

Diplomarbeit

„Automatische Fischfütterungsanlage mit Webinterface“

I. Erklärung

Hiermit erklären wir, dass wir diese Diplomarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt und keine anderen als die angegebenen Quellen oder Hilfsmittel verwendet haben.

Schwaz, am 10. Mai 2012

Innsbruck, am 10. Mai 2012

.....

Bernhard Fritz

.....

Patrick Schwarz

II. Kurzfassung

Der Einsatz von Computern spielt auch in der artgerechten Fischzucht eine wichtige Rolle. In den letzten Jahren wurden automatische Fischfütterungsanlagen nur für große Fischzuchtbetriebe im Ausland entwickelt. Die meist familiären Betriebe und wissenschaftlichen Projekt-Betreiber in Österreich können sich diese hohen Investitionskosten nicht leisten.

Diese Diplomarbeit beschäftigt sich mit der Ausarbeitung eines Konzeptes sowie der Entwicklung eines Prototyps für eine **automatische Fischfütterungsanlage mit Webinterface**, wie es in Österreich für diese spezielle Zielgruppe umgesetzt werden kann.

Primäre Anforderung war die Entwicklung einer individuellen Software-Lösung, die nicht nur einen Futterautomaten selbständig steuert, sondern auch wichtige Parameter überwacht, allfällige Störungen im Betriebsablauf protokolliert und an eine „Zentrale“ meldet. Die Programmiersprache wurde mit JAVA gewählt, weil diese Programmiersprache plattformunabhängig ist und somit auf jedem Betriebssystem ausgeführt werden kann. Daneben werden über diese Programmiersprache Socket- bzw. Datenbankverbindungen aufgebaut und zeitgesteuerte Prozesse umgesetzt.

Für die Entwicklung des Prototyps wurden vergleichsweise kostengünstige Arbeitsbehelfe verwendet, weil dieses vereinfachte Modell eines Futterautomaten nur als Einzelstück gefertigt wurde. Für den Einsatz im Echtbetrieb muss dieser Futterautomat noch „nachgerüstet“ werden.

Mit dem vorliegenden Prototypen und der entwickelten Software wurde aus der Sicht der Verfasser das Ziel erreicht,

- Futtermaterial in definierten Mengen und beliebigen Intervallen an einer oder an mehreren Fütterungsstellen abzugeben (verlässliche Fütterung)
- den Einsatz menschlicher Arbeitskraft „vor Ort“ zu optimieren (Arbeitszeit- und Kosteneinsparung)
- spezielle Daten über Sensoren zu erheben, in Datenbanken abzulegen und auf Knopfdruck auszuwerten (Statistik)
- wichtige Informationen an den Fischzüchter/Projekt-Betreiber rasch weiterzuleiten (Warnung mittels E-Mail)

Abschließend werden noch Erweiterungen des entwickelten Programmes diskutiert, die für die Verwendung im kommerziellen und wissenschaftlichen Einsatz notwendig sind oder von Vorteil wären, wie z.B. Zuchtbeckenüberwachung mittels Webcam, automatische Steuerung der Bruthaus-Beleuchtung, Auswertung von Futterdaten.

In der Aufzucht von Fischen – mit unternehmerischem oder wissenschaftlichem Hintergrund – spielt die Fütterung eine bedeutende Rolle. Einerseits soll das eingesetzte Futter einen größtmöglichen Zuwachs bringen, auf der anderen Seite soll nicht zu viel menschliche Arbeitszeit für die Fütterung aufgewendet werden. In diesem „Spannungsfeld“ haben die Verfasser der vorliegenden Dokumentation eine Lösung vorgestellt, die mit einem angemessenen Kostenaufwand in der Praxis umgesetzt werden kann.

III. Abstract

In these days, you cannot think of a life without computers. Even fish farms have to rely on computer technology. Up to now, automatic fish feeding systems were only developed for huge fish farms abroad. Most of Austrian's fish farms wouldn't be able to afford the high investment costs.

This diploma project is about developing a general concept as well as a low-budget fish feeding prototype for this specific target group.

A primary goal was the development of a custom software solution, which not only controls the fish feeding device by its own, but also monitors many different parameters, such as sensor values as well as any malfunctions, which will be reported to a "central". For programming the language JAVA was chosen because JAVA is well-known for its platform independence and can be executed on any operating system. Furthermore this programming language is capable of establishing socket and database connections as well as implementing time-controlled processes, which are essential for this project.

By simplifying the prototype the project team was able to reduce the costs to an affordable minimum. For real operation this prototype would need some "upgrades".

By developing the prototype as well as the custom software following goals were achieved:

- It is possible to drop a specific amount of feed at a pre-defined time
- Less human resources are needed to fulfill the fish feeding tasks (cost savings)
- Sensor data is fetched continuously and is written into a database (statistics)
- Warnings are immediately sent to the owner of the fish feeding system by E-Mail

In the end of this diploma thesis the project team is going to refer to various extensions which are necessary to make use of the prototype for commercial and scientific reasons. For example:

- Automatic hatchery lights
- Using webcams for observing breeding basins
- Creating statistics for feed consumption

Breeding fish – for commercial or scientific reasons – depends on the right feeding methods. On the one hand feeding the fish improves their growth-rate. On the other hand feeding requires a lot of human work which is very expensive. For this reason the team of the diploma project has written down their low-budget solution to eliminate this problem.

IV. Vorwort

Das Schuljahr 2012 neigt sich seinem Ende zu. Für die Schüler des 5. Jahrganges der HTL-Anichstraße sind diese Tage noch einmal eine besonders lernintensive Zeit, gilt es doch, sich nach dem Abschluss der 5-jährigen Ausbildung auf die Reifeprüfung vorzubereiten. Dieses „Schicksal“ teilen wir

Bernhard FRITZ und Patrick SCHWARZ

mit anderen Schülern unseres Jahrganges, Zweig Wirtschaftsingenieurwesen. In den letzten fünf Jahren wurden wir in allgemeinbildenden Fächern, wie z.B. Deutsch, Mathematik, Englisch, Physik, Chemie und Geschichte unterrichtet. Darüber hinaus konnten wir ein umfassendes Wissen in den Bereichen Informatik, Technik, Wirtschaft und Kommunikation, uvm., erwerben. Unsere Professoren haben uns nicht nur solide ausgebildet, sondern uns auch tolle Chancen für unsere Zukunft eröffnet. Erst mit dieser Ausbildung haben wir die Möglichkeit, an einer Fachhochschule oder Universität weiter zu studieren oder direkt in das Berufsleben einzusteigen – **dafür wollen wir uns bei den HTL-Professoren ganz herzlich bedanken.**

Unser theoretisches Wissen, aber auch handwerkliche Erfahrungen in der HTL-Werkstatt und viele praktischen Übungen aus Laborstunden haben wir uns als Diplomanden für die Diplomarbeit „Automatische Fischfütterungsanlage mit Webinterface“ zum Nutzen gemacht, die vorliegende Dokumentation zu verfassen. Die Entscheidung für dieses Thema gründet auf jahrelange Bemühungen des Tiroler Fischereiverbandes und namhafter Limnologen der Universität Innsbruck, die mit viel Engagement und persönlichem Einsatz versuchen, heimische Fischarten vor dem Aussterben zu bewahren. Dabei werden Elterntiere von ausgewählten Tiroler Gewässern zur Gewinnung von Laichmaterial entnommen aus dem in weiterer Folge Jungfische in einer heimischen Fischzucht gewonnen und später wieder in die Ursprungsgewässer ausgesetzt werden. Auf diese und ähnliche wissenschaftliche Arbeiten hat der Tiroler Fischereiverband in seiner Einladung zum „Tag der Offenen Tür“ in der Fischzuchtanstalt Thaur am 26. August 2011 hingewiesen. Bei dieser Veranstaltung hat der anwesende Limnologe der Universität Innsbruck die Diplomanden über praxisrelevante Aspekte für die Rettung der vom Aussterben bedrohten Tiroler Inn-Äsche informiert. Während dieser Veranstaltung hat der vortragende Referent den Interessierten das Bruthaus, die Zuchtbecken und das umliegende Gelände der Fischzucht Thaur ausführlich erklärt. Nachdem die im Rahmen der wissenschaftlichen Aufzucht anfallenden Kosten nur durch Subventionen des Landes Tirol gedeckt werden können, ist es besonders wichtig, mit den vorhandenen Ressourcen (u.a. Futterkosten, menschliche Arbeitskraft) möglichst „schonend“ umzugehen.

Nunmehr wurde von uns Diplomanden untersucht, ob der Einsatz eines **individuell computergesteuerten Futterautomaten** für kleine Fischzuchtanstalten und Betreiber wissenschaftlicher Projekte möglich ist. Für diese Dokumentation haben wir einen „Prototyp“ eines kostengünstig und verlässlich arbeitenden, elektronisch gesteuerten Futterautomaten mit vergleichsweise geringem Mitteleinsatz gebaut. Mit dieser Lösung wollen wir Diplomanden nur die Möglichkeiten einer individuellen Computer-Steuerung mit den Schwerpunkten verlässliche Futterabgabe ohne direkten Einsatz menschlicher Arbeitskraft „vor Ort“ aufzeigen, die Messung verschiedener Parameter mit allfälligem Einfluss auf das Fressverhalten von Jungfischen (Luft-/Wasser-Temperatur) ermöglichen, die automatische Rückmeldung von Störungen im planmäßigen Fütterungs-Ablauf umsetzen sowie die Protokollierung entscheidungsrelevanter Daten sicherstellen.

Dieser „Prototyp“ eines individuell steuerbaren Fischfütterungs-Automaten soll ein funktionsfähiges, aber vereinfachtes Versuchsmodell darstellen. Es wird aber allein aufgrund der verwendeten Materialien (Legobausteine, Elektromotor, usw.) rein äußerlich einem serienreifen Endprodukt nicht entsprechen. Dieser Prototyp wurde als Einzelstück gebaut, das nur ein bestimmtes Konzept – die elektronisch steuerbare Fischfütterung – illustrieren soll. In Bezug auf Technik (Software-Einsatz!) kann das „serienreife Endprodukt“ mit dem Versuchsmodell weitgehend identisch sein.

Die beiden Diplomanden wollen sich an dieser Stelle bei dem diplomarbeit-begleitenden HTL-Fachlehrer, **Herrn Engelbert Gruber**, ganz herzlich bedanken, der bei der Erstellung der Diplomarbeit mit seinem umfassenden Wissen jederzeit zur Verfügung gestanden ist.

Schwaz, am 10. Mai 2012

Innsbruck, am 10. Mai 2012

.....
Bernhard Fritz

.....
Patrick Schwarz



Abbildung 1: Bruthaus mit mechanischen Futterautomaten in der Fischzuchtanlage Thaur



Abbildung 2: Zuchtbecken in der Fischzuchtanlage Thaur

Inhaltsverzeichnis

1	Einführung.....	12
2	Allgemeines	14
2.1	Ideenfindung.....	14
2.2	Ziel.....	14
2.3	Funktionsumfang	14
2.4	Programmtechnische Umsetzung	14
3	JAVA	15
3.1	Was ist JAVA?.....	15
3.2	Warum JAVA?	15
3.3	Vor- und Nachteile?	15
3.4	Alternativen?	15
3.5	Genaueres zum Code?	16
3.5.1	DBManager.....	16
3.5.2	Wie funktioniert das Aufbauen einer Datenbankverbindung in Java?	16
3.5.3	Wie funktioniert die Kommunikation mit dem AVR-NetIO-Board?.....	16
3.5.4	SocketManager.....	16
3.5.5	Wie funktioniert das Einlesen von Sensorwerten?	17
3.5.6	Wie funktioniert das Schreiben/Lesen von Dateien?	18
3.5.7	Property-Files	18
3.5.8	log4j	18
3.5.9	JFreeChart.....	19
3.6	Programmkonzepte	20
3.6.1	Konzept „DBManager“	20
3.6.2	Konzept „Fetch“	21
3.6.3	Konzept „Feed“	22
3.6.4	Konzept „MainThread“	23
3.7	Klassen	24
3.7.1	Klasse Tool	24
3.7.2	Klasse „SocketManager“	27
3.7.3	Klasse DBManager	28
3.7.4	Klasse Daten	29

3.7.5	Klasse „JavaMailThread“	29
3.7.6	Klasse Data	30
3.7.7	Klasse MainThread	30
3.7.8	Klasse Launcher	30
3.8	Servlets	30
3.8.1	Das Konfigurations-Servlet	30
3.8.2	Das Datenbank-Servlet	31
3.8.3	Das Anmelde-Servlet	31
3.8.4	Das Setup-Servlet	31
3.8.5	Das Abmelde-Servlet	31
3.8.6	Das ChartViewer-Servlet	31
3.8.7	Das LuftThermo-Servlet	32
3.8.8	Das WasserThermo-Servlet	32
3.9	JavaServer Pages	32
3.10	Apache Tomcat Server	32
4	Eclipse IDE	34
4.1	Was ist Eclipse?	34
4.2	Verwendung	34
4.2.1	Java	34
4.2.2	Eclipse installieren	34
4.2.3	Eclipse starten	35
4.2.4	Arbeitsverzeichnis(Workspace)	35
4.2.5	Benutzeroberfläche	35
5	MySQL	37
5.1	Verwendung mit JAVA	37
5.2	MySQL-Statements	37
5.2.1	CREATE-Statement	37
5.2.2	INSERT-Statement	37
6	SQLite	38
6.1	Verwendung mit JAVA	38
6.2	SQLite-Statements	38
7	HTML	39
7.1	Was ist HTML?	39

7.2	Webinterface	39
7.2.1	Warum?	39
7.2.2	Design	39
7.2.3	Features	39
7.2.4	Webinterface Screenshots	40
8	AJAX (Asynchronous Javascript and XML)	46
8.1	Allgemeines über AJAX	46
8.2	AJAX in unserer Diplomarbeit	47
9	Github - „Social Coding“	48
9.1	Was ist Github?	48
9.2	Funktionsumfang	48
9.3	Alternativen	48
9.4	git – Grundlagen	49
9.5	Wie wird 'github' verwendet?	51
9.5.1	Einrichtung	51
9.5.2	Verwendung	54
10	Elektronik	57
10.1	Feststellen des Widerstandswerts eines Widerstands	57
10.2	Löten	57
10.3	Verdrahtung	58
10.4	Sensorschaltung	58
10.5	Verstärkerschaltung	59
10.6	Diodenschaltung	60
10.7	Komplette Schaltung	60
11	Hardware	62
11.1	AVR-NET-IO-Board	62
11.1.1	Allgemeines über das AVR-NET-IO-Board	62
11.1.2	Features	62
11.1.3	Befehle	63
11.1.4	Alternativen	63
11.2	K8IO-Relaiskarte	64
11.2.1	Allgemeines über die K8IO-Relaiskarte	64
11.2.2	Relais	64

11.2.3 Alternativen	64
11.3 KTY-222 Temperatursensor	65
11.4 LM324-Verstärkungs-IC	65
11.5 Lichtschrankensensor EESX-671	66
12 Materialkosten	66
13 Zusammenbau	67
13.1 Grundplatte	67
13.2 AVR-NET-IO-Board	67
13.3 K8IO-Relaiskarte	67
13.4 Streifenrasterplatine.....	67
13.5 Lego-Motor und -Getriebe	67
13.6 Futterbecher	68
13.7 Ausgleichgewicht	68
13.8 Galgen.....	68
13.9 Kompletter Aufbau	69
14 Sicherheit	71
14.1 Login.....	71
14.2 Session	71
14.3 Szenarien	71
14.3.1 Server fällt aus.....	71
14.3.2 Strom fällt aus	71
14.3.3 Netzwerkverbindung unterbrochen.....	71
14.3.4 Internetverbindung unterbrochen.....	72
15 Umsetzungsentscheidungen.....	73
15.1 Konstruktion „Box mit Förderband“	73
15.1.1 Allgemeines über diese Konstruktionsidee.....	73
15.1.2 Warum haben wir uns nicht für diese Möglichkeit entschieden?	73
15.2 Konstruktion „Rohr“	73
15.2.1 Allgemeines über diese Konstruktionsidee.....	73
15.2.2 Warum haben wir uns nicht für diese Möglichkeit entschieden?	74
15.3 Programmiersprache JAVA.....	74
15.3.1 Allgemeines über diese Programmiersprache	74
15.3.2 Warum haben wir uns für diese Programmiersprache entschieden?	74

15.4	MySQL-Datenbank.....	75
15.4.1	Allgemeines über diese Datenbank.....	75
15.4.2	Warum haben wir uns nicht für diese Datenbank entschieden?	75
15.5	SQLite-Datenbank.....	75
15.5.1	Allgemeines über diese Datenbank.....	75
15.5.2	Warum haben wir uns für diese Datenbank entschieden?.....	75
15.6	AVR-NET-IO-Board und K8IO-Relaiskarte	75
15.6.1	Allgemeines über das AVR-NET-IO-Board	75
15.6.2	Allgemeines über die K8IO-Relaiskarte	75
15.6.3	Warum haben wir uns für diese Hardware entschieden?	75
15.7	Verstärkerschaltung.....	76
15.7.1	Allgemeines über die Verstärkerschaltung	76
15.7.2	Warum haben wir uns nicht für diese Möglichkeit entschieden?	76
15.8	Diodenschaltung.....	76
15.8.1	Allgemeines über die Diodenschaltung.....	76
15.8.2	Warum haben wir uns nicht für diese Möglichkeit entschieden?	76
16	Nennenswerte Probleme	77
16.1	Temperaturmessung	77
16.2	Diodenschaltung.....	77
16.3	Verstärkerschaltung.....	77
16.4	Messaufbau	77
16.5	Apache und Twitter	77
16.6	Apache - SQLite Datenbankpfad.....	78
16.7	Apache - Grafik	78
16.8	Socketmanager	79
16.9	JFreeCharts	79
17	Erkenntnisse aus der Diplomarbeit.....	80
18	Ausblick	81
19	Zusammenfassung, Ergebnis	82

1 Einführung

Solange es Menschen gibt, ist auch der Fischfang betrieben worden. Von den ersten Anfängen einer eigentlichen Fischzucht bzw. von der Anlage von Teichen berichten die römischen Schriftsteller Cicero und Plinius im ersten Jahrhundert vor Christus. Demnach haben einige reiche Römer Fischteiche und -Becken angelegt. Der Beginn der eigentlichen Aufzucht liegt in den Jahren um 500 n.Chr. Aus dem 16. Jahrhundert stammen auch die ersten Bücher über die Teichwirtschaft, **Einrichtung von Laich- und Brutteichen und Fütterungsplänen**. Erst in der ersten Hälfte des 18. Jahrhunderts verfolgte man die Idee, den laichreifen Fischen Eier und Samen zu entnehmen, die Eier künstlich zu befruchten und zur Entwicklung zu bringen (Geburtsstunde der „künstlichen Fischzucht“). Man beschäftigte sich zunächst ausschließlich mit der Vermehrung und Aufzucht der heimischen Fischarten. Parallel dazu entstanden die Vorläufer der verschiedenen Fischereiorganisationen und -verbände, die sich die Förderung der künstlichen Fischzucht (einem Teil der Landwirtschaft) und eine systematische Besetzung der Gewässer mit Jungfischen zum Ziel setzten.¹

Bei der professionellen Aufzucht von Fischen in der Neuzeit spielen für die Wirtschaftlichkeit der unternehmerischen Tätigkeit die Faktoren **Futterkosten** und im Speziellen die für die Fütterung eingesetzte **menschliche Arbeitszeit** eine wesentliche Rolle. Nunmehr gilt es, alle eingesetzten Ressourcen so zu optimieren, dass einerseits alle Anforderungen einer erfolgreichen Fischaufzucht mit sparsamen Ressourcen-Umgang erfüllt werden, andererseits aber eine ausreichende Wertschöpfung erreicht wird.

Fischzüchter in unserem heimatlichen Umfeld, welche die Aufzucht in **kleineren Unternehmensformen** (z.B. Familienbetrieben) betreiben, stehen deshalb vor der nicht ganz einfachen Ausgangssituation, dass sie **technische Hilfsmittel** wie sie etwa von riesigen nur im Ausland ansässigen Fisch-Mastbetrieben eingesetzt werden, aus wirtschaftlichen Überlegungen nicht nutzen können. Neben diesen klein strukturierten Betrieben ist an dieser Stelle auch noch eine Reihe von **wissenschaftlichen Projekten** zu erwähnen, die sich in unserem Bundesland auch mit der Aufzucht von Fischen befassen – z.B. Tiroler „Äschen“-Projekt (Betreiber: Tiroler Fischereiverband) oder Projekt „Urforelle“ (Betreiber: länderübergreifend, u.a. Universität Innsbruck). Diesen Projekten gemeinsam ist die Entnahme von Elterntieren aus ausgewählten Gewässern, die wissenschaftliche Untersuchung, die Gewinnung von Laichmaterial, die Aufzucht von Jungfischen bzw. der Aufbau ursprünglicher Zuchtstämme in einer **heimischen Fischzucht** und das Aussetzen der in der Fischzucht reproduzierten Fische im Ursprungsgewässer.

Bis zum erfolgreichen Abschluss von Aufzucht-Programmen müssen die Züchter eine ganze Reihe von **Anforderungen an die Fischfütterung** erfüllen, wobei eine **individuelle Computer-Steuerung** die Zielerreichung (zuverlässige, alltagstaugliche Fischfütterung in einem kleineren Rahmen) wesentlich **erleichtert**, und zwar:

- Fische sollen täglich regelmäßig, mehrmalig, langsam und angepasst an das Umfeld gefüttert werden. Die entsprechenden Futtermengen und Fütterungsintervalle können optimal an das Fressverhalten der Fische angepasst werden.

¹ (Greenberg, 1966)

technische Lösung:

Aufstellung eines/mehrerer computergesteuerter Futterautomaten

Jungfische, insbesondere forellenartige Fische, folgen in der Natur bei ihren Fressgewohnheiten keinem Zeitplan. Daraus schließt ein Fischzüchter, dass es umso besser ist, je öfter in der Gefangenschaft gefüttert wird, besonders wenn die Fische jung sind. Die Brut wird täglich etwa fünfmal gefüttert, kleine Setzlinge viermal, mittlere und große Setzlinge zwei- bis dreimal und heranwachsende Fische ein- bis zweimal täglich.

Der hemmende Faktor ist die **Arbeitskraft**. Deshalb müssen mehr und mehr **technische (elektronisch gesteuerte) Hilfen** für die täglichen Fütterungen zum Einsatz kommen.

- Präzise und unterschiedliche Futtergaben an beliebig vielen Futterstellen (auch Kleinstmengen, z.B. bei Brut, usw.)

technische Lösung:

Aufstellung des computergesteuerten Futterautomaten an mehreren Stellen; individuelles Vermischen von mehreren Komponenten je Futterstelle.

- Arbeitszeiterparnis und geringerer Personalaufwand durch automatische Fütterung, höhere Tageszunahmen durch regelmäßiges Weiterfüttern an Wochenenden, Feiertagen, Urlaubszeit.

technische Lösung:

Fütterungsintervalle über das Webinterface einprogrammieren

- Beliebig viele Betriebsstellen sind fernbedienbar und überwachbar

technische Lösung:

Fernwartung

Ergebnis

Die vorstehenden Ausführungen und die Erfahrungen von Fischzüchtern belegen, dass Rationalisierungsmaßnahmen nicht nur das A und O in der Wirtschaft sind, sondern für eine erfolgreiche Fischzucht neben besonderen kaufmännischen und wirtschaftlichen Gesichtspunkten auch die Verwendung neuer arbeitstechnischer Hilfsmittel, wie z.B. der **Einsatz einer elektronisch gesteuerten Fischfütterung**, unumgänglich ist. Das in einer Zeit, wo Arbeitskräfte – insbesondere fachvorgebildete – fehlen und alles darauf ankommt, einen klein strukturierten Betrieb mit überschaubaren finanziellen Mitteln zu technisieren und auch zu rationalisieren.

2 Allgemeines

2.1 Ideenfindung

Zur Bestimmung unseres Projektthemas haben wir versucht, möglichst viele Aspekte unserer Ausbildung zum Wirtschaftsingenieur in unsere Diplomarbeit einfließen zu lassen. Nach einem kurzen Brainstorming und Abwägen der Vor- und Nachteile der einzelnen Ideen haben wir uns für die automatische Fischfütterungsanlage mit Webinterface entschieden. Dieses Projekt verlangt informationstechnische Kompetenz sowie Kenntnisse in der Elektro- und Fertigungstechnik. In unserer fünfjährigen Ausbildung zum Wirtschaftsingenieur haben uns diese Fachgebiete besonders gefallen und so konnten wir unser Projekt mit voller Motivation und Begeisterung beginnen umzusetzen.

2.2 Ziel

Ziel unseres Projekts ist, eine vollfunktionierende, kostengünstige und automatische Fischfütterungsanlage mit Webinterface zu entwickeln. Das Webinterface soll ein Fernwarten der Fischfütterungsanlage ermöglichen und besonders einfach zu bedienen sein.

2.3 Funktionsumfang

- Es können bis zu 24 Fütterungen pro Tag konfiguriert werden
- Die Futtermenge kann je nach Belieben eingestellt werden
- Wasser- und Lufttemperatur werden kontinuierlich gemessen
- Ein userfreundliches Webinterface zur Überwachung und Konfiguration der Fischfütterungsanlage ist verfügbar
- Zur leichten Temperaturdatenauswertung werden Temperaturgraphen erstellt
- Bei zu hohen sowie zu niedrigen Temperaturen wird der User mittels E-Mail-Nachricht gewarnt
- Das Webinterface ist durch eine Anmeldeprozedur gegen Missbrauch geschützt
- Die Software läuft auf jedem beliebigen Betriebssystem

2.4 Programmtechnische Umsetzung

Im Rahmen der Planung war von den Verfassern der Diplomarbeit anfangs noch die Frage zu klären, welche Programmiersprache für die technische Umsetzung der automatischen Fischfütterungsanlage zum Einsatz kommen sollte.

Die Wahl fiel auf **JAVA**, weil Java

- eine einfache und vertraute Programmiersprache ist (reduzierter Sprachumfang)
- eine sichere Programmiersprache ist
- eine robuste Programmsprache ist (weniger ungewollte Systemfehler)
- eine leistungsfähige und effiziente Programmiersprache ist
- weil sie eine dynamische Programmsprache ist

3 JAVA

3.1 Was ist JAVA?

Java ist eine objektorientierte Programmiersprache und gleichzeitig auch eine eigene Plattform. Die Definition von Plattform ist die Kombination aus Computer und Prozessor. Da Java eine objektorientierte Programmiersprache ist, arbeitet man in Java nahezu nur mit Objekten. Sogenannte Klassen sind Blaupausen für diese Objekte!



