Bachelorarbeit

Implementierung einer präzisen Geschwindigkeitsregelung in einem biometrischen Fischroboter

Victor Brekenfeld

Freie Universität Berlin
Fachbereich Mathematik und Informatik
Institut für Informatik
AG collective intelligence and biorobotics
Biorobotics Lab

Betreuer

Prof. Dr. Tim Landgraf

Selbstständigkeitserklärung

Name:	Victor
Vorname:	Brekenfeld
geb.am:	08.05.1994
Matr.Nr.:	4681675

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Berlin, den

(Victor Brekenfeld)

Inhaltsverzeichnis

Abstract

Das RoboFish Projekt dient der Erforschung des Schwarmverhaltens von Fischen mit Hilfe eines Roboterfisches. Im Zuge dieser Arbeit soll eine entscheidende Verbesserung in Bezug auf die Genauigkeit der Steuerung des Roboterfisches erreicht werden. Hierdurch sollen präzisere Interaktionen mit echten Fischen möglich werden um beispielsweise Angstreaktionen der Tiere zu vermeiden. Es wird gezeigt, wie durch Einsatz von Drehzahlkontroll-Motoren eine präzise Steuerung einzelner Roboterfische erreicht werden kann, sowie welche Herausforderungen im konkreten Kontext des RoboFish-Systems bei der Implementierung zu überwinden waren. Die Ergebnisse zeigen, dass die absolute Geschwindigkeit eines Roboters nun mit einer sehr kleinen Fehlerrate im Vorfeld berechnet und der Roboter so akkurat gesteuert werden kann.

Einleitung

In der Verhaltensforschung in der Biologie ist in den letzten Jahren durch biometrische Roboter ein mächtiges neues Werkzeug entstanden um insbesondere Schwarmverhalten von Tieren zu erforschen. Durch die Möglichkeit ein Individuum in Form eines Roboters in einem Schwarm zu kontrollieren sind vielfältige neue Experimente möglich in denen einzelne Parameter nach belieben angepasst werden können um das Reaktion des Schwarms zu analysieren und dessen Verhalten dadurch zu verstehen.

Das Schwarmverhalten von Fischen beispielsweise ist sehr komplex für die primitiven Aktionen der für sich genommenen Individuen. Das RoboFish System [1] hat gezeigt, dass die Akzeptanz eines Roboterfisches ein ausreichend hohes Maß erreichen kann um solche Verhaltensforschung an Fischschwärmen zu betreiben [2]. Durch einen fahrenden Roboter wird hierbei eine Nachbildung eines Fisches an einen Magneten durch ein flaches Becken bewegt um echte Fische zu täuschen und so verschiedene Interaktionen innerhalb des Schwarms zu simulieren.

Die Geschwindigkeit des Roboters wird durch verschiedene modellierte Verhaltensweisen geregelt, jedoch ist die absolute Geschwindigkeit bisher von vielen nicht direkt beeinflussbaren Faktoren abhängig. So treten bei den verschiedenen Robotern unterschiedliche Reibungen beim Antrieb auf, was trotz gleicher Steuerung zu unterschiedlichen Geschwindigkeiten führt. Desweiteren findet die Regelung über ein kodiertes Tastverhältnis statt, aus dem die angelegte Spannung des Motors abgeleitet wird, anstatt mit metrischen Einheiten. Die absolute Geschwindigkeit eines Roboters ist praktisch nicht vorher zu sagen und variiert stark je nach eingesetzter Hardware.

Ziel dieser Arbeit ist es die Präzision der Steuerung des Roboters entscheidend zu erhöhen, sodass genaue Vorhersagen über die absolute Geschwindigkeit des Roboters und damit über seine zukünftige Position möglich sind, um feinere Interaktion mit den Tieren zu ermöglichen. So können beispielsweise besser Angstreaktionen der Fische vermieden werden.

Zur Lösung wurde das System dahingehend umgebaut, dass die Motoren der Roboter gegen Drehzahlkontroll-Motoren getauscht wurden um konstant gegebene metrische Geschwindigkeiten halten zu können. Ziel ist, dass alle Roboter gleiches Verhalten für gegebene Parameter zeigen.

State of the Art

Das Robofish System entwickelt an der Freien Universität Berlin von den Mitgliedern der Biorobotics Lab und Prof. Dr. Jens Krause von der Humboldt-Universität zu Berlin ist nach bestem Wissen relativ einzigartig in seiner konkreten Implementierung.

Eine Fisch-Replica befestigt über einen transparenten Stab an einem Magneten (siehe Abbildung 2.1) wird durch einen Roboter unter dem flachen Becken durch dieses bewegt. Die Replica ist komplett unmotorisiert, wodurch eine lebensechte Größe einfach zu realisieren ist. Der Roboter fährt auf einer Plexiglas-Platte unter dem Becken, Kameras von oben und unten tracken die Fische beziehungsweise den Roboter und identifizieren so die Replica um sie von den lebenden Fischen zu unterscheiden. Gesteuert wird der Roboter via Wifi von einer selbst entwickelten Software die live die Position des Roboters und er Fische auswertet um unterschiedliche Verhaltensweisen nachbilden zu können.

Abbildung 2.1: Replica im Wasser

Viele vergleichbare Projekte existieren, ein sehr ungewöhnliches Projekt stellt hierbei die Erforschung des Verhaltens von Nilhechten dar, welche zur Elektrosensorik fähig sind, wodurch die Arbeit von *Donati et al.* [3] die einzige ist in der durch Elektrokommunikation ein an einer Aufhängung befestigter Roboterfisch erfolgreich weitere Fische rekrutiert und in offene Gebiete führt.

Auch in weniger speziellen Fällen funktionieren aufgehängte Replica, so zum Beispiel bei einem Goldbrassen-Roboter an einer festen Position in einem Wassertunnel um die Attraktivität auf Grund von Farbe und Frequenz des Schwanzflossenschlags zu erforschen. [4] Aufhängungen können auch genutzt werden um wenige konkrete Bewegungsmuster abzubilden, so beispielsweise um das charakteristische Werben eines männlichen Rotschwanzkarpflings abzubilden. [5]

Deutlich flexibler und daher häufiger anzutreffen sind jedoch Roboter mit elektronischer Schwanzflosse, welche sich selbst antreiben können, sodass sich zum Beispiel die Reaktion ganzer Schwärme von Zebrafischen auf verschiedene Faktoren testen lassen. [6] Alternativ können größere Raubfische nachgebildet werden, welche wieder andere Reaktion wie Stress und Angst in den Zebrafischen auslösen können, alles in einer sicheren und kontrollierten Umgebung. [7]

Nicht selten führt die benötigte Technik hier zu Robotern deutlich über lebensechter Größe, welche je nach Art vernachlässigt werden kann oder auch trotzdem interessante Ergebnisse erzieht. So werden Goldbrassen auch von Fischen fast doppelte Größe in ihrer Risikobereitschaft beeinflusst [8] und bei Koboldkärpflingen ist das Größenverhältnis im Gegensatz zur absoluten Größe wohl ebenfalls sehr ausschlaggebend für die Akzeptanz des Roboters. [9] Auch Zebrafischen scheinen Farbe und relatives Größenverhältnis wichtiger, sodass große autonome Roboter hohe Akzeptanz erreichen können. [10]

Auch wenn autonom arbeitenden Roboter eine Vielzahl von Vorteilen besitzen, ist der Größenunterschied jedoch in unserem Fall nicht hinnehmbar.
Unsere Lösung lebensnahe Replica's einzusetzen ist demnach sehr auf diese Vorgabe spezialisiert und muss dadurch anderen technischen Anforderungen gerecht werden, die autonome Roboter aufgrund ihrer Größe nicht erfüllen können.

Zwei Systeme sind relativ ähnlich (Replica angebunden via Magnet an fahrenden Roboter) jedoch trotzdem nicht vergleichbar.
Zum einen ist der motorisierte Roboter (ebenfalls in Interaktion mit Zebrafischen) der Gruppe um *Bonnet et al.* [11] auch optional über eine Plattform unter dem Tank zu steuern, ähnlich dem RoboFish-System, doch waren hier leider keine Implementierungsdetails finden, welche sich vergleichen ließen.
Ebenfalls ähnlich ist die Arbeit von *Swain et al.* [12], doch ist der genutzte Roboter ein proprietäres System, dessen Implementierungsdetails sich deshalb ebenfalls nicht vergleichen ließen.

Die Lösung des RoboFish-Systems ist dadurch technisch nicht direkt vergleichbar mit den genannten bisherigen Arbeiten auf diesem Gebiet.

Implementierung

1. Einbau der Motoren

Zur Steuerung des Roboters wird ein Arduino Duo Board verwendetet. Die alten Motoren vom Typ 2619S006 SR 33:1 (Abbildung 3.1.2) sind über ein zusätzliches MotorShield Modul angeschlossen. Die Spannung für die Motoren wird direkt von der Batterie an das MotorShield gegeben, welches entsprechend Spannung an die DC-Motoren weiterreicht (Abbildung 3.1.1). Der Arduino kommuniziert mittels eines WifiShield Moduls mit der Steuerungssoftware.

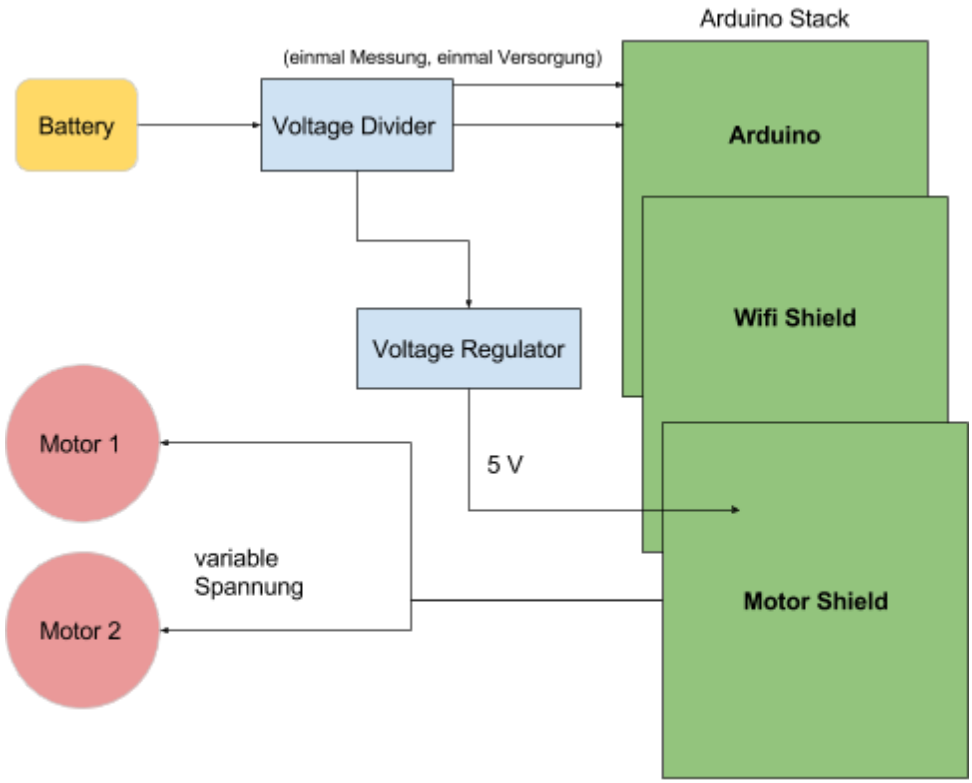

Abbildung 3.1.1: Überblick der Komponenten und deren Versorgung des alten Roboters

Die Anforderungen an das neue System beschränkten sich im Großteil darauf, dass der Roboter mittels einer metrischen Geschwindigkeitsvorgabe gesteuert werden können sollte im Gegensatz zu dem codierten Tastverhältnis des Signals vom Arduino, wodurch die absolute Geschwindigkeit vielen unkontrollierbaren Faktoren der Hardware unterliegt. Des Weiteren sollte diese Steuerung möglichst präzise den gegebenen Wert umsetzen.

Ein erster Versuch durch mehrere Messungen die alten Werte des Tastverhältnisses einer immer gleichen metrischen absoluten Geschwindigkeit zuzuordnen stellten sich leider als zu ungenau heraus. Trotz gleichem Wert schien die konkrete Geschwindigkeit des Roboters weiteren Faktoren zu unterliegen und zudem musste aufgrund der Unterschiede zwischen den Robotern die Messungen für jeden Roboter wiederholt werden, sodass dieser Ansatz schnell verworfen wurde.

Die Lösung des Problems in Hardware mittels eines Speedcontrollers lag nah um nicht jeden Roboter einzeln kalibrieren zu müssen. Auf Grund von relativ wenig Platz im Gehäuse fiel die Entscheidung auf Motoren mit eingebauten Speedcontrollern.
Die Faulhaber 2622S006B SC 33:1 (Abbildung 3.1.2) besitzen einen nahezu identischen Formfaktor wie die alten Motoren sowie den gewünschten Speedcontroller. Wie in Abbildung 3.1.3 zu erkennen ist der neue Motor nur ein wenig länger und besitzt statt zwei Kontakten ein Flachbandkabel.

Abbildung 3.1.2: Einer der alten Motoren des Roboters

Abbildung 3.1.3: Einer der neuen Motoren mit SpeedController

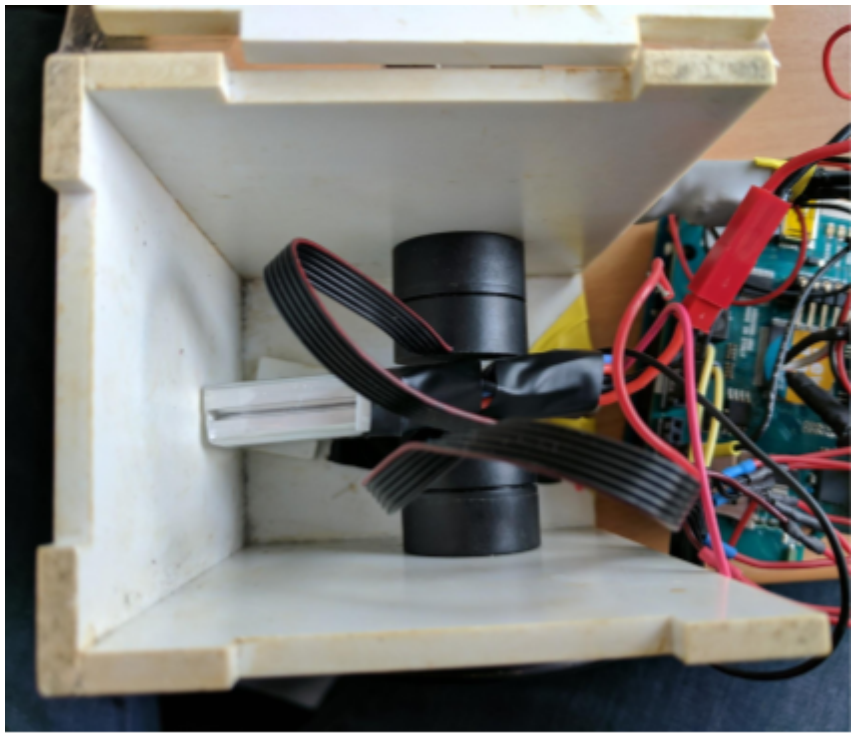

Abbildung 3.1.4: Neue Motoren im Gehäuse

Die Steuerung der neuen Motoren ist wesentlich komplexer. Wie dem Datenblatt [Anhang 5] zu entnehmen, besitzt ein Motor sechs Pins über die die Stromzufuhr und Steuerung erfolgt. Pin 1-3 dienen ausschließlich der Stromzufuhr der verschiedenen Motorkomponenten und werden daher direkt mit dem Voltage Regulator (siehe Abbildung 3.1.5) verbunden, welcher 5V zur Verfügung stellt.

Aus der angelegten Spannung resultiert eine Maximalgeschwindigkeit von 151 rpm, wie sie aus den Werten des Datenblattes errechnet werden kann. Die genaue Berechnung ist beschrieben in [Anhang 1], sowie der folgenden Evaluation.

Gemessene Höchstgeschwindigkeit unserer Fische betrug im Schnitt 20 cm/s, bei 6 cm Reifendurchmesser ist unser Roboter theoretisch in der Lage ein Maximum von 35 cm/s zu erreichen. Die zu erreichende Geschwindigkeit ist demnach mehr als ausreichend.

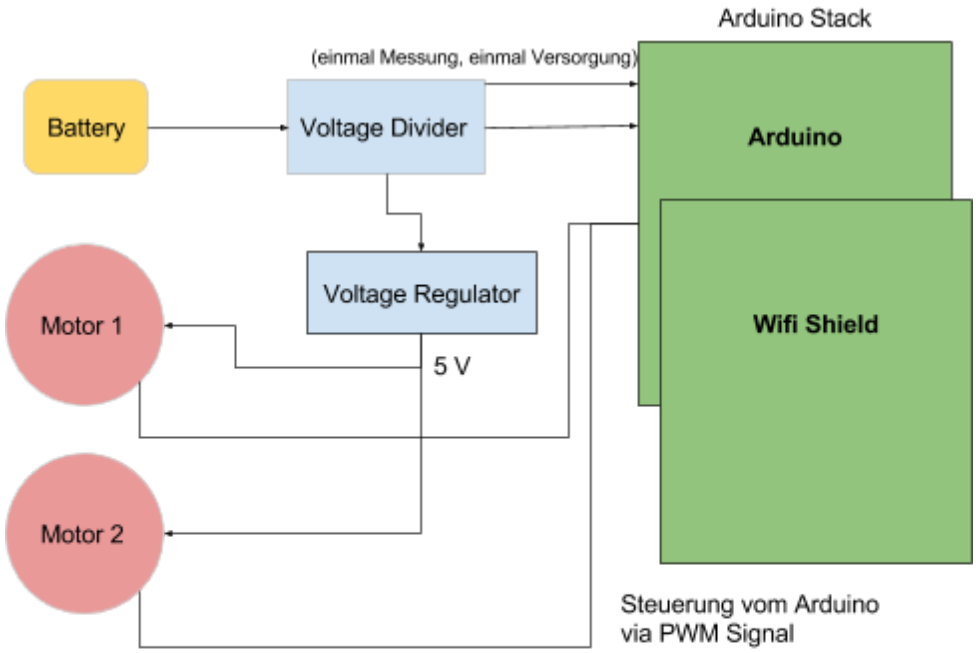

Abbildung 3.1.5: Komponenten und deren Verbindungen im neuen Roboter.

Vergleiche mit Abbildung 3.1.1. MotorShield ist weggefallen, Motoren werden direkt über den Voltage Regulator versorgt und könnte direkt vom Arduino gesteuert werden. Wesentlich verringerte Komplexität.

Die eigentliche Geschwindigkeitssteuerung erfolgt über Pin 4, welcher im Rahmen von 0V bis zur Spannung von Pin 1 (also 5V) betrieben werden kann. Wie dem Handbuch [Anhang 6] zu entnehmen ist, kann die Steuerung auch über ein PWM Signal geregelt werden. In diesem Fall bestimmt der Duty-Cycle die Geschwindigkeit des Motors. So kann der Motor direkt mit dem Arduino verbunden werden, welcher ein PWM Signal mit 5V Spannung liefern kann.

Pin 5 regelt die Drehrichtung und kann über ein digitales Signal (0 < 0,5V, 1 > 3V) regelt werden, welches wir ebenfalls mit dem Arduino bereitstellen können.

Pin 6 dient zur Frequenzmessung und ist in unserem Aufbau überflüssig. Er wird nicht verwendet und abisoliert.

Somit steht der Aufbau unseres neuen Roboters. (Abbildung 3.1.5 & 3.1.6)

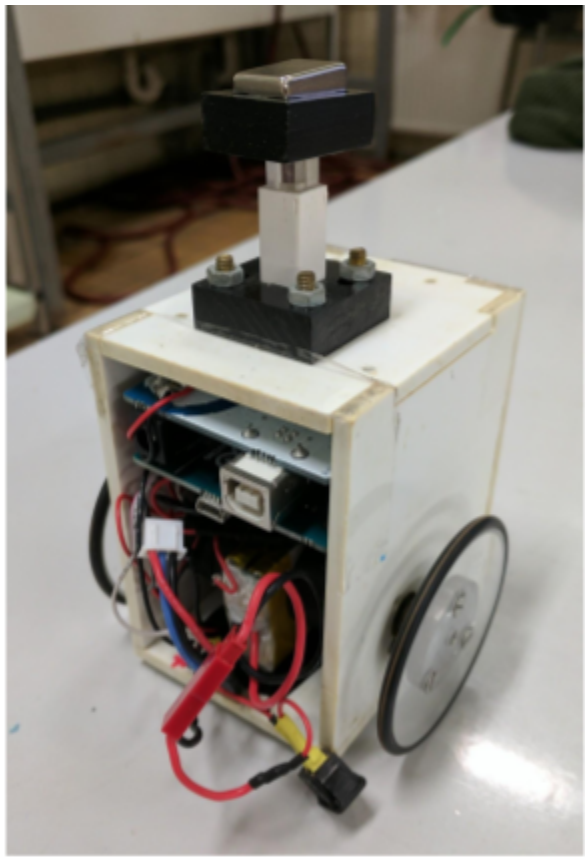

Abbildung 3.1.5: Der fertig zusammengesetzte Roboter mit den neuen Motoren

2. Firmwareanpassungen

Größte Schwierigkeit beim Anpassen der Software war, dass es nicht gelang trotz vorhandenem Quellcode die alte Firmware neu zu kompilieren. Die verwendete Compiler/IDE-Versionen waren nicht dokumentiert worden und der Treiber des verwendeten WifiShield ist durch den Hersteller seit Release nicht gewartet worden. Mittels einiger Anpassungen der Community gelangt zwischenzeitlich zwar das Kompilieren mit neueren Versionen, doch kam es mit der neuen Firmware wiederholt zu Verbindungsausfällen oder gar nicht erst eine Wifi-Verbindung zustande. Gelöst wurde das Problem durch den Tausch des WifiShields durch ein anderes Modell [Anhang 7] und entsprechende Anpassungen an den neuen Treiber.

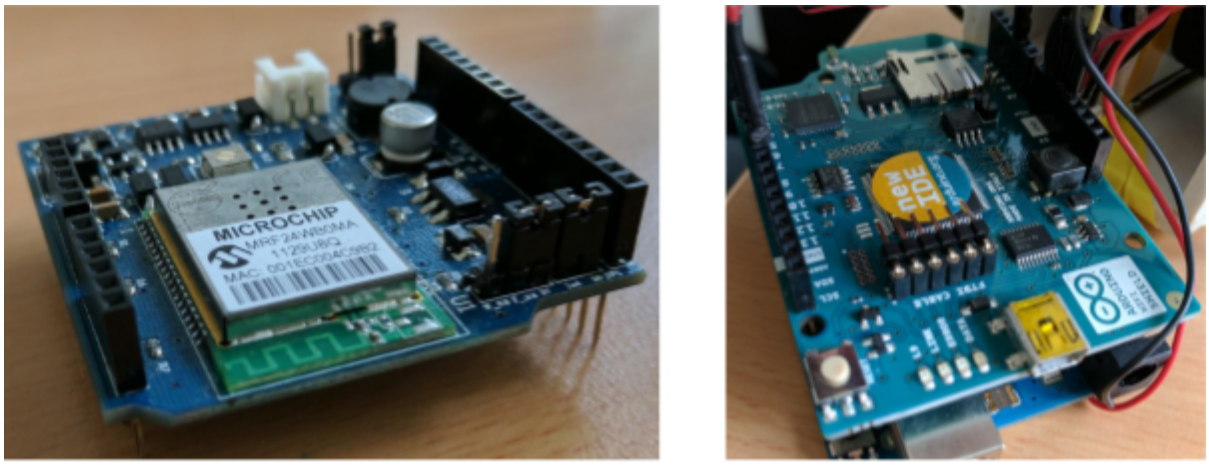

Abbildung 3.2.1: Links das alte Wifi-Shield, rechts das neue im Einsatz

Das Netzwerk-Protokoll wurde dahingehend verändert, dass der Roboter keinerlei Berechnungen zu Geschwindigkeit mehr selber durchführt. Die alten Berechnungen sind ohnehin inkompatibel mit den neuen Motoren und so sind in Zukunft hoffentlich weniger Firmwareanpassungen notwendig.

Außerdem wird bei der Initialisierung der Verbindung nun eine Protokollversion mitsendet. Um zur alten Firmware kompatibel zu bleiben wird nun spezieller Code zur Erkennung in der Steuerungssoftware benötigt. Anpassungen der Firmware auf den alten Robotern war leider wegen genannter Schwierigkeiten beim Rekompilieren nicht möglich, weshalb dieser Weg der Implementierung gewählt wurde.

Die Pakete sind als Union sowie mit einem char als Typ kodiert, sodass das Hinzufügen eines neuen Pakettyps die Kompatibilität nicht beeinträchtigt (siehe Abbildung 3.2.2). Die Steuerungssoftware kann so je nach Antwort zwischen der alten und neuen Firmware unterscheiden.

```
// Packet types
...
const char DiscoveryReply           = 'd'; //deprecated, only used by old protocol
const char DiscoveryVersionReply    = 'e'; //new protocol reply
...

struct Packet
{
        char type;
        union {
                ...
                uint8_t robotID; // DiscoveryReply
                struct {
                        uint8_t robotID;
                        uint8_t version;
                } discovery;     // DiscoveryVersionReply
                ...
        } data;
}
```

Abbildung 3.2.2: Code zur Erweiterung des Netzwerkprotokolls um Abwärtskompatibilität zu bewahren.

Ein kleines Pythonskript [Anhang 2] zur direkten Steuerung des Roboters wurde außerdem entwickelt um die Anforderungen der komplexen Steuerungssoftware für schnelle Tests zu umgehen. Es stelle sich später außerdem sehr hilfreich zu Evaluation heraus um die Tests in großen Teilen zu automatisieren. Ein Anwendungsfall für den die Steuerungssoftware nicht entwickelt wurde.

3. Steuerungssoftwareanpassungen

Durch die neue Protokollversion konnte die Steuerungssoftware schnell mit dem neuen Roboter lauffähig gemacht werden ohne den aktuellen Betrieb zu beeinflussen. Je nach erkannter Roboter-Version wurden die alten Geschwindigkeitswerte auf das neue Protokoll angepasst bevor sie an den Roboter gesendet wurden. So konnte schnell verifiziert werden, dass der neue Roboter ebenso funktioniert und erste Tests gemacht werden.

Beim Umbau fiel auf, dass die Software intern nicht mit metrischen Einheiten arbeitete, sondern mit dem Tastverhältnis des PWM Signals zur Codierung der Geschwindigkeit. Einige Teile des Programms rechneten, um diese Einschränkung zu umgehen, diesen Wert mittels einer vorher durch eine Testreihe erstellte und anschließend gefittete Funktion in einen metrischen Wert um.

Diese Funktion war es auch, die die schnelle Unterstützung des neuen Roboters möglich machte, da der alte Wert bequem in einen metrischen umgerechnet werden konnte und so an den neuen Roboter gesendet wurde. Die angestrebte Genauigkeit litt unter diesen unnötigen und teils mehrfachen Umrechnungen allerdings erheblich.

So wurden die eh notwendigen Umbaumaßnahmen dazu genutzt, gleich die gesamte Codebase auf metrische Einheiten umzustellen und Abwärtskompatibilität zu den alten Robotern über die Inverse genannter Funktion bereitzustellen, so dass nun maximal eine Konvertierung und diese auch nur für die ohnehin weniger präzisen alten Roboter notwendig ist.

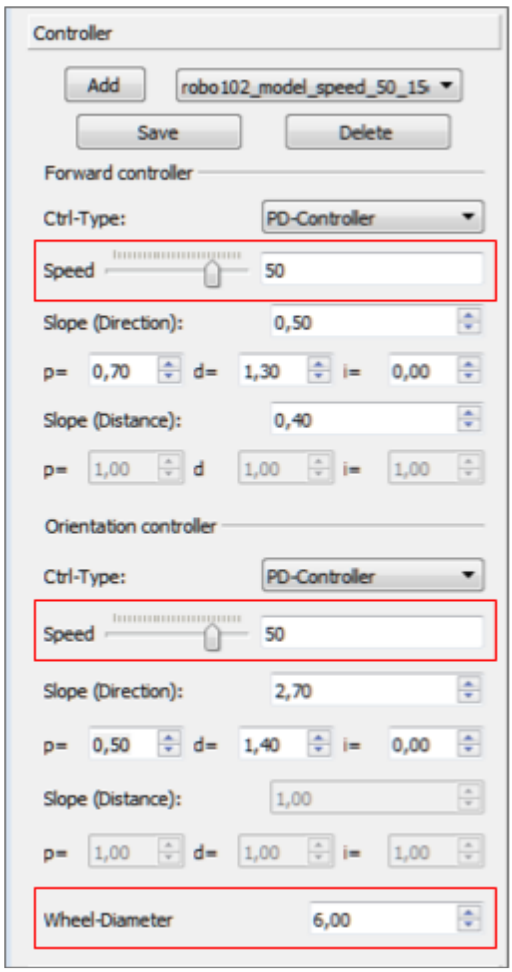

Abbildung 3.3.1: GUI der Roboter-Konfiguration.

Hervorgehoben sind Änderungen. Hinzugefügt wurden stufenlose Geschwindigkeitsslider in cm/s (vorher diskret und Duty-Cycle orientiert), sowie ein Feld zum Setzen des Reifendurchmessers zur Anpassung an für die Geschwindigkeitsberechnung.

Zuletzt wurden GUI Elemente (siehe Abbildung 3.3.1) zu den neuen Parametern der Geschwindigkeitsberechnung wie zum Beispiel Reifendurchmesser hinzugefügt und sämtliche neuen Berechnungen durch Code-Annotationen und einen externen Wikiartikel [Anhang 1] dokumentiert.

Evaluation

Zur Evaluation wurde die tatsächliche Geschwindigkeit mit der im Vorfeld berechneten erwarteten Geschwindigkeit verglichen.

Ein Beispiel: Der getestete Roboter besaß einen Reifendurchmesser von `6 cm`. Bei 100% Tastverhältnis (konstant 5V) erreicht der Motor eine Input-Rotation von bis zu `5000 rpm`. Um eventuelle Schwankungen auszugleichen, sowie um eine höhere Genauigkeit beim Setzen von Geschwindigkeiten zu erreichen, wurde diese auf `3500 rpm` durch die Motor-Konfiguration begrenzt. Durch das Umsetzungsverhältnis von `33:1` des Motors wird eine tatsächliche Rotationsgeschwindigkeit von `3500 / 33 = 106` rpm maximal erreicht.

Mit diesen Werte können wir die zu erwartende Geschwindigkeit (`x in cm/m`) bei gegebenem Tastverhältnis (`t in 0-255`) errechnen.

```
x = t / 255 * 106 * 6
```

Der Versuchsaufbau basierte hierbei auf dem Aufbau der Experimente. Der Roboter wurde unter dem Fischtank platziert und eine Replica im Tank entsprechend positioniert. Mittels einer Kamera wurden mehrere gefahrene Strecken der Replica aufgezeichnet. Jede Strecke hätte nach der berechneten Geschwindigkeit 30 cm lang sein sollen. Der Roboter fuhr stets die Zeit, die er nach gegebener Geschwindigkeit für 30 cm hätte brauchen sollen. Die Aufnahmen [Anhang 3] wurden danach in Einzelbilder umgerechnet und entsprechend gekürzt, sodass das Video nur die tatsächliche Bewegung zeigte sowie zusätzlichen ohne zehn Frames am Anfang und Ende um Beschleunigung und Bremsung nicht zu erfassen. Aus der verbleibenden Anzahl der Frames wurde mit Hilfe der Framerate die Fahrzeit berechnet. Durch ein entlang der Versuchsstrecke gelegtes Maßband konnte die zurückgelegte Entfernung des Roboters zwischen dem ersten und letzten Frame des verkürzten Videos im Nachhinein bestimmt werden. Hiermit ließ sich dann die tatsächliche Geschwindigkeit bestimmen [Anhang 4].

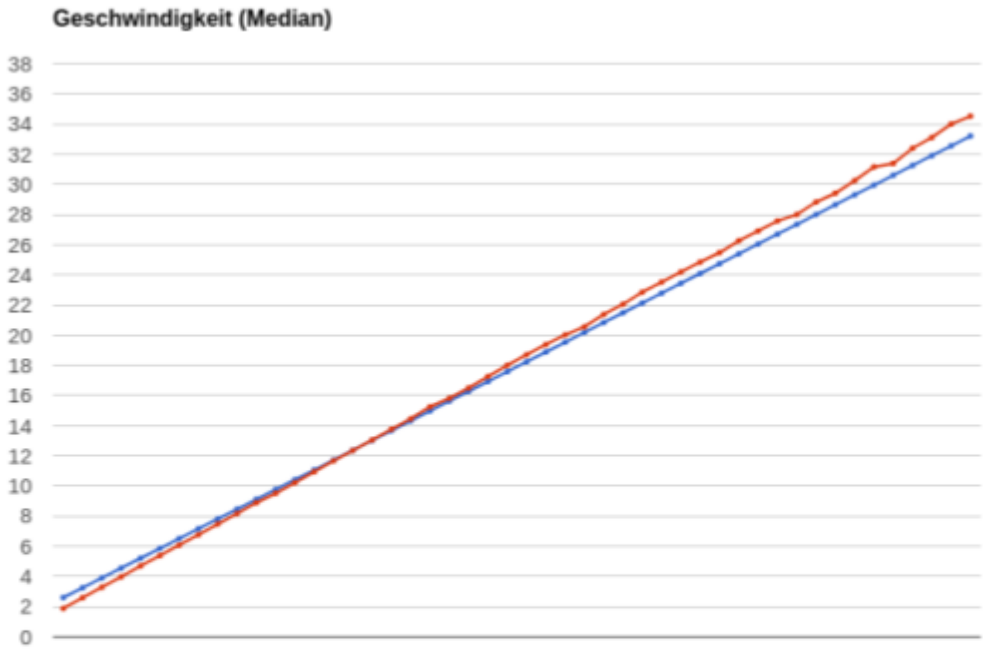

Abbildung 4.1: Erwartete Geschwindigkeit vs Messungen
Blau: erwartete Geschwindigkeit des Roboters bei gegebenen Geschwindigkeiten (links in cm/s).
Rot: Median der tatsächlichen Geschwindigkeit aus vier Messungen

Die Experimente wurden alle 5 Einheiten des codierten modulierten Tastverhältnis des PWM Signals des Arduino (`0-255`) beginnend bei `20` durchgeführt. Außerdem wurde jede Messung drei Male wiederholt. Das Ergebnis des Medians aller vier Werte gegen die erwartete Geschwindigkeit wird durch Abbildung 4.1 gezeigt. Die maximale Abweichung beträgt `1,3 cm/s` im Extremfall bei höchstmöglicher Geschwindigkeit. Die mittlere absolute Abweichung beträgt `0,3 cm/s`.

Diskussion

Zwei Faktoren haben vermutlich zu dem Fehler beigetragen, die noch verhindert werden könnten.
Zum einen ist der Roboter nicht immer genau am Maßband entlang gefahren, da sich die Orientierung durch die starke plötzliche Beschleunigung bei besonders hohen Geschwindigkeiten oft verdreht hat, sodass die Distanzmessung Messfehlern insbesondere bei hohen Geschwindigkeiten unterliegen könnte.

Zum anderen hätte vorher evaluiert werden müssen, wie der Motor besonders kleine Werte interpretiert. Da der Motor nach Datenblatt garantiert erst ab 0,3 V rotiert, stellt sich die Frage, mit welcher Mindestgeschwindigkeit sich der Motor mit 0,3 V tatsächlich dreht. Die aktuelle Geschwindigkeitsberechnung geht von einem linearen Anstieg ab 0 V aus, diese Annahme ist eventuell falsch und könnte zu beobachteter Abweichung beitragen.

Die aktuelle Genauigkeit ist erstens jedoch ausreichend für die geplanten Experimente und stellt zweitens eine deutlich Verbesserung gegenüber dem ungetesteten alten Robotermodell dar.

Insofern wurde das Ziel erreicht und eine präzise Steuerung ermöglicht.

Ausblick

Eine neue kompaktere Version mit den gleichen Motoren ist aktuell in Entwicklung von Hauke Moenck. Durch die kompaktere Bauform, sollen mehrere Roboter einfacher gleichzeitig agieren können.

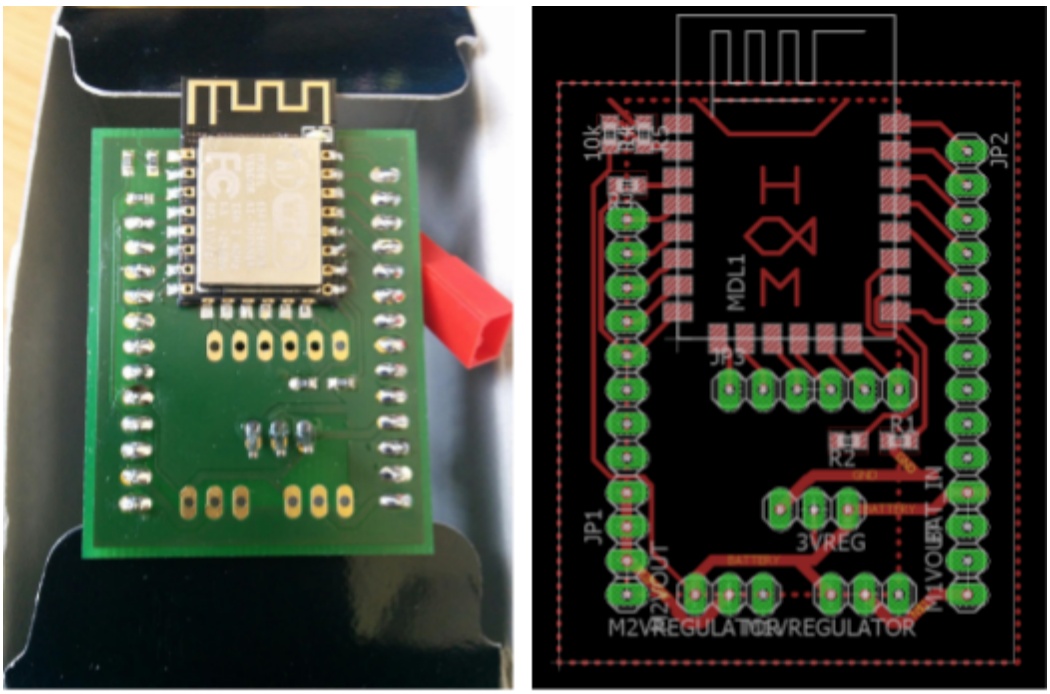

Abbildung 5.1: Links das neue PCB mit ESP-12F Modul, rechts das Board-Layout.

Die kompaktere Bauweise wird durch ein deutlich kleineres PCB erreicht (siehe Abbildung 5.1). Dieses beinhaltet bereits Spannungsteiler und Regler für die Motoren und das nun zum Einsatz kommende ESP-12F Modul. Letzteres ist mit Arduino und Wifi-Shield Software kompatibel, sodass die neuen Roboter mit minimalen Firmwareanpassungen direkt lauffähig sind.

Anhang

[1] Robofish v3 Wikiartikel

[2] Source Code

[3] Messungsvideos

[4] Messungsevalutation

[5] Datenblatt Faulhaber 2622S006B SC 33:1

[6] Handbuch Faulhaber 2622S006B SC 33:1

[7] Arduino WifiShield Overview

Literaturverzeichnis

[1] Interactive Robotic Fish for the Analysis of Swarm Behavior.

Raúl Rojas, Tim Landgraf, Hai Nguyen, Stefan Forgo, S. Schneider, J. Schröer, C. Krüger, Henrik Matzke, R. Clément, Jens Krause
Published 12 June 2012 - ICSI 2013. Lecture Notes in Computer Science, vol 7928.

[2] RoboFish: increased acceptance of interactive robotic fish with realistic eyes and natural motion patterns by live Trinidadian guppies

Tim Landgraf, David Bierbach, Hai Nguyen, Nadine Muggelberg, Pawel Romanczuk and Jens Krause
Published 12 January 2016 - Bioinspiration & Biomimetics, Volume 11, Number 1

[3] Investigation of Collective Behaviour and Electrocommunication in the Weakly Electric Fish, Mormyrus rume, through a biomimetic Robotic Dummy Fish

Elisa Donati, Martin Worm, Stefano Mintchev, Marleen van der Wiel, Giovanni Benelli, Gerhard von der Emde and Cesare Stefanini
Published 30 November 2016 - Bioinspiration & Biomimetics, Volume 11, Number 6

[4] Fish and robots swimming together in a water tunnel: robot color and tail-beat frequency influence fish behavior.

G Polverino, P Phamduy, M Porfiri
Published 25 October 2013

[5] Fish and robot dancing together: bluefin killifish females respond differently to the courtship of a robot with varying color morphs

P Phamduy, G Polverino, R C Fuller and M Porfiri
Published 27 August 2014 - Bioinspiration & Biomimetics, Volume 9, Number 3

[6] Collective Response of Zebrafish Shoals to a Free-Swimming Robotic Fish

Sachit Butail, Tiziana Bartolini, Maurizio Porfiri
Published 16 October 2013

[7] A Robotics-Based Behavioral Paradigm to Measure Anxiety-Related Responses in Zebrafish

Valentina Cianca, Tiziana Bartolini, Maurizio Porfiri, Simone Macrì
Published 29 July 2013

[8] Modulation of risk-taking behaviour in golden shiners (Notemigonus crysoleucas) using robotic fish

Nicole Abaid, Stefano Marras, Corine Fitzgibbons, Maurizio Porfiri
Published 20 July 2013

[9] Mosquitofish (Gambusia affinis) responds differentially to a robotic fish of varying swimming depth and aspect ratio

Giovanni Polverino, Maurizio Porfiri
Published 14 May 2013

[10] Zebrafish responds differentially to a robotic fish of varying aspect ratio, tail beat frequency, noise, and color

Nicole Abaid, Tiziana Bartolini, Simone Macrì, Maurizio Porfiri
Published 4 June 2012

[11] Infiltrating the zebrafish swarm: design, implementation and experimental tests of a miniature robotic fish lure for fish–robot interaction studies

Frank Bonnet, Yuta Kato, Jos´e Halloy, and Francesco Mondada
Published 25 July 2016

[12] Real-time feedback-controlled robotic fish for behavioral experiments with fish schools

Daniel T. Swain, Iain D. Couzin, Naomi Ehrich Leonard
Published 19 October 2011