Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Erik Matthiessen

Entwicklung einer Gangschaltungs- und Bremsensteuerung für ein Fahrradergometer

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Erik Matthiessen

Entwicklung einer Gangschaltungs- und Bremsensteuerung für ein Fahrradergometer

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Phillip Jehnke

Eingereicht am: 26. November 2015

Erik Matthiessen

Thema der Arbeit

Entwicklung einer Gangschaltungs- und Bremsensteuerung für ein Fahrradergometer

Stichworte

EmotionBike, Fahrradphysik, Human-Computer-Interface, Immersion, Intuitives Interface, Natürliches Interface, RaspberryPi, Fahrradsimulation

Kurzzusammenfassung

Diese Arbeit handelt von der Erweiterung eines bestehenden Interfaces um eine Bremse und eine Gangschaltung. Im EmotionBike-Projekt an der HAW wird ein Ergometerfahrrad als Controller für ein Spiel verwendet. Mithilfe dieses Setups sollen die Emotionen des Fahrers gemessen und provoziert werden.

Zu Beginn der Arbeit wird der Rahmen des EmotionBike-Projekts näher erklärt. Anschließend werden die Anforderungen an das neue Modul beschrieben und wie dieses in den Kontext des Projekts passt. Nach dem die Anforderungen geklärt sind wird die Hardware vorgestellt, welche für die Realisierung verwendet wurde. Anschließend werden die theoretischen Grundlagen beschrieben, die, die Grundlage für das Physikmodell sind, welches den Kern der Simulation bildet. Aus diesen Grundlagen wird die Implementierung entwickelt.

Am Ende wird gefragt, ob die Umsetzung der Anforderungen erfolgreich war und wie sich das Projekt noch erweitern lässt. Es wird auch darauf eingegangen welche Probleme es während der Arbeit an diesem Projekt gegeben hat.

Erik Matthiessen

Title of the paper

Develloping a Gear and Brake Controller

Keywords

EmotionBike, Bicyclephysics, Human Computer Interface, Immersion, Intuitives Interface, Natural Interface, RaspberryPi, Bicyclesimulation

Abstract

This work subjects the extension of an existing interface by an brake-, and a gearsystem. The EmotionBike project, at the HAW, uses a dynamometer as a gamecontroller. In this setup it is intendet to meassure and provoke the users emotions.

The first part of the work focus on the EmotionBike project. After that it will show the requirements of the new module and how to fit it in the existing project. The next part is presenting the used hardwarecomponents, followed by the mathmatical fundamentals, used to create the physicsmodel, which is the center of the module. Based on this basics the implementaion is created.

The last part is about the question weather the implementatio was sucessfull or not and on which way this projectpart could go on. And there is a section about the problems occured during the implementation process.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Wie steuert man einen Computer?

Mit dieser Frage müssen sich die Menschen schon die gesamte, kurze, Geschichte des Computers beschäftigen. Die verschiedenen Ideen reichen dabei von Lochkarten, Magnetbändern und Druckern über Mäuse, Tastatuen und Monitore bis zum Touchscreen und der Sprachsteuerung.

Dabei ist es auffällig das alle diese Interaktionsmöglichkeiten den Anspruch haben die Interaktion mit dem Computer zu vereinfachen. Dabei wird der Computer, aus Benutzersicht, immer abstrakter beschrieben und die Steuerung hat immer weniger mit den internen Abläufen im Computer zu tun. Da die Computersysteme immer komplexer werden, muss die Steuerung auch immer leichter werden, damit der Computer für die breite Masse als Werkzeug zur Verfügung steht.

Viele Menschen haben keine Lust sich lange mit Bedienungsanleitungen zu beschäftigen. Ob das daran liegt, dass es ihnen von der Industrie vorgehalten wird, dass es nicht nötig ist oder es eine menschliche Eigenart ist, ist dabei nicht wichtig. Um den Kunden zu erreichen, sollen Anwendungen möglichst so designt sein, dass man an ihrer Darstellung und ihrem Aufbau ihre Funktion erkennen kann.

In unserem Alltag werden die Computer immer präsenter, ein Leben ohne Mikrocontroller die all die Geräte im Haushalt und in unserem Umfeld steuern ist kaum mehr vorstellbar. Dabei verschwinden diese auch immer weiter im Hintergrund und viele wissen nicht einmal, dass in bestimmten Geräten Computer versteckt sind. Daran lässt sich erkennen, dass der Mensch in der Lage ist mit den Computern zu interagieren, ohne zu wissen wie ein Computer intern aufgebaut ist.

Doch wie funktioniert es, dass der Mensch erkennt wie er ein Gerät benutzen kann ohne das er erst erlernen muss wie die internen AblÃ¤ufe funktionieren? Wie kann man etwas gestalten mit dem ein Mensch ganz intuitiv umzugehen weiß?

1.2 Ursprung des Projekts

Das EmotionBike-Projekt entstand ursprünglich um die Nutzung von Emotionen als zusätzliches Element für ein Mensch Computer Interface zu erforschen. Die Maschine sollte auf die emotionalen Zustände des Benutzers reagieren oder gezielt bestimmte Emotionen provozieren können um sein Interaktionsmöglichkeiten zu erweitern. Damit der Proband nicht bewusst oder unbewusst, durch sein Wissen das seine Emotionen beobachtet werden, Einfluss auf die Beobachtung nimmt soll er durch bestimmte Aufgabenstellungen abgelenkt werden. Es musste eine komplette Testumgebung gestaltet werden. Dazu wurde das EmotionBike-Game entwickelt. Der Proband fährt bei diesem Spiel durch eine Welt, welche den Fahrer vor verschiedene Aufgaben stellt. Diese Aufgaben sind so gestaltet um gezielt bestimmte Emotionen im Fahrer zu provozieren, wie zum Beispiel Stress, Frustration oder Schreck aber auch die gegenteiligen Emotionen wie Freude nach dem bestehen einer Aufgabe. Um zu ermöglichen, dass sich de Fahrer, gedanklich gut in das Spiel versetzen kann musste eine passende Schnittstelle zwischen dem Benutzer und dem Computer geschaffen werden. Hier wurde ein Ergometerfahrrad ausgewählt und so umgebaut, dass es sich als Controller für das Spiel nutzen lässt. Das Gefühl auf einem echten Fahrrad zu sitzen steigert die Immersion[1] und lässt den Fahrer leichter vergessen, dass er sich in einem Labor befindet und nicht durch einen Wald fährt.

1.3 Zielsetzung dieser Arbeit

Die Zielsetzung für dieses Projekt ist, das bereits bestehende Interface des EmotionBike-Projekts um eine Gangschaltung und eine Bremse zu erweitern. Diese Komponenten sollen das Fahrerlebnis des Benutzers realistisch erweitern. Die neuen Komponenten sollen intuitiv, also ohne Erklärung, zu bedienen sein. Die Komponenten müssen sich in die bestehenden Konzepte und die Architektur einpassen.

1.4 Vergleichbare Arbeiten

Gerade was den Bereich der, Mensch Computer Interaktion betrifft, gibt es eine große Zahl an Publikationen die sich mit dem Thema aus einander setzten. Besonders populär scheint dabei der Bereich der Gestenerkennung mit einer Kinect-Kamera zu sein.

[1] Als Immersion wird der Vorgang beschrieben, in dem eine Person eine fiktive Welt als real annimmt. Dabei verliert sie, zu einem gewissen Teil, das Bewusstsein für die reale Umgebung und gewinnt ein erhöhtes Bewusstsein für die Eindrücke einer nicht realen Welt. Bei einem Computerspiel zum Beispiel wird die Spielfigur als Teil der eigenen Person betrachtet. Dieser Effekt lässt sich auch beim Lesen von Büchern erleben, wenn das Gelesene eher als Film im Kopf und weniger als gelesene Worte wahrgenommen wird. Siehe auch Rittmann (2008).

Die Arbeit von Marie Schacht beschäftigt sich dazu im Kontrast mit dem Konzept von anfassbaren Benutzerschnittstellen [(Schacht, 2010)]. Dabei geht es darum, reale Gegenstände zur Bedienung eines Computersystems, zu benutzen. dabei wird darauf geachtet die Interaktion möglichst einfach und verständlich zu halten.

Hans Dieter Hellige beschreibt in seiner Arbeit die Anfänge der Computer Interaktion [Hellige (1998)]. Bei den Computern handelt es sich noch um mechanische Rechenmaschinen, welche von einigen Experten bedient werden. Die Benutzer konnten sehen, hören und fühlen wie die Maschine gearbeitet hat. Mit der Maschine zu arbeiten war also eine sehr direkte Angelegenheit. Hellig geht weiterhin darauf ein, das die modernen Interfaces eine Abstraktion des Computers darstellen. Es ist dem Benutzer nicht mehr offensichtlich, welche Abläufe durch seine Handlungen ausgelöst werden. Je stärker die Handhabung durch die Abstraktion vereinfacht wird, desto geringer ist die Transparenz in der Benutzung.

Die Arbeit von Michael Friedwald [Friedewald] setzt sich mit dem Aspekt auseinander, dass die Computer immer weiter in den Hintergrund treten und unmerklich in den Alltag des Menschen integriert werden. Es wird die Frage aufgeworfen, ob es sich bei der intuitiven Steuerung wirklich um ein benutzerfreundliches System handelt oder um die Gewöhnung des Benutzers an die Steuerung, welche diese einfach erscheinen lässt. Diese Frage wird auch in der Arbeit von Arne Bernin gestellt [Bernin (2011)].

Eine weitere sehr hilfreiche Arbeit ist von Carsten Bielmeier. Er hat sich im Rahmen seiner Staatsprüfung sehr ausgiebig mit den physikalischen Zusammenhängen des Fahrrads beschäftigt. Und dabei auch die wichtigen Grundlagen zusammengetragen, die in dieser Arbeit verwendet wurden[(Bielmeier, 2012)].

2 Analyse

2.1 Beschreibung des EmotionBike-Projekts

Im Zentrum des EmotionBike-Projekts steht ein, intuitiv zu bedienendes, Mensch Computer Interface. Das Ziel ist es, dem Benutzer Kontrolle zu ermöglichen ohne das erst etwas Neues erlernt werden muss. Ganz selbstverständlich kann die Schnittstelle zum Computer bedient werden. Dazu wird bereits Erlerntes übertragen und auf die gegenwärtige Situation angewendet.

Das EmotionBike ist ein Ergometerfahrrad. Dieses Fahrrad ist der Controller für eine virtuelle Umgebung. In dem Fall, dass der Fahrer bereits Erfahrungen mit Fahrrädern gemacht hat, wird er, ohne dass weitere Erklärungen notwendig sind, wissen, wie der Controller zu bedienen ist. Wenn der Benutzer ein Element des Controllers sieht, entsteht bei ihm anhand des Aussehens eine Erwartung über die Funktionalität. Er vergleicht das gesehene Element mit Dingen die er so oder ähnlich schon mal erlebt hat und kann somit aus seiner Erfahrung Rückschlüsse auf die Funktion des Elements ziehen. In seinem Kopf entsteht ein mentales Bild, welches ihm die Funktionsweise des Controllers zeigt. Die einzelnen Elemente haben einen hohen Wiedererkennungswert. Der Fahrer überträgt Erfahrungen aus anderen Bereichen, zum Beispiel das normale Fahrradfahren, auf das was er als Controller vor sich sieht.Vorrausgesetzt der Benutzer hat bereits Erfahrungen mit Fahrr“adern gesammelt, wird er die Pedale an dem Ergometerfahrrad wiedererkennen. Die Pedale an seinem Fahrrad ermöglichen es ihm die Geschwindigkeit seines Fahrrads zu verändern, also kann er davon ausgehen, dass auch die Pedale am Controller die Geschwindigkeit verändern wenn er sie mit den Füßen tritt. Die Stimuli, die das Ergometerfahrrad erzeugt, werden an das Spiel übertragen. Das Spiel reagiert entsprechend darauf, indem es beispielsweise den Avatar[1] beschleunigt, wenn der Fahrer in die Pedalen tritt. Auch wenn der Fahrer nicht genau weiß, welche Geschwindigkeit eine bestimmte Trittfrequenz zur Folge hat, wird er doch merken, dass das virtuelle Fahrrad schneller wird, wenn er beschleunigt. Wenn das Treten zu schwer wird, kann er an der Gangschaltung den Gang verringern. Das virtuelle Fahrrad wird daraufhin langsamer, das Treten jedoch leichter

[1] Als Avatar wird die virtuelle Präsenz einer Person in einer virtuellen Umgebung bezeichnet. Anstelle der eigenen Person wird der Avatar dazu benutzt mit der virtuellen Welt zu interagieren.

werden. Seine Erwartungshaltung wird erfüllt. Dabei ist es egal, was genau im Hintergrund, in der Maschine, passiert.

Etwas Ähnliches kann man auch sehen, wenn man sich die verschiedenen Arten betrachtet, mit denen man einen Computer steuern kann. Die traditionellen Mittel, um direkt auf den Computer einzuwirken, sind Tastatur und Maus. Dabei ist besonders die Maus eine recht abstrakte Sache. Sie wird hin und her geschoben, um an einem räumlich entfernten Ort einen Zeiger zu bewegen. Mit dem Klicken einer Taste kann man Dateien oder Fenster greifen. Die wenigsten Menschen sind sich der Computer internen Abläufe bewusst, welche von einem Mausklick ausgelöst werden. Der Mauszeiger funktioniert eher als eine Erweiterung des eigenen Körpers, um mit dem Computer interagieren zu können. Deutlich näher ist die Touch-Steuerung, bei der ich, zwar immer noch durch den Bildschirm eingeschränkt, direkt mit den Objekten interagieren kann. Ich drücke direkt auf ein Icon, bin also räumlich viel näher am Geschehen, und halte es so lange mit meinem Finger fest, bis ich sie an den Ort geschoben habe, an dem ich es haben möchte. Die Touchsteuerung verwendet ein weniger abstraktes Medium[2] , wodurch sich die Interaktion mit dem Computer viel direkter anfühlt. Der Lernaufwand für eine solche Steuerung richtet sich eher darauf zu verstehen wie das entsprechende Betriebssystem organisiert ist und weniger darauf die eigentliche Steuerung zu begreifen. Wie in der Arbeit von Arne Bernin [Bernin (2011)] angedeutet wird, ist der Begriff der Intuition eher als die Einfachheit mit dem Gerät umzugehen zu verstehen und weniger eine Ableitung von natürlichem Verhalten.

Das Spiel hat jedoch auch Einfluss auf den Controller. Wenn der Fahrer einen Berg hinauffahren möchte, wird das Treten durch die Steigung schwerer. Das alles soll den Fahrer vergessen lassen, dass er es mit einem Spiel zu tun hat. Das Verhalten der Spielumgebung soll auch dafÃ¼r sorgen, das der Fahrer nicht darüber nachdenken müss wie er bestimmte Handlungen ausführt, sondern bereits erlernte einsetzen. Dadurch soll der Fahrer übersehen, dass er sich nur in einer Spielwelt bewegt und sich in das erlebte hinein versetzen. Wenn die Immersion gut genug ist, wird er sich natürlich verhalten.

