FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Arduino im Informatikunterricht

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Magister der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Masterstudium Informatikmanagement

eingereicht von

DI (FH) Thomas Greiner
Matrikelnummer 0727314, E066922

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuerin: Ass. Prof. Dr. Monika DI Angelo

Wien, 01.06.2012

(Unterschrift Verfasser) (Unterschrift Betreuerin)

Eidesstattliche Erklärung

DI (FH) Thomas Greiner
Guglgasse 8/2/102
1110 Wien

„Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Wien, am 1. Juni 2012

Thomas Greiner

Danksagung

Mein Dank gilt all jenen Personen, die zur Entstehung dieser Arbeit in fachlicher sowie persönlicher Form beigetragen haben.

Ein besonderes Dankeschön möchte ich *Alexandra* aussprechen – sie hat mir den nötigen Freiraum überlassen.

Weiters möchte ich meiner Betreuerin *Monika* ein großes Lob für die hervorragende und vorbildliche Unterstützung aussprechen.

Vielen Dank!

Kurzfassung

Die vorliegende Arbeit befasst sich mit der Entwicklung bzw. Konzipierung eines Lehrbuches zum Thema „Arduino“, welches speziell für den schulischen Alltag und deren Anwendungsgebiete optimiert ist. Die Hauptaugenmerke sind hier ein strukturierter Aufbau für die schulische Verwendung sowie die Möglichkeit zum selbständigen Erarbeiten gewisser Themenbereiche.

Arduino ist ein kostengünstig entwickelter Microcontroller, der seinen Ursprung in einer Universität in Italien hat. Diverse Anwendungen, welche von Design- und ElektrotechnikstudentInnen konstruiert wurden, verlangten nach einer Steuerung „von Außen“ und somit einer Sichtbarmachung für die Außenwelt – die elektronische Platine „Arduino“ war die Antwort.

Das Arduino-Projekt ist als Open Source verfügbar – Projekte auf Basis des Arduinos können somit ohne Lizenzprobleme verwendet werden. Die verfügbare Dokumentation im Internet und die riesige Anhängerschaft dieses Projektes bieten Hilfestellungen in fast allen auftretenden Fragen, welches ein großer Vorteil der Open Source-Lizenzierung ist.

Mit dem Arduino wird es möglich, „Technik greifbar“ zu machen. Der explorative Zugang zur Elektronik bzw. zur Programmierung ist gerade zu perfekt für den schulischen Alltag. Speziell beim Programmieren kann jede kleine Änderung rasch visualisiert werden (z. B. mit Hilfe von LEDs).

Das Ergebnis der Arbeit ist ein „Lehrbuch für Arduino“. Die Evaluation des Lehrbuches wurde durch ein Sommerpraktikum an der TU Wien ermöglicht. Hierbei lernten PraktikantInnen mit Hilfe des Lehrbuches die Thematik „Arduino“ kennen, sowie diverse Projekte selbständig umzusetzen.

Abstract

This thesis presents a textbook for the electronic device „Arduino". The textbook should be specially developed for schools. Structural design and the possibility to self-study of some special topics should be main parts of the book.

Arduino is a cost-efficient developed microcontroller, which has its seeds in a university in Italy. Design- and electronical-students developed the Arduino. They had particular cases for which they need some reaction with the environment – the electronic device „Arduino" was the answer.

The Arduino-Project is Open Source, so projects created with the Arduino can be used without licencing problems. The big available documentation on the Internet and the huge community around the project can help in nearly all occuring questions, which are big advantages of the Open Source licencing policy.

„Living technology" can be realised with the helping hand of Arduino. The explorative entry to electronic systems respectively to coding / programming is perfect for schools. Especially examples in coding can be made visible within seconds (LEDs on/off, LEDs blinking, …).

The result of the thesis ist the textbook for Arduino. Students should use and review the book in a summer training course at the Vienna University of Technology (TU Vienna). They should use the book to get closer to the theme of „Arduino" and finally they should realise projects by there own.

Inhaltsverzeichnis

1. Einleitung

Im schulischen Alltag wird sehr viel Wert auf theoretisches Grundlagenwissen gelegt – meist wird die Vermittlung der Theorie über das Verständnis und der praktischen Anwendung gestellt. Dieses Defizit in der Vermittlung von Wissen und des Anwendens von Wissen soll mit Hilfe des Arduino-Mikrocontrollers minimiert bzw. beseitigt werden.

Arduino bezeichnet eine Microcontroller-Familie, welche der Universität IDII[1] entwickelt wurde, die sich in der Stadt Ivrea (Provinz Turin, Region Piemont) befindet.
Ein Gemeinschaftsprojekt zwischen Design- und Elektrotechnik-StudentInnen führte zur Entwicklung dieser kostengünstigen und einfach zu programmierenden Elektronikplatine. Die große Beliebtheit und die ständig wachsende Community bestätigen den riesigen Erfolg dieses Projektes.

Der Arduino [BAN08], [BRU10] ist eine Mikrocontroller-Plattform, bei der es mittlerweile viele verschiedene Varianten gibt. Der günstige Preis, die Quelloffenheit – der Mikrocontroller sowie die Software zur Entwicklung von Programmen steht frei zur Verfügung (siehe 2.1 Open Source) – und die einfache Handhabe sowie Programmierung haben dem Arduino-Projekt zum Erfolg und großer Anhängerschaft verholfen.

Abbildung 1, Arduino Uno

Die Entwicklung von Arduino-Projekten zielt auf MINT ab – MINT ist ein gebräuchliches Akronym für die Fachgebiete Mathematik, Informatik, Naturwissenschaften und Technik – und ermöglicht einen einfachen Einstieg in die Nutzung und Verwendung von Mikrocontrollern, die relativ einfach über eine JAVA-ähnliche Hochsprache programmiert werden können.

[1] IDII - Interaction Design Institut Ivrea [online], Verfügbar unter http://interactionivrea.org/en/index.asp [Zugang am 20. Jänner 2012]

1.1. Ziel der vorliegenden Arbeit

Im Rahmen dieser Arbeit wird ein Lehrbuch für SchülerInnen entwickelt, welches anschließend in einem Sommerpraktikum von einigen SchülerInnen evaluiert wird. Der Aufbau des Buches ist in Lehreinheiten unterteilt, welche wiederum einer Gliederung in drei Hauptteile – wie folgt – unterliegen:

1) Was ist von der jeweiligen Lehreinheit zu erwarten? Fragen aufwerfen
2) Theorie- und Praxisteil
3) Theoriefragen bzw. Übungen zum Vertiefen des Erlernten

Das erstellte Lehrbuch dient dem einfachen Einstieg in die Welt von Mikrocontrollern und der Programmierung und stellt das Arbeiten mit dem Arduino möglichst einfach dar. Es ist zum Selbststudium für SchülerInnen geeignet.

Die Zielgruppe ist die neunte bzw. zehnte Schulstufe, d. h. Oberstufe in Gymnasien bzw. erste und zweite Klasse von Höheren Berufsbildenden Schulen.

Nach kurzer Einführung wird anhand des Lehrbuches – hauptsächlich im Selbststudium – die Grundlagen der Elektronik und die Möglichkeiten der Programmierung des Arduino erarbeitet. Einfache Beispiele sollen das Verständnis von Mikrocontrollern und damit die eigene Kreativität und den Ideenreichtum fördern. Das Setzen von Aktionen und der daraus resultierenden Reaktionen in der Programmierung des Mikrocontrollers soll das spielerische sowie kognitive Lernen fördern.

Da das Konzept des Arduinos stark zum eigenen Explorieren anregt, soll das selbständige Umsetzen von eigenen Projekten ein erklärtes Ziel sein.

Zur Evaluation einer ersten Version des Handbuches werden SommerpraktikantInnen herangezogen, die in vier Wochen das Buch und sein Konzept testen.

Um eine qualitative Rückmeldung der Praktikantinnen und Praktikanten zu erhalten, wird wöchentlich eine Präsentation abgehalten. Diese sollen einen Rückblick über die vergangene Woche sowie ein Ausblick für die nächste Woche zum Inhalt haben. Außerdem werden aufgetretene Fragen eruiert und behandelt. Am Ende des Praktikums soll das entwickelte Lehrbuch mittels eines Fragebogen bewertet werden - neue Ideen bzw. Verbesserungsvorschläge sollen, so weit wie möglich, in die Lehreinheiten eingearbeitet werden.

2. Grundlagen

Eine zielgerichtete Übersicht sowie kurze Erklärung von grundlegenden Begriffen ist ein hilfreicher Einstieg in inhaltlich komplexe Themen.

Hier wird einerseits der Begriff Open Source näher definiert, da Arduino ein Open Source Projekt ist. Danach werden didaktische Grundlagen kurz umrissen, die für den Aufbau des Lehrbuches notwendig sind.

2.1. Open Source Software

2.1.1. Definitionen

Open Source Software beschreibt Software mit Quelloffenheit, d. h. Benutzer dürfen Einblick in den Quellcode haben, sowie diesen Quellcode verändern bzw. weitergeben.

Die Thematik Open Source verlangt durch ihre immer größer werdende Präsenz sowie der enormen Vielfalt an Ausprägungen von vorhandenen Definitionen nach einer ausführlichen Begriffsdefinition.

2.1.1.1. Free Software und Open Source Software

Die erste grundlegende Definition, die Free Software Definition, stammt von der Free Software Foundation (FSF). Sie bringt sie auf die folgende Kurzformel:

*„**Free software** is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:*

- *The freedom to run the program, for any purpose (freedom 0).*
- *The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.*
- *The freedom to redistribute copies so you can help your neighbor (freedom 2).*
- *The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this."*[2]

Der entscheidende Unterschied zwischen Freier Software und herkömmlicher Software besteht damit in der umfassenden urheberrechtlichen Nutzungsbestimmung, d. h. freies Kopieren, Bearbeiten, Untersuchen und Verbreiten wird durch Freie Software ermöglicht.

Das heißt aber noch nicht, dass Freie Software „nicht-kommerziell" oder stets kostenlos wäre. Die erwähnten Freiheiten gestatten ausdrücklich die kommerzielle Betätigung mit Freier Software (z. B. den Verkauf) – allein Lizenzgebühren sind ausgeschlossen, da sie den freien Zugang zu der Software verhindern [JAE06].

Der Begriff „Frei" bezieht sich also nicht auf den Kostenaspekt, wie das englische „free" in der Bedeutung „gratis" bzw. „kostenlos" vermuten lässt, sondern auf die genannten Freiheiten, die Nutzer und Programmierer an der Software haben.

[2] The Free Software Foundation [online], Verfügbar unter http://www.fsf.org/ [Zugang am 19. März 2011]

Der Begriff Freie Software, englisch „free software“, existiert seit Mitte der achtziger Jahre. Zunächst war „free software“ der allgemein gebräuchliche Ausdruck, um diesen speziellen Softwaretyp zu beschreiben.

Aufgrund von Berührungsängsten der Softwareindustrie mit diesem Begriff – man befürchtete eine Assoziation mit der Bezeichnung „gratis“ und somit des Verschenkens, was natürlich ein Feindbild in der Geschäftstätigkeit darstellte – suchte man einen neuen moderneren Ausdruck, um die Bedeutung der Freien Software auch im Handel und Verkauf etablieren zu können.
Deshalb fanden sich am 3. Februar 1998 in Paulo Alto in Kalifornien eine Gruppe von Pionieren der Freien Software Szene zusammen und gründeten die „Open Source Initiative“ - es wurde beschlossen, ab diesem Zeitpunkt den Ausdruck „**Open Source Software**“ zu verwenden. Der Begriffswechsel hatte durchschlagenden Erfolg.[3]

Viele namhafte Firmen, wie IBM, Sun Microsystems oder Oracle, nutzten diesen Begriffswechsel und entwickelten ebenfalls Programme unter dem Pseudonym „Open Source“. Kritiker, darunter der Urvater der Freien Software Bewegung Richard Matthew Stallman, befürchteten, dass der Grundgedanke von „free software“ verloren gehen würde und auch „unfreie“ Programme als Open Source Programme bezeichnet werden. Die Folge waren die diversesten Open Source Lizenzen – die bekanntesten werden später erläutert.

Umfassende Erklärungen der Schlagwörter sowie deren Geschichte und weiterführende Grundlagen findet man unter:

- Free Software
  - http://www.fsf.org/ [Zugang am 19. März 2011]
- Open Source
  - http://www.opensource.org/ [Zugang am 19. März 2011]

Interessante freie Projekte und diverse Sammlungen von Open Source Programmen findet man unter:

- Open Source Technology Group (OSTG)
  - http://openmagazine.net/ [Zugang am 20. März 2011]

2.1.1.1.1 Copyleft und Non-Copyleft

Alle Open Source Lizenzen müssen den oben genannten Definitionsmerkmalen der Freien Software entsprechen, dennoch gibt es erhebliche Unterschiede zwischen vorhandenen Lizenzen.

Es ergeben sich grundsätzliche Gegensätze in der Einschätzung, was „Freiheit“ bedeutet, nämlich ob es der freien Entscheidung des Bearbeiters Freier Software überlassen sein sollte, wie er das weiterentwickelte Programm lizenziert, oder ob er durch die Lizenz gezwungen werden soll, solche Weiterentwicklungen ebenfalls nur als Open Source Software zu vertreiben.

„Copyleft“ sind Klauseln in Open Source Lizenzen, die verhindern, dass geänderte Programme – die als Grundlage eine Freie Software haben – „proprietär“ vertrieben werden können. Sie können somit nicht „unfrei“ werden.

„Non-Copyleft“ sind Klauseln in Open Source Lizenzen, die solche Schutzbedingungen nicht enthalten. Es wird damit ermöglicht, Software, deren Grundlage eine Freie Software ist, in eine eigene „proprietäre“ Form umzuwandeln und unter einer eigenen Lizenz weiter zu vertreiben.

[3] Open Source [online], Verfügbar unter http://www.opensource.org/history [Zugang am 20. März 2011]

2.1.1.1.2 Abgrenzung zu Public Domain Software, Freeware, Shareware und Shared Source Software

Im Folgenden werden Lizenzmodelle [JAE06], die mit dem Open Source Modell oft verwechselt und zu Begriffsverwirrungen führen, erläutert.

1. **Public Domain Software**
   Public Domain Software ist ein “öffentliches Gut” (Übersetzung aus dem Englischen), das von dem Softwarehersteller bzw. –eigentümer kostenlos der Allgemeinheit zur Verfügung gestellt wird. Im Gegensatz zu Open Source muss der Quellcode von Public Domain jedoch nicht frei sein.
   Anmerkung
   Der vollständige Verzicht auf das Urheberrecht ist im kontinentaleuropäischen Recht wegen der persönlichkeitsrechtlichen Komponente aber nicht möglich. § 19 Abs. 2 UrhrG schließt dies explizit aus. Eine Public Domain kann in Österreich erst dann entstehen, wenn ein Werk nach Ablauf der urheberrechtlichen Schutzfrist von 70 Jahren post mortem auctoris „frei“ wird (§ 61 UrhrG).[4] Die Auslegung nach österreichischem Schutzbereich kann als Einräumung eines einfachen Nutzungsrechts an alle verstanden werden. Die Urheberpersönlichkeitsrechte bleiben aber beim Urheber.

2. **Freeware**
   Freeware bezeichnet die kostenfreie Überlassung der Software. In der Regel liegt der Freeware kein Quellcode bei, meist wird die Veränderung ausdrücklich verboten. Zudem kann die Nutzung vom Ersteller beschränkt werden – z. B. die Software darf nicht für den kommerziellen Gebrauch verwendet werden. Freeware gehört damit zur proprietären Software.

3. **Shareware**
   Als Shareware wird eine Verkaufsstrategie bezeichnet, die den Nutzer über eine beschränkte Zeitspanne, das Programm kostenlos zur Verfügung stellt - auch bekannt unter dem Prüf-vor-Kauf-System. Nach Ablauf der Probezeit wird dem Benutzer technisch verhindert, das Programm weiter zu testen, wenn nicht gegen Zahlung eines Entgelts eine Freischaltung erkauft wird. Shareware fällt deshalb in die Kategorie der herkömmlichen kommerziellen Software.

4. **Shared Source Software**
   Mit dem zunehmenden Erfolg von Open Source Modellen und dem zunehmenden Wunsch aller Kunden, Einblick in den Quellcode zu erhalten, führte Microsoft 2001 das Shared Source Lizenz-Modell ein. Das Microsoft-Betriebssystem für den Embedded-Bereich, Windows CE, wurde unter dieser Lizenz an die Öffentlichkeit gebracht. Der Kern dieses Modells sieht vor, Kunden dieser Produkte ein Nutzungsrecht zu verschaffen, jedoch ist bei Weitervertrieb dieser nur unter Zahlung von Lizenzgebühren an Microsoft möglich.

[4] Bundeskanzleramt Österreich – Rechtsinformationssystem [online], Verfügbar unter http://www.ris.bka.gv.at/ [Zugang am 20. März 2011]

2.1.1.2. Open Source Lizenzen

Die Definition zu Freier Software bzw. Open Source Software lässt einen weiten Spielraum für die Lizenzgestaltung. Entsprechend den unterschiedlichen Bedürfnissen der Lizenzgeber ist eine Vielzahl an Lizenzen entstanden [JAE06].

In den folgenden Abschnitten werden in groben Zügen die wichtigsten und bekanntesten Lizenzen vorgestellt.

2.1.1.2.1 Lizenzen mit strenger Copyleft-Klausel

Lizenzen mit strenger Copyleft-Klausel verlangen, dass sämtliche Bearbeitungen bei der Weitergabe der Ursprungslizenz zu unterstellen sind.

1. **GNU GPL - General Public License**
   GNU bezeichnet ein rekursives Akronym und bedeutet **G**NU's **n**ot **U**nix. Es wurde von Richard Matthew Stallmann erschaffen und entstand dem GNU-Projekt, welches die Entwicklung eines unix-ähnlichen freien Betriebssystems zum Ziel hatte.

   Die GPL wurde ebenfalls von Richard Matthew Stallmann, dem Gründer von Free Software Foundation (FSF), geschaffen und ist die weitverbreitetste freie Lizenz. Sie liegt in der aktuellen Version 2 seit 1991 vor und gilt als „Mutter" zahlreicher Open Source Lizenzen. Einige wichtige freie Programme sind der GPL unterstellt, darunter wesentliche Teile des Betriebssystems GNU/Linux.
   Zur Zeit steht die 2. Ausgabe der Version 3 der GNU – Lizenz zur Diskussion – nähere Informationen dazu unter http://www.gnu.de/documents/gplv3.de.html [Zugang am 27. März 2011].

2. **CPL - Common Public License**
   Die CPL ist eine von IBM für eigene Open Source Programme erstellte Copyleft-Lizenz, die der IBM Public License (IPL) nachgefolgt ist. Eine besondere Bedeutung erlangt die CPL durch die Entwicklungsumgebung Eclipse – seit September 2004 wird Eclipse unter EPL (Eclipse Public License) vertrieben. Sie ist nahezu identisch mit der CPL, jedoch wird der Eclipse Foundation anstatt IBM die Kontrolle über Änderungen des Lizenztextes überlassen.
   Im Vergleich zur GPL ermöglicht die CPL eine einfachere Kombination von Softwarebestandteilen unter verschiedenen Lizenzbedingungen.

2.1.1.2.2 Lizenzen mit beschränkter Copyleft-Klausel

Lizenzen mit beschränkter Copyleft-Klausel lassen Ausnahmen von der Lizenzierungspflicht von Bearbeitungen zu.

1. **MPL – Mozilla Public Licence**
   MPL entstand aus einer Krise des Softwareherstellers Netscape. Die starke Konkurrenz im Internet Browser – Bereich durch den Internet Explorer von Microsoft zwang Netscape im Jahre 1998 zu handeln – sie legten den Source Code des Netscape Navigators frei, um das Programm künftig unter einer freien Lizenz im Rahmen des Mozilla Projects weiter zu entwickeln.
   Die Gründung der neuen eigenen Lizenz wurde damit vertreten, dass einerseits die GPL nicht verwendet werden konnte – Netscape Navigator stellte eine proprietäre Lösung dar und konnte somit nicht unter GPL weiterentwickelt werden – und andererseits fürchtete man, dass bei „non-Copyleft"-artigen Lizenzen diverse Weiterentwicklungen nicht an die Community unter freier Lizenz zurückgegeben werden. Somit entstand die MPL.

Der Netscape Navigator, der ab diesem Zeitpunkt (1. April 1998) den Projektnamen „Mozilla“ hatte, wurde in seiner ersten Version unter MPL im Juni 2002 veröffentlicht. Das Projekt kann als Erfolg angesehen werden, denn „Firefox“, der aus Mozilla ausgegliederte Web-Browser erfreut sich hoher Beliebtheit und macht nun seinerseits dem Internet Explorer einen spürbaren Marktanteil streitig.
Auch der Mailclient Mozilla Thunderbird und die Terminverwaltung Sunbird finden weite Verbreitung.

2. **LGPL - Lesser General Public License**
   Die GNU Lesser General Public License basiert in ihren wesentlichen Bestimmungen auf der GPL, sie wandelt diese jedoch im Hinblick auf die besonderen Rechtsfragen ab, welche sich aus der Nutzung von Softwarebibliotheken ergeben. Die aktuelle Version ist 2.1 aus dem Jahre 1999. Ursprünglich wurde sie als Library General Public License eingeführt, jedoch später umbenannt.
   Eine der wichtigsten Programmbibliotheken, die GNU C Library, steht unter der LGPL – sie wird mit dem freien Betriebssystem GNU/Linux vertrieben und soll ermöglichen, dass auch proprietäre Anwendungsprogramme, die auf die klassischen Bibliotheken zugreifen müssen, gemeinsam mit dem Betriebssystem vertrieben werden können.

2.1.1.2.3 Lizenzen ohne Copyleft-Klausel

Lizenzen ohne Copyleft-Klausel enthalten keine Pflichten für die Lizenzierung von neu hinzugefügtem oder geändertem Code.

1. **BSD Copyright - Berkeley Software Distribution**
   BSD steht für Berkeley Software Distribution, eine Unix-Variante, die seit Mitte der 70er Jahre an der Universität Berkeley entwickelt wird.
   Im Gegensatz zu anderen Open Source Lizenzen beschränken sich die BSD Copyright – Lizenzen auf einige wenige Sätze, da vor allem die Einräumung von Nutzungsrechten geregelt wird, Klauseln zum Copyleft sind jedoch nicht vorgesehen.

2. **Apache Software License**
   Die Apache Software License gibt es aktuell in drei Versionen: Version 1.0, Version 1.1 und Version 2.0. Sie unterscheiden sich nicht grundlegend, sondern nur in diversen Bestimmungen für Lizenznehmer bei der Nutzung der Software.
   Die Apache Webserver-Software wird seit Februar 1995 von der Apache Group entwickelt, seit 1999 hat die Apache Software Foundation die Organisation des Open Source Projekts übernommen.
   Apache ist eines der erfolgreichsten freien Software-Projekte – der Marktanteil bei Webserver-Software beträgt ca. 70 %.

2.1.2. Bekannte Mitbegründer

Das Open Source Projekt, dass in den letzten Jahren immer mehr zu einer großen Konkurrenz zu herkömmlicher kommerzieller Software wurde, hat sein Entstehen in der Zusammenarbeit von vielen hochmotivierten und klugen Köpfen – trotzdem treten auch hier Persönlichkeiten in den Vordergrund, welche diese Bewegung mitbegründeten und mitbeeinflussten [THE04].
Im Folgenden werden drei Personen dieser erlesenen Gruppe angeführt, um den abstrakten Begriff eine Persönlichkeit zu verleihen.

2.1.2.1. Richard Stallman

Richard Matthew Stallman wurde 1953 in New York geboren. Er studierte an der Harvard Universität und am MIT (Massachusetts Institute of Technology) im renommierten AI Lab (Artificial Intelligent Laboratory).
Seine erste große Errungenschaft bzw. sein erster großer Erfolg gelang ihm mit der Entwicklung des Texteditors Emacs, welchen er durch Anregungen und Änderungswünsche diverser Nutzer immer weiterentwickelte. Das erste Problem, dass Richard Stallman, mit geschützter Software hatte, sollte sogleich die Geburt der Lebensphilosophie von freier Software werden. Ein Druckertreiberproblem, welches er bei vorhandenem Quellcode hätte selbst beheben können, veranlasste ihn, das GNU Projekt ins Leben zu rufen. Ziel dieses ehrgeizigen Projektes war, ein freies Betriebssystem auf Basis von Unix zu schaffen. Anfangs bestand das GNU Projekt aus einer Sammlung von nützlichen Programmen wie dem GNU Emacs Editor oder dem bekannten GNU C-Compiler (GCC).
Aufgrund der großen positiven Resonanz aus seinem Umfeld, gründete er im Jahre 1985 die FSF (Free Software Foundation), die für den Schutz von freier Software sorgen soll. Unter anderem stammt die Klausel Copyleft sowie die Lizenz GNU GPL von der FSF.

Das GNU Projekt wurde im Laufe der Zeit immer erfolgreicher und populärer, obwohl das eigentliche Ziel, ein freies Betriebssystem zu schaffen, noch nicht erreicht wurde – es fehlte weiterhin das „Herz“ eines Systems, der sogenannte „Kernel“. Dieser sollte seinen Ursprung in Finnland haben, wie im folgenden Kapitel dargestellt wird.

2.1.2.2. Linus Torvalds

Linus Benedict Torvalds wurde am 28. 12. 1969 in Helsinki geboren. Er studierte Computerwissenschaften an der Universität von Helsinki, bei der er auch zum ersten Mal mit Unix – einem Betriebssystem – in Berührung kam. Die Faszination an diesem Betriebssystem veranlasste ihn, sich intensiv mit der Programmierung dieses Systems zu beschäftigen.
Konzeptionelle Grundlage von Linux – das von Linus entwickelte Betriebssystem – war Minix (das Wort setzt sich aus Mini Unix zusammen). Minix wurde als Grundkonzept für ein Betriebsystem an der Universität von Helsinki von Andrew S. Tanenbaum entworfen.
Im Jahr 1991 war eine erste freie Version von Linux verfügbar und mit der Zeit entwickelte sich eine eingeschworene Fan-Gemeinde, die Tester waren und Verbesserungsvorschläge machten – was einer Software in der Entwicklungsphase einen enormen Qualitätsmehrwert bietet. Linus stellte Linux unter die GNU GPL Lizenz und brachte die bereits etablierten Programme des GNU Projekts auf Linux zum Laufen – es entstand GNU/Linux. Mit der Entwicklung einer grafischen Benutzerschnittstelle (X-Window; vom MIT) sowie der Implementierung von TCP/IP (dem Defacto-Standard für Internetanbindungen) stand einem weiteren Erfolgslauf nichts mehr im Wege.

2.1.2.3. Bill Gates

Sollte sich eine gewisse Verwunderung verbreiten, warum Bill Gates in dieser Arbeit als einer der wichtigen „Mitbegründer“ genannt wird, ist es durchaus verständlich. „Mitbegründer“ deshalb, weil er indirekt die Open Source Bewegung als größter vorzeigbarer Konkurrent motiviert hat.
William Henry Gates III wurde im Jahre 1955 in Seattle geboren. Er gründete zu Beginn seiner Studienzeit in Harvard mit seinem Freund Paul Allen die Firma „Traf-O-Data“. Sie entwickelten Software für die ersten Minicomputer auf Basic-Basis.
1975 benannten sie die Firma in Microsoft um - MS Basic war eines der ersten Produkte dieser Softwareschmiede. Anfang der 80er Jahre kaufte Bill Gates ein Betriebssystem von der Firma Seattle Computer Products – SCP-DOS, welches er nach einigen Modifikationen in MS-DOS umbenannte.
Der große Durchbruch gelang mit der Weitergabe einer Lizenz des Microsoft DOS an IBM (International Business Machines), deren IBM-PC ein Verkaufsschlager wurde. Anfang der 90er Jahre gelang ein weiterer Durchbruch mit dem allseits bekannten Microsoft Windows.

Aus der negativen Einstellung gegenüber freier Software, machte Bill Gates kein Geheimnis. Sein bekannter „offener Brief an Hobbyprogrammierer“[5] aus dem Jahr 1976 deren Kernaussage ist – Hobbyprogrammierer könnten keine gute Software ohne Bezahlung realisieren – entpuppt sich mehr und mehr als die große Herausforderung für Entwickler von freier Software, diese Aussage zu widerlegen. Eines der besten Beispiele dafür ist das freie Betriebssystem GNU/Linux, welches zum größten Konkurrenten von Microsoft avanciert.

2.1.3. Unsicherheiten bei Open Source Produkten

Bei der Philosophie [HUE06] von freier Software und deren großer Bedeutung in der heutigen Wirtschaft trifft man immer wieder auf diverse Unsicherheitsfaktoren, die das Verwenden der Produkte unterbindet. Die häufig gestellten Fragen zum Thema - Unterstützung / Support sowie fehlende Weiterentwicklung - werden zum Abschluss dieses Kapitels neutral beleuchtet.

2.1.3.1. Unterstützung / Support

- *Wie sieht eine technische Unterstützung bei Open Source Produkten aus?*
- *Welchen Support kann man erwarten?*

Fakt ist, „verpflichtende“ Unterstützung bzw. Support – falls zum jeweiligen Produkt überhaupt vorhanden – wird mit monetären Mitteln zu begleichen sein.

Viele Communities (das sind Gruppen, mit einer gemeinsamen Zielsetzung) bieten freiwilligen und selbstlosen Support und technische Hilfestellungen ihrer Produkte. Es herrscht somit eine andere Art von Unterstützung im Gegensatz zu kommerziellen Software Paketen. Weiters muss diese Form der Serviceleistung nicht zwangsläufig schlechter sein als bei bezahlter Software.

Bei der Auswahl eines Open Source Produktes ist dieses Thema mit hoher Wichtigkeit zu betrachten und darauf zu achten, dass sich die Gruppe der Entwickler nicht auf einen zu kleinen Kreis beschränkt bzw. die Aktivität und Veränderung am Produkt in einem großen Ausmaß erfolgt.

[5] „an open letter to hobbyists“ [online], Verfügbar unter http://tranquileye.com/cyber/1976/gates_open_letter_to_hobbyists.html [Zugang am 27. März 2011]

2.1.3.2. Fehlende Weiterentwicklung

Die fehlende Weiterentwicklung der Open Source Produkte wird oft als einer der Hauptgründe genannt, diese nicht in Unternehmen einzusetzen. Dieser vermeintliche Nachteil von freier Software kann aber auch auf kommerzielle Produkte zutreffen – zugegeben ist die Wahrscheinlichkeit bei Open Source Software höher, da gewisse rechtliche Absicherungen nicht vorhanden sind, aber auszuschließen ist es dennoch nicht. Da der Source Code bei freien Applikationen verfügbar ist, könnte im Notfall ein Dritter bzw. eine Eigen-Weiterentwicklung des Produktes erfolgen, was bei kommerziellen Programmen auf keinen Fall aus rechtlichen Gründen möglich wäre.