Abbildung 3: Java-Logo

3.2 Warum JAVA?

Da wir seit der dritten Klasse das Programmieren mit Java lernen, auch in der Freizeit einiges in Java programmieren und dadurch bereits viel Erfahrung im Umgang mit Java gesammelt haben, entschieden wir uns für diese Programmiersprache. Ein weiterer Grund ist die große Unterstützung dieser Sprache im Internet, und die Plattformunabhängigkeit.

3.3 Vor- und Nachteile?

Vorteile	Nachteile
Java ist gratis	Java ist speicheraufwändig
Java läuft auf jedem Betriebssystem mit einer installierten Java virtuellen Maschine.	Java ist ein wenig langsamer als andere Programmiersprachen
In Java gibt es einigen bereits vorprogrammierten Code	Ein paar Klassenbibliotheken sind noch nicht fertig ausprogrammiert
Die Java Community ist die größte Programmiercommunity weltweit.	

Tabelle 1: Vor- und Nachteile von Java

3.4 Alternativen?

- C++
- C#
- Python

3.5 Genauer zum Code?

3.5.1 DBManager

Funktion dieser Klasse ist es die Verbindung zur Datenbank zu „managen“. Dabei ist die Art der Datenbank egal. Diese wird nämlich in einer Properties-Datei bestimmt. Mehr dazu später. Der DBManager verwaltet die Tabellen „sensordaten“ und „users“. Mittels diverser Methoden des DBManagers wird uns das schnelle speichern und auslesen von Sensordaten und Userdaten ermöglicht.

3.5.2 Wie funktioniert das Aufbauen einer Datenbankverbindung in Java?

Dazu ist ein .jar-Container für die jeweilige Datenbank nötig. Dieser beinhaltet den jeweiligen Code zum Aufbauen einer Verbindung zur jeweiligen Datenbank. Ein Beispiel dafür wäre z.B. der „MYSQL-Connector“ oder „SQLITE-JDBC“.

Code der beispielsweise beim Erstellen einer MYSQL-Datenbankverbindung aufgerufen wird:

```
Connection con;
public DBManager()
{
    Class.forName("com.mysql.jdbc.Driver");
    con = (Connection) DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/fish","root","");
}
```

3.5.3 Wie funktioniert die Kommunikation mit dem AVR-NetIO-Board?

Diese wird mithilfe des SocketManagers durchgeführt.

3.5.4 SocketManager

Aufgabe des SocketManagers ist es Socket-Verbindungen zu „managen“. Socket-Verbindungen werden dazu benötigt um eine Netzwerkverbindung mit unserem AVR-NetIO-Board aufzubauen. Die Methoden des Socket-Managers ermöglichen uns das schnelle Senden und Empfangen von Nachrichten. Beispiele für Befehle die wir dem AVR-NetIO-Board senden:

- GETADC 1 → Liefert die Spannung am ADC1 als ganzzahligen Wert der zwischen 0 und 1023 liegt.
- SETPORT 1 1 → Schließt das Relais 1 der am AVR-NetIO-Board angeschlossenen K8IO-Relaiskarte

3.5.5 Wie funktioniert das Einlesen von Sensorwerten?

Unser AVR-NetIO-Board sendet uns die Spannung als digitalen ganzzahligen Wert mithilfe des GETADC-Befehls, welcher von unserem SocketManager gesendet wird. Dieser Wert wird dann von unserer Tool-Klasse in eine Spannung umgerechnet. Das geschieht folgendermaßen:

$$ADC - Spannung = \frac{ADC - Wert * Referenzspannung}{2^{Bitbreite_{ADC}}}$$

Formel 1: Formel zur Berechnung der ADC-Spannung

Wobei die Referenzspannung in unserem Fall 5,0V und die Bitbreite 10 ist.

Beispiel (ADC-Wert ist 256):

$$ADC - Spannung = \frac{256 * 5,0}{1024} = 1,25V$$

Formel 2: Beispielhafte Berechnung einer ADC-Spannung

Diese Spannung muss nachher in einen Temperaturwert umgerechnet werden. Da bei unserem Temperatursensor (KTY-222) im Bereich zwischen 0 und 50° Celsius ein lineares Verhältnis zwischen Widerstand und Temperatur besteht, ist es möglich die Temperatur mittels einer arithmetischen Gleichung erster Ordnung zu bestimmen.

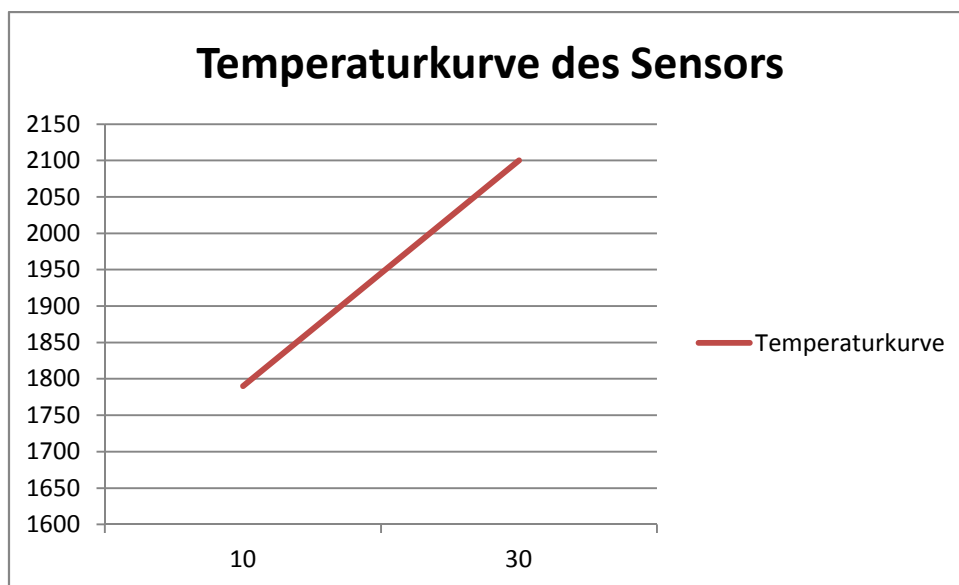


Abbildung 4: Temperaturkurve des Sensors

Berechnung der Geradengleichung:

$$y[\Omega] = k * x[^{\circ}C] + d$$

Formel 3: Geradengleichung

Durch Einsetzen in die Gleichung ist es uns möglich die Werte k und d zu bestimmen.

→ k=15,5

→ d=1635

3.5.6 Wie funktioniert das Schreiben/Lesen von Dateien?

Dazu schrieben wir uns zwei Methoden (read, write) in unsere Tool-Klasse.

Read-Methode:

Diese Methode hat als Parameter den Dateinamen der Datei und als Rückgabewert ein String-Array. Im Grunde genommen wird die Datei zeilenweise ausgelesen und in die einzelnen Zeilen in ein Array geschrieben. Das zu retournierende Array wird so vorbereitet, dass es nur so groß ist wie die Datei Zeilen beinhaltet.

Write-Methode:

Diese Methode hat als Parameter den Dateinamen der Datei und ein String-Array. Das String-Array beinhaltet die einzelnen Zeilen, die dann ausgelesen werden und in eine Datei geschrieben werden.

3.5.7 Property-Files

In Java ist es durchaus üblich sogenannte Property-Files zu erstellen. Diese Dateien erlauben es das Programm komfortabel zu konfigurieren. Als „Properties“ (deutsch Eigenschaften) werden statische Variablen konfiguriert wie z.B. Datenbank-Adressen, IP-Adressen usw. Property-Files haben üblicherweise über die Dateiendung „.properties“.

3.5.8 log4j

log4j ist ein Tool zum Loggen von Nachrichten (Informationen, Warnungen, Fehler...) in Java. Verwendet wird dieses in nahezu jedem größeren Open-Source-Java-Projekt. Es ist viel komfortabler dieses Tool zu verwenden als die von Java mitgelieferten Logger-Klassen. Der log4j-Logger kann mittels Property-Files konfiguriert werden.

Code zum Erstellen eines Loggers und ausgeben einer Informations-Nachricht:

```
public logger=Logger.getLogger („Logger“);  
logger.info(“Hello World!“);
```

3.5.9 JFreeChart

Das JFreeChart Projekt wurde im Jahre 2000 von David Gilbert ins Leben gerufen. Heutzutage ist JFreeChart die meist genutzte Graphenerstellungs-Software weltweit. Wie der Name schon sagt ist die Benutzung von JFreeChart kostenlos. JFreeChart lässt sich optimal in den Java Code integrieren.²

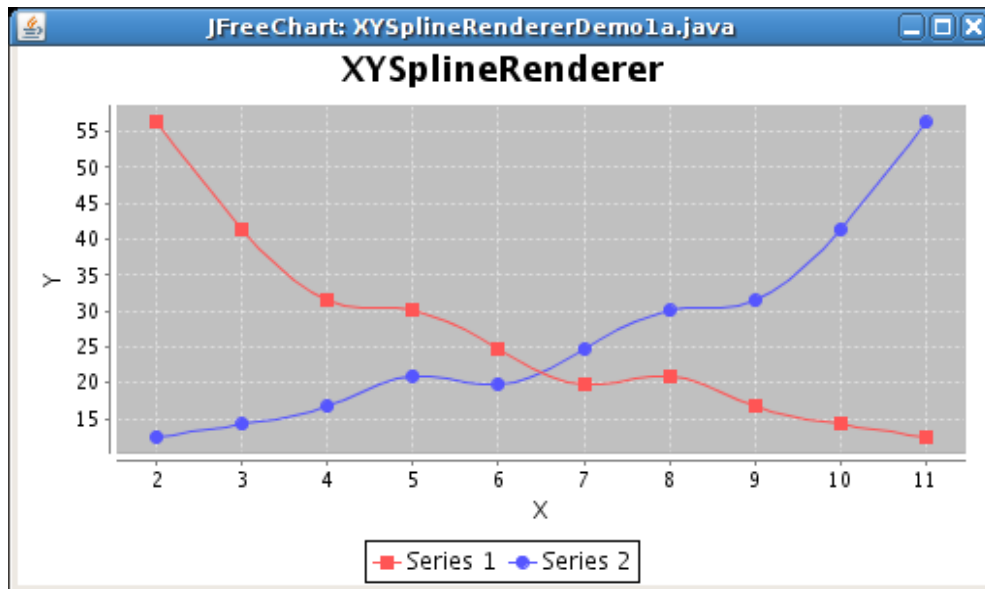


Abbildung 5: JFreeCharts - Beispiel

Alternativen zu JFreeChart:

- Simile Widgets
- rrdTool

Ähnlich wie JFreeChart sind diese Graphenerstellungs-Programme in der Lage Daten grafisch darzustellen. Der Nachteil bei diesen Anwendungen ist jedoch, dass sie nicht für Java entworfen wurden. Sie sind daher relativ umständlich in ein Java-Programm integrierbar. Deshalb fiel unsere Wahl auf JFreeChart.

² (JFreeChart)

3.6 Programmkonzepte

Bevor man mit der eigentlichen Programmierung beginnt, ist es sinnvoll ein sogenanntes Programmkonzept zu erstellen. Dieses stellt den Programmablauf grafisch dar und vermittelt somit leicht verständlich welche Aufgaben ein Programm verrichten soll.

Wir haben die Hauptaufgaben unseres Programms herausgenommen und versucht diese möglichst einfach und verständlich zu visualisieren.

3.6.1 Konzept „DBManager“

Dieses Konzept beschreibt wie in unserem Projekt grundsätzlich eine Datenbankverbindung aufgebaut wird.

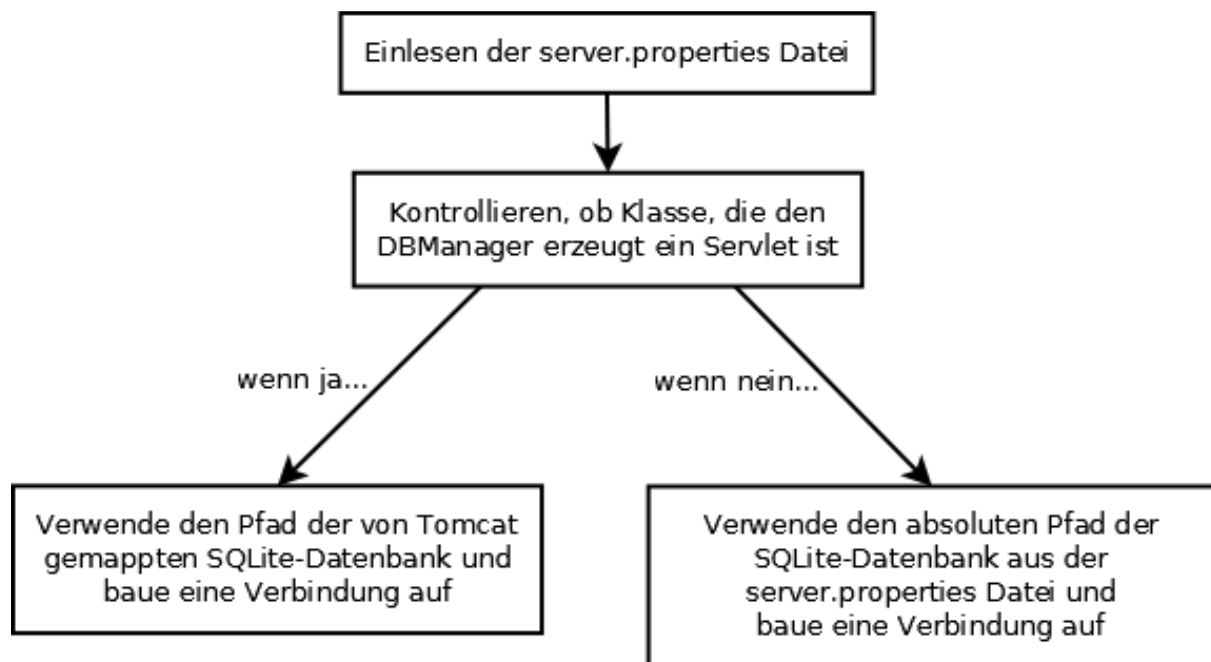


Abbildung 6: Konzept – „DBManager“

3.6.2 Konzept „Fetch“

Dieses Konzept beschreibt das Einlesen von Sensordaten wie z.B. Wasser- und Lufttemperaturdaten über die Netzwerkverbindung sowie das Ablegen der Daten in die SQLite-Datenbank.

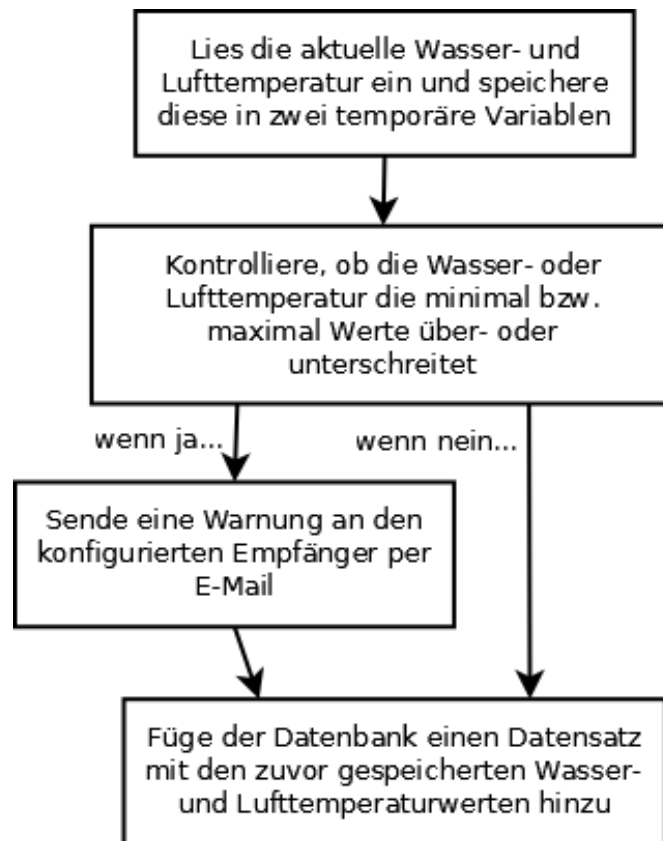


Abbildung 7: Konzept - "Fetch"

3.6.3 Konzept „Feed“

Dieses Konzept beschreibt wie der grundsätzliche Fütterungsvorgang von Statten geht.

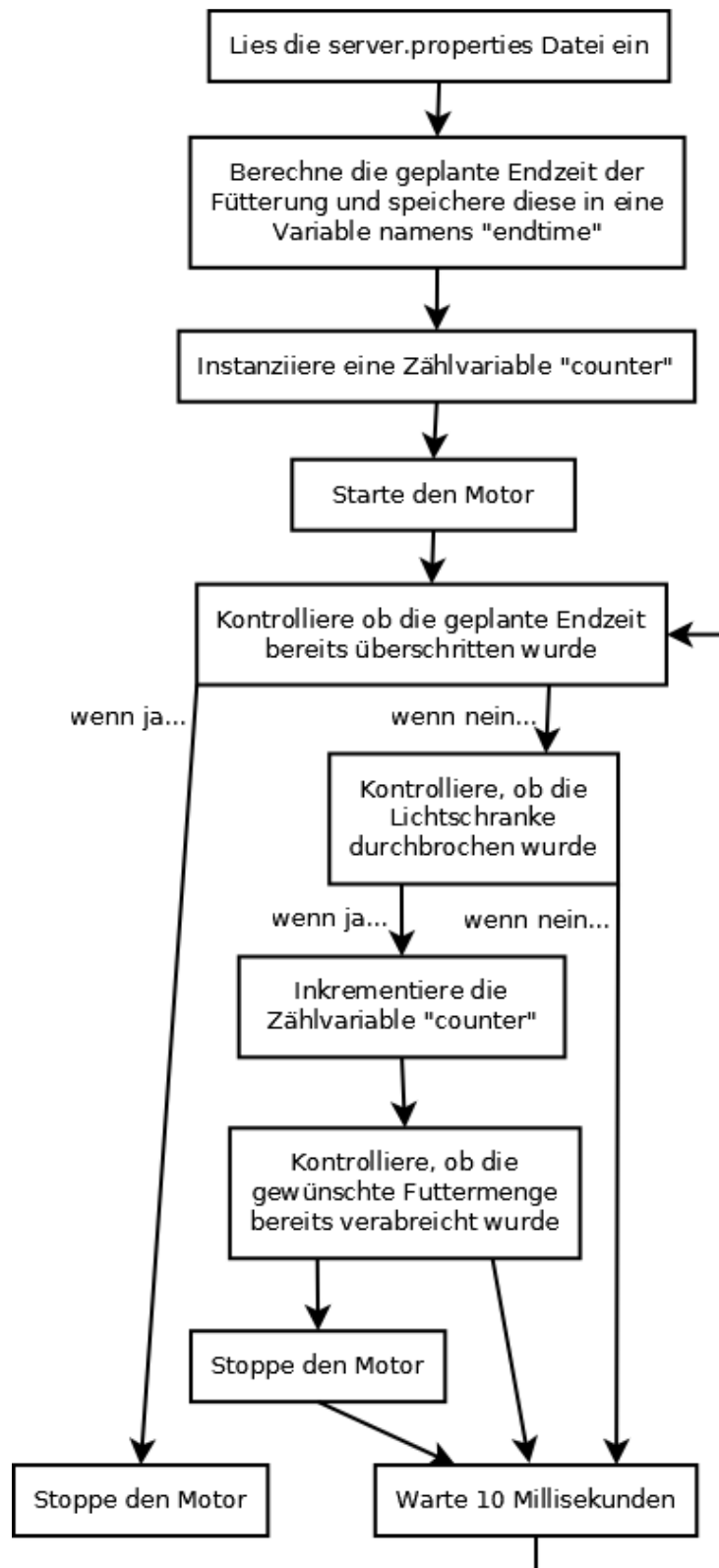


Abbildung 8: Konzept - "Feed"

3.6.4 Konzept „MainThread“

Dieses Konzept beschreibt den allgemeinen Programmablauf des Hauptprogramms.

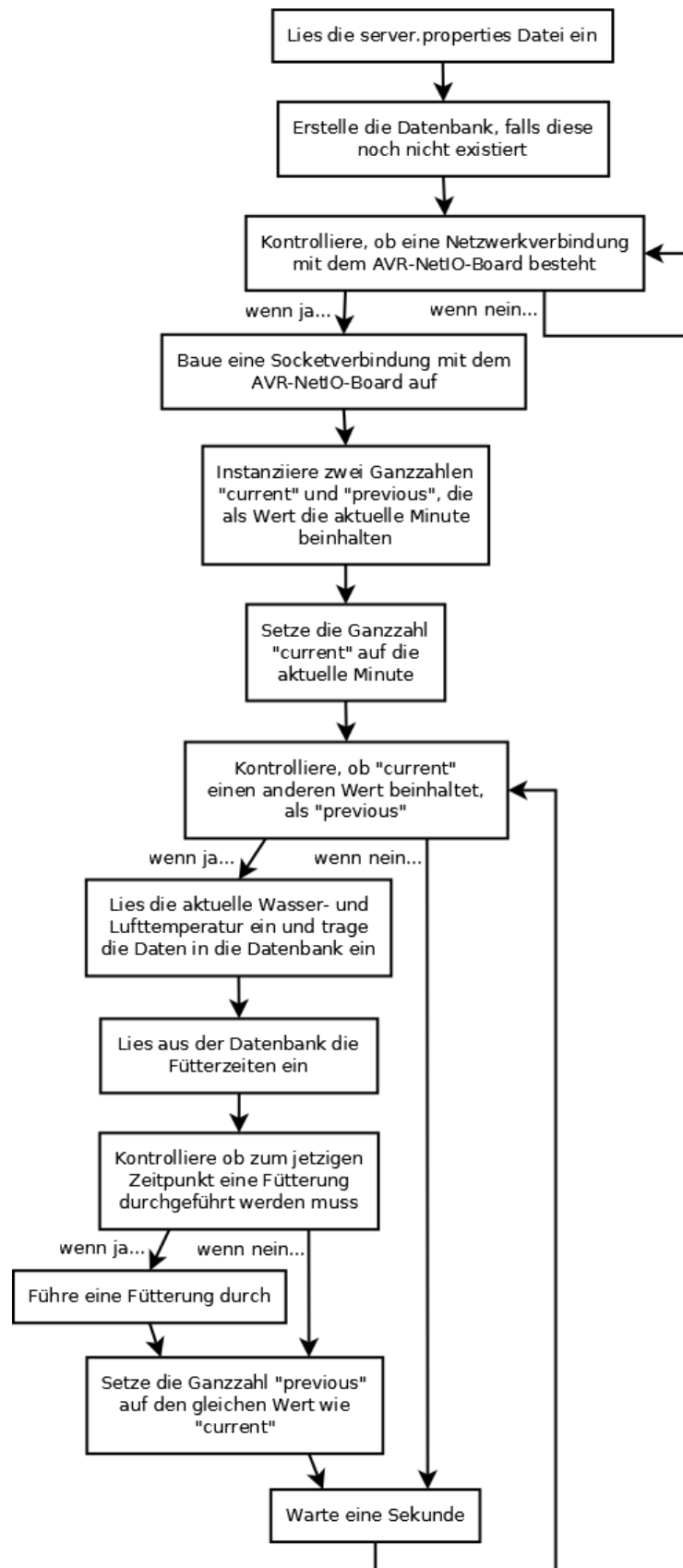


Abbildung 9: Konzept - "MainThread"

3.7 Klassen

3.7.1 Klasse Tool

Diese Klasse beinhaltet beinahe alle Funktionen, die für unsere Software benötigt werden. Die Funktionen sind alle als „static“ deklariert, sodass kein Objekt der Klasse Tool erzeugt werden muss um die Funktionen aufzurufen.

3.7.1.1 Methoden und Funktionen der Klasse Tool

Methode „write“

```
void write(String path, String[] str)
```

Schreibt das Array „str“ in eine Datei mit dem Pfad „path“.

Wird in unserem Fall für das Speichern der Konfiguration (Uhrzeiten der geplanten Fütterung) verwendet. Methode „write“ wird für den jetzigen Stand der Programmierung nicht mehr benötigt.

Funktion „read“

```
String[] read(String path)
```

Liest eine Datei mit dem Pfad „path“ ein und gibt den Inhalt als String-Array zurück.

Wird in unserem Fall für das Einlesen der Konfiguration (Uhrzeiten der geplanten Fütterung) verwendet. Methode „read“ wird für den jetzigen Stand der Programmierung nicht mehr benötigt.

Funktion „readFishConfig“

```
String[] readFishConfig()
```

Ruft die Funktion „read“ mit dem Pfad der Fischfütterungsdatei auf, falls diese überhaupt existiert und gibt den Inhalt als String-Array zurück. Funktion „readFishConfig“ wird für den jetzigen Stand der Programmierung nicht mehr benötigt.

Methode „writeFishConfig“

```
void writeFishConfig(String[] str)
```

Ruft die Methode „write“ mit dem Pfad der Fischfütterungsdatei auf, falls diese überhaupt existiert. Existiert sie nicht, wird eine neue Datei erstellt. In die Datei wird ein String-Array geschrieben. Methode „writeFishConfig“ wird für den jetzigen Stand der Programmierung nicht mehr benötigt.

Funktion „fishConfigExists“

```
boolean fishConfigExists()
```

Schaut nach, ob die Fischfütterungsdatei existiert und returniert „true“ falls diese existiert, ansonsten wird „false“ returniert. Funktion „fishConfigExists“ wird für den jetzigen Stand der Programmierung nicht mehr benötigt.

Funktion „getTemperatureFromVoltage“

```
float getTemperatureFromVoltage(float voltage)
```

Berechnet und returniert die resultierende Temperatur aus der gemessenen Spannung.

Funktion „getTemperature“

```
float getTemperature(SocketManager sman, int adc)
```

Ruft die Funktion „getVoltageFromDigital“ und „getTemperatureFromVoltage“ auf, um die Temperatur als float-Variable zu returnieren.

Funktion „round“

```
float round(float d, int decimalPlace)
```

Diese Funktion rundet die Zahl „d“ ab der Kommastelle „decimalPlace“ auf oder ab. Es handelt sich hier um algebraisches Runden. Zurückgegeben wird eine float-Variable.