Weiterhin hat das EmotionBike eine Reihe von Sensoren um den Zustand des Fahrers zu beobachten. Es gibt eine Kinect-Kamera, welche ein Tiefenbild des Gesichts des Probanden aufnimmt. Das dient dazu, die Mimik des Fahrers zu erfassen. Es gibt auch Puls- und Temperaturmesser, sowie EDA (elektrodermale Aktivität) und EEG (Elektroenzephalografen). Diese erfassen den Körperlichen Zustand des Fahrers. Mit diesen Daten wird ein Bild der Emotionen des Fahrers gezeichnet.

[2] Medium ist durchaus so zu verstehen, als das etwas darüber übertragen wird. Eine Aktion des Fahrers, wie das Drehen des Lenkers, wird über das Benutzerinterface in die Spielwelt übertragen. Einflüsse aus der Spielwelt wirken über das Medium auf den Fahrer ein. Ziel ist es, dieses Medium so unmerklich wie möglich zu gestalten.

2.2 Aufbau des EmotionBike-Projekts

Wie weiter oben und in den Arbeiten von Jonas Hornschuh [Hornschuh (2015)] und Sebastian Zagaria [Zagaria (2015)] beschrieben, besteht das Projekt aus mehreren Komponenten, welche fortschreitend erweitert werden. Zu diesen Komponenten gehören verschiedene Sensoren, welche das Verhalten des Fahrers erfassen und weiterleiten. Weiterhin gibt es eine Reihe von Modulen, welche die Sensordaten auswerten und an das EmotionBike-Game weiterleiten oder für statistische Erhebungen und Emotionsmessungen verarbeiten. Der Fahrer wird während der Fahrt, mit Audiodaten, gefilmt um später die Reaktionen zu bestimmten Zeitpunkten nachvollziehen zu können. Zusammen mit den anderen Daten, wie der Atemfrequenz, dem Puls und einer Emotionserkennung anhand des Gesichtes lässt sich ein komplexes Bild über den Zustand des Fahrers zeichnen, während sich dieser durch die Spielwelt bewegt.

Es soll möglichst einfach sein neue Komponenten dynamisch in das Gesamtsystem zu integrieren. Aus diesem Grund wurde zum Datenaustausch ein Messagebroaker[3] gewählt. Hier können sich die diversen Module gezielt auf einzelne Kanäle anmelden, von denen sie bereits wissen, dass dort Nachrichten verbreitet werden, welche für dieses Modul wichtig sind. Das gewählte Publisher-Subscriber-System hat dabei den Vorteil, dass die Nachrichten an alle Module gesendet werden, welche sich auf einem Kanal angemeldet haben. Im Idealfall ist jedes Modul ein in sich vollständig geschlossenes System, welches ohne Probleme aus dem System entfernt oder hinzugefügt werden kann. Bei einer Änderung sollte keines der anderen Module modifiziert werden müssen. In der Realität ist dies jedoch nicht immer umzusetzen.

Ein weiterer Vorteil ist das nur wenige Komponenten auf die Neuen Nachrichten angepasst werden müssen. Diese Komponenten sind vor allem die Sammelstellen, welche alle anfallenden Daten sammeln und sie für eine spätere Untersuchung speichern. Hier werden die Daten zusammengeführt und synchronisiert. Später kann man sich alle Daten zu einem bestimmten Zeitpunkt betrachten oder die gesamte Fahrt eines Probanden nachverfolgen um zu prüfen ob die Provokation bestimmter Emotionen durch das Level Design erfolgreich war. Außerdem können Korrelationen zwischen verschiedenen Messwerten und Spielereignissen leichter erkannt und nachvollzogen werden. Um die Daten synchronisiere zu können wird eine bestimmte Nachricht über das Netzwerk geschickt, die den Beginn der Aufzeichnungen markiert. Dieses Vorgehen umgeht das Problem der unterschiedlichen lokalen Zeiten auf den einzelnen Modulen, wodurch sich die Daten nicht sauber durch einen Timestamp synchronisieren lassen. Das letzte Problem ist hier noch das Delay durch das Netzwerk, sowie die Verarbeitungszeit auf den einzelnen Modulen.

[3] Bei dem Messagebroaker handelt es sich um ActivMQ, auf den im Kapitel ÄctivMQ, Messagebroker"noch genauer eingegangen wird.

Um auch die Video- und Audioaufzeichnung mit den restlichen Daten in Einklang bringen zu können, wird von diesem Modul auch eine LED und ein Lautsprecher angesteuert. Die LED ist im Bild der Kamera zu sehen und der Lautsprecher gibt einen Ton von sich, welches auf der Audioaufnahme zu hören ist. Das Leuchten der Lampe und das Ertönen der Lautsprecher markieren den Erhalt der Synchronisationsnachricht über das Netzwerk. F“ur die Anlayse können diese beiden Signale erkannt und das Video und Audiomaterial mit den anderen Messdaten synchronisiert werden.

2.2.1 Struktureller Aufbau des Projekts

Der Großteil des Projektaufbaus orientiert sich am Model-View-Controll-Pattern [Zagaria (2015)]. “Das MVC-Paradigma wird durch drei Objekte umgesetzt. Das Model-Objekt stellt das Anwendungsobjekt dar, das View-Objekt seine Bildschirmrepräsentation, und das Controll-Objekt bestimt die Möglichkeiten, mit denen die Benutzerschnittstelle auf Benutzereingaben reagieren kann.“Erich Gamma (1996).

Das View-Objekt ist die direkte Schnittstelle zwischen dem Benutzer und dem System. Das Model-Objekt ist eine Abbildung des Systemzustandes. Hier werden alle relevanten Informationen gespeichert, um eine zentrale Sammelstelle für alle Informationen zu haben. Das Controller-Objekt hängt zwischen den anderen beiden Komponenten. Es ist dafür verantwortlich, die Sensordaten zu verarbeiten, den Systemzustand entsprechend anzupassen und die Reaktionen des Systems an das View-Objekt weiterzureichen. Das Projekt selbst verfügt über mehrere Module, welche sich alle einer oder mehrerer Schichten des MVC Patterns zuordnen lassen.

Jetzt ist es streitbar, in welchen Bereich das Ergometerfahrrad fällt. In der Arbeit von Zagaria [Zagaria (2015)] wird es zum Model gezählt, da es Daten an das Controll-Objekt zur Verfügung stellt und den Zustand des Fahrrads abbildet. Da das Fahrrad jedoch auch die Benutzereingabe entgegen nimmt, und an das Controll-Objekt weiterleitet und auch ein Feedback an den Spieler weitergibt, kann es auch als View-Objekt betrachtet werden, wie in Abbildung 2.1 zu sehen ist.

2.3 Aufgabenstellung dieses Arbeit

Die Aufgabe in dieser Arbeit besteht darin, wie oben bereits beschrieben, das bereits bestehende Human-Computer-Interface um weitere Funktionalitäten zu erweitern und dadurch die Steuerung sowie die Immersion für den Benutzer zu verbessern. Zu diesem Zweck soll das als Controller fungierende Ergometerfahrrad um eine Bremse und eine Gangschaltung erweitert werden.

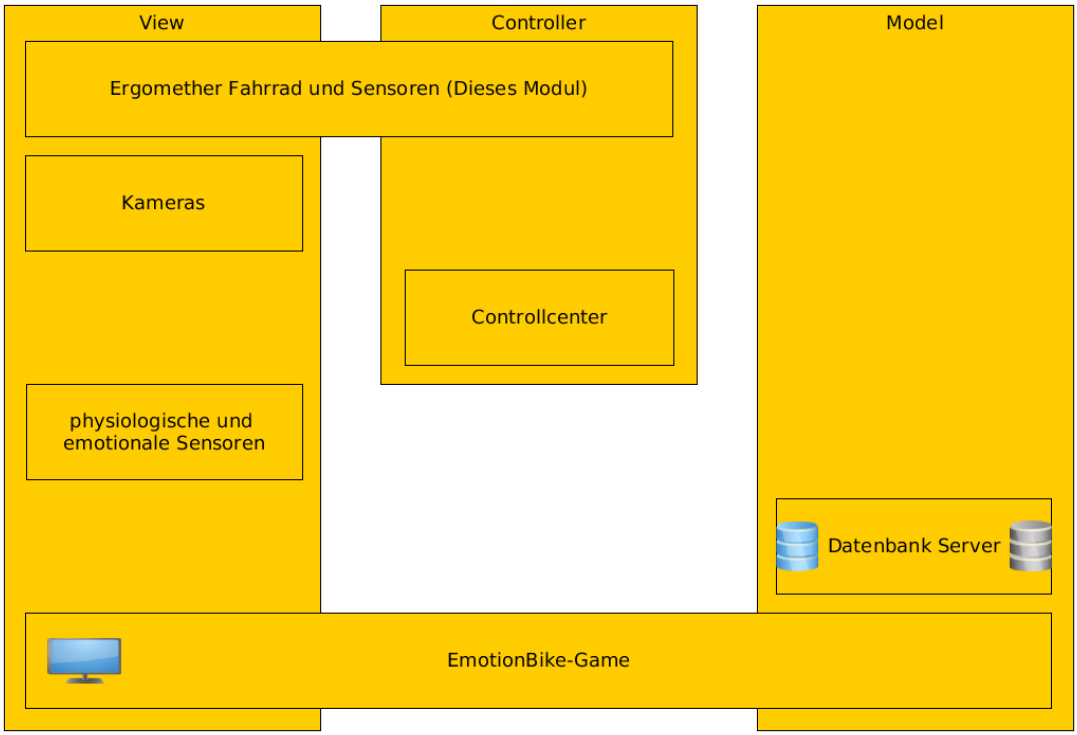

Abbildung 2.1: Darstellung der Projektmodule und ihrer Rollen im MVC-Pattern

Die Bremse und die Gangschaltung sollen sich wie ihre echten Gegenstücke verhalten. Die Funktionalität dieser Objekte wird digital umgesetzt, da es am Ergometer keine Hardware für eine Bremse oder eine Gangschaltung, wie an einem Fahrrad, gibt. Für eine SoftwarelÃ¶“osung muss ein Physikmodell erstellt werden, welches die Bremse und die Gangschaltung realistisch darstellt und die entsprechenden Auswirkungen berechnen kann. Diese Informationen müssen an das EmotionBike-Game weitergegeben werden, damit der Fahrer mit diesen Komponenten auch Einfluss nehmen kann.

Das neue Modul muss dabei intuitiv bedienbar sein und darf die Immersion des Benutzers nicht stören. Sowohl die Bremse als auch die Gangschaltung müssen sich zu diesem Zweck erwartbar verhalten. Das bedeutet, das sie sich deterministisch Verhalten müssen. Der Benutzer muss erkennen können, nach welchen Gesetmäßigkeiten sich das System verhält und diese müssen immer eingehalten werden. Der Benutzer entwickelt gegenÃ¼ber dem system eine Erwartungshaltung, welche sich aus seinen Erfahrungen ableitet. Die Erwartungen an das Verhalten müssen erfüllt werden. Andernfalls sind die Ergebnisse der Handlungen nicht vorhersagbar und das System ist nicht benutzbar. Um dem Benutzer eine möglichst leichte

Handhabung zu ermÃ¶glichen, soll das Modul bereits an die Erwartungshaltung des Benutzers aus vorherigen Erfahrungen angepasst sein. So muss der Benutzer nicht erst neue ZusammenhÃ¤nge erlernen. Weiterhin muss das System den Echtzeitanforderungen genügen, damit keine störenden Wartezeiten und Verzögerungen entstehen. Um zu erreichen, dass sich das Modul erwartbar und natürlich verhält, muss es eine physikalisch korrekte Abbildung des Verhaltens geben. Das Design und die Handhabung der einzelnen Komponenten muss ihren realen Vorbildern entsprechen. Dadurch kann der Fahrer die Resultate seiner Handlung abschätzen. Weiterhin muss sich das Verhalten des Moduls auch in die bestehende Spielwelt einfügen, um ein rundes Gesamtbild zu erzeugen. Das bedeutet wiederum, dass die physikalische Abbildung angepasst werden muss, um ein besseres Gefühl beim Fahrer zu erzeugen. Damit der Benutzer mit dem System interagieren kann muss es eine Schnittstelle geben, welche er benutzen kann. Der Fahrer soll anhand dieser Schnittstelle erkennen können, wie diese zu bedienen ist und welchen Einfluss sie auf das Spielgeschehen haben wird. Im Idealfall kann er seine eigenen Fahrradfahrerfahrungen auf die neu erstellten Controllerkomponenten übertragen und die Gangschaltung sowie Bremse ohne Erklärungen bedienen.

Damit die neue Komponente mit den anderen Modulen des Projekts, allen voran das EmotionBike-Game, kommunizieren kann, ist es wichtig sich in die bestehende Architektur einzupassen. Das bedeutet vor allem die Nutzung der etablierten Middleware und Protokolle zur Nachrichtenübermittlung. Weiterhin muss darauf geachtet werden, die Funktion anderer Module nicht zu stören.

2.4 Verwendete Hardware

2.4.1 Gangschaltung und Bremse

Die Gangschaltung und die Bremse sollen wie oben beschrieben intuitiv bedienbar sein. Sie müssen also vom Aussehen und von der Bedienung auf ihre Funktion hindeuten, indem sie sich an bereits bekannten Mustern orientieren. Aus diesem Grund wurde eine echte Gangschaltung und eine echte Bremse gewählt. Da das Ergometer jedoch nicht über solche Geräte verfügt und auch eine entsprechende Funktionalität nicht in dem gewünschten Maße vorsieht, mussten sie neu an dem Lenker montiert werden.

Zum ermitteln der ZustÃ¤nde musste das Ergomether um Hardwarekomponenten erweitert werden. Zu diesem Zweck wurden zwei Potentiometer am Lenker montiert und über Seilzüge mit der Bremse und der Gangschaltung verbunden. Der Zustand der Potentiometer lässt sich nun über die Gangschaltung und die Bremse manipulieren.