Auch hier gilt: Sorgfältige Auswahl des Produktes - hinsichtlich eines großen Aktivitätsaufkommens – erspart nachträgliche Schwierigkeiten.

2.2. Didaktik

Der Begriff Didaktik behandelt in erster Linie das „Was“ und das „Wie“ in der Wissenschaft vom Unterrichten.

Die Didaktik wird in der Fachliteratur als Teilgebiet der Pädagogik angesehen.

Definition Pädagogik

> *„Das Wort Pädagogik stammt vom griechischen paidagogike – welches als Erziehungskunst bezeichnet werden kann. Die Pädagogik ist eine Sammelbezeichnung für alle Wissenschaften, die sich mit Erziehung und Ausbildung befassen.“ [KRO05]*

Definitionen Didaktik

Es gibt viele Definitionen in den diversen Lexika – im Folgenden werden zwei weit verbreitete zitiert:

> *„Didaktik bezeichnet die Wissenschaft vom Unterricht, vom Lernen und Lehren, wobei sie sich mit dem Lernen in allen Formen und dem Lehren aller Art unabhängig vom Lehrinhalt befasst.“*[6]

> *„Die Didaktik ist ein Teilgebiet der Pädagogik, das sich mit Lehren und Lernen, besonders in Schulen, beschäftigt. Die Fach-Didaktik befasst sich mit Bildungsinhalten und Lehrplänen eines Schulfachs.“ [HAR94]*

Die Didaktik befasst sich somit mit dem Inhalt und den Methoden des Lehrens und Lernens im Bildungsbereich. Die Bedeutung ist jedoch nicht nur auf die Fragen „Was wird gelehrt/gelernt?“ und „Wie wird etwas vermittelt?“ beschränkt. Eine Zusammenstellung von wichtigen Fragen bezüglich Inhalte, Methodik, Zielen usw. werden im Folgenden aufgelistet und als Grundfragen der Didaktik bezeichnet.

Grundfragen der Didaktik [KRO05]

Was?	Welche Inhalte muss ich vermitteln?
Wie?	Mit welchen Methoden, Sozialformen, Medien usw. organisiere ich die Vermittlung der Inhalte?
Wozu?	Welche Ziele verfolge ich dabei?
Warum?	Welche Begründungen sind für meine Planungen und Vorhaben maßgeblich?
Wem?	Wer sind meine Adressaten?
Wo?	In welcher Umgebung finden die Lehr- und Lernprozesse statt?
Wann?	Zu welcher Tages- oder Jahreszeit findet die Arbeit statt?
Wer?	Welche Rollen nehmen die Akteure in den Lehr- und Lernprozessen ein?

[6] Lexikon für Psychologie und Pädagogik [online], Verfügbar unter http://lexikon.stangl.eu/706/didaktik/ [Zugang am 20. Jänner 2012]

In der Literatur wird die allgemeine Didaktik, sprich die Wissenschaft vom Lehren und Lernen, unterschiedlich zur „speziellen Didaktik“[7] gesehen. Diese behandelt einzelne Lehrbereiche (z. B. sprachlich, musikalisch, politisch. …) bzw. verschiedene Schulformen (Volks-, Hauptschulen, Gymnasien,…).

In der allgemeinen Didaktik gibt es vier Ansätze, welche im folgenden Abschnitt kurz erläutert werden.

2.2.1. Theoretische Ansätze der allgemeinen Didaktik

Theorie und Praxis gehen meist nicht Hand in Hand, dennoch haben theoretische Grundsätze immer einen mehr- oder minderstarken Bezug zur Praxis. Die theoretischen Ansätze der allgemeinen Didaktik [HUB07], [GUD08] mögen im ersten Eindruck nicht die Sinnhaftigkeit bzw. den Zweck offenbaren, aber sie haben beim Unterrichten und in der Unterrichtsplanung sowie –gestaltung einschneidende Spuren hinterlassen.

2.2.1.1. Bildungstheoretischer Ansatz

Der bildungstheoretische Ansatz wird auch als „Göttinger Schule“ bezeichnet und hat als zentrales Leitbild die bildende Begegnung des Menschen mit der kulturellen Wirklichkeit.

Bekannte Vertreter: Wolfgang Klafki, Josef Derbolav, Erich Weniger

Nach W. Klafki ist der bildungstheoretische Ansatz wie folgt definiert:

> *„Die Aufnahme und Aneignung von Inhalten ist stets verbunden mit der Formung, Entwicklung und Reifung von körperlichen, seelischen und geistigen Kräften.“*

Dieser Ansatz zielt speziell auf die Allgemeinbildung ab.

2.2.1.2. Lerntheoretischer Ansatz

Der lerntheoretische Ansatz wird auch als „Berliner Didaktik“ bezeichnet und stellt die Didaktik als Theorie des Lehrens und Lernens dar.

Im bildungstheoretischen Ansatz wird ausschließlich das „Was ist zu lehren?“ hinterfragt. Hiervon unterscheidet sich der lerntheoretische Ansatz, der zusätzlich das „Wie soll ein Unterrichtsstoff vermittelt werden?“ berücksichtigt.

Bekannte Vertreter: Paul Heimann, Gunter Otto, Wolfgang Schulz

2.2.1.3. Informationstheoretisch-kybernetischer Ansatz

Im informationstheoretisch-kybernetischen Ansatz erfolgt eine Reduzierung der Didaktik auf die reine Methodik, d.h. kybernetische Methoden und Begriffe werden auf die Planung des Unterrichts angewendet. Die Vereinfachung der Didaktik in diesem Ansatz beschreibt als Ziel, dass es beim Lernen grundsätzlich um Vermittlung bzw. Erwerb von Informationen geht.

Diese Reduzierung aufs Minimale hat in manchen Bereichen Gefallen gefunden (zB Militär, Industrie), aber im schulischen Bereich ist der Begriff Didaktik umfangreicher und facettenreicher zu definieren.

[7] Psychologische Begriffsbestimmungen [online], Verfügbar unter http://www.stangl.eu/psychologie/definition/Didaktik.shtml [Zugang am 21. Jänner 2012]

Bekannte Vertreter: Felix von Cube, Helmar Gunter Frank

Der Versuch, mathematische Modelle auf den menschlichen Bereich abzubilden und dadurch keine Rücksicht auf menschliche Ziele (z. B. Wille der Selbstverwirklichung) zu nehmen, verfehlt in manchen Bereichen die gewünschten Resultate.

2.2.1.4. Kritisch-kommunikative Didaktik

In diesem theoretischen Ansatz werden die behandelten Themen sowie auch die Gegebenheiten bzw. Tatsachen ständig kritisch hinter fragt. Die kritisch-kommunikative Didaktik beschäftigt sich vor allem mit dem Klima im Unterricht sowie mit der Beziehung zwischen LehrerInnen und SchülerInnen. Das Unterrichten soll kein Instruieren der SchülerInnen sondern ein Interagieren mit den SchülerInnen sein.

Weiters werden auch mögliche Störungen im Unterricht hervorgehoben und in verschiedene Störungsklassen unterteilt:

- Störungsarten
  - Disziplin, Provokation, Verweigerung
- Störungsfestlegungen
  - Vom Lehrenden, vom Lernenden, vom Lehrprozess
- Störungsrichtungen
  - SchülerInnen vs LehrerInnen, SchülerInnen vs SchülerInnen
- Störungsfolgen
  - Unterbrechungen, soziale Verletzungen
- Störungsursachen
  - Schulischer Kontext, gesellschaftlicher Bereich

Bekannte Vertreter: Rainer Winkel, Karl-Hermann Schäfer, Klaus Schaller

2.3. Unterrichtsformen[8]

Das Arbeiten mit dem Arduino bietet geradezu in der interdisziplinären Unterrichtsplanung hervorragende Anwendungsmöglichkeiten.

Die interdisziplinäre Unterrichtsplanung bietet die Möglichkeit, Brücken zwischen diversen Gegenständen zu schlagen. Der Arduino kann in den weit verbreiteten und meist angewandten Unterrichtsformen ideal eingesetzt werden.

Fächerkoordinierend (interdisziplinär)
Es wird ein gemeinsames Thema in mehreren Fächern behandelt.

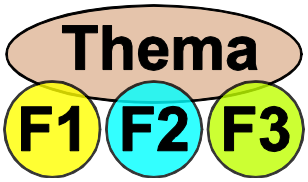


Abbildung 2, Interdisziplinärer Unterricht

Fächerübergreifend
Es wird ein Thema in mehreren Fächern behandelt, jedoch in jedem Unterrichtsfach aus einem anderen Blickwinkel.

Abbildung 3, fächerübergreifender Unterricht

Fächerverbindend
Es werden die Unterrichtseinheiten gemeinsam abgehalten, dabei wird das Thema von mehreren Wissensgebieten beleuchtet.

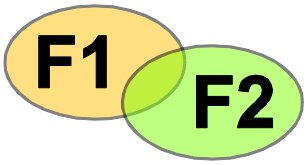


Abbildung 4, fächerverbindender Unterricht

[8] Interdisziplinäre Unterrichtsplanung (188.184), Vorlesung und Übung TU Wien, 2008

Projektorientiert
Für einen definierten Zeitraum wird in verschiedenen Fächern ein Thema behandelt – die „klassische“ Unterrichtsform wird aufgelöst.

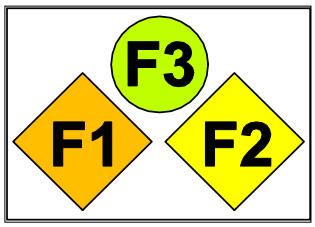


Abbildung 5, projektorientierter Unterricht

2.4. Zielgruppe

Die entwickelten Lehreinheiten sollen einen möglichst offenen und uneingeschränkten Zugang zur Thematik Arduino bieten, dennoch ist eine Fokussierung auf eine Zielgruppe für eine zielgerichtete und umfassende Darstellung unerlässlich.

Das entwickelte Lehrbuch soll speziell für den schulischen Bereich die Materie rund um Mikrocontrollern, Elektronik und Programmierung darstellen. Die Zielgruppe ist die neunte bzw. zehnte Schulstufe, d. h. Oberstufe in Gymnasien bzw. erste und zweite Klasse von Höheren Berufsbildenden Schulen - hier bieten Höhere Technische Lehranstalten nahezu ein optimales Einsatzgebiet.

2.5. Lehrpläne

Die Lehrpläne bilden die pädagogische und rechtliche Grundlage für die Bildungsangebote an österreichischen Schulen.[9]

Speziell Informatik als Unterrichtsfach ist in den letzten Jahren zu einem zentralen Element in schulischen Lehrplänen avanciert, deshalb wird im Folgenden ein kurzer Auszug aus und Verweis zu den diversen Lehrplänen angeführt, für die das entwickelte Lehrbuch konzipiert ist.

2.5.1. Auszug aus dem Lehrplan AHS (9. – 10. Schulstufe) [10]

Der Lehrplan der AHS Oberstufe im Pflichtfach Informatik sieht in groben Zügen folgende inhaltliche Aufgaben vor:

Bildungs- und Lehraufgabe:
Der Informatikunterricht fasst vorhandene Fähigkeiten von Schülerinnen und Schülern durch Beschäftigung mit Entwurf, Gestaltung und Anwendung von Informationssystemen zusammen und baut sie aus. Bei der kritischen Auseinandersetzung mit den dabei ablaufenden Prozessen und deren Ergebnissen sollen die Schülerinnen und Schüler ihr kognitives, emotionales und kreatives Potenzial nützen. Dies soll die Jugendlichen bei der Entwicklung einer persönlichen Werthaltung unterstützen.

Beiträge zu den Bildungsbereichen:
Sprache und Kommunikation:
- Die Informatik trägt wesentlich zu einer Veränderung der Kommunikation bei. Unterschiedliche Formen von Information ergänzen die traditionelle Verständigung und erfordern neue Denkstrukturen.

Mensch und Gesellschaft:
- Arbeitswelt und privates Umfeld der Menschen verändern sich durch den Einfluss der Informationstechnologien.

Natur und Technik:
- Durch Modellbildung, Formalisierung und Abstraktion leistet die Informatik einen wesentlichen Beitrag zur Auseinandersetzung mit Natur und Technik und führt zu einer verbesserten Entscheidungs- und Handlungskompetenz.

Kreativität und Gestaltung:
- Der Umgang mit Informationstechnologie gibt den Schülerinnen und Schülern Gelegenheit, selbst Gestaltungserfahrungen zu machen.

Gesundheit und Bewegung:
- Die Verantwortung für den eigenen Körper erfordert bei der Arbeit am Computer gezielte Bewegung als Ausgleich.

Lehrstoff:
Die Schülerinnen und Schüler sollen:
- Informationsmanagement und Lernorganisation für die eigene Lernarbeit und Weiterbildung mit geeigneter Software in der Praxis umsetzen und dabei vorhandene Informationsquellen erschließen und unterschiedliche Informationsdarstellungen ausgehend von den Vorkenntnissen anwenden
- Inhalte systematisieren und strukturieren sowie Arbeitsergebnisse zusammenstellen und multimedial präsentieren können
- ein vernetztes Informationssystem für die individuelle Arbeit aufbauen und nutzen können
- den sicheren Umgang mit Standardsoftware zur schriftlichen Korrespondenz, zur Dokumentation, zur Publikation von Arbeiten, zur multimedialen Präsentation sowie zur Kommunikation erreichen
- wesentliche Maßnahmen und rechtliche Grundlagen im Zusammenhang mit Datensicherheit, Datenschutz und Urheberrecht kennen lernen sowie die Auswirkungen des Technikeinsatzes auf die Einzelnen und die Gesellschaft nachvollziehen
- Einsatzmöglichkeiten der Informatik in verschiedenen Berufsfeldern kennen lernen und somit in ihrer Berufsorientierung Unterstützung finden

[9] HTL – Bildung mit Zukunft, Verfügbar unter http://www.htl.at/de/htlat/lehrplaene.html [Zugang am 20. Jänner 2012]

[10] Lehrpläne der AHS Oberstufe, Verfügbar unter http://www.bmukk.gv.at/schulen/unterricht/lp/lp_ahs_oberstufe.xml [Zugang am 20. Jänner 2012]

Zwei Bildungsbereiche von den allgemeinen Bildungszielen sind für das Umfeld Arduino herauszustreichen:

- Natur und Technik
  - Das Natur und Technik Hand in Hand gehen, ist allgemein bekannt – durch Projekte im Arduino-Umfeld sind diese auch visualisierbar. Physikalische Themen (z. B. elektrischer Strom) können dargestellt und angreifbar gemacht werden.
- Kreativität und Gestaltung
  - Durch die individuelle Lösungssuche sowie der zielgerichteten Problemlösung bietet die Arduino-Welt den nötigen Gestaltungsraum und fördert in einem hohen Maße die Kreativität.

In den allgemeinen didaktischen Grundsätzen werden die „Stärkung von Selbsttätigkeit und Eigenverantwortung“ sowie das „Herstellen von Bezügen zur Lebenswelt“ erwähnt. Das Arbeiten mit dem Arduino trifft diese beiden Grundsätze sehr genau.

Gegenstände
In der AHS Oberstufe sind die folgenden Gegenstände sehr einfach in Bezug auf die Arduino-Welt zu setzen:

- In der 9. Bzw. 10. Schulstufe
  - Mathematik
  - Physik
  - Informatik

2.5.2. Auszug aus dem Lehrplan HTL EDVO[11],[12]

Es gibt eine große Anzahl an berufsbildende Schulen in Österreich. Eine Liste mit zugehörigen Lehrplänen wird unter folgendem Link angeführt:

- http://www.abc.berufsbildendeschulen.at/de/dlcollection.asp
  [Zugang am 20. Jänner 2012]

Die Einschränkung und den Verweis auf die Höhere Lehranstalt für Elektronische Datenverarbeitung und Organisation ist dadurch begründet, da dieser Schultyp geradezu prädestiniert für Anwendungen im Arduino-Bereich ist. Eine weitere Ausbildung der berufsbildenden Schulen ist die Elektrotechnik-Richtung, in die die Thematik hervorragend platzfinden kann.

[11] Lehrplan der Höheren Lehranstalt für elektronische Datenverarbeitung und Organisation, Anlage 1, Verfügbar unter http://www.htl.at/fileadmin/content/Lehrplan/HTL/BGBl._Anlage_1_302-97.pdf [Zugang am 20. Jänner 2012]

[12] Lehrplan der Höheren Lehranstalt für elektronische Datenverarbeitung und Organisation, Verfügbar unter http://www.htl.at/fileadmin/content/Lehrplan/HTL/HL_EDVO_Anlage_1.3.1_BGBl._382-98.pdf [Zugang am 20. Jänner 2012]

Im Allgemeinen Bildungsziel des Lehrplanes befindet sich der erste Lehrplanbezug zum behandelten Thema.

Allgemeines Bildungsziel
Fachkompetenz
Kenntnis der mit dem Berufsfeld zusammenhängenden fachlichen Inhalte in Theorie und Praxis

Methodenkompetenz
Fähigkeit, Informationen zu beschaffen und Problemlösungen zu planen, geeignete Lösungsmethoden auszuwählen und durchzuführen

Sozialkompetenz
Fähigkeit zu Kooperation und Kommunikation, Teamfähigkeit

Selbstkompetenz
Fähigkeit zu aktiver Lebens- und Berufsgestaltung, zu Selbstorganisation, Eigeninitiative und Weiterbildung

Arbeiten mit dem Arduino fördert alle vier angeführten Kompetenzen, besonders natürlich die Fachkompetenz und die Methodenkompetenz. Durch das Finden von eigenen Lösungswegen auf Basis von theoretischem Grundlagenwissen wird die Fähigkeit der Problemlösung wesentlich gefördert.

In den fachrichtungsspezifischen Bildungszielen findet sich, aufgrund der Thematik, der nächste Bezug.

Fachrichtungsspezifische Bildungsziele
- Fundierte Allgemeinbildung
- EDV Wissen
  * Hardware, Software
  * Projektplanung und -abwicklung
    -- Insbesondere in der Softwareentwicklung
- Kaufmännische Kenntnisse in betrieblicher Organisation und Rechnungswesen

Die Thematik rund um den Arduino zielt in erster Linie auf ein fundiertes EDV Wissen ab. Hierbei können sowohl die Hardware sowie die Software (Programmierung) im Vordergrund stehen.

Eine Entwicklung eines Programmes im Arduino Umfeld bewirkt auch immer ein gewisses Maß an Planung – somit kann diese Gegebenheit wunderbar in die Bildungsziele für die Projektplanung und –abwicklung einfließen.

Gegenstände

Es gibt einige hervorragend geeignete Pflichtgegenstände in der HTL EDVO:

- In der 9. Bzw. 10. Schulstufe
  - Angewandte Mathematik
  - Angewandte Physik
  - Grundlagen der elektronischen Datenverarbeitung
  - Programmieren

- Ab der 11. Schulstufe
  - Projektentwicklung (ab 11. Schulstufe)

3. Lehrbuch

3.1. State of the Art

Die schnelllebige Zeit bzw. die kurzen Lebenszyklen von neuen Technologien in der Informatik und den Computerwissenschaften erfordern ein ständiges Lernen und eine laufende Anpassung an den Stand der Technik.

Mit dem Integrieren von Microcontrollern in den Ausbildungsplan von Studenten [PAP01] nimmt die McNeese State University (MSU, Louisiana) eine Vorreiterrolle ein. Microcontroller finden in der interdisziplinären Ausbildung oft zu wenig Beachtung. Die MSU hat in dieser Thematik gegengesteuert und sich zum Ziel gesetzt, Projekte zu initiieren, wie Microcontroller in der Informatik-Ausbildung fächerübergreifend eingesetzt werden können. Diese Projekte sollen den Studenten eine neue Sichtweise für die Nutzung von Microcontrollern und deren Anwendung über die Grenzen der Informatik-Ausbildung hinaus geben.

Die Gebiete mit hoher Industrialisierung (z. B. Texas und Louisiana) in denen Unternehmen (speziell Unternehmen im Umweltbereich) laufend Daten sammeln und auswerten müssen, bieten ideale Anwendungsfälle. Dieser Umstand führte an der MSU zur Entwicklung von Kursen, welche die Thematiken rund um den Arduino abdecken. Es wurden Kurse in verschiedenen Schwierigkeitsstufen (Levels) entwickelt, welche in den Informatiklehrplan eingearbeitet wurden.
Angestrebtes Ziel ist, das Bild der Informatik-Ausbildung aufzuwerten – Informatiker heißt nicht nur am PC sitzen und zu programmieren – die „physisch greifbare“ Technik, wie sie mit dem Arduino abgebildet werden kann, ist dazu ein fast ideales Instrument.

Arduino-basierende Projekte in der Informatik [PAP02] wurden umgesetzt, um Wirtschaft, Industrie und Forschung mit der Informatikausbildung zu verbinden. Die interessantesten drei Projekte in der Robotik an der MSU sind im Folgenden kurz dargestellt – sie erforschen die Navigation und Kontrolle von selbständig fahrenden U-Booten:

1) *Selbständig fahrendes Speedboot*

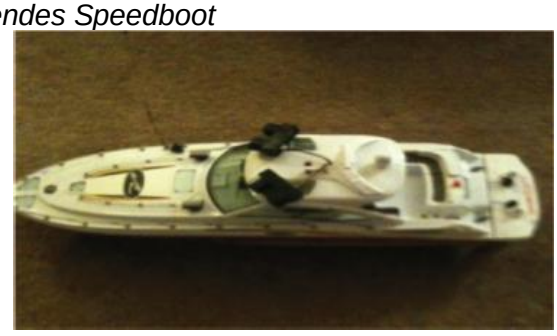


Abbildung 6, Speedboot

Die wissenschaftliche Erforschung in diesem Projekt:

- Verstehen der Unterschiede der 2-dimensionaler Navigation zwischen Wasser und Land

2) *Halbselbständig fliegendes Luftschiff*

Abbildung 7, Luftschiff

Die wissenschaftliche Erforschung in diesem Projekt:

- Verstehen der Unterschiede der Kollisionsvermeidung in 2- bzw 3-dimensionalen Welten

3) *Luftkissenboot*

Abbildung 8, Luftkissenboot

Die wissenschaftliche Erforschung in diesem Projekt:

- Verstehen von Richtungsänderungen durch Strahlen bzw. Düsen

Die Rückmeldung der Studenten war sehr positiv und eindeutig - sie wünschen sich mehr „greifbare“ Technik in Kursen.

Die steigende Anzahl an Tutorials bzw. Kursen spiegelt das große Interesse an dem Thema wider. Kurse wie „praktisches Arbeiten mit dem Arduino“ [PAP03] bzw. „Einführung in die Programmierung unter Verwendung des Microcontrollers Arduino“ [PAP04] machen dieses sehr deutlich. Die vielseitige Verwendbarkeit des Arduinos sowie die günstige Anschaffung und die Einsatzmöglichkeit im schulischen Bereich und Universitäten sind maßgeblich am Erfolg beteiligt - das Open-Source-Konzept trägt hierzu seinen Teil bei.

Die Vielfalt an Anwendungsgebieten wird durch die enorme Anzahl an vorhandenen Projekten dargestellt.

Im Gesundheitsbereich kann z. B. eine Herzschlag- und Temperaturmessung [PAP07] erfolgen. In einem Projekt der Massey Universität (Neuseeland) wird ein Arduino-Microcontroller dazu verwendet, Messdaten von einer Gruppe von freiwilligen Personen an einen PC zu senden - gesendete Daten werden verschlüsselt übertragen. Zur Übertragung wird ein Wireless-XBee-Module verwendet.

Im Bereich der Heimanwendungen sind den Ideen der EntwicklerInnen keine Grenzen gesetzt. Handgesteuerte Bedienung für Heimanwendungen („Handmote“) [PAP06] sowie Bluetooth-basierte Heimsystem-Automationen [PAP05] erfreuen sich immer größerer Beliebtheit.

3.2. Konzept

Mittlerweile existiert eine Vielzahl an Arduino-Büchern und eine beträchtliche Anzahl an Online-Seiten zum Thema Arduino, was den großen Erfolg des Arduino-Projektes bestätigt.

Das Konzept des Arduino und die damit verbundenen Realisierung von Projekten ist darauf ausgelegt, schnell Erfolgserlebnisse zu generieren: Nach dem Prinzip downloaden, installieren und schon läuft das erste kleine Programm. In der physikalischen Entwicklung ist das erste Programm häufig ein blinkendes LED – vergleichbar mit dem „Hello World!“-Programm in der Java-Welt.

Namhafte Verlage bieten Bücher [BAN08], [BRU10], [ODE10] an, die einen hervorragenden Einstieg in die Entwicklung von Arduino Projekten bieten. Technik-Interessierten und Personen mit ersten Programmiererfahrungen sind diese Lektüren sicherlich eine geeignete Hilfe, um erste Erfahrungen im Umgang mit Microcontrollern zu machen. Mit Unterstützung der mittlerweile riesigen Online-Community lassen sich auch innerhalb kurzer Zeit anspruchsvolle Projekte[13] realisieren.

Das generelle Konzept von Büchern zielt – naturgemäß bzw. aus marktwirtschaftlichen Aspekten – auf eine breite Masse von Personen ab. Hierbei steht eine Fokussierung auf eine bestimmte Zielgruppe außer Acht. Für Anfänger werden Bücher zum Verkauf angeboten, die einen Praxiseinstieg bzw. ein einfaches „Getting started“ beschreiben. Ambitionierte Personen oder solche, die bereits erste Erfahrungen im jeweiligen Umfeld gesammelt haben, wird eine große Sammlung an Projekten in Buchform geboten.

Die Fokussierung auf eine Zielgruppe bietet die Möglichkeit, eine spezielle Herangehensweise an das zu verfassende Thema zu wählen. Dabei muss nicht die Allgemeinheit angesprochen werden, sondern es kann speziell eine Lesergruppe definiert werden, für die die Thematik klar und strukturiert aufbereitet werden kann. Die SchülerInnen je Schulstufe stellen eine derart definierte Zielgruppe dar.

Bücher im schulischen Bereich bzw. Unterlagen für Schulungen sollten speziell für jenes Anwendungsgebiet erstellt werden, in dem es Anwendung findet. Die Didaktik und die damit verbundenen Grundfragen der Didaktik sollten Einfluss auf die Form, Umfang und Struktur nehmen. Allgemeingehaltene Bücher haben diesen Fokus nicht – hierbei wird meist nur die Frage gestellt: „Welches Thema wird behandelt?“. Dabei werden meist die Themen in chronologischer Reihenfolge Kapitel für Kapitel bzw. Abschnitt für Abschnitt durch- und abgearbeitet.

Die didaktischen Grundfragen bleiben meist auf der Strecke, da Fokussierung und Spezialisierung auf eine Zielgruppe sowie Strukturiertheit und Einfachheit keine vordergründigen Rollen spielen. Hierbei sind Lehrbücher, welche speziell für SchülerInnen konzipiert werden, klar im Vorteil.

Wie lernt man mit einfachen Mitteln das Programmieren? Wie kann man elektronische Grundlagen „angreifbar“ machen? Eine sehr gute Antwort auf beide Fragen kann sein: „Arbeitet mit dem Arduino“ – diese günstig-konzipierten Mikrocontroller können für das

[13] Arduino [online], Verfügbar unter http://www.arduino.cc/forum/ [Zugang am 22. Jänner 2012]

Erlernen von Softwareentwicklungskonzepten und grundlegendes Verstehen der Elektronik herangezogen werden. Das Arduino-Projekt ist Open Source - eine große Community hat sich innerhalb kürzester Zeit entwickelt und dazu geführt, dass viele Beispiele zur Verfügung stehen. Die Mikrocontroller sind ideal, um das Zusammenspiel von Aktion und Reaktion – Programmierung und Mikrocontroller – darzustellen.

Das im Rahmen dieser Arbeit erstellte Lehrbuch unterscheidet sich von vorhandenen Büchern durch die Herangehensweise an die Thematik. Viele Bücher bieten einen sehr guten Überblick und Einstieg in die Arduino-Welt, sind jedoch nicht für den Unterricht in Schulen explizit vorgesehen. Im eigenen Lehrbuch wurde darauf Wert gelegt, eine klare Struktur in den Lehreinheiten zu schaffen. Dabei wurde Bezug auf den schulischen Einsatz durch einen Theorieteil mit Fragen und einen Praxisteil mit Beispielen genommen.

Der Aufbau in Lehreinheiten soll alle wichtigen Bereiche für das Arbeiten mit dem Arduino vermitteln. Hauptthemen sind somit:

- Installation der Hardware und Software
- Grundlagen im Hinblick auf den Arduino bzw. der Elektronik und
- zahlreiche Beispiele für Anfänger bis Fortgeschrittene.

Die Konzeption im Lehrbuch bildet aufbauende Thematiken ab.

Die wichtigsten drei Unterscheidungsmerkmale zu den bereits vorhandenen erhältlichen Büchern sind:

- Selbststudium
    - das Lehrbuch wurde so zu konzipiert, dass ein selbstständiges Durcharbeiten ermöglicht wird.
- Praxisorientiert
    - Durch den explorativen Ansatz – SchülerInnen sollen (durch eigene Ideen) Beispiele entwickeln und umsetzen – werden aktuelle Thematiken und Interessen geweckt und realisiert.
- SchülerInnengerecht
    - Ein Hauptaugenmerk wurde auf die schulische Verwendungsmöglichkeit gelegt.

Außerdem wurde explizit auf die didaktischen Grundfragen eingegangen und diese im Lehrbuch abgebildet. Die „8-W“-Fragen der Didaktik stellen die Basis für speziell konzipierte Bücher dar.
Sind diese Fragen berücksichtigt worden?

Frage		berücksichtigt
Was?	Welche Inhalte muss ich vermitteln?	Ja
Wie?	Mit welchen Methoden, Sozialformen, Medien usw. organisiere ich die Vermittlung der Inhalte?	Ja
Wozu?	Welche Ziele verfolge ich dabei?	Ja
Warum?	Welche Begründungen sind für meine Planungen und Vorhaben maßgeblich?	Ja
Wem?	Wer sind meine Adressaten?	Ja
Wo?	In welcher Umgebung finden die Lehr- und Lernprozesse statt?	Ja
Wann?	Zu welcher Tages- oder Jahreszeit findet die Arbeit statt?	Ja
Wer?	Welche Rollen nehmen die Akteure in den Lehr- und Lernprozessen ein?	Ja

3.3. Evaluation

Durch ein 4-wöchiges Sommerpraktikum an der TU Wien, Institut für computerunterstützte Automatisierung, war es möglich, das entworfene Lehrbuch praxisnah von SchülerInnen evaluieren zu lassen. Die Praktikantin und die Praktikanten besuchten im Sommermonat Juli (2011) die TU Wien und behandelten die Thematiken rund um die Microcontroller-Familie Arduino.
Für das Praktikum wurde, unter der Berücksichtigung der Schulstufe bzw. des Schultyps sowie Schülerinnen bzw. Schüler, eine ideale Auswahl getroffen. Am Praktikum nahmen somit teil:

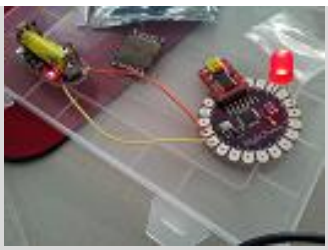
- Eine Schülerin, 10. Schulstufe, AHS
- Ein Schüler, 10. Schulstufe, AHS
- Ein Schüler, 11. Schulstufe, HTBL (Fachrichtung Elektrotechnik)

3.3.1. Berichte und Präsentationen

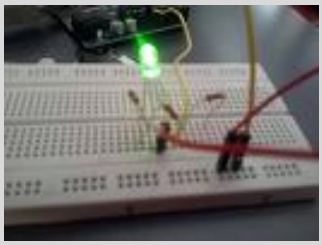
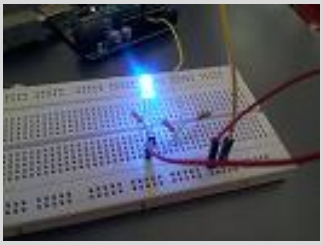
Die PraktikantInnen erhielten die Aufgabe, wöchentlich einen Bericht zu führen und eine Präsentation über die Erfahrungen bzw. Tätigkeiten zu halten. Die Präsentationen wurden auch dazu genutzt, um aufgetretene Fragen bzw. Probleme in der Gruppe zu diskutieren und zu lösen.

Beispiel eines Wochenberichtes

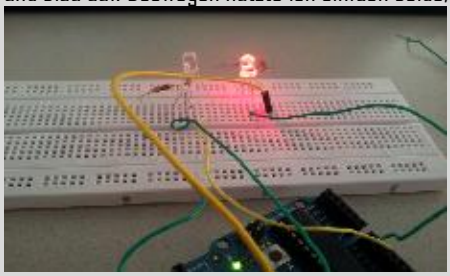
(11.07.2011) An diesem Vormittag erzählte uns jede/r FerialpraktikantIn über deren Tätigkeiten der letzten Woche mithilfe einer kleinen Präsentation. Unser Projektleiter Thomas G. war auch anwesend für eventuelle Fragen. Nach langem Überlegen was ich als Nächstes machen könnte, fiel mir auf, dass ich noch gar nicht das RGB LED ausprobiert habe. Die RGB Leuchtdiode kann nämlich verschieden Farben wiedergeben. Diesen Versuch beschloss ich am nächsten Tag auszuprobieren.
(12.07.2011) Heute haben wir neue technische Zusatzbauteile bekommen, unter anderem auch das legändere Lilypad. Ich erkundigte mich mithilfe von Links, die wir unseren Projektleiter erhielten, was man so alles mit dem Lilypad machen könnte. Die Hauptfunktion ist, dass man es auf Textilien annähen kann. Es gibt auch besondere Zusatzbauteile für das Lilypad, die man auch alle annähen kann wie z.B.: eine Leuchtdiode. Somit entschloss ich mich, ein derartiges Projekt zu starten. Bevor ich aber etwas Komplexes starten konnte, musste ich mich erst mit dessen Funktionen auseinandersetzen.



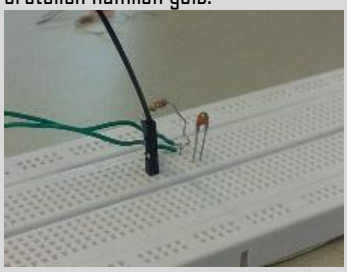
Am obigen Bild habe ich mit einer externen Energie-Quelle(Batterie) eine LED zum blinken gebracht(auch per USB möglich).
Als nächstes probierte ich die RGB LED aus. Es hat insgesamt 4 Anschlüsse, wobei der längste die Kathode ist uns somit mit dem GND verbunden werden muss. Bei jedem der drei Anschlüsse leuchtet eine unterschiedliche Farbe auf, man braucht einfach nur ein Verbindungskabel zum 5V Anschluss am Arduino.

(13.07.2011) Mithilfe von einem Sketch ist es mir gelungen, die RGB LED zum stetigen Farbwechsel zu bringen. Am Vortag habe ich es ohne Programm zum Testen probiert, bei welchem Anschluss die LED rot, grün oder blau aufleuchtet. Dabei habe ich einen Kurzschluss verursacht, weil ich das 5V Kabel zum Ground Kabel gesteckt habe. Die RGB LED kann jetzt nur noch rot und blau aufleuchten, deswegen habe ich eine Neue genommen. Bei dieser tritt leider auch ein Problem auf, denn die LED leuchtet nur grün und blau auf. Deswegen nutzte ich einfach beide, da sie zusammen in allen drei Farben aufleuchten konnten.



Des Weiteren habe ich ein Testprogramm von Thomas bekommen zum Thema RGB LED. Dieser Sketch lässt alle Farben nach der Reihe langsam aufleuchten und wieder verschwinden. Im Internet stand, dass man auch andere Farben erzeugen kann, indem man einfach zwei Farben gleichzeitig aufleuchten lässt. Mit dieser Methode habe ich die Farben Magenta und Türkis erzeugt. Mehr konnte ich leider nicht erzeugen, da alle beiden LEDs nicht ganz funktionstüchtig waren. Ich erkundigte mich im Internet nach den Lilypad Zusatzbauteilen und gab Monika Di Angelo Bescheid, dass ich für mein Vorhaben mehr Bauelemente benötige.
(14.07.2011) Heute erfuhr ich, dass die Bauteile für das Lilypad bereits bestellt worden sind und im Laufe der nächsten Woche eintreffen werden. Zunächst überlegte ich mir, welchen Stoff ich für mein zukünftiges Projekt nehmen sollte. Ich entschied mich für den Anfang für ein einfaches Leinentuch. Danach beschloss ich, mich mit dem Temperatursensor zu beschäftigen. Dieser hat zwei Anschlüsse, den einen verband ich mit 5V, den anderen mit dem GND und zusätzlich mithilfe von einem Widerstand mit einem Analogen Eingang. Mithilfe von einem Sketch soll die aktuelle Temperatur ausgelesen und am Serial Monitor ausgegeben werden. Dieses Vorhaben ist auch gelungen. Auf einmal fand Martin(Ferialpraktikant) eine neue RGB LED, die ich sofort mit meinem vorhandenen Sketch austestete. Diese konnte in allen drei Farben aufleuchten und zusätzlich konnte ich noch eine weitere Farbe erstellen nämlich gelb.



Da wir am Montag wieder Präsentationen halten mussten, stellte ich eine Power Point über meine Tätigkeiten der letzten Woche zusammen.

Die anfängliche Schüchternheit bzw. Ablehnung bei den wöchentlichen Präsentationen verflog rasch und es entwickelte sich sehr schnell ein hervorragendes Klima, welches den Berichten den anfänglichen Schrecken nahm.

3.3.2. Informationsaustausch / Rückmeldung / Bewertung

Interviews
Es wurde versucht, die Kommunikationswege kurz zu halten – es wurden qualitative Interviews abgehalten, persönliche Gespräche geführt und Beispiele mit dem Arduino gemeinsam erarbeitet bzw. durchgeführt.

EMail
Die ständige Erreichbarkeit meinerseits – die Vorort-Anwesenheit war nur an zwei Tagen in der Woche gegeben – wurde durch Email Kommunikation gewährleistet. Im Folgenden ein paar kurze Auszüge aus den elektronischen Nachrichten:

Nach der ersten Einführung in die Thematik, wurde eine erste Meinung über den Microcontroller erfragt.

```
Hallo Sabrina,
Hallo Martin,

wie gehts euch mit dem Arduino? Habt ihr irgendwelche Fragen? Bitte nicht zögern,
einfach fragen...

Ich hätte eine Bitte...
Die erste Woche geht schön langsam dem Ende zu und Ihr müßt ja am Ende des
Praktikum einen Bericht
verfassen. Damit das Ganze nicht bis zuletzt aufgehoben wird, wäre es toll, wenn
ihr bis Montag/Dienstag
eine erste Zusammenfassung der vergangenen Woche machen könntet.
Was soll zusammengefasst werden?
- Einstieg mit Arduino - Wie waren die Unterlagen?
- Was habt ihr an Beispielen gemacht.
- Welche Programme/Beispiele würdet ihr gerne machen?

Viel Spaß noch,
-Thomas


P.S.: Könnt' Ihr mir bitte eine kurze Mail zurück schreiben, damit ich weiß, dass
ich die richtigen Email-Adressen verwendet habe... Danke...
```

Antwort 1

```
Hallo thomas,
Danke für dein email.
Ja werden wir machen.wir haben mit Monika besprochen,dass wir alle am mo eine
präsentation halten und einen bericht der vergangenen woche abgeben.
Also wenn du am montag da bist kannst du die Berichte gleich haben.

Wünsche dir noch eine schöne Woche/Wochenende
Mfg martin
```

Antwort 2

```
Hallo Thomas!
Mir geht es mit dem Arduino sehr gut und blicke dank deinem Lehrbuch sehr gut
durch... Bin gerade dabei mich im Internet zu erkundigen, was man noch so alles mit
dem Arduino anstellen kann bzw könnte.
Lg sabrina
```

Es wurde sehr schnell offensichtlich, dass zum Thema Arduino im Internet sehr viel Information auffindbar ist. Spezielle Projekte wurden informell vorgestellt, um Anreize zum Realisieren zu geben.
Der Arduino Lilypad sowie Lego Mindstorms in Kombination mit dem Arduino Uno stießen auf offene Ohren.

```
Hallo Sabrina,
Hallo Martin,
anbei ein paar Links bezüglich LilyPad...

LilyPad Arduino
http://www.arduino.cc/en/Main/ArduinoBoardLilyPad
http://web.media.mit.edu/~leah/LilyPad/
http://www.sparkfun.com/products/9266
http://www.slideshare.net/marrije/lilypad-arduino-a-small-introduction-presentation
http://www.ckuehnel.ch/PDF/LilyPad.pdf

LilyPad Arduino Projects (... nur ein kleiner Auszug ;-)
http://www.youtube.com/results?search_query=lilypad+arduino&aq=f
http://www.youtube.com/watch?v=L4a89n4ZJ5w
http://ifdblog.org/technology/?tag=wearable-computing-electronics-lilypad-arduino-
emerging-technologies
http://www.flickr.com/groups/lilypad_arduino/
http://www.flickriver.com/groups/lilypad_arduino/pool/interesting/

… und ein paar Links zum Ultra Sonic Sensor und Arduino...

http://shop.lego.com/de-AT/Ultraschallsensor-9846?langid=1033&p=9846
http://ls12-www.cs.tu-
dortmund.de/teaching/courses/ws0708/es/uebungen/download/Mindstorms_Bericht.pdf
http://www.scribd.com/doc/24698262/Rj12-and-Lego-Nxt
http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT
http://en.wikipedia.org/wiki/Modular_connector
http://sites.google.com/site/mccolganrobotics/Home/arduino-and-nxt-sensors
http://arduino.cc/forum/index.php?action=printpage;topic=62899.0

Liebe Grüße,
-Thomas
```

Es wurde versucht, eine offene und lockere Kommunikationsebene zu schaffen, um aufgetretene Probleme bzw. Fragen zeitnah lösen bzw. beantworten zu können.

```
Hallo Sabrina,
Hallo Martin,
möchte nur mal nachfragen, wie es euch geht...

Gibts vielleicht irgendwelche offenen Probleme / Fragen?

Zur Info: für das LilyPad habe ich einen Button und 3 LEDs bestellt - hoffe,
ich bekomme diese bis nächste Woche...

Wünsch euch noch einen schönen Tag und ein schönes Wochenende, -Thomas
```

Antwort

```
Hallo Thomas!
Derzeit gibt es von meiner Seite aus keine weiteren Fragen.
Das Programm für das RGB LED funktioniert einwandfrei.
Bis Montag
Sabrina
```

Auch ein erfolgreiches Projekt eines Praktikanten (Wärmebildkamera) wurde per Email kommuniziert:

```
Hallo Thomas,

Ich habe die Berichte zu Hause und werde sie dir dann schicken.
weiters funktioniert die Wärmebildkamera super -->
```

Fragebogen

In der letzten Praktikumswoche ist der Online-Fragebogen – hierfür wurde das Online-Portal Haekchen[14] verwendet – den PraktikantInnen via Email übermittelt worden.
Auszug aus dem Email, welches den Link für den Fragebogen enthielt.

```
…
Ich habe noch einen kurzen Fragebogen bezüglich des Lehrbuches
zusammengestellt - nehmt euch bitte ein paar Minuten Zeit und gebt mir
ein kurzes Feedback.
http://www.haekchen.at/haekchen/fragebogen.asp?uid=8143&id=1

Jetzt kann ich nur mehr sagen, dass es für mich eine große Freude und
Erfahrung war, mit euch zu arbeiten - Danke nochmal...
Liebe Grüße,
-Thomas
…
```

Der Fragebogen wurde anonym durchgeführt und ermöglichte eine gezielte Rückmeldung über das Lehrbuch bezüglich Gefallen, Aufbau, Ausgewogenheit von Theorie und Praxis sowie Fehlendem.

[14] Haekchen [online], Verfügbar unter http://www.haekchen.at/haekchen/ [Zugang am 27. Juli 2011]

Online-Fragebogen » Internetseite

Lehrbuch für Arduino

Praktikum an der TU Wien - 07/2011

Das Praktikum hat eine Einführung in die Welt des Mikrocontrollers Arduino geboten. Zusätzlich wurden Grundlagen der Elektronik und der Softwareentwicklung vermittelt. Praktische Anwendungs-beispiele standen im Vordergrund.

1. Wie hat dir der Aufbau des Lehrbuches gefallen?

2. Ist die Theorie verständlich verfasst?

3. Sind ausreichend praktische Beispiele vorhanden?

4. Werden die wichtigsten Aspekte für den Arduino behandelt?

5. Ist der Umfang des Lehrbuches für die Einführung ausreichend?

6. Was hat dir gut gefallen?

7. Was hat dir weniger gut gefallen?

8. Verbesserungsvorschläge...

Erstellt mit Häkchen - www.haekchen.at/haekchen/

Online-Fragebogen » Auswertung

1. Wie hat dir der Aufbau des Lehrbuches gefallen?

Durchschnitt:	4,7 Sterne	**1 Stern:**	keine	0
		2 Sterne:	keine	0
		3 Sterne:	keine	0
		4 Sterne:	33,30%	1
		5 Sterne:	66,70%	2

2. Ist die Theorie verständlich verfasst?

Durchschnitt:	4,3 Sterne	**1 Stern:**	keine	0
		2 Sterne:	keine	0
		3 Sterne:	keine	0
		4 Sterne:	66,70%	2
		5 Sterne:	33,30%	1

3. Sind ausreichend praktische Beispiele vorhanden?

Durchschnitt:	3,0 Sterne	**1 Stern:**	keine	0
		2 Sterne:	33,30%	1
		3 Sterne:	33,30%	1
		4 Sterne:	33,30%	1
		5 Sterne:	keine	0

4. Werden die wichtigsten Aspekte für den Arduino behandelt?

Durchschnitt:	4,3 Sterne	**1 Stern:**	keine	0
		2 Sterne:	keine	0
		3 Sterne:	keine	0
		4 Sterne:	66,70%	2
		5 Sterne:	33,30%	1

5. Ist der Umfang des Lehrbuches für die Einführung ausreichend?

Durchschnitt:	4,3 Sterne	**1 Stern:**	keine	0
		2 Sterne:	keine	0
		3 Sterne:	33,30%	1
		4 Sterne:	keine	0
		5 Sterne:	66,70%	2

6. Was hat dir gut gefallen?

Aufbau, Formulierung

der Aufbau, die Theorie, Lehrnziele, Zusatzinformationen (Widerstandbestimmung)

Das Lehrbuch ist sehr gut aufgebaut: Der Basis-Programmaufbau wird gleich zu Beginn mithilfe von einfachen Beispielen erklärt, wodurch man sich gleich in erste eigene Versuche stürzen kann und der darauffolgende Theorieteil ist kurz und einfach gehalten. Die Einteilung in Lehreinheiten ermöglicht es, sein angelerntes Wissen weiter zu vertiefen.

7. Was hat dir weniger gut gefallen?

Es wird zu wenig auf das Steckbrett eingegangen.
Es gibt kaum etwas an dem Lehrbuch auszusetzen, jedoch ist es wirklich nur für die ersten 5 Tage nützlich, danach ist man auf der Arduino-Website besser aufgehoben, jedoch kann das auch daran liegen, dass ich bereits Erfahrung mit Programmieren hatte. Als reines Einstiegsbuch, welches Jungprogrammierer nicht abschrecken soll, ist es geradezu perfekt.

8. Verbesserungsvorschläge...

Auf jeden Fall mehr Beispiele und Erklärungen für das Steckbrett.
mehr praktische Beispiele
Es wäre gut gewesen, wenn der Unterschied zwischen den Looparten genauer erklärt werden würde, z.B.: kopf- und fußgesteuert, Zählschleife, Anwendung, etc., da Schleifen ein wichtiger Aspekt der Programmierung sind.

Bei der Auswertung der Fragen, welche mit Sternen bewertet wurden, gilt:

- ein Stern ist die schlechteste Bewertung
- fünf Sterne ist die beste Bewertung

Bemerkungen zum Ergebnis

Das Lehrbuch für Arduino hat in den Bereichen Aufbau, Verständlichkeit der Theorie, wichtige Aspekte sowie Umfang sehr gut abgeschnitten. Hier waren die durchschnittlichen Bewertungen zwischen 4,3 – 4,7 Sternen.

Nur ein Mittelmaß im Fragebogen erreichte der Punkt, ob ausreichend Beispiele vorhanden waren. Eine durchschnittliche Bewertung von 3 Sternen macht diesen Umstand sehr deutlich bzw. offensichtlich. Aus den schriftlichen Rückmeldungen geht ebenfalls hervor, dass das Steckbrett (Breadboard) zu wenig behandelt wurde. Für Programmiererfahrene ist das Lehrbuch nur die erste Woche hilfreich, hingegen dürfte es für Programmieranfänger aber sein gestecktes Ziel – Einführung in die Thematik Arduino – in einem sehr hohen Maße erfüllen.

Die angeführten Verbesserungsvorschläge sowie das gesamte Ergebnis der Auswertung dienten der Weiterentwicklung des Lehrbuches. Ergänzende Beispiele und Erklärungen für das Steckbrett, praktische Beispiele und die Behandlung von Schleifen und deren Verwendung sind in die Fertigstellung eingeflossen.

4. Zusammenfassung / Resümee

Die Möglichkeit, für ein Praktikum eine spezielle Unterlage zu konzipieren und zu entwickeln, ist eine große Chance, theoretisches Fachwissen in praktische Ausführungen fließen zu lassen. Die Organisation eines Sommerpraktikums zum Thema „Arduino" an der TU – Wien bot diese Gelegenheit.

Die Begeisterung, der Ehrgeiz für sowie der Wissensdrang über die Thematik war bei den PraktikantInnen enorm. Die starke Präsenz des Themas im Internet trug hier sicherlich maßgeblich bei.

Fachwissen vs „Wissen vermitteln" – Die schwierige Aufgabe ist, komplexes Wissen möglichst einfach und plausibel zu transportieren. Praktische Erfahrung ist in diesem Fall unterlässlich und kann nicht durch rein theoretisches Fachwissen ersetzt werden.

Die Meinungen und Rückmeldungen über das Praktikum sowie über das Lehrbuch waren durchwegs positiv. Obwohl einige Lücken rasch entdeckt und Erweiterungswünsche schnell geäußert wurden, ist das Lehrbuch ein guter Begleiter im Kurs geworden. Eine tolle Ergänzung wäre eine Beispielsammlung, in der verschiedenste Anwendungen – nach Schwierigkeitsstufen unterteilt – nachschlagbar sind.

Die Thematiken rund um den Arduino bieten facettenreiche Möglichkeiten der Anwendung. Besonders im schulischen Alltag kann das Arbeiten, Entwickeln und Forschen mit dem Microcontroller hervorragend integriert werden, sei es in speziellen Unterrichtsformen (z. B. fächerübergreifend, fächerverbindend, interdisziplinär oder projektorientiert) oder in einem Spezialfach (z. B. Informatik, Programmieren, Elektrotechnik, ...) – Anwendungen in der Arduino-Umwelt fördern die Methodenkompetenz, erhöhen die Fachkompetenz und machen noch dazu sehr viel Spaß.

5. Literaturverzeichnis

5.1. Bücher

[BAL05] Balzert, H.: Lehrbuch der Objektmodellierung – Analyse und Entwurf mit der UML 2, 2. Auflage, Spektrum Akademischer Verlag, 2005

[BAN08] Banzi, M.: Getting started with Arduino, O'Reilly, 2008

[BRU10] Brühlmann, T.: Arduino Praxiseinstieg, mitp – eine Marke der Verlagsgruppe Hüthig Jehle Rehm GmbH, 2010

[DEI00] Deimel, F. / Hasenzagl, A. / Krikava, F. / Ruhswurm, H. / Seiser, J.: Grundlagen der Elektronik, Band 1, 10. Auflage, R. Oldenbourg GmbH, 2000

[GUD08] Gudjons, H.: Pädagogisches Grundwissen, 10. Auflage, Julius Klinkhardt, 2008

[HAR94] Harenberg Kompaktlexikon in 5 Bänden, Band 1; Zweite, teilweise überarbeitete und aktualisierte Auflage, Oktober 1994

[HUB07] Hubwieser, P.: Didaktik der Informatik, 3. Auflage, Springer-Verlag GmbH, 2007

[HUE06] Hüttenegger, G.: Open Source Knowledge Management, Springer-Verlag Berlin Heidelberg, 2006

[JAE06] Jaeger, Dr. T. / Metzger, Dr. M.: Open Source Software – Rechtliche Rahmenbedingungen der Freien Software, 2. Auflage, Verlag C. H. Beck, 2006

[KRO05] Kron, Friedrich W.: Grundwissen Didaktik, 5. Auflage, Ernst Reinhardt Verlag, 2005

[ODE10] Odendahl, M. / Finn, J. / Wenger, A.: Arduino – Physical Computing für Bastler, Designer und Geeks, 2. Auflage, O'Reilly, 2010

[REC06] Rechenberg P. / Pomberger G.: Informatik-Handbuch, 4. aktualisierte und erweiterte Auflage, Carl Hanser Verlag, 2006

[THE04] Themelidis, M.: Open Source – Die Freiheitsvision der Hacker, Verlag: Books on Demand GmbH, 2004

5.2. Artikel

[PAP01] **Integrating microcontrollers in undergraduate curriculum**
Wiliam Albrecht / Paul Bender / Kay Kussmann, McNeese State University, Published in „Journal of Computing Sciences in Colleges“, Volume 27 Issue 4, April 2012, Pages 45-52

[PAP02] **Arduino based projects in the computer science capstone course**
Paul Bender / Kay Kussmann, McNeese State University, Published in „Journal of Computing Sciences in Colleges“, Volume 27 Issue 5, May 2012, Pages 152-157

[PAP03] **Hands-on computing with Arduino**
John Vaughn, Hobart and William Smith Colleges, Published in „Journal of Computing Sciences in Colleges“, Volume 27 Issue 6, June 2012, Pages 105-106

[PAP04] **Using Arduino for introductory programming courses**
J. Dean Brock / Rebecca F. Bruce / Susan L. Reiser, UNC Asheville, Published in „Journal of Computing Sciences in Colleges“, Volume 25 Issue 2, December 2009, Pages 129-130

[PAP05] **Bluetooth based home automation system using cell phone**
R. Piyare / M. Tazil, Dept. of Electr. & Electron. Eng., Fiji Nat. Univ., Suva, Fiji, Published in „Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium“, June 2011, Pages 192-195

[PAP06] **Hand gesture based remote control for home appliances: Handmote**
U. V. Solanki / N. H. Desai, G.H. Patel Coll. of Eng. & Technol., Gujarat Technol. Univ., Vallabh, India, Published in „Information and Communication Technologies (WICT), 2011 World Congress“, December 2011, Pages 419-423

[PAP07] **Wireless network for health monitoring: heart rate and temperature sensor**
A. H. Kioumars / Tang Liqiong, Sch. of Eng. & Adv. Technol., Massey Univ., Palmerston North, New Zealand, Published in „Sensing Technology (ICST), 2011 Fifth International Conference“, December 2011, Pages 362-369

5.3. Internet-Links

adafruit industries [online], Verfügbar unter
http://www.adafruit.com
[Zugang am 26. April 2011]

„an open letter to hobbyists“ [online], Verfügbar unter
http://tranquileye.com/cyber/1976/gates_open_letter_to_hobbyists.html
[Zugang am 25. März 2011]

Arduino [online], Verfügbar unter
http://www.arduino.cc/
[Zugang am 25. Februar 2011]

Berufsbildende Schulen [online], Verfügbar unter
http://www.htl.at/de/home/lehrplaene.html
[Zugang am 19. April 2011]

Bundeskanzleramt Österreich – Rechtsinformationssystem [online], Verfügbar unter
http://www.ris.bka.gv.at/
[Zugang am 24. März 2011]

Bundesministerium für Unterricht, Kunst und Kultur [online], Verfügbar unter
http://www.bmukk.gv.at/schulen/unterricht/lp/lp_ahs_oberstufe.xml
[Zugang am 19. April 2011]

Freie TRIZ-Lernplattform für Erfinder & Entwickler [online], Verfügbar unter
http://www.triz.it/ebf/tct11.htm
[Zugang am 18. Juni 2011]

Funnel [online], Verfügbar unter
http://funnel.cc/
[Zugang am 28. April 2011]

Haekchen [online], Verfügbar unter
http://www.haekchen.at/haekchen/
[Zugang am 27. Juli 2011

HTL – Bildung mit Zukunft [online], Verfügbar unter
http://www.htl.at/de/htlat/lehrplaene.html
[Zugang am 20. Jänner 2012]

Interaction Design Institut Ivrea [online], Verfügbar unter
http://interactionivrea.org/en/index.asp
[Zugang am 20. Jänner 2012]

Interdisziplinäre Unterrichtsplanung (188.184), Vorlesung und Übung TU Wien, 2008

IT Wissen – Das große Online-Lexikon für Informationstechnologie [online], Verfügbar unter
http://www.itwissen.info/definition/lexikon/Erdung-GND-ground.html
[Zugang am 21. April 2011]

Lehrpläne der AHS Oberstufe [online],
http://www.bmukk.gv.at/schulen/unterricht/lp/lp_ahs_oberstufe.xml
[Zugang am 20. Jänner 2012]

Lehrplan der Höheren Lehranstalt für elektronische Datenverarbeitung und Organisation, Anlage 1 [online],
http://www.htl.at/fileadmin/content/Lehrplan/HTL/BGBl._Anlage_1_302-97.pdf
[Zugang am 20. Jänner 2012]

Lehrplan der Höheren Lehranstalt für elektronische Datenverarbeitung und Organisation [online],
http://www.htl.at/fileadmin/content/Lehrplan/HTL/HL_EDVO_Anlage_1.3.1_BGBl._382-98.pdf
[Zugang am 20. Jänner 2012]

modern device [online], Verfügbar unter
http://www.moderndevice.com
[Zugang am 26. April 2011]

Open Source [online], Verfügbar unter
http://www.opensource.org/
[Zugang am 25. März 2011]

Österreichische Professoren Union [online], Verfügbar unter
http://www.oepu-noe.at/recht/lp/index.htm
[Zugang am 19. April 2011]

PAPERduino's design [online], Verfügbar unter
http://lab.guilhermemartins.net/2009/05/06/paperduino-prints/
[Zugang am 26. April 2011]

Physical Computing Austria [online], Verfügbar unter
http://www.physicalcomputing.at
[Zugang am 26. April 2011]

Psychologische Begriffsbestimmungen [online], Verfügbar unter
http://www.stangl.eu/psychologie/definition/Didaktik.shtml
[Zugang am 21. Jänner 2012]

Qualitative vs Quantitative Methoden [online],
http://imihome.imi.uni-karlsruhe.de/nquantitative_vs_qualitative_methoden_b.html
[27. Februar 2012]

seed studio [online], Verfügbar unter
http://www.seedstudio.com
[Zugang am 26. April 2011]

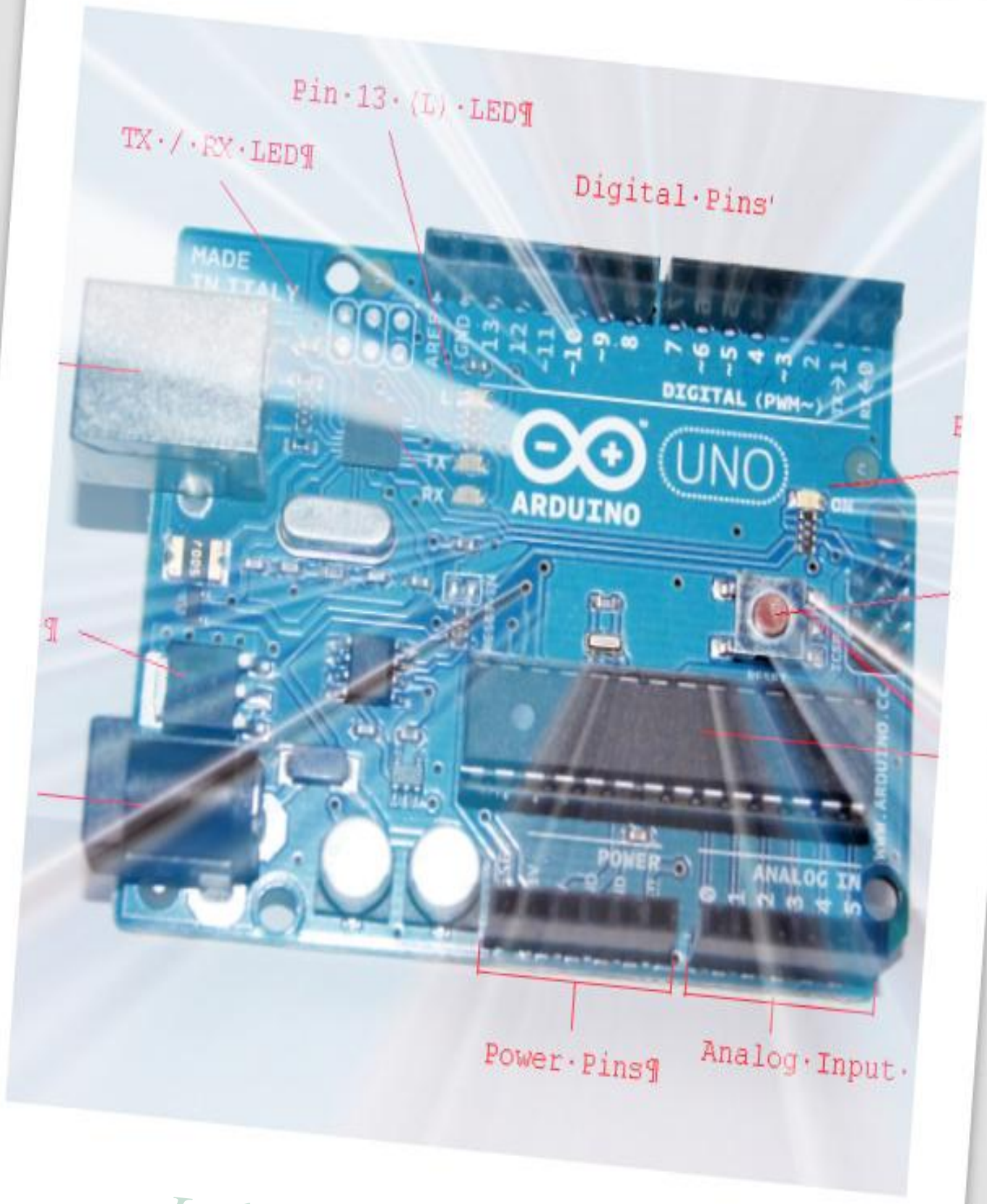
The Free Dictionary by Farlex [online], Verfügbar unter
http://www.thefreedictionary.com/
[Zugang am 29. April]

The Free Software Foundation [online], Verfügbar unter
http://www.fsf.org/
[Zugang am 25. März 2011]

Your UK source for Arduino & DIY electronics fun [online], Verfügbar unter
http://oomlout.co.uk/?p=189
[Zugang am 26. April 2011]

Anhang - Lehrbuch für Arduino

Im Anhang befindet sich das ausgearbeitete Lehrbuch für Arduino.



Lehrbuch für Arduino

Lehrbuch für Arduino

Version 1.0

Version 1.0

INHALTSVERZEICHNIS

1. Lehrbuch für Arduino

1.1. Einleitung

Dieses Lehrbuch soll allen Technik- und Elektronik-Begeisterten einen relativ einfachen und strukturierten Weg in die Arduino-Welt bereiten.

In Anlehnung an das Buch [BAL05]

„Lehrbuch der Objektmodellierung – Analyse und Entwurf mit der UML 2“

von Heide Balzert wird die Thematik mit einer Vielzahl an ausführlich beschriebenen Beispielen und Lernkontrollen dem Leser bzw. der Leserin nähergebracht. Dieses Lehr- und Lernbuch dient nicht nur als Selbststudium, sondern kann auch im klassischen Schulunterricht eingesetzt werden.

Die einzelnen Einheiten sind aufbauend und in Schwierigkeit und Umfang zunehmenend. Weiters wurde darauf geachtet, dass keine speziellen Vorkenntnisse für die Durcharbeitung des Lehrbuches Voraussetzung sind.

Das Lehrbuch gliedert sich in folgende Einheiten:

- *Lehreinheit 1*
  - Installation Hard- und Software, erstes Beispiel
- *Lehreinheit 2*
  - Allgemeines über Arduino
- *Lehreinheit 3*
  - Schalter, Ein- und Ausgänge, Serielle Schnittstelle
- *Lehreinheit 4*
  - Elektronisches Grundlagenwissen
- Beispiel – Lösungen
  - LEDsOnOff
  - TrafficLights

Theoretische Grundlagen, praktische Beispiele sowie kurze Überprüfungskapiteln versuchen einen ausgewogenen Mix zur Erarbeitung des Themas zu bieten.

1.2. Lehreinheit 1

Wissen, ...

- wie die Hardware / Software für Arduino installiert wird
- wie ein Programm erstellt wird
- wie mit der Entwicklungsumgebung gearbeitet wird

Verstehen / Erklären

- erklären können, wie ein Programm aufgebaut sein muss
- verstehen, wie ein Sketch an den Microcontroller übertragen wird
- verstehen, wie Variablen und Funktionen eingesetzt werden
- erklären können, wie eine IDE funktioniert und wie damit gearbeitet wird

1.2.1. Installation der Software / Hardware

Die nachfolgende Installationsanleitung wurde für den Microcontroller *Arduino Uno* und für das Betriebssystem *Windows XP* (Service Pack 3) erstellt.

1.2.1.1. Kurzanleitung

Die Kurzanleitung bietet einen Leitfaden bzw. eine kurze Zusammenfassung für versierte und fortgeschrittenere Anwender, um die Installation der Software / Hardware des Arduino durchzuführen.

> *Step 1: download der Software für den Arduino von http://www.arduino.cc/*
> *Step 2: entpacken unter C::\Programme oder C:\Program Files*
> *Step 3: Arduino über USB anstecken und Treiber installieren*
> *Step 4: Anschluss-Port des Arduino auslesen*

1.2.1.2. Step by Step

Die Installation der Software beginnt mit dem Herunterladen dieser mittels eines üblichen Web Browsers von der offiziellen Arduino Homepage.

> *Step 1: download der Software für den Arduino von http://www.arduino.cc/*

Auf der Homepage unter dem Menüpunkt *Download* kann die Open Source Software für Windows, Mac OS X und Linux bezogen werden.

In diesem Lehrbuch wird die Installation unter Windows (im speziellen Windows XP) etwas näher erläutert, daher zum Download auf den Link *Windows* klicken – es öffnet sich automatisch ein Download-Dialog des Browsers.

Es wird ein Archiv (.zip) mit dem Namen *arduino-00xx.zip* heruntergeladen - ‚xx' steht dabei für die aktuelle Versionsnummer der Software; im Mai 2011 war dies die Nummer 22.

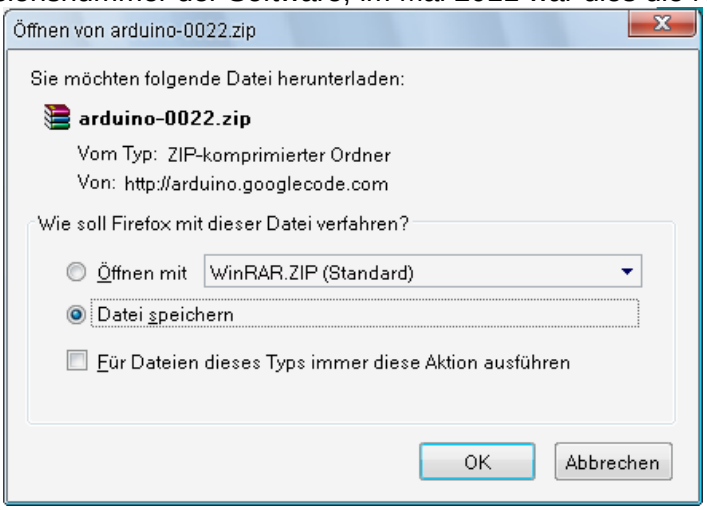


Abbildung 1: Download-Dialog der Arduino Software

Den Download-Dialog mit ‚OK' bestätigen. Das File ‚arduino-0022.zip' ist ca. 85 MB groß.

Zum Entpacken der Datei, kann einfach mit der Maus auf das heruntergeladene File doppelt geklickt werden.

Step 2: entpacken unter C:\Programme oder C:\Program Files

Als Empfehlung für eine reibungslose Verwendung der Software, sollte das Zip-File in das Verzeichnis *C:\Programme* oder *C:\Program Files* extrahiert werden.

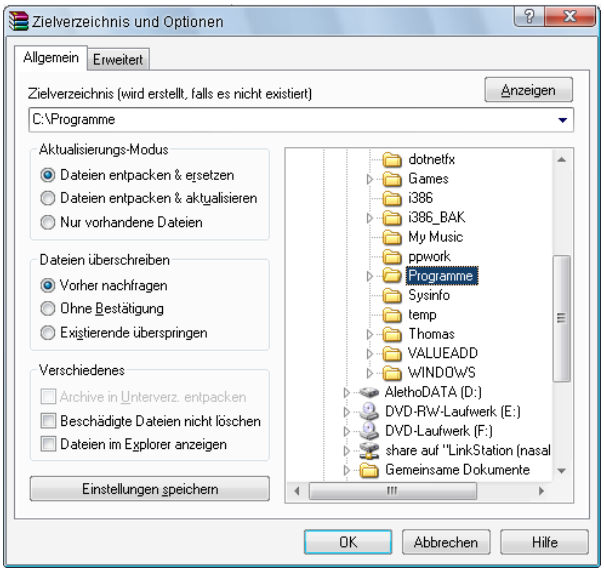


Abbildung 2: Datei-Auswahl-Dialog (Winrar)

Im entpackten Zustand umfasst die Arduino Software (Version 22) 5745 Elemente und hat eine Größe von etwa 232 MB.

Nun kann der Arduino Uno über ein herkömmliches USB Kabel an den PC angeschlossen werden.

Step 3: Arduino über USB anstecken und Treiber installieren

Der automatisch öffnende Dialog „Assistent für das Suchen neuer Hardware“ kann für die Installation des Treibers verwendet werden.

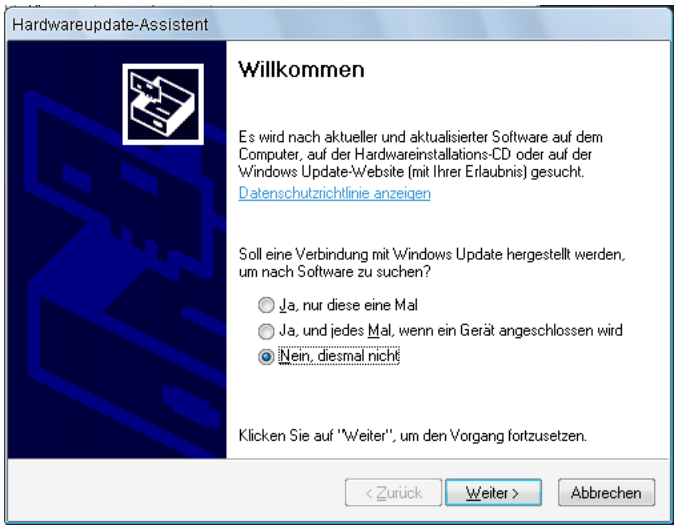


Abbildung 3: Hardwareupdate-Assistent

Nach dem Bestätigen mit *Weiter* kann der Punkt

‚Software automatisch installieren (empfohlen)’

ausgewählt werden.

Die erfolgreiche Installation des Arduino UNO meldet der Hardwareupdate-Assistent mit folgendem Dialog:

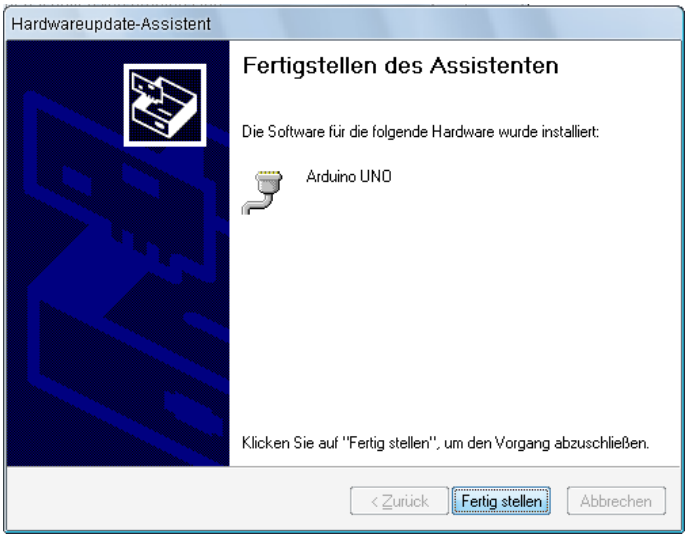


Abbildung 4: Hardwareupdate-Assistent

Falls bei der Hardwareinstallation die Meldung über das nicht bestehen des Windows-Logo-Test – welcher die Kompatibilität mit Windows XP verifiziert - angezeigt wird, muss die Schritte unter Installation des USB Treibers unter Windows ausführen.

Letzter Schritt in der Installation ist die Überprüfung, welcher Anschluss dem Arduino zugewiesen wurde. Dies kann über den *Geräte Manager* von Windows gemacht werden.

Step 4: Anschluss-Port des Arduino auslesen

Wie gelangt man zum Windows-Geräte-Manager?
Über das Menü

Start Einstellungen Systemsteuerung System

oder

Windowstaste + Pausetaste

öffnet sich der *Systemeigenschaften-Dialog* von Windows. Uber den Tab *Hardware* gelangt man zum Geräte-Manager.

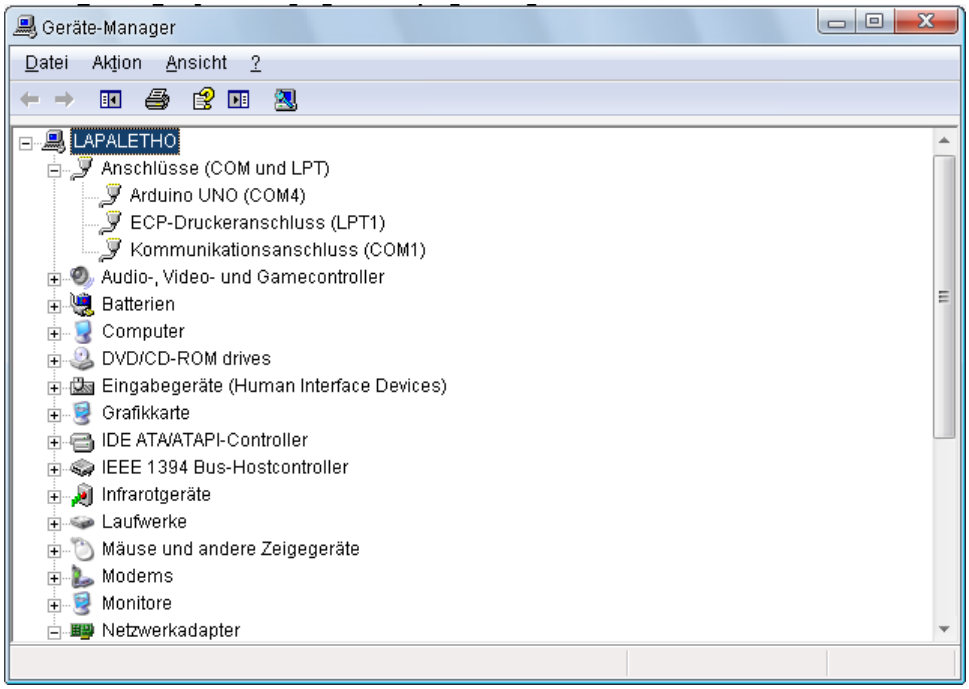


Abbildung 5: Geräte-Manager

Unter dem Punkt *Anschlüsse (COM und LPT)* kann nun das Anschluss-Port – rechts in Klammer neben dem Namen Arduino Uno – abgelesen werden.
Im obigen Beispiel ist dies COM4 – dieses wird bei der Installation der IDE (Integrated Development Environment) benötigt.

Hinweis
Damit der Arduino Mikrokontroller in dieser Liste aufscheint, muß er über das USB-Kabel an den PC bzw. Laptop angeschlossen sein.

1.2.1.2.1 Installation des USB Treibers unter Windows

Bei gewissen Windows Versionen kann der Fall eintreten, das der USB Treiber für den Arduino Uno aus dem Service Pack 3 nachinstalliert werden muss. Die Meldung über den fehlgeschlagenen Windows-Logo-Test ist ein Hinweis auf diese nicht vorhandene Komponente.

Der Hinweis bei der Hardwareinstallation lautet:

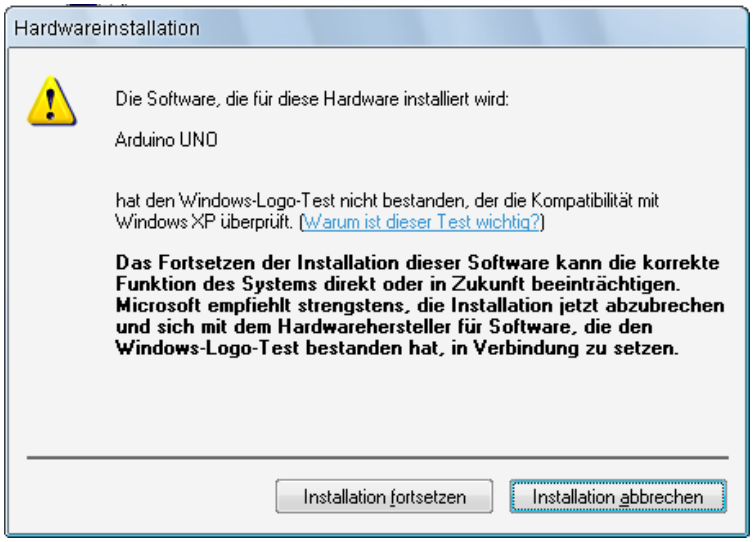


Abbildung 6: Hardwareinstallations-Assistent (1)

Nach dem Klicken auf den Button *‚Installation fortsetzen'* wird nach dem Verzeichnis bzw. der CD von Service Pack 3 verlangt.

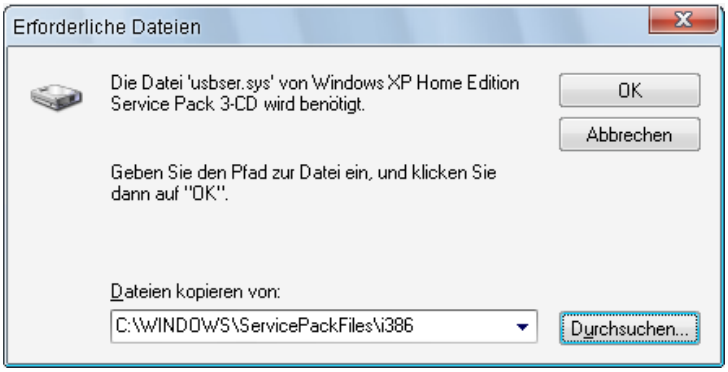


Abbildung 7: Hardwareinstallations-Assistent (2)

Nach der korrekten Angabe des Verzeichnisses bzw. Pfades erscheint die Bestätigung, dass die Installation erfolgreich war (siehe *Abbildung 4*).

1.2.2. Funktionscheck und erstes Beispiel

Wie weiß ich, ob das Board funktioniert?
Um die korrekte Funktionsweise des Microcontrollers zu überprüfen, ist die einfachste und bewährteste Methode, ein kleines Beispiel-Programm auszuführen.

Das Pendant in der Arduino-Welt zum ‚Hello World'-Beispiel-Programm aus der Java-Welt, ist das Blinkende-LED-Beispiel.

Dazu wird die zuvor installierte IDE (Integrated Development Environment) gestartet.

Mit Doppelklick auf <*Installationsverzeichnis*>*\arduino-022\arduino.exe*
startet eine leere Entwicklungsumgebung.

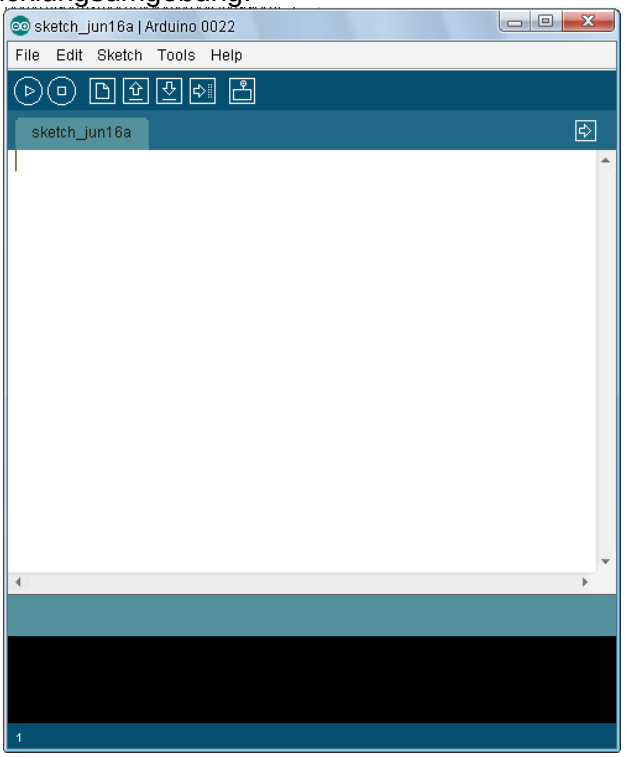


Abbildung 8: Arduino Entwicklungsumgebung (DIE)

Detaillierte Informationen zur Arduino IDE gibt es unter Arduino – Entwicklungsumgebung.

Bevor das vordefinierte Test-Beispiel ausgeführt werden kann, müssen zwei Einstellungen in der IDE vorgenommen werden:

1) Auswahl des verwendeten Boards
   a. Unter dem Menüpunkt ***Tools Board*** wird das benutzte Board ausgewählt (z. B. Arduino Uno)
2) Auswahl des seriellen Ports
   a. Unter dem Menüpunkt ***Tools Serial Ports*** wird das zugewiesene Port (welches im Geräte-Manager aufscheint) eingestellt (z. B. COM4)

Hinweis
Das Arduino Board muss für diese Einstellungen und während der Entwicklung über USB am PC bzw. Laptop angeschlossen sein.

Im Installationsverzeichnis von Arduino unter Examples befinden sich einige vordefinierte Programme – ein Programm für den Arduino wird üblicherweise als Sketch bezeichnet [BRU10].

Über den Menüpunkt *File Open* gelangt man in den Dateiauswahl-Dialog. Hier muss man zum Installationsverzeichnis von Arduino navigieren und unter dem Verzeichnis *examples\1.Basics\Blink* das File **Blink.pde** auswählen.

<Installationsverzeichnis>\arduino-022\examples\1.Basics\Blink\Blink.pde

Es öffnet sich ein weiteres Fenster mit dem Source-Code des Blink-Sketches.

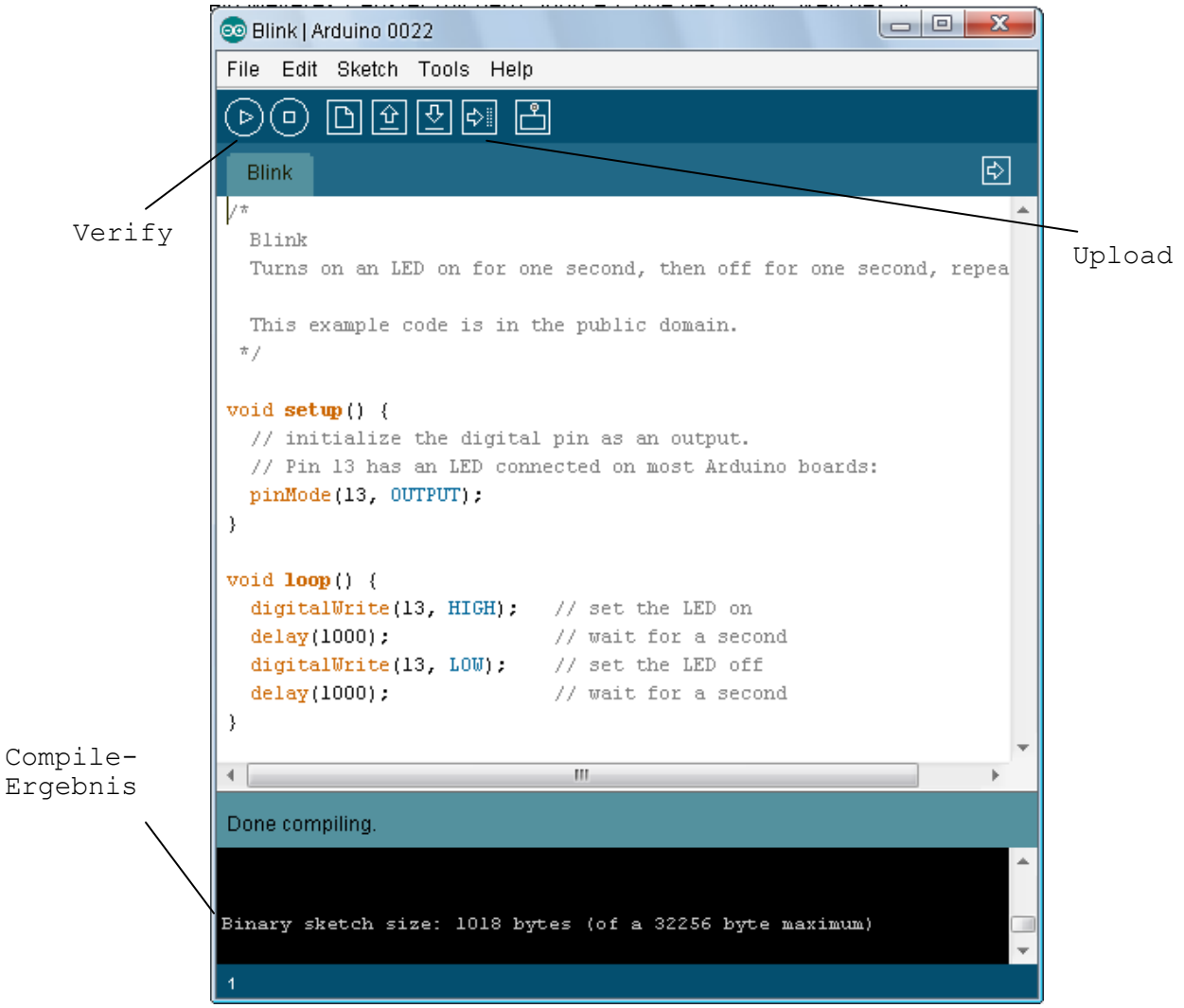


Abbildung 9: Arduino Blink-Sketch

Um das Programm auf den Arduino zu bringen, muss der Blink-Sketch upgeloadet werden – ‚Upload' bezeichnet die Übertragung des Sketches auf den Microcontroller.

Der Sketch wird mit *Verify* kompilieren (das Compile-Ergebnis findet man im unteren Bereich des IDEs) – Upload-Button drücken und der Sketch wird auf den Arduino transferiert.

Der Pin 13 (L) LED blinkt im Sekundentakt, somit ist die einwandfreie Funktion und korrekte Installation des Arduino Uno bestätigt. Gratulation!

1.2.3. Programmaufbau

Jedes Arduino-Programm – wie bereits erwähnt, wird als Sketch bezeichnet – hat einen einheitlichen Aufbau. Die einfachste Methode, die Sketches zu erlernen bzw. zu verstehen, ist die „Trial-and-Error" Methode – also die „Versuch-und-Irrtum" Methode[15].

Dazu werden Beispiel-Sketches ausgeführt und selbständig erweitert, um an der veränderten Reaktion des Microcontrollers zu lernen.

Die Programmiersprache für den Arduino ist von der bekannten und weitverbreiteten Entwicklungssprache C abgeleitet und bietet durch die relativ einfache Handhabe einen raschen Einstieg und Programmiererfolg.

Eine ausführliche „Language Reference" ist unter http://arduino.cc/en/Reference/HomePage [Zugang am 24.06.2011] verfügbar.

1.2.3.1. Struktur

Die einheitliche Struktur eines Sketches muss folgende zwei Funktionen (Programm-einheiten) enthalten:

1) void setup() { ... }
2) void loop() { ... }

Beispiel: Blink-Sketch

```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

1) void setup() { … }

In der Setup-Funktion werden Grundeinstellungen oder die Konfiguration der Seriellen-Schnittstelle vorgenommen. Diese Funktion wird einmalig bzw. bei jedem Reset des Microcontrollers durchlaufen.

Diese Funktion muss vorhanden sein, selbst dann, wenn die Funktion keine Anweisungen enthält.

2) void loop() { … }

In der Loop-Funktion erfolgen die Anweisungen zur Problemlösung. Wie der Name bereits vermuten lässt, wird diese Funktion in einer Schleife endlos durchlaufen. Die Abarbeitung erfolgt nach der Setup-Funktion und sie übernimmt die Aufgabe eines Hauptprogrammes in einem Sketch.

Diese Funktion muss vorhanden sein, selbst dann, wenn die Funktion keine Anweisungen enthält.

[15] Freie TRIZ-Lernplattform für Erfinder & Entwickler [online], Verfügbar unter http://www.triz.it/ebf/tct11.htm [Zugang am 18. Juni 2011]

1.2.3.2. Funktionen

Setup() und Loop() sind in Arduino-Programmen Pflicht-Funktionen – diese müssen vorhanden sein. Weiters können selbstentworfene Funktionen erstellt werden. Diese bieten den Vorteil, dass Programmteile gekapselt und wiederverwendet werden können.

Funktionen haben folgenden Aufbau:

```
Datentyp Funktionsname ( Datentyp Parametername, ... ) {
  Anweisungen
}
```

Beispiele

```
// Funktion ohne Rückgabewert, mit Parameter
void setLEDon( int pin ) {
  digitalWrite( pin,HIGH );
  delay(1000);
}
```

```
// Funktion mit Rückgabewert, ohne Parameter
float getTemperature( ) {
  float value;
  value = analogRead( 0 );
  return value;
}
```

Beispiel: Blink-Sketch mit Funktionen

Die Verwendung von Funktionen in diesem einfachen Beispiel dient nur der Veranschaulichung – Nutzen und Erfordernis werden außer betracht gelassen.

