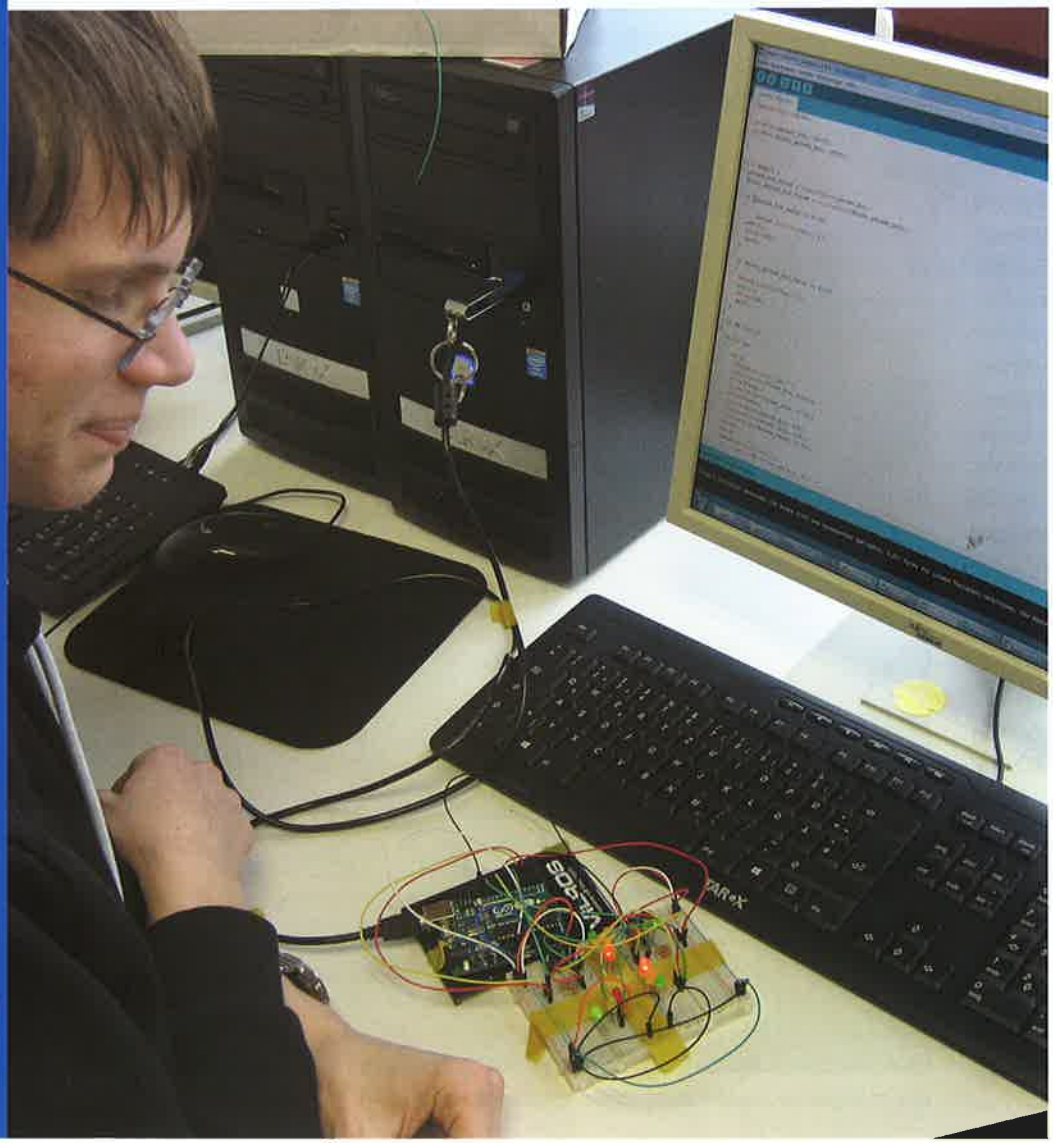
TECHNIK STUNDE 284

INFORMATION UND KOMMUNIKATION



Arduino und ArduBlock

Eine Einführung Teil 1

KLASSE
ab der 9. Klasse

ZEIT
9–12 Unterrichtsstunden

MATERIALIEN
pro Gruppe ein Arduino-Starterpaket, kostenlose deutsche Software

WERKZEUGE
Grundausstattung für Elektronik und PC

KOSTEN
einmalig ca. 40 € pro Schülergruppe

Unterrichtspraxis, IuK, 5

ALSVERLAG

VORBEMERKUNG

Das Thema „Einführung in den einfachen Umgang mit dem Arduino und der ArduBlock-Software" wird in den folgenden beiden Beiträgen der Technikstunde zugänglich gemacht. Der erste Beitrag ist eher theoretisch, hier werden die Software und die Hardware vorgestellt. Der zweite Beitrag ist der praktische Teil, er baut auf Teil 1 auf und beinhaltet die Installation der Software. Die Arbeitsblätter stellen das Herzstück des Themas dar und erfassen die praktischen Aufgaben für die Schülerhand.

WARUM ARDUINO UND ARDUBLOCK?

Im Technikunterricht verlangen eine Reihe von Bildungsplänen mit den Modulen „Information und Kommunikation" oder „Systeme und Prozesse" oder „Steuern und Regeln" nach praktischen und handlungsorientierten Unterrichtskonzepten.

Viele Schulen haben sich aus diesem Grund von LEGO® oder fischertechnik diverse Experimentier- und Baukästen zugelegt. Diese ca. 300 € teuren Systeme lassen vielseitige sowie robuste Hardware-Konstruktionen zu und sind durch visuelle Programmieroberflächen recht einfach zu programmieren.

Preiswertere Mikrocontroller, wie z. B. die BASIC Stamp, sind heute an den Gymnasien im NWT-Unterricht in Verwendung. Leider sind dort Programmierabläufe mit ziemlich kryptischen und englischen Befehlen unvermeidbar (` {$STAMP BS1} oder DEBUG CR, $crystal = 100000 ...).

Der Arduino wurde von der Hard- und Software her im Sinne von „open source" gerade für Quereinsteiger entwickelt. Der millionenfache Verkauf und Siegeszug des Arduino mit seiner USB-Ansteuerung (damit auch ohne extra Stromversorgung programmierbar), den vielseitigen Erweiterungen (für Bluetooth, TFT, WLAN, Roboter), der Kompatibilität für Windows, Mac OS und Linux, der großen Flexibilität und den vielen Dokumentationen im Web lässt sich nicht mehr stoppen. Durch die ständige Weiterentwicklung gibt es mittlerweile sehr vielseitige und leistungsstarke Arduino-Versionen.

Die integrierte Entwicklungsumgebung (IDE) wurde als plattformunabhängige Java-Anwendung u. a. mit Plug-ins dahingehend verbessert, dass man nun nicht mehr in der ursprünglichen Programmiersprache C++ schreiben muss, sondern auch eine visuelle und vereinfachte Programmieroberfläche mit meist deutschen Befehlen verwenden kann.

Dieses kostenlose Plug-in heißt ArduBlock (ähnlich zu „Scratch") und wird in verbesserter Weise vom Fachbereich Mechatronik der Hochschule Reutlingen letsgoing.de angeboten. Dort werden vielfältige Kursmaterialien und Projekte vor allem für Gymnasien zur Verfügung gestellt: wiki.letsgoing.de.

Mithilfe der preiswerten Arduino-Hardware und der einfachen ArduBlock-Software ist nun auch eine unkomplizierte Einbindung in den Projekt- oder Technikunterricht von Haupt- und Realschulen denkbar.

WAS KANN EIN ARDUINO?

Er kann

- **überwachen:** z. B. Taster, Schalter, Kontakte, Magnetkontakte, Schall-, Druck- und Lage-Sensoren
- **steuern:** z. B. Treppenlicht, Anzeigen, Soundanlagen, Motoren, Roboter
- **regeln:** z. B. Heizplatten, Lichtanlagen

=> Mikrocontroller wie Arduinos sind in fast allen technischen Geräten eingebaut. Man findet sie in

Abb. 1: Aufgebaute Ampelschaltung mit dem Programm in ArduBlock

Spielzeugen, Handys, Robotern, Autos, Waschmaschinen, Haussteuerungen, …

Alles geschieht nach dem EVA-Prinzip:

Abb. 2: Das EVA-Prinzip

Hier einige weiterführende Projekte, die man mit dem Arduino so machen kann:

Useless Machines	https://www.youtube.com/watch?v=-PqcCjFaf3I
13 Fun Arduino Projekte	https://www.youtube.com/watch?v=QqiU-OalhiI
LED Cube 3*3*3	https://www.youtube.com/watch?v=GLx6aA75CZY
3-D LED Cube 16*16*16	https://www.youtube.com/watch?feature=player_embedded&v=SyuLhEUImng
Ferngesteuertes Auto	http://www.svenbluege.de/de/blog/hardware/91-blog-de/hardware/136-arduino-controlled-rc-car
E-Gitarre	https://www.youtube.com/watch?v=xkrGyFWPb5o
Zeichen-Maschine	http://vimeo.com/67882372
Verschiedene Roboter	http://becuo.com/arduino-robot
Sound-Anzug	https://incom.org/projekt/2967
Fun LED Lampe Dandy-Light	http://www.interaction-venice.net/iuav11-12lab2/projects/dandylight/
Spur-Roboter	http://meyleankronemann.de/lumibots.html
RC-Auto mit PS3 Controller	https://www.youtube.com/watch?v=5ZptMi1j_w8
3-D-Drucker	https://www.youtube.com/watch?v=pyFZKc356eQ

DIDAKTISCHE ÜBERLEGUNGEN

Diese Unterrichtsarbeit eignet sich gut für eine Projektarbeit, bei der kleine Schülergruppen an einem Arduino arbeiten können.

Diese Einführung kann dazu genutzt werden, im klassischen Sinne mit den Schülern bearbeitet zu werden. Jedoch können sich die Schüler auch selbstständig die Arbeitsblätter anhand der Tipps und Lösungen erarbeiten. Entscheidend ist der problemorientierte und handlungsorientierte Ansatz, der durch praktische Nachahmung und eigene Experimente das Thema verständlich machen soll.

Im naturwissenschaftlichen Unterricht wird in vielen Bildungsplänen die Halbleiterphysik behandelt. Dabei werden die elektrischen Grundgrößen, Halbleiter, pn-Übergang, Diode, LED, Solarzellen und der Transistor bearbeitet. Im Technikunterricht wird meistens passend dazu der praktische Umgang mit diesen Sachverhalten bearbeitet. Oftmals werden nach den Transistorgrundschaltungen Kippstufen und zeitgesteuerte Abläufe über RC-Glieder angegangen.

Die Arbeit mit dem Mikrocontroller bietet sich hier an, schon nach dem naturwissenschaftlichen Teil und den technischen Transistorgrundschaltungen bearbeitet zu werden. Neben der Elektronik können die Schüler auch das Programmieren eines Mikrocontrollers erlernen, den sie selbst in der Praxis weiter nutzen können.

Die Beschäftigung mit dem Programmieren wird in Videoclips von „Start coding!" übrigens schön beworben:
http://start-coding.de

Die Installation des Treibers des Arduinos ist fast der anspruchsvollste Teil. Dieser Prozess kann von einem Netzwerkberater entweder einmal durchgeführt und in ein Image eingepflegt werden. Oder man macht den Prozess einmal mit den Schülern durch. Dies hätte den Vorteil, dass man sich selbst den Aufwand spart und gleichzeitig den Charme, dass die Schüler das auch einmal selbst auf dem eigenen PC durchführen können.

Allgemein gilt: Keine Angst! Kaputt machen kann man nach dem Lesen der Einweisungen eigentlich nichts.

UNTERRICHTSERFAHRUNGEN

Der Arduino-Lehrgang wurde in der zehnten Klasse einer Realschule durchgeführt.

Der Lehrgang ist so gestaltet, dass spielerisch geübt werden kann, wie die reale und virtuelle Welt miteinander kommunizieren können. Den Schülern machte die Arbeit mit dem Arduino Spaß, wenn tatsächlich die LEDs leuchteten, so wie sie es programmiert hatten. Die schnellen Erfolge mit den Aufgaben weckten ihr Interesse und sorgten dafür, dass sie selbstständig den Lehrgang bearbeiteten.

Abb. 3: Schüler bei der Überprüfung des Programms

Abschließend sollten sie sich ein größeres Projekt vornehmen, das sie selbstständig planen, aufbauen und programmieren sollten. Ergebnisse waren u. a. eine „useless machine", eine Ampelsteuerung, eine LED-Lichtorgel und ein LED-Cube.

LERNZIELE FÜR DEN UNTERRICHT

Die Schüler lernen:

- verschiedene elektronische Bauteile theoretisch und praktisch kennen
- verschiedene elektronische Bauteile mit einem Mikrocontroller zu verbinden
- von Sensoren die Daten einzulesen, zu verarbeiten und Aktoren ansteuern zu lassen
- Programmabläufe zu planen und in einem Programmablaufplan aufzuzeichnen
- mithilfe einer Software Hardware selbstständig zu programmieren
- selbstständig mit einer Anleitung und dem angebotenen Hilfesystem zu arbeiten
- Aufgabe, Funktion und den groben Aufbau eines Informations-und-Kommunikations-Systems (IuK) kennen
- Vor- und Nachteile programmierter IuK-Systeme zu beurteilen
- abzuschätzen, welche Kompetenzen z. B. in Mechatronic- oder Elektronikberufen nötig sind.

VORAUSSETZUNGEN

Inhaltliche Voraussetzungen

Folgende Bereiche sollten im Physik- und Technikunterricht zuvor bearbeitet worden sein:

- Spannung, Stromstärke, Widerstand
- Strom- und Spannungsteiler, Vorwiderstandsberechnungen
- LED und die wichtigsten Sensoren
- sicherer Umgang mit elektrischen und elektronischen Bauteilen (ESD)
- grundlegende Kenntnisse im richtigen Umgang mit Computern.

Sollten diese Grundlagen nicht komplett vorhanden sein, sollten die fehlenden Kenntnisse möglichst bei den anstehenden Problemen eingeführt werden.

Materielle Voraussetzungen

Benötigt wird pro Schülergruppe:

- ein Arduino-Uno-R3-Starter-Kit für ca. 40 €
- eine Pinzette oder Spitzzange zum Stecken der Bauteile auf dem Board
- einen PC oder ein Notebook und
- ein Multimeter.

Die wichtigsten Komponenten

In Abb. 4 sind die wichtigsten Komponenten zu sehen, die bei fast jedem Arduino-Set mitgeliefert werden:

- verschiedenfarbige LEDs
- verschiedene Widerstände (10 kΩ, 220 Ω)
- RGB-LEDs
- Taster
- LDR
- Poti (10 kΩ)
- Piezo-Summer
- Steckbrett mit Kabel

Je nach Form verwendet man auch männliche Stecker oder weibliche Kupplungen auf dem Steckbrett.

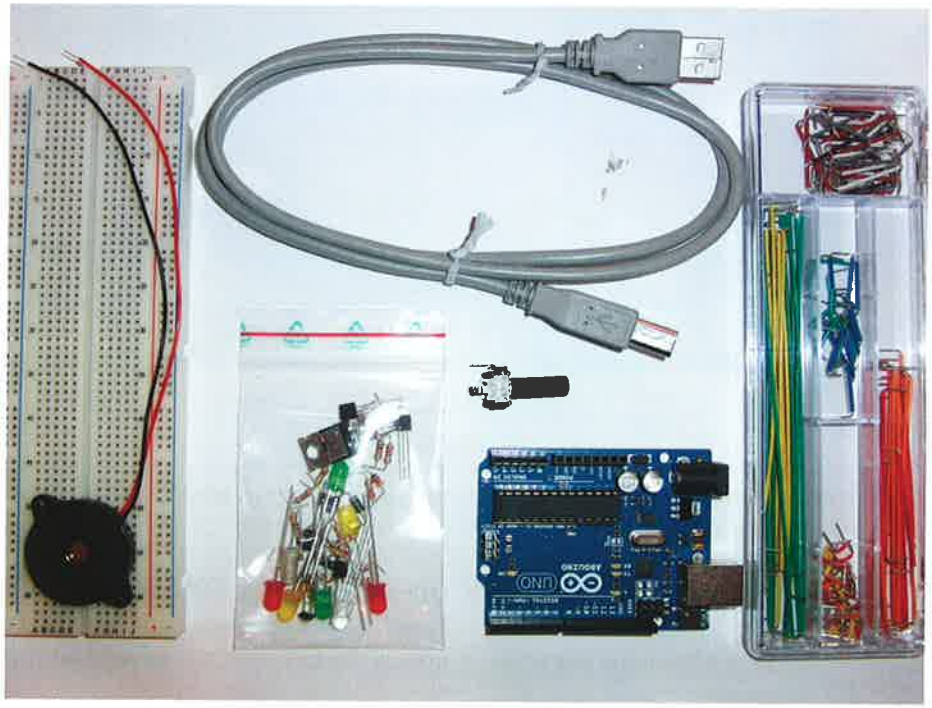


Abb. 4: Arduino-Starter-Kit

TECHNIKSTUNDE · © 2015 ALS-Verlag GmbH
www.als-verlag.de · info@als-verlag.de
Voltastraße 3 · 63128 Dietzenbach
Schriftleitung: Dietmar Waibel, Schwäbisch Gmünd

Größere Sets enthalten Servos, Motoren, Schrittmotoren, Bewegungsmelder, Ultraschallsensoren, LED-Matrix-Display, … , die für den Einstieg aber nicht unbedingt benötigt werden.

DIE HARDWARE – DER ARDUINO

Die Hardware besteht aus einem Mikrocontroller (ATmega328) mit analogen und digitalen Ein- und Ausgängen auf einem Board. Die Entwicklungsumgebung basiert auf der Programmiersprache „Processing", die auch technisch weniger versierten Menschen den Zugang zur Programmierung und zu Mikrocontrollern erleichtern soll.

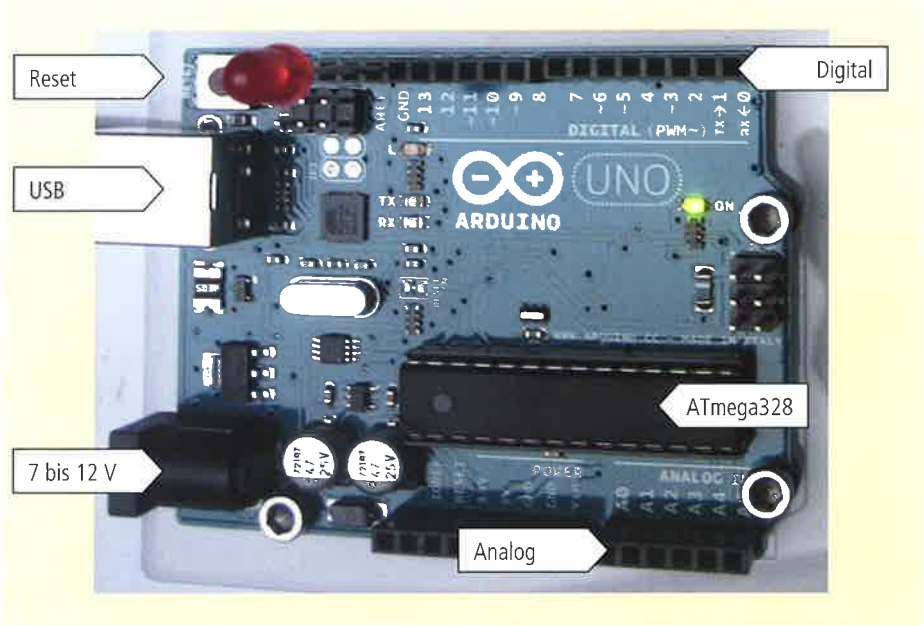


Abb. 5: Arduino-Platine

Das erste Arduino-Board wurde 2005 von Massimo Banzi und David Cuartielles entwickelt und nach einem Pub benannt. David Mellis schrieb die Programmiersprache dazu.

Konzeptionell werden alle Boards über eine serielle Schnittstelle programmiert.

Der Mikrocontroller ist mit einem Boot-Loader (Betriebssystem wie z. B. Windows) vorprogrammiert, wodurch die Programmierung direkt über die serielle Schnittstelle ohne externes Programmiergerät erfolgen kann.

Mittlerweile gibt es viele robuste Versionen von Arduinos und zusätzlichen, ansteckbaren Erweiterungen (sogenannten „Shields").

Über einen Pin kann max. 40 mA Stromstärke fließen; über den GND 200 mA. Dies reicht für viele LEDs gut aus, macht jedoch bei Motorsteuerungen und Servos schnell eine externe Stromversorgung notwendig.

DIE ARDUBLOCK-SOFTWARE

ArduBlock ist **eine grafische Entwicklungsumgebung.** Sie kann direkt aus der Arduino-Oberfläche aufgerufen werden.

Mit ArduBlock können die Programmbefehle wie Bausteine bzw. Blöcke leicht kombiniert werden. Dadurch erhalten Anfänger sehr schnell Erfolgserlebnisse, die motivieren, sich tiefer mit dem Programm auseinanderzusetzen.

Der erste Kontakt mit textbasierten Programmiersprachen wie Java oder C demotiviert viele Anfänger beim Einstieg in das eigentlich spannende Thema Softwareentwicklung. Der Fokus liegt dort zwangsläufig auf – für die Logik von Algorithmen eigentlich unwichtigen – Details, wie die sklavische Einhaltung der Syntaxregeln. Bevor dort nur das einfachste Programm läuft, sind sehr viele Fehler zu vermeiden.

Die Programmbefehle liegen bei ArduBlock als grafische Blöcke vor. **Verschiedene Farben** kennzeichnen, um welche Art von Befehlen es sich handelt: So sind z. B. alle Befehle, die mit der **Steuerung** des Programms zu tun haben, **gelb** hinterlegt. Befehle, die einen **Output** erzeugen sind **blau.** Befehle, die **Werte einlesen (Input)** sind **hellblau** (siehe Abb.6).

In weiteren Menüs findet man mathematische oder logische Vergleichsoperatoren. Blöcke für Variablen oder Kommunikation sind ebenfalls in eigenen Menüs zusammengefasst.

Nicht alle Befehle passen aneinander. ArduBlock sorgt dafür, dass nur so ein Programmcode erzeugt wird, der syntaktisch korrekt ist. Normalerweise besteht ein Programm aus einem SETUP, das einmalig am Anfang des Programms ausgeführt wird. Dort werden Befehls-Bibliotheken eingebaut und aufgerufen. Der zweite Teil ist der sich wiederholende LOOP. Dort ist die Hauptschleife mit den Unterprogrammen enthalten. ArduBlock kümmert sich auch um das Setup. Das heißt, dass man sich erst einmal keine Gedanken um die Variablendeklaration oder die Festlegung der einzelnen Pins

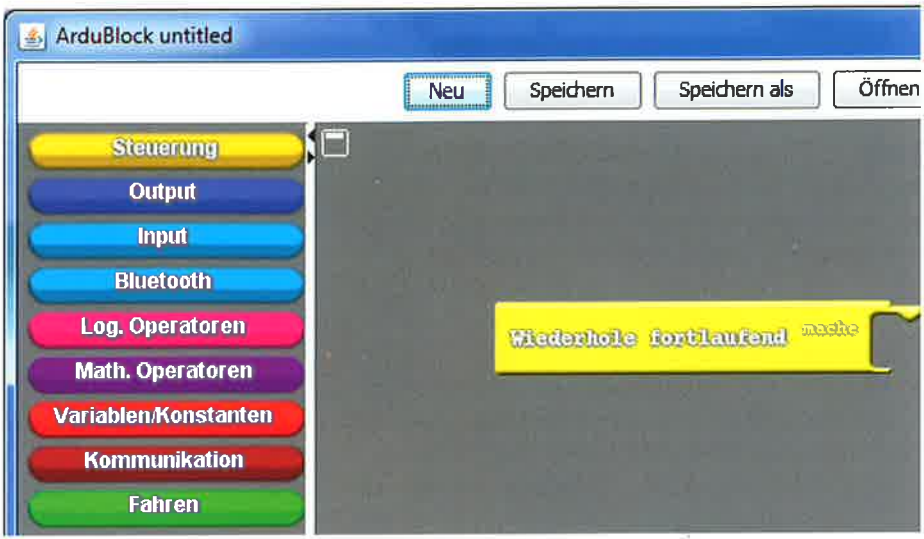


Abb. 6: ArduBlock-Fenster mit den farblich gekennzeichneten Befehlsarten

als Input- oder Outputpins machen muss. **Dadurch werden viele Anfängerfehler automatisch vermieden.**

Wenn ein Programm aus verschiedenen Blöcken zusammengebaut wurde, kann es mit dem Button „Hochladen" auf den Arduino übertragen werden. Aus den Blöcken wird ein Arduino-Code erzeugt, der ab und an im Arduinofenster betrachtet werden kann und sollte. Gleichzeitig wird das Programm in (für Menschen völlig unleserliche) Maschinensprache übersetzt – das nennt man „kompilieren". Beim Kompilieren wird überprüft, ob im Programm grammatikalische Fehler sind. Da sich ArduBlock um die ganze Zeichensetzung kümmert, sollte der Code in der Regel ohne Fehler auf den Arduino übertragen werden.

Anmerkung: Wenn ein Code im Arduinofenster verändert wird, kann man ihn über den Button „Upload" auf den Arduino übertragen – die Veränderungen werden allerdings nicht in das ArduBlock-Fenster bzw. -Programm übernommen!

WEITERE INFORMATIONEN

Dieser Artikel umfasst nur einen kleinen Teil der Arbeit mit dem Arduino. Weitere Informationen findet man hier:

Offizielle Homepage	http://www.arduino.cc/ und in deutsch: http://playground.arduino.cc/Deutsch/HomePage
Für Lehrer, Eltern, Coder, ...	http://starthardware.org/
Fritzing – Layoutsoftware	http://fritzing.org/home/
Funduino	http://funduino.de/index.php/vorwort
letsgoING Schüler-Projekte + ArduBlock	http://letsgoingwiki.reutlingen-university.de/mediawiki/index.php/Hauptseite https://Github.com/letsgoING
ArduBlock Video Tutorial	https://www.youtube.com/watch?v=akk1sNXk9C8
Unterrichtsideen	http://popovic.info/html/arduino/arduinoUno_1.html

AUSBLICK

Im nächsten Beitrag der Technikstunde wird die Installation der Software ausführlich beschrieben.

Die dort beigelegten Arbeitsblätter sind das zentrale Element für Schüler und Lehrer zur Bearbeitung des Themas.

Des Weiteren werden inhaltliche und schulische Ausblicke für die fortgeschrittene Arbeit mit dem Arduino gegeben.

Autor: Andreas Luckert, Reutlingen

TECHNIKSTUNDE · © 2015 ALS-Verlag GmbH
www.als-verlag.de · info@als-verlag.de Voltastraße 3 · 63128 Dietzenbach
Schriftleitung: Dietmar Waibel, Schwäbisch Gmünd

VORBEMERKUNG

In der letzten Technikstunde haben wir den Arduino bereits theoretisch kennengelernt. Es wurde ein Überblick über die Software sowie die Hardware gegeben, sodass nun die Installation der Software folgt. Sie wird hier Schritt für Schritt aufgeführt und kann gemeinsam mit den Schülern vorgenommen werden. Da die Treiberinstallation etwas knifflig ist, kann diese Aufgabe vorab auch ein Netzwerkberater übernehmen.

DIE SOFTWARE-INSTALLATION (WIN)

Die Installation muss bei einem PC in der Regel nur einmal durchgeführt werden. Wechselt man des Öfteren den PC, ist es praktisch, die ganze Software auf einen USB-Datenträger zu speichern, sodass man dann nur noch die Treiberinstallation durchführen muss.

Die Arduino-Software

Die Hochschule Reutlingen hat den Installationsaufwand auf ein Minimum reduziert.

Hier kann man sich die Arduino-Version 1.6 inklusive ArduBlock herunterladen (siehe Abb. 1): https://Github.com/letsgoING

Unter dem zweiten Punkt „Arduino" gelangt man zum nächsten Fenster (siehe Abb. 2):

Abb. 1: Homepage der Hochschule Reutlingen

Abb. 2: Download der ZIP-Datei

Man drückt mit der linken Maustaste auf „Download ZIP" und lädt die ca. 205 MB große Datei „Arduino-master.zip" herunter (siehe Abb. 3).

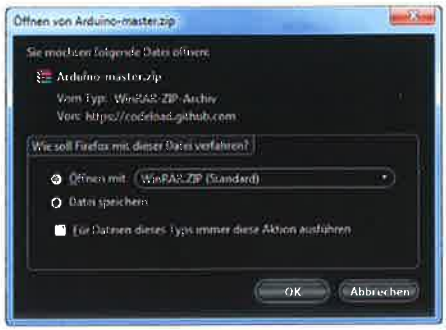


Abb. 3: Fenster zum Abspeichern der ZIP-Datei

Mit einem Doppelklick auf diese Datei entpackt man sie in ein beliebiges Verzeichnis, z. B. Eigene Dateien oder auf einen USB-Stick (siehe Abb. 4).

Dieser Vorgang des Entpackens dauert einige Minuten, da über 8 000 Dateien extrahiert werden müssen.

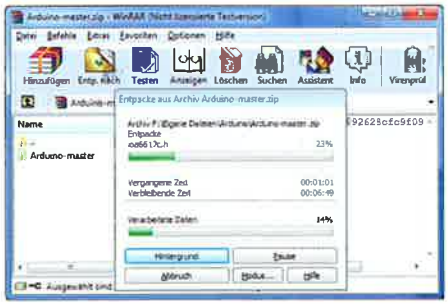


Abb. 4: Entpacken der gezippten Dateien

Im entpackten Verzeichnis Arduino-master/Arduino-1-6 ist es vorteilhaft, mit der rechten Maustaste eine Verknüpfung für die arduino-Anwendung auf dem Desktop zu erstellen (Drag & Drop auf Desktop => Verknüpfung hier erstellen) (siehe Abb. 5).

Abb. 5: Arduino-Symbol der Desktop-Verknüpfung

Den Arduino mit dem USB-Kabel verbinden.

Durch Doppelklick auf das Arduino-Icon wird die Arduino-Software gestartet.

Wenn der Arduino zum ersten Mal mit dem PC verbunden wird, schlägt wahrscheinlich die Inbetriebnahme fehl, weil noch der Treiber fehlt.

Die Treiberinstallation

Wenn Windows anzeigt, dass die Gerätetreibersoftware nicht installiert wurde, dann (angemeldet als Admin) auf diese Sprechblase klicken (siehe Abb. 6).

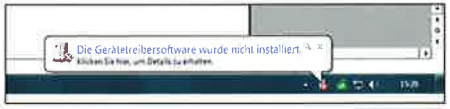


Abb. 6: Fehlermeldung wegen fehlender Treiber-Software

Oder (als Admin) die Systemsteuerung öffnen => System & Sicherheit => System => Gerätemanager (siehe Abb. 7).

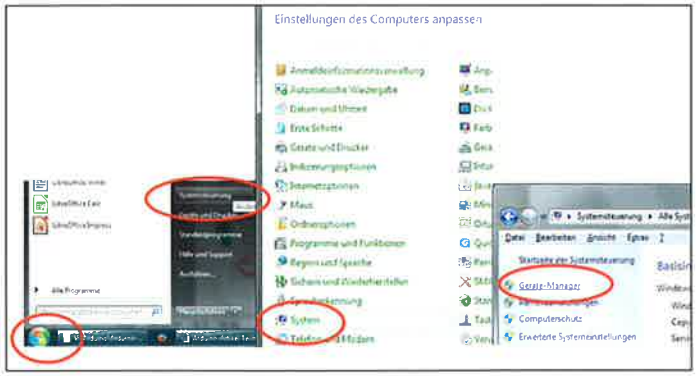


Abb. 7: Aufrufen des Geräte-Managers in der Systemsteuerung

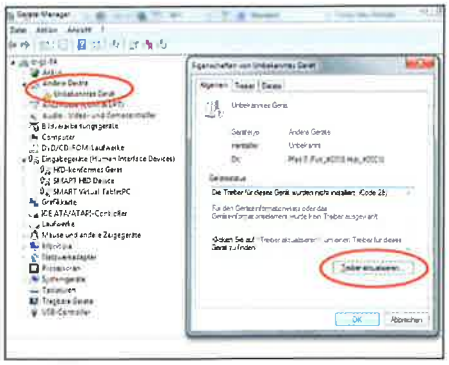


Abb. 8: Hinweis auf das Gerät mit fehlendem Treiber

Im Unterpunkt Ports (COM/LPT) erscheint „Unbekanntes Gerät" (siehe Abb. 8).

Mit der rechten Maustaste „Unbekanntes Gerät" anklicken und „Treiber aktualisieren" auswählen.

„Auf dem Computer nach Treibersoftware suchen" wählen und das Verzeichnis „...Arduino\Arduino-master\Arduino-1-6\Drivers" auswählen. Mit „Weiter" installiert Windows den richtigen Treiber (siehe Abb. 9).

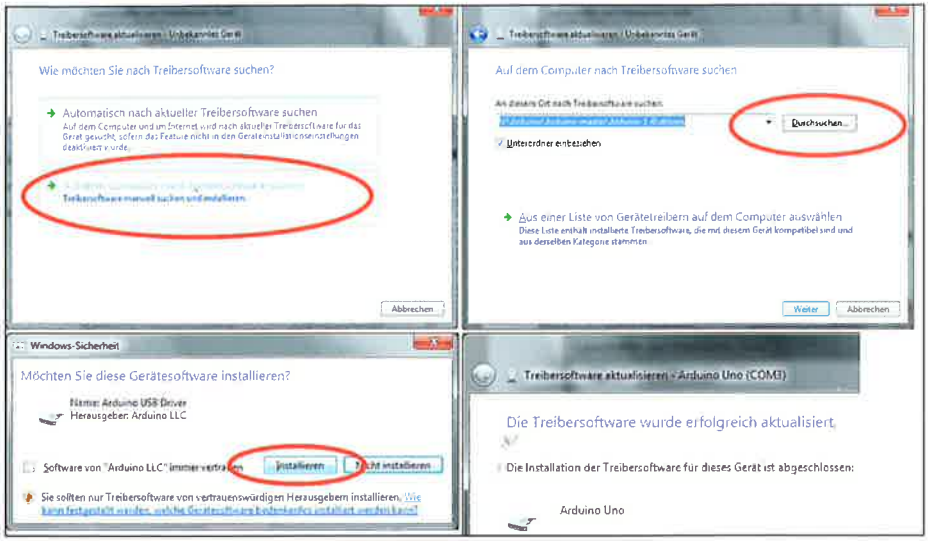


Abb. 9: Installation der Treibersoftware

Jetzt wird der richtige COM-Port angezeigt, den man später bei der Arduino-Software einstellen muss (siehe Abb. 10).

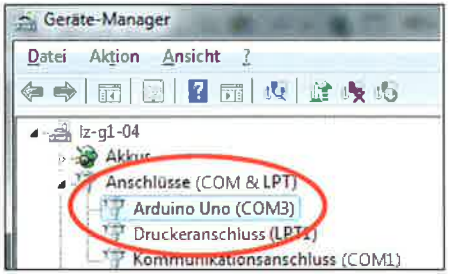


Abb. 10: Anzeige des Ports

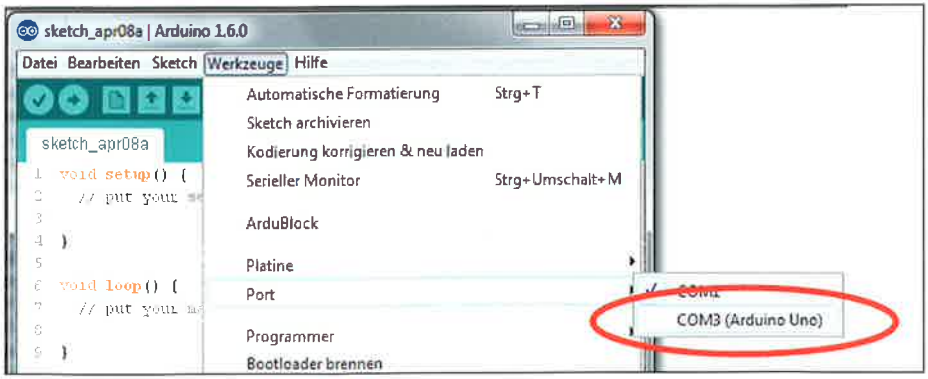


Abb. 11: Auswählen des richtigen Ports in der Arduinosoftware

Danach muss ArduBlock gestartet werden.

Unter „Werkzeuge" => „Port" muss nun der richtige COM-Port ausgewählt werden (im Beispiel Haken bei COM3 setzen) (siehe Abb. 11).

Unter „Werkzeuge" nun ArduBlock ausführen (siehe Abb. 12) und so erhält man dieses zweite Programmierfenster (siehe Abb. 13):

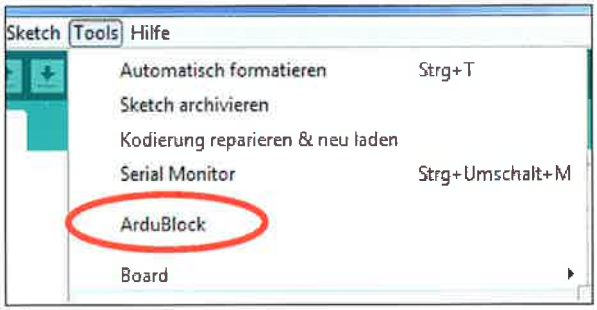


Abb. 12: Aufruf des ArduBlock-Fensters

Abb. 13: Ansicht des ArduBlock-Fensters

ARBEITSBLÄTTER

Damit ist die Installation der Software abgeschlossen. Nun können sich die Schüler den Aufgaben auf den Arbeitsblättern widmen.

Die Arbeitsblätter stellen den für diese Unterrichtseinheit wichtigsten Teil dar. Mit ihrer Hilfe können die Schüler selbstständig mit Arduino und ArduBlock umgehen und auch bereits schwierigere Aufgaben alleine bewältigen.

Zur Selbstkontrolle bestehen die letzten drei Arbeitsblätter aus Lösungen, die die Schüler mit ihren Ergebnissen abgleichen können. Der Lehrkraft bleibt die Entscheidung vorbehalten, ob sie die Lösungen austeilen möchte oder nicht.

Alternativ können die Schüler auch im Internet nach weiteren Informationsmaterialien zum Arduino und zu ArduBlock recherchieren. Dort lassen sich viele Tutorials finden, die den Einstieg erleichtern. Statt der offenen Recherche kann die Lehrkraft aber auch entsprechende Dateien für eine geschlossene Internetrecherche bereitstellen.

AUSBLICK

Inhaltlich

Der Lehrgang mit den angehängten Arbeitsblättern ist ein erster Schritt in die Welt der Mikrocontroller und deren Programmierung. Einige Punkte, die ich auch für wichtig erachte, möchte ich hier noch erwähnen.

Die Pulsweitenmodulation bei einer RGB-LED

In Zukunft werden zunehmend RGB-LEDs verbaut, sodass es für die Schüler eine sinnvolle Übung ist, sich mit der Ansteuerung und Farbmischung einer RGB-LED auseinanderzusetzen (siehe Abb. 14). Wenn man in einen Tischtennisball ein 5 mm großes Loch bohrt und diesen über diese RGB-LED stülpt, erhält man beeindruckende Farben.

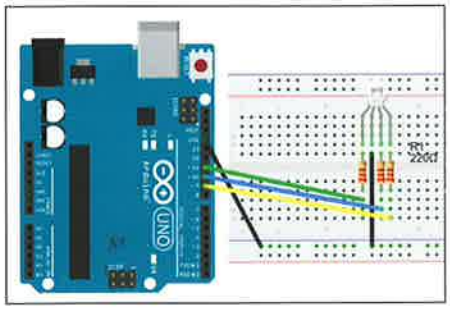


Abb. 14: RGB-LED mit Verkabelung

Michael Herrmann hat dazu ein bewährtes Aufgaben- und Lösungsbuch („Aufgaben RGB") geschrieben, das man über ihn beziehen kann: http://lets-goingwiki.reutlingen-university.de/mediawiki/index.php/Kursmaterial:_Aufgabenstellungen_%28Auszug%29.

Das Ansteuern von Motoren

Motoren sind natürlich ein für Jugendliche sehr spannendes Thema, da diese für bewegte Fahrzeuge und Roboter essentiell sind. Leider ist dies für den Betrieb mit Arduinos bei beschränktem

Budget kein einfaches Thema, da sehr schnell die maximale Stromstärke der Ausgänge von 400 mA überschritten wird.

Eine einfache Möglichkeit ist der Servomotor, der Jugendlichen vor allem von funkferngesteuerten Modellen bekannt sein dürfte. Servomotoren übernehmen z. B. bei einem RC-Auto die Steuerung der Lenkung. So ein Servomotor kann gut über den Arduino gesteuert werden (siehe Abb. 15).

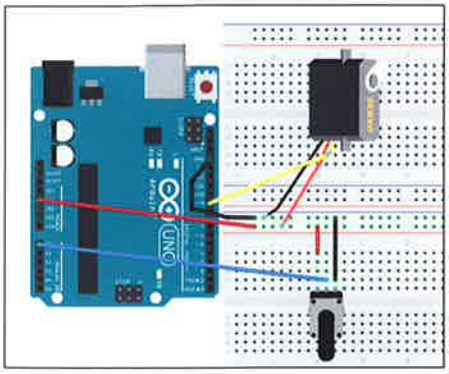


Abb. 15: Servo-Motor und seine Verkabelung mit einem Poti

Bei mehreren Motoren ist eine zusätzliche Stromversorgung z. B. über Batterien oder ein Netzteil notwendig.

Hier möchte ich auf den Beitrag von Prof. Dr. S. Mack verweisen, der das Thema „Legoino: Low Cost Lego-Roboter auf Arduinobasis" ausführlich bearbeitet und eine preiswerte Lösung durch den Umbau von 180°-Servos gefunden hat (http://www.tec.reutlingen-university.de/prof-mack/aktuelle-projekte/legoino/).

Die Fritzing-Software

Viele der symbolischen Darstellungen dieses Beitrages sind mit diesem kostenlosen Programm erstellt worden. Hiermit lassen sich sehr schnell und einfach mit Drag & Drop elektronische Schaltungen erstellen, zeichnen und interaktiv testen. Mit dem Programm kann auch das Layout einer Leiterplatte automatisch erzeugt werden, die man sich mit der Fritzing-Fab auch produzieren und nach Hause senden lassen kann (http://fritzing.org/media/uploads/publications/FritzingInfoBroschuereDE.pdf).

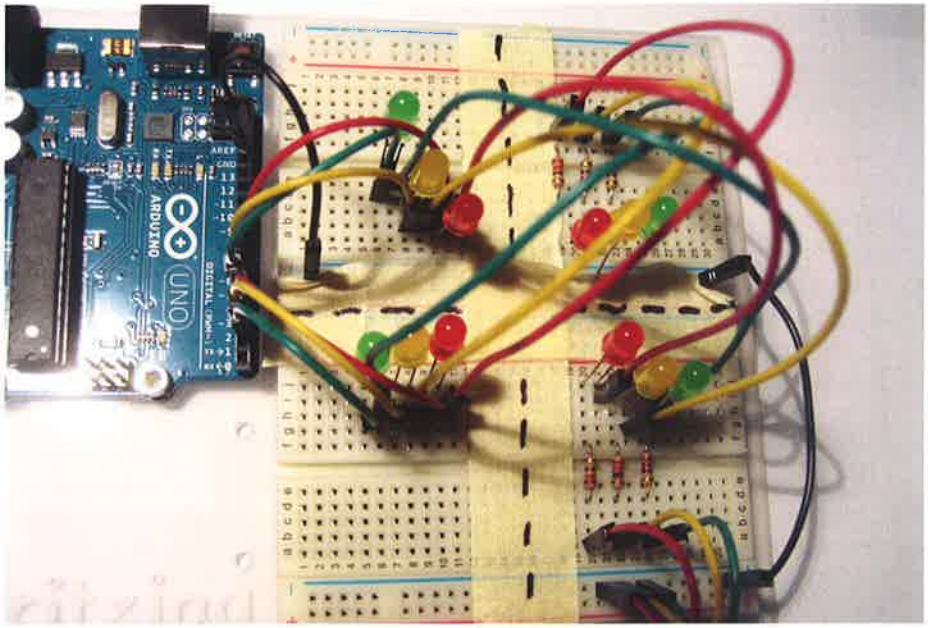


Abb. 17: Ampelschaltung auf zwei Steckbrettern

PIN		1	2	3	4	5	6	7	8	9	10
4	Grün	X									X
5	Gelb		X							X	
6	Rot			X	X	X	X	X	X	X	
7	Grün					X	X				
8	Gelb				X			X			
9	Rot	X	X	X	X				X	X	X

Abb. 18: Hilfreiches Prozessdiagramm für eine Ampelschaltung

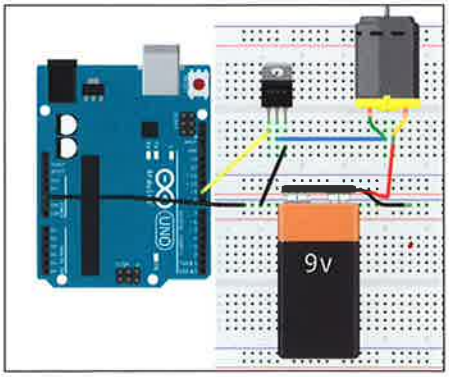


Abb. 16: Grafik der Fritzing Software, die eine Motoransteuerung über eine externe 9-V-Spannungsversorgung zeigt

Schulisch

Je nach Zeit, Lust und Fähigkeit der am Unterricht Beteiligten lässt sich noch eine Phase anschließen, in der die Schüler selbst versuchen sollten, ein größeres Werkstück zu schaffen.

Folgende Arbeiten sind bei uns entstanden bzw. möglich:

Ampelschaltung

Eine einfache Ampelschaltung an einer Straßenkreuzung. Diese kann durch Bedarfstaster ergänzt werden. Dazu ist ein Prozessdiagramm für den Ablauf sehr hilfreich (siehe Abb. 17 und 18).

Useless machine

Mit dem mitgelieferten Servomotor lässt sich eine Klappe in einer Holzbox öffnen und über eine Feder wieder schließen. Dies kann für lustige Spielchen genutzt werden.

Eine Demonstration dafür zeigt dieses Youtube-Video: https://www.youtube.com/watch?v=-Pqc-CjFaf3I.

LED-Cube

Diese Aufgabe ist wirklich nur etwas für Schüler, die viel Zeit, Geld, Wissen und Motivation mitbringen. Da es aber ein cooles Projekt ist, möchte ich es hier auch kurz vorstellen. Je nach Größe benötigt man z. B. 1000 RGB-LED (40 € über Amazon), Draht (10 €), Platine mit ATMega2560-16AU (110 €). Zum Zusammenlöten des Würfels mit verschiedenen Lötvorrichtungen werden über 100 Stunden benötigt, das Planen und Entwickeln einer Platine wird ca. 30 Stunden Zeit benötigen. Das anschließende Programmieren des Würfels wird nochmals viele Stunden Zeit in Anspruch nehmen.

Hier das Ergebnis in YouTube-Videos:
https://www.youtube.com/watch?v=5Ip2_RDZ5jE
https://www.youtube.com/watch?v=3K8i0JQzx2w

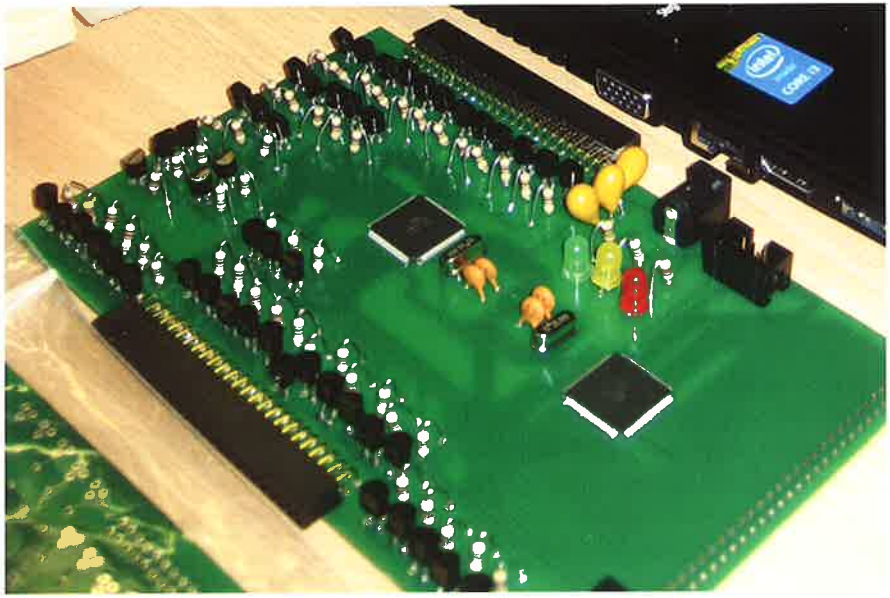


Abb. 19: Selbst entwickelte Platine

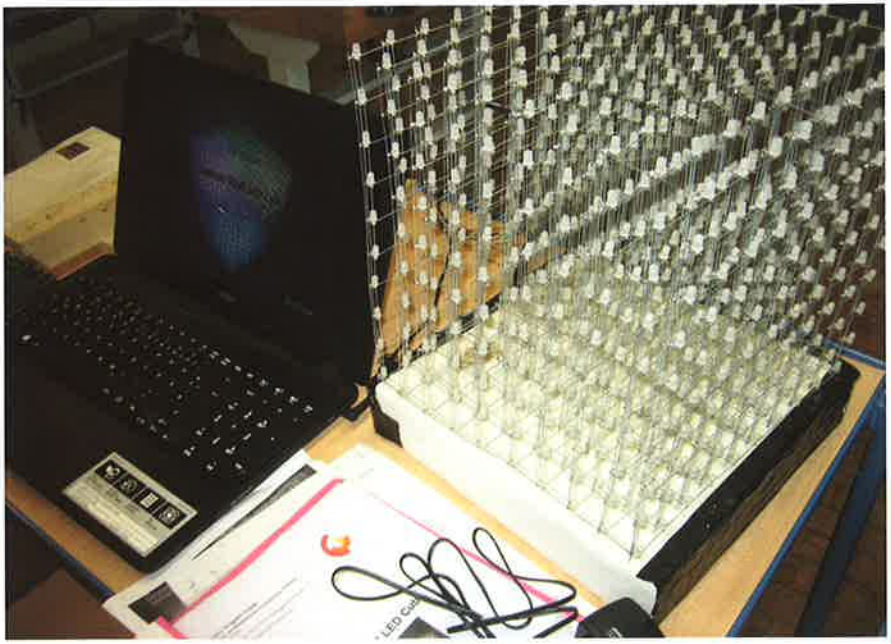


Abb. 20: 1000er-RGB-LED-Cube

Autor: Andreas Luckert, Reutlingen

TECHNIKSTUNDE · © 2015 ALS-Verlag GmbH
www.als-verlag.de · info@als-verlag.de
Voltastraße 3 · 63128 Dietzenbach
Schriftleitung: Dietmar Waibel, Schwäbisch Gmünd

Bauteile anschließen

Das Arduino-Board besitzt **6 analoge Eingänge**, über die Werte zwischen 0 und 1023 eingelesen werden können. Die **14 digitalen Anschlüsse** (Pins) sind sowohl zur Ein- als auch zur Ausgabe (0 oder 1) geeignet.

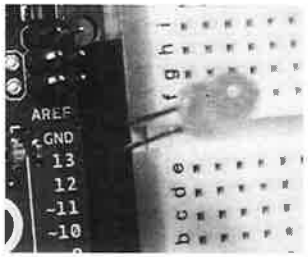
6 Pins (Pin 3, 5, 6, 9, 10 und 11) können auch als **analoge Ausgänge** verwendet werden, indem ein 8-Bit-Wert zwischen **0 und 255** mittels **Pulsweitenmodulation (PWM)** ausgegeben wird. Diese Pins sind mit dem ~ -Symbol markiert.

Über einen Pin kann eine max. Stromstärke von **40 mA** fließen; über den GND max. 200 mA. Dies reicht für viele LEDs gut aus, macht jedoch bei Motorsteuerungen und Servos bald eine externe Stromversorgung notwendig.

Pin 0 und 1 werden in der Regel nur vom Profi für die serielle Kommunikation verwendet.

Ein erster Test!

Du testest mit einer LED, ob der Arduino mit der Software harmoniert. Zuerst wird alles **spannungsfrei** gesetzt und elektrostatisch entladen (ESD). (Die USB-Symbole an der Seite sollen dich stets daran erinnern.)



Jetzt steckst du **eine LED mit dem langen Fuß (die Anode) in PIN 13 rein.** Den **kurzen Fuß** (die Kathode) in den Kontakt GND (–).

Der Pin 13 kann mit der Software auf **HIGH** gesetzt werden, was bedeutet, dass dann eine Spannung von 5 V zwischen Pin 13 und GND (Ground = **LOW**) anliegt. So leuchtet deine LED.

Normalerweise würde nun die LED zerstört werden, da du keinen Vorwiderstand eingefügt hast. Da aber der Pin 13 bereits über einen eingebauten Vorwiderstand verfügt, klappt dein Test ausnahmsweise auch so.

Jetzt kannst du das **USB-Kabel in den PC** einstecken und das kommende Programm hochladen.

Aufgabe 1

Berechne den üblicherweise notwendigen Vorwiderstand.

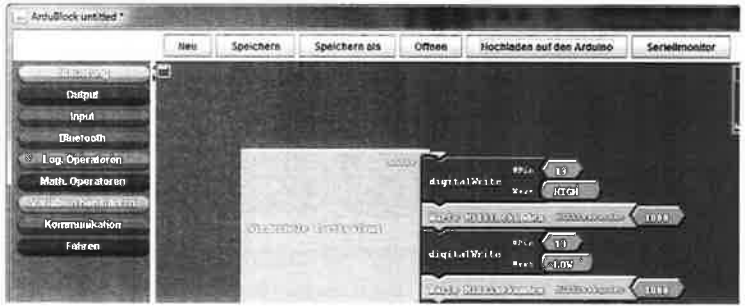
Name:		Arduino und ArduBlock	Maßstab:
Klasse:	Datum:	Aufgabe 1	Blatt-Nr.:

Das erste Programm

Die Blöcke findest du in den jeweiligen Menü-Punkten auf der linken Seite. Diese können per Drag & Drop in den Editor nach rechts hineingezogen werden.

Zum Ändern des Pin-Wertes von 1 auf 13 genügt ein Klick in das jeweilige Symbol und die Eingabe des richtigen Wertes 13 mit der Tastatur. Über den roten Abwärtspfeil lassen sich ebenso Werte verändern. HIGH oder LOW ist eine sogenannte Boolsche Variable, die nur zwei Zustände (aus oder ein bzw. 0 oder 1) speichern kann.

Mit der Maus kannst du die Blöcke hin- und herschieben. Ein Klick-Ton zeigt die richtige Verbindung der Blöcke an. Nicht mehr benötigte Blöcke können so auch wieder nach links rausgeschoben werden. Übernimm nun dieses Programm (das Programm wird auch als Sketch bezeichnet):



Um dem Arduino das Programm zu übermitteln, musst du auf **„Hochladen auf den Arduino"** klicken. Wenn keine Fehlermeldung kommt, geht nun die LED abwechselnd an und aus.

Aufgabe 2

1. Verändere die Zeiten.
2. Speichere das Programm unter deinem Ordner „Meine Arduino-Programme" als „Blinktest" ab.
3. Programmiere nun ein SOS-Signal: 3 * kurz – 3 * lang – 3 * kurz – Pause
4. Speichere das Programm unter deinem Ordner „Meine Arduino-Programme" als „SOS" ab.

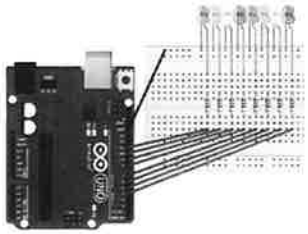
Tipp: Hier hilft ein rechter Mausklick auf „KLONEN" sehr.

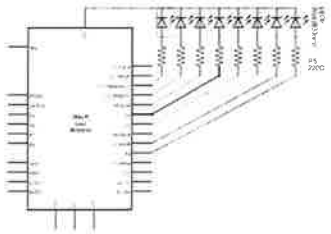
Bonus: Programmiere deinen Namen im Morse-Alphabet.

Name:		Arduino und ArduBlock	Maßstab:
Klasse:	Datum:	Aufgabe 2	Blatt-Nr.:

Das Lauflicht

Baue jetzt ein kleines Lauflicht mit 8 LEDs auf dem Steckbrett:



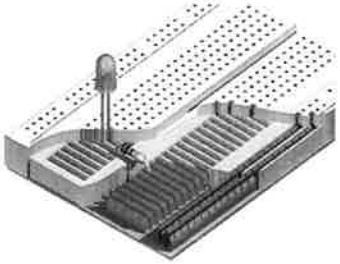


Dazu wird die **Masse (GND) mit der obersten Minus**-Schiene verbunden.

Danach werden die LEDs mit den **kurzen Beinen** auf die obere Minusschiene gesetzt.

Jede LED benötigt einen passenden Vorwiderstand mit ca. 220 Ω.

Das **Steckbrett** verbindet die Löcher mit den unterirdischen Metallschienen. Die Verbindungen sind i. d. R. wie in der Abbildung rechts.



Prüfe mit dem Multimeter die Verbindungen auf dem Steckbrett gründlich durch!

Die Widerstände werden mit den digitalen Ausgängen **D4, D5, ..., D11** verbunden (siehe Abbildungen).

Aufgabe 3

1. Programmiere ein Lauflicht, das von rechts nach links blinkt. Den Pausenwert kannst du auf z. B. 200 ms setzen.

Tipp: Schalte die LED an, lass den Computer warten und schalte dann erst die LED wieder aus.

2. Willst du die Pause insgesamt erhöhen oder verkleinern, musst du ziemlich häufig die Werte verändern. Dies kannst du so besser programmieren:

Setze deine erste analoge Variable. Sie kann jeden beliebigen Namen tragen und kann hier eine Zahl zwischen –32767 und +32767 einnehmen (sie ist eine sogenannte analoge Integer-Variable).

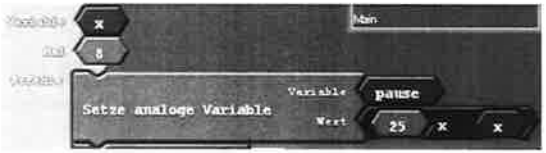
Zu Beginn musst du mit dem roten Block in die Variable einen Wert z. B. 50 setzen. Danach ersetzt du alle Zahlen bei Millisekunden mit der Variable „Pause".

3. Probiere nun dein Programm mit verschiedenen Pause- bzw. Taktzeiten (1–1000).

Name:		**Arduino und ArduBlock**	Maßstab:
Klasse:	Datum:	**Aufgabe 3, Teil 1**	Blatt-Nr.:

4. Verwende dazