/233398835_Monitoring_Intake_Gestures_using_Sensor_Fusion_Microsoft_Kinect_and_Inertial_Sensors_for_Smart_Home_Tele-Rehab_Setting/links/02e7e515626a7eee59000000/Monitoring-Intake-Gestures-using-Sensor-Fusion-Microsoft-Kinect-and-Inertial-Sensors-for-Smart-Home-Tele-Rehab-Setting.pdf

– Überprüfungsdatum 2017-10-09

[6] P.A. HARLING UND A.D.N. EDWARDS: Hand Tension as a Gesture Segmentation Cue, Bd. 1997. In: *Progress in Gestural Interaction*, S. 75–88

[7] KURTENBACH G. ; HULTEEN E.A: Gestures in Human-Computer Communication. In: *The Art of Human-Computer Interface, 1990*, S. 309–317

[8] FAYYAD, Usama ; PIATETSKY-SHAPIRO, Gregory ; SMYTH, Padhraic: *From data mining to knowledge discovery in databases*. In: *AI Magazine* 1996, Nr. 17, S. 37–54

[9] ESTER, Martin ; SANDER, Jörg: *Knowledge Discovery in Databases*. München : Springer, 2000

[10] ASTROSTATISTICS: *Cluster Analysis* : *Steps in the 2-means algorithm*. URL http://astrostatistics.psu.edu/su09/lecturenotes/clus2.html

– Überprüfungsdatum 2017-07-09

[11] YARPIZ: *DBSCAN Clustering in MATLAB*. URL http://yarpiz.com/255/ypml110-dbscan-clustering

[12] PROF. DR. O. BITTEL: *Mobile Roboter-Positionierung und Orientierung*. Konstanz, Hochschule Konstanz Technik, Wirtschaft und Gestaltung. WS 16/17. URL http://www-home.htwg-konstanz.de/~bittel/ain_robo/Vorlesung/02_PositionUndOrientierung.pdf

– Überprüfungsdatum 2017-08-29

[13] BARTZ, Markus: *Quaternionen* : *Seminar Computergraphik*. Vortrag 12. April 2001. Koblenz : Universität Koblenz-Landau, 2001

[14] CHEN-YU HSU: *Extracting Gait Velocity and Stride Length from Surrounding Radio Signals*. Cambridge, MA. URL https://people.csail.mit.edu/cyhsu/papers/wigait_chi17.pdf

– Überprüfungsdatum 2017-08-24

[15] JOEVEO: *Temperfect mug*. URL http://joeveo.com/#bg-03

– Überprüfungsdatum 2017-08-24

[16] POWER PRACTICAL: *The Jül: Heated Smart Mug for Coffee & Tea*. URL https://www.kickstarter.com/projects/powerpractical/the-jul-heated-smart-mug-for-coffee-and-tea/

– Überprüfungsdatum 2017-08-24

[17] PHILLIP SCHMIDT: *Fast Quaternion Integration for Attitude Estimation* : *Phillip's Technology Corner*. URL http://philstech.blogspot.de/2014/09/fast-quaternion-integration-for.html

– Überprüfungsdatum 2017-08-21

[18] PHILLIP SCHMIDT: *Complimentary Filter Example: Quaternion Based IMU for Accel+Gyro sensor* : *Phillip's Technology Corner*. URL http://philstech.blogspot.de/2015/06/complimentary-filter-example-quaternion.html

– Überprüfungsdatum 2017-08-21

[19] ELAD KIVELEVITCH: *PlotClusters(Data,IDX,Centers,Colors)*

[20] BODMER: *Arduino IDE 1.6.x Compiler Optimisations* : *= Faster Code*. URL http://www.instructables.com/id/Arduino-IDE-16x-compiler-optimisations-faster-code/

– Überprüfungsdatum 2017-04-09

[21] GIT-HUB COMUNITY: *Arduino core for ESP32 WiFi chip*. URL https://github.com/espressif/arduino-esp32