```
/* Blink Sketch
   schaltet das LED (13) im Abstand von einer Sekunde ein und wieder aus
*/
void setup() {
  pinMode(13, OUTPUT); // Pin 13 als Ausgang definieren
}

void loop() {
  setLEDon( 13 );  // LED einschalten
  setLEDoff( 13 ); // LED ausschalten
}

void setLEDon( int pin ) {
  digitalWrite( pin,HIGH ); // LED einschalten
  delay(1000); // eine Sekunde warten
}

void setLEDoff( int pin ) {
  digitalWrite( pin,LOW ); // LED ausschalten
  delay(1000); // eine Sekunde warten
}
```

Kommentare

Wie in Programmiersprachen üblich, ist die Verwendung von Kommentaren hilfreich und meist sehr sinnvoll. Bei der Arduino-Entwicklungsumgebung gibt es die typischen Kommentar-Möglichkeiten der Programmiersprache C / C++:

1) /* Kommentar */
    a. Kommentar kann über mehrere Zeilen verlaufen
2) // Kommentar
    a. Kommentar bis ans Ende der Zeile

1.2.3.3. Variablen und Typen

In einem Sketch können Werte zwischengespeichert werden, welches mittels Variablen erfolgt.

Variablen [BRU10] werden mit folgender Angabe erzeugt und mit einem Datentyp, einem Wert und einem Namen versehen – man spricht von ‚deklarieren' und ‚definieren':

```
Datentyp Variablenname = Wert;
```

Fehlt die Wertangabe, wird die Variable auf 0 gesetzt.

Beispiel

```
int aussentemp = 20; // integer- Variable (ganzzahlig) wird auf 20 gesetzt
```

Je nach Wertebereich kann eine Variable mit verschiedenen Datentypen vereinbart werden. Die wichtigsten in der Arduino-Entwicklung werden im Folgenden aufgeführt [BRU10]:

Datentyp	Wertebereich / Specherbedarf	Erklärungen / Beispiele
int	-32.768 bis 32.767 Speicherbedarf 2 Bytes	Ganzzahlige Werte int aussentemp = 20;
byte	0 bis 255 Speicherbedarf 1 Byte	Niedrige Werte bype dimmer = 125;
long	-2.147.483.648 bis 2.147.483.647 Speicherbedarf 4 Bytes	Ganzzahlige Zahlen long zaehler = 100000;
float, double	-3,4028235E+38 bis 3,4028235E+38 Speicherbedarf 4 Bytes	Werte mit Kommastellen float aussentemp = 20.55;
char	-128 bis 127 Speicherbedarf 1 Byte	Wert für die Speicherung eines Zeichens char buchstabe = 'A';
boolean	true / false Speicherbedarf 1 Bype	Wert für Ein / Aus, Ja / Nein boolean isOn = true;
string	Länge je nach Definition Speicherbedarf je nach Bedarf	Zeichenketten (z. B. Texte) string bezeichnung[] = 'Uno';
array	Array mit Größe gemäß Definition Speicherbedarf je nach Array-Größe	Werte von Variablen in Tabellenform int ports[] = {8,9,10,11};

Um die Lesbarkeit von Programmen [BRU10] zu gewährleisten, sollten sprechende Namen für die Variablen verwendet werden. Dazu sollten je nach Verwendung und Bedeutung die jeweiligen Bezeichnungen ausgewählt werden.

```
// gute Variablennamen
int zaehler = 1;
boolean isHigh = true;
float aussenTemp = 20.50;

// schlechte Variablennamen
int a = 1;
boolean b = true;
float c = 20.50;
```

1.2.3.4. if - Anweisung

Die if – Anweisung [BRU10] erlaubt eine bedingte Ausführung eines Programmcodes. Diese Anweisung bietet die Möglichkeit, Teile eines Programmes nur nach Auswertung einer Bedingung abzuarbeiten.

Diese Anweisung muss folgenden Aufbau haben:

```
if( <Bedingung) {
  // Anweisungen
}
```

Beispiel

```
//
if( aussentemp < 20 ) {
  Serial.print("It’s getting cold…");
}
```

Die if - Anweisung ist auch bekannt unter dem Namen „Wenn-Dann“-Bedingung, d. h. wenn eine Bedingung eintritt, dann führe aus.

Weiters gibt es die Möglichkeit, in der Anweisung einen ‚else’-Fall abzubilden:

```
//
if( aussentemp < 20 ) {
  Serial.print("It’s getting cold…");
} else {
  Serial.print("It’s still warm…");
}
```

Wenn der Wert ‚aussentemp’ kleiner als 20 ist, dann schreibe den Text „It’s getting cold...“, wenn nicht (else), dann schreibe den Text „It’s still warm...“.

1.2.3.5. Schleifen

In der Programmierung dienen Schleifen [BRU10], auch ‚Loops’ genannt, dem wiederholen von Programmcode. Es gibt in der Arduino-Entwicklung drei verschiedene Schleifen:

1) for – Schleife
2) while – Schleife
3) do – while – Schleife

for-Schleife

```
for( initialisierung; bedingung; erweiterung ) {
  // Anweisungen
}
```

Beispiel

```
int i;
for( i=0; i < 100; ++i ) {
  Serial.print(i);
}
```

while-Schleife

```
while( bedingung ) {
  // Anweisungen
}
```

Beispiel

```
int i = 0;
while( i < 100 ) {
  Serial.print(i);
  ++i;
}
```

do-while-Schleife

```
do {
  // Anweisungen
} while ( bedingung ) {
```

Beispiel

```
int i = 0;
do {
  Serial.print(i);
  ++i;
} while( i < 100 )
```

1.2.4. Arduino –Entwicklungsumgebung

Die Entwicklungsumgebung (IDE, Integrated Development Environment) ist in Java geschrieben und basiert auf der Entwicklungssprache Processing[16].

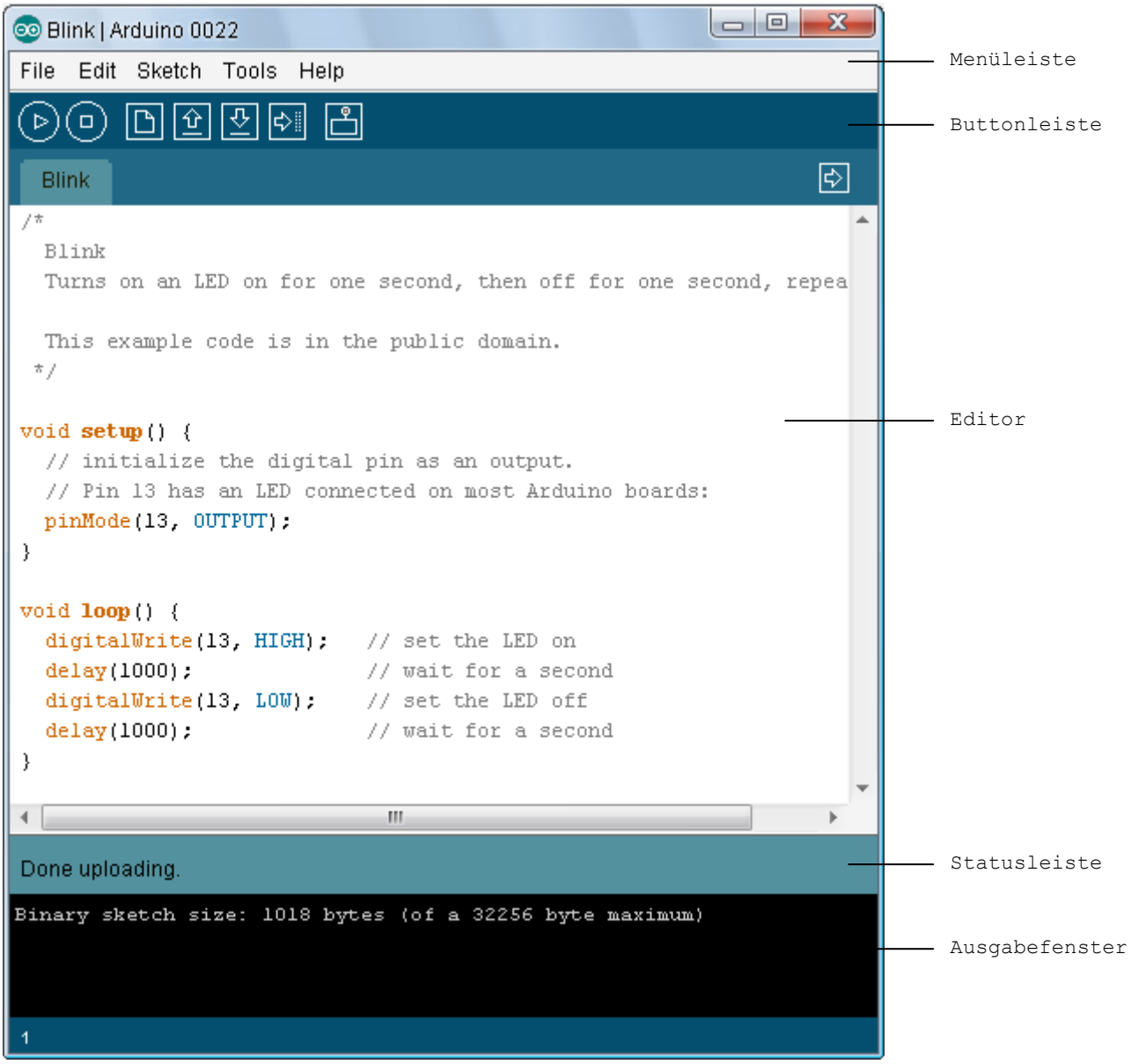


Abbildung 10: Arduino IDE

Menüleiste
Die einzelnen Menüpunkte der Menüleiste dienen der allgemeinen Organisation von Sketches und Verwaltung der Grundeinstellungen der IDE.

[16] Arduino [online], Verfügbar unter http://arduino.cc/en/Main/Software [Zugang am 16. Juni 2011]

Buttonleiste

Verify
Mit dem Button ‚Verify' wird der aktuelle Programmcode kompiliert, d. h. es wird die Richtigkeit der Syntax im Sketch überprüft. In der Statuszeile und im Ausgabefenster werden über Erfolg bzw. Misserfolg der Kompilierung Meldungen angezeigt. Zusätzlich wird im Ausgabefenster – nach erfolgreicher Verifizierung – die Größe des aktuellen Programmcodes angezeigt.

Stop
Mit einem Klick auf den ‚Stop' Button wird die Ausgabe und Anzeige im seriellen Monitor unterbrochen.

New
Mit dem Button ‚New' wird ein neuer, leerer Editor geöffnet. Ist der aktuelle Programmcode noch nicht gespeichert, wird eine dementsprechende Meldung angezeigt.

Open
Über den Button ‚Open' können die Sketches, welche im ‚Sketchbook' abgelegt wurden, geöffnet werden. Das Sketchbook ist das Standardverzeichnis, in dem die Eigenentwicklungen abgelegt werden. Dieses Verzeichnis kann über die Einstellung der ‚Sketchbook location' unter dem Menüpunkt *File Preferences* frei eingestellt werden.
Weiters kann über den ‚File-Dialog' in ein beliebiges Verzeichnis gewechselt werden um einen Sketch zu öffnen.

Save
Mit einem Klick auf den ‚Save' Button wird der aktuelle Programmcode gespeichert.

Upload
Mit dem Button ‚Upload' wird der aktuelle Sketch kompiliert und anschließend auf den Microcontroller übertragen. Eine erfolgreiche Kompilierung wird mit einer Meldung im Ausgabefenster bestätigt.

Am angeschlossenen Microcontroller lässt sich durch das Aufblicken des Receive LED (RX) und Transmit LED (TX) die Übertragung erkennen, danach wird der Programmcode am Microcontroller ausgeführt.

Serial Monitor
Mit einem Klick auf den Button ‚Serial Monitor' öffnet sich ein neues Fenster, in welchem Daten an die serielle Schnittstelle geschickt werden können.
Es ist ebenfalls möglich, diverse Daten, Zustände oder Variablenwerte vom Microcontroller an den seriellen Port zu schicken, welche danach im ‚Serial Monitor' angezeigt werden. Diese Darstellung der Daten kann besonders in der Entwicklungsphase eines Programmcodes hilfreich sein.

Editor
Der Editor dient der Entwicklung der Arduino-Programme. Durch die Syntax-Erkennung (Syntax-Highlighting), welches definierte Variablen- und Funktionsnamen farblich hervorhebt, unterstützt er den Entwickler bei der Programmcodeerstellung. Zusätzlich werden zusammenhängende Klammernpaare dem Programmierer im Editor angezeigt.

Statusleiste
In der Statusleiste werden Meldungen sowie Fehlermeldungen zum aktuellen Sketch während und nach dem Kompilieren angezeigt.

Ausgabefenster
Das Ausgabefenster dient der detaillierten Ausgabe von Meldungen bzw. Fehlermeldungen zum aktuellen Sketch während und nach der Kompilierung.

1.2.5. Lernziele

Im Folgenden werden typische Lernziel-Fragen bzw. Aufgaben angeführt.

Fragen / Aufgaben

1) Installation unter Windows durchführen.

2) Welche Schritte sind notwendig, um den angeschlossen Microcontroller in der IDE zu setzen?
    a. Welche Einstellungen sind vorzunehmen.

3) Wie überprüft man die korrekte Funktionsweise eines angeschlossenen Microcontrollers?

4) Wie muss ein Sketch aufgebaut sein?

5) Struktur der Deklaration von Variablen und Funktionen erklären.

6) Beispiel: Blinkendes LED

1.3. Lehreinheit 2

Wissen, ...

- woher der Name Arduino stammt
- für was/wen Arduino geschaffen wurde
- welche verschiedene Arduino vorhanden sind

Verstehen / Erklären

- Erklären können, warum und wozu Arduino entwickelt wurde
- Erklären können, was „Prototyping“ bedeutet
- Verstehen, was „Tinkering“ bedeutet

1.3.1. Arduino Uno[17]

Der Arduino Uno ist das neueste Board der Arduino-Mikrocontroller-Reihe [BRU10] – das „Uno“ in der Namensgebung, was bekanntlich in der italienischen Sprache für „1“ steht, soll ein Hinweis auf die aktuelle Version / Release des Microkontrollers sein =Version 1.0.

Werfen wir einen Blick auf das Arduino Uno Board...

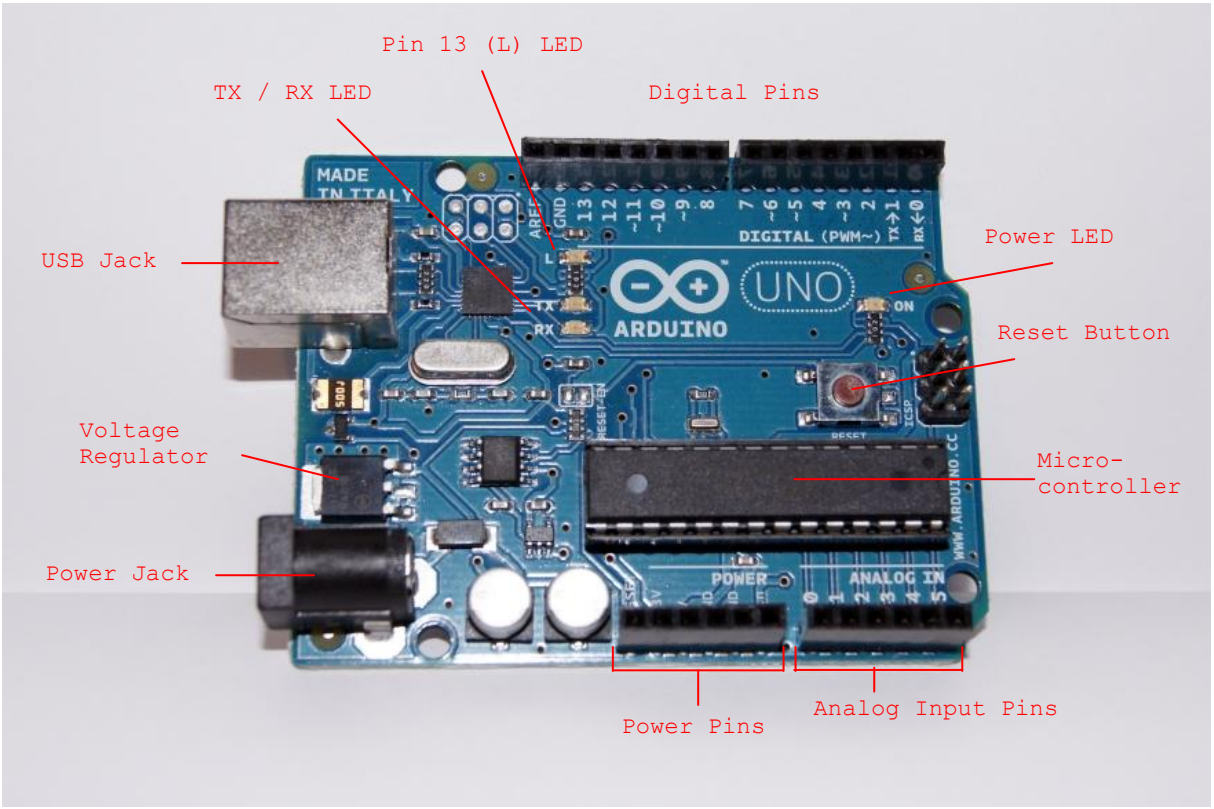


Abbildung 11: Arduino Uno

[17] Arduino – Hardware, Verfügbar unter http://arduino.cc/en/Main/Hardware [Zugang am 20. April 2011]

Digital Pins
An der Oberseite des Boards befinden sich die digitalen Ein- und Ausgänge, welche auf zwei 8-polige Buchsenleisten verteilt sind – somit sind 14 digitale Ein- und Ausgänge vorhanden. (Port 0-13).

Pin 13 (L) LED
Das Onboard LED – gekennzeichnet mit L – ist ein, speziell für den Port 13 (digital Pin), vorgesehens LED.

TX / RX LED
Diese beiden LEDs lassen erkennen, ob Daten vom Board empfangen (RX, received) oder gesendet (TX, transmitted) werden.

USB Jack
Der USB Stecker (USB Typ B) bietet einerseits eine Schnittstelle für die Programmierung und Kommunikation mit dem Board und andererseits die Möglichkeit, den Arduino über einen USB Port mit Strom zu versorgen.

Voltage Regulator
Mit Hilfe des Spannungsreglers wird die Betriebsspannung beim Arduino Board reguliert.

Power Jack
Der Power Stecker bietet einen Anschluss für eine externe Stromversorgungsquelle. Typbezeichnung: 2.1-mm-Stromstecker, externe Versorgungsspannung 6-20 V DC

Power Pins
Stromversorgung für den Arduino
- USB Typ B (z. B. über PC / Laptop)
- Externe Stromversorung
    - AC/DC Adapter
    - Batterie

Die Strom Pins sind
- Vin
    - Volt in – Die Eingangsspannung des Arduino Boards, falls eine externe Stromquelle verwendet wird. Über diesen Pin können auch Komponenten, welche mit dem Board verbunden werden können, mit Strom versorgt werden.
- 5V
    - Eine 5 Volt große Stromquelle, welche für den Microcontroller und andere Komponenten verwendet wird.
- 3,3V
    - Eine 3,3 Volt große Stromquelle, welche vom on-board Regulator erzeugt wird.
- GND
    - Die Abkürzung GND (Ground) ist der englische Begriff für Masse.
      „Die Erdung oder Masse (GND) dient der Potentialfreiheit von Geräten, Anlagen und Systemen. Sie stellt eine elektrische Verbindung mit dem Erder und damit dem Erdreich dar. Durch diese Verbindung wird ein Berührungsschutz für Personen erreicht (VDE 100) und ein eindeutiges Bezugspotential für die Schirmung der aktiven und passiven Komponenten.“[18]

[18] IT Wissen – Das große Online-Lexikon für Informationstechnologie [online], Verfügbar unter http://www.itwissen.info/definition/lexikon/Erdung-GND-ground.html [Zugang am 21. April 2011]

Analog Input Pins

An der rechten Unterseite des Boards befinden sich die analogen Eingänge, welche auf einer 6-poligen Buchsenleiste verteilt sind (Port A0-A5).

Ein analoges Signal kann – im Gegensatz zum digitalen – einen beliebigen Wert in einem Spannungsbereich annehmen. Beim Arduino liegt der Spannungsbereich zwischen 0 und 5 Volt. Der Arduino besitzt einen internen A/D-Wandler (Analog/Digital-Wandler), welcher Werte von Sensoren mit einer Auflösung von 10 Bits digitalisiert.

Was heißt das? Nehmen wir an, man verwendet einen Temperatursensor, der von 0 bis 20° messen kann und er wird an den analogen Eingang A0 angeschlossen. Falls nun der Wert von A0 ausgelesen wird und dieser z. B. 500 digitalisiert (!) beträgt, kann man auf die tatsächliche Temperatur rückrechnen:

- ⇨ maximal mögliche Temperatur * gelesener Wert / maximal möglicher Wert
- ⇨ 20 * 500 / 1024 = 9,77°

Der maximal mögliche Wert ergibt sich durch die Auflösung des 10 Bit Wandlers (2^{10} = 1024).

Microcontroller

Im Arduino Uno wird ein Mikrokontroller der Firma Atmel eingesetzt – Bezeichnung: ATmega328.

Im Microcontroller ist standardmäßig das Bootloader-Programm gespeichert. Dieses dient dazu, Arduino-Programme (sogenannte Sketches) in den Speicher des Microcontrollers zu laden.

Reset Button

Der Reset Button startet das Programm (bei Arduino auch als Sketch bezeichnet), welches im Speicher ist, neu. Es wird nichts gelöscht.

Power LED

Das Power LED leuchtet, wenn das Board mit Strom versorgt wird.

Technische Daten[19]

Mikrocontroller	ATmega328
Betriebsspannung	5V
Versorgungsspannung: (empfohlen)	7-12V
Versorgungsspannung: (limit)	6-20V
Digitale Ein/Ausgänge	14 (6 davon können als PWM verwendet werden)
Analoge Eingänge	6
Strom per Ein/Ausgangs Pin	40 mA
Strom bei 3.3V Pin	50 mA
Flash Memory	32 KB, (0,5 KB für Bootloader)
SRAM	2 KB
EEPROM	1 KB
Taktung	16 MHz

[19] Physical Computing Austria, Verfügbar unter http://www.physicalcomputing.at [Zugang am 20. April 2011]

Die Rückseite ist weniger spektakulär…

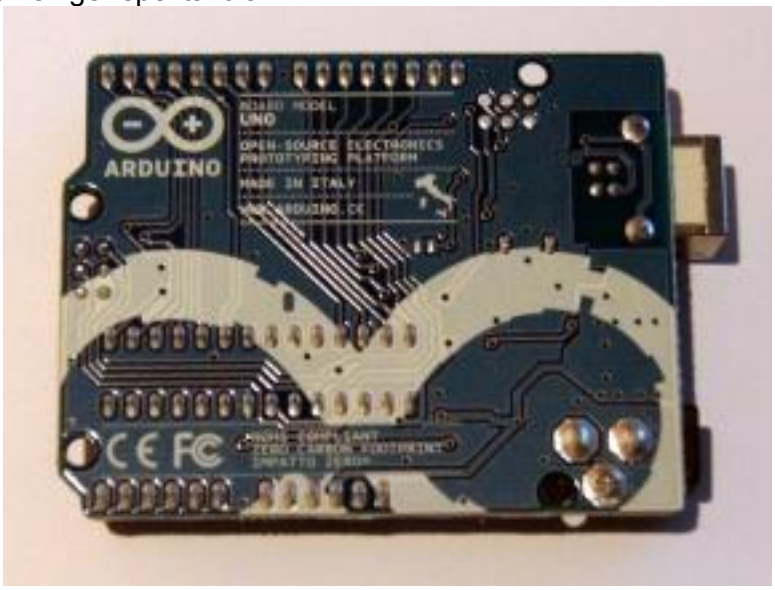


Abbildung 12: Arduino Uno (Rückseite)

Die Besonderheit des Arduino Uno ist – im Gegensatz zu seinen Vorgängermodellen (siehe 1.2.3. Boards), dass er einen modifizierten USB-auf-Serial Chip besitzt (Bezeichnung „ATMega8U2“), welcher eine einfache Verbindung – ohne zusätzliche Treiber-Installation – zu einem PC ermöglicht.

1.3.2. Arduino Plattform

Die Arduino Plattform besteht aus zwei Komponenten:

a) Hardware – Komponente
b) Software – Komponente

Hardware - Komponente

Diese Komponente wurde im Kapitel „1.2.1 Arduino Uno“ bereits beschrieben.

Die Hardware ermöglicht mit Ein- und Ausgängen die Kommunikation mit der Außenwelt – über die Eingänge können z. B. Daten von Sensoren ausgelesen werden und über die Ausgänge können diverse elektronische Bauelemente wie z. B. Leuchtdioden oder Motoren gesteuert werden.

Einer der größten Vorteile der Hardware des Arduino Boards ist sicherlich, der äußerst kostengünstige Anschaffungspreis sowie die relativ einfache Programmierung (inkl. der frei verfügbaren Software.

Preis-Überblick[20]

Arduino Uno	€ 26,20
Einsteiger-Kit mit Arduino Uno	€ 62,--
Einsteiger-Kit V2 mit UNO	€ 60,--

Die Einsteiger-Kits sind besonders für interessierte Anfänger ein hervorragender Einstieg in die Arduino-Welt. Sie bieten genügend Einzelteile, um erste Anwendungen mühelos

[20] Physical Computing Austria [online], Verfügbar unter http://www.physicalcomputing.at [Zugang am 26. April 2011]

umsetzen zu können – von Drahtbügeln über Breadboard bis zum Elektronik Set ist in diesem Set enthalten.

Software - Komponente
Der Software-Teil ist frei verfügbar und kann über die Homepage „www.arduino.cc“[21] kostenlos bezogen werden. Die Software ist eine sogenannte IDE (Integrated Development Environment) und dient der Entwicklung von Programmen sowie der Übertragung dieser auf den Microcontroller. Die IDE basiert auf der ebenfalls frei verfügbaren Programmiersprache „Processing“ (www.processing.org).

Die Software-Programme für den Microcontroller werden als „Sketch“ bezeichnet. Die Sketches werden seriell via USB-Verbindungskabel an das Board übertragen. Ist das Programm im Speicher des Arduino Boards einmal vorhanden, kann dieses als eigenständiger Mini-Computer agieren. Die Entwicklungsumgebung besitzt auch die Möglichkeit, über einen seriellen Monitor, Daten „sichtbar“ zu machen. Dieses kann beim Testen eines Sketches äußerst hilfreich sein.

1.3.3. Boards

Als Boards [BRU10] werden die verschiedenen, je nach Anwendungsfall aufgebauten, Varianten des Arduino bezeichnet.

Das derzeit aktuellste Standard-Board - Stand April 2011 - ist der Arduino Uno. Zusätzlich zu den Standard-Boards gibt es eine Vielzahl an kompatiblen Ausführungen (Clones), welche bekannt sein sollten:

1) Boarduino
    a. http://www.adafruit.com [Zugang am 26. April 2011]
2) Seeeduino
    a. Seeestudio, http://www.seedstudio.com [Zugang am 26. April 2011]
3) Bar Bone Boards
    a. http://www.moderndevice.com [Zugang am 26. April 2011]

Da die Anzahl der Clones ständig wächst, wurde hier nur eine kurze Liste angeführt – weitere sind unter http://www.arduino.cc/playground/Main/SimilarBoards [Zugang am 26. April 2011] zu finden.

Für technisch versierte bzw. erfahrene Anwender gibt es auch die Möglichkeit, ein komplettes Arduino Board selbst aufzubauen. Dies ist mitunter möglich, da auch der verbaute Microcontroller ATmega328 der Open-Source-Lizenz unterliegt.

Zum Aufbau eigener Boards stehen die CAD-Daten (Computer Aided Design) auf der Arduino Homepage zur Verfügung. Kreativste Projekte sind unter anderem

1) Breadboard Based Arduino Compatible (BBAC)
    a. ... ist gedacht für den Aufbau auf einem Steckbrett, http://oomlout.co.uk/?p=189 [Zugang am 26. April 2011]
2) Paperduino
    a. Merkmal - die Leiterplatte ist ein Stück Karton, http://lab.guilhermemartins.net/2009/05/06/paperduino-prints/ [Zugang am 26. April 2011]

[21] Arduino [online], Verfügbar unter http://www.arduino.cc/ [Zugang am 27. April 2011]

1.3.3.1. Duemilanove

Dieses Board ist der direkte Vorgänger vom Arduino Uno und wurde benannt nach dem Jahr seiner Einführung. Die Ausstattung der beiden Boards ist sehr ähnlich - die auffallendsten Unterschiede sind der Microcontroller – im Duemilanove arbeitet ein ATmega168, wobei auch eine Variante mit dem ATmega328 aufgebaut wurde – sowie der im Uno weiterentwickelte USB-Controller ATmega8u2.

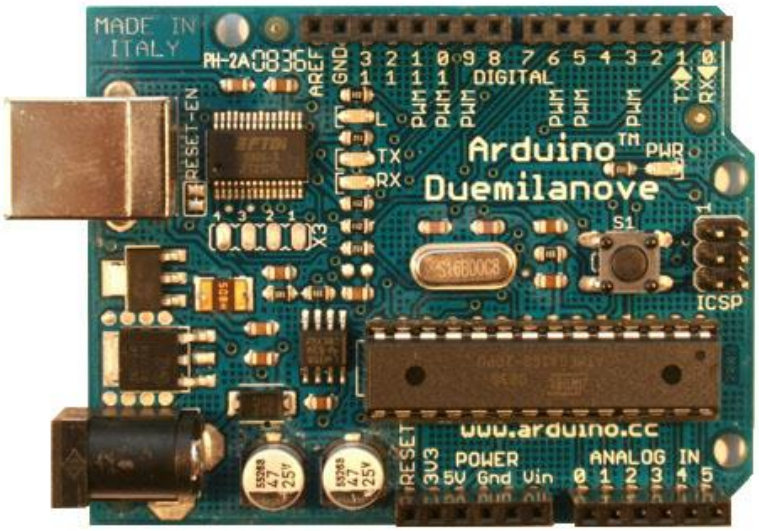


Abbildung 13: Arduino Duemilanove [Quelle: http://www.arduino.cc]

Technische Daten[22]

Mikrocontroller	ATmega168 / ATmega328
Betriebsspannung	5V
Versorgungsspannung: (empfohlen)	7-12V
Versorgungsspannung: (limit)	6-20V
Digitale Ein/Ausgänge	14 (6 davon können als PWM verwendet werden)
Analoge Eingänge	6
Strom per Ein/Ausgangs Pin	40 mA
Strom bei 3.3V Pin	50 mA
Flash Memory	16 KB / 32 KB, (2 KB für Bootloader)
SRAM	1 KB / 2 KB
EEPROM	512 Bytes / 1 KB
Taktung	16 MHz

1.3.3.2. Diecimila

Das Vorgängermodell vom Arduino Duemilanove unterscheidet sich von diesem nur in einer Eigenschaft: der Stromversorgung.

Das Diecimila (italienisch für Zehntausend) erkennt nicht automatisch die angeschlossene Stromquelle – hier muss mittels Schalter (Jumper) eingestellt werden, ob die Stromversorgung über USB Stecker (USB Jack) oder AC Adapter (Power Jack) erfolgt.

[22] Arduino [online], Verfügbar unter http://www.arduino.cc/ [Zugang am 27. April 2011]

1.3.3.3. Mega

Das Mega ist das Power-Board unter den verschiedenen Arduino-Varianten. Auffälligster Unterschied zu den übrigen Boards ist die große Anzahl an digitalen Ein- /Ausgängen sowie der analogen Eingänge und das vorhanden sein von mehreren seriellen Schnittstellen.

Das Einsatzgebiet des Mega sind naturgemäß Projekte, in denen jede Menge Ein- und Ausgänge verwendet werden müssen (z. B. Schaltung mit sehr vielen LED Anschlüssen) oder externe Porterweiterungen zu aufwändig sind.

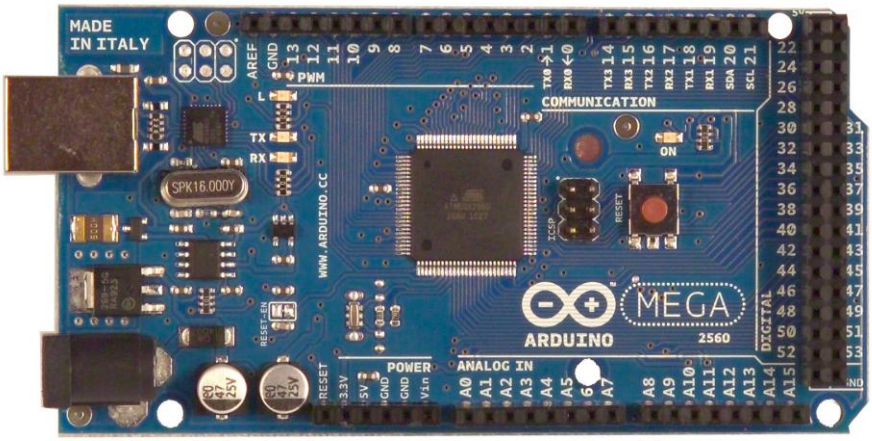


Abbildung 14: Arduino Mega [Quelle: http://www.arduino.cc]

Technische Daten[23]

Mikrocontroller	ATmega2560
Betriebsspannung	5V
Versorgungsspannung: (empfohlen)	7-12V
Versorgungsspannung: (limit)	6-20V
Digitale Ein/Ausgänge	54 (14 davon können als PWM verwendet werden)
Analoge Eingänge	16
Strom per Ein/Ausgangs Pin	40 mA
Strom bei 3.3V Pin	50 mA
Flash Memory	256 KB, (8 KB für Bootloader)
SRAM	8 KB
EEPROM	4 KB
Taktung	16 MHz

1.3.3.4. Nano

Dieses Arduino-Board – die Abmessungen betragen 18,5 x 43 mm – gehört zu den kleinsten Varianten. Der Leistungsfähigkeit zollt es deshalb aber keinen Tribut - einzig und allein auf den DC Versorgungsstecker wird verzichtet und der USB B Anschluss vom z. B. Arduino Duemilanova ist durch einen Mini-USB B Anschluss ersetzt.

[23] Arduino [online], Verfügbar unter http://www.arduino.cc/ [Zugang am 27. April 2011]

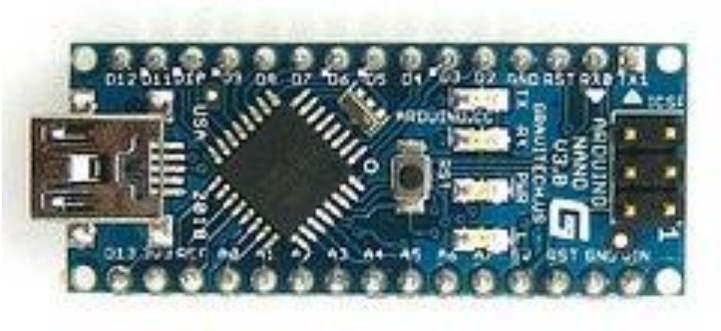


Abbildung 15: Arduino Nano [Quelle: http://www.arduino.cc]

Die technischen Daten entsprechen der Aufstellung vom Arduino Duemilanova.

1.3.3.5. Mini

Der Mini ist der kleinste Vertreter der Arduino – Serie. Er ist hervorragend für den Einsatz auf Breadboards ausgelegt und optimal für Einsatzgebiete, in denen Platz Mangelware ist.

Um diesen Microcontroller programmieren zu können, muss ein Mini USB Adapter angeschlossen werden, dadurch ermöglicht man, wie gewohnt, eine Verbindung zum PC.

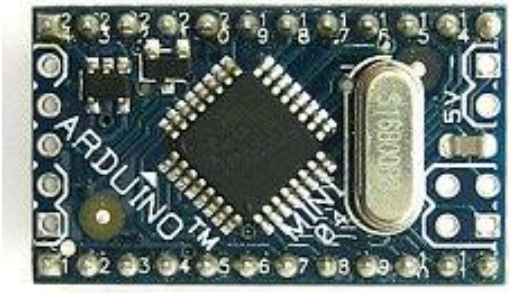


Abbildung 16: Arduino Mini [Quelle: http://www.arduino.cc]

Technische Daten[24]

Mikrocontroller	ATmega168
Betriebsspannung	5V
Versorgungsspannung	7-9V
Digitale Ein/Ausgänge	14 (6 davon können als PWM verwendet werden)
Analoge Eingänge	8
Strom per Ein/Ausgangs Pin	40 mA
Flash Memory	16 KB / 32 KB, (2 KB für Bootloader)
SRAM	1 KB
EEPROM	512 Bytes
Taktung	16 MHz

1.3.3.6. BT (Bluetooth)

Der Arduino BT ist eine der Wireless-Varianten der Microcontroller-Familie. Bei diesem Board wurde die USB Schnittstelle eingespart – Kontakt mit der Außenwelt kann nur über die integrierte Bluetooth-Schnittstelle aufgenommen werden.

[24] Arduino [online], Verfügbar unter http://www.arduino.cc/ [Zugang am 27. April 2011]

Wegen diesem speziellen Wireless-Interface gehört der Arduino BT zu der teureren Variante der Microcontroller-Boards. Das verbaute Bluetooth-Modul ist das Bluegiga WT11 – dieses ermöglicht eine kabellose serielle Kommunikation (es ist aber nicht kompatibel mit Bluetooth-Headsets). Der WT11 ist speziell konfiguriert, um im Arduino Board seine Dienste zu verrichten – der standardmäßige Name ist gesetzt auf ARDUINOBT und das Passcode auf 12345.

Abbildung 17: Arduino BT [Quelle: http://www.arduino.cc]

1.3.3.7. LilyPad

Das LilyPad ist eine kreisförmige Leiterplatte der Arduino Boards. Trotz der geringen Größe – der Durchmesser beträgt nur 5 cm – bietet es 14 digitale und 6 analoge Ports.

Das Anwendungsgebiet sind die sogenannten „Wearables“-Applicationen. Hierbei werden die Microcontroller in Kleidungsstücke integriert, um diverse Sensoren auszulesen bzw. Aktoren zu steuern. Die Programmierung muss über einen USB Adapter erfolgen, da dieser aus Platzgründen eingespart wurde. Die Betriebsspannung ist geringer als bei anderen Boards – es ermöglicht somit das Verwenden von kleineren und platzsparenden Batterien.

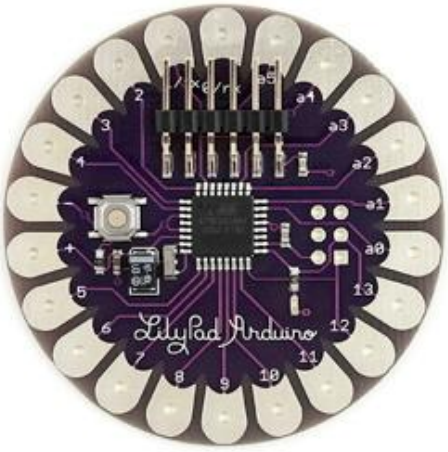


Abbildung 18: Arduino LilyPad [Quelle: http://www.arduino.cc]

1.3.3.8. Fio

Das Arduino Fio (Funnel I/O) zählt seit Anfang 2010 zu den Arduino Boards. Es hat als Basis das LilyPad und ist speziell für Wireless-Anwendungen konzipiert. Es wurde gegenüber dem Arduino LilyPad um diverse Anschlussmöglichkeiten erweitert - z. B. XBee Adapter (Funkmodul) oder externe Lithium Polymer Batterie.

Das Fio ist kompatibel zum Funnel Projekt[25], welches ein Toolkit für Physical-Computing-Anwendungen bereitstellt. Dieses ermöglicht, mit verschiedenen Scriptsprachen, wie z. B. Processing, Ruby oder ActionScript 3 – Daten von elektronischen Bauelementen zu verarbeiten.

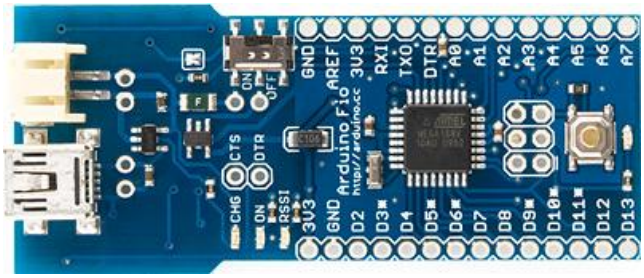


Abbildung 19: Arduino Fio [Quelle: http://www.arduino.cc]

1.3.3.9. Pro / Pro Mini

Die Boards Pro und Pro Mini sollen das Interesse von Professionals auf sich ziehen. Das Board ist in der Anschaffung günstig, aber, da es für fortgeschrittene Anwender gedacht ist, bedarf es zusätzlicher Komponenten und muss, den Bedürfnissen entsprechend angepasst bzw. zusammengesetzt werden.

Die Unterschiede zwischen dem Pro und dem Pro Mini sind die physische Größe sowie die mögliche Ausführung – den Arduino Pro gibt es mit dem ATmega168 oder ATmega328 Microcontroller, den Mini nur mit dem ersteren.

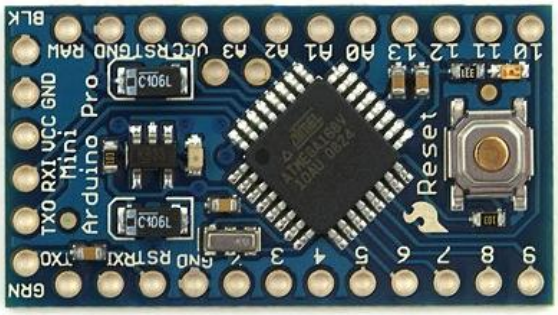


Abbildung 20: Arduino Pro Mini [Quelle: http://www.arduino.cc]

[25] Funnel [online], Verfügbar unter http://funnel.cc/ [Zugang am 28. April 2011]

1.3.4. Background-Wissen

Grundlagen des Background-Wissens: [BAN08], [BRU10]

1.3.4.1. Geschichte

Die meisten Erfindungen bzw. Lösungen entstehen aus einer Not – man trifft auf ein Problem und findet heraus, dass kein Weg existiert, es zu beseitigen. Vor dieser Herausforderung stand auch das Design Institut in Ivrea (Italien) – komplette Institutsbezeichnung „Interaction Design Institute Ivrea (IDII). Es existierten nämlich keine (leistbaren) Microcontroller für die Design-Studenten und so schlossen sich kurzerhand Dozenten mit Elektronikingenieuren zusammen und entwickelten einfache und kosteneffiziente Microcontroller – der Arduino war geboren.

Die Namensgebung, welche die Dozenten des Institutes ausgewählt hatten, führt geschichtlicher Natur ins Jahr 1001 bis 1015 zurück. In dieser Zeit war Arduin von Ivrea (Ivrea ist eine Stadt im Piemont) der Marktgraf der Stadt – in den Jahren 1002 bis 1004 war Arduin auch König von Italien.

Die Arduino-Serie ist komplett offen (keine Lizenzverträge) konzipiert und damit hat man den Grundstein für den großen Erfolg gelegt – einzig der Name „Arduino“ ist als Marke geschützt und gehört der Firma tinker.it.

Durch die Quelloffenheit (Open-Source) der Schaltpläne für den Arduino sowie die frei verfügbare Entwicklungsumgebung hat sich die Community rund um den Microcontroller rasch vergrößert. Jeder Bastler stellt seine Lösungen zur Verfügung und trägt so zu der ständigen Weiterentwicklung bei.

Es gibt mittlerweile viele bekannte Arduino-Projekte – folgende sind beispielhaft angeführt:

Baker Tweet, http://www.bakertweet.com/ [Zugang am 29. April]
: die Welt wird über Twitter informiert, welche leckern Brötchen gerade aus dem Ofen geholt erden

Botanicalls, http://www.botanicalls.com/ [Zugang am 29.April]
: In diesem Projekt wird über Feuchtigkeitssensoren einer Pflanze gemeldet, das Wassermangel herrscht. Diese Nachricht wird ebenfalls über Twitter verbreitet.

1.3.4.2. Prototyping

Ein Prototyp hat in der Literatur – abhängig vom Anwendungsgebiet bzw. Tätigkeitsbereich – mehrere Bedeutungen, wie im Folgenden angeführt[26]:

1) jemand, der als Inbegriff oder als typisches Beispiel für etwas gilt
2) ein Vorbild oder Muster, eine Grundidee für etwas
3) techn., wirtschaftl.: ein Einzelstück, das zur Erprobung vor einer Serienfertigung gebaut wird

Im Anwendungsgebiet Arduino (der Microcontroller-Entwicklung bzw. Programmierung), trifft die Bedeutung des Vorbilds oder Musters bzw. die Schaffung einer Grundidee, am besten zu. Man versucht, diverse Ideen in eine Schaltlogik einzubauen, um am Ende ein Muster für folgende Projekte zu haben.

[26] The Free Dictionary by Farlex [online], Verfügbar unter http://www.thefreedictionary.com/ [Zugang am 29. April]

In der Arduino-Welt befindet man sich ebenfalls in der Softwareentwicklung und hier wird der Begriff „Prototyping“ folgendermaßen definiert:

> *„Prototyping ist ein Verfahren in der Software-Entwicklung, bei dem Prototypen entworfen, konstruiert, bewertet und revidiert werden. Prototyping schafft eine Kommunikationbasis für alle beteiligten Gruppen, vermittelt experimentelle und praktische Erfahrungen für die Auswahl zwischen Designalternativen und ist eine dynamische Beschreibung des sich entwickelnden Softwaresystems.“* [REC06]

Bei der Programmierung eines Microcontrollers, wie z. B. Arduino Uno einer ist, wird dieses entwerfen, konstruieren, bewerten und abändern zur Grundlage für Arduino-Projekte. Der Arduino-Weg soll ein einfacher sein und soll technisch-interessierte auf einem schnellen Weg zum gewünschten Ergebnis verhelfen – auch ohne detailliertem Elektronikwissen.

1.3.4.3. Tinkering

Frei übersetzt bedeutet Tinkering: “basteln”, “flicken”
Der grundlegende Ansatz ist, aus einzelnen elektronischen Bauelementen etwas neues zu schaffen, ein neues Produkt zu kreieren. Der Arduino soll – in seiner Beschaffenheit – diese ermöglichen und schnell zu Ergebnissen führen.

Den Kreislauf des Entwickelns von Arduino-Projekten könnte man schematisch folgendermaßen darstellen:

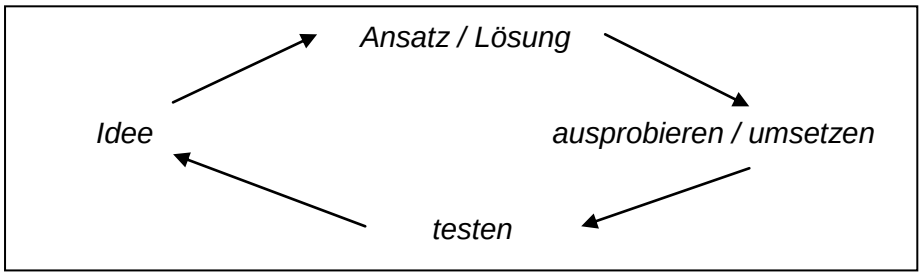


Abbildung 21: schematischer Entwicklungsprozess in Arduino

Ein sehr treffendes Zitat zur Arduino Philosophie ist:

> *„Classic engineering relies on a strict process for getting from A to B:*
> *the Arduino Way delights in the possibility to getting lost on the way and finding C instead.“ [BAN08]*

1.3.5. Lernziele

Im Folgenden werden typische Lernziel-Fragen angeführt.

Fragen / Aufgaben

1) Arduino – Warum wurde er entwickelt?
    a. Woher bekam er seinen Namen?

2) Arduino – Was kann man damit machen?
    a. Bekannte Projekte anführen

3) Verschieden Varianten der Arduino-Serie aufzählen und die Besonderheiten nennen können.

4) Vorteile eines nicht rein technischen Lösungsweges erläutern.

5) Begriffe „Prototyping“ und „Tinkering“ erklären können.

6) Aus welchen Teilen besteht die Arduino Plattform?
    a. Die wichtigsten Eigenschaften der Teile nennen können.

1.4. Lehreinheit 3

Wissen, ...

- was ein Breadboard ist
- wie ein Schalter verwendet wird
- wieviele Ein- bzw. Ausgänge vorhanden sind
- wie man Debug-Anweisungen „sichtbar“ macht

Verstehen / Erklären

- verstehen, was Bouncing ist und warum es auftritt
- was Debugging bedeutet
- was ein digitaler bzw. analoger Ein- / Ausgang ist

1.4.1. Breadboard - Steckbrett

Ein Steckbrett [BRU10] – die engl. Bezeichnung lautet Breadboard - ist eine Leiterplatte mit vordefinierten Verbindungen (in Abbildung 22 gelb markiert), in das man diverse elektronische Bauelemente wie Leuchtdioden, Widerstände, Schalter usw, stecken kann, um Schaltkreise ohne aufwendiges Löten darzustellen bzw. abzubilden. Das Breadboard ist insbesondere in der Testphase und zur Überprüfung von elektrischen Schaltkreisen sinnvoll und sehr hilfreich.

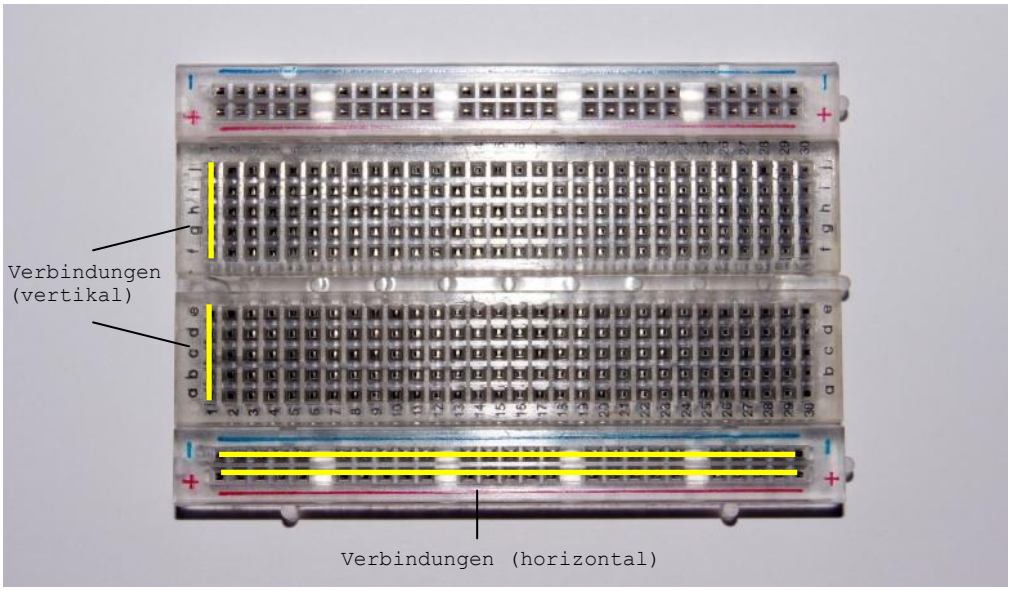


Abbildung 22: mittelgroßes Steckbrett (Breadboard)

In der obigen Grafik ist ein mittelgroßes Steckbrett abgebildet, welches vertikale und horizontale Verbindungen besitzt. Die horizontalen werden meist zur Stromverteilung und Schaltkreisverbindung genutzt.

1.4.2. Blinking LED ...

Das erste Beispiel – das blinkende LED am PIN 13 – kann ebenfalls mit Hilfe des Steckbrettes abgebildet werden.
Für die folgenden Beispiele benötigt man einen Microcontroller (z. B. Arduino Uno) sowie ein Breadboard – die Größe ist dabei nicht relevant.

1.4.2.1. ... am Steckbrett

Für die Abbildung des Schaltkreises am Steckbrett benötigt man

- ein LED
- einen Widerstand
- zwei Verbindungskabeln

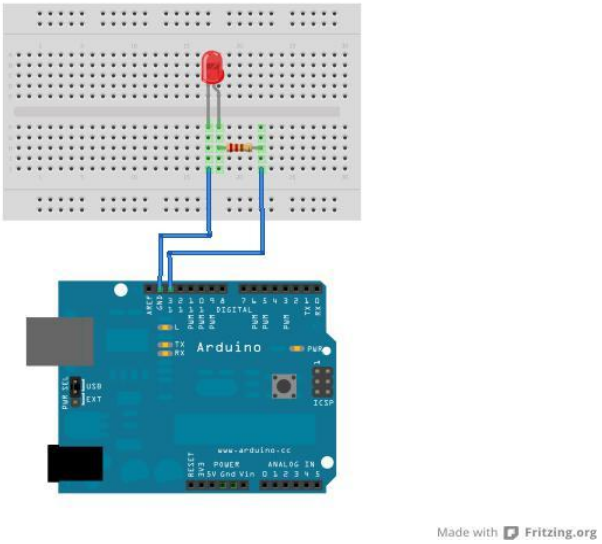


Abbildung 23: Blinking LED am Steckbrett

Programmcode