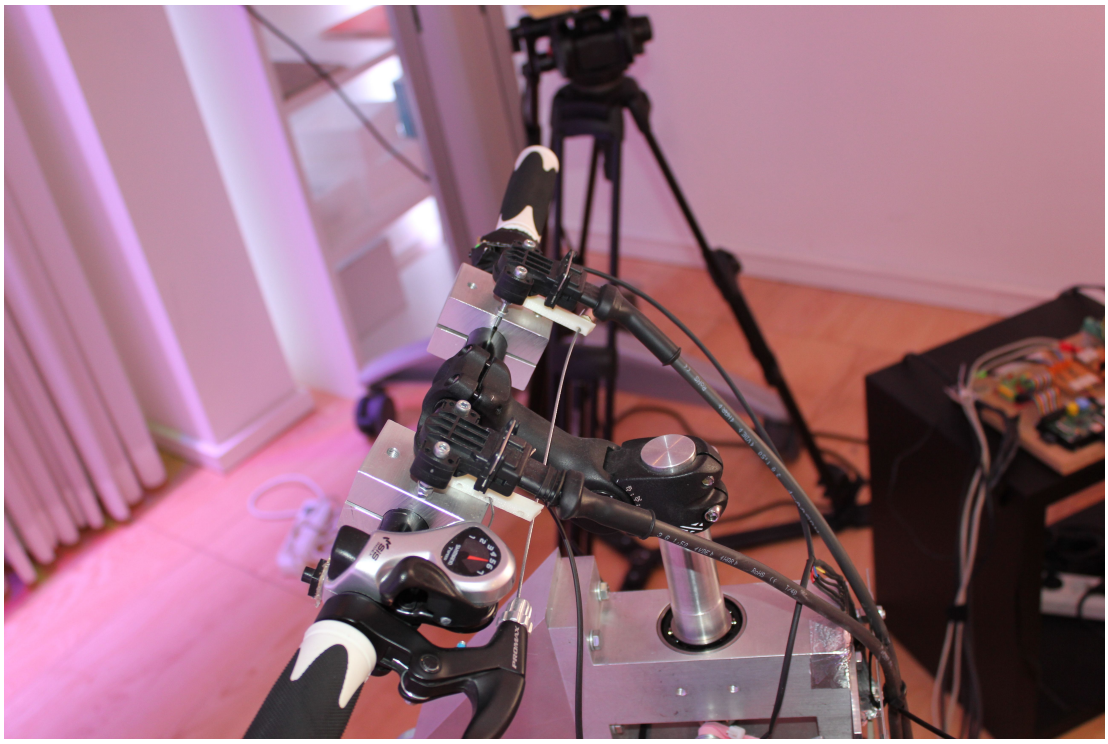

Abbildung 2.2: Der Lenker, die Gangschaltung und die Bremse sind über Bremszüge mit dem Potenziometern verbunden.

2.4.2 Potentiometer

Ein Potentiometer ist ein Spannungsteiler mit zwei Variablen Widerständen. Bei einem Spannungsteiler werden zwei Widerstände eingesetzt, um eine Spannung zu reduzieren. Die beiden Widerstände sind in Reihe geschaltet. Die gesamte Spannung, die an das Bauteil angelegt wird, fällt über die beiden Widerstände ab. Über die einzelnen Widerstände fällt jedoch nur ein Anteil der gesamt Spannung ab. Die Menge des über einen Widerstand abfallenden Stroms entspricht dabei dem Anteil des Widerstandes am gesamt Widerstand.

Das Potentiometer funktioniert genauso, nur das die beiden Widerstände variabel sind. Die Versorgungsspannung wird bei A angelegt und fällt über den Widerstand 2 komplett nach B ab. Mit einem Schleifkontakt 3 wird dieser Widerstand abgefahren. Dadurch verändert sich die Größe der Widerstände vor und nach dem Kontakt, vergleiche Abbildung 2.3. Der Schleifkontakt wird von außen über einen Drehregler oder einen Hebel gesteuert. Jetzt kann man die Spannung messen, welche über einen der beiden Widerstände abfällt und Rückschlüsse

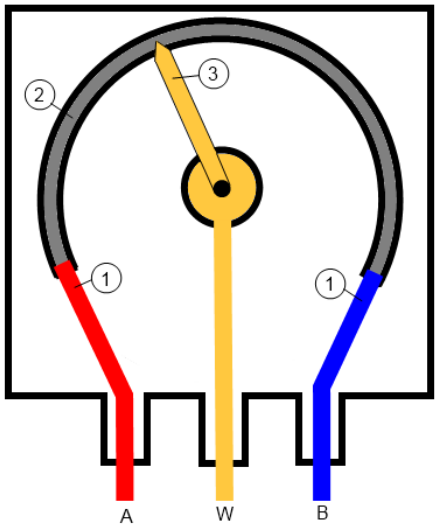

Abbildung 2.3: Darstellung des Aufbaus eines Potentiometers [Inc (2013)]

daraus ziehen, wie weit der Regler gedreht ist bzw. auf welcher Position er im Augenblick steht. Siehe auch Inc (2013).

Die Widerstände lassen sich dabei quasi stufenlos einstellen, die Rastehrung ist nur von dem Messgerät abhängig, mit dem der Spannungsabfall gemessen wird. Die Kontakte A, W und B werden mit dem Raspberry Pi verbunden. Dabei Sind A und B die Quellspannung und die Masse und W wird mit dem Analog-Digital-Konverter verbunden. Dieser erzeugt aus der analog anliegenden Spannung einen digitalen Wert, welcher durch Software verarbeitet werden kann.

2.4.3 Raspberry Pi

Als zentrale Berechnungseinheit des Moduls wurde der Raspberry Pi gewählt. Hierbei handelt es sich um einen mini Computer, welcher über ein Betriebssystem, welches auf Linux basiert, verfügt. Er ist mit einer Reihe IO-Pins, USB Anschlüssen, einem Ethernet Anschluss, einem Audioausgang sowie weiter Anschlüssen ausstattet. Das Betriebssystem unterstützt die Netzwerkanbindung sowie den Sound Ausgang. Über die IO Pins wird das Gertboard angeschlossen um die Peripherie der Raspberry Pis um weiter Funktionen zu erweitern.

Gertboard

„Das Gertboard wurde vom Broadcom-Mitarbeiter Gert van Loo entworfen und ist daher die »offiziellste« RaspberryPi-Erweiterungsplatine“ Monk (2013). Es ist bereits mit einer ganzen Reihe an Peripherie Anschlüssen ausgestattet, darunter IO-Pins, welche benutzt werden, um die

Buttons auszulesen, sowie die Analog-Digital-Konverter (ADC). Der ADC wird dazu benutzt, aus einem analogen Signal einen digitalen Wert zu erzeugen.

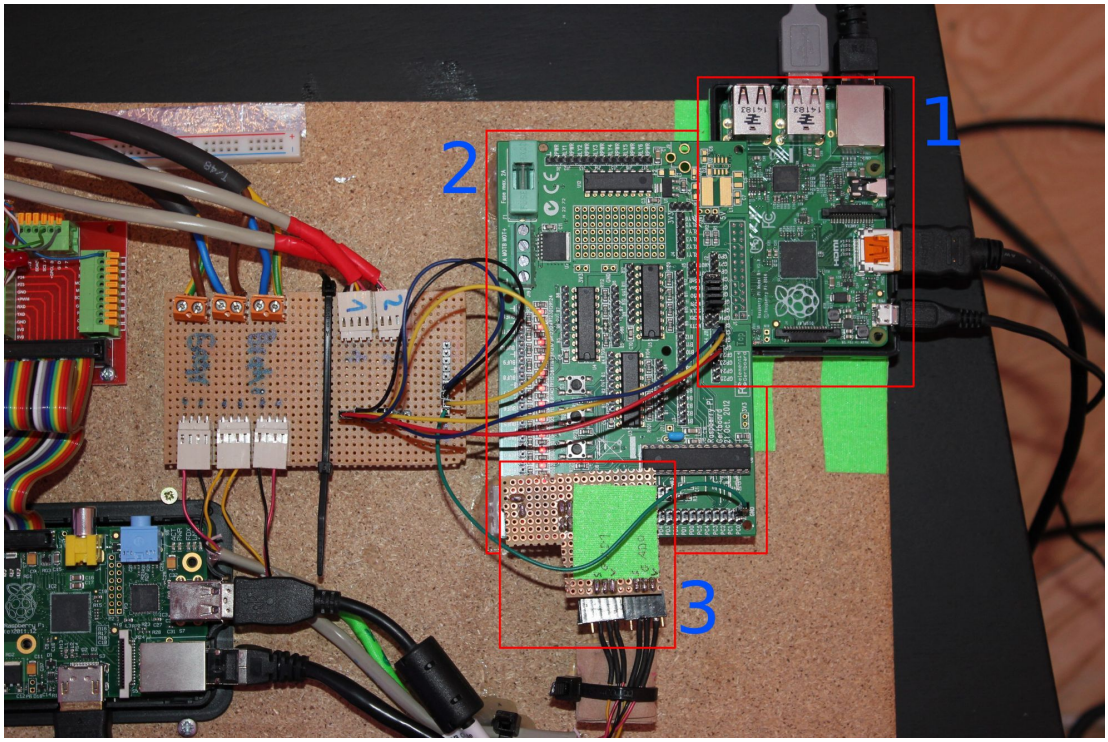

Abbildung 2.4: Der Raspberry Pi(1) und das Gertboard(2) mit dem ADC(3).

Die WiringPi Library unterstützt die Verwendung des Gertboards, was die Kommunikation mit der erweiterten Peripherie erleichtert.

WiringPi

Die WiringPi Library ist ein Set aus Methoden und Routinen, welche die Funktionen des Gertboards kapseln und eine einfache Schnittstelle zur VerfÃ¼gung stellen. Sie wurde in C programmiert und es gibt für die häufigsten Programmiersprachen Wrapper, welche die Verwendung ermöglichen.

Die Library wird unter anderem dafür verwendet, um die GPIO-Pins zu konfigurieren, sie also als Aus- oder Eingang zu verwenden. Weiterhin werden über WiringPi die Signale auf die Pins gelegt und ausgelesen. Die Bibliothek unterstützt auch die Verwendung des Gertboards. Das hat zu Folge, dass sie auch in der Lage ist spezielle Peripherie anzusprechen und beispielsweise LEDs und Taster zu verwenden.

Analog-Digital-Konverter

Der Analog-Digital-Konverter des Gertboards wandelt das analog anliegende Signal in eine 10-Bit Zahl um. Es wird also eine Zahl zwischen 0 und 1023 erzeugt. Diese Zahl repräsentiert ein Spannung zwischen 0 und 3,3 Volt. Dieser Wert ergibt sich aus der am ADC angelegten Spannung. Als Spannungsquelle kann das Potenziometer angeschlossen werden und erlaubt somit Rückschlüsse auf den Zustand des Gerätes welches das Potentiometer manipuliert. Der ADC ist also eine der Schnittstellen zwischen der Hardware und der Softwareseite des Mensch-Computer-Interfaces.

2.5 ActivMQ, Messagebroker

Die Übermittlung der Nachrichten im System wird mittels des Messagebroakers ActivMQ realisiert. Dieser wird von Apache angeboten. Diese Methode der Nachrichtenübermittlung ist bereits im EmotionBike-Projekt etabliert und dieses Projekt übernimmt die Vorgabe, diesen Messagebroker zu verwenden.

Der Messagebroaker hat die Aufgabe Nachrichten im System zu verteilen. Dazu werden sie von einem Produzenten erzeugt und an einen oder mehrere Teilnehmer verteilt. Es gibt eine breite Unterstützung für verschiedene Programmiersprachen, wodurch auch Module, welche in unterschiedlichen Sprachen verfasst wurden, Nachrichten austauschen können. Im Rahmen des EmotionBike-Projekts sind das vor allem C++ und C#.

Es gibt zwei Möglichkeiten, wie die Nachrichten verteilt werden können, wobei Beide immer einen Produzenten, welcher Nachrichten erzeugt und verschickt, und mindestens einen Konsumenten, welcher Nachrichten empfängt und verarbeitet, kennt, siehe auch The Apache Software Foundation (2004).

Queue

Im Queue Modell werden die Nachrichten von einem Producer erzeugt und in einer Queue eingereiht. Alle Consumer werden darüber informiert, dass eine neue Nachricht vorhanden ist. Der Consumer, welcher am schnellsten reagieren kann, entnimmt der Queue die Nachricht und kann sie verarbeiten. Dabei wird die Nachricht jedoch aus der Queue entfernt und kein anderer Consumer hat noch die Chance sie zu lesen.

Topic

Im Topic Modell werden Nachrichten von einem Publisher erzeugt und verteilt. Ein Subscriber kann die Nachricht empfangen und verarbeiten. Im Gegensatz zum Queue Modell werden die Nachrichten nicht gelöscht, sondern allen Subscribern zur Verfügung gestellt.

Im EmotionBike-Projekt wird das Topic Modell verwendet. Das hat die Vorteile, dass Nachrichten welche an mehrere Ziele gesendet werden müssen nicht speziell sicherstellen müssen, dass sie von jedem Konsumenten empfangen werden. Weiterhin ist es einfacher neue Module ein zu bauen. Das entsprechende Modul beginnt auf allen relevanten Topics zu lauschen und zu senden. Relevante Nachrichten werden von dem neuen Modul verarbeitet und alle anderen ignoriert. Das System hat jedoch auch den Nachteil das, für ein Modul nicht relevante, Nachrichten empfangen werden. Diese Nachrichten werden trotzdem verarbeitet und müssen ignoriert werden. Das kann zu hohem Aufwand bei den Modulen führen.

2.6 JSON-Protokoll

Um die Reibungslose Kommunikation sicher zu stellen, ist ein Protokoll notwendig, welches eine einheitliche Kommunikation gewährleistet. Das ist besonders dann wichtig, wenn leicht neue Module hinzugefügt werden sollen. Dieses Protokoll legt den Aufbau der Nachrichten fest. In diesem Projekt wurde das JSON-Protokoll als Standard festgelegt.

JSON ist ein Daten-Austausch-Format, welches auf einer Key Value Map basiert. In diesem Format werden die Werte einem bestimmten Key zugeordnet. Man kann die Map nach einem bestimmten Key befragen und bekommt den entsprechend eingetragenen Wert zurückgegeben. Die Werte sind entweder Strings, nummerische Werte, Arrays, Key Value Maps oder die drei primitiven Zustände true, false und null, vergleiche The Jason Groupe.

Dadurch, dass sich auch weitere Key Value Maps als Wert übertragen lassen, ist es möglich geschachtelte Objektstrukturen auf eine JSON Nachricht abzubilden. Jedes Objekt innerhalb des ersten, zu übertragenen Objekts, ist eine weitere Map, welche das Objekt darstellt. Je größer diese Nachrichten werden, desto größer ist jedoch auch der Aufwand, sie zu erzeugen, über das Netzwerk zu schicken und dann wieder zu parsen.

3 Systemdesign

3.1 Sense-Plan-Act-Pattern

Das Sense-Plan-Act-Pattern findest hauptsächlich in der Robotik Anwendung. Es basiert darauf, dass das Steuersystem in Module aufgeteilt wird, welche unterschiedlichen Aufgaben wahrnehmen und mit einander in Verbindung stehen. Diese Module sind hierarchisch organisiert und jedes Modul produziert Informationen die nach unten weitergegeben werden. Der Ablauf wird im Diagramm 3.1 beschrieben.