Funktion „getVoltageFromDigital“

```
float getVoltageFromDigital(int i)
```

Berechnet und returniert die resultierende Spannung aus dem digitalen Wert „i“.

Methode „wait“

```
void wait(int milliseconds)
```

Wartet „milliseconds“ Millisekunden. Die Exceptions werden bereits in der Methode selbst behandelt.

Funktion „ping“

```
boolean ping()
```

Pingt das AVR-NetIO-Board und returniert „true“ falls der Ping erfolgreich war. Ansonsten wird false returniert.

Methode „sendMail“

```
void sendMail(String subject, String text)
```

Startet den „JavaMailThread“, welcher ein E-Mail mit dem Titel „subject“ und dem Inhalt „text“ an den konfigurierten Empfänger versendet.

Funktion „SgetTime“

```
String SgetTime(String format)
```

Gibt die aktuelle Zeit als String mit dem Format „format“ zurück.

Funktion „IgetTime“

```
int IgetTime(String format)
```

Gibt die aktuelle Zeit als Integer mit dem Format „format“ zurück.

Methode „fetch“

```
void fetch(SocketManager sman)
```

Baut mithilfe des SocketManagers eine Socket-Verbindung mit dem AVR-NetIO-Board auf und liest die Wasser- und Lufttemperatur ein. Ist ein Wert kritisch wird eine Warnnachricht als E-Mail versendet. Zusätzlich werden die Daten in die SQLite-Datenbank gespeichert.

Methode „feed“

```
void feed(SocketManager sman)
```

Baut mithilfe des Socket-Managers eine Socket-Verbindung mit dem AVR-NetIO-Board auf und startet den Motor und somit die Fütterung. Mit einer Zeitverzögerung von 500ms wird auch noch die Spannung an der Lichtschranke gemessen. Ändert sich diese wird der Motor gestoppt. Andernfalls hält der Motor auf jeden Fall nach 4 Sekunden.

Funktion „timeFormat“

```
String timeFormat(int i)
```

Returniert eine „korrekt“ formatierte Zeit. Beispielsweise wird diese Funktion zum Umformatieren von 9:15 in 09:15 verwendet.

Funktion „getGauge“

```
String getGauge(float temperature)
```

Ist für die grafische Darstellung der Temperatur beim Web-Interface verantwortlich. Dazu wird ein transparentes Pixel skaliert um ein Balkendiagramm zu erzeugen. Dieser ist grün wenn die Temperatur einen normalen Wert aufweist, rot wenn dieser zu heiß ist und blau wenn dieser zu kalt ist.

Funktion „md5“

```
String md5(String input)
```

Hat die Aufgabe den String „input“ mit dem MD5-Hash-Algorithmus zu hashen. Der gehashte Wert wird zurückgegeben. Da das Passwort zum Anmelden beim Web-Interface nur in gehashter Form in der Datenbank gespeichert ist, muss das eingegebene Passwort gehasht werden und kann erst dann mit dem in der Datenbank gespeicherten gehashten Passwort verglichen werden.

Funktion „StringToDate“

```
Date StringToDate(String s)
```

Wandelt einen Datums-String in ein Date-Object um und returniert dieses. Diese Funktion muss beim Einlesen des Datums aus der SQLite-Datenbank verwendet werden. Das SQLite-Datums-Objekt ist nämlich nicht mit den Java-Datums-Objekten kompatibel.

Funktion „createThermometer“

```
JFreeChart createThermometer(String s)
```

Erstellt einen JFreeChart Thermometer-Chart für die Wasser- oder Lufttemperatur und returniert diesen. Er wird dann auf der „index.jsp“ mit dem jeweiligen Servlet angezeigt.

Funktion „createGraph“

```
JFreeChart createGraph()
```

Erstellt einen JFreeChart Timeseries-Chart für die Wasser- und Lufttemperatur und returniert diesen. Er wird dann auf der „graph.jsp“ Seite mithilfe des ChartViewer-Servlets angezeigt.

3.7.2 Klasse „SocketManager“**3.7.2.1 Methoden und Funktionen der Klasse „SocketManager“****Methode „init“**

```
void init()
```

Erzeugt die Socket-Verbindung mit dem AVR-NetIO-Board und ermöglicht somit die Kommunikation.

Methode „close“

```
void close()
```

Beendet die Socket-Verbindung.

Funktion „isConnected“

```
boolean isConnected()
```

Returniert „true“ falls eine Verbindung mit dem AVR-NetIO-Board aufgebaut wurde. Andernfalls wird „false“ returniert.

Funktion „GETADC“

```
int GETADC(int adc)
```

Sendet den Befehl GETADC mit dem Parameter „adc“ an das AVR-NetIO-Board. Der empfangene String wird in einen Integer umgewandelt und returniert.

Methode „SETPORT“

```
void SETPORT(int port, int status)
```

Sendet den Befehl SETPORT mit den Parametern „port“ und „status“ an das AVR-Net-IO-Board.

Methode „schreibeNachricht“

```
void schreibeNachricht(String nachricht)
```

Schickt einen Befehl an das AVR-NetIO-Board.

Funktion „leseNachricht“

String leseNachricht()

Empfängt eine Nachricht vom AVR-NetIO-Board.

3.7.3 Klasse DBManager**3.7.3.1 Methoden und Funktionen der Klasse DBManager****Methode „createDB“**

void createDB()

Erzeugt eine neue Datenbank-Tabelle „sensordaten“, „users“ und „konfiguration“ falls diese noch nicht existieren. Weiters wird ein User „foo“ mit Passwort „bar“ erzeugt. Folgendes SQL-Statement wird verwendet:

```
create table if not exists sensordaten
(
  id integer primary key autoincrement,
  wassertemperatur float default 0,
  lufttemperatur float default 0,
  zeitpunkt timestamp default current_timestamp
);

create table if not exists users
(
  id integer primary key autoincrement,
  username varchar(8) not null,
  password varchar(32) not null
);

create table if not exists configuration
(
  id integer primary key autoincrement,
  string varchar(999) not null
);

insert into users (username, password) values ("foo", "bar");
```

Methode „recreate“

void recreate()

Löscht die Tabellen „sensordaten“, „users“ und „konfiguration“ und erzeugt diese neu.

Methode „close“

void close()

Beendet die Datenbankverbindung.

Funktion „login“

```
boolean login(String username, String password)
```

Kontrolliert, ob der eingegebene Username mit dem eingegebenen Passwort in der Datenbank existiert. Dazu muss das eingegebene Passwort vorher gehasht werden um es mit der Datenbank zu vergleichen. Ist der Username mit dem dazugehörenden Passwort in der Datenbank vorhanden wird „true“ returniert. Ansonsten wird „false“ returniert.

Methode „speichern“

```
void speichern(Daten d)
```

Speichert die Eigenschaften des Objekts „Daten“ in die Datenbank“.

Funktion „getAll“

```
List<Daten> getAll()
```

Returniert eine Liste mit allen Daten aus der Datenbank.

Funktion „getLastEntries“

```
List<Daten> getLastEntries(int count)
```

Returniert eine Liste der letzten „count“ Daten aus der der Datenbank. Wird für die Erzeugung des Temperaturgraphen benötigt.

Funktion „getConfig“

```
String[] getConfig()
```

Liest die Zeichenkette „string“ aus der Tabelle „konfiguration“ aus, erzeugt ein String-Array und returniert dieses.

Methode „setConfig“

```
void setConfig(String s[])
```

Schreibt das String-Array „s“ in Datenbank-Tabelle „konfiguration“.

3.7.4 Klasse Daten

Dieses Objekt stellt einen Datensatz für die Datenbank dar. Es beinhaltet die Eigenschaften id, wtemp, ltemp, zeitpunkt. Diese können mit Getter- und Setter-Methoden abgefragt und gesetzt werden.

3.7.5 Klasse „JavaMailThread“

Ist für das Senden von E-Mails verantwortlich. Da dies im Hintergrund passieren soll, wurde hierfür ein Thread programmiert. Dieser wird mithilfe der Methode „sendMail“ in der Klasse Tool erzeugt und gestartet. Wurde ein E-Mail gesendet wird dieser automatisch wieder gestoppt.

3.7.6 Klasse Data

Diese Klasse liest die Datei server.properties ein und speichert deren Eigenschaften. Diese Klasse wird immer dann benötigt, wenn z.B. eine Datenbank-Verbindung, eine Socket-Verbindung oder Ähnliches gestartet wird.

3.7.7 Klasse MainThread

Dieser Thread stellt das Hauptprogramm dar. Wenn der Thread gestartet wird, wird vorerst sichergestellt, dass überhaupt eine Verbindung mit dem AVR-NetIO-Board hergestellt wurde. Dafür wird die Methode „ping“ gestartet. Danach wird gewartet bis eine neue Minute beginnt. Ist dies der Fall wird die Methode „fetch“ aufgerufen und die Sensor-Daten in die Datenbank gespeichert. Falls in der Fischfütterungsdatei, die aktuelle Uhrzeit als geplante Fütterungszeit gespeichert ist, wird zusätzlich die Methode „feed“ aufgerufen und die Fische werden gefüttert.

3.7.8 Klasse Launcher

Diese Klasse ist für das Starten des „MainThread“ verantwortlich. Der „MainThread“ wird gestoppt durch Eingabe eines beliebigen Zeichens oder Worts in die Konsole.

3.8 Servlets

Servlets sind Java-Klassen, die auf Webservern liegen und Anfragen von Clients entgegennehmen und beantworten. Hauptsächlich werden sie dazu verwendet Session-Daten auszulesen und weiterzuverarbeiten. Für den grafischen Aufbau verwendet man eher JavaServer-Pages. Deshalb wird auch nach dem Entwurfsmuster „Model-View-Controller“ programmiert, wobei sich die JavaServer-Pages im „View-“ und die Servlets sich im „Controller-Package“ befinden.

3.8.1 Das Konfigurations-Servlet

Ein tolles Feature unseres Web-Interfaces ist der Konfigurationsmanager. Eine Kombination der Vorteile von Servlets, Java Server Pages und AJAX-Scripts bietet optimalen Komfort beim Konfigurieren der Fischfütterungszeiten.

3.8.2 Das Datenbank-Servlet

Der „Datenbankviewer“ unseres Webinterfaces beinhaltet neben der Funktion zum Anzeigen von Sensorwerten aus einer SQLite-Datenbank eine innovative Methode zur grafischen Darstellung von Temperaturdaten. Dieses Feature wurde mithilfe eines transparenten Pixels und einer Farbenskala (blau = zu kalt, grün = normal, rot = zu heiß) realisiert. Unterschreitet beispielsweise die gemessene Temperatur einen Temperaturmindestwert wird die grafische Darstellung für diesen Datensatz in blauer Farbe durchgeführt.

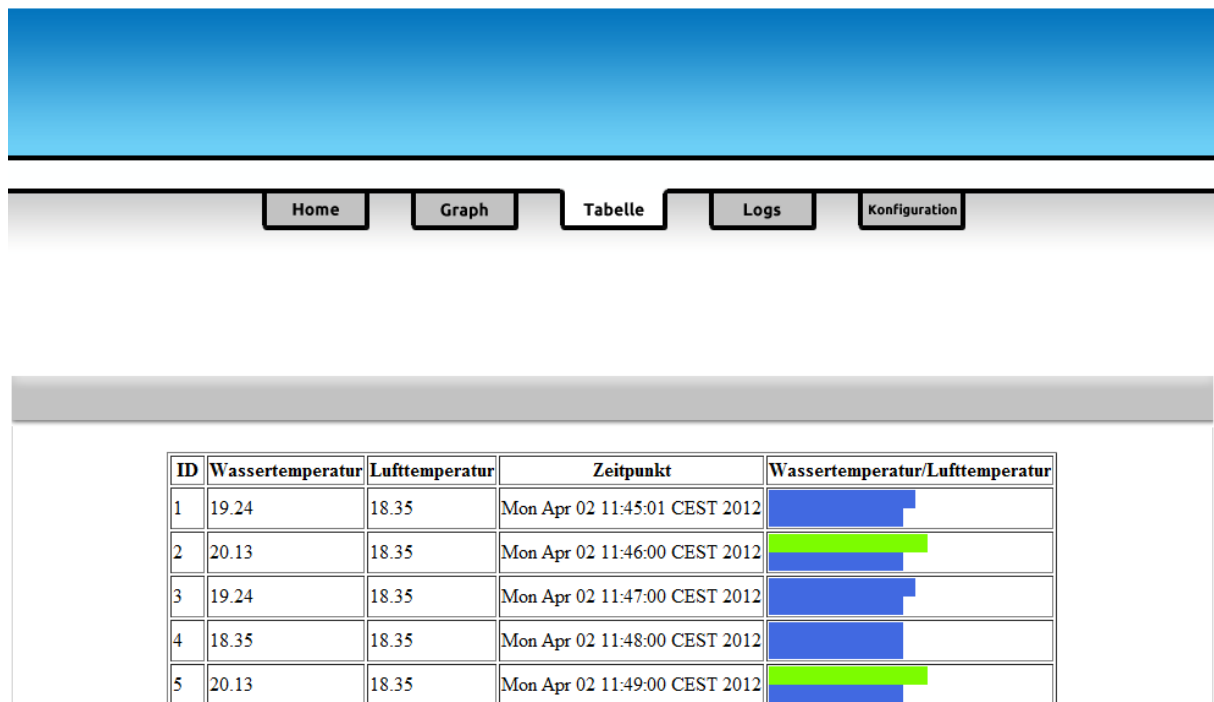


Abbildung 10: Datenbank-Servlet

3.8.3 Das Anmelde-Servlet

Aufgabe dieses Servlets ist es, den generellen Zugriff auf das Web-Interface zu gewähren. Dazu werden die Userdaten, die beim „login.jsp“ eingegeben wurden mit der SQLite-User-Datenbank verglichen. Das eingegebene Passwort muss jedoch zuerst mit einem MD5-Hash gehasht werden um eine Abfrage zu ermöglichen.

3.8.4 Das Setup-Servlet

Dieses Servlet wird aufgerufen, wenn der Benutzer sein Passwort vergessen hat, oder wenn der Benutzer seinen Usernamen/sein Passwort ändern will.

3.8.5 Das Abmelde-Servlet

Dieses Servlet erfüllt lediglich die Aufgabe, den Zugang auf das Web-Interface wieder zu sperren und die Session zurückzusetzen.

3.8.6 Das ChartViewer-Servlet

Dieses Servlet ist für die Darstellung des Temperaturgraphen auf der „graph.jsp“ Seite verantwortlich. Hierbei wird die Grafik direkt auf den Output-Stream geschrieben.

3.8.7 Das LuftThermo-Servlet

Dieses Servlet ist für die Darstellung der Luftthermometer-Grafik auf der „index.jsp“ Seite verantwortlich. Hierbei wird die Grafik direkt auf den Output-Stream geschrieben.

3.8.8 Das WasserThermo-Servlet

Dieses Servlet ist für die Darstellung der Wasserthermometer-Grafik auf der „index.jsp“ Seite verantwortlich. Hierbei wird die Grafik direkt auf den Output-Stream geschrieben.

3.9 JavaServer Pages

Mit JavaServer Pages (JSP) ist es möglich, dynamischen Webinhalt in einer Kombination aus HTML und der Programmiersprache Java zu erstellen. Da HTML eine Markup-Language ist und somit nicht in der Lage ist Variablen zu deklarieren geschweige denn zu nutzen, stellen für uns JavaServer Pages die optimale Alternative zu HTML dar. JavaServer Pages erlauben es Java-Code in ein HTML-Dokument zu implementieren. Dabei ist es jedoch nicht empfehlenswert den ganzen Java Code in ein JSP zu verlegen. Das führt zu Geschwindigkeitsverlusten und zu Unübersichtlichkeit.

3.10 Apache Tomcat Server

Der Apache Tomcat Server ist ein Open-Source-Software-Projekt der Apache Software Foundation. Dieser Webserver ist in der Lage Java Servlets und JavaServer Pages zu verwalten. Dabei verfügt der Tomcat Server über einen JSP-Compiler namens Jasper. Dieser übersetzt JavaServer Pages in Servlets und bringt diese zur Ausführung. Die aktuellste Version des Apache Tomcat Servers ist 7.0.25.³



Abbildung 11: Apache-Logo

Mit der Java-Entwicklungsumgebung ist das Arbeiten mit Apache Tomcat besonders komfortabel. Ohne Entwicklungsumgebung wäre es relativ umständlich ein Servlet am Apache Tomcat zu registrieren und lauffähig zu machen. Mit Eclipse ist es möglich einen Server ähnlich wie eine gewöhnliche Klasse zu erstellen. Mit Rechtsklick → New → Other → Server wird ein neuer Server erstellt. Im darauffolgenden Konfigurationsdialog kann man sich für eine gewünschte Version des Apache Tomcat Servers entscheiden. In unserem Fall war das die Version 7.

Ist ein Server erstellt kann dem Server ein „Dynamic Web Project“ hinzugefügt werden. Wenn man anschließend den Server startet ist wird gleichzeitig die Webanwendung am Server deponiert und lauffähig gemacht.

Soll die Webanwendung irgendwann zu produktiven Zwecken genutzt werden ist es mit Eclipse möglich das Projekt in eine WAR (Web Archive)-Datei zu exportieren. Diese Datei kann dann im Apache Tomcat Verzeichnis unter Webapps deponiert werden.

³ (Apache Tomcat)

Um einen Apache Tomcat Server manuell zu starten muss man im bin-Verzeichnis des Tomcats die Batch-Datei startup.bat starten. Dazu muss jedoch im Vorhinein bereits der Java-Pfad als Systemvariable deklariert worden sein. Zum Stoppen des Tomcat Servers ist lediglich die Batch-Datei shutdown.bat auszuführen.

Beim Starten des Tomcats sowie zu seiner Laufzeit wird jede Änderung einer Datei im Verzeichnis Webapps erkannt und nötige Schritte durchgeführt. Wird beispielsweise ein Web Archiv im Verzeichnis Webapps deponiert entpackt der Tomcat Server dieses Archiv und bereitet die Webapplikation für die Ausführung vor.

```
Information: validateJarFile(D:\Deployment\apache-tomcat-7.0.25\webapps\fish\WEB-INF\lib\servlet-api.jar) - jar not loaded. See Servlet Spec 2.3, section 9.7.2.
Offending class: javax/servlet/Servlet.class
MΣr 05, 2012 9:08:48 PM org.apache.catalina.startup.HostConfig deployDirectory
Information: Deploying web application directory D:\Deployment\apache-tomcat-7.0.25\webapps\docs
MΣr 05, 2012 9:08:48 PM org.apache.catalina.startup.HostConfig deployDirectory
Information: Deploying web application directory D:\Deployment\apache-tomcat-7.0.25\webapps\examples
MΣr 05, 2012 9:08:49 PM org.apache.catalina.startup.HostConfig deployDirectory
Information: Deploying web application directory D:\Deployment\apache-tomcat-7.0.25\webapps\host-manager
MΣr 05, 2012 9:08:49 PM org.apache.catalina.startup.HostConfig deployDirectory
Information: Deploying web application directory D:\Deployment\apache-tomcat-7.0.25\webapps\manager
MΣr 05, 2012 9:08:49 PM org.apache.catalina.startup.HostConfig deployDirectory
Information: Deploying web application directory D:\Deployment\apache-tomcat-7.0.25\webapps\ROOT
MΣr 05, 2012 9:08:49 PM org.apache.coyote.AbstractProtocol start
Information: Starting ProtocolHandler ["http-bio-8080"]
MΣr 05, 2012 9:08:49 PM org.apache.coyote.AbstractProtocol start
Information: Starting ProtocolHandler ["ajp-bio-8009"]
MΣr 05, 2012 9:08:49 PM org.apache.catalina.startup.Catalina start
Information: Server startup in 613 ms
```

Abbildung 12: Apache-Kommandozeile

4 Eclipse IDE

4.1 Was ist Eclipse?

Eclipse wurde grundsätzlich von IBM entwickelt. Der Quellcode wurde jedoch von IBM Ende 2001 freigegeben und seitdem, von einer Open-Source-Gemeinschaft weiterentwickelt und in verschiedenen Bereichen verwendet, wie zum Beispiel Java und Android Anwendungen. Eclipse ist den meisten Leuten als eine IDE(integrated development environment) bekannt, eine integrierte Entwicklungsumgebung für die Programmiersprache Java. Bis heute hat Eclipse einen Marktanteil von ca. zwei Drittel erreicht. Eclipse ist somit die führende Entwicklungsumgebung im Bereich Java.



Abbildung 13: Eclipse-Logo

Das Eclipse Projekt wird von der Eclipse Foundation geleitet und betreut. Die Eclipse Foundation ist ein gemeinnütziges Unternehmen, das vor allem die Open-Source-Gesellschaft und miteinander komplementäre Produkte fördern will. Die Eclipse IDE kann mit vielen Zusatzfunktionen(Plugins) ausgestattet werden. Andere Unternehmen und Gemeinschaften haben Eclipse bereits mit vielen Komponenten aufgerüstet.⁴

4.2 Verwendung

4.2.1 Java

Eclipse benötigt mindestens Java 6 (oder 1.6). Es gibt 2 Arten von Java Installationen, die Java Runtime Environment (JRE) und das Java Development Kit (JDK). JRE beinhaltet nur das, was man benötigt, um Java Programme auszuführen, wohingegen das JDK zusätzliche Inhalte zum Entwickeln enthält, wie zum Beispiel den Compiler. Falls Java bereits installiert ist, kann man die installierte Version mit dem Kommandozeilenbefehl `java -version` in der Windows-Shell überprüfen.

4.2.2 Eclipse installieren

Um Eclipse zu installieren, besucht man die Homepageabteilung unter ⁽¹⁾ und downloadet das Paket "Eclipse IDE for Java Developers". Diese zip-Datei wird nun auf der Festplatte entpackt und kann auch schon verwendet werden.

⁽¹⁾ <http://www.eclipse.org/downloads/>

⁴ (Eclipse (IDE))

4.2.3 Eclipse starten

Das Verzeichnis, das vorher entpackt wurde, enthält eine eclipse.exe(unter Windows) , die Eclipse startet. Nach dem Öffnen wird man aufgefordert einen Pfad für das Arbeitsverzeichnis anzugeben. Das Arbeitsverzeichnis ist das Verzeichnis in das die Javaprojekte gespeichert werden.

Jetzt wird Eclipse gestartet und der Willkommensbildschirm angezeigt. Dieser kann mit dem X, unter dem Menübalken, weg geklickt werden.

4.2.4 Arbeitsverzeichnis(Workspace)

Das Arbeitsverzeichnis ist der Pfad auf dem Rechner, in dem gearbeitet wird. Wie bereits beschrieben, wird bereits beim Start von Eclipse nach dem Arbeitsverzeichnis verlangt, kann aber auch durch das Menü verändert werden(Unter File → Switch Workspace → Others). Alle Projekte und die dazugehörigen Dateien werden in dieses Verzeichnis gelegt.

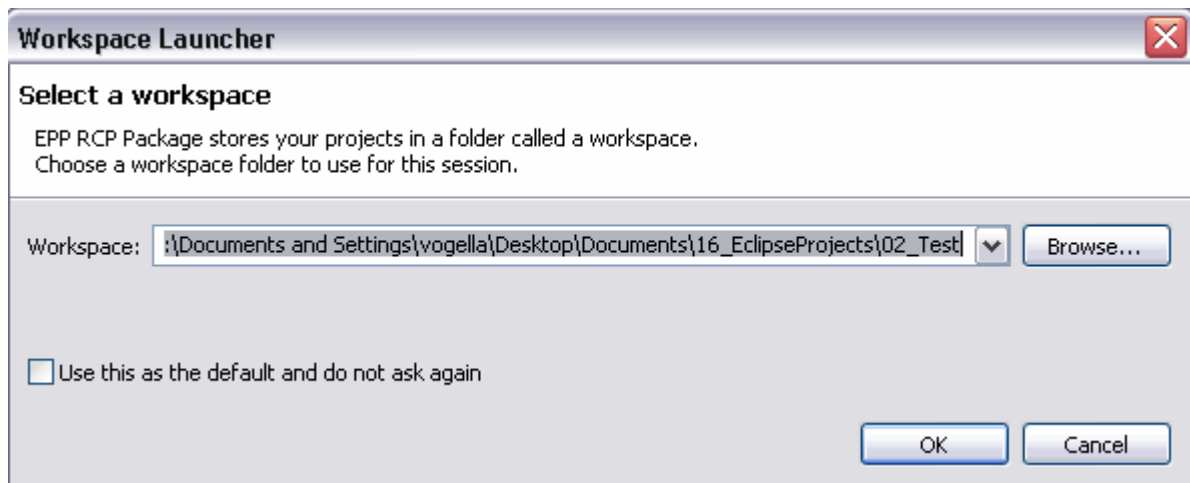


Abbildung 14: Setzen des Arbeitsverzeichnisses

4.2.5 Benutzeroberfläche

Die Eclipse Benutzeroberfläche stellt Betrachter und Editoren zur Verfügung. Diese wiederum formen zusammen Perspektiven, zwischen denen man schnell hin und her schalten kann. Diese Perspektiven setzen sich aus "Parts" zusammen, die in der Größe angepasst und auch verschoben werden können. Diese "Parts" können dann entweder ein Betrachter oder Editor sein. Es gibt vordefinierte Perspektiven, die bestimmte Parts beinhalten, aber es besteht auch die Möglichkeit, benutzerdefinierte Perspektiven zu erstellen.

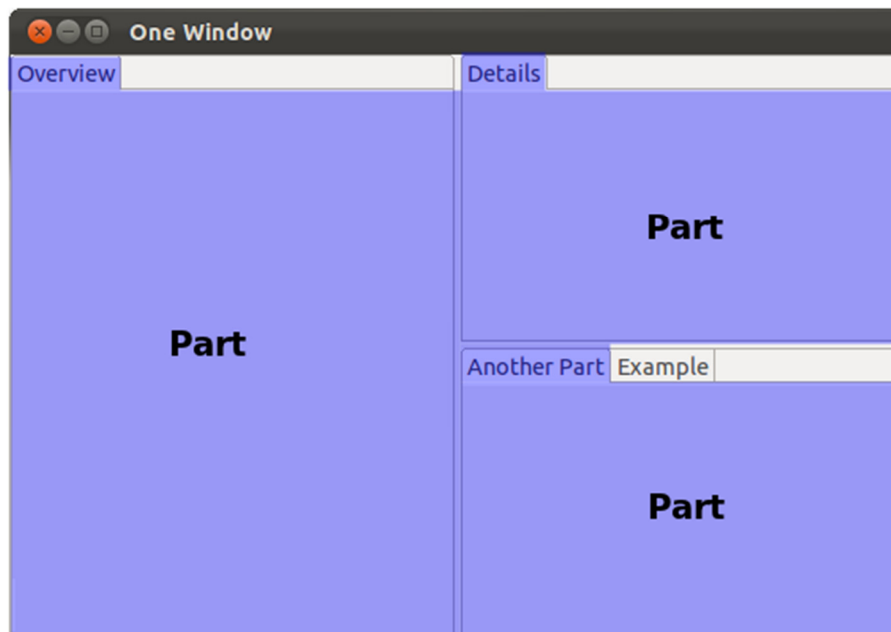


Abbildung 15: Benutzeroberfläche

In Abbildung 16 sieht man die "Java"-Perspektive, die mit der Eclipse IDE standardmäßig dabei ist. Hier kann man rechts einen Betrachter sehen, nämlich den "Java Package Explorer", mit dem man durch die vorhandenen Javaprojekte stöbern kann. Falls im "Java Package Explorer" eine Datei geöffnet wird, wird diese im Editor in der Mitte geöffnet. Dieser ist ein gewöhnlicher Texteditor und wird auch so verwendet. Unten ist auch noch ein Betrachter für verschiedene Sachen wie zum Beispiel Probleme oder Server, die mit den Reitern gewechselt werden können.

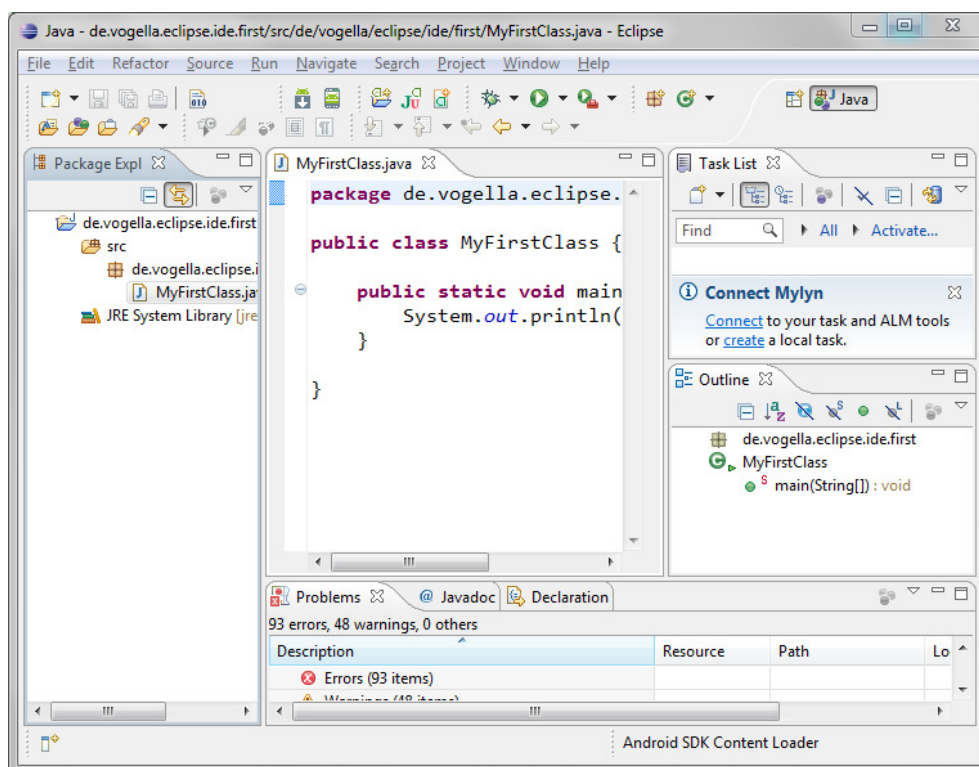


Abbildung 16: Java-Perspektive

5 MySQL

MySQL ist eines der weltweitverbreitetsten relationalen Datenbanksysteme. Es ist für private Nutzung vollkommen kostenfrei und bildet die Grundlage für viele dynamische Webauftritte. MySQL funktioniert auf beinahe allen Betriebssystemen.



Abbildung 17: MySQL-Logo

5.1 Verwendung mit JAVA

Um MySQL mit JAVA anzusprechen, muss ein sogenannter „MySQL-Connector“ verwendet werden. Dieser kann auf der MySQL-Homepage (www.mysql.de) heruntergeladen werden. Ist der Download erst einmal beendet, muss man den „MySQL-Connector“ dem „Java-Build-Path“ hinzufügen. Mithilfe der Programmierungsumgebung „Eclipse“ stellt diese Aufgabe kein Problem dar.

5.2 MySQL-Statements

Die am häufigsten verwendeten Statements sind CREATE- und INSERT-Statements.

5.2.1 CREATE-Statement

Wird hauptsächlich zum Erzeugen von Datenbank-Tabellen verwendet.

Beispiel:

```
create table if not exists users
(
  id integer primary key autoincrement,
  username varchar(8) not null,
  password varchar(32) not null
);
```

5.2.2 INSERT-Statement

Wird zum Einfügen von Daten in eine Datenbank-Tabelle verwendet.

Beispiel:

```
insert into users (username, password) values ("foo", "bar");
```

6 SQLite

SQLite ist eines der bekanntesten und weitverbreitetsten Datenbanksysteme weltweit. Es handelt sich hierbei um keine Client-/Serveranwendung wie z.B. MySQL es handelt sich vielmehr um eine Programmbibliothek, die direkt in direkt



Abbildung 18: SQLite-Logo

in eigene Anwendungen integriert werden kann. Die gesamte Datenbank befindet sich in einer einzigen Datei. Genau aus diesem Grund haben wir uns in unserem Projekt für die SQLite-Datenbank entschieden. Außerdem ist SQLite für private Nutzung vollkommen kostenfrei.

6.1 Verwendung mit JAVA

Zur Verwendung mit JAVA wird ein sogenannter „SQLite-Connector“ benötigt. Dieser ist unter dem Namen „SQLite JDBC“ im Internet (<http://www.zentus.com/sqlitejdbc/>) erhältlich. Ist der Download erst einmal beendet, muss man den „SQLite-Connector“ dem „Java-Build-Path“ hinzufügen. Mithilfe der Programmierumgebung „Eclipse“ stellt diese Aufgabe kein Problem dar.

6.2 SQLite-Statements

Sind ähnlich bzw. beinahe gleich wie bei MySQL.

Ausnahmen anhand folgender Beispiele:

SQLite: autoincrement MySQL: auto_increment

SQLite: current_timestamp MySQL: now

7 HTML

7.1 Was ist HTML?

HTML ist eine textbasierte Skriptsprache die zur Strukturierung von Inhalten wie Texten, Bildern und Hyperlinks verwendet wird. HTML-Dokumente sind die Grundlage des World Wide Web und werden von einem Webbrowser dargestellt.

7.2 Webinterface

7.2.1 Warum?

Das Webinterface ermöglicht einen komfortablen Zugriff auf Informations- und Konfigurationsmöglichkeiten für den Endbenutzer. Es ist keine Installation von zusätzlicher Software nötig. Es ist nicht einmal ein Computer nötig, um die Informations- bzw. Konfigurationsmöglichkeiten zu nutzen. Bereits ein interfähiges Handy ist dazu in der Lage. Die grafische Darbietung des Webinterfaces bietet Komfort und verringert das Fehlerrisiko des Endbenutzers.

7.2.2 Design

Das Design ist schlicht und modern. Die Menüführung ist leicht zu verstehen. Zusätzliche AJAX-Elemente fördern das Weberlebnis. Temperaturkurven erleichtern es dem Endbenutzer die Temperaturen der letzten Tage mit zu verfolgen. Farbliche Darstellungen von Temperaturwerten lassen kritische Temperaturwerte leicht erkennen.

7.2.3 Features

Farbige Balken zur grafischen Anzeige von Temperaturen

Dieses Feature erlaubt es Temperaturwerte zusätzlich je nach Höhe der Temperatur grafisch zu markieren. Rote bzw. blaue Werte deuten auf einen kritischen Temperwert (zu heiß bzw. zu kalt) hin. Grüne Werte symbolisieren eine perfekte Temperatur.

Thermometerbild zur Anzeige der aktuellen Temperatur

Dieses Feature erlaubt es, die aktuelle Wasser- und Lufttemperatur grafisch anzuzeigen. Ein Thermometer zeigt grün für normale Werte, orange für gefährliche Werte und rot für kritische Werte an.

Temperaturgraph zur grafischen Darstellung der Temperaturwerte aus der Datenbank

Der Temperaturgraph zeigt die Temperaturwerte der letzten Stunde grafisch an. Dazu werden die letzten 60 Wasser- bzw. Lufttemperaturwerte die minütlich gemessen wurden aus der Datenbank entnommen und mit JFreeCharts eine Grafik erzeugt.

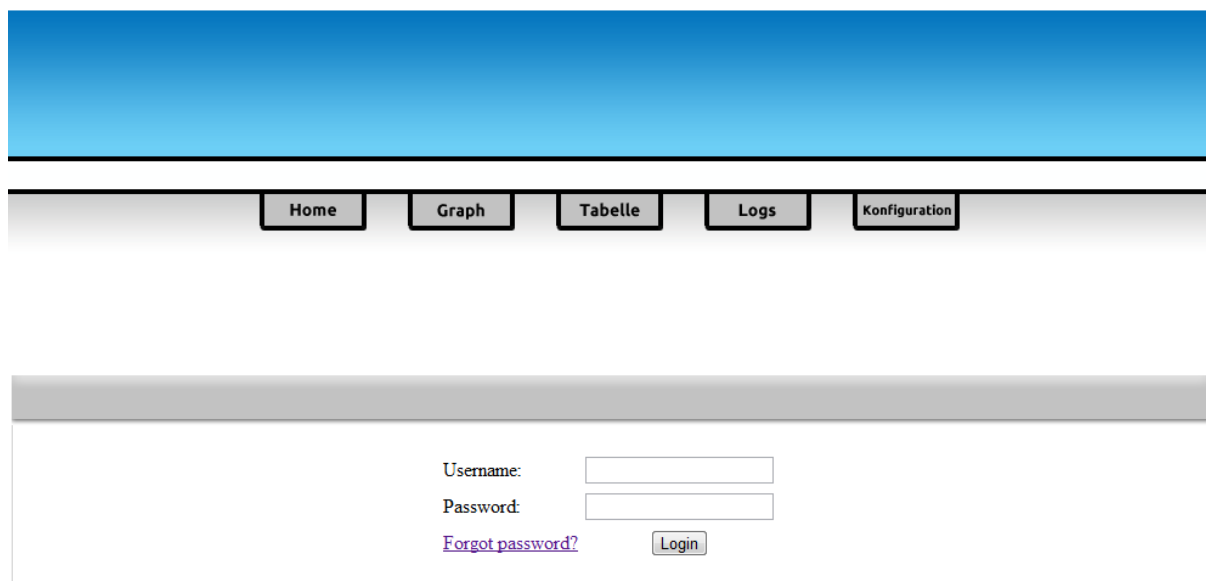
Benutzermanagement

Um das Webinterface verwenden zu können ist es nötig sich zu Authentifizieren. Dazu ist die Eingabe eines Usernamens und eines Passworts nötig. Die Userdaten sind in einer Datenbank gespeichert. Das Passwort ist natürlich gehasht.

7.2.4 Webinterface Screenshots

Login

Hier findet das Authentifizieren statt. Ist man nicht authentifiziert, hat man keinen Zugriff auf die verschiedenen Elemente des Webinterfaces. Nach erfolgreichem Anmelden wird eine Boolean-Variable in Session gespeichert. Nach 15 Minuten wird man dann automatisch abgemeldet.

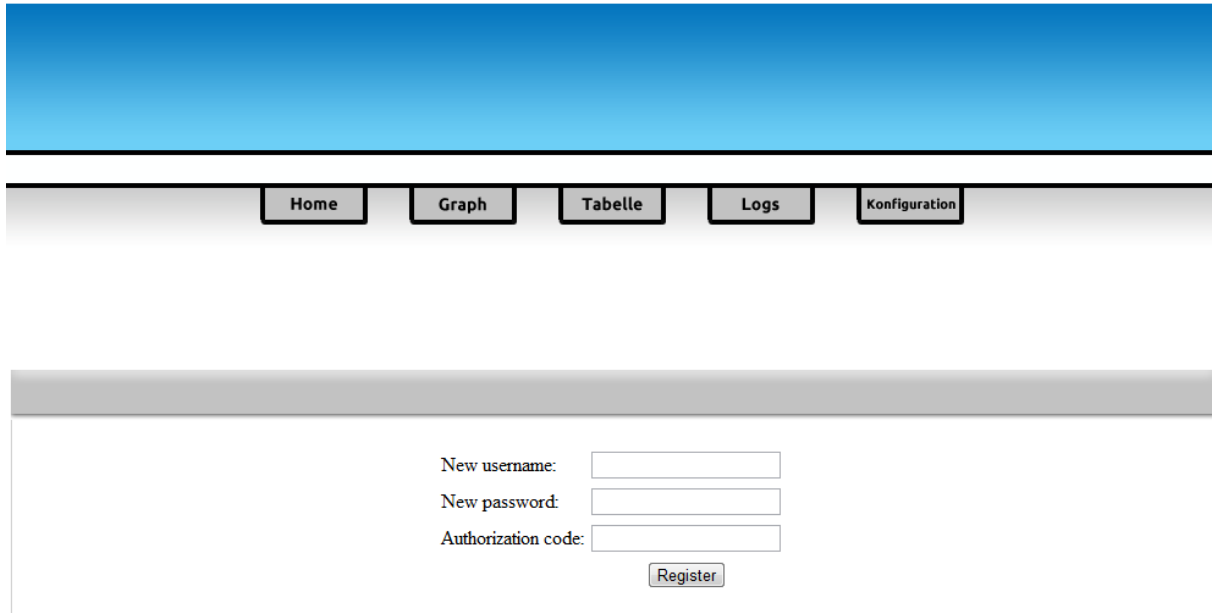


The screenshot shows the login page of the web interface. At the top, there is a blue gradient header. Below it is a navigation bar with five buttons: 'Home', 'Graph', 'Tabelle', 'Logs', and 'Konfiguration'. The main content area is white and contains the login form. The form has two input fields: 'Username:' and 'Password:'. Below the 'Password:' field is a link labeled 'Forgot password?' and a 'Login' button.

Abbildung 19: Login

Setup

Hat man einmal sein Passwort vergessen ist dies keine Tragödie. Mithilfe eines „Setups“ kann man sein Passwort ohne Schwierigkeiten wieder zurücksetzen. Dazu ist jedoch ein „Authorization Code“ nötig. Dieser kann unter ⁽¹⁾ begutachtet werden.



The screenshot shows the 'Setup' page of the automatic fish feeding system. At the top, there is a blue gradient header. Below it is a navigation bar with five buttons: 'Home', 'Graph', 'Tabelle', 'Logs', and 'Konfiguration'. The 'Konfiguration' button is highlighted. The main content area has a light gray background and contains three input fields labeled 'New username:', 'New password:', and 'Authorization code:'. Below these fields is a 'Register' button.

Abbildung 20: Setup

⁽¹⁾ mpLiBGsDwwZS8ntAsiWg6Zmm3WF6TNDGCXyMmrYjn8Cuu55nmaUsAGeCmslFGUV

Home-Tab

Hier sieht man, ob das AVR-NetIO-Board verbunden ist, oder nicht. Ein grünes Lämpchen bedeutet, dass möglich ist das AVR-NetIO-Board zu pingen. Ein rotes Lämpchen bedeutet, dass das AVR-NetIO-Board wahrscheinlich nicht mit dem Netzwerk verbunden ist. Des Weiteren kann man die aktuelle Wasser- und Lufttemperatur grafisch ablesen. Dieses Feature wurde mithilfe von JFreeCharts („ThermometerChart“) verwirklicht.

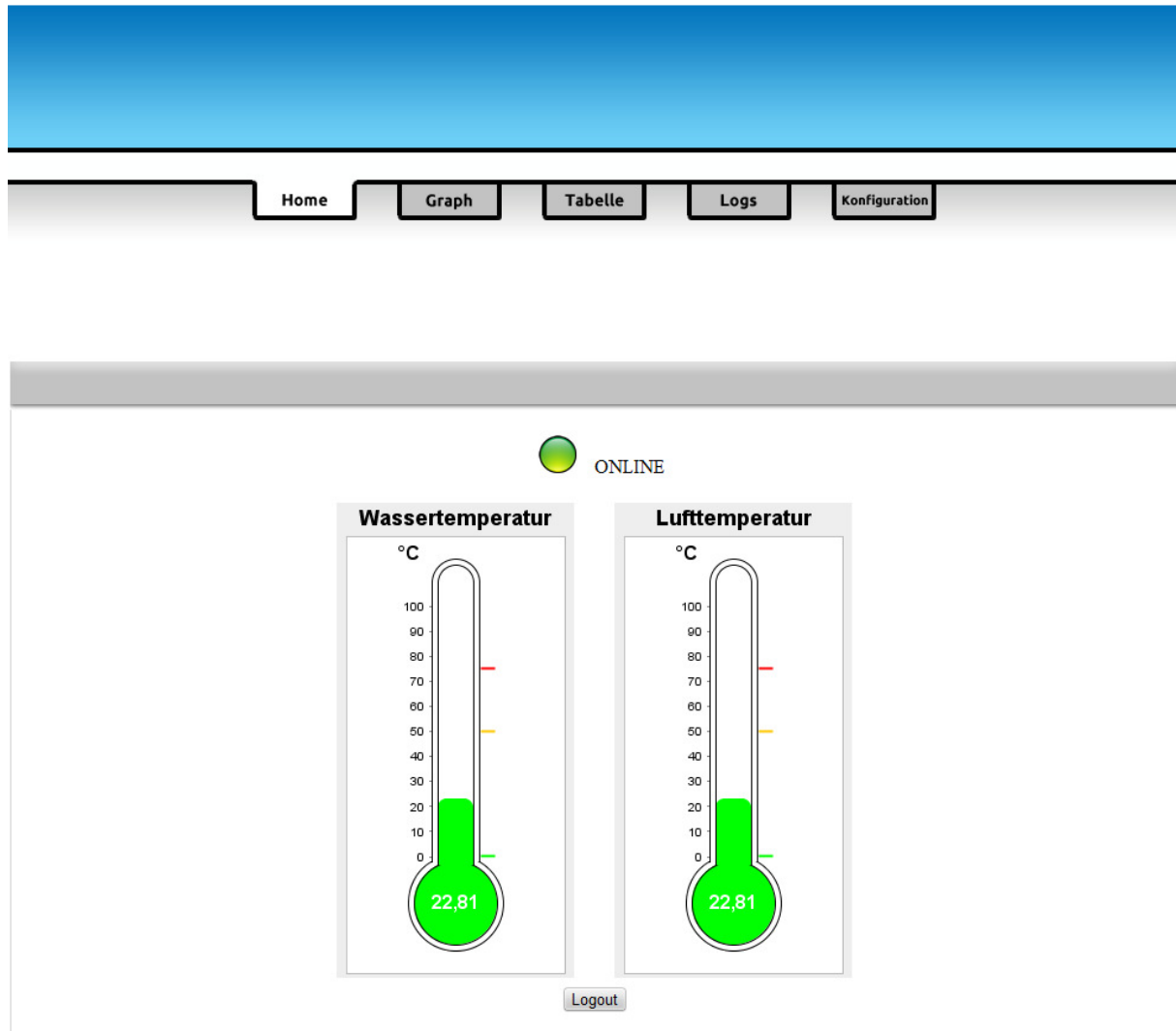


Abbildung 21: Home-Tab

Graph-Tab

Hier werden die Wasser- und Lufttemperaturwerte mithilfe eines Graphen versinnbildlicht. Die Wassertemperaturkurve wird rot dargestellt. Die Lufttemperaturkurve wird blau dargestellt. Die X-Achse zeigt den Zeitverlauf an, wohingegen die Y-Achse den Temperaturverlauf anzeigt. Der Graph beinhaltet alle Werte innerhalb einer Stunde. Fehlt ein Wert, wird interpoliert.

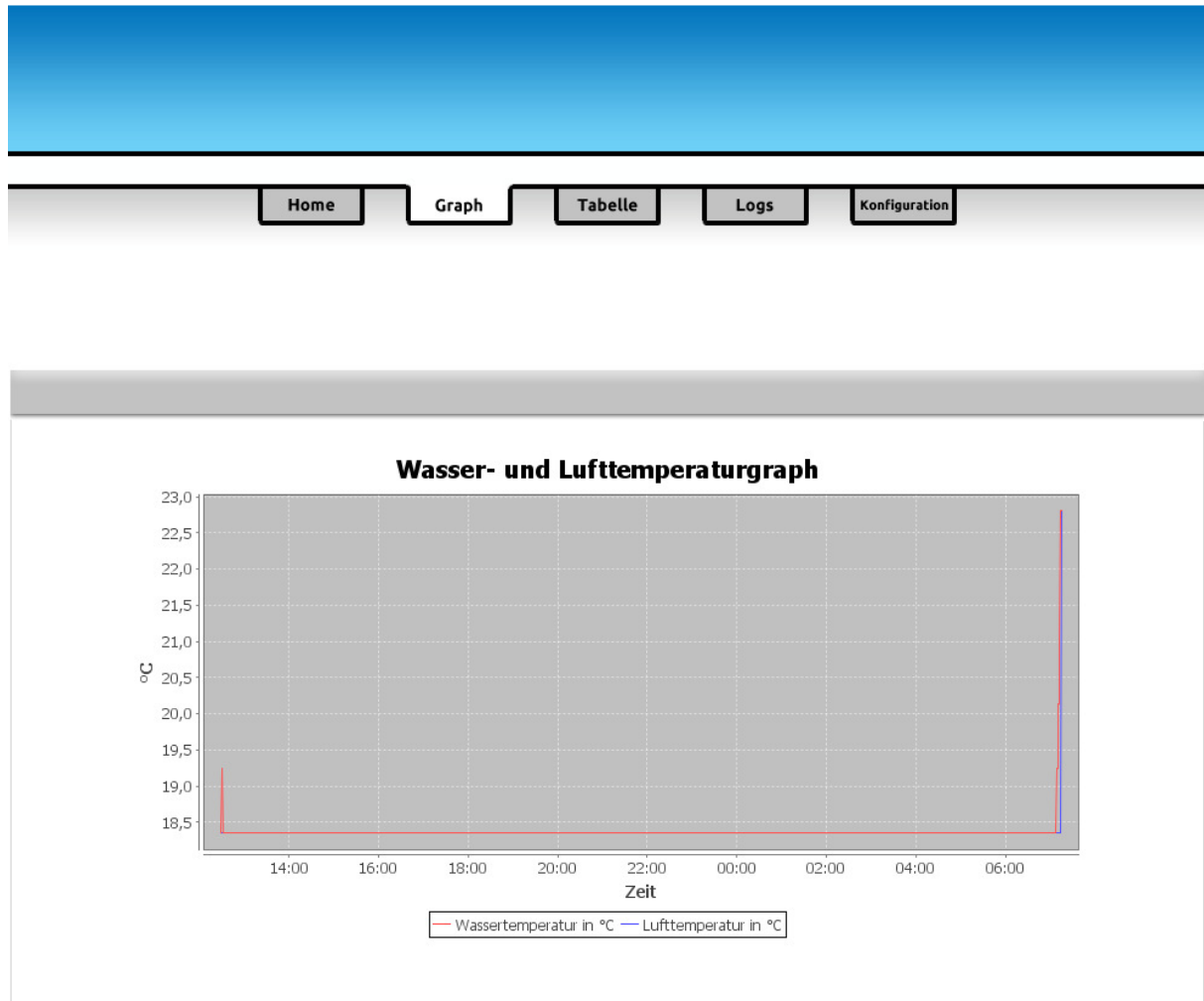
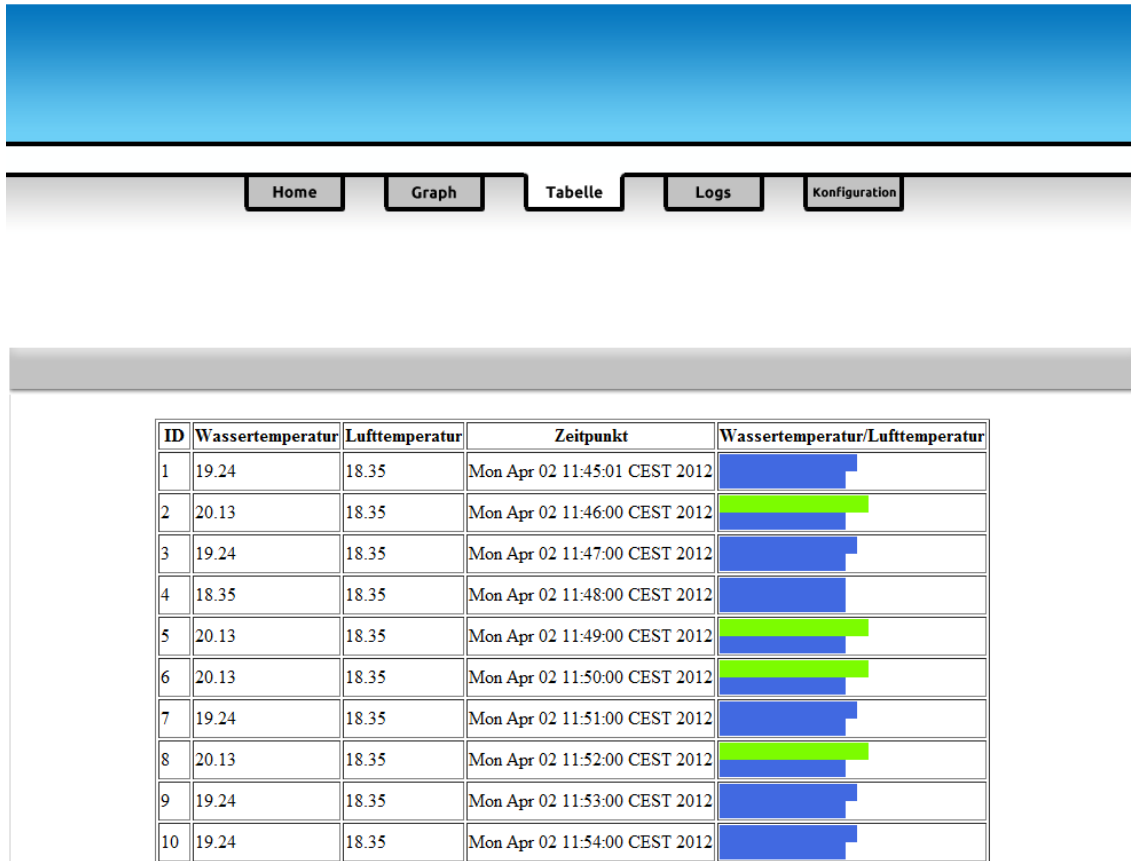


Abbildung 22: Graph-Tab

Tabelle-Tab

Hier wird die gesamte Datenbank tabellarisch angezeigt. Außerdem werden der Wasser- und Lufttemperaturwert mit farbigen Balken zur schnelleren Kontrolle dargestellt.

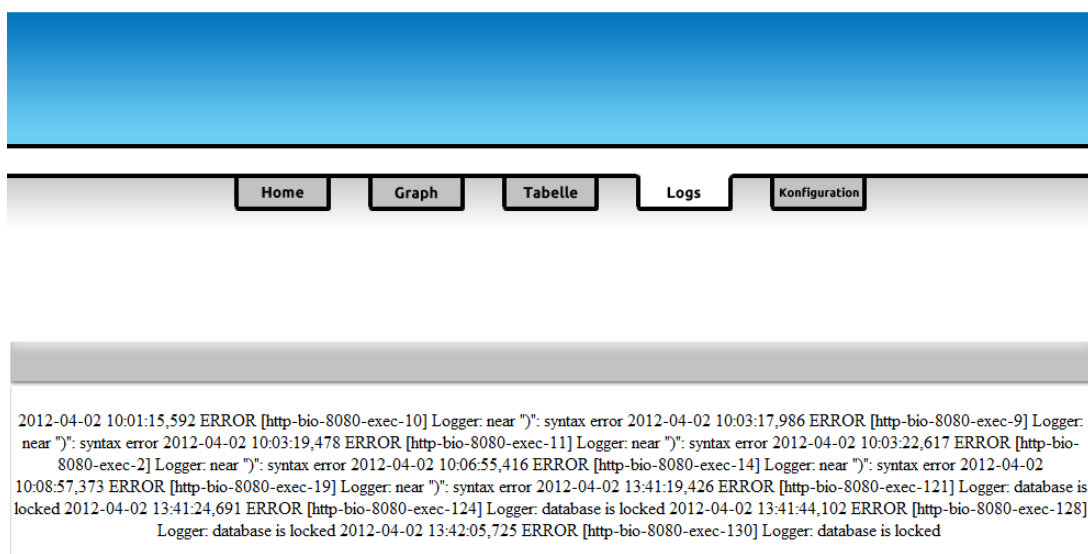


ID	Wassertemperatur	Lufttemperatur	Zeitpunkt	Wassertemperatur/Lufttemperatur
1	19.24	18.35	Mon Apr 02 11:45:01 CEST 2012	
2	20.13	18.35	Mon Apr 02 11:46:00 CEST 2012	
3	19.24	18.35	Mon Apr 02 11:47:00 CEST 2012	
4	18.35	18.35	Mon Apr 02 11:48:00 CEST 2012	
5	20.13	18.35	Mon Apr 02 11:49:00 CEST 2012	
6	20.13	18.35	Mon Apr 02 11:50:00 CEST 2012	
7	19.24	18.35	Mon Apr 02 11:51:00 CEST 2012	
8	20.13	18.35	Mon Apr 02 11:52:00 CEST 2012	
9	19.24	18.35	Mon Apr 02 11:53:00 CEST 2012	
10	19.24	18.35	Mon Apr 02 11:54:00 CEST 2012	

Abbildung 23: Tabelle-Tab

Logs-Tab

Falls Fehler auftreten, können diese im Webinterface angezeigt werden.



2012-04-02 10:01:15,592 ERROR [http-bio-8080-exec-10] Logger: near ")": syntax error 2012-04-02 10:03:17,986 ERROR [http-bio-8080-exec-9] Logger: near ")": syntax error 2012-04-02 10:03:19,478 ERROR [http-bio-8080-exec-11] Logger: near ")": syntax error 2012-04-02 10:03:22,617 ERROR [http-bio-8080-exec-2] Logger: near ")": syntax error 2012-04-02 10:06:55,416 ERROR [http-bio-8080-exec-14] Logger: near ")": syntax error 2012-04-02 10:08:57,373 ERROR [http-bio-8080-exec-19] Logger: near ")": syntax error 2012-04-02 13:41:19,426 ERROR [http-bio-8080-exec-121] Logger: database is locked 2012-04-02 13:41:24,691 ERROR [http-bio-8080-exec-124] Logger: database is locked 2012-04-02 13:41:44,102 ERROR [http-bio-8080-exec-128] Logger: database is locked 2012-04-02 13:42:05,725 ERROR [http-bio-8080-exec-130] Logger: database is locked

Abbildung 24: Logs-Tab

Konfiguration-Tab

Hier können die geplanten Futterzeiten, sowie die Futtermenge konfiguriert werden. Mithilfe von AJAX bietet das Konfigurations-Interface optimalen Userkomfort und ermöglicht eine einfache Eingabe und Speicherung der Futterzeiten und Futtermengen. Es können bis zu 24 Fütterungen pro Tag konfiguriert werden.

The screenshot shows the 'Konfiguration' tab selected in a navigation bar. Below the navigation bar, there is a form for configuring feedings. The form includes a label 'Anzahl der Fütterungen pro Tag:' followed by a text input field containing the number '3'. Below this, there are three rows of configuration for individual feedings. Each row consists of a number (1, 2, or 3), the word 'Fütterung', the word 'um', a time selection dropdown (all set to '15'), a colon, another time selection dropdown (all set to '44'), the word 'Uhr', the word 'mit', a third time selection dropdown (all set to '01'), and the text 'Futtereinheit(en)'. Below the configuration rows is a 'Speichern' button.

Row	Fütterung	um	:	Uhr	mit	Futtereinheit(en)
1.	Fütterung	um	:	15	Uhr	mit 01 Futtereinheit(en)
2.	Fütterung	um	:	15	Uhr	mit 02 Futtereinheit(en)
3.	Fütterung	um	:	15	Uhr	mit 03 Futtereinheit(en)

Speichern

Abbildung 25: Konfiguration-Tab

8 AJAX (Asynchronous Javascript and XML)

8.1 Allgemeines über AJAX

AJAX ist keine neue Programmiersprache, aber eine neue Art HTML und XML zu verwenden. Mit AJAX kann man dynamische Webseiten erstellen, die Befehle und Serveranfragen ausführen und dadurch auch die Seite verändern können, ohne die ganze Webseite neu laden zu müssen.

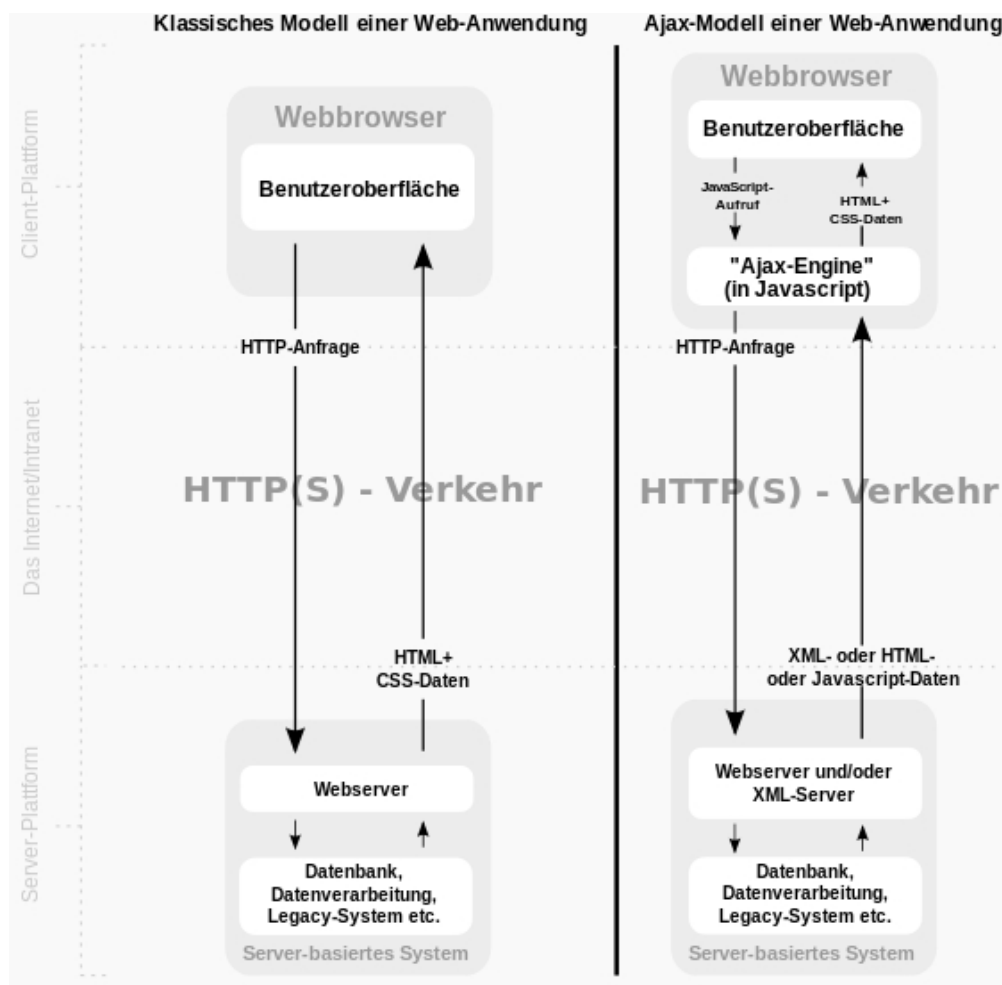


Abbildung 26: AJAX-Modell

AJAX funktioniert wie Javascript, inkludiert jedoch noch einen Server und verwendet diesen für Abfragen. Jede Benutzeraktion, die für gewöhnlich eine HTTP-Anfrage erzeugen würde, erzeugt nun einen JavaScript-Aufruf, der an die AJAX-Engine geschickt wird. Die AJAX-Engine verarbeitet dann in Echtzeit den Aufruf und führt ihn aus. Dadurch kann man eine desktopähnliche Oberfläche einer Webseite erstellen.

8.2 AJAX in unserer Diplomarbeit

In unserer Diplomarbeit haben wir AJAX nur beschränkt verwendet. Wir haben es einmal bei der Futterzeiteingabe verwendet. Funktionsweise: Es können beliebig viele Futterzeiten zwischen 1 und 24 Zeiten eingegeben werden. Dabei wird im Webinterface in ein Textfeld eine Zahl zwischen 1 und 24 eingegeben, je nachdem wie viele Fütterungszeiten man wünscht. Sobald die Zahl eingegeben wird, werden mittels AJAX so viele Futterzeit- und Futtereinheiteneingabefelder wie eingegeben angezeigt.

Anzahl der Fütterungen pro Tag:

1. Fütterung um : Uhr mit Futtereinheit(en).

2. Fütterung um : Uhr mit Futtereinheit(en).

3. Fütterung um : Uhr mit Futtereinheit(en).

Abbildung 27: AJAX im Webinterface

9 Github - „Social Coding“

9.1 Was ist Github?

Github ist eine Hosting-Website für Softwareprojekte.

Im Vordergrund stehen Funktionalitäten wie zum

Beispiel Versionsverwaltung, sowie Ähnlichkeiten zu

einem „Social Network“. Github wird auch häufig

genutzt, um für Open-Source-Projekte Freiwillige zu finden, die sich daran beteiligen

möchten, denn durch die einfache Verwendung von Github, ist es leicht, Code zu teilen und weiterzugeben.



Abbildung 28: Github-Logo

9.2 Funktionsumfang

Im Funktionsumfang ist zwischen 'git' und 'github' zu unterscheiden. 'git' stellt das Versionsverwaltungssystem dar, dass von dem Unternehmen entwickelt wurde, wobei 'github' der Onlinedienst ist, dass das 'git'-System mit Onlinehosting zur Verfügung stellt.

'github' ist kostenlos, jedoch ist eine Anmeldung notwendig und der Code und die zum Projekt hinzugefügten Dateien wird zwingend Open-Source. Jedoch gibt es die Möglichkeit sogenannte 'private repositories' gegen eine monatliche Gebühr zu erwerben.

'repositories'(dt. Lager, Aufbewahrungsort) sind Verzeichnisse auf 'github', in denen die hinaufgeladenen Dateien gelagert werden. 'private repositories' sind für niemanden zugänglich, der nicht als Mitarbeiter im Projekt eingeschrieben ist. Dies kann vom 'master' des Projekts festgelegt werden. Während der kostenlosen Nutzung von 'github' sind alle 'repositories' öffentlich, und für jeden 'github'-Verwender zugänglich. Dieser kann das Projekt ansehen oder auch für sich klonen, jedoch kann er keine aktive Änderung am Projekt vornehmen. Für diese Berechtigung ist wiederum eine Einschreibung als Mitarbeiter nötig, die auch wieder vom Projekt-'master' festgelegt werden muss.

'github' stellt unter anderem auch andere nützliche Funktionen zur Verfügung. Es wird zum Beispiel eine SSL-Verschlüsselung zur Übertragung der Daten angeboten und die Mitarbeiter eines Projekts können leicht über E-mails verwaltet, eingeteilt und informiert werden.

Außerdem gibt es eine Wiki-Abteilung auf der Webseite, die eine einfache Speicherung und Darstellung von Informationen ermöglicht. Eine Diskussionsabteilung ist ebenfalls vorhanden, bei der über das Projekt betreffende Probleme diskutiert werden kann.

9.3 Alternativen

Wir waren hauptsächlich auf der Suche nach Versionskontrolltools. Diese sind Tools zum Verwalten von Code, der von mehreren Personen bearbeitet wird. Diese Verwaltung verhindert zum Beispiel, Überschreibung von neuem Code mit altem und bietet Wiederherstellungsmöglichkeiten für frühere Versionen.

Von uns betrachtete Alternativen waren 'CVS' und 'Mercurial'. CVS wurde von uns im Unterricht bereits verwendet, deshalb war dies unsere erste Möglichkeit. Da wir aber eine Online-Lösung bevorzugen würden (für verbesserte Zusammenarbeit), sind wir zunächst auf 'Mercurial' gestoßen. Nachdem wir uns über 'Mercurial' informiert haben, haben wir von Prof. Gruber den Verweis auf 'github' bekommen. 'CVS' ist ein zentralliegendes System und deshalb für die Onlinezusammenarbeit nicht gut zu gebrauchen. 'Mercurial' bietet diese Onlineunterstützung, jedoch haben wir uns für 'github' entschieden, da es hervorragende Zusammenarbeitsstools zur Verfügung stellt. Wir hatten bisher keine Schwierigkeiten mit diesem Dienst, und es hat uns eine Menge Organisationsarbeit abgenommen.

9.4 git – Grundlagen

Das 'git' VKS (Versionskontrollsystem) unterscheidet sich in vielen Punkten von anderen bekannten VKS, wie zum Beispiel Subversion oder Perforce. Ein großer Unterschied ist die Art, mit der das 'git'-System Daten betrachtet. Die meisten VKS speichern Änderung der Daten dateibezogen ab, das heißt, es wird die Datei gespeichert und alle Änderungen, die gemacht wurden. 'git' speichert keine Änderungen ab, sondern speichert einen Schnappschuss vom Zustand des Projekts bei jeder Änderung. Aufgrund von Speichereffizienz werden Dateien, die nicht verändert wurden, nicht doppelt gespeichert, sondern werden zur bereits gespeicherten Datei referenziert.

Auch der Operationsbereich des Systems ist hauptsächlich lokal, wo die meisten anderen VKS über das Netzwerk arbeiten. Dies erhöht die Arbeitsgeschwindigkeit des Systems um Einiges, weil das ganze Projekt auf der lokalen Festplatte vorhanden. Um die Historie des Projekts anzuzeigen, muss mit keinem Server kommuniziert werden. Es lässt sich alles aus der lokalen Datenbank auslesen und es lassen sich auch Änderungen zwischen 2 Versionen einer Datei kalkulieren.

Ein weiterer Unterschied ist, das 'git' alles was es speichert, mit einer Prüfsumme referenziert. Das heißt, es vermeidet automatisch Übertragungsfehler und Dateibeschädigung mit dem Prüfsummenverfahren SHA-1. Es speichert Dateien nicht unter ihrem Name ab, sondern unter dem Hashwert des Inhalts.

Das 'git'-System wird in 3 Bereiche unterteilt, das Arbeitsverzeichnis, das Vormerkverzeichnis und das 'git'-Verzeichnis. Das Arbeitsverzeichnis ist nur ein Schnappschuss des Projekts in dem man arbeiten kann. Falls Änderungen gemacht wurden können sie per Zeilenkommando

```
git add .
```

in das Vormerkverzeichnis geladen werden. Dieses besteht nur aus einer Datei, in der die zu übertragenen Daten vorgemerkt werden.

Sollte nun der Zustand des Projekts gespeichert werden, wird der Projektschnappschuss aus dem Vormerkverzeichnis mit dem Zeilenkommando

git commit

in das lokale 'git'-Verzeichnis gespeichert. Dort ist es dann auf 'github' verfügbar und kann von anderen verwendet werden. Das Zeilenkommando

git add .

lädt alle Daten in das Vormerkverzeichnis, doch es ist auch möglich nur einzelne Dateien und Ordner hineinzuladen. Es muss nur statt dem '.' ein Ordner beziehungsweise eine Datei angegeben werden. Zuletzt wird noch der Befehl

git push

verwendet. Dieser synchronisiert das lokale 'repository' (nicht das Arbeitsverzeichnis!) mit dem 'github'-server.

Hier noch eine Abbildung zur Veranschaulichung.

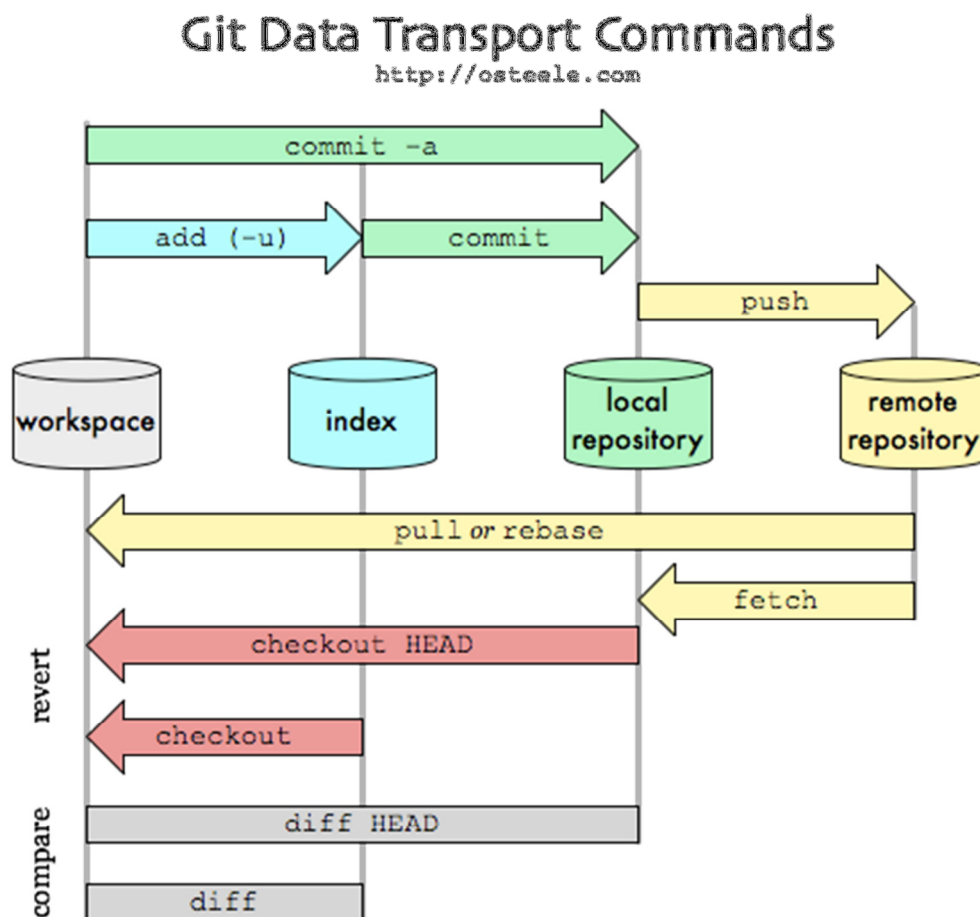


Abbildung 29: Github-Befehle

9.5 Wie wird 'github' verwendet?

9.5.1 Einrichtung

Wie bereits erwähnt kann jeder mit einer E-mail Adresse 'github' verwenden. Es ist für Windows, MAC OSX und für Linux verfügbar. Als Erstes wird die Software heruntergeladen und installiert. Heruntergeladen wird 'git' unter ⁽¹⁾. Unter Windows wird 'git' mit dem Installer installiert. Nun ist 'git' installiert, muss jedoch noch eingerichtet werden. Dazu starten wir die neu installierte Anwendung 'git Bash'. Angefangen wird mit dem Erstellen eines neuen SSH-Schlüssels, um eine sichere Verbindung zu gewährleisten. Dabei wird zuerst überprüft ob bereits ein Schlüssel vorhanden ist. Zeilenkommando:

```
cd ~/.ssh
```

Wird eine Fehlermeldung ausgegeben, ist kein Schlüssel vorhanden. Wenn nicht, dann müssen wir den alten Schlüssel absichern und ihn dann löschen. Zeilenkommando:

```
mkdir key_backup
```

```
cp id_rsa* key_backup
```

```
rm id_rsa*
```

Nun wo kein Schlüssel vorhanden ist, können wir einen neuen erstellen. Das geht mit Zeilenkommando:

```
ssh-keygen -t rsa -C 'E-mail'
```

Statt „E-mail“ wird die auf 'github' registrierte E-mail angegeben. Jetzt wird gefragt in welchem Verzeichnis der Schlüssel gespeichert werden soll. Für die Standardeinstellung wird 'enter' gedrückt. Nun wird eine Passphrase festgelegt, die den SSH-Schlüssel schützt. Diese wird zweimal eingegeben und sollte sich dann gemerkt oder mit einem Passwort-Tool gespeichert werden. Es sollte als Ausgabe eine Art Bild geben und ein paar Informationen über den SSH-Schlüssel.

Danach müssen wir den Schlüssel auf 'github' laden. Dazu loggen wir uns auf ⁽²⁾ ein und klicken rechts oben auf "Account Settings",

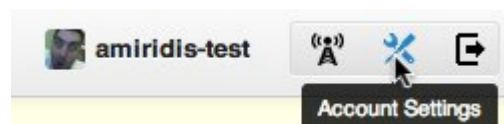


Abbildung 30: "Account Settings"

dann links auf "SSH Public Keys" und danach auf "Add another public key". Hier wird der Schlüssel rein kopiert, doch zuerst müssen wir ihn holen. Dazu öffnen wir das Verzeichnis, in dem der Schlüssel gespeichert ist und öffnen die Datei "id_rsa.pub" mit einem Texteditor.

Dann wird der gesamte Inhalt der Datei kopiert und in das dafür vorgesehene Textfeld eingefügt.

Hinzugefügt wird der Schlüssel mit dem Knopf "Add Key".

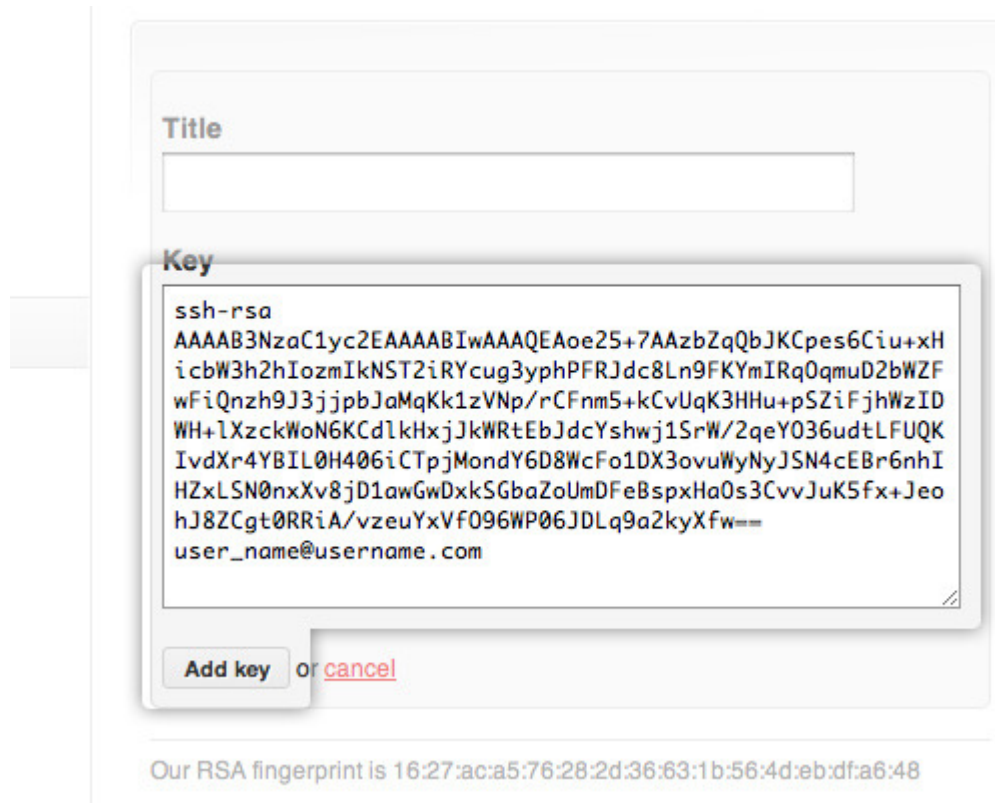


Abbildung 31: SSH-Public-Key

Um die Verbindung zu testen werden wir jetzt mit 'github' über SSH kommunizieren. Zeilenkommando dafür:

```
ssh -T git@github.com
```

Falls die Meldung *"The authenticity of host 'github.com (207.97.227.239)' can't be established"* aufscheint, wird *"yes"* eingegeben und 'enter' gedrückt. Nun sollte man mit seinem Benutzernamen begrüßt werden.

Weil 'github' bei allen Änderungen Benutzername und E-mail überprüft und speichert, müssen wir nun diese Informationen setzen. Dies wird mit diesem Zeilenkommando gemacht:

```
git config --global user.name "Vorname Nachname"
```

```
git config --global user.email "E-mail"
```

Einige Tools verbinden sich ohne SSH zu 'github'. Damit diese einwandfrei funktionieren, müssen wir einen API-Token konfigurieren, mit dem wir uns bei 'github' registrieren. Diesen API-Token finden wir auf der 'github'-Homepage. Dazu loggen wir uns erneut auf ⁽²⁾ ein und besuchen rechts oben "Account Settings" und hier wiederum links auf "Account Settings".



Abbildung 32: "Account Settings"

Hier sollte der API-Token angeführt sein.

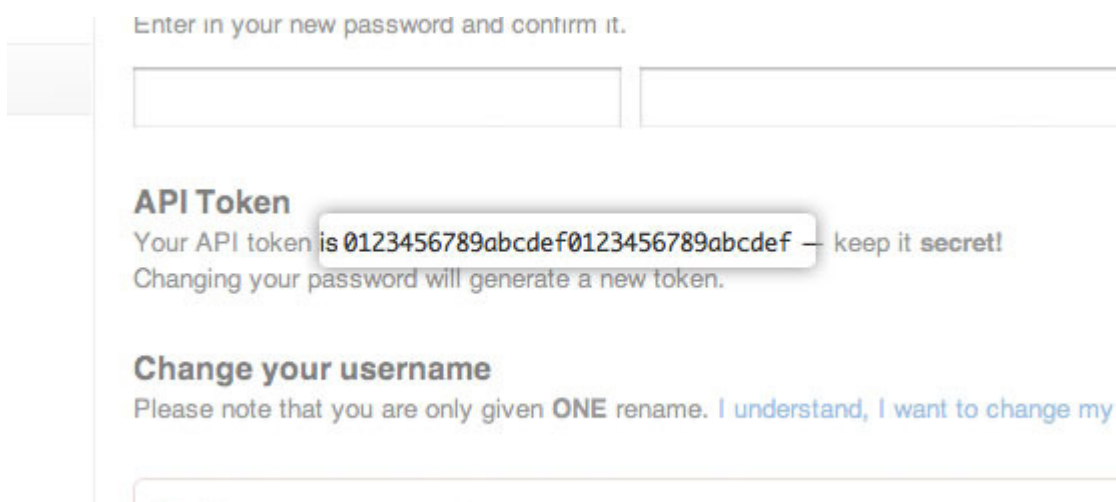


Abbildung 33: API-Token

Um ihn zu registrieren schreiben wir in die Kommandozeile:

```
git config --global github.user Benutzername
```

```
git config --global github.token ....
```

Wobei der Token hinter "github.token" geschrieben wird. Falls man auf 'github' das Passwort ändert, wird ein neuer Token erzeugt und man muss den neuen API-Token erneut in der Kommandozeile eingeben.

Wurden diese Einstellungen ohne Probleme konfiguriert, ist 'git' betriebsbereit und kann verwendet werden.

9.5.2 Verwendung

Um 'github' zu verwenden, muss als aller Erstes ein 'repository' erstellt werden, um das Projekt zu verwalten. Es wird mit einem Login auf ⁽²⁾ begonnen. Auf der Startseite ist unten rechts ein Button mit "New Repository" vorhanden. Für ein neues 'repository' wird dieser betätigt.

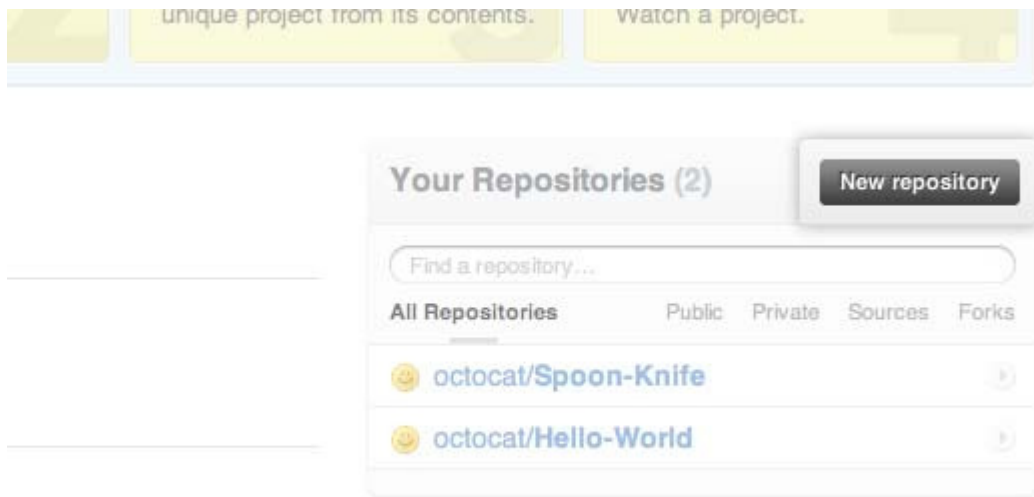


Abbildung 34: "New repository"

Nun wird man aufgefordert, das Projekt zu benennen und zu beschreiben.

Project Name: Name des Projekts

Description: Beschreibung des Projekts

Homepage URL: Falls eine Homepage des Projekts vorhanden ist, hier eingeben.

Die Auswahlmöglichkeiten ganz unten, regeln die Zugriffsberechtigungen des Projekts. Für ein kostenloses 'repository', muss das Projekt für alle 'github'-Benutzer freigegeben sein. Falls man eine privates Projekt starten möchte, ist dies gegen eine monatliche Gebühr möglich.

Create a New Repository

Project Name

Description (optional)

Homepage URL (optional)

Who has access to this repository? (You can change this later)

☒ **Anyone** ([learn more about public repos](#))

☐ **Only the people I specify** ([learn more about private repos](#))

Create repository

Abbildung 35: "Create repository"

Jetzt kann man schon mit dem Projekt beginnen. Nun wird beschrieben wie man 'github' während des Projekts verwendet.

Um die Verwendung besser zu veranschaulichen, wird ein alltäglicher Verwendungsprozess beschrieben.

Ein Benutzer von 'github' ist Teil eines Projekts und hat auf seinem Rechner die Datei 'readme.txt' erstellt. Diese will er nun mit seinen Projektkollegen teilen. Dazu muss er die Datei in das Arbeitsverzeichnis legen. Da er noch kein Arbeitsverzeichnis hat, wird zuerst eines angelegt.

```
mkdir ~/Hello-World
```

Dieser Befehl erstellt einen Ordner auf dem Rechner des Benutzers, jedoch muss noch ein lokales 'repository' in diesem Ordner erstellt werden.

```
cd ~/Hello-World
```

```
git init
```

Der erste Befehl wechselt in den neu erstellten Ordner. Der zweite Befehl initialisiert 'git' in diesem Ordner und erstellt ein lokales 'repository'. Es sollte eine solche ähnliche Meldung erscheinen:

"Initialized empty Git repository in /Hello-World/.git/"

Jetzt ist das Arbeitsverzeichnis erstellt und der Benutzer kann die "readme.txt" in das Arbeitsverzeichnis geben.

Jetzt fügt er die Datei zum Vormerkverzeichnis hinzu. Dies geschieht mit

```
git add readme.txt
```

Danach wird das Vormerkverzeichnis, in das die Datei gegeben wurde, in das lokale 'repository' geschrieben. Kommandozeilenbefehl:

```
git commit -m 'Message'
```

Nach **-m** kann eine beliebige Beschreibung hinzugefügt werden, die die Veränderung, das heißt den Inhalt des Vormerkverzeichnisses, beschreibt.

Alle Schritte bis jetzt haben nur auf lokaler Ebene gewirkt. Um das lokale Verzeichnis mit 'github' zu verbinden, muss es auf 'github' geschoben(push) werden. Zuerst wird jedoch bestimmt, in welches 'repository' wir das lokale 'repository' schieben möchten.

```
git remote add origin git@github.com:Benutzer/Hello-World.git
```

Jetzt wo das Ziel definiert wurde, kann synchronisiert werden:

```
git push origin master
```

Jetzt wurde die Datei auf den 'github'-Server geladen und kann auch online auf ⁽²⁾, nach erfolgreichem Login, unter dem definiertem 'repository' eingesehen werden.

⁽¹⁾ <http://help.github.com/win-set-up-git/>

⁽²⁾ <https://github.com/>

10 Elektronik

10.1 Feststellen des Widerstandswerts eines Widerstands

Ein jeder Widerstand verfügt über einen aufgedruckten Farbcode. Mithilfe dessen kann der tatsächliche Widerstand herausgefunden werden. Der erste, zweite und dritte Farbstreifen wird dann folgendermaßen zusammengerechnet:

Der erste Farbstreifen repräsentiert die erste Ziffer einer zweistelligen Zahl; der zweite Farbstreifen repräsentiert die zweite Ziffer einer zweistelligen Zahl. Die zusammengesetzte Zahl wird anschließend mit der Zahl, die durch den dritten Streifen repräsentiert wird, multipliziert. Der vierte Streifen repräsentiert lediglich die Toleranz des Widerstands.

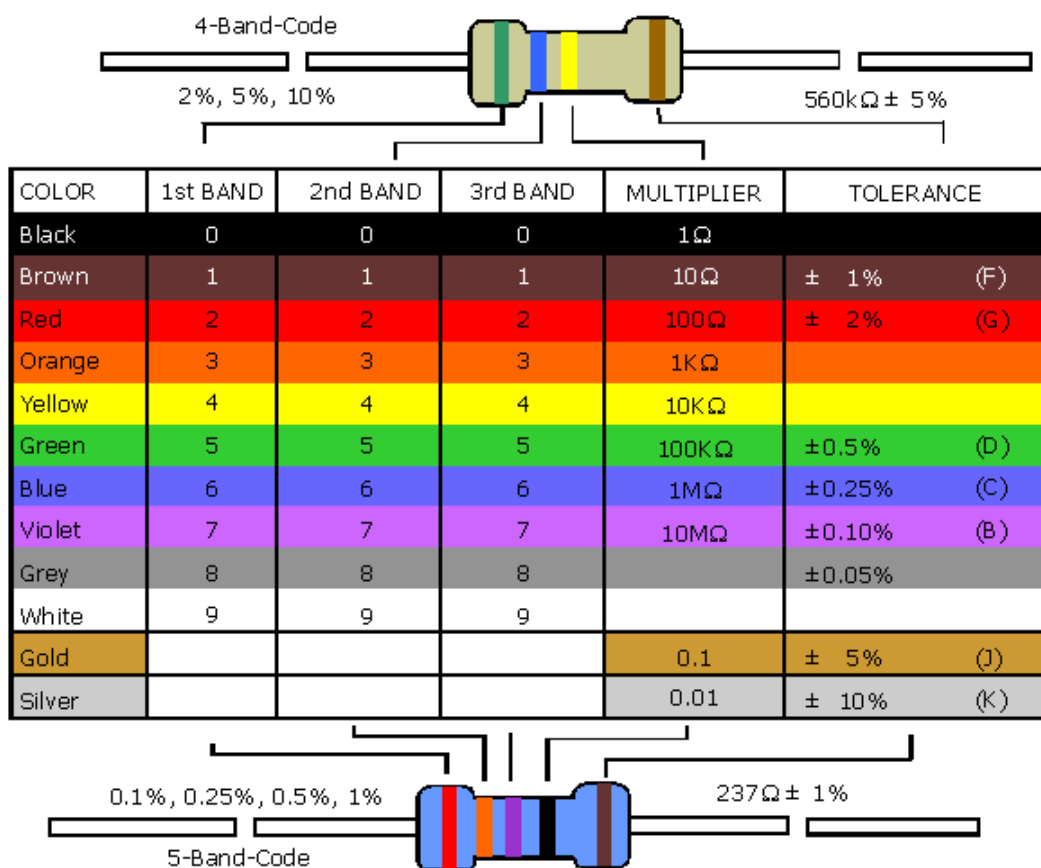


Tabelle 2: Ermittlung des Widerstandswerts

10.2 Löten

Mittels Löten erzeugt man bedingt lösbare Verbindungen zwischen zwei Elektroden. Ziel dabei ist es, dass elektrischer Strom zwischen den Verbindungen fließen kann und dass das Lot dabei einen möglichst geringen Widerstand verursacht. Lotverbindungen stellen aus folgenden Gründen potentielle Probleme dar:

- Sie sind eine Quelle für Korrosionen (Galvanischer Effekt)
- Sie können sich durch Temperaturwechsel lockern
- Oxidschichten verursachen einen erhöhten Widerstand an Lötverbindungsstellen

10.3 Verdrahtung

Einige unserer elektronischen Schaltungen mussten vor dem Lötvorgang auf Funktionalität auf einem Steckbrett getestet werden (z.B. Verstärkerschaltung). Beim Verdrahten von elektronischen Bauteilen ist es meist sinnvoll sich vor Beginn der Arbeit einen Schaltplan aufzuzeichnen. Freeware-Tools wie z.B. Dia eignen sich dafür optimal. Vor dem Verdrahten sollte man sich die für die Schaltung benötigten Bauteile auf einer Arbeitsplatte vorbereiten. Dadurch vermeidet man den Verlust diverser Bauteile. Ein hell beleuchteter Arbeitsplatz ist Pflicht! Sind alle Vorbereitungen getroffen, kann mit der Verdrahtung begonnen werden. Dabei ist jedoch zu beachten, dass Drähte in mehreren Farben verwendet werden sollten. Dies hat den Vorteil, dass man die Schaltung besser und auch schneller nachvollziehen kann.

10.4 Sensorschaltung

Unser Temperatursensor (KTY-222) weist im Bereich von 0 bis 50°C eine nahezu optimale Linearität im Verhältnis zwischen Temperatur und Widerstand auf. Deshalb hält sich unsere Sensorschaltung in Grenzen:

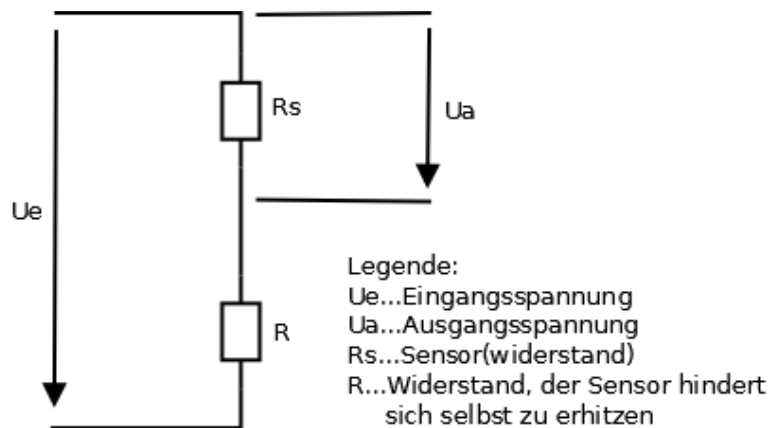


Abbildung 36: Sensorschaltung

10.5 Verstärkerschaltung

Damit wir die Temperatur genauer bestimmen konnten versuchten wir - so gut wie möglich - den gesamten Spannungselebereich des AVR-NetIO-Boards ausnützen. Dazu probierten wir, die Ausgangsspannung der Sensorschaltung mit einer Verstärkerschaltung zu verstärken. Wir verwendeten dazu ein Bauelement namens „LM324“. Dieses verfügt über vier Verstärker.

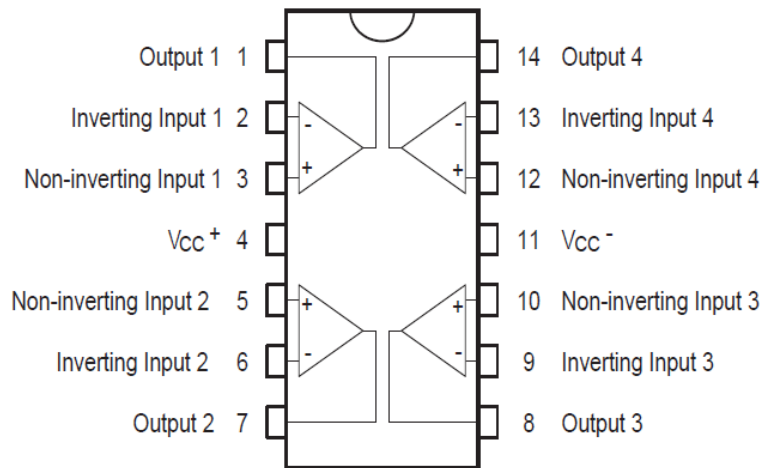


Abbildung 37: LM324

Wir entschieden uns für eine doppelte Verstärkung. Dazu wurde folgende Schaltung mit folgenden Widerstandswerten entworfen:

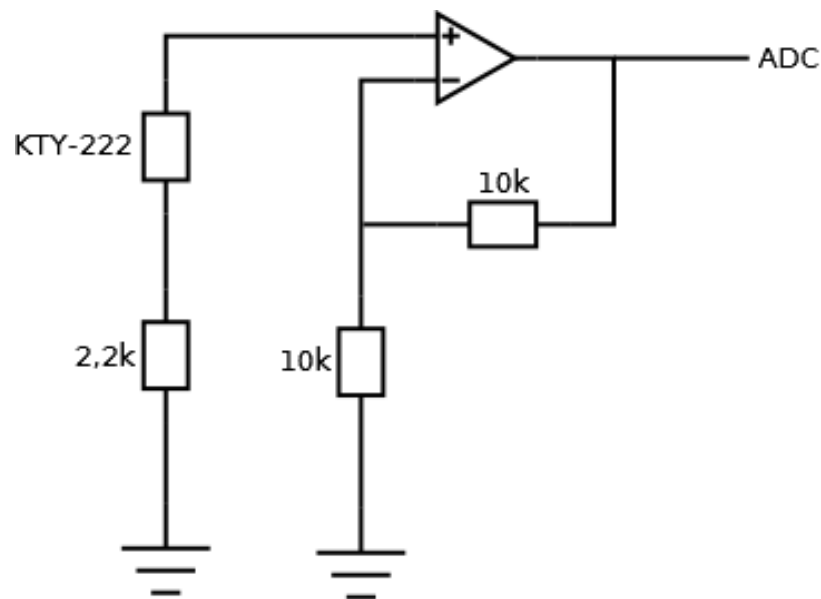


Abbildung 38: Verstärkerschaltung

Als die Schaltung vollständig aufgebaut war, testeten wir den Spannungsausgang und wir konnten tatsächlich eine Verstärkung der Eingangsspannung feststellen. Die Schaltung wurde anschließend abgebaut. Als wir sie wieder benötigten, funktionierte diese aus uns nicht bekannten Gründen nicht mehr. Wir entschieden uns deshalb für eine nicht-verstärkte Ausgangsspannung.

10.6 Diodenschaltung

Bei dieser Schaltung versuchten wir die Referenzspannung des AVR-NetIO-Boards von 5V auf 1V zu minimieren. Dazu verwendeten wir eine in verkehrter Richtung eingebaute Z-Diode in Kombination mit einem 1M Ω Widerstand. Die Spannung konnte somit erfolgreich auf 1.1V reduziert werden. Eine Steigerung der Messgenauigkeit konnte jedoch nicht festgestellt werden. Die externe Referenzspannung wurde sozusagen vom AVR-NetIO-Board einfach ignoriert.

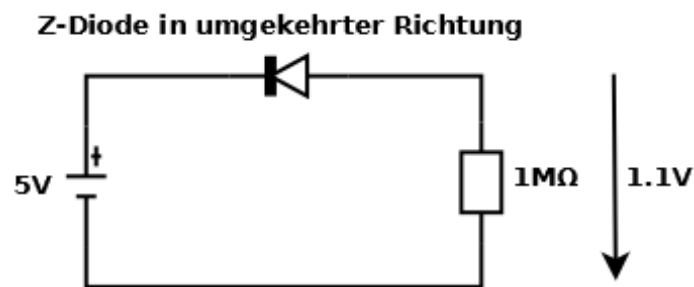


Abbildung 39: Diodenschaltung

10.7 Komplette Schaltung

Die komplette Schaltung besteht aus:

- 1 x AVR-NetIO-Board
- 1 x K8IO-Relais-Karte
- 2 x Temperatursensoren
- 1 x Lichtschranke
- 2 x 10K-Widerstand
- 1 x 1K-Widerstand
- 1 x Motor
- 1 x 9V Spannungsversorgung (Netzteil)
- 1 x 4.5V Spannungsversorgung (Netzteil)
- 1 x Motor

Ein Bild der kompletten Schaltung befindet sich auf der nächsten Seite.

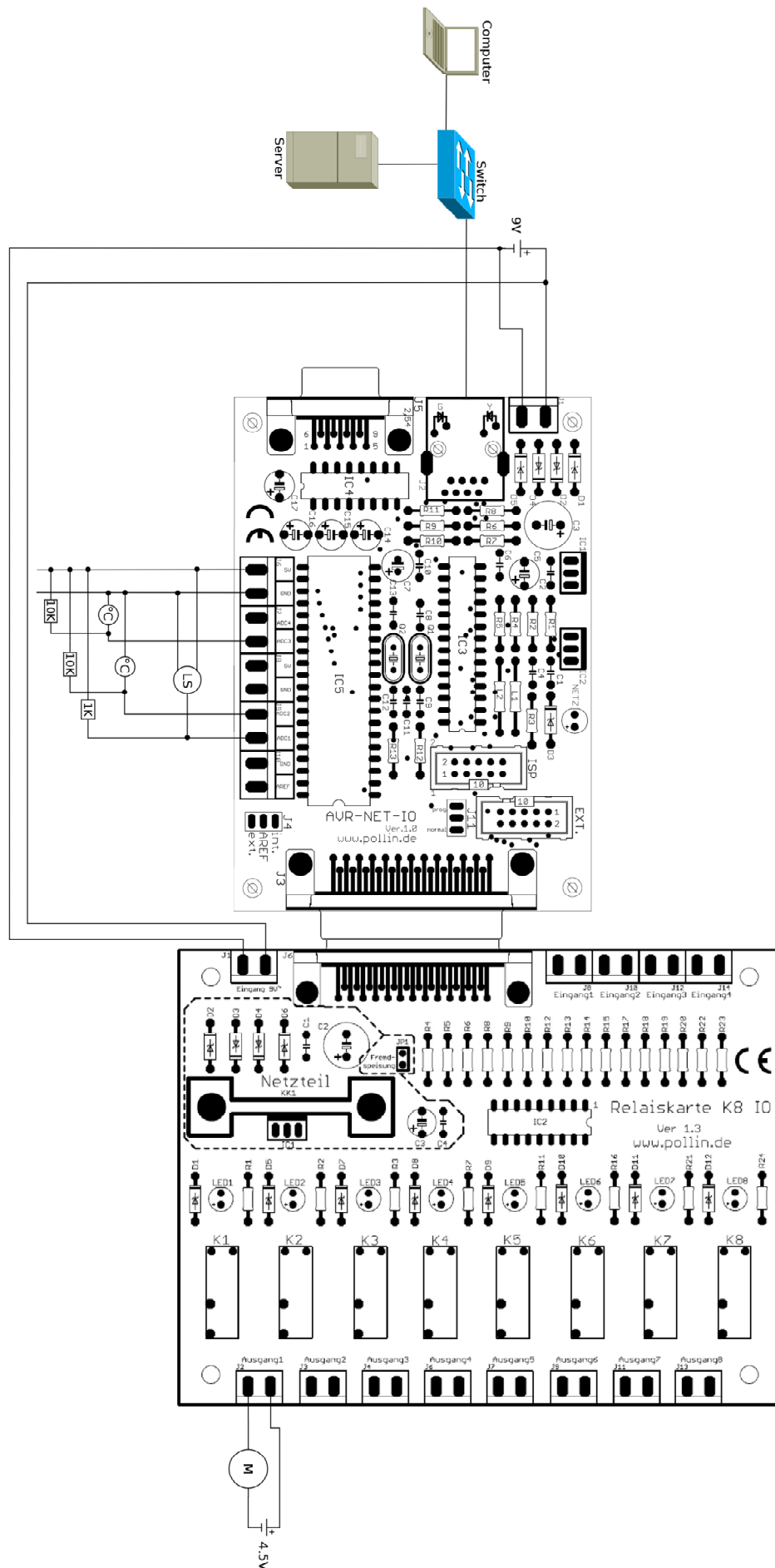


Abbildung 40: Komplette Schaltung

11 Hardware

11.1 AVR-NET-IO-Board

11.1.1 Allgemeines über das AVR-NET-IO-Board

Das AVR-NET-IO-Board ist eine Experimentier- und Lernplatine, zudem ist sie auch eine vollständig unabhängig arbeitende Ethernetplatine, die mit einem Netzwerkcontroller und einem ATmega32 Microcontroller ausgestattet ist. Ein Datenblatt befindet sich im Anhang.

Das Board ist vor allem für Temperaturmessungen ausgelegt. Der einfache Befehlsatz ermöglicht leichte Einbindung in Projekte und Anwendungen, auch ohne die mitgelieferte Software 'netserver'. Auch eigene Firmware kann entwickelt werden und über eine ISP-Schnittstelle auf den Microcontroller übertragen werden.⁵



Abbildung 41: AVR-NET-IO-Board

11.1.2 Features

- Netzwerkcontroller ENC28J60
- Microcontroller ATmega32
- 8 digitale Ausgänge
- 4 digitale Eingänge
- 4 ADC-Eingänge (10 bit)
- RS232-Schnittstelle
- Separate Referenzspannungszuführung
- Betriebsspannung 9V~
- Abmaße: 108x76x22 cm

⁵ (AVR-NET-IO - Fertigmodul)

11.1.3 Befehle

Alle Befehle können in einem Terminal eingegeben oder als TCP/IP Daten an den Port 50920 der Platine geschickt werden.

Analogen Eingang abfragen

GETADC X

Gibt den aktuellen Analogwert am Eingang X zurück

Ausgang schalten

SETPORT X.Y

Setzt den Ausgang X auf Y (high-1, low-0)

Diese 2 Befehle haben wir in unserer Diplomarbeit verwendet, GETADC um den Wert der Temperatursensoren abzufragen und in Temperaturwerte umzuwandeln und SETPORT um die Relaiskarte anzusteuern.

11.1.4 Alternativen

Da es sich bei dieser Platine um eine Experimentier- und Lernplatine handelt ist sie preislich sehr günstig zu erwerben. Alternativen zu diesem Board wären durchaus vorhanden gewesen, wären aber preislich und im Funktionsumfang 'overkill' (mehr als nötig) gewesen.

11.2 K8IO-Relaiskarte

11.2.1 Allgemeines über die K8IO-Relaiskarte

Eine Karte mit 8 Relais zum Schalten von diversen Geräten. Anschließbar über den seriellen Port eines PCs oder, wie in unserem Fall, über den seriellen Port des AVR-NET-IO-Boards. Ein Datenblatt befindet sich im Anhang.

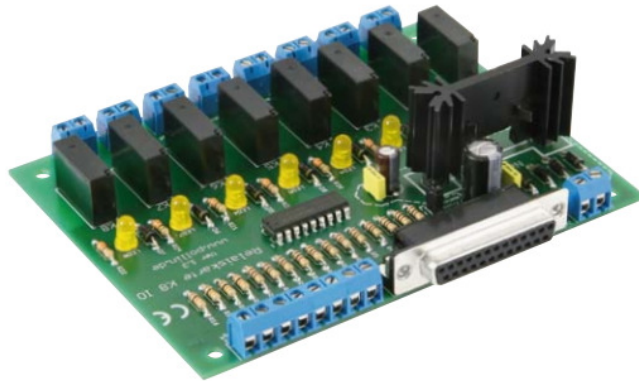


Abbildung 42: K8IO-Relaiskarte

Wir verwenden diese Relaiskarte in Verbindung mit dem AVR-NET-IO-Board um den Motor der Fütterungsmechanik anzusteuern.

11.2.2 Relais

Ein Relais ist ein Schalter der durch elektrischen Steuerstrom betrieben wird. Der Steuerstromkreis (im Bild, linke Anschlüsse) schaltet durch einen Elektromagnet einen Lastenstromkreis (im Bild, rechte Anschlüsse). Durch einen Elektromagneten wird der Lastenstromkreis geschlossen, sobald der Steuerstromkreis durchschaltet.⁶

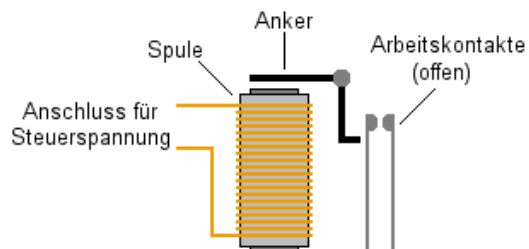


Abbildung 43: Funktionsweise eines Relais

11.2.3 Alternativen

Es gibt genug Relaiskarten, die geeignet wären unsere Anforderungen zu erfüllen, diese Relaiskarte wurde von uns als erstes gewählt.

⁶ (Relais)

11.3 KTY-222 Temperatursensor

Der KTY-222 Temperatursensor ist ein einfacher und kostengünstiger Sensor, der eine annähernd lineare Temperaturkurve besitzt. Ein Datenblatt befindet sich im Anhang.

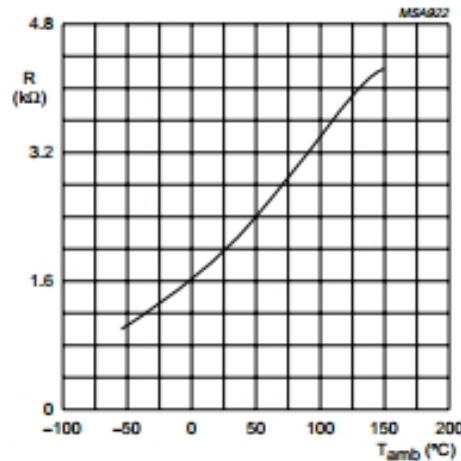


Abbildung 44: Temperaturkurve des KTY-222

X-Achse: Temperatur

Y-Achse: Widerstand

In unserem Diplomprojekt haben wir 2 solche Sensoren verwendet, einen Wassertemperatursensor und einen Lufttemperatursensor.

11.4 LM324-Verstärkungs-IC

Der LM324 ist ein IC mit 4 Verstärkern, der in unserem Projekt den Messbereich der Analogwerte vergrößern sollte.

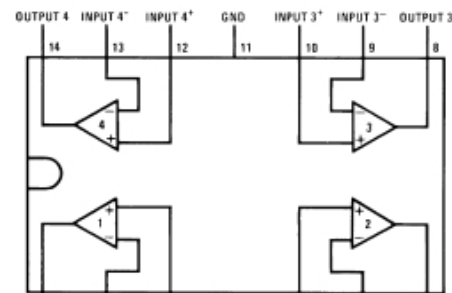


Abbildung 45: LM324-Verstärkungs-IC

Unsere analogen Werte reichten für den Temperaturbereich 10-40° von 0,6-0,9V. Diesen wollten wir auf 2,4-4,5V erhöhen.

Dieses Bauteil wurde uns von Fachlehrer Gruber empfohlen und bereitgestellt.

11.5 Lichtschrankensensor EESX-671

Der Lichtschrankensensor EESX-671 kann bei Unterbrechung des Infrarotstrahls einen Spannungsabfall bzw. Spannungsanstieg erzeugen, der dann ausgewertet werden kann. Dies wird meist für Positionsabfragen verwendet. Ein Datenblatt befindet sich im Anhang.



Abbildung 46:
EESX-671

In unserem Projekt benötigen wir eine Positionsabfrage am Futterbehälter, um zu wissen, wann der Behälter 1 Umdrehung/Fütterung erledigt hat.

12 Materialkosten

Wie bereits am Anfang unserer Diplomschrift erwähnt, soll unsere Diplomarbeit möglichst preiswert umgesetzt werden können. Unser Ziel war es eine automatische Fischfütterungsanlage zu konstruieren, deren Kosten nicht über € 100,- hinauslaufen dürfen. Unter Berücksichtigung dieses einschränkenden Faktors ist es uns jedoch gelungen, einen Prototyp zu einem überschaubaren Preis von € 77,10 zu erstellen!

Stücklistentabelle mit dazugehörigen Kosten:

Artikel	Kosten
AVR-Net-IO-Board	€ 27,95
K8IO-Relaiskarte	€ 12,95
Schaltdrähte und Schaltlitzen	€ 4,95
Streifenrasterplatine	€ 0,60
Lötzinn	€ 7,95
Kabelführungs-Deckel (silber)	€ 7,30
Chrom-Spray	€ 8,80
Sperrholz-Platte	€ 0,60
Alu-Rohr	€ 4,00
Langschraube	€ 0,50
Kleinmaterial	€ 1,50
Gesamt:	€ 77,10

Tabelle 3: Materialkosten

13 Zusammenbau

Der Zusammenbau unseres Prototypen besteht aus 8 Komponenten.

13.1 Grundplatte

Es handelt sich hierbei um eine gewöhnliche Sperrholzplatte mit den Abmaßen 30x35x2cm. Beinahe alle Komponenten werden mit der Grundplatte verschraubt.

13.2 AVR-NET-IO-Board

Dieses Bauteil stellt das Kernstück unserer Elektronik dar. Es ermöglicht das Einlesen von Spannungswerten und überträgt die Daten über die Netzwerkschnittstelle.

13.3 K8IO-Relaiskarte

Dieses Bauteil ermöglicht das Ein- und Ausschalten des Motors. Es ist mit der seriellen Schnittstelle des AVR-NET-IO-Boards verbunden und kann somit auch über das Netzwerk angesteuert werden.

13.4 Streifenrasterplatine

Auf dieser Platine befinden sich die erforderlichen Sensorschaltungen. Die Spannungs-Ein und -Ausgänge des AVR-NET-IO-Board sind mit der Streifenrasterplatine verbunden. Die Messung der Spannung wurde somit relativ komfortabel realisiert.

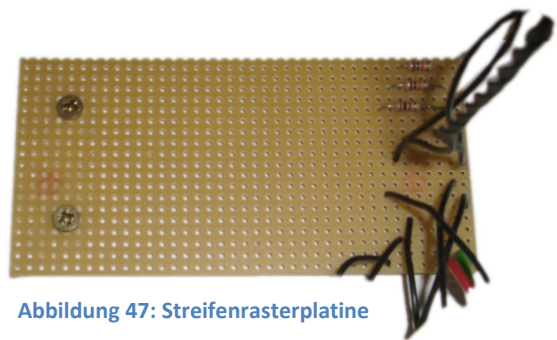


Abbildung 47: Streifenrasterplatine

13.5 Lego-Motor und -Getriebe

Zum Rotieren des Futterbechers wird ein Lego-Motor in Kombination mit einem Lego-Getriebe verwendet. Da sich der Motor allein zu schnell drehen würde, wird ein Lego-Getriebe zur Reduzierung der Drehzahl verwendet.

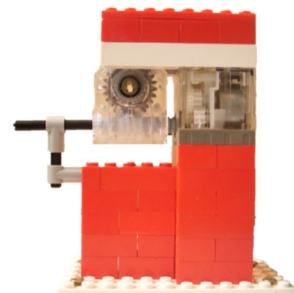


Abbildung 48: Lego

13.6 Futterbecher

Als Futterbecher wurde auf Empfehlung von Herrn Gruber ein gewöhnlicher Joghurtbecher verwendet. Dieser ermöglicht durch seine konische Form die Abgabe des Futters. Zur besseren Kontrolle des Streuwinkels des Futters wurde ein Kabelkanaldeckel an der Öffnung des Bechers befestigt. Noch dazu wurde ein kleines Plättchen zur Durchbrechung des Lichtschranks am Kabelkanaldeckel angebracht.



Abbildung 49:
Futterbecher

13.7 Ausgleichgewicht

Das Ausgleichsgewicht wird zu Stabilitätszwecken auf der Achse des Futterbechers befestigt. Schwingungen sowie Vibrationen werden dadurch vermindert.



Abbildung 50:
Ausgleichgewicht

13.8 Galgen

Dieser besteht aus:

- einem Aluminiumrohr
- einer Schraube
- einem Holzstück
- einer Lichtschranke

Der Galgen muss genau ausgerichtet werden um das Passieren des Plättchens am Futterbecher zu gewährleisten.

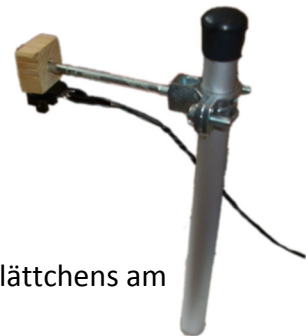


Abbildung 51: Galgen

13.9 Kompletter Aufbau

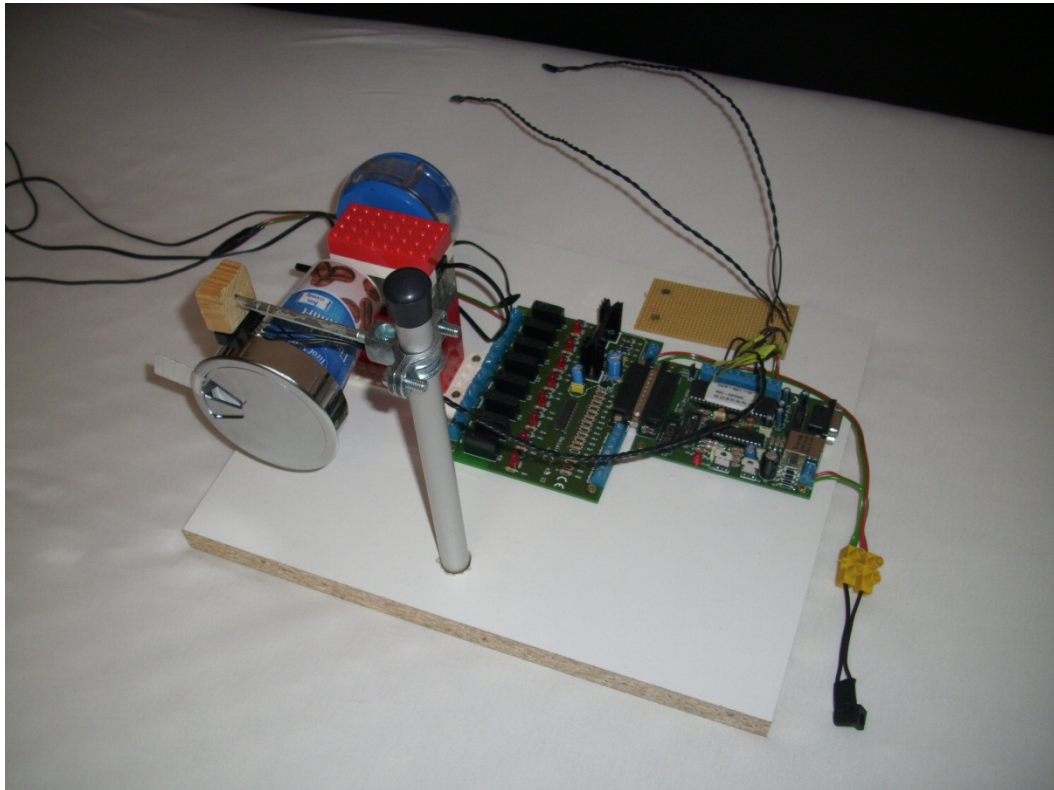


Abbildung 52: Kompletter Aufbau 1

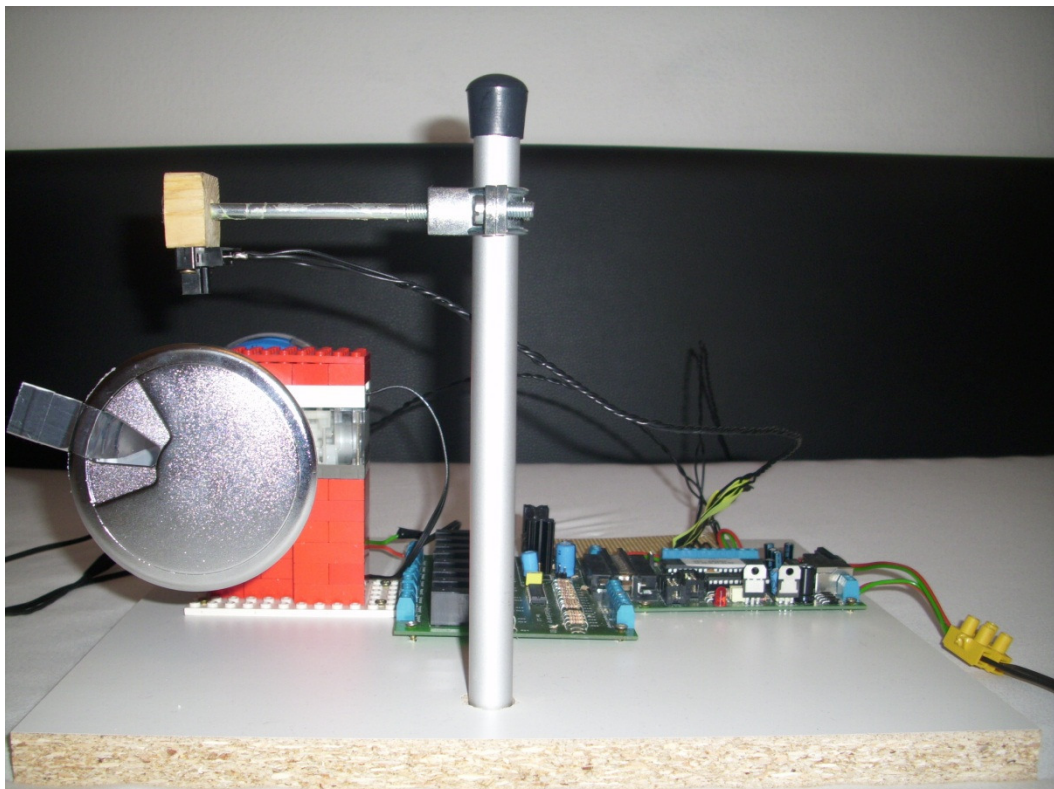


Abbildung 53: Kompletter Aufbau 2

14 Inbetriebnahme der Anlage

In diesem Kapitel wird erklärt, wie die Software für unsere Fischfütterungsanlage lauffähig gemacht werden kann. Um die folgenden Schritte durchzuführen, ist die auf einer CD befindliche Datei „fishapp.zip“ erforderlich.

14.1 Entpacken der erforderlichen Dateien

Zum Entpacken von ZIP-Dateien ist im Normalfall keine weitere Software nötig. Zu entpacken ist lediglich die Datei „fishapp.zip“.

14.2 Festlegen der Umgebungsvariablen

Je nach Betriebssystem gibt es verschiedene Möglichkeiten Systemvariablen festzulegen. Hier wird nun am Beispiel von Windows 7 gezeigt wie Systemvariablen festgelegt werden können. Dazu klickt man mit einem Rechtsklick auf „Computer“ und klickt auf „Eigenschaften“. Anschließend klickt man auf „Erweiterte Systemeinstellungen“ und anschließend auf den Button „Umgebungsvariablen...“. Hier definiert man 2 neue Systemvariablen und zwar:

- JAVA_HOME → Im Normalfall befindet sich der JAVA_HOME – Pfad unter „C:\Program Files\Java\jre7\bin“
- JRE_HOME → Im Normalfall befindet sich der JRE_HOME – Pfad unter „C:\Program Files\Java\jre7“

14.3 Starten des Servers

Zum Starten des Servers muss die Datei „startserver.bat“ im Verzeichnis „fishapp“ gestartet werden.

14.4 Starten des Hauptprogrammes

Zum Starten des Hauptprogramms muss die Datei „startapp.bat“ im Verzeichnis „fishapp“ gestartet werden. Um volle Funktionalität zu gewährleisten, muss beim ersten Starten des Hauptprogramms der Befehl „reset“ eingegeben werden und mit der Eingabetaste bestätigt werden.

14.5 Lokales Testen des Webinterfaces

Dazu muss nur ein beliebiger Internetbrowser geöffnet werden und die Adresse „localhost:8080/fish“ geöffnet werden. Hat alles geklappt erscheint nun der Anmeldebildschirm des Webinterface. Hier kann man sich nun sofort mit Username „foo“ und Password „bar“ anmelden, oder man erstellt gleich den richtigen Benutzer. Dazu klickt man auf den Link „Forgot password?“ und man kann seine gewünschten Anmeldedaten eingeben. Unter „Authentication code“ muss folgender Code eingetragen werden:

mpLiBGsDwwZS8ntAsiWg6Zmm3WF6TNDGCXyMmrYjn8Cuu55nmaUsAGeCmsIFGUV

14.6 Abschließende Konfiguration

Wenn die obigen Schritte durchgeführt wurden, können bereits die geplanten Futterzeiten und –mengen unter dem Tab „Konfiguration“ konfiguriert werden.

15 Sicherheit

15.1 Login

Um in unser Projekt auch den Sicherheitsaspekt einzubauen, haben wir uns überlegt, eine Loginstruktur zu erstellen. Der User kann sich registrieren und wird dann in eine User-Datenbank gespeichert, die wir zu diesem Zweck angelegt haben. In dieser wird Loginname und das gehashte Passwort abgelegt, das beim Login abgefragt wird.



The screenshot shows a web interface for a fish feeding system. At the top is a blue header. Below it is a navigation bar with five buttons: 'Home', 'Graph', 'Tabelle', 'Logs', and 'Konfiguration'. The main content area is white and contains a login form. The form has two input fields: 'Username:' and 'Password:'. Below the 'Password:' field is a link that says 'Forgot password?'. To the right of the 'Forgot password?' link is a 'Login' button.

Abbildung 54: Anmeldung am Webinterface

15.2 Session

Erweiterte Sicherheit haben wir mit Sessions sichergestellt. Ohne einen erfolgreichen Login kann das Webinterface nicht in Betrieb genommen werden, da wir in der aktuellen Session nachschauen, ob ein erfolgreicher Login vollzogen wurde, da wir 'leeren' Sessions den Zugang verweigern. Zu dem löschen wir Sessions, die länger als 15 Minuten inaktiv waren.

15.3 Szenarien

15.3.1 Server fällt aus

Zugriff auf das Webinterface ist nicht mehr möglich. Auch die Fütterung fällt aus, da der Server keine Befehle mehr an das NET-IO-Board senden kann.

15.3.2 Strom fällt aus

NET-IO-Board und Server kein Problem. Datenbanken bleiben durch SQLite unbeschädigt. Stromausfall während Datenveränderungen kann die Datensätze verunstalten.

15.3.3 Netzwerkverbindung unterbrochen

Fütterung ist nicht mehr möglich, da keine Verbindung mehr zum NET-IO-Board besteht. Auch Temperaturdaten werden nicht mehr erhalten.

15.3.4 Internetverbindung unterbrochen

Da das ganze Projekt auch ohne Internetverbindung funktionieren kann, ist ein Verbindungsausfall kein großes Problem. Nur die E-Mail Benachrichtigungen können nicht mehr weggeschickt werden.

16 Umsetzungsentscheidungen

16.1 Konstruktion „Box mit Förderband“

16.1.1 Allgemeines über diese Konstruktionsidee

Diese Konstruktionsidee entstand in den ersten Phasen unserer Diplomarbeit. Ein Förderband soll das in einer Box befindliche Futter bei Bedarf abtransportieren. Ist das Ende des Förderbands erreicht, fällt das Futter ins Wasser und die Fische werden gefüttert.

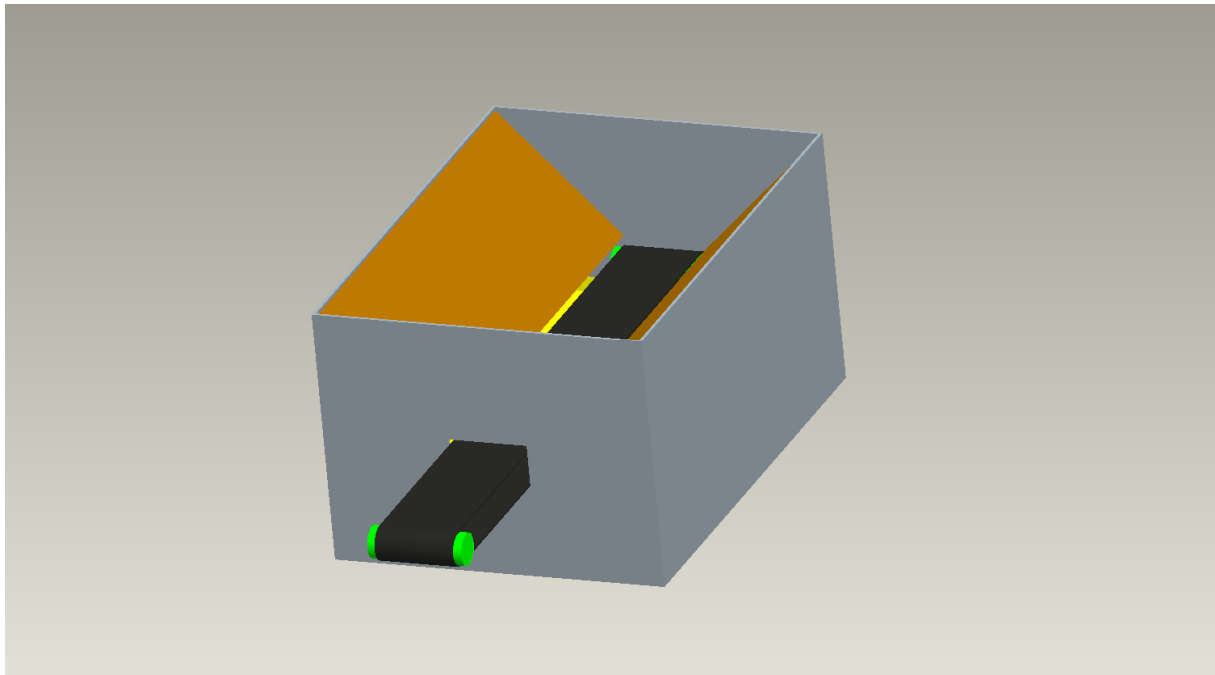


Abbildung 55: Box mit Förderband

16.1.2 Warum haben wir uns nicht für diese Möglichkeit entschieden?

Das Problem bei diesem Konstruktionsansatz ist, dass die Reinigung dieses Aufbaus relativ schwer möglich ist, sowie allgemein der Konstruktionsaufwand zu groß und zu zeitintensiv wäre.

16.2 Konstruktion „Rohr“

16.2.1 Allgemeines über diese Konstruktionsidee

Diese Konstruktionsidee wurde nach Ablehnung der ersten Konstruktionsidee geplant und konstruiert. 2 Rohre sollen das Futter durch eine Drehung portionieren und anschließend den Fischen verabreichen.

16.2.2 Warum haben wir uns nicht für diese Möglichkeit entschieden?

Diese Konstruktionsidee wurde nach kurzer Zeit auch abgelehnt, da die fertigungstechnische Umsetzung relativ schwierig ist und es für einen normalen Motor beinahe unmöglich ist die beiden nicht-gelagerten Rohre zu rotieren. Des Weiteren kann die Reinigung dieses Aufbaus nicht problemlos von Statten gehen.

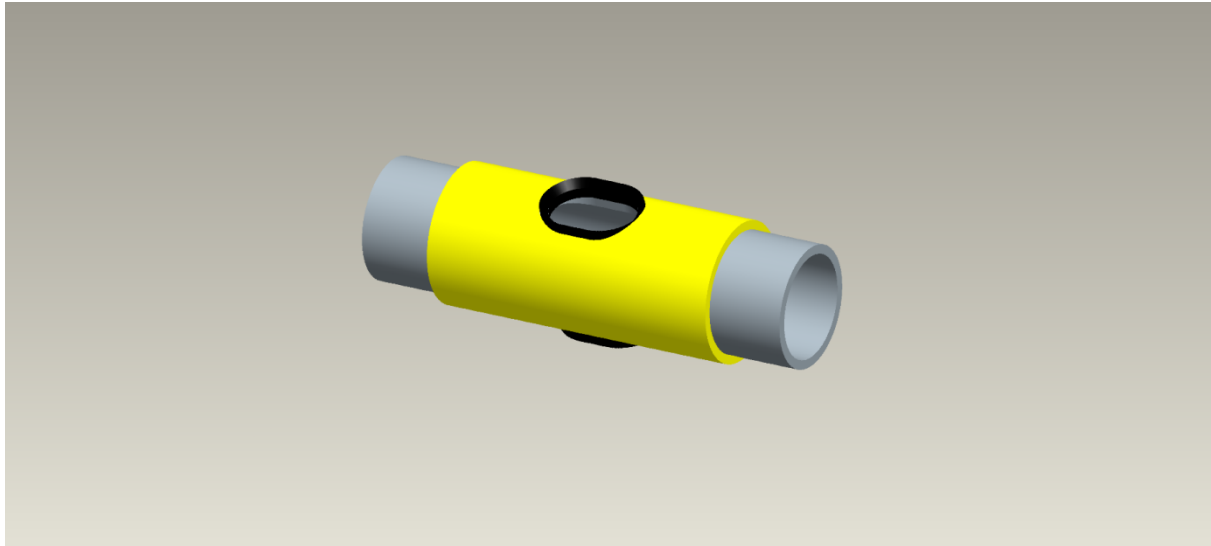


Abbildung 56: Rohr

16.3 Programmiersprache JAVA

16.3.1 Allgemeines über diese Programmiersprache

JAVA ist eine objektorientierte Programmiersprache, die auf beinahe jedem Betriebssystem ausführbar ist. Außerdem ist JAVA vollkommen kostenlos und JAVA findet die meiste Unterstützung im Internet (Foren, etc.).

16.3.2 Warum haben wir uns für diese Programmiersprache entschieden?

Mit JAVA haben wir in unserer Schulausbildung, sowie in unserer Freizeit am meisten Erfahrung gesammelt und sind deshalb in diesem Gebiet schon relativ kompetent. Ein weiterer Grund warum wir uns für die Programmiersprache JAVA entschieden haben ist, dass JAVA auf beinahe jedem Betriebssystem mit installierter JAVA-Virtuellen-Maschine funktioniert.

16.4 MySQL-Datenbank

16.4.1 Allgemeines über diese Datenbank

MySQL ist eines der weltweitverbreitetsten relationalen Datenbanksysteme und bildet die Grundlage für viele dynamische Webauftritte.

16.4.2 Warum haben wir uns nicht für diese Datenbank entschieden?

Das Problem bei MySQL ist, dass MySQL eine Client-/Serveranwendung ist und deshalb eine weitere Software benötigt wird. Diese Software funktioniert nicht auf jedem Betriebssystem und deshalb ist MySQL für unser Projekt nicht geeignet.

16.5 SQLite-Datenbank

16.5.1 Allgemeines über diese Datenbank

SQLite ist eines der bekanntesten und weitverbreitetsten Datenbanksysteme weltweit.

16.5.2 Warum haben wir uns für diese Datenbank entschieden?

SQLite ist keine Client-/Serveranwendung wie z.B. MySQL es handelt sich vielmehr um eine Programmbibliothek, die direkt in eigene Anwendungen integriert werden kann. Die gesamte Datenbank befindet sich in einer einzigen Datei und kann von jedem Betriebssystem gelesen werden. Genau aus diesem Grund haben wir uns in unserem Projekt für die SQLite-Datenbank entschieden.

16.6 AVR-NET-IO-Board und K8IO-Relaiskarte

16.6.1 Allgemeines über das AVR-NET-IO-Board

Das AVR-NET-IO-Board ist eine Experimentier- und Lernplatine, zudem ist sie auch eine vollständig unabhängig arbeitende Ethernetplatine, die mit einem Netzwerkcontroller und einem ATmega32 Microcontroller ausgestattet ist. Das AVR-NET-IO-Board wird in unserem Projekt zum Einlesen von Spannungen an Temperatursensoren sowie Lichtschranken verwendet. Außerdem können wir mithilfe des AVR-NET-IO-Boards die K8IO-Relaiskarte ansteuern und somit beispielsweise einen Motor ein- und ausschalten.

16.6.2 Allgemeines über die K8IO-Relaiskarte

Die K8IO-Relaiskarte verfügt über 8 Relais zum Schalten von diversen Geräten. Anschließbar ist sie über den seriellen Port eines PCs oder, wie in unserem Fall, über den seriellen Port des AVR-NET-IO-Boards.

16.6.3 Warum haben wir uns für diese Hardware entschieden?

Herr Fachlehrer Gruber hat uns diese Hardware empfohlen und da für unsere Diplomarbeit Netzwerkfähigkeit erforderlich ist, ist diese Hardwarekombination optimal.

16.7 Verstärkerschaltung

16.7.1 Allgemeines über die Verstärkerschaltung

Zur genaueren Bestimmung der Temperatur können wir den Spannungsabfall am Temperatursensor verstärken.

16.7.2 Warum haben wir uns nicht für diese Möglichkeit entschieden?

Die Verstärkerschaltung wurde aufgebaut und hat für kurze Zeit funktioniert. Aus uns nicht bekannten Gründen funktionierte sie nach erneutem Aufbauen jedoch nicht mehr und wir gaben uns mit der nicht so genauen Methode zufrieden.

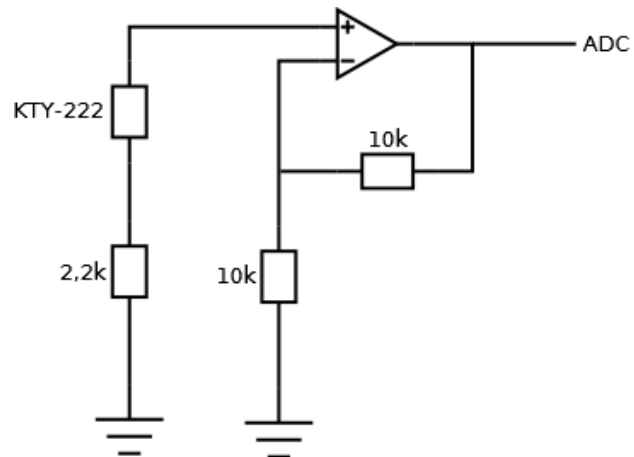


Abbildung 57: Verstärkerschaltung

16.8 Diodenschaltung

16.8.1 Allgemeines über die Diodenschaltung

Eine Diode in entgegengesetzter Richtung besitzt einen sehr großen Widerstand. Wir haben versucht die Referenzspannung (AREF) des AVR-NetIO-Boards zu minimieren und dadurch einen Genauigkeitsgewinn bei der Temperaturmessung zu erlangen.

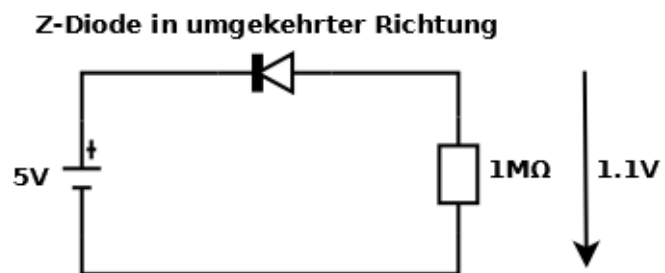


Abbildung 58: Diodenschaltung

16.8.2 Warum haben wir uns nicht für diese Möglichkeit entschieden?

Nach dem Aufbauen und erfolgreichen Minimierung der Referenzspannung von 5 auf 1,1 V haben wir erneut die Genauigkeit des Messaufbaus überprüft. Es konnte aus uns nicht bekannten Gründen keine Steigerung der Genauigkeit festgestellt werden.

17 Nennenswerte Probleme

Erwartungsgemäß sind auch bei der Realisierung unseres Projektes Probleme aufgetreten. In einem ersten Schritt haben wir das jeweilige Problem analysiert, Lösungsvarianten diskutiert und zu guter Letzt dann doch die Problemlösung gefunden und umgesetzt.

17.1 Temperaturmessung

Es war unser erklärtes Ziel, die Genauigkeit der Temperaturmessung zu verbessern. Obwohl dieses Problem keine unmittelbaren Auswirkungen auf unsere Diplomarbeit hatte, haben wir doch mehrere Möglichkeiten ausprobiert und dokumentiert. Die Variante der Vergrößerung des Messbereiches durch eine Regulierung der Betriebsspannung wurde nicht weiter verfolgt.

Bemerkung: Ein besserer Temperatursensor hätte das Problem rasch gelöst, jedoch haben wir – nicht zuletzt aus Kostengründen – diesen Lösungsansatz fallen gelassen.

17.1.1 Diodenschaltung

Der Versuch, mit einer Diodenschaltung die Referenzspannung so weit zu regulieren, dass im Ergebnis genauere Temperaturmessungen erreichbar waren, hat leider nicht das gewünschte Ergebnis gebracht und wurde deshalb nicht weiter verfolgt.

17.1.2 Verstärkerschaltung

Auch den Lösungsansatz, mit einer Verstärkerschaltung die Messspannung zu verstärken und damit die Messgenauigkeit zu verbessern, haben wir intensiv ausprobiert. Obwohl das Internet einige Schaltbilder angeboten hat, hat vorerst keine der von uns aufgebauten Schaltungen funktioniert. Mit weiteren Versuchen haben wir zwar eine 2-fache Verstärkung (1.2V – 1.8V) erreicht, doch das angestrebte Ziel der 5-fach-Verstärkung war auch mit der Änderung der Widerstände außer Reichweite. Weil die Schaltung nicht funktionieren wollte, haben wir Herrn FL Gruber befragt. Er hat uns darauf hingewiesen, dass eine 5-fache Verstärkung bei 5V Betriebsspannung nicht umsetzbar sei. Beim Messen der 5V Betriebsspannung ist ein weiteres Problem aufgetreten.

17.1.3 Messaufbau

Als wir die 5V Betriebsspannung messen wollten, hat das Multimeter 5.84V angezeigt. Auf Befragung des Herrn FL Gruber hat dieser uns informiert, dass es sich bei diesen Werten nicht mehr um Toleranzen handeln kann, da solche maximal 10% betragen würden. Zunächst haben wir vermutet, dass das AVR-NET-IO Board defekt sein könnte oder die Bauteile nicht ordnungsgemäß funktionieren würden. Dieser Verdacht hat sich jedoch nicht bestätigt, weil Herr FL Gruber festgestellt hat, dass die Masse falsch gesteckt worden ist und deswegen noch 0,84V von einer anderen Quelle dazugekommen sind.

17.2 Apache und Twitter

Beim Umsetzen unserer Idee, für allfällige Benachrichtigungen auch Twitter zu verwenden, wurde eine weitere Besonderheit offenkundig. Die Twitter-Benachrichtigungen funktionieren alleine ganz normal, doch bei Verwendung in unserem Projekt traten zwischen Apache Tomcat und Twitter-Benachrichtigungen Fehler auf. Das Problem war, dass die Benachrichtigung und der Tomcat den gleichen Port – nämlich den Port 80 – verwenden.

17.3 Apache - SQLite Datenbankpfad

Um die SQLite Datenbank deploy-fähig zu machen, mussten wir diese im mithilfe des Apache Tomcat Servers „mappen“. Man versteht darunter das Zugänglichmachen der Datenbank über den Server. Aufgrund der verschiedenen System und Pfadstrukturen wäre ein relativer Pfad zur Datenbank von großem Vorteil, jedoch fand der Server die Datenbank durch einen relativen Pfad nicht. Nach mehreren Versuchen den Pfad vielleicht zu berichtigen und trotz Hilfe von Prof. Greinöcker konnte das Problem nicht behoben werden. Prof. Greinöcker hat uns dann geraten, doch einen absoluten Pfad zu verwenden. Dadurch entsteht dann die Herausforderung, den ganzen und korrekten absoluten Pfad für jedes System zu bestimmen.

17.4 Apache - Grafik

Um unsere Temperaturgraphik auf dem Webinterface zu aktualisieren, haben wir einfach die in das JSP hereinzuladende Graphik überschrieben. Dadurch sollte der Server bei Neuladen (oder nach 60 sec.) die Graphik aktualisieren - jedoch machte er dies nicht. Unser erster Verdacht war, dass Tomcat einen Cache besitzt, der das Bild speichert und dadurch nicht aktualisiert. Nach langer Zeit sind wir aber auf das eigentliche Problem gestoßen: Wir haben aus Versehen das Bild im JSP mit dem Tag <iframe> anzeigen lassen. Nach Verändern des Tags <iframe> auf funktionierte die Aktualisierung einwandfrei.

17.5 Socketmanager

Am Anfang unseres SocketManagers kam es von Zeit zu Zeit zu Fehlern, da er manche Befehle falsch oder gar nicht ausgeführt hat. Nach weiterem Behandeln des Problems fiel uns jedoch auf, dass wir für das Senden der Befehle 'print' verwendet hatten. Dadurch schickt er die Befehle zwar zum Board, jedoch aufgrund der fehlenden Bestätigung kam es zu Fehlern. Dieses Problem wurde durch den Befehl 'println' gelöst, der die Befehle abschickt und danach einen Zeilenvorsprung setzt, um zu bestätigen.

17.6 JFreeCharts

Um die Daten für die Graphik bereitzustellen hatten wir vor, diese in Echtzeit vom Sensor in die Graphik einzutragen. Nach einigen Tests stellten wir jedoch fest, dass JFreeCharts dies nicht unterstützt. Die Lösung war einfach: Wir erstellten eine JFreeCharts-Grafik mithilfe der Daten aus der Datenbank und ließen diese als „ObjectStream“ im Webinterface anzeigen.

18 Erkenntnisse aus der Diplomarbeit

Im Zuge der Diplomarbeit wurden viel Erfahrung und Wissen im Umgang mit diversen Programmier- und Skriptsprachen sowie durch die technischen und auch elektrotechnischen Problemlösungen gesammelt.

In folgenden spezifischen Gebieten wurden im Laufe unserer Diplomarbeit Erfahrungen gesammelt:

- **Informatik**

- Java
 - Aufbauen einer Socketverbindung
 - Aufbauen einer Datenbankverbindung
 - Parsen von Zeichenketten
 - Verwenden von Servlets und JSPs
 - Zeitgesteuerte Prozesse
 - Senden von E-Mails
 - Umgang mit JFreeCharts
- Datenbanken
 - Kennenlernen verschiedener Datenbanken
 - Umgehen mit erweiterter SQL-Syntax
- HTML
 - Erlernen neuer Befehle
- AJAX
 - Verwendung dieser Technologie zur Verbesserung der Benutzerfreundlichkeit

- **Elektronik**

- Messen und Verarbeiten von Sensorwerten
 - Umrechnen eines Spannungswertes in einen Temperaturwert
- Erstellen von Schaltungen
 - Auswahl der passenden Hardware
 - Komprimieren und Vereinfachen von Schaltungsentwürfen
- Löttechnik

- **Fertigungstechnik**

- Konstruktion mithilfe der Konstruktionssoftware „Pro Engineer Wildfire 4“
- Umsetzung eines fertigungstechnischen Problems
 - Von der Planung und der Konstruktion bis zur eigentlichen Fertigung

19 Ausblick

In der vorliegenden Dokumentation haben die Diplomanden einige Möglichkeiten aufgezeigt, wie eine zuverlässige, alltagstaugliche Fischfütterung im Rahmen kleiner Aufzuchtmaßnahmen mit dem Einsatz individueller Computer-Software sinnvoll gesteuert werden kann. Aus den nunmehr gewonnenen Erkenntnissen geben die Diplomanden einen Ausblick, wie sich die Aufzucht von Fischen in der Zukunft noch einmal weiterentwickeln könnte.

Zuchtbeckenüberwachung mittels Webcam

Eine an der Decke montierte Videokamera legt ihren „Fokus“ auf das Zuchtbecken und liefert „rund um die Uhr“ lückenlose Bilder über die „Geschehnisse“ in einem Zuchtbecken (Ansammlung von toten Jungfischen an der Wasseroberfläche, lethargisch wirkende Jungfische, die kein Futter mehr aufnehmen deuten u.U. auf Fischkrankheiten hin, usw.). Weil die Kamera beweglich montiert ist - Schwenken, Neigen, Zoomen - können auch einzelne Bildausschnitte aus dem Zuchtbecken betrachtet werden. Die Kamera sendet diese Bilder über das Netzwerk an einen PC, wo der User die notwendigen Entscheidungen treffen kann oder auf Programm-Warnungen sofort reagieren kann.

Nächtliche Bruthaus-Beleuchtung

Mit der Verwendung von künstlichem Licht kann in Bruthäusern ein Bestand an Eiern früher erzeugt werden. Man geht im Allgemeinen davon aus, dass sich Eier so früh ausbrüten lassen, dass dadurch zusätzliche Zeit für das Wachstum gewonnen wird. Außerdem können Futterkosten eingespart werden. Deshalb wird die Lichtversorgung des Bruthauses an ein PC-Programm angeschlossen. Diese Software ist für die durchgehende Beleuchtung der Zuchträume verantwortlich, wenn kein Tageslicht mehr verfügbar ist.

Auswertung der Futterdaten

Ein modernes Managementprogramm unterstützt den Fischzüchter – liefert z.B. genaue Aufzeichnungen und Auswertungen über Futterverbrauch (rechtzeitiges Nachbestellen von Futtermitteln) und Futtervorratshaltung.

20 Zusammenfassung, Ergebnis

In den letzten Jahren hat der Einsatz von Computer-Programmen in großen, professionellen Fischzuchtbetrieben deren Wirtschaftlichkeit revolutioniert. Die Installation solcher Software bleibt allerdings aufgrund der hohen Kosten nur außerhalb Österreichs auftretenden „Fischmastbetrieben“ vorbehalten. Unter günstigen Rahmenbedingungen können sich aber die auf den ersten Blick hohen Investitionskosten schon in ein paar Jahren amortisieren. Derartige Fütterungsanlagen arbeiten computergesteuert vollautomatisch.

Für klein strukturierte Betriebe und für die Aufzucht von Fischen im Rahmen wissenschaftlicher Projekte – beispielhaft wird auf die laufenden Projekt-Arbeiten in der Fischzuchtanstalt Thaur verwiesen - sind solche computer-gesteuerten Anlagen aus Kostengründen unerschwinglich. Trotzdem muss auch bei solchen Aufzuchtmaßnahmen mit den vorhandenen Ressourcen sparsam umgegangen werden, weil im Einzelfall nur kostendeckende Budgets bereitgestellt werden. Im Rahmen dieser Diplomarbeit kamen die Diplomanden zur Erkenntnis, dass sich auch kleine Aufzuchtmaßnahmen mit individueller Computer-Software steuern lassen. Dafür wurde für diese Diplomarbeit ein Prototyp eines Futterautomaten geschaffen, der speziell im Bruthaus eingesetzt werden kann, wo in vielen Fütterungsdurchgängen möglichst kleine Futterportionen verabreicht werden müssen, damit das hochwertige und teure Brutfutter so optimal wie möglich verwertet wird. Die Verantwortlichen dieser Fischzucht kennen die hohen Anforderungen, die notwendig sind, um eine Effizienzsteigerung durch die Fütterung zu erreichen. Diese ließ sich aber bisher nur durch persönlichen Ehrgeiz und viel uneigennütziges Engagement realisieren. Die automatische Fütterung würde menschliche Arbeitszeit sparen, Personalkosten reduzieren und die Tageszunahmen der Jungfische beschleunigen, weil auch an Wochenenden, Feiertagen und während der Urlaubszeit computer-gesteuert gefüttert werden könnte. Eine Weiterentwicklung dieser Software könnte auch genaue Aufzeichnungen und Auswertungen über den Futterverbrauch (Futternorratshaltung) liefern. In regelmäßigen Zeitabständen könnte auch die Wassertemperatur in den Zuchtbecken (spielt im Verdauungsvorgang der Jungfische eine Rolle) und die Raumtemperatur im Bruthaus gemessen werden – allfällige Störungen außerhalb des „Normbereiches“ würden den Verantwortlichen kommuniziert (→ Internet-Überwachung). Über die laufenden Kontrollen könnten genaue Aufzeichnungen protokolliert und präzise Auswertungen bereitgestellt werden. Zuchtbecken-Überwachung mittels Webcam und vollautomatische, nächtliche Bruthaus-Beleuchtung könnten in einer Weiterentwicklung der entwickelten Software den Zuchterfolg zusätzlich absichern.

Der Computer ist heute am Arbeitsplatz und in vielen anderen Bereichen selbstverständlich geworden. Nach Ansicht der Diplomanden könnten auch kleinstrukturierte Fischzuchtbetriebe von der „digitalen Revolution“ mit vergleichsweise geringen finanziellen Mitteln profitieren. Mit der Vernetzung von Computern würden sich für die Anwender darüber hinaus auch Möglichkeiten eröffnen, den Kostenfaktor „Arbeit“ zu entlasten, weil dringende Aktivitäten nicht mehr unbedingt „vor Ort“ verrichtet werden müssten.

Danksagung

Für die Unterstützung bei unserer Diplomarbeit möchten wir uns ganz herzlich bei folgenden Personen bedanken:

- **Herrn Fachlehrer Engelbert Gruber (HTL-Innsbruck, Anichstraße)**
für die vielen hilfreichen Anregungen, die konstruktive Kritik, die freundliche Bereitstellung von Arbeitsmaterial und Hilfsmitteln und sein umsichtiges Begleiten;
- **Herrn Professor DI Dr. Albert Greinöcker (HTL-Innsbruck, Anichstraße)**
für die vielen wertvollen Tipps im Umgang mit dem Apache Tomcat Server und die Kontrolle/Prüfung der programmtechnischen Umsetzung der Diplomarbeit;
- **Herrn Mag. Nikolaus Medgysey (Universität Innsbruck)**
für seine interessanten Ausführungen und Denkanstöße anlässlich des „Tages der Offenen Tür“ in der Fischzucht Thaur (August 2011);
- **unseren Eltern**
für die Unterstützung in den letzten fünf Jahren, die uns die Ausbildung an der HTL-Anichstraße erst ermöglicht haben und die finanzielle Unterstützung zur Anschaffung der Arbeitsmaterialien für die Umsetzung der Diplomarbeit.

Abbildungsverzeichnis

Abbildung 1: Bruthaus mit mechanischen Futterautomaten in der Fischzuchtanlage Thaur ...	6
Abbildung 2: Zuchtbecken in der Fischzuchtanlage Thaur	6
Abbildung 3: Java-Logo.....	15
Abbildung 4: Temperaturkurve des Sensors	17
Abbildung 5: JFreeCharts - Beispiel	19
Abbildung 6: Konzept – „DBManager“	20
Abbildung 7: Konzept - "Fetch"	21
Abbildung 8: Konzept - "Feed"	22
Abbildung 9: Konzept - "MainThread"	23
Abbildung 10: Datenbank-Servlet	31
Abbildung 11: Apache-Logo	32
Abbildung 12: Apache-Kommandozeile	33
Abbildung 13: Eclipse-Logo	34
Abbildung 14: Setzen des Arbeitsverzeichnisses	35
Abbildung 15: Benutzeroberfläche	36
Abbildung 16: Java-Perspektive	36
Abbildung 17: MySQL-Logo	37
Abbildung 18: SQLite-Logo	38
Abbildung 19: Login.....	40
Abbildung 20: Setup	41
Abbildung 21: Home-Tab	42
Abbildung 22: Graph-Tab	43
Abbildung 23: Tabelle-Tab	44
Abbildung 24: Logs-Tab	44
Abbildung 25: Konfiguration-Tab	45
Abbildung 26: AJAX-Modell.....	46
Abbildung 27: AJAX im Webinterface	47
Abbildung 28: Github-Logo	48
Abbildung 29: Github-Befehle.....	50
Abbildung 30: "Account Settings"	51
Abbildung 31: SSH-Public-Key	52
Abbildung 32: "Account Settings"	53
Abbildung 33: API-Token	53
Abbildung 34: "New repository"	54
Abbildung 35: "Create repository"	55
Abbildung 36: Sensorschaltung.....	58
Abbildung 37: LM324	59
Abbildung 38: Verstärkerschaltung.....	59
Abbildung 39: Diodenschaltung	60
Abbildung 40: Komplette Schaltung.....	61

Abbildung 41: AVR-NET-IO-Board	62
Abbildung 42: K8IO-Relaiskarte.....	64
Abbildung 43: Funktionsweise eines Relais	64
Abbildung 44: Temperaturkurve des KTY-222	65
Abbildung 45: LM324-Verstärkungs-IC	65
Abbildung 46: EESX-671	66
Abbildung 47: Streifenrasterplatine.....	67
Abbildung 48: Lego.....	67
Abbildung 49: Futterbecher	68
Abbildung 50: Ausgleichgewicht	68
Abbildung 51: Galgen	68
Abbildung 52: Kompletter Aufbau 1	69
Abbildung 53: Kompletter Aufbau 2	69
Abbildung 54: Anmeldung am Webinterface.....	71
Abbildung 55: Box mit Förderband	73
Abbildung 56: Rohr.....	74
Abbildung 57: Verstärkerschaltung.....	76
Abbildung 58: Diodenschaltung	76

Tabellen- und Formelverzeichnis

Tabelle 1: Vor- und Nachteile von Java	15
Tabelle 2: Ermittlung des Widerstandswerts	57
Tabelle 3: Materialkosten	66
Formel 1: Formel zur Berechnung der ADC-Spannung	17
Formel 2: Beispielhafte Berechnung einer ADC-Spannung.....	17
Formel 3: Geradengleichung	18

Anlagenverzeichnis

Anlage 1: AVR-NET-IO Datenblatt („D810073B.PDF“) auf CD

Anlage 2: K8IO Datenblatt („D710722B.PDF“) auf CD

Anlage 3: KTY-222 Datenblatt („KTY81-2SERIES_4.pdf“) auf CD

Anlage 4: EESX-671 Datenblatt („ee-sx47_67_dsheetsheet_csm483.pdf“) auf CD

Anlage 5: Software „fishapp.zip“ auf CD

Literaturverzeichnis

Apache Tomcat. Abgerufen am 25. April 2012 von Wikipedia - Die freie Enzyklopädie:

http://de.wikipedia.org/wiki/Apache_Tomcat

AVR-NET-IO - Fertigmodul. Abgerufen am 25. April 2012 von Pollin Electronic:

http://www.pollin.de/shop/dt/Njl5OTgxOTk-/Bausaetze_Module/Bausaetze/AVR_NET_IO_Fertigmodul.html

Eclipse (IDE). Abgerufen am 25. April 2012 von Wikipedia - Die freie Enzyklopädie:

[http://de.wikipedia.org/wiki/Eclipse_\(IDE\)](http://de.wikipedia.org/wiki/Eclipse_(IDE))

Greenberg, D. B. (1966). *Forellenzucht*. Verlag Paul Parey.

JFreeChart. Abgerufen am 25. April 2012 von Wikipedia - Die freie Enzyklopädie:

<http://de.wikipedia.org/wiki/JFreeChart>

Relais. Abgerufen am 25. April 2012 von Wikipedia - Die freie Enzyklopädie:

<http://de.wikipedia.org/wiki/Relais>