```
/*  Beispiel 01 - LED13
    Programm bringt ein LED am digitalen Pin 13 zum Blicken.
    Wartezeit zwischen Ein- und Auschalten beträgt 0.5 Sekunden.
*/

// globale Variablen (Konstanten)
int LED = 13;    // gibt den digitalen Pin an
int DELAY = 500; // definiert die Dauer zwischen Ein- und Ausschalten

// Initialisierungsfunktion
void setup() {
  pinMode(LED,OUTPUT);  // digitaler Pin wird als Output definiert
}

// Hauptfunktion
void loop() {
  // digitaler Pin wird auf HIGH gesetzt; LED leuchtet
  digitalWrite(LED,HIGH);
  // es wird eine halbe Sekunde gewartet
  delay(DELAY);
  // digitaler Pin wird auf LOW gesetzt; LED leuchtet nicht
  digitalWrite(LED,LOW);
  // es wird eine halbe Sekunde gewartet
  delay(DELAY);
  //
}
```

1.4.2.2. … am Steckbrett mit Button – Version 1

Für die Abbildung des Schaltkreises am Steckbrett benötigt man

- ein LED
- einen Button
- zwei Widerstände
- sechs Verbindungskabeln

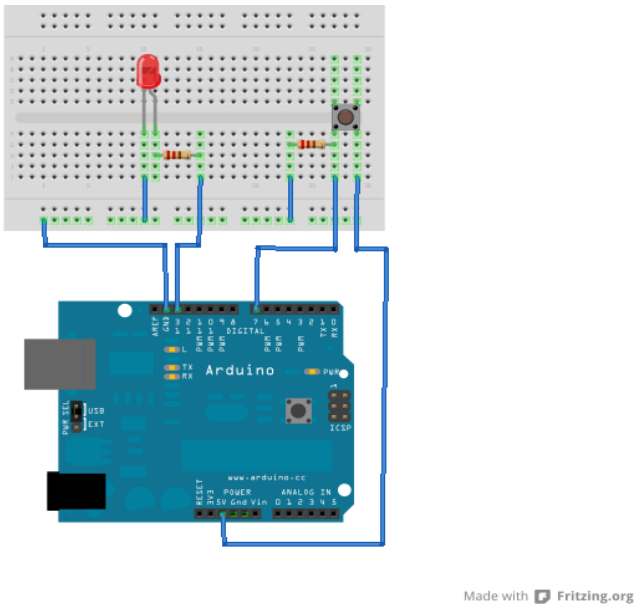


Abbildung 24: Blinking LED am Steckbrett mit Button

Programmcode

```
/*  Beispiel 02 - PushButton v1
    Programm bringt ein LED durch Drücken eines Pushbuttons zum Leuchten.
*/

// globale Variablen (Konstanten)
int LED = 13;    // gibt den digitalen Pin des LEDs an
int BUTTON = 7;  // gibt den digitalen Pin des Buttons an

// globale Variablen
int val = 0; // speichert den Status des Buttons

// Initialisierungsfunktion
void setup() {
  pinMode(LED,OUTPUT);   // digitaler Pin wird als Output definiert
  pinMode(BUTTON,INPUT); // digitaler Pin wird als Input definiert
}

// Hauptfunktion
void loop() {
  // Status des Buttons auslesen
  val = digitalRead(BUTTON);
  //
  // prüfen, ob Button gedrückt ist
  if( val == HIGH ) {
    // digitaler Pin wird auf HIGH gesetzt; LED leuchtet
    digitalWrite(LED,HIGH);
  }
  else {
    // digitaler Pin wird auf LOW gesetzt; LED leuchtet nicht
    digitalWrite(LED,LOW);
  }
}
```

1.4.2.3. ... am Steckbrett mit Button – Version 2

Die Abbildung des Schaltkreises ist ident mit jener in Punkt 1.4.2.2 ... am Steckbrett mit Button – Version 1.
Die Version 2 des Programmes soll ermöglichen, dass nach dem Drücken des Buttons das LED eingeschaltet bzw. ausgeschaltet bleibt. Hierzu kann folgender Programmcode verwendet werden.

Programmcode

```
/*  Beispiel 02 - PushButton v2
    Das Programm schaltet ein LED durch Drücken eines Pushbuttons ein
    bzw. aus.
*/

// globale Variablen (Konstanten)
int LED = 13;    // gibt den digitalen Pin des LEDs an
int BUTTON = 7;  // gibt den digitalen Pin des Buttons an

// globale Variablen
int val = 0;   // speichert den Status des Buttons
int state = 0; // 0 bedeutet LED ausgeschaltet, 1 bedeutet LED eingeschaltet

// Initialisierungsfunktion
void setup() {
  pinMode(LED,OUTPUT);   // digitaler Pin wird als Output definiert
  pinMode(BUTTON,INPUT); // digitaler Pin wird als Input definiert
}

// Hauptfunktion
void loop() {
  // Status des Buttons auslesen
  val = digitalRead(BUTTON);
  //
  // ist Button gedrückt, Status ändern
  if( val == HIGH ) {
    state = 1 - state;
  }
  //
  // prüfen, ob LED eingeschaltet werden soll
  if( state == 1 ) {
    // digitaler Pin wird auf HIGH gesetzt; LED leuchtet
    digitalWrite(LED,HIGH);
  }
  else {
    // digitaler Pin wird auf LOW gesetzt; LED leuchtet nicht
    digitalWrite(LED,LOW);
  }
}
```

Eine Programmcodezeile sollte näher erläutert werden und zwar:

state = 1 – state;

Die Anweisung ermöglicht ein Wechseln der Variablenwerte zwischen 0 und 1. Warum? Die Variable ‚state' nimmt nur die Werte 0 oder 1 an. Durch den kleinen mathematischen Trick, wird somit der Wert von ‚state' auf 1 gesetzt, wenn ‚state' 0 war und auf 0 gesetzt, wenn ‚state' 1 war.
z. B. ‚state' enthält zu Beginn 0

erster Durchlauf: state = 1 – 0; // ‚state' wird auf 1 gesetzt
zweiter Durchlauf: state = 1 – 1; // ‚state' wird auf 0 gesetzt
dritter Durchlauf: state = 1 – 0; // ‚state' wird auf 1 gesetzt
usw.

Leider erkennt man beim Ausprobieren des Beispiels sehr rasch, dass durch das Drücken des Buttons nicht immer das gewünschte Ergebnis, nämlich das LED aus- bzw. einzuschalten, erzielt wird.

Warum tritt dieser Effekt auf?
Ein Microcontroller (z. B. der Arduino Uno) verarbeitet interne Befehle sehr rasch – bis zu 16 Millionen Befehle in der Sekunde. Es wird also deutlich, dass bei jedem Drücken des Buttons der Status einige tausendmal gelesen, gesetzt, gelesen, gesetzt usw. wird. In diesem Fall ist das Ergebnis, ob das LED ein- bzw. ausgeschaltet werden soll, nicht vorhersehbar.

Lösung
Der Status ‚state' darf nur zum Zeitpunkt geändert werden, wenn der Button gedrückt wird. Dazu wird der ‚alte' Wert des Buttons gespeichert, um ihn mit dem aktuellen Wert zu vergleichen (siehe Bouncing-Beispiel).

1.4.3. Bouncing

1.4.3.1. Was ist das?

Das Bouncing bzw. Prellen bezeichnet bei Schaltern (Buttons) eine Signalfolge von mehreren HIGHs bzw. LOWs beim Auslesen digitaler Eingänge, d. h. bei jedem Tastendruck wird nicht nur ein HIGH oder LOW gelesen, sondern eine undefinierte Anzahl dieser.

1.4.3.2. Wie kann ich es beheben?

Um einen Schalter (Button) zu entprellen, gibt es mehrere Möglichkeiten – einerseits kann

eine Hardware- bzw. andererseits eine Softwarelösung angewendet werden.

Für die Hardwarelösung sind zusätzliche elektronische Bauelemente erforderlich – diese Methode wird hier nicht näher erläutert. Detaillierte Informationen darüber sind unter http://www.mikrocontroller.net/articles/Entprellung [Zugang am 24. Juni 2011] verfügbar.

Softwaretechnisch ist ein entprellen bzw. de-bouncing recht einfach umzusetzen. Es wird nach jeder Statusänderung eine Verzögerung von 10-50 ms eingebaut, dadurch wird die Zeit des ‚Prellens' überbrückt und die Stati können korrekt gelesen werden.

Mittlerweile wird für Arduino-Anwendungen eine spezielle Bibliothek zur Verfügung gestellt. Mit dieser lässt sich auf einfache Weise ebenfalls das Entprellen darstellen. Nähere Informationen dazu sind unter http://www.arduino.cc/playground/Code/Bounce [Zugang am 24. Juni 2011] verfügbar.

Im folgenden Beispiel wird auf diese Bibliothek verzichtet.

Programmcode [BAN08]

```
/*  Beispiel 03 - PushButton v3
    Das Programm schaltet ein LED durch Drücken eines Pushbuttons ein
    bzw. aus.
    Nur jedes Drücken des Buttons löst eine Statusänderung aus bzw.
    ist das Entprellen abgebildet.
*/

// globale Variablen (Konstanten)
int LED = 13;    // gibt den digitalen Pin des LEDs an
int BUTTON = 7;  // gibt den digitalen Pin des Buttons an

// globale Variablen
int val = 0;     // speichert den Status des Buttons
int oldval = 0;  // speichert den 'alten' Status des Buttons
int state = 0;   // 0 bedeutet LED ausgeschaltet, 1 bedeutet LED eingeschaltet

// Initialisierungsfunktion
void setup() {
  pinMode(LED,OUTPUT);   // digitaler Pin wird als Output definiert
  pinMode(BUTTON,INPUT); // digitaler Pin wird als Input definiert
}

// Hauptfunktion
void loop() {
  // Status des Buttons auslesen
  val = digitalRead(BUTTON);
  //
  // ist Button gedrückt und 'alter' Wert ungleich 'aktuellem' Wert
  if( (val == HIGH) && (oldval != val) ) {
    state = 1 - state;
    delay(10); // Verzögerung für das Entprellen
  }
  // den gelesenen Wert des Buttons speichern
  oldval = val;
  //
  // prüfen, ob LED eingeschaltet werden soll
  if( state == 1 ) {
    // digitaler Pin wird auf HIGH gesetzt; LED leuchtet
    digitalWrite(LED,HIGH);
  }
  else {
    // digitaler Pin wird auf LOW gesetzt; LED leuchtet nicht
    digitalWrite(LED,LOW);
  }
}
```

`if( (val == HIGH) && (oldval != val) )`
Um bei der raschen Abarbeitung der einzelnen Befehle nur eine Änderung des Status zu erlauben, wird mit dem ‚alten' bzw. ‚vorherigen' Wert des Buttons verglichen. Nur wenn dieser Ungleich (!=) dem aktuell gelesenen Wert ist, wird der Status geändert.

`delay(10); // Verzögerung für das Entprellen`
Eine Verzögerung von 10 ms genügt, um das Prellen des Schalters zu überbrücken.

`oldval = val;`
Der aktuelle bzw. gelesene Wert (‚val') kann einfach in der Variablen ‚oldval' gespeichert

bzw. gesichert werden, um einen Vergleich mit dem nächsten gelesenen zu ermöglichen.

1.4.4. Eingänge und Ausgänge

Die Eingänge sowie Ausgänge eines Microcontrollers [BAN08], [BRU10] ermöglichen die Kommunikation mit der Außenwelt. Diese Schnittstellen ermöglichen ein Austauschen von Informationen mit der Umwelt – durch diese können zum Beispiel Sensoren ausgelesen und Aktuatoren angesteuert werden.

1.4.4.1. Serielle Schnittstelle

Über die serielle Kommunikation ist es möglich, mit externen Modulen Daten auszutauschen (z. B. angeschlossener PC). Hierbei wird beim Arduino Uno die USB-Kabel-Verbindung dazu verwendet, diese Kommunikation abzubilden.

Serielle Datenübertragung bedeutet, dass die Datenbits hintereinander gesendet werden. Um die korrekte Funktionsweise der seriellen Schnittstelle zu ermöglichen, muss die Übertragungsgeschwindigkeit und das verwendete Protokoll zwischen den austauschenden Modulen (z. B. PC und Arduino Uno) vereinbart sein.

Nachfolgend wird ein kurzes Beispiel angeführt, welches einen Text an den seriellen Monitor schreibt.

Programmcode

```
/*  Beispiel 04 - SerialCom v0
    Dieses Programm schreibt zu Testzwecken an die serielle Schnittstelle.
    Die Ausgabe ist am Serial Monitor sichtbar.
*/

// globale Variable
long durchlaeufe = 1;

// Initialisierungsfunktion
void setup() {
  Serial.begin(9600);  // öffnet den seriellen Port und setzt
                       // die Übertragungsrate auf 9600 bps (Bits/s)
}

// Hauptfunktion
void loop() {
  // Text 'Schleife: ' im Serial Monitor ausgeben
  Serial.print("Schleife: ");
  // Anzahl der Durchläufe ausgeben; danach in die nächste Zeile wechseln
  Serial.println(durchlaeufe);
  // 20 Sterne ausgeben; nach jeder Ausgabe 1 Sekunde warten
  for( int i = 0; i < 20; ++i ) {
    Serial.print("*");
    delay(1000);
  }
  // Durchläufe erhöhen
  durchlaeufe = durchlaeufe + 1;
  Serial.println("");
  //
}
```

Der serielle Monitor (Serial Monitor) kann über die Buttonleiste bzw. über den Menüpunkt *Tools Serial Monitor* geöffnet werden – mit der Tastenkombination *Strg+Umschalt+M* öffnet sich ebenfalls das serielle Ausgabefenster.

Debugging
Das serielle Ausgabefenster ist ebenfalls ein sehr gutes Hilfsmittel, falls ein Programmcode nach Fehlern untersucht werden muss. Das Fehlersuchen mit Hilfe von Variablenausgaben wird in der Programmierung auch als *Debugging* bezeichnet.

Das folgende Beispiel soll eine mögliche Anwendungsmöglichkeit für das Testen bzw. Debuggen eines Programmcodes auflisten.

Programmcode

```
/*  Beispiel 05 - SerialCom v1
    Mit diesem Programm kann ein Text an die serielle Schnittstelle
    gesendet werden. Über den ‚Serial Monitor' kann die Kommunikation
    sichtbar gemacht werden.
*/

// globale Variablen (Konstanten)
const boolean DEBUG = true;
const int laenge = 100;

// globale Variablen
char incomingtxt[laenge];

// Initialisierungsfunktion
void setup() {
  Serial.begin(9600);  // öffnet den seriellen Port und setzt
                       // die Übertragungsrate auf 9600 bps (Bits/s)
  clearArray();        // leeren des Eingabebuffers
  // ist Debugging eingeschaltet
  if( DEBUG ) {
    Serial.println("Debugging aktiviert!");
  }
}

// Hauptfunktion
void loop() {
  int i=0;
  // überprüfen, ob Daten gesendet werden
  if (Serial.available() 0) {
    // lesen der Daten
    while( Serial.available() ) {
      incomingtxt[i] = Serial.read();
      delay(2);
      // ausgeben der Debug-Informationen falls DEBUG == true
      if( DEBUG ) {
        Serial.print(incomingtxt[i]);
        Serial.println(" character read.");
      }
      i = i + 1;
    }
    // Daten am seriellen Monitor ausgeben
    Serial.print("I received: ");
    Serial.println(incomingtxt);
    //
    clearArray();
  }
}

// Funktion, welche einen Zeichenbuffer löscht
void clearArray() {
  for( int i=0;i<laenge;++i ) {
    incomingtxt[i] = NULL;
  }
}
```

1.4.4.2. Digitale Ein- / Ausgänge

Der Arduino Uno hat 14 digitale Ein- bzw. Ausgänge. Diese Ports befinden sich in der oberen Steckerleiste und sind von 0 bis 13 durchnummeriert. Ein Port bei Microcontrollern bezeichnet einen Anschlusspin, welcher als Ein- oder Ausgang verwendet werden kann.

Die digitalen Ports können nur die Werte 1 oder 0 bzw. HIGH oder LOW annehmen – HIGH bedeutet, dass eine Spannung am Anschlusspin von 5V / 3,3V anliegt; bei LOW liegt keine Spannung an, also 0V.

Funktionen
Die wichtigen Funktionen für das setzen, auslesen und schreiben eines Ports sind:

- pinMode(pin,mode)
- digitalRead(pin,value)
- digitalWrite(pin,value)

Als sehr einfaches Beispiel, welches die genannten Funktionen verwendet, dient das Blinking LED Beispiel.

1.4.4.3. Analoge Ein- / Ausgänge

Der Arduino Uno besitzt 6 analoge Ports – diese befinden sich am rechten untern Rand und sind von A0 bis A5 durchnummeriert.

Der größte Unterschied der analogen Signale zu den digitalen Signalen ist jener, dass sie nicht nur 0 und 1 annehmen können. Die analogen Ports können Werte im jeweiligen verwendeten Spannungsbereich des Arduino annehmen – also von 0 V bis 5 bzw. 3,3 V.

Die analogen Eingangssignale werden mittels des integrierten A/D-Wandlers des Microcontrollers digitalisiert. Was heißt das nun?

A/D-Wandler
Der Analog/Digital-Wandler empfängt die analogen Signale (im Bereich der Versorgungsspannung des Arduino; z. B. 0 – 5 V) und wandelt diese in Werte von 0 bis 1023 um. Der A/D-Wandler arbeitet mit einer Auflösung von 10 Bit – mit 10 Bit können 1024 Werte abgebildet werden (2^{10} = 1024).
z. B. 0 V liefert am analogen Port 0
5 V liefert am analogen Port 1023
2.5 V liefert am analogen Port 512

PWM
Mit Hilfe der Pulsweitenmodulation können Analogwerte an Ports ausgegeben werden. Bei der PWM erfolgt der umgekehrte Weg wie bei einem A/D-Wandler – es wird aus einem digitalen Signal ein analoges generiert.

Es wird demnach das PWM-Signal – Ausgabefrequenz ist ca. 500 Hz - immer zwischen 1 und 0 gewechselt und damit kann mit der veränderbaren Pulszeit (bezeichnet die Zeit, bei der das PWM-Signal 1 bzw. HIGH ist) ein Wert von 0 bis 100 % dargestellt werden.

Die Auflösung liegt bei 8 Bit – somit können Werte von 0 bis 255 (2^8) dargestellt werden.
z. B. Wert 0 ... Pulszeit / Highphase 0%
Wert 127 ... Pulszeit / Highphase 50%
Wert 255 ... Pulszeit / Highphase 100%

Die digitalen Ein- bzw. Ausgänge 3, 5, 6, 9, 10 und 11 können als analoge Ausgänge verwendet werden (gekenntzeichnet mit PWM~).

Funktionen
Die wichtigen Funktionen für das setzen, auslesen und schreiben eines Ports sind:

- analogRead(pin)
- analogWrite(pin,value)

Im folgenden Beispiel wird ein Fotowiderstand (LDR) ausgelesen - die gelesenen Werte werden am seriellen Monitor dargestellt.

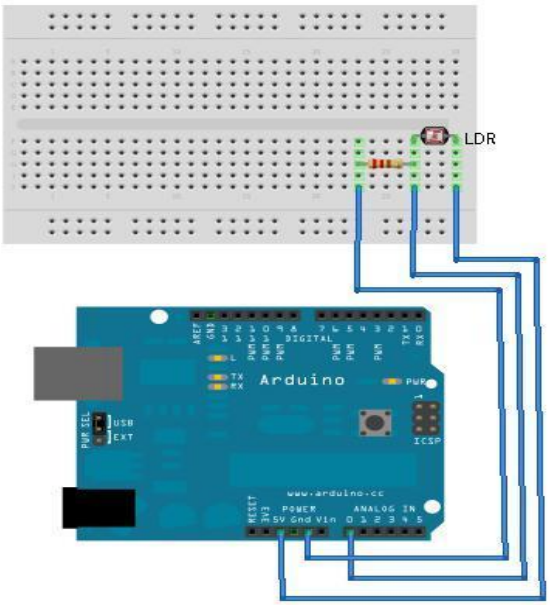


Abbildung 25: Light Dependent Resistor (LDR)

Programmcode

```
/*  Beispiel 06 - LDR v1
    Mit diesem Programm kann der Wert eines Fotowiderstandes gelesen werden.
    Über den ‚Serial Monitor' wird der Wert sichtbar gemacht.
*/

// globale Variablen (Konstanten)
const int SENSOR = 0;

// globale Variablen
int val = 0;

// Initialisierungsfunktion
void setup() {
  Serial.begin(9600);  // öffnet den seriellen Port und setzt
                       // die Übertragungsrate auf 9600 bps (Bits/s)
}

// Hauptfunktion
void loop() {
  // liest den Sensor
  val = analogRead(SENSOR);

  // Ausgabe am Serial Monitor
  Serial.println(val);

  // 100 ms warten
  delay(100);
}
```

1.4.5. Fritzing

Fritzing, *http://fritzing.org/* [Zugang am 28. Juni 2011], ist ein Open-Source Projekt der Fachhochschule Potsdam (Deutschland).

Es bietet die Möglichkeit, elektronische Schaltungen mit grafischen Hilfsmitteln zu entwerfen und diese als Projekte zu speichern. Weiters erlaubt es den Export in diverse Formate, wie z. B. PDF, oder in Grafiken, wie z. B. JPG - um nur die wichtigsten zu nennen.

Durch die sehr übersichtliche grafische Aufbereitung können elektronische Schaltungen rasch bildlich dargestellt werden und zur Veröffentlichung verteilt oder zur Speicherung genutzt werden (z. B. Abbildung 25 wurde mit Fritzing entworfen).

Das Programm ist unter der Homepage
http://fritzing.org/download/
verfügbar.

Nach der Installation, das Standard-Installationsverzeichnis ist
C:\Programme\ fritzing.2011.02.18.pc,
kann das Programm mit *Fritzing.exe* gestartet werden.

Nach dem Öffnen wird Fritzing in der Ansicht ‚Steckplatine' dargestellt.

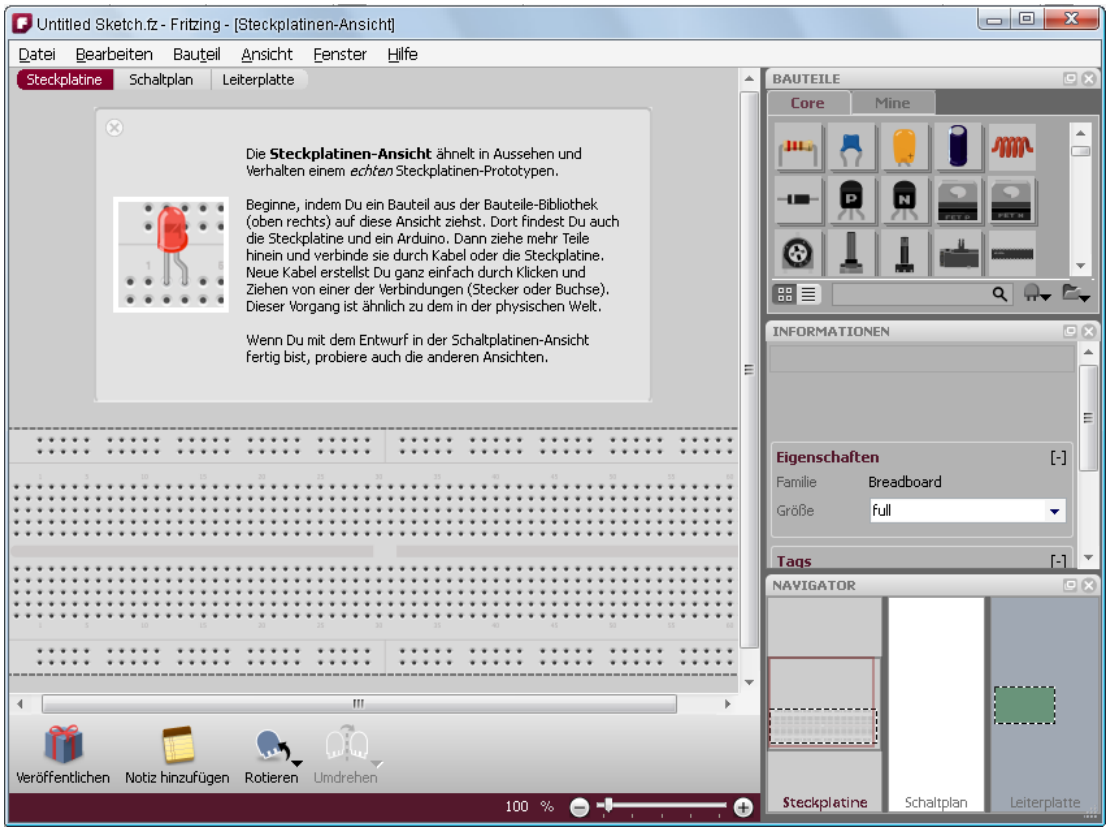


Abbildung 26: Fritzing - Startbildschirm

In dieser Ansicht können durch einfaches Ziehen mit der Maus, Bauteile aus dem rechten oberen Kästchen genommen werden. Es steht ebenfalls ein Arduino zur Verfügung.

Kabelverbindungen sind ebenfalls mit der Technik „Klicken und Ziehen“ zu setzen – z. B. möchte man den analogen Port A0 des Arduinos mit dem Steckbrett verbinden, klickt man mit der Maus auf A0 und zieht bis zum gewünschten Punkt am Breadboard und schon erhält man die gewünschte Verbindung.

Für jedes Bauteil können bestimmte Eigenschaften geändert bzw. gesetzt werden. Ein, zum Beispiel, grünes LED erhält man, in dem man das im Bauteil-Kasten verfügbare rote LED auf das Steckbrett zieht und unter „Informationen“ (rechtes mittleres Kästchen) die gewünschte Farbe auswählt.

Das Programm bietet ebenfalls eine Fülle an Beispielen (Menüpunkt *Datei Beispiele*), anhand Entwürfe schnell erlernt werden können.

Fritzing bietet drei Hauptansichten, welche über den „Navigator“ (rechtes unteres Kästchen) ausgewählt werden können:

1) Steckplatine (z. B. Abbildung 27)
2) Schaltplan (z. B. Abbildung 28)
3) Leiterplatte (z. B. Abbildung 29)

Die folgenden Ansichten betreffen das Beispiel 06 – LDR v1.

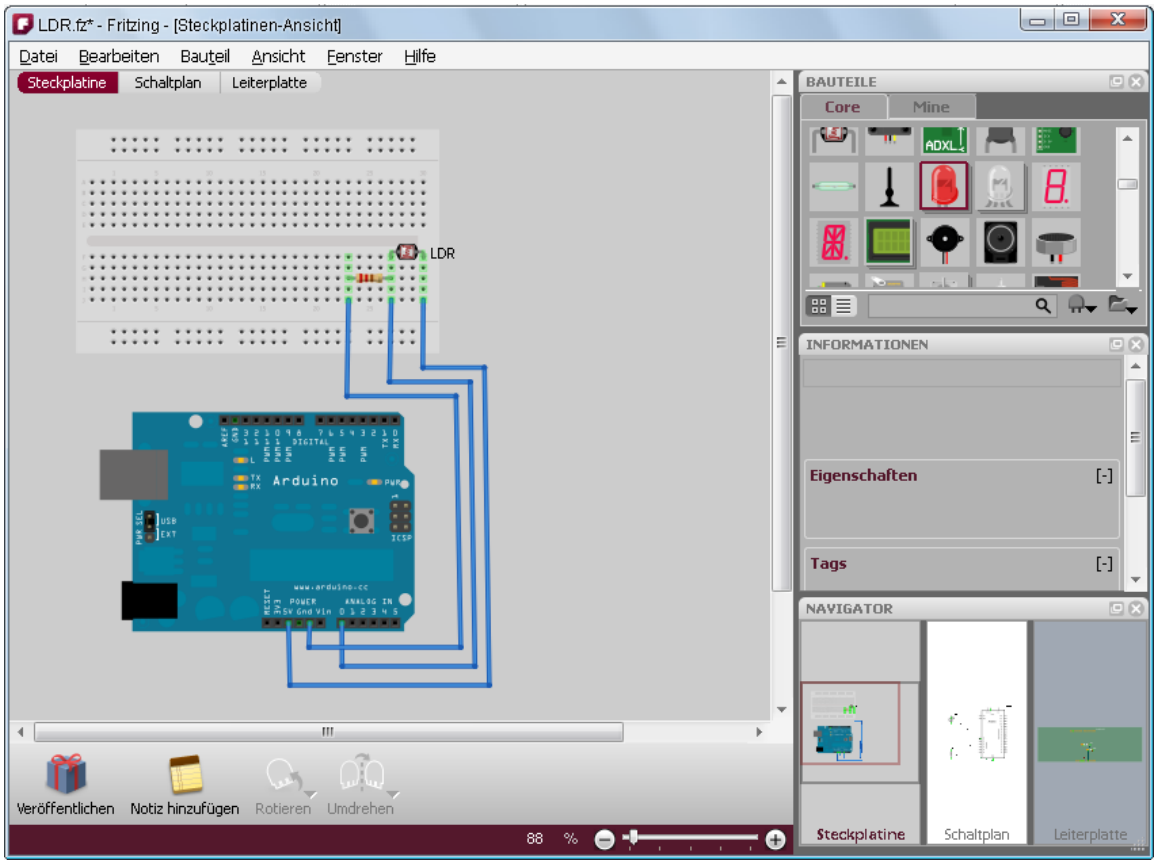


Abbildung 27: Fritzing - Steckplatine

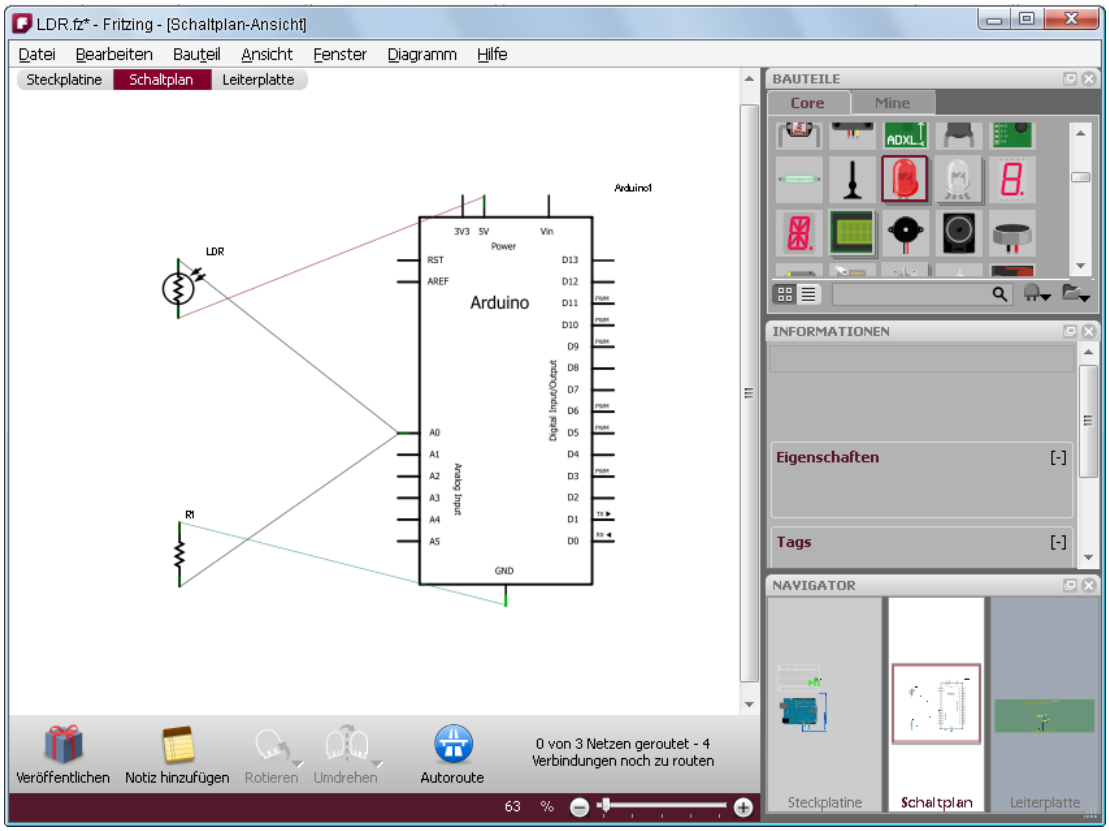


Abbildung 28: Fritzing - Schaltplan

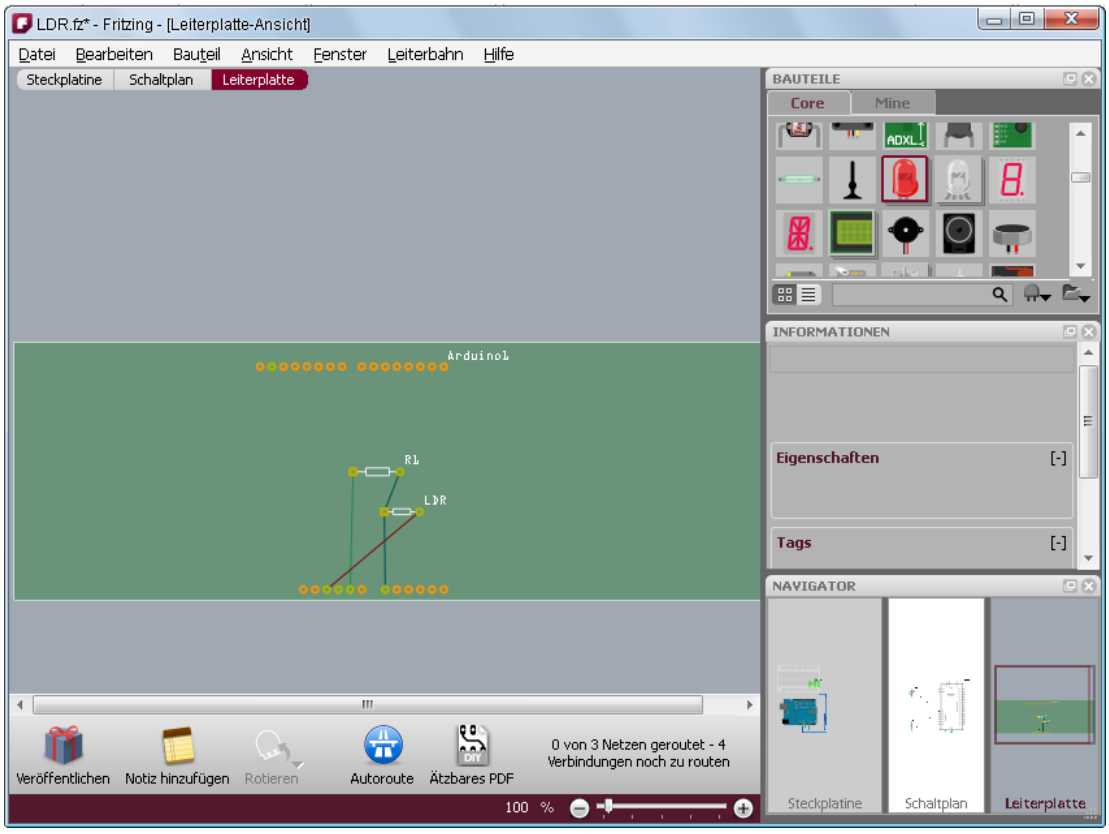


Abbildung 29: Fritzing - Leiterplatte

1.4.6. Lernziele

Im Folgenden werden typische Lernziel-Fragen bzw. Übungsbeispiele angeführt.

Fragen

1) Die Vorteile eines Steckbrettes erklären.

2) Blinking LED - Beispiel am Steckbrett verstehen.
    a. Ohne und mit Button.

3) Verschieden Varianten der Arduino-Serie aufzählen und die Besonderheiten nennen können.

4) Den Begriff „Bouncing“ erläutern.
    a. Was ist es?
    b. Wie kann man es lösen?

5) Eingänge und Ausgänge.
    a. Ein- und Auslesen können.

6) Serielle Schnittstelle – Kommunikation mit dem Arduino abbilden können.

7) Fritzing – Was ist das?
    a. An- und verwenden können.

Beispiele

1) LEDsOnOff

Das Programm soll die Möglichkeit bieten, drei verschiedene LEDs über die serielle Schnittstelle (Serielle Schnittstelle) ein- bzw. auszuschalten. Das Keyboard am PC bzw. Laptop soll dabei die Steuerung übernehmen.
Folgende Logik soll implementiert werden:

- Drücken von ‚5‘
    - grünes LED soll ein- bzw. ausgeschaltet werden
- Drücken von ‚7‘
    - gelbes LED soll ein- bzw. ausgeschaltet werden
- Drücken von ‚9‘
    - rotes LED soll ein- bzw. ausgeschaltet werden
- Drücken von ‚0‘
    - Statusanzeige soll am seriellen Monitor ausgegeben werden, welches LED leuchtet und welches nicht
- jede abweichende Eingabe wird mit Angabe der Länge am seriellen Monitor ausgegeben

2) Trafficlights

Das Programm soll eine Verkehrsampel simulieren.
Die Lösung soll ein Steckbrett, die notwendigen LEDs (Rot, Gelb, Grün) sowie nötige Widerstände verwenden.Die Ampelschaltung soll zusätzlich mit einem Button ein- bzw. ausschaltet werden können. In der Ausgangssituation soll die Ampel nicht eingeschaltet sein – keine LEDs leuchten. Die Simulation beginnt erst zu laufen, wenn der Button gedrückt wird. Der Ablauf der Ampelphasen soll - wie folgt - abgebildet sein:

- ROT leuchtet für drei Sekunden
- ROT und GELB leuchten für zwei Sekunden
- ROT und GELB werden ausgeschaltet
- GRÜN leuchtet für drei Sekunden
- GRÜN blinkt drei Mal
- ROT leuchtet für drei Sekunden
- …

Hinweis

Damit der Status des Buttons jederzeit ausgelesen werden kann, muß eine eigene ‚delay'-Funktion implementiert werden.

1.5. Lehreinheit 4

Wissen, ...
- welche Elektronikbauteile häufig verwendet werden
- wie das Ohmsche Gesetz lautet
- was elektrischer Strom ist

Verstehen / Erklären
- was eine (Leucht)-Diode ist
- warum ein Widerstand bei einer LED verwendet werden muss
- was das Ohmsche Gesetz besagt
- was die technische Stromrichtung ist

Grundlagen / Literatur zur Lehreinheit 4: [DEI00], [BRU10]

1.5.1. Was ist elektrischer Strom?

Der elektrische Strom [DEI00] ist in der heutigen Zeit allgegenwärtig und zur Selbstverständlichkeit geworden - nahezu in jedem Lebensbereich wird Strom verbraucht.

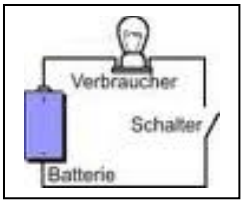
In Österreich wird elektrischer Strom hauptsächlich in Wasserkraftwerken gewonnen. Der Anteil beträgt derzeit etwa 60 %. Die restlichen 40 % werden fast zur Gänze von Wärmekraftwerken lukriert – ein derzeit noch geringer Teil wird durch Windkraftwerke gewonnen.

Was ist nun elektronischer Strom? In der Literatur bzw. Schulbüchern wird man auf folgende Definition treffen:

Strom ist bewegte elektrische Ladung.

Um diese Aussage einigermaßen zu verstehen, muss man wohl oder über etwas Grundlagenwissen anhäufen.

Ein elektrischer Stromkreis ist recht einfach zu erklären.



Er besteht immer aus einer Quelle (z. B. Batterie), einem Verbraucher (z. B. Glühbirne) und den dazwischenliegenden Leitungen.

Wird über den Schalter der Stromkreis geschlossen, fließt der Strom – dabei beginnt der Strom an jeder Stelle des Stromkreises gleichzeitig zu fließen.

Welche Richtung fließt der Strom?
Die Aussage „Gegensätze ziehen sich an“ kommt nicht von ungefähr, denn eine Quelle – in unserem Beispiel die Batterie – hat immer einen Pluspol (oben) und einen Minuspol (unten).

Wird der Schalter nun geschlossen, werden vom Minuspol die negativ geladenen Teilchen (die sogenannten Elektronen) „abgestoßen“ und wandern Richtung Pluspol. Dieser Strom von Minuspol zum Pluspol wird Elektronenstrom genannt.

In der Praxis wird fast hauptsächlich von der technischen Stromrichtung gesprochen – diese beschreibt die Bewegungsrichtung der positiven Ladungsträger (Protonen). Somit ist die ***technische Stromrichtung*** vom Pluspol zum Minuspol.

Wenn wir das Ganze mit etwas Abstand zur Physik und den damit verbunden Begriffen betrachten, nehmen wir das einfache Beispiel des Blinking LED am Arduino.

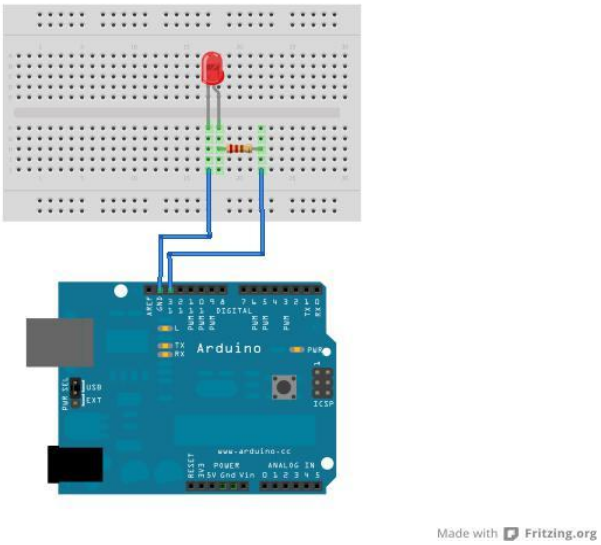


Abbildung 30: Blinking LED

Wird der Arduino mit Strom versorgt (z. B. über den USB Anschluss des PCs) stellt sich folgende Stromrichtung ein. Vom Pin GND bewegen sich die Elektronen (negativ geladene Teilchen) in Richtung LED, Widerstand, Pin 13 und dem PC.

Die technische Stromrichtung ist somit in umgekehrter Richtung – vom PC über Pin 13, Widerstand, LED zum Pin GND.

1.5.1.1. Einfaches Beispiel zum Ohmschen Gesetz

Jede Schülerin bzw. jeder Schüler hat das Ohmsche Gesetz gelernt, weiß wie es lautet und kann es erklären. Leider sieht die Realität etwas anders aus...

In vielen Physik-Unterrichtsstunden wird eine Formel erklärt, welche drei Buchstaben enthält. Leider fehlt in den meisten Fällen eine einfache bildliche Darstellung, was das Verständnis deutlich einfacher machen würde. Ein Bild oder eine banale Eselsbrücke hilft meistens, um schwierige naturwissenschaftliche Erkenntnisse zu verstehen.

Nehmen wir das Wort URI – diese drei Buchstaben bilden (fast) bereits die Formel für das Ohmsche Gesetz. Die Formel lautet nämlich U = R * I. Die Formel kann nun durch einfache mathematische Umformungsregeln in die für die Praxis wichtigen Gleichungen gebracht werden.

Ohmsches Gesetz:	I = U / R	U = R * I	R = U / I

Was sagt einem nun diese Formel - für das Verständnis kann als Beispiel ein Schlauch (z. B. ein Gartenschlau) herangezogen werden.

Wenn man bei einem Gartenschlauch den Wasserhahn nur ein wenig aufdreht, fließt auch nur wenig Wasser aus dem Schlauch =der Druck ist niedrig.
Umgekehrt formuliert heißt das, wird der Wasserhahn stark aufgedreht, fließt auch viel Wasser
=der Druck ist groß.

Es gibt also ein Verhältnis zwischen der Wassermenge und dem Wasserdruck – ersetzt man nun die Wassermenge durch die Stromstärke und den Wasserdruck durch die Stromspannung, erhält man eine wichtige Aussage des Ohmschen Gesetzes.
Die Stromstärke (I) ist proportional zur Stromspannung (U): I ~ U

Wenn der Wasserhahn nun halb aufgedreht wird, wird mittelmäßig viel Wasser aus dem Schlauch fließen. Beginnt man durch drehen am Schlauchkopf die Wassermenge zu reduzieren, steigt der Widerstand gegen die Wassermenge und es wird weniger Wasser fließen – also gibt es ein umgekehrtes Verhältnis zwischen Wassermenge und dem Widerstand. Durch das Ersetzen der Wassermenge mit der Stromstärke, erhält man eine weitere wichtige Aussage des Ohmschen Gesetzes.
Die Stromstärke (I) ist umgekehrt proportional zum Widerstand (R): I ~ 1 / R

Die beiden Aussagen zusammengeführt, ergeben formelmäßig das Ohmsche Gesetz.

I = U / R

Grundlagen aus dem Physik-Buch
Das Ohmsche Gesetz wurde von Georg Simons Ohm 1826 entdeckt. Er erkannte, dass die Stromstärke in einem Stromkreis von der treibenden elektrischen Kraft, aber auch von den entgegenstellenden Widerständen abhängig ist.

In der wichtigen Formel dargestellt:

I = U / R ... Stromstärke = Stromspannung / Widerstand

1.5.2. Elektronikbauteile

Im Folgenden werden häufig verwendete elektronische Bauteile [BRU10] dargestellt und kurz beschrieben.

Die Auswahl der Bauteile basiert auf dem Elektronik Set, welches als „Einsteiger-Kit V mit Uno“ erhältlich ist. Dieses Kit kann über die Homepage http://physicalcomputing.at erworben werden und kostet € 60,--.

1.5.2.1. Halbleiterdiode

Die Halbleiterdiode, allgemein auch bezeichnet als Diode, leitet den Strom praktisch nur in der Durchlassrichtung.

Abbildung 31: diverse Dioden

[Quelle: http://www.elektronik-kompendium.de]

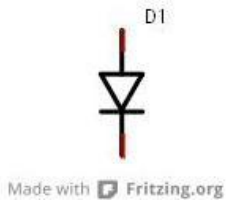


Abbildung 32: Schaltzeichen für eine Diode

In der Abbildung 32 wird das Schaltzeichen einer Diode dargestellt – in diesem Fall wäre die Duchlassrichtung von oben (Pfeil) nach unten (Balken). Das Leiten des Stromes von „unten nach oben“ wird unterbunden, sozusagen gesperrt, deshalb wird diese Gegenrichtung auch als Sperrrichtung bezeichnet.

Die Durchlass-Seite (Pfeil-Seite) wird als Anode und die Sperr-Seite (Balken) als Kathode bezeichnet. Durch das Aufdrucken des Schaltzeichens auf das Bauelement wird die Durchflussrichtung kenntlich gemacht. Falls die Diode für das Bedrucken zu klein sein sollte, wird die Sperr-Seite mit dem sogenannten Kathoden-Ring gekennzeichnet (in Abbildung 31, bei der untersten Diode auf der linken Seite ersichtlich).

Dioden können als Schutzelemente oder als Gleichrichter bei Wechselspannung verwendet werden. In Schaltungen können Dioden Signale verbinden.

1.5.2.2. Leuchtdiode

Die Leuchtdiode – umgangssprachlich als LED (Light Emitting Diode) bezeichnet – sind eine Sonderform der Dioden. Sie arbeiten somit wie Halbleiterdioden, geben aber in Durchfluss-richtung Licht ab – diese Eigenschaft wird in elektronischen Schaltungen meist als Anzeige verwendet.

Die Leuchtdioden sind in verschiedenen Größen bzw. Farben erhältlich.

Abbildung 33: Leuchtdioden (diverse Farben)
[Quelle: http://physicalcomputing.at]

Abbildung 34: Schaltzeichen für ein LED

Um die beiden Anschlüsse der Leuchtdiode zu unterscheiden, ob es die Anode bzw. Kathode ist, gibt es zwei Möglichkeiten:

1) die Anode ist der längere Anschluss bzw.
2) beim Durchschauen des LEDs (wenn möglich) bezeichnet der größere Kontakt die Kathode

Wichtig
LEDs müssen immer mit einem Vorwiderstand betrieben bzw. verwendet werden.
Warum? Durch den Vorwiderstand wird der Strom, der durch das LED fließt, begrenzt.

Eine spezielle Form der LED-Familie ist das RGB LED.

Abbildung 35: RGB LED [Quelle: http://physicalcomputing.at]

Wie in der obigen Abbildung ersichtlich, bietet das RGB LED 4 Anschlüsse. Der längste Anschluss ist die gemeinsame Kathode, jeder weitere dient als Anschluss für die Farben Rot (R), Grün (G) und Blau (B).

1.5.2.3. Widerstand

Der Widerstand ist das am häufigsten verwendete elektrische Bauelement. Es wird zur Spannungsbegrenzung verwenden.

Abbildung 36: Widerstände

[Quelle: http://physicalcomputing.at]

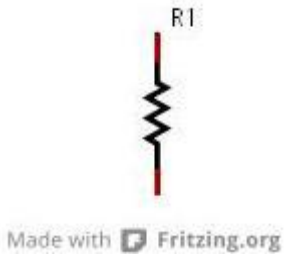


Abbildung 37: Schaltzeichen für einen Widerstand

Der elektrische Widerstand hat das Formelzeichen R (engl. Resistor) und die Maßeinheit Ohm (Ω).

Widerstände gibt es in verschiedensten Ausführungen. Der Widerstandswert kann über den Farbcode bzw. -ringe am Widerstand selbst abgeleitet werden.

Widerstandsbestimmung (4 Ringe) bei Kohleschichtwiderständen

Ringfarbe	1. Ring	2. Ring	3. Ring (Multiplikator)	4. Ring (Toleranz)
schwarz	0	0	-	-
braun	1	1	× 10	1 %
rot	2	2	× 100	2 %
orange	3	3	× 1000	-
gelb	4	4	× 10000	-
grün	5	5	× 100000	0,5 %
blau	6	6	× 1000000	0,25 %
violett	7	7	× 10000000	0,1 %
grau	8	8	-	-
weiß	9	9	-	-
gold	-	-	× 0,1	5 %
silber	-	-	× 0,01	10 %

Abbildung 38: Widerstandsbestimmung [Quelle: http:// www.elektronik-kompendium.de]

Es gibt ebenfalls Widerstände mit 5 Farbringen, welche Metallschichtwiderstände beschreiben. Bei diesen Bauteilen ist der Widerstandswert genauer angegeben.

1.5.2.4. Schalter / Drucktaster

In elektronischen Schaltungen werden Schalter meist dazu verwendet, um ein Signal zu senden. Sie dienen ebenfalls dazu, mehrere Schaltkreise voneinander zu trennen.

Im Microcontroller-Bereich werden Schalter meist auch als Power-On-Schalter benötigt

Abbildung 39: Drucktaster

[Quelle: http://physicalcomputing.at]

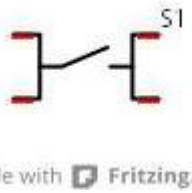


Abbildung 40: Schaltzeichen für einen Drucktaster

1.5.2.5. Temperatursensor

Temperatursensoren gibt es in den verschiedensten Varianten. Der hier angeführte – im Einsteiger Set enthaltene – Sensor gehört zur Kategorie der temperaturabhängigen Widerstände.
Hierbei gibt es zwei Unterscheidungen:

1) NTC (Negative Temperature Coefficient, Heißleiter)
    a. Dieser hat bei hohen Temperaturen seinen niedrigsten Widerstand (z. B. Silizium)
2) PTC (Positive Temperature Coefficient, Kaltleiter)
    a. Dieser hat bei niedriegen Temperaturen seinen geringsten Widerstand (z. B. Glühlampe)

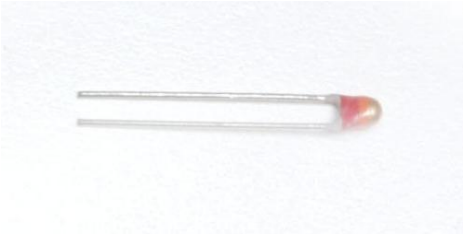


Abbildung 41: Temperatursensor

[Quelle: http://physicalcomputing.at]

Abbildung 42: Schaltzeichen für einen Temperatursensor

1.5.2.6. Potentiometer

Potentiometer sind einstellbare bzw. veränderbare Widerstände. Die häufigsten einstellbaren Widerstände sind Dreh- bzw. Schiebewiderstände.

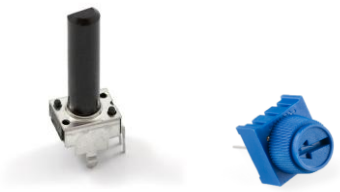


Abbildung 43: Dreh- und Trimmpotentiometer

[Quelle: http://physicalcomputing.at]

Abbildung 44: Schaltzeichen für einen Potentiometer

Eine Sonderform der Drehpotentiometer sind die Trimmpotentiometer. Bei diesen wird der Wert mit Hilfe eines Werkzeuges (meist reicht ein Schraubenzieher) eingestellt.

1.5.2.7. Piezo Schallwandler

Der Piezo Schallwandler dient als einfacher Lautsprecher und Mikrofon.

Abbildung 45: Piezo Schallwandler

[Quelle: http://physicalcomputing.at]

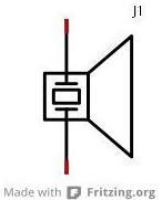


Abbildung 46: Schaltzeichen für einen Piezo Schallwandler

1.5.2.8. Reed Relais

„Relais sind elektromagnetische oder elektromechanische Schalter. Sie werden zum Ein-, Aus- oder Umschalten von Stromkreisen verwendet. Das klassische Relais ist ein elektromagnetischer Schalter.“[27]

Das besondere an Reed Relais ist, dass sie eine wesentlich höhere Lebensdauer haben als normale Relais - 1000 Millionen Schaltungen und darüber sind keine Seltenheit.[28]
Die Schaltung erfolgt über Kontakte (federnd gelagert), welche sich in einem luftleeren oder mit Schutzgas gefüllten Raum befinden.

Abbildung 47: Reed Relais

[Quelle: http://physicalcomputing.at]

Abbildung 48: Schaltzeichen für einen Reed Relais

[27] Elektronik Kompedium [online], Verfügbar unter http://www.elektronik-kompendium.de/ [Zugang am 28. Juni 2011]

[28] IT Wissen – Das große Online-Lexikon für Informationstechnologie [online], Verfügbar unter http://www.itwissen.info/ [Zugang am 21. April 2011]

1.6. Beispiel-Lösungen

1.6.1. Beispiel: LEDsOnOff

Überblick
Im folgenden Beispiel sollen drei LEDs über die Tastatur ein- bzw. ausgeschaltet werden können.

Für die Abbildung des Schaltkreises am Steckbrett wird benötigt:
- drei LEDs (z. B. rot, gelb und grün)
- drei Widerstände
- sieben Verbindungskabeln

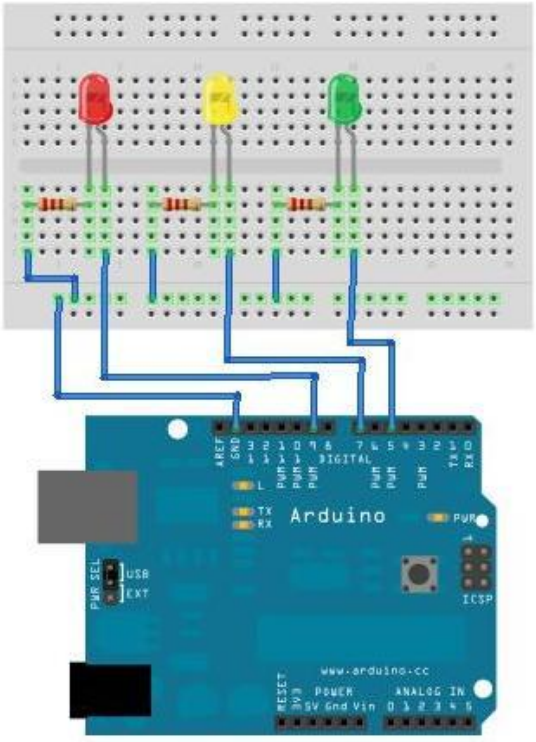


Abbildung 49: LEDsOnOff - Beispiellösung

Beschreibung
In diesem Beispiel wird die Handhabe mit der seriellen Schnittstelle sowie das Arbeiten mit Textfeldern etwas näher betrachtet.
Das Keyboard am PC bzw. Laptop soll die Steuerung für LEDs übernehmen. Je nach Eingabe der PIN-Nummer, an der das jeweilige LED angeschlossen ist, soll dieses ein- bzw. ausgeschaltet werden.
Folgende Logik ist implementiert:
- Drücken von ‚5‘
  - grünes LED wird ein- bzw. ausgeschaltet
- Drücken von ‚7‘
  - gelbes LED wird ein- bzw. ausgeschaltet
- Drücken von ‚9‘
  - rotes LED wird ein- bzw. ausgeschaltet
- Drücken von ‚0‘

    - Statusanzeige wird am seriellen Monitor ausgegeben werden, z. B. LED rot: leuchtet, LED gelb: leuchtet nicht, LED grün: leuchtet
- jede abweichende Eingabe wird mit Angabe der Länge am seriellen Monitor ausgegeben

Um die Eingabe von mehreren Zeichen auszuwerten, werden diese in einem Textfeld gespeichert. Dadurch wird ermöglicht, dass die jeweilige Länge des eingegebenen Textes am seriellen Monitor ausgegeben wird.

Im Folgenden wird eine mögliche Variante dieser Eingabe und der damit verbundenen Steuerung der LEDs dargestellt.

Programmcode