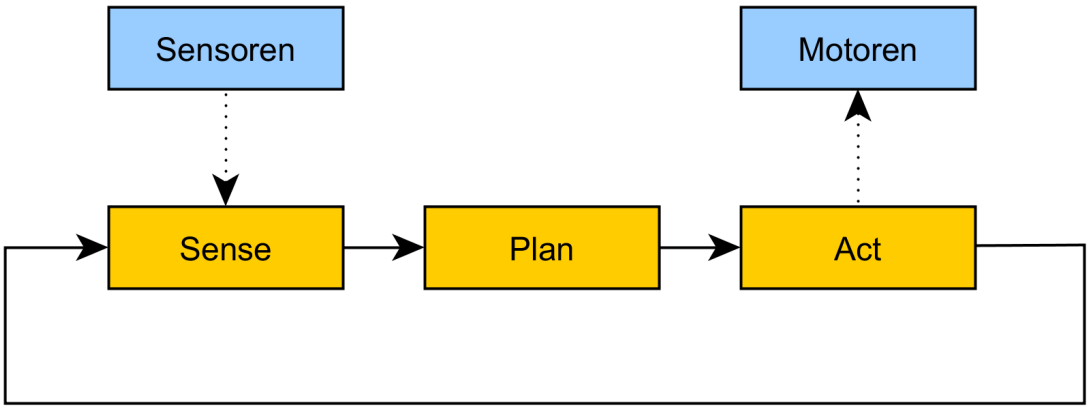

Abbildung 3.1: Sense-Plan-Act Ablaufdiagramm

Das Sense-Modul ist die Schnittstelle zu den Sensoren. Hier werden die Zustände der Sensoren und anderer Inputs, wie Netzwerknachrichten, erfasst. Die Informationen werden an das Plan-Modul weitergeleitet.

Das Plan-Modul besteht aus den Daten des Systemzustands sowie den Sensorwerten, welche vom Sense-Modul gesammelt wurden. Aus diesen Daten wird ein Ist-Zustand des Systems und seiner Umgebung generiert. Abhängig von diesen Informationen wird das weitere Vorgehen berechnet werden. Die so generierten Anweisungen werden an das Act-Modul weitergegeben.

Das Act-Modul bildet die ausführende Hardwareschnittstelle und führt die entsprechenden Aktionen aus. Die Anweisungen, welche von dem Plan-Modul generiert werden, können dabei auch schon konkret sein. Die genaue Umsetzung ist jedoch im Act-Modul gekapselt.

Die Reihenfolge in der diese drei Module nacheinander aufgerufen werden ist dabei fest. So entsteht die Abfolge aus Erfassen, Planen und Ausführen. Vergleiche dazu Murphy (2000).

Für dieses Projekt wurde das Modell als Vorlage gewählt. Es bietet im Vergleich zu anderen Pattern, wie zum Beispiel Model-View-Controller, den Vorteil das es einen linearen Informationsfluss gibt. Da die Module über das Netzwerk miteinander kommunizieren würde wechselseitige Kommunikation viel drastischer durch die Latenz beeinflusst werden als ein unidirektionaler Informationsfluss.

Ein weiter wichtiger Punkt ist, dass dieses Projekt keine richtige View-Instanz hat. Es soll zwar Benutzereingaben aufnehmen und verarbeiten, dies geschieht jedoch über Sensoren, welche in jedem Berechnungszyklus abgefragt werden. Eine Umstellung der Darstellung auf eine andere View ist nicht vorgesehen, weswegen ein Pattern das bidirektionale Kommunikation enthält eher ungeeignet ist.

3.2 Architektur des Systems

Das hier beschriebene System orientiert sich im Aufbau am, oben beschriebenen, Sense-Plan-Act-Pattern. Wie das Designschema vorgibt teil sich das System in drei Teile: Sense, Plan und Act. Um eine möglichst große Flexibilität zu erhalten, werden die drei Module als getrennte Klassen entwickelt und in unterschiedliche ausführbare Programme aufgeteilt. Das führt dazu, dass das Sense- und das Act-Module auf einem embedded System, wie dem Raspberry Pi, ausgeführt werden können. Der womöglich rechenaufwändige Plan-Part kann entweder auf dem gleichen Gerät gestartet werden oder auf einem leistungsstärkeren ausgelagertem Computer. Die Kommunikation findet dabei über den ActivMQ Messagebroker statt. Wie in der Abbildung 3.2 zu sehen ist, sind die Drei Module über die ActivMQ-Topics “Topic_SP“ und “Topic_PA“ verbunden. Diese wurden neu generiert und werden sonst nicht im Projekt verwendet. Sie dienen nur dafür um die interne Kommunikation zwischen Sense, Plan und Act zu realisieren.

Die Aufteilung der Module auf verschiedene Systeme hat jedoch auch zu Folge, dass die strikte Einhaltung des Ablaufes nicht gegeben ist. Das Sense-Modul liest selbst, zyklisch, die Sensoren aus und schickt die Werte über das Netzwerk an das Plan-Modul. Das Act-Modul hingegen reagiert immer dann, wenn es über das Netzwerk eine Anweisung erhält. Im Plan Modul verhält es sich so, dass zyklisch der Speicher abgefragt wird, in dem die Sensorwerte

abgelegt sind. Daraus werden die nächsten Anweisungen berechnet. Jedoch besteht in Plan selbst eine Miniatur des Sense-Plan-Act-Patterns. Man kann annehmen, dass die Instanz, welche die von Sense gesendeten Nachrichten aufnimmt, selbst einem Sense-Modul entspricht. Und das Objekt, welches die Nachrichten an Act sendet, steht für das Act Modul.

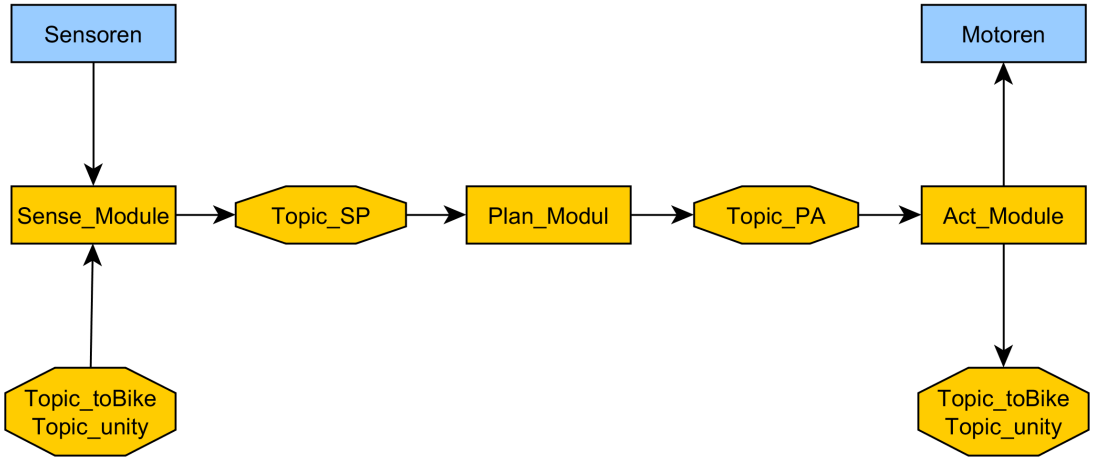

Abbildung 3.2: Modul Darstellung mit ActivMQ Topics

3.2.1 Sense

Das Sense-Modul besteht aus dem Sensor_Checker, welches wiederum aus einer ActivMQ_Verbindung sowie einem Sensor_Interface besteht, siehe Abbildung3.3.

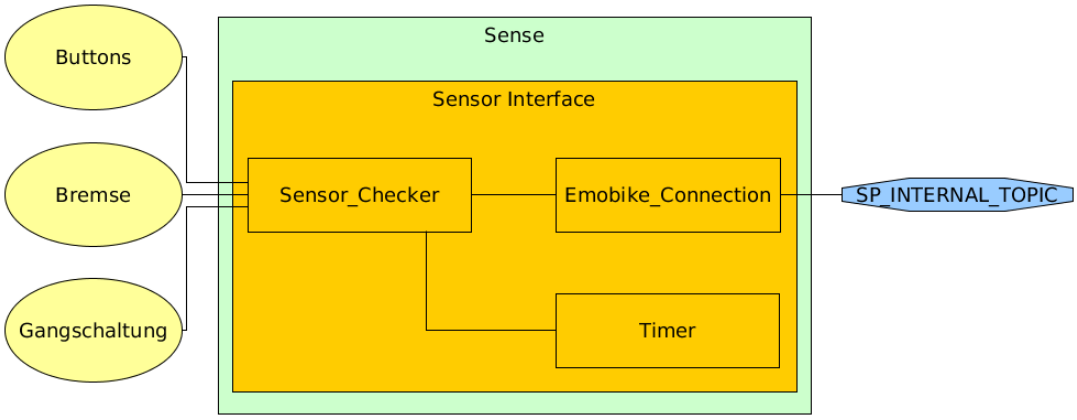

Abbildung 3.3: Aufbau Sense_Module

Das Sensor_Interface enthält einen Timer, welcher die periodische Abfrage der Sensoren auslöst. Die Sensordaten werden mithilfe der WiringPi Library abgefragt. Dies ist notwendig, da sie an das Gertboard angeschlossen sind. Die abgefragten Sensoren sind die drei Buttons und zwei Potentiometer, welche die Bremse und die Gangschaltung messen. Die Daten werden an die Emobike_Connection geschickt und von dort aus, über das interne ActivMQ Topic (SP_INTERNAL_TOPIC), an Plan weitergeleitet.

Es gibt verschiedene Möglichkeiten die Zustände der Sensoren zu überwachen. Man kann entweder auf den Eingang horchen und bei jeder Änderung des Zustandes einen Interrupt generieren, um kein Sensorergebnis zu verpassen oder man prüft periodische den Zustand des Sensors ab. Dabei geht man jedoch das Risiko ein, Änderungen zu verpassen. Für dieses Modul wird die periodische Messung verwendet. Dabei wird in einem festen Intervall der Zustand der Sensoren erfasst, geprüft ob er sich im Vergleich zur letzten Messung verändert hat und gegebenenfalls an Plan weiter geschickt. Dieses Vorgehen ist besonders für die Potentiometer geeignet, da hier keine Veränderung beobachtet werden muss, sondern nur die absoluten Werte interessant sind. Anders als bei einem Inkrementalgeber gibt der Wert eines Potentiometers seinen aktuellen Zustand wieder und muss nicht permanent überwacht werden. Ein Inkrementalgeber[1] hingegen arbeitet mit relativen Werten, bei denen keiner übersehen werden darf, da man sonst nicht mehr sicher sagen kann wie der aktuelle Zustand des Sensors ist, vergleiche dazu Hornschuh (2015). Ein weiterer wichtiger Punkt ist, dass die Potentiometer nicht absolut genaue Werte produzieren [2]. In Abhängigkeit von den verwendeten Anschlüssen, Stromleitungen und der erfassenden Hardware kommt es zu Schwankungen in den Endergebnissen. Diese Schwankungen der Signalstärke sind meistens recht klein, würden aber jedes Mal einen Interrupt auslösen. Dies würde einen erheblichen Rechenaufwand provozieren und womöglich dafür sorgen, dass die Interrupt-Service-Routinen permanent durch neue Interrupts unterbrochen werden.

Für die Buttons könnte man argumentieren, dass es sinnvoll wäre, diese über einen Interrupt ab zu handeln, da hier nur zwei feste Zustände verarbeitet werden müssen. Jedoch ist das Intervall so klein gewählt, dass es sehr unwahrscheinlich ist, dass der Benutzer den Knopf drückt und wieder loslässt, bevor der Sensorzustand abgefragt wird. Aus diesem Grund kann auch hier eine periodische Zustandsabfrage vorgenommen werden. Um das besonders schnelle,

[1] Ein Inkrementalgeber arbeitet in der Regel mit vier eindeutigen Zuständen, von den jeder in seine zwei benachbarten Zustände überführt werden kann. Diese Zustände sind so angeordnet, dass es eine feste, ringförmige Kette gibt, welche in beide Richtungen durchlaufen werden kann. In dem ich mir jede Überführung in einen neuen Zustand angucke, kann ich entscheiden in welche Richtung diese Kette durchlaufen wird und Rückschlüsse darauf ziehen wie der Inkrementalgeber gedreht wird. Wichtig ist dabei, dass ich wissen muss in welchem Zustand ich mich befinde und welche Transition zu einem neuen Zustand durchgeführt wird.

[2] Dies konnte während der praktischen Arbeit durch Messungen ermittelt werden.

mehrfache Drücken nicht zu verpassen, könnte man die Abtastrate erhöhen, dies würde jedoch nur dann sinnvoll sein, wenn auch das Plan-Modul entsprechend hoch getaktet wird oder die Buttons auch per Interrupt verarbeiten würde. Ansonsten würde das mehrfache Drücken wie ein einfaches Drücken verarbeitet werden. Wenn ein Benutzer also einen Button öfter als einmal pro Abtastzyklus drückt, würde also Datenverlust entstehen. Da die Buttons jedoch eher ein historisches Überbleibsel sind, wurde auch hier der periodische Ansatz wie für die Potenziometer gewählt.

3.2.2 Plan

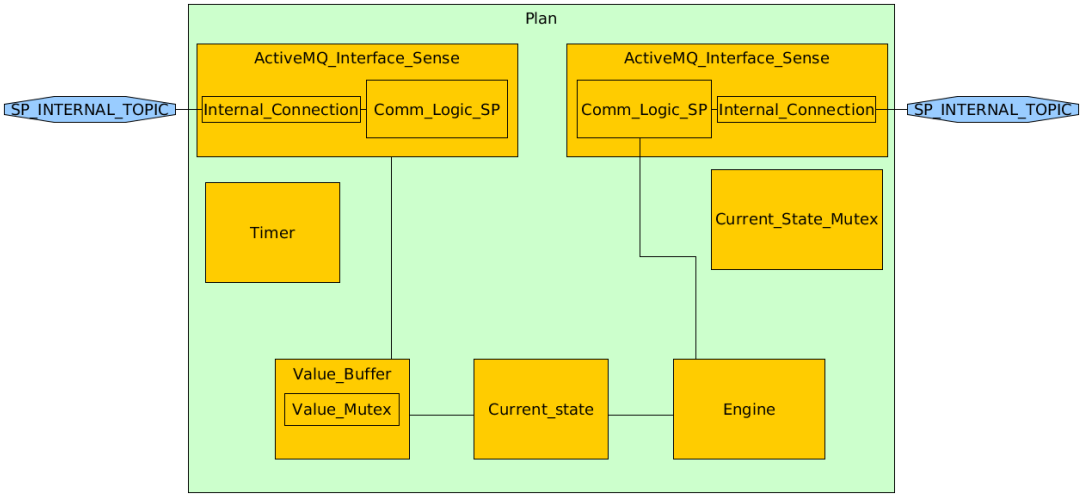

Abbildung 3.4: Aufbau Plan_Module

Das Plan_Module besteht aus zwei ActivMQ_Interfaces, einem Value_Buffer, einem Current_State sowie der Engine, siehe Abbildung 3.4. Die beiden ActivMQ_Interfaces dienen zur Kommunikation über den ActivMQ_Messagebroker. Über diese beiden Schnittstellen wird die Kommunikation mit Sense und Act sichergestellt, sowie Nachrichten von anderen Modulen empfangen. Die Kommunikation zu Sense und Act findet ausschließlich über die beiden internen Topics “SP_INTERNAL_TOPIC“ und “PA_INTERNAL_TOPIC“ statt. Weiterhin werden Nachrichten von anderen Modulen, auf den entsprechenden Topics empfangen.

Das ActivMQ_Interface nimmt Nachrichten von ActivMQ entgegen und reicht sie an die Comm_Logic weiter. In der Comm_Logic wird entschieden, ob die Nachricht von Interesse ist und ob sie weitergeleitet wird. Die Comm_Logic kann leicht so angepasst werden, dass andere Nachrichten weitergeleitet werden oder die Nachrichten noch vorverarbeitet werden.

In der Comm_Logic werden die Nachrichten durch den Parser decodiert und die übertragenen Werte in den Value_Buffer geschrieben.

Der Value_Buffer dient als Speicher für die Daten, welche von Sense geschickt werden. Diese Daten werden hier zwischengespeichert und warten darauf in den Current_State übernommen zu werden. Dabei kann es durch aus sein, dass Daten mehrfach überschrieben werden, bevor sie in den Current_State gelangen. Das führt jedoch nicht zu Informationsverlust, da nur aktuelle Zustände und keine Änderungen berücksichtigt werden. Wenn der Timer, welcher mit einer Periode von 100 Millisekunden läuft, an den Value_Buffer ein Signal schickt, wird der Mutex, der die Daten schützt, reserviert und die Daten werden in den Current_State übertragen. Während der Übertragung können sich die Daten nicht ändern. Das sorgt dafür, dass es einen konsistenten Schnappschuss des Zustandes gibt, auf dessen Basis die Berechnungen durchgeführt werden können. Während ein Berechnungszyklus läuft, können sich die Daten nicht verändern.

Der Current_State ist der oben beschriebene konsistente Zustand. Hier werden die aktuellsten Sensordaten gespeichert und der Engine zugänglich gemacht. Die Daten die im Current_State gespeichert sind, werden durch einen Mutex geschützt, welcher verhindert, das Daten von der Engine ausgelesen werden können, während Daten vom Value_Buffer übertragen werden. So können keine Berechnungen aufgrund von inkonsistenten Daten erfolgen.

Die Engine ist dafür verantwortlich die Daten, die von Sense geliefert werden auszuwerten. Dabei ist es wichtig, die Abhängigkeiten zwischen den einzelnen Werten zu berücksichtigen und daraus die richtige Verarbeitungsreihenfolge abzuleiten. Nachdem ein neuer Wert berechnet wurde, wird dieser über das zweite ActivMQ_Interface an Act weitergeleitet. Die Werte werden nicht vor der Berechnung gelöscht. Da die Reihenfolge der Abhängigkeiten eingehalten wird, werden nie veraltete Daten zur Berechnung benutzt. Die Engine sichert sich die Zugriffsrechte auf den Current_State bevor die Berechnung startet. Dadurch wird sichergestellt, dass die Daten nicht zwischen zeitig durch die Daten des Value_Buffers ersetzt werden.

3.2.3 Act

Das Act_Module ist ähnlich wie das Sense_Module aufgebaut. Es gibt ein ActivMQ_Interface und einen Hardware_Controller. Das ActivMQ_Interface nimmt auf dem internen Topic “PA_INTERNAL_TOPIC“ die Nachrichten von Plan entgegen und entnimmt diesen die entsprechenden Aufgaben. Wenn es gefordert ist, werden Daten an das Game oder das Bike_Module weitergeleitet, um dort für die entsprechenden Aktionen zu sorgen, siehe Abbildung 3.5.

Alternativ können die Nachrichten dafür sorgen, dass der Hardware_Controller selbst direkt die Hardware ansteuert. Seine Aufgabe besteht darin eine LED an und auszuschalten sowie

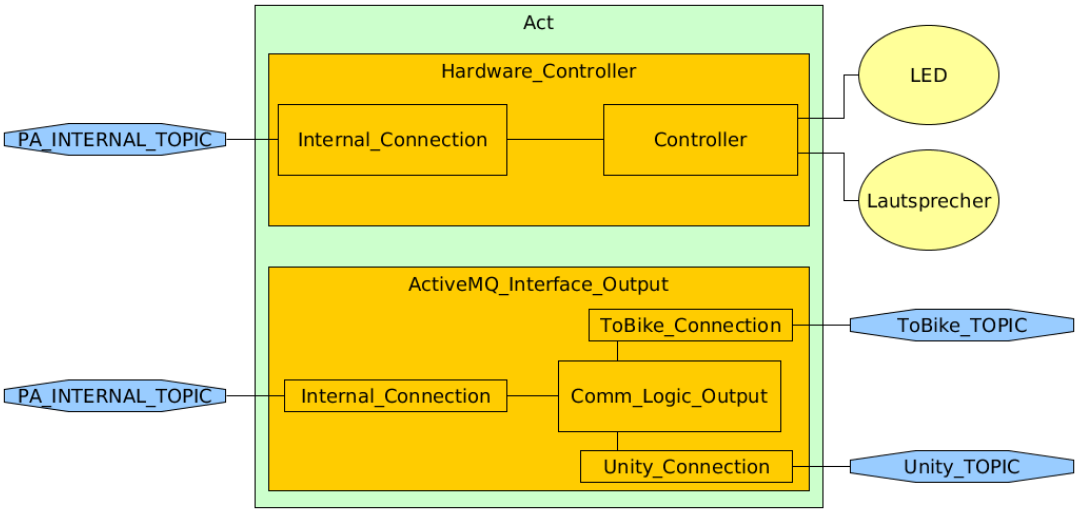

Abbildung 3.5: Aufbau Act_Module

einen Ton abzuspielen. Dazu bedient er sich, wie das Sense_Module, der wiringPi Library und greift auf das Gertboard zu. Der Hardware_Controller enthält einen Controller, welcher als eine Abstraktionsebene zwischen dem Controller und der Hardware funktioniert. Sollte sich die Hardware ändern, so muss nur der Controller entsprechend angepasst werden, die Schnittstellen können dabei gleich bleiben.

3.3 Beschreibung der mathematischen Grundlagen

Die Grundlagen der hier beschriebenen Fahrradphysik stammen aus einer Arbeit von Carsten Bielmeier, welcher sich sehr ausführlich mit den physikalischen Zusammenhängen des Fahrrads auseinandergesetzt hat [Bielmeier (2012)] sowie den Arbeiten von Joachim Schlichting und Wilfried Suhr [Schlichting und Suhr (2007b)][Schlichting und Suhr (2007a)]. Ein wichtiger Aspekt dieser Arbeit ist die Abbildung der physikalischen Wirklichkeit auf das digitale Fahrradmodel. Die nahezu unendliche Vielfalt der physikalischen Phänomene und Wechselwirkungen welche in der Realität auftreten, werden auf wenige, handhabbare Formeln reduziert. Einige dieser Gleichungen sind elementar für die Lösung dieser Aufgabe, andere haben sich in der laufenden Arbeit als sehr hilfreich erwiesen. Die Kernpunkte sind die Gangschaltung, die Geschwindigkeit und der Tretwiderstand. Da diese Faktoren in Wechselwirkung zu einander stehen ist die Reihenfolge entscheidend, in welcher diese berechnet werden. Die Gangschaltung ist die einzige Berechnung, welche unabhängig von anderen ist und wird als erstes beleuchtet.

3.3.1 Gangschaltung

Die Gangschaltung des Fahrrads funktioniert nach dem mechanischen Prinzip der Übersetzung. Dabei verändert eine physikalische Größe zwar ihre Maßzahl, jedoch bleibt die Einheit erhalten. Für die Umsetzung benötigt man im Grunde nur 2 Zahnräder und eine Kette, mit welcher Kraft vom ersten zum zweiten Zahnrad übertragen werden kann. Wenn jetzt beide Zahnräder die gleiche Größe haben und das Erste gedreht wird, dreht sich das Zweite mit derselben Geschwindigkeit. Die Drehung wird im Verhältnis 1:1 übertragen. Größe beschreibt hier die Anzahl der Zähne am Zahnrad, siehe auch Fischer u. a. (2012).

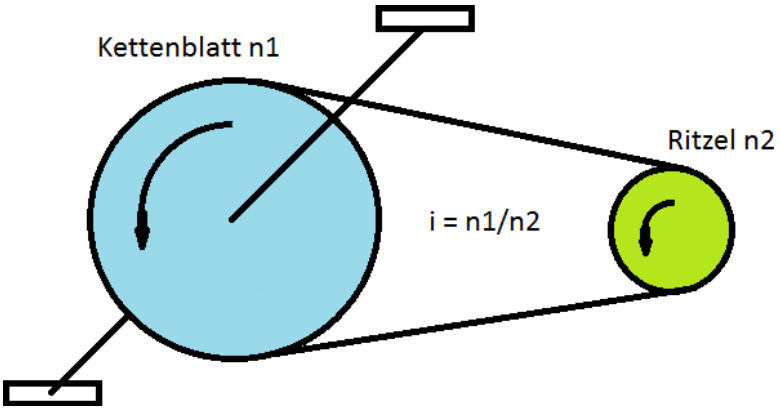

Abbildung 3.6: Übersetzung der Gangschaltung

Wenn jetzt eines der Zahnräder größer als das andere ist so verschiebt sich das Verhältnis. Angenommen das zweite Zahnrad ist kleiner, so dreht es sich mit einer Umdrehung des ersten Zahnrads öfter als ein Mal. Die Umsetzung i berechnet sich durch $i = \frac{n1}{n2}$, wobei n1 und n2 die Größen der beiden Zahnräder sind. Wenn das Kettenblatt eine Größe von 46 Zähnen hat und das Ritzel, welches das Laufrad antreibt, eine Größe von 33 Zähnen hat ist das Verhältnis 1:1,39. Der Gang der an der Gangschaltung eingestellt wird entscheidet über welche Zahnräder die Kette läuft. Dadurch werden die Werte n1 und n2 eingestellt und die Übersetzung reguliert.

3.3.2 Bremse

Bei der Benutzung einer klassischen Bremse wird die Kraft, die der Fahrer zum Drücken des Bremshebels am Lenker aufbringt, über ein Medium an das Bremssystem am Rad übertragen. Auch hier ist eine Kraftübertragung zu beobachten. In der Abbildung 3.7 ist das Schema einer Seilzugbremse zu sehen.

Dieses System übersetzt die Kraft F_{Hand} nach F_{B-N}. Dabei kommen die gleichen Mechanismen zum Tragen wie bei der, oben beschriebenen, Übersetzung der Gangschaltung. Der Unterschied ist lediglich, das dass Übersetzungsverhältnis i nicht durch das Verhältnisse von Zahnräder sondern der Hebellängen bestimmt wird. Die Übersetzung der Kraft i entspricht dem Verhältnis von $\frac{b \cdot d}{c \cdot a}$.

Die Kraft, welche auf die Räder einwirkt senkt die Umlaufgeschwindigkeit und kann als eine negative Beschleunigung betrachtet werden die das Fahrrad im ganzen verlangsamt, siehe Bielmeier (2012).

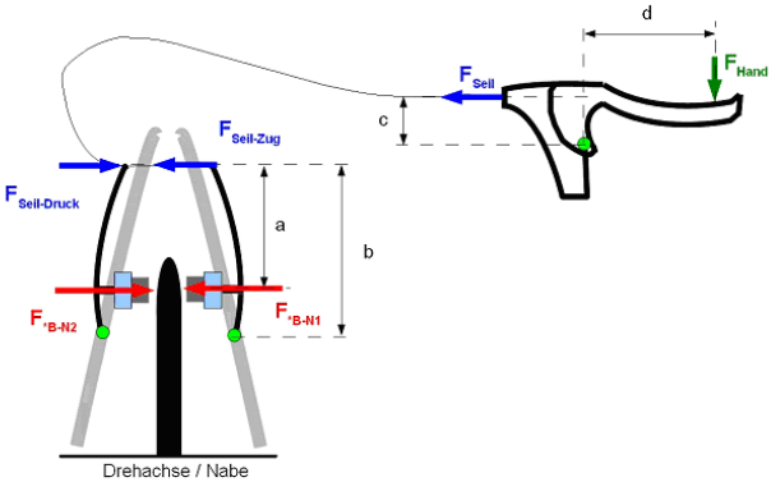

Abbildung 3.7: Darstellung einer V-Bremse [Bielmeier (2012)]

3.3.3 Geschwindigkeit

Die aktuelle Geschwindigkeit des Fahrrades $\ddot{x}$ entspricht dem Integral über die Beschleunigung $\dot{x}$. Um jedoch die Beschleunigung zu kennen muss man bei den Umdrehungen pro Minute (rpm), die der Fahrer an den Pedalen leistet, beginnen. Die rpm sind die Winkelgeschwindigkeit $\dot{\varphi}_P$ an den Pedalen. Die Winkelbeschleunigung an den Pedalen $\ddot{\varphi}_P$ ist die über die Zeit abgeleitete Winkelgeschwindigkeit. Die Winkelbeschleunigung $\ddot{\varphi}_P$ wird jetzt mit der, oben beschriebenen, Übersetzung i multipliziert. Das Ergebnis ist die Winkelbeschleunigung am Antriebsrad $\ddot{\varphi}_A = \ddot{\varphi}_P \cdot i$. Die Beschleunigung des Fahrrades ist das Produkt aus der Winkelgeschwindigkeit und dem Radradius $\ddot{x} = \ddot{\varphi}_A \cdot r_A$. Jetzt ist die Beschleunigung bekannt und es kann das Integral gebildet werden um so die Geschwindigkeit zu bestimmen $\dot{x} = \int \ddot{x}$.

3.3.4 Tretwiderstand

Die Berechnung des Tretwiderstandes beginnt damit, dass gilt: $\sum M_A = F_k \cdot r_R - F_A \cdot r_A = 0$ die Summe der Drehmomente ist Null. M_A sind dabei die Momente auf der Antriebsseite, F_K ist die Kraft mit der die Kette gezogen wird und F_A ist die Kraft mit der das Fahrrad vorwärts bewegt wird. In einem Stillstehenden System wäre die Kraft F_A gleich groß wie die entgegengesetzte Kraft F_W, welche die Summe aller Widerstände ist. Ein Drehmoment ist etwas, das einen Körper um einen bestimmten Punkt dreht, es ist das Produkt aus einer Kraft und der Entfernung zur Drehachse, als dem Radius des bewegten Objekts, an der die Kraft wirkt $M = F \cdot r$. Wenn man in die oben beschriebene Formel nun die Widerstandskraft F_W einsetzt, welche F_A entgegen wirkt, so ist das Ergebnis nicht mehr 0, sondern das Drehmoment, welches sich aus der Winkelbeschleunigung am Rad und dem Trägheitsmoment ergibt.

$$F_K \cdot r_R - F_W \cdot r_A = J_{R2} \cdot \ddot{\varphi}_A \tag{3.1}$$

Die Kraft F_K ist die Kraft, mit welcher die Kette gezogen wird und diese berechnet sich aus der Pedalkraft F_P welche mit dem Verhältnis $\frac{r_P}{r_K}$ multipliziert wird. Es gilt also $F_P \cdot r_P = F_K \cdot r_K = M_P$ und daraus folgt $F_K = F_P \cdot \frac{r_P}{r_K}$. Da die Kraft an den Pedalen berechnet werden soll, muss in die Formel 3.1 F_K ersetzt werden, daraus folgt dann:

$$F_P \cdot \frac{r_P}{r_K} \cdot r_R - F_W \cdot r_A = J_{R2} \cdot \ddot{\varphi}_A \tag{3.2}$$

Die Widerstandskräfte F_W setzt sich aus mehreren Kräften zusammen. Der Fokus liegt auf dem Reibungswiderstand $F_{Verlust}$, dem Luftwiderstand F_{Luft} und der Trägheit $m \cdot \ddot{x}$. Der Reibungswiderstand beschreibt nicht die Reibung auf dem Untergrund, sondern die Reibung der beweglichen Teile des Fahrrads. Also die Reibung der Radachse und der Zahnräder der Gangschaltung. Diese kann pauschal auf etwa 5 Newton festgelegt werden, vergleiche Schlichting und Suhr (2007b). Als Zweites wird der Luftwiderstand berechnet. Dieser ergibt sich in Abhängigkeit zur Geschwindigkeit, wobei „Der Widerstand F_l der verdrängten und verwirbelten Luft quadratisch mit der Geschwindigkeit zunimmt„Schlichting und Suhr (2007b). Daraus ergibt sich folgende Formel: $F_{Luft} = m_k \cdot v_n^2$. Dabei ist m_k ein konstanter Wert, welcher sich aus der Luftwiderstandsfläche $c_w A$ und der Luftdichte p, durch $m_k = c_w A \cdot p$, berechnet. Der Wert für die Luftwiderstandsfläche ist konstant, sieh auch Schlichting und Suhr (2007a). Dabei ist jedoch zu beachten, dass das hier verwendete Model davon ausgeht, das der Wind still ist und nur durch die Geschwindigkeit des Fahrrads Luftwiderstand erzeugt wird. Da das EmotionBike-Game kein Wetter simuliert ist das vollkommen ausreichend. Die Massenträgheit

ist das Produkt der Masse m und Beschleunigung $\ddot{x}$. Die Widerstandskraft setzt sich also wie folgt zusammen: $F_W = F_{Verlust} + F_{Luft} + (m \cdot \ddot{x})$. Mit dem rechten Ausdruck wird nun F_W in der Formel 3.2 ersetzt.

$$F_P \cdot \frac{r_P}{r_K} \cdot r_R - (F_{Verlust} + F_{Luft} + (m \cdot \ddot{x})) \cdot r_A = J_{R2} \cdot \ddot{\varphi}_A \tag{3.3}$$

Wenn die Formel 3.3 nach F_P umgestellt wird, erhalten wir einen Ausdruck, welcher uns verrät wie groß die Kraft ist, welche sich dem Fahrer bei an den Pedalen entgegenstemmt also zur Fortbewegung überwunden werden muss.

$$F_P = ({}_{R2} \cdot \ddot{\varphi}_A + (F_{verlust} + F_{Luft} + (m \cdot \ddot{x})) \cdot r_A) \cdot \frac{r_P}{r_K \cdot r_R} \tag{3.4}$$

Da das Ergometerfahrrad gerne eine Leistung entgegen nimmt, muss die Kraft F_P noch mit der oben berechneten Geschwindigkeit $\dot{x}$ multipliziert werden. Das Ergebnis ist die Leistung P.

4 Implementierung

4.1 Umsetzung der Mathematik

4.1.1 Der Berechnungszyklus in der Engine Klasse

Der Berechnungszyklus ist eine periodisch aufgerufene Methode. Die Sensorwerte, welche das Plan-Modul von dem Sens-Modul erhalten hat, werden eingelesen und in der Engine-Klasse verarbeitet. Anschließend werden die Ergebnisse über den ActivMQ Messagebroker weiter geschickt. Gestartet wird der Berechnungszyklus dabei von einem Timer. Dieser ist auf 100 Millisekunden eingestellt.

Zu Beginn eines Berechnungszyklus werden die aktuellen Daten aus dem Current_State geladen. Das sind die RPM, die Steigung, Gear und Brake_Value, die Zustände der Buttons, ob der Booster ausgelöst wurde, das Reset Signal und die Nachrichten für den Start und das Ende der Messung.

Wie oben beschrieben ist die Abfolge der Berechnungen sehr wichtig. Ergebnisse einer Berechnung bilden die Grundlage für weitere Funktionen. Beispielsweise ist die Beschleunigung abhängig von der der Gangschaltung, weswegen zuerst die Gangschaltung berechnet werden muss. Ansonsten würde die Funktion zur Berechnung der Beschleunigung mit veralteten Daten arbeiten. Wie in der Abbildung 4.1 gezeigt ist die Abfolge:Gangschaltung, Bremse, Beschleunigung, Geschwindigkeit, Widerstand und Leistung. Nachdem ein Berechnungszyklus durchlaufen wurde, wird geprüft, welche Werte sich geändert haben. Diese Werte werden über ActivMQ an das Act_Modul weitergeleitet. Act ist dafür verantwortlich diese den richtigen Systemkomponenten zukommen zu lassen.

4.1.2 Gangschaltungsberechnung

Die Gangschaltung ist als Automat implementiert. Der aktuelle Gang ist dabei der Zustand, in dem sich der Automat befindet, der Wert wird in Gear_State gespeichert. Jeder Gang hat einen oberen und unteren Grenzwert. In dieser Methode wird der Wert betrachtet, auf welchen das Potenziometer eingestellt ist, dieser Wert heißt Gear_Value. Wenn der Wert von Gear_Value kleiner als der untere Grenzwert oder größer als der obere Grenzwert ist, wird

Abbildung 4.1: Berechnungsreihenfolge im Plan_Modul

der Gang um einen Stufe erhöht oder verringert. Das liegt daran wie das Potenziometer an den ADC angeschlossen ist. Der Gang kann sich also pro Berechnungszyklus nur um eine Stelle verschieben. Das Schema wird in der Abbildung 4.2 verdeutlicht. Wenn keiner der beiden Schwellenwerte unter beziehungsweise überschritten wird bleib der Gang unverändert. Der neue Gang überschreibt den alten Gear_State.

Gang	unterer Grenzwert	oberer Grenzwert
1	305	MAX
2	262	305
3	234	261
4	220	242
5	184	219
6	117	183
7	0	116

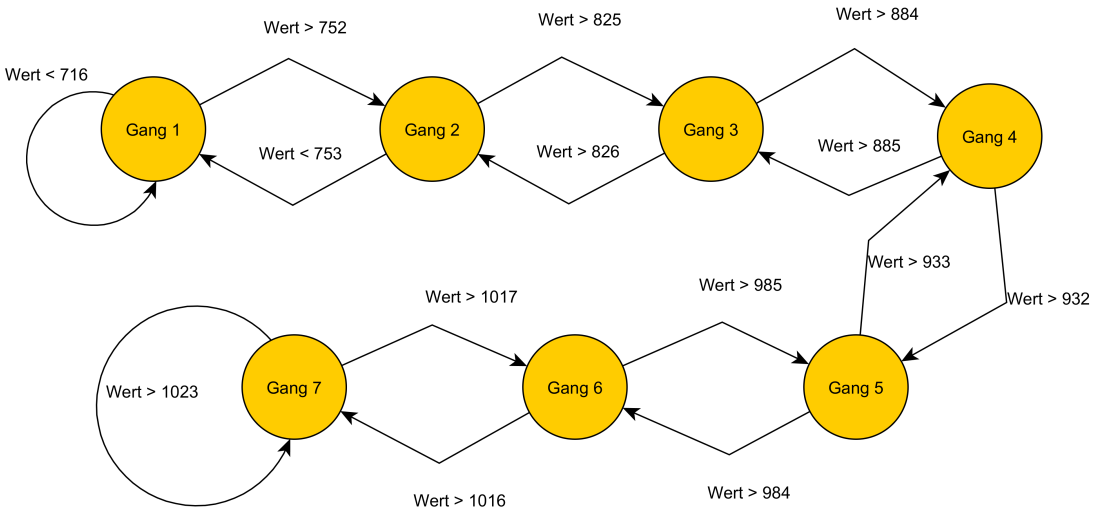

Abbildung 4.2: Darstellung der Umsetzung de Gangschaltung als Automat

Wenn ein neuer Gang bestimmt ist, werden noch die Übersetzungen von Kettenblatt nach Ritzel, in dem Wert i_P_nach_A, vorberechnet. Das sorgt dafür das nur an einer Stelle im Code nach dem Gang gefragt werden muss und jetzt an den entsprechenden Stellen nur noch diese Variable eingefügt werden muss. Da die Größen der Ritzel und des Kettenblatts konstant sind, wird bei Programmstart ein Lookup-Table mit den entsprechenden Werten für alle Kombinationen gefüllt. Es muss also nur noch auf den, dem Gang entsprechenden, Eintrag referenziert werden.

Der Einfluss den die Gangschaltung auf die Geschwindigkeit nimmt, lässt sich in der Abbildung 4.3 nachvollziehen. Die Gangschaltung verändert, nach Sekunde 15, ihren Zustand, sie wird nach und nach hoch geschaltet, zu erkennen am stufenförmigen Verlauf des blauen Graphen. Die Geschwindigkeit vollzieht den gleichen stufenförmigen Verlauf, auch wenn dieser nicht sauber ist, sondern Schwankungen unterliegt. Dies ist dadurch zu erklären, dass

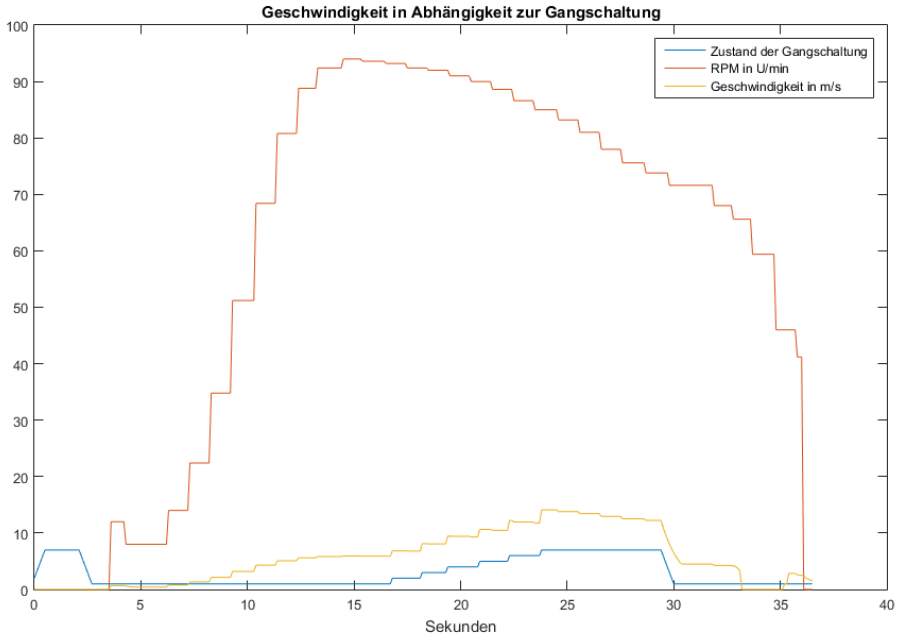

Abbildung 4.3: Die Beeinflussung der Geschwindigkeit durch die Gangschaltung.

die anderen Faktoren, wie die Trittgeschwindigkeit, aus denen sich die Geschwindigkeit berechnet nicht konstant sind. Anzumerken ist, dass das Erhöhen des Gangs stärker wirkt als die Verringerung der RPM, da die resultierende Geschwindigkeit immer noch steigt.

4.1.3 Bremsenberechnung

Die Bremse ist wie die Gangschaltung als ein Automat modelliert. Im Gegensatz zu Gangschaltung hat die Bremse jedoch nur drei Zustände. Der Zustand der Bremse ist in Brake_State gespeichert. Die entscheidende Größe ist die Spannung, die über den variablen Widerstand des Potenziometers abfällt, welcher mit der Bremse verbunden ist. Dieser Wert ist in Brake_Value gespeichert. Die Übergänge sind auch hier über untere und obere Grenzwerte geregelt. Wenn diese unter oder überschritten werden, kommt es zu einer Zustandsänderung. Der neue Zustand der Bremse überschreibt den alten Wert von Brake_State. Dieser Wert wird für spätere Berechnungen vorgehalten. Die Bremse überspringt genau wie die Gangschaltung keine Zustände.

Bremszustand	unterer Schwellenwert	oberer Schwellenwert
0	0	220
1	210	540
2	535	MAX

4.1.4 Beschleunigungsberechnung

Die Beschleunigung bestimmt sich aus mehreren Teilbeschleunigungen. Dies sind die Beschleunigung die der Fahrer erzeugt, Beschleunigung_Pedale, die durch die Steigung entsteht, Beschleunigung_Slope, und die, welche durch die Bremse verursacht wird, Beschleunigung_Brake. Diese drei Teilbeschleunigungen werden zu einer Gesamtbeschleunigung verrechnet.

Beschleunigung durch den Fahrer

Die Beschleunigung der Pedale sind der Anteil, welcher der Fahrer mit seiner persönlichen Energie leistet. Die RPM, also die Umdrehungen pro Minute der Pedale, werden mit der Übersetzung multipliziert um die Umdrehungen pro Minute am Hinterrad zu bestimmen. Aus den RPM am Hinterrad wird jetzt die aktuelle Winkelgeschwindigkeit berechnet $winkel\_speed = rpm\_now \cdot (2 \cdot Pi)/60$. Das Dividieren durch 60 schneidet die Minuten auf Sekunden herunter, winkel_speed ist also in m/s. Jetzt muss die Differenz zwischen der neuen und der alten Winkelgeschwindigkeit bestimmt werden rps_diff. Um die Winkelbeschleunigung am Hinterrad zu bestimmen, wird jetzt das numerische Differenzial gebildet werden $winkelbeschleunigung\_a = rps\_diff/t\_sek$. t_sek ist die Dauer eins Berechnungszyklus in Sekunden, also $0,1s$. Die Beschleunigung ist die Winkelbeschleunigung am Hinterrad multipliziert mit dem Radradius $Beschleunigung\_Pedale = winkelbeschleunigung\_a \cdot RadRadius$. Damit wäre der erste Schritt abgeschlossen.

Beschleunigung durch die Steigung

Für die Berechnung der Beschleunigung durch die Steigung muss erst einmal bestimmt werden in welchem Winkel sich das Fahrrad bewegt. Dazu wird der Slope Wert betrachtet. Dieser ist die Steigung, welche vom EmotionBike-Game geschickt werden, die das virtuelle Fahrrad zu bewältigen hat. Die Steigung beginnt bei 0°, was für eine Fahrt über eine Ebene steht, geht bis 90° bergauf und zwischen 270° bis 360° geht die Fahrt bergab, sieh Abbildung 4.4. Wenn die Fahrt bergab geht wird das Fahrrad zusätzlich beschleunigt. Auf der Ebene und der Fahrt bergauf nicht. Das es keine negative Beschleunigung beim bergauf Fahren gibt, liegt vor allem daran, dass das Fahrrad nicht rückwärts fahren kann.

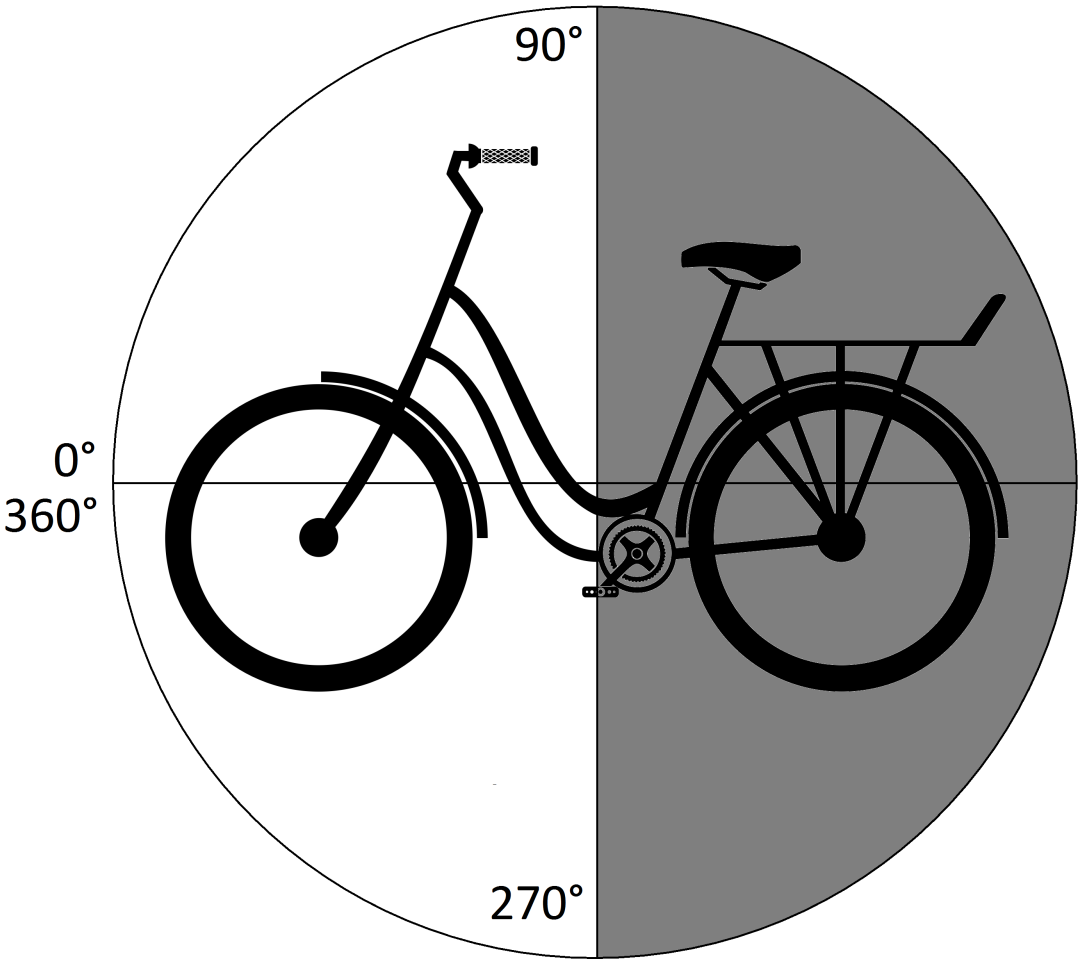

Abbildung 4.4: Darstellung der Winkel, relativ zum Fahrer.

Beim bergab fahren muss zuerst die Steigung umgerechnet werden. Das liegt daran, das der Steigungswinkel, welcher von EmotionBike-Game gesendet wird von Vorne über den Fahrer hinweg gezogen wird, wie in Abbildung 4.4 dargestellt ist. Das Bergabfahren hat also einen Winkel von über 270°. Die wirkliche Steigung ist $slope = 360 - slope$. Die Hangabtriebskraft ist das Produkt aus Sinus der Steigung und der Gewichtskraft $F_h = sin(slope) \cdot G$. Die Hangabtriebskraft wird nun durch die Masse geteilt um die Hangabtriebsbeschleunigung zu bestimmen $beschleunigung\_slope = f_h / MASS$.

Beschleunigung durch die Bremse

Der dritte Teil ist die negative Beschleunigung durch die Bremse. In diesem Modell wird die Bremse in drei Zustände aufgeteilt, welche das Fahrrad entweder gar nicht, ein bisschen oder komplett bremsen. Um zu bestimmen wie stark die Bremse gezogen ist, wird der Brake_State abgefragt und je nach dem Zustand eine negative Beschleunigung von $-1m/s^2$, $-20m/s^2$ oder $-1000m/s^2$ als Bremsbeschleunigung gesetzt. Der Wert von -1 für die ungebremste Fahrt ist dafür da, um das Ausrollen des Fahrrads zu simulieren.

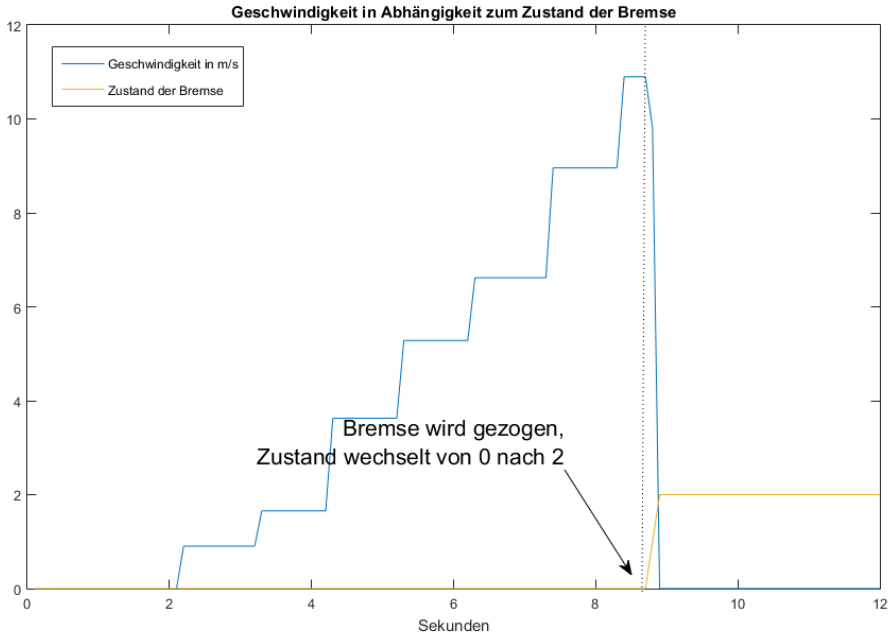

Abbildung 4.5: Die Bremse beeinflusst die Geschwindigkeit.

Auf der Abbildung 4.5 wird an der markierten Stelle die Bremse schlagartig voll durchgezogen. Der Geschwindigkeitsgraph macht daraufhin einen kleinen Knick und fällt anschließen auf 0. Der Knick erklärt sich durch den Zustandsübergang von 0 über 1 nach 2. Im Zustand 0 wird nicht gebremst. Im Zustand 1 ist die Bremse leicht gezogen und bremst das Fahrrad ab. Der letzte Zustand bremst das Fahrrad komplett ab, indem die Geschwindigkeit auf 0 gesetzt wird.

Beschleunigung

Bei der Verrechnung der drei Beschleunigungswerte muss auch wieder auf die Steigung geachtet werden, die das Fahrrad zu bewältigen hat. Wenn die Fahrt bergauf oder über die

Ebene geht, werden einfach die Beschleunigungen durch den Fahrer und der Bremse addiert $beschleunigung\_now = beschleunigung\_pedale + beschleunigung\_brake$. Bei der Fahrt bergab muss zuerst geprüft werden, ob die Beschleunigung des Fahrers stärker ist als die Beschleunigung durch die Abfahrt. Der größere der beiden Werte wird für die Berechnung der Beschleunigung zur Gesamtbeschleunigung addiert.

4.1.5 Geschwindigkeitsberechnung

Für die Geschwindigkeitsberechnung ist zuerst der Brake_State wichtig. Wenn dieser 2 ist, also voll durchgezogen, wird die Geschwindigkeit direkt auf 0 gesetzt. Andernfalls wird die Geschwindigkeit als das numerische Integral der Beschleunigung betrachtet $speed\_now = speed + beschleunigung \cdot t\_sek$. Als Letztes muss nur noch geprüft werden, ob die Geschwindigkeit kleiner als 0 ist. Sollte das der Fall sein, wird sie auf 0 gesetzt, da das Fahrrad nicht rückwärts fährt.

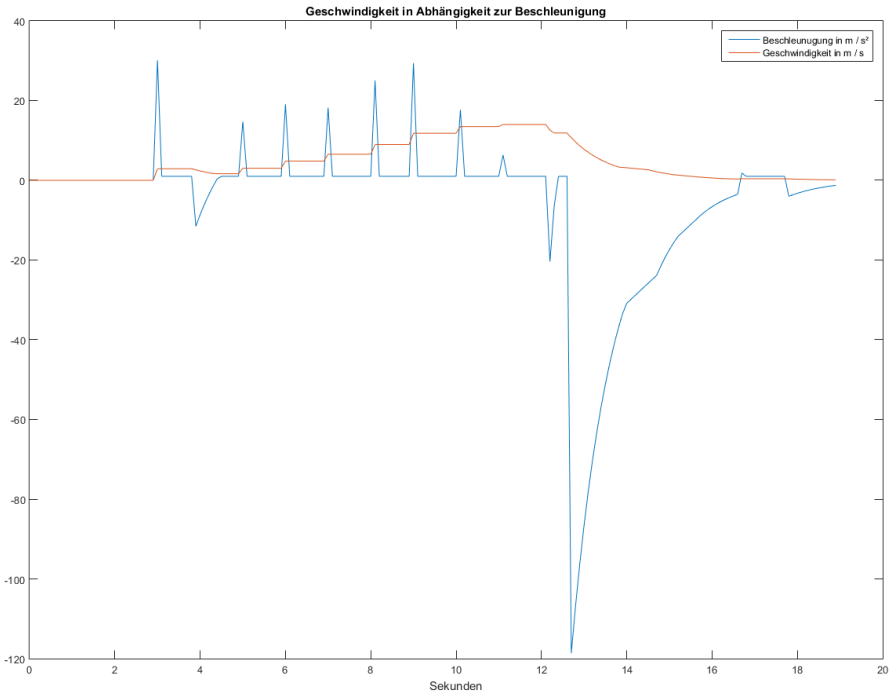

Abbildung 4.6: Die Geschwindigkeit ist direkt abhängig von der Beschleunigung.

In der Abbildung 4.6 kann man den Einfluss der Beschleunigung auf die Geschwindigkeit erkennen. Die starke negative Beschleunigung, nach der zwölften Sekunde, lässt das Fahrrad abbremsen und danach ausrollen. Davor kann man den Zusammenhang zwischen den Beschleunigungspeaks und der Geschwindigkeitsänderung erkennen. Bei einer positiven Beschleunigung wird das Fahrrad konstant schneller. Wenn die Beschleunigung auf 0 ist, verändert sich die Geschwindigkeit nicht weiter.

4.1.6 Widerstandsberechnung

Der Widerstand besteht auch aus drei Komponenten, dem Luftwiderstand, dem Steigungswiderstand und dem Bremswiderstand. Diese werden einzeln berechnet und anschließend zum Gesamtwiderstand verrechnet.

Luftwiderstand

Zuerst wird der Luftwiderstand berechnet. Dieser ist das Produkt der Luftwiderstandskonstante und dem Quadrat der Geschwindigkeit des Fahrers. Die Luftwiderstandskonstante ergibt sich aus der Luftdichte und der Querschnittfläche des Fahrers. Diese Werte werden, wie auch die Masse von Fahrer und Fahrrad, als konstant und für jeden gleich angenommen.

Steigungswiderstand

Um den Steigungswiderstand zu berechnen, wird zunächst geprüft, ob sich der Fahrer parallel zum Boden, bergauf oder bergab bewegt. Dazu wird Slope geprüft. Wenn die Fahrt bergauf geht, muss zusätzlich noch die Gewichtskraft überwunden werden, welche das Fahrrad den Berg hinunter zieht. Die Gewichtskraft ist das Produkt aus dem Sinus der Steigung und dem Produkt aus der Masse des Fahrrads und der Erdbeschleunigung $f_h = sin(slope) \cdot MASS \cdot 9,81$. in den beiden anderen Fällen wird kein zusätzlicher Widerstand angenommen.

Bremswiderstand

Der Bremswiderstand ist als eine einfache Tabelle implementiert. Je nach Brake_state wird entweder kein Widerstand, etwas Widerstand oder ein sehr großer Widerstand verwendet. Dieser Wert wird als Bremswiderstand gespeichert.

Gesamtwiderstand

Da es keine negativen Widerstände gibt, können einfach alle drei Werte addiert werden $Widerstand = Luftwiderstand + Steigungswiderstand + Bremswiderstand$. Es muss

nur noch einmal der Slope betrachtet werden. Wenn das Fahrrad bergab fährt und die Beschleunigung_Pedal kleiner ist als die Beschleunigung_Slope wird der Widerstand auf 0 gesetzt. Dies stellt den Effekt da, dass das Fahrrad schneller rollt als der Fahrer in der Lage ist zu treten und, obwohl getreten wird, das Fahrrad noch im Leerlauf rollt. Dabei spürt der Fahrer keinen Widerstand.

4.2 Leistung

Die Leistung lässt sich aus dem Gesamtwiderstand bestimmen. Der Gesamtwiderstand wirkt auf das Antriebsrad des Fahrers entgegen der Antriebsrichtung. Diese muss nun durch die Gangschaltung auf die Pedale übersetzt werden um zu ermitteln wie groß die Kraft ist, welche an den Pedalen erbracht werden muss $kraft_p = gesamtwiderstand \cdot Übersetzung$. Die Übersetzung muss nun jedoch vom Antriebsrad zu den Pedalen bestimmt werden. Die Leistung p an den Pedalen entspricht dem Produkt der Kraft $kraft_p$ und der aktuellen Geschwindigkeit $p = kraft_p \cdot speed$.

4.3 Weitere Aufgaben der Engine Klasse

4.3.1 Button Verarbeitung

Auch die drei Eingabeknöpfe am Lenker werden von Plan berücksichtigt. Die Buttons werden der Reihe nach aus dem Current_State geladen. Anschließend wird eine Nachricht an Act geschickt, in der die Zustände der einzelnen Buttons übermittelt werden. Es erscheint etwas sinnlos eine Information zwischenzuspeichern und CPU-Zeit darauf zu verwenden sie einfach nur durchzureichen, wenn man sie auch direkt von Sense nach Act schicken könnte. Jedoch kann die Methode in der Engine, welche die Überprüfung der Zustände übernimmt leicht durch eine Methode ausgetauscht werden, welche zusätzliche Informationen aus den Buttons ziehen kann. So zum Beispiel einen Geschwindigkeitsbooster für Testzwecke.

4.3.2 LED Sound Verarbeitung

Die letzte Aufgabe die die Engine hat ist die Verarbeitung der meassureStarted und meassureStoped Nachrichten. Diese Nachrichten werden aus dem CotrollCenter geschickt. Sie geben an das die Messung des Probanden begonnen hat bzw. abgeschlossen ist. Um später das Videomaterial leichter mit den Messdaten synchronisieren zu können, wird durch diese Nachricht eine LED im Sichtfeld der Kamera angesteuert und ein Ton abgespielt. Die Engine

zieht den Zustand der meassureStarted und meassureStoped Variablen aus dem Current_State. Diese geben an, ob Sense ein entsprechende Nachricht empfangen hat oder nicht. Wenn eine Variable gesetzt ist, wird eine entsprechende Nachricht an Act geschickt die LED anzusteuern und den Ton abzuspielen. Die meassuerStoped Nachricht hat dabei eine höhere Priorität als die meassuerStarted Nachricht. Das bedeutet, falls beide Variablen gesetzt sind wird zuerst die meassureStarted Nachricht an Act weitergereicht, was zur Folge hat, dass sie auch zuerst ankommt. Dann wird Act den Ton zweimal, sehr schnell hintereinander spielen und am Schluss die LED löschen. Die Variablen werden nach der Abfrage von der Engine im Current_State gelöscht.

5 Evaluation

5.1 Zusammenfassung

Das EmotionBike-Projekt stellt eine Plattform zur Messung emotionaler und physiologischer Zustände eines Benutzers da. Das Projekt stellt eine Umgebung bereit, in welcher an der Erkennung von Emotionen gearbeitet werden kann. Ziel ist es die Emotionen des Benutzers zuverlässig zu erkennen. Als zu bearbeitendes Objekt ist eine Spielwelt vorhanden, die mehrere Funktionen erfüllt. Zum einen soll sie den Probanden davon ablenken, dass er beobachtet wird und so möglichst unverfälschtes Messen möglich machen, zum anderen provoziert sie gezielt emotionale Reaktionen bei den Benutzern. Das Ergometerfahrrad dient dabei als intuitiv benutzbarer Controller für das Spiel und erweitert gleichzeitig die Palette des Feedbacks. Dadurch, das der Benutzer nicht lernen muss mit dem Controller umzugehen, kann er sich voll und ganz auf das Absolvieren der Strecken des Spiels konzentrieren.

Diese Arbeit soll den Controller um die Funktionalität des Bremsens und der Gangschaltung erweitern, welche den Handlungsspielraum des Benutzers erweitern und gleichzeitig das Bild des Fahrrads als Controller vervollständigen. Dem Fahrer würde erweiterte Kontrolle über die Geschwindigkeit des Avatars ermöglicht. Mit der Gangschaltung wäre die Überwindung langer Strecken und starker Steigungen vereinfacht, indem der Benutzer die Übersetzung an seine Bedürfnisse anpassen kann. Die Bremse soll es ermöglichen, die Geschwindigkeit schnell zu verringern. Das neue Controllerelement muss sich in das gesamte EmotionBike-Projekt einfügen, in dem es die bereits vorhandene Infrastruktur nutzt und sich an die Standards der Kommunikationsprotokolle hält.

5.2 Erfolg der Lösung

Das Erfassen der verschiedenen Sensoren und die Kommunikation mit den anderen Modulen und Systemkomponenten wurde erfolgreich implementiert und das Modul fügt sich in das Gesamtsystem ein. Die Kommunikation innerhalb des Moduls funktioniert zuverlässig und es entstehen kaum störenden Latenzen. Weiterhin ist es gelungen das Modul, auch wenn es auf mehrere Computer aufgeteilt wird, als ein in sich geschlossenes System zu implementieren.

Es ist gelungen, die Berechnungen, auf eine Art abzubilden, dass für den Benutzer sowohl ein flüssiges Fahrgefühl entsteht, sich das Fahrrad aber auch an die physikalische Realität annähert. Das ist vor allem für die Immersion hilfreich, da der Fahrer, vor allem aus seinen Erfahrungen mit Fahrrädern, abschätzen kann, wie es sich verhält. Das Physikmodell ist aber auch dafür verantwortlich, dass sich das Fahrrad, in seiner Umgebung, immer gleich verhält. Ein Benutzer der noch nie Erfahrungen mit einem Fahrrad gemacht hat, kann das Verhalten des virtuellen Fahrrad erlernen und es wird keinen, ungewollten, Bruch im Verhalten des Fahrrads geben.

Da eine echte Bremse und eine echte Gangschaltung als Interface für den Benutzer verwendet wurden, ist es recht leicht für die Probanden zu erkennen, um was für Geräte es sich handelt und wie diese zu bedienen sind. Das Modul verhält sich entsprechend der Erwartungen der Benutzer und fügt sich in den Kontext des EmotionBike ein. Es wurde nicht als unpassend oder störend beschrieben.

Zusammenfassend kann gesagt werden, das die zu Beginn des Projektes erstellten Anforderungen, erfolgreich umgesetzt wurden.

5.3 Aufgetretene Probleme

Es gibt ein Problem mit der verzögerten Reaktion der Bremse. Bei den Tests ist aufgefallen, dass die Bremse des Ergometers erst verspätet auf Änderungen des Tretwiderstandes reagiert. Das führt zu dem Effekt, das Ereignisse, die den Tretwiderstand erhöhen, bereits zurückliegen aber der Tretwiderstand, für den Fahrer, erst spürbar wird. Gerade bei kurzen aber starken Beschleunigungen wird die berechnete, zu erbringende Leistung, extrem hoch. Auch wenn diese Leistung nur sehr kurz erbracht werden muss. Durch Verzögerungen im Gesamtsystem und am Ergometer kann es dazu kommen, dass die hohe Leistung länger abgefordert wird als eigentlich notwendig. Es wäre zu klären, ob es sich dabei um ein Problem mit der Bremse am Ergometer oder ein Fehler in der Schnittstelle zum Ergometer handelt.

Weiterhin gibt es eine Differenz zwischen der berechneten und der wahrgenommenen Wirklichkeit des Spiels. Das Fahrrad scheint sich in der Spielwelt deutlich langsamer zu bewegen. Die eingeblendete Geschwindigkeit erscheint sinnvoll und deckt sich auch mit den Erfahrungswerten des eigenen Fahrrads. Dennoch wird die Darstellung der Bewegung im EmotionBike-Game als wesentlich langsamer empfunden. Um dies auszugleichen, wird die Geschwindigkeit noch mit einem Faktor verrechnet, bevor sie an das Spiel geschickt wird.

Es kam im Laufe der Arbeit, nach der Entdeckung dieses Phänomens, der Verdacht auf, dass es am Leveldesign liegen kann. Es scheint als würde es durch die Dimensionen der Objekte im

Verhältnis zueinander und zum Fahrrad und damit auch zum Spieler, so verzogen sein das die Bewegung als langsamer wahrgenommen wird, als sie tatsächlich ist.

Die andere Möglichkeit wäre ein Fehler in den Parametern der Berechnung, wobei sich diese an gemessenen Werten eines echten Fahrrads orientieren. Ein zusätzlicher Faktor könnte ein Mangel an Feedback sein. Dem Fahrer weht kein Wind entgegen, an dem er erkennen könnte, wie schnell er sich bewegt und es fehlt generell das Gefühl von Beschleunigung beim Fahren.

Die Modifizierung der berechneten Geschwindigkeit durch diesen Offset ist eine Verfälschung der berechneten Realität. Das ist jedoch zu vernachlässigen, solange dadurch das Spielerlebnis angenehmer gestaltet ist. Es ist also zu sagen, dass ein gutes Gefühl beim Erleben der Simulation höher zu bewerten ist als die physikalische Korrektheit, solange das Verhalten der Simulation noch als korrekt empfunden wird.

5.4 Weiterführende Ideen

Die, im Rahmen dieser Arbeit entwickelten, Erweiterungen des EmotionBike-Interfaces sind noch lange nicht auf ihrem Maximum angekommen. Es wäre beispielsweise möglich eine Bildschirmrepräsentation zu schaffen, die den Zustand der Bremse und der Gangschaltung widerspiegelt. Das Physikmodell könnte noch so erweitert werden, dass der Untergrund über den sich das Fahrrad bewegt, berücksichtigt wird.

Durch die zusätzlichen Instrumente wird der Bereich erweitert, auf den der Benutzer achten muss. Um den Gang zu prüfen, muss der Blick vom Bildschirm abgewendet werden und die Aufmerksamkeit wandert wieder in den "Realraum". Das kann dazu führen, dass der Benutzer aus seiner Immersion, im Sinne der Vertiefung in die Spielwelt, gerissen und sich seiner tatsächlichen Umwelt bewusst wird.

Eine nützliche Erweiterung ist ein physikalisches Feedback für die Bremse. Es fühlt sich mehr als nur merkwürdig an, den Bremshebel ohne jeden Widerstand komplett bis zum Griff durch Ziehen zu können. Der Benutzer hat kein Gefühl dafür, wie stark er die Bremse gezogen hat. Wenn eine entsprechende Möglichkeit gefunden ist, könnte man auch die Kraft die nötig ist den Bremshebel zu ziehen entsprechend auf das Physikmodell abbilden und die Beschleunigung sowie die Geschwindigkeit entsprechend anpassen. Das würde dem Benutzer die Möglichkeit geben fein aufgelöst auf die gegebenen Umstände im Spiel zu reagieren. Da eine Bremse, die sich widerstandslos bewegen lässt, eher etwas Untypisches ist oder für eine Fehlfunktion spricht, sollte dies noch angepasst werden, um nicht mit dem Erwartungswert des Benutzers zu brechen und ein natürlicheres Abbild einer Bremse zu gewährleisten.

Weiterhin wäre ein Level mit korrekten Proportionen eine Möglichkeit, um zu prüfen, ob die erlebte Geschwindigkeit etwas mit dem Leveldesign zu tun hat oder ob es noch andere Effekte gibt, welche sich auf das Geschwindigkeitsgefühl auswirken. Die neuen Funktionen, die das erweiterte Interface bietet, können auch in neuen Herausforderungen für das Leveldesign umgesetzt werden. Denkbar wäre hier eine stark abschüssige Strecke mit engen Kurven. Mit der Bremse muss verhindert werden, von der Strecke zu fallen, während man gleichzeitig versuchen muss die Strecke möglichst schnell zu bewältigen.

Weitere Ergänzungen wären Hardwarekomponenten, welche die Spielwelt für den Fahrer spürbar machen. Vorstellbar wären Vibrationsmotoren, die die Beschaffenheit des Untergrundes an den Spieler übertragen. Für ein besseres Geschwindigkeitsgefühl könnte man sich ein System vorstellen, welches mithilfe von Ventilatoren den Fahrwind simuliert.

Literaturverzeichnis

[Bernin 2011] Bernin, Arne / Hochschule für Angewandte Wissenschaften Hamburg. 2011. – Forschungsbericht

[Bielmeier 2012] Bielmeier, Carsten / Julius-Maximilians-Universität Würzburg. 2012. – Forschungsbericht

[Erich Gamma 1996] Erich Gamma, R.H.J.V.R.J.: *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 1996 (Professionelle Softwareentwicklung). – ISBN 3-8273-1862-9

[Fischer u. a. 2012] Fischer, R. ; Küçükay, F. ; Jürgens, G.: *Das Getriebebuch*. Springer Vienna, 2012 (Der Fahrzeugantrieb). – ISBN 9783709108772

[Friedewald] Friedewald, Michael: – Forschungsbericht

[Hellige 1998] Hellige, Hans D. / Universität Bremen. 1998. – Forschungsbericht

[Hornschuh 2015] Hornschuh, Jonas: Weiterentwicklung eines Fahrradergometers als intuitive Steuerung für virtuelle Welten / Hochschule für Angewandte Wissenschaften Hamburg. 2015. – Forschungsbericht

[Inc 2013] Inc, Phidgets: *Phidgets*. 2013. – URL http://www.phidgets.com/docs/Potentiometer_Primer. – letzter Zugriff: 09.06.2015

[Monk 2013] Monk, S.: *Raspberry Pi programmieren*. Haar bei München : Franzis Verlag GmbH, 2013 (Franzis Professional Series). – ISBN 9783645602617

[Murphy 2000] Murphy, R.: *Introduction to AI Robotics*. MIT Press, 2000. – ISBN 0-262-13383-0

[Rittmann 2008] Rittmann, T.: *MMORPGs als Virtuelle Welten – Immersion und Repräsentation*. Verlag Werner Hülsbusch, 2008. – ISBN 978-3-940317-20-9

[Schacht 2010] Schacht, Marie / Technische Universität Dresden. 2010. – Forschungsbericht

[Schlichting und Suhr 2007a] Schlichting, Hans-Joachim ; Suhr, Wilfried: Fahrradfahren Mit Pedalkraft gegen Berg und Wind. In: *Physik in unserer Zeit*. Weinheim : WILEY-VCH Verlag GmbH & Co. KGaA, 2007

[Schlichting und Suhr 2007b] Schlichting, Hans-Joachim ; Suhr, Wilfried: Physik des Fahrradfahrens Moderne Zentauren. In: *Physik in unserer Zeit*. Weinheim : WILEY-VCH Verlag GmbH & Co. KGaA, 2007

[The Apache Software Foundation 2004] The Apache Software Foundation: *ActiveMQ*. 2004. – URL http://activemq.apache.org/. – letzter Zugriff: 09.06.2015

[The Jason Groupe] The Jason Groupe: *Einführung in JSON*. – URL http://www.json.org/json-de.html. – letzter Zugriff: 11.05.2015

[Zagaria 2015] Zagaria, Sebastian / Hochschule für Angewandte Wissenschaften Hamburg. 2015. – Forschungsbericht

Abbildungsverzeichnis

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 26. November 2015 Erik Matthiessen