– Überprüfungsdatum 2017-04-09

Anhang

7.1 Dokumente

7.1.1 Datenstruktur und Variablennamen

Tabelle 9: Datensatzstruktur Matlabskript CVS_Read_DC20, Variablennamen

Spalte	Variablen Name	Beschreibung
1	time_ext	Timestamp extern
2	time_int	Timestamp intern
3	MIC	Microphone
4	IRTEMP_amb	IR-Temperatursensor Objekt
5	IRTEMP_obj	IR-Temperatursensor Ambiente
6	LUX_lux	Helligkeitssensor Sichtbar
7	LUX_ir	Helligkeitssensor IR
8	LUX_full	Helligkeitssensor volles Spektrum
9	PRTEMP_temp	Drucksensor Temperatur
10	PRTEMP_pre	Drucksensor Druck
11	PRTEMP_alt	Drucksensor Höhe
12	IMU_ax	IMU Beschleunigung x-Achse
13	IMU_ay	IMU Beschleunigung y-Achse
14	IMU_az	IMU Beschleunigung z-Achse
15	IMU_mx	IMU Magnetometer x-Achse
16	IMU_my	IMU Magnetometer y-Achse
17	IMU_mz	IMU Magnetometer z-Achse
18	IMU_gx	IMU Gyro x-Achse
19	IMU_gy	IMU Gyro y-Achse
20	IMU_gz	IMU Gyro z-Achse
21	IMU_pi	IMU Pitch
22	IMU_ro	IMU Roll
23	IMU_he	IMU Heading
24	IMU_ang	IMU Winkel
25	IMU_qx	IMU Quaternion x-Achse
26	IMU_qy	IMU Quaternion y-Achse
27	IMU_qz	IMU Quaternion z-Achse
28	WEIGHT_we	Gewichtssensor Gewicht in mg
29	WEIGHT_vo	Gewichtssensor Volumen
30	WEIGHT_fi	Gewichtssensor Füllstand

Im Matlabskript CVS_Read_DC21 steht der skalare Wert der Quaternion (IMU_qw) in Spalte 25, wodurch die restlichen Werte um eine Spalte nach hinten rücken.

7.1.2 Testtasse bestellte Komponenten

1x digital infrared temperature

https://www.reichelt.de/?ARTICLE=191254&PRO-VID=2788&wt_mc=amc141526782519998&gclid=CJqj2dv279QCFUuRGwod2WUGPg

1x Barometer

http://www.exp-tech.de/sparkfun-barometric-pressure-sensor-bmp180-breakout-barometer-druck-sensor?___SID=U

1x Lichtsensor

http://www.exp-tech.de/en/adafruit-tsl2561-digitaler-lichtsensor

1x 9DOF

http://www.exp-tech.de/sparkfun-9dof-sensor-stick

1x Drucksensor

http://www.exp-tech.de/force-sensing-resistor-1-5cm-diameter-circle-short-tail-fsr-402-short

1x Micro

http://www.exp-tech.de/adafruit-electret-microphone-amplifier-max4466-with-adjustable-gain

1x RFID Reader

http://www.watterott.com/de/Mifare-1356Mhz-RC522-RFID-Card-Reader-Module

3x RGB LEDs

https://www.conrad.de/de/led-mehrfarbig-rot-blau-gruen-rund-5-mm-300-mcd-700-mcd-1700-mcd-50-20-ma-195-v-33-v-33-v-kingbright-l-154a4su-1050465.html

7.1.3 Pinbelegung

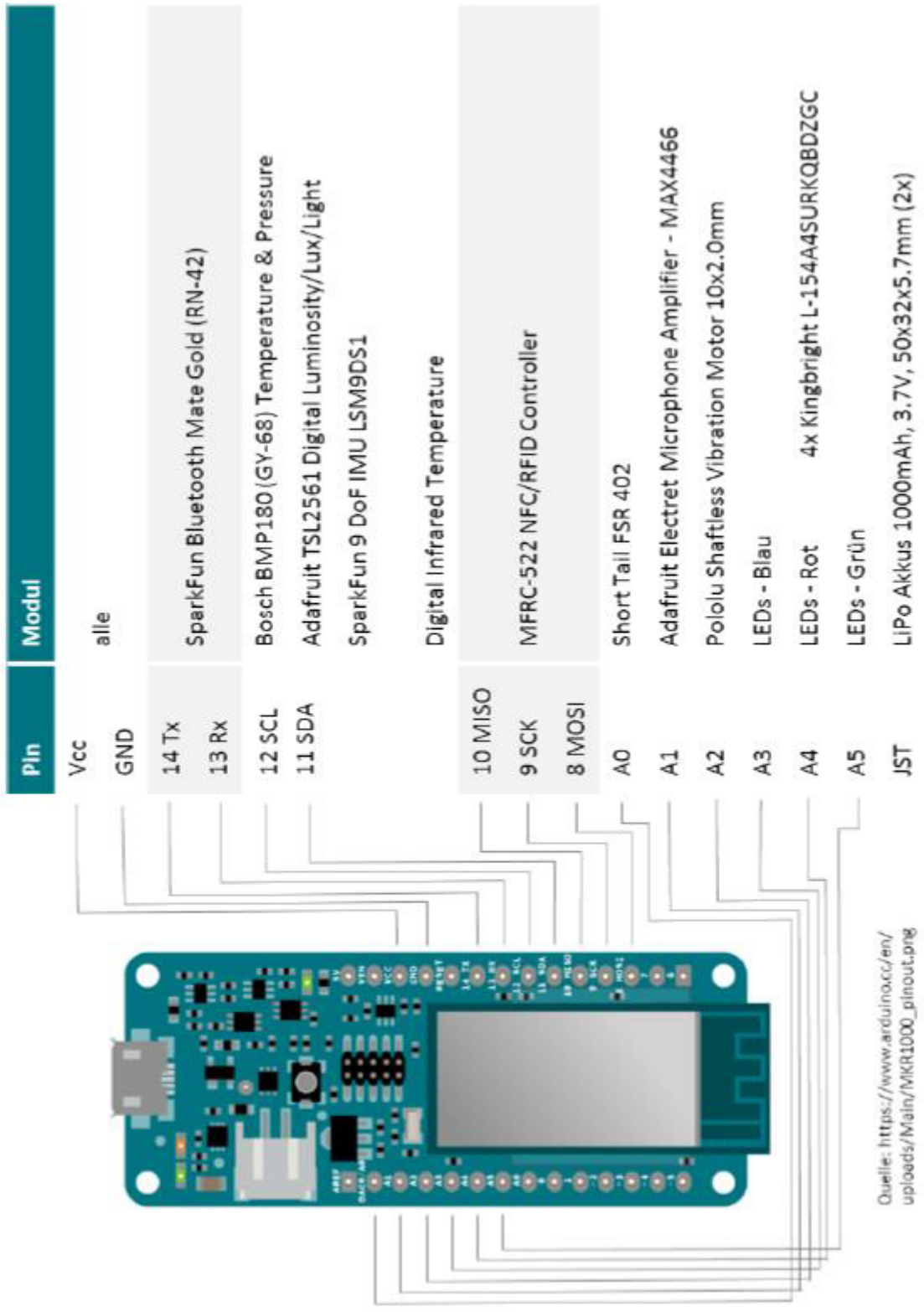

Abbildung 16: Pinbelegung aus [2 S.39]

7.1.4 Schaltplan

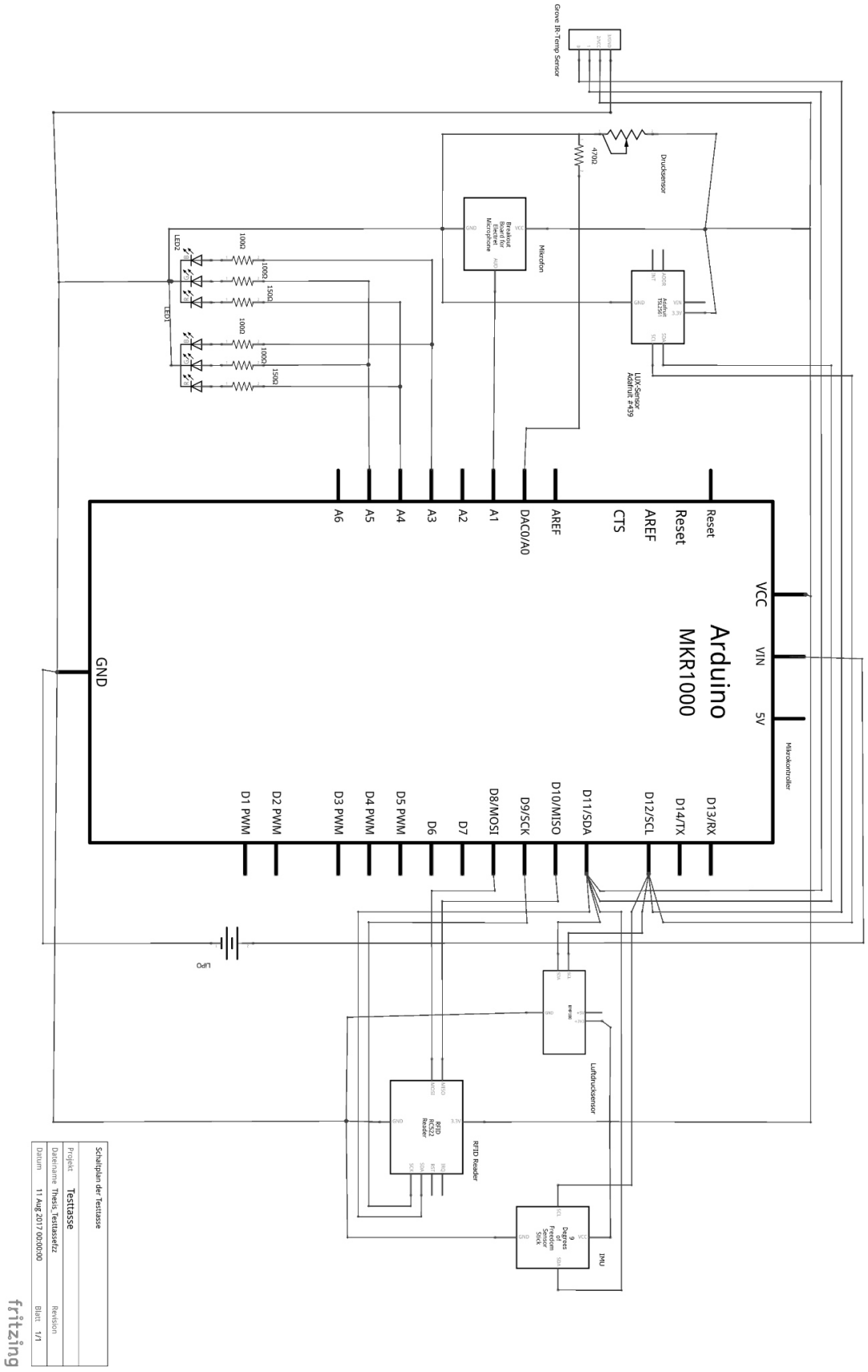

Abbildung 17: Schaltplan Testtasse

7.2 Programme:

7.2.1 Arduino IDE 1.8.2

Die Arduino IDE 1.8.2 wurde zur Programmierung der Tasse verwendet. Die aktuellste Version und Installationsanweisungen sind zu finden unter: https://www.arduino.cc/en/main/software

7.2.2 Real Term 2.0.0.70

Unter: https://sourceforge.net/projects/realterm/
kann die aktuellste Version des Programms Real Term heruntergeladen werden. Unter der Kategorie „Wiki“ und „Support“ finden sich Installationsanweisungen und Hilfestellungen.
Nachfolgend sollen die expliziten Schritte vorgestellt werden welche vor jedem Experiment zum Einstellen des Programms getätigt wurden.

Vorgenommene Einstellungen:

Nach jedem Start von Real Term muss in der Registerkarte Port (Abbildung 18) unter 1 die Verwendete Baudrate eingestellt, bei 2 gegebenenfalls der Port der Tasse gewählt und die Änderungen mit Klick auf 3 übernommen werden.

Anschließend muss in der Registerkarte Capture (Abbildung 19) zuerst der externe Time Stamp Unix ausgewählt werden. Wird dies vergessen, kommt es zu Komplikationen bei der Datenverarbeitung. Unter 2 kann das gewünschte Textdokument zum Speichern der Daten gewählt werden. In dieser Arbeit wurde der Dokumentenname capture.txt beibehalten und als Speicherort wurde das Standard Matlab Work-Verzeichnis „C:\Users\USER_NAME\Documents\MATLAB“ verwendet. Mit Klick auf 3 wird das Speichern der Daten gestartet. Die Datei capture.txt wird dabei überschrieben. Die eingehenden Daten werden nun nicht mehr im Real Term Terminal angezeigt, sondern direkt in der ausgewählten Datei gespeichert. Außerdem färbt sich der Bereich Capture der Capture-Registerkarte rot um eine aktive Datenspeicherung anzuzeigen. Die Speicherung der eingehenden Daten kann mit Klick auf Stop Capture (über 2) oder durch Trennen der Verbindung mit der Tasse beendet werden.

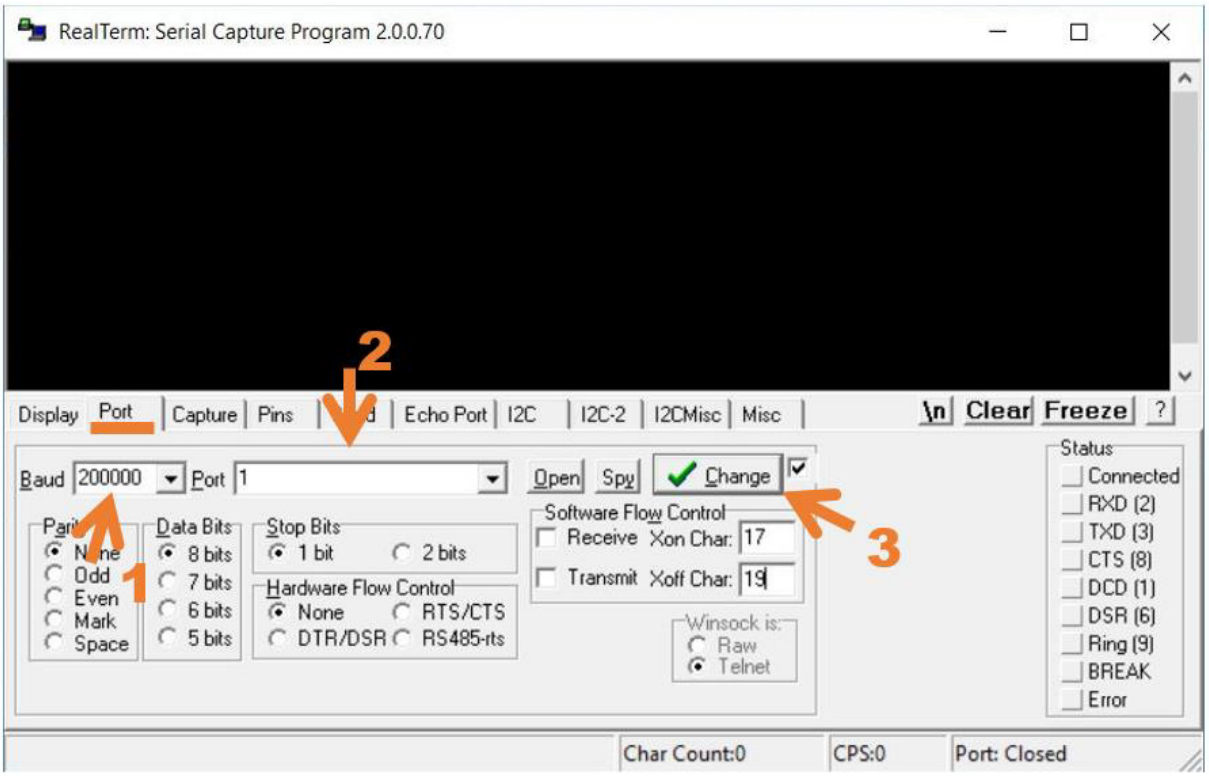

Abbildung 18: Real Term, Port-Register

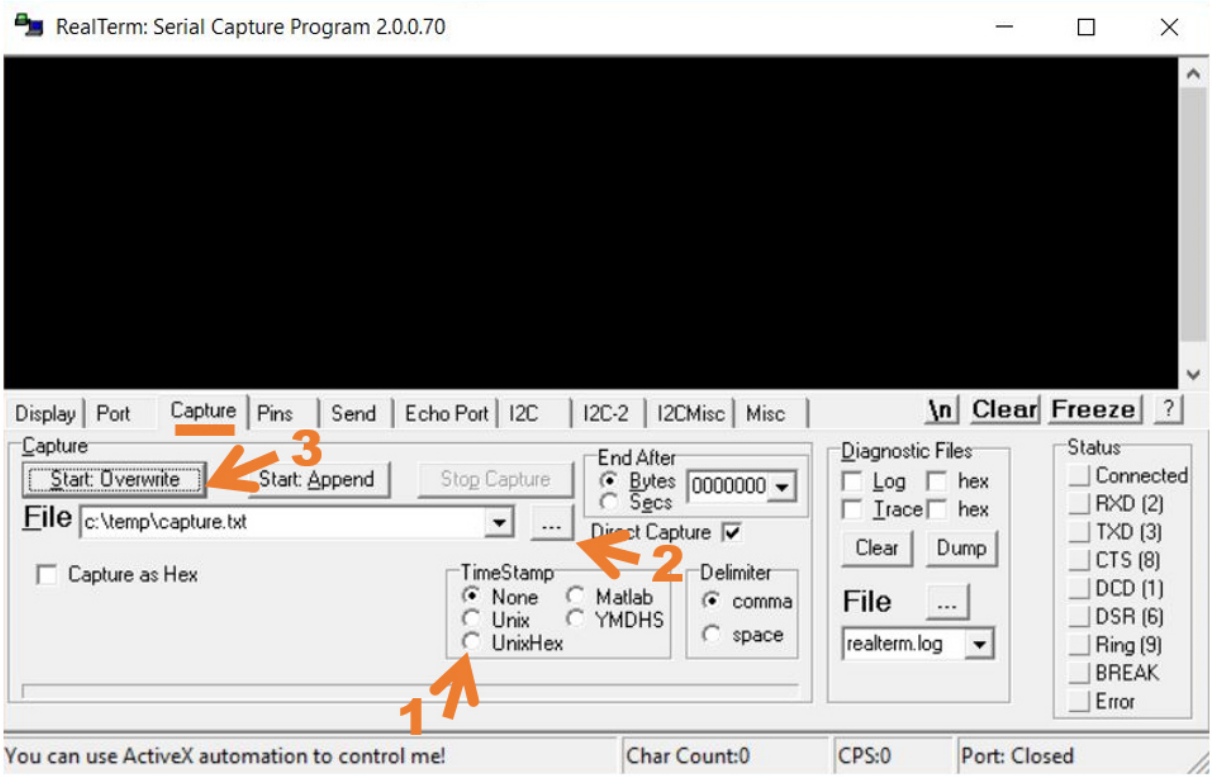

Abbildung 19: Real Term, Capture-Register

7.2.3 Fritzing 0.9.3

Das Programm Fritzing ist ein in auf Brake-out-Boards spezialisiertes, simples Schaltplanerstellungsprogramm. Unter http://fritzing.org/home/ sind umfangreiche Anleitungen, Hilfestellungen und die neuste Version zu finden.

7.2.4 Teensyduino 1.38

Durch Teensyduino können die Mikrocontroller der Teensy-Familie mit der Arduino IDE programmiert werden. Die aktuellste Version inklusive Installationsanweisung steht auf der Internetseite: https://www.pjrc.com/teensy/td_download.html

7.3 CD

In Tabelle 10 steht der Inhalt der beigefügten CD.

Tabelle 10: CD Inhalt

Ordnername	Inhalt	Hinweis
Matlab	Entwickelte & verwendete Matlabskripte, Beispiel-capture.txt Datei	Enthaltene Dateien in Matlab Workverzeichnis kopieren
Arduino	Geschriebene Datacollect-Programme und verwendete Bibliotheken	In entsprechende Arduino-Verzeichnisse zu kopieren
Experimente	Daten, Videos und Plots der durchgeführten Experimente	
Quellen	In dieser Arbeit als Quellen verwendete Werke soweit als Lokale Datei vorhanden	
	Bachelorthesis als Druck und PDF-Version, inkl. Worddatei	

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Palmer

Vorname: Tobias

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Zustandserkennung und Schätzung des aktuellen Szenarios anhand von Sensordaten einer smarten Tasse

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Hamburg	15/09/2017	
Ort	Datum	Unterschrift im Original