```
/*  Beispiel 08 - LEDsOnOff v1
    Das Programm ermöglicht das Ein- bzw Ausschalten von LEDs am Steckbrett. Durch
    Drücken der jeweiligen Nummer (PIN-Nummer) auf der Tastatur, wird das LED,
    welches an diesem PIN angeschlossen ist, ein- bzw. ausgeschaltet. Falls 0
    gedrückt wird, soll eine Statusanzeige – welches LED leuchtet welches nicht -
    im Serialmonitor ausgegeben werden
*/

#define DEBUG 1

#define LED_R 9 // gibt den digitalen Pin des roten LEDs an
#define LED_Y 7 // gibt den digitalen Pin des gelben LEDs an
#define LED_G 5 // gibt den digitalen Pin des grünen LEDs an

int stat_red = 0; // speichert den Status des roten LEDs
int stat_yel = 0; // speichert den Status des gelben LEDs
int stat_gre = 0; // speichert den Status des grünen LEDs

const int laenge = 100;   // maximale Länge der Eingabe
char incomingtxt[laenge]; // Speicher für eingegeben Text

void setup() {
  pinMode(LED_R,OUTPUT); // digitaler Pin wird als Output definiert
  pinMode(LED_Y,OUTPUT); // digitaler Pin wird als Output definiert
  pinMode(LED_G,OUTPUT); // digitaler Pin wird als Output definiert
  //
  Serial.begin(9600);     // öffnet die serielle Schnittstelle und setzt die Daten-
                          // rate auf 9600 bps (bit/s)
  Serial.println("Ready!"); // „Ready“ am Seriellen Monitor ausgeben
  //
  clearArray(); // Eingabefeld löschen
}

void loop() {
  int i=0;
  int check;
  // prüfen, ob Daten gesendet werden
  if (Serial.available() 0) {
        // ankommende Bytes lesen
        while( Serial.available() ) {
          // prüfen, ob die Länge des Eingabefeldes erreicht ist
          if( i < laenge ) {
            // Byte in den Eingabebuffer übernehmen
            incomingtxt[i++] = Serial.read();
            delay(2);
          } else {
            // Meldung am Seriellen Monitor ausgeben
            Serial.println("Eingabebuffer voll!");
            break;
          }
        }
        // Debugging Information
        if( DEBUG ) {
```

```
          // Eingabe ausgeben
          Serial.print("I received: ");
          Serial.println(incomingtxt);
        }
        // Eingabe auswerten und LEDs schalten
        switchLEDs(incomingtxt);
        // Eingabebuffer leeren
        clearArray();
  }
}

// Funktion, um den Eingabebuffer zu leeren
void clearArray() {
  for( int i=0;i<laenge;++i ) {
    incomingtxt[i] = '\0'; // Zeichenkettenendezeichen ‚\0' (Null Charakter)
  }
}

// Funktion, welche die Länge eines Textfeldes zurück gibt
int lengthArray(char *c) {
  int i = 0;
  while( c[i++] != '\0' ) { };
  return i-1;
  // Alternative zum Retournieren der Länge
  // return strlen(c);
}

// einfache Funktion, welche eine kurze Statusinfo über die LEDs ausgibt
void printStatus() {
  // Status ausgeben
  Serial.println("***");
  Serial.print("LED rot:   "); Serial.println(getStatus(stat_red));
  Serial.print("LED gelb:  "); Serial.println(getStatus(stat_yel));
  Serial.print("LED gruen: "); Serial.println(getStatus(stat_gre));
  Serial.println("***");
}

// liefert „leuchtet", falls der Parameter ‚stat' true bzw. 1 ist, ansonsten
// wird „leuchtet nicht" geliefert
char* getStatus( int stat ) {
  if( stat ) {
    return "leuchtet";
  } else {
    return "leuchtet nicht";
  }
}

// diese Funktion wertet den eingegebenen Text aus
// falls der eingegeben Text länger als 1 ist, wird die Auswertung übersprungen
void switchLEDs(char* txt) {
  //
  if( lengthArray( txt ) != 1 ) {
    // do nothing
    // Debugging Information
    if( DEBUG ) {
      Serial.print("length: ");
      Serial.println(lengthArray( txt ));
    }
  } else {
    // Debugging Information
    if( DEBUG ) {
      Serial.print("txt: ");
      Serial.println(txt);
    }
    // wenn ein Zeichen eingegeben wurde, wird es ausgewertet…
    switch( *txt ) {
      // bei Eingabe von 9, wird das rote LED ein- bzw. ausgeschaltet
      case '9': stat_red = !stat_red;
                digitalWrite(LED_R,stat_red);
                break;
```

```
      // bei Eingabe von 7, wird das gelbe LED ein- bzw. ausgeschaltet
      case '7': stat_yel = !stat_yel;
                digitalWrite(LED_Y,stat_yel);
                break;
      // bei Eingabe von 5, wird das grüne LED ein- bzw. ausgeschaltet
      case '5': stat_gre = !stat_gre;
                digitalWrite(LED_G,stat_gre);
                break;
      // bei Eingabe von 0, wird eine Statusausgabe im seriellen Monitor angezeigt
      case '0': printStatus();
                break;
    }
  }
}
```

1.6.2. Beispiel: TrafficLight

Überblick
Im folgenden Beispiel wird die Funktionsweise und die Ablaufsteuerung einer Verkehrsampel nachgestellt.

Für die Abbildung des Schaltkreises am Steckbrett wird benötigt
- drei LEDs (rot, gelb und grün)
- einen Button
- vier Widerstände
- zehn Verbindungskabeln

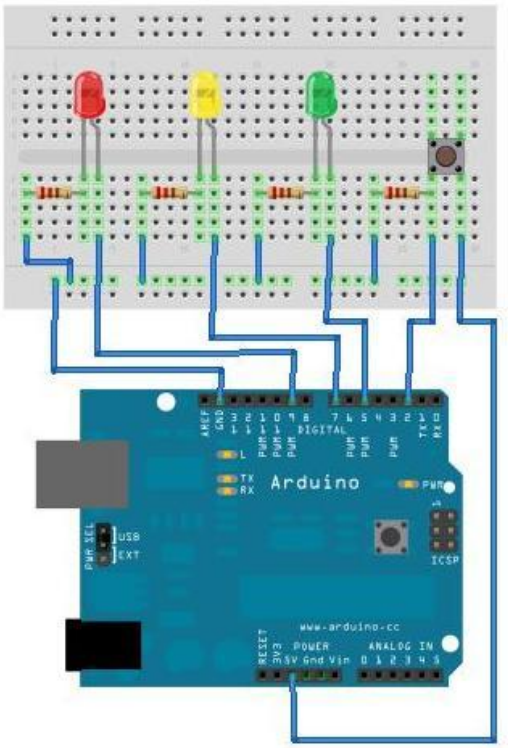


Abbildung 50: TrafficLight - Beispiellösung

Beschreibung
Die Darstellung bzw. der Ablauf der LED Schaltung ist recht einfach und schnell implementiert. Das etwas knifflige Realisieren des Ein- und Ausschaltens der Verkehrsampel stellt in diesem Beispiel die Herausforderung dar.

Der Laufzyklus der Ampel wurde im Programm wie folgt festgelegt:
1) ROT leuchtet
    a. zwei Sekunden warten
2) GELB beginnt ebenfalls zu leuchten
    a. zwei Sekunden warten
3) ROT und GELB werden ausgeschaltet und GRÜN eingeschaltet
    a. drei Sekunden warten
4) GRÜN blinken simulieren
    a. GRÜN ausschalten
        i. Eine halbe Sekunde warten

b. GRÜN einschalten
    i. Eine halbe Sekunde warten
c. Punkt 4 noch zwei Mal wiederholen

5) GRÜN ausschalten
6) wieder bei Punkt 1 beginnen

Hier kann man rasch erkennen, dass dieser Ablauf relativ einfach zu Programmieren ist. Was passiert aber, wenn bei einem ‚Warten' der Button gedrückt wird?
Es wird schnell klar, dass mit der Verwendung der Standardfunktion ‚delay' diese Gegebenheit nicht abgebildet werden kann. Es ist somit notwendig, diese Funktion zu ‚imitieren' bzw. zu ‚überlagern', d. h. eine eigene Variante einer Wartefunktion zu schreiben. Im Beispiel wurde diese ‚delayWithInterrupt' genannt. Diese ermöglicht, in der Wartezeit den Status des Buttons jederzeit auszulesen und falls eine Statusänderung erfolgt ist – sprich der Button gedrückt wurde – einen Abbruch des Ablaufes herbeizuführen und somit die Verkehrsampel auszuschalten.

Im Folgenden wird eine mögliche Variante dieser Ablaufsteuerung mit der Überlagerung der Funktion ‚delay' dargestellt.

Programmcode

```
/*  Beispiel 07 - Traffic Lights with PushButton v1
    Das Programm simuliert eine Verkehrsampel. Durch Drücken eines Buttons kann
    die Verkehrsampel aus- bzw. wieder eingeschaltet werden.
*/

#define BUTTON 2 // gibt den digitalen Pin des Buttons an

#define LED_G 5  // gibt den digitalen Pin des grünen LEDs an
#define LED_Y 7  // gibt den digitalen Pin des gelben LEDs an
#define LED_R 9  // gibt den digitalen Pin des roten LEDs an

int val = 0;     // speichert den Status des Buttons
int oldval = 0;  // speichert den vorherigen Status des Buttons
int state = 0;   // dient der Auswertung des Status
int i = 0;       // Hilfsvariable für For-Schleife

// Initialisierungsfunktion
void setup() {
  pinMode( BUTTON,INPUT ); // digitaler Pin wird als Input für den Button
                           // definiert
  pinMode( LED_G,OUTPUT ); // digitaler Pin wird als Output definiert
  pinMode( LED_Y,OUTPUT ); // digitaler Pin wird als Output definiert
  pinMode( LED_R,OUTPUT ); // digitaler Pin wird als Output definiert
}

// Hauptfunktion
void loop() {
  // Funktion zum Auslesen des Status des Buttons
  readButton();

  // prüfen, ob Status geändert wurde und ...
  if (state == 1) {
    // ... die Simulation der Verkehrsampel starten
    runTrafficLight();
  } else {
    // ... die Verkehrsampel ausschalten
    clearTrafficLight();
  }
}

// Funktion zum Auslesen des Status des Buttons
int readButton( ) {
  // Hilfsvariable, speichert eine Statusänderung des Buttons
  int stateChanged = 0;
```

```
  // Status des Buttons auslesen
  val = digitalRead(BUTTON);

  // prüfen, ob Status geändert wurde ...
  if ((val == HIGH) && (oldval == LOW)) {
    // ... und globale Variable setzen
    state = 1 - state;
    delay(10); // Verzögerung für das Entprellen
    // Änderung des Status speichern
    stateChanged = 1;
  }

  // den gelesenen Wert des Buttons speichern
  oldval = val;
  // Statusänderung retournieren
  return stateChanged;
}

// spezielle Funktion, welche ein 'delay()' durchführt bzw. bei einer
// Statusänderung des Button die 'Wartezeit' abbricht
int delayWithInterrupt( unsigned long t ) {
  // Zeit speichern
  unsigned long t0 = millis();

  // Schleife durchlaufen, maximal bis 't' (übergebene Wartezeit) erreicht oder
  // der Button gedrückt wurde
  while( millis() - t0 <= t ) {
    if( readButton() ) {
      return 1;
    }
  }
  return 0;
}

// Funktion, welche eine Verkehrsampel simuliert
void runTrafficLight() {
  // Red LED
  digitalWrite( LED_R,HIGH );
  if( delayWithInterrupt( 2000 ) ) { return; }
  // Yellow LED
  digitalWrite( LED_Y,HIGH );
  if( delayWithInterrupt( 2000 ) ) { return; }
  digitalWrite( LED_Y,LOW );
  digitalWrite( LED_R,LOW );
  // Green LED
  for( i=1;i<=4;++i ) {
    if( i==1 ) { // first loop
      // for 3 seconds
      digitalWrite( LED_G,HIGH );
      if( delayWithInterrupt( 3000 ) ) { return; }
    } else {
      // for 0.5 seconds
      digitalWrite( LED_G,LOW );
      if( delayWithInterrupt( 500 ) ) { return; }
      digitalWrite( LED_G,HIGH );
      if( delayWithInterrupt( 500 ) ) { return; }
    }
  }
  digitalWrite( LED_G,LOW );
}

// Funktion, um alle LEDs auszuschalten
void clearTrafficLight() {
  digitalWrite( LED_G,LOW );
  digitalWrite( LED_Y,LOW );
  digitalWrite( LED_R,LOW );
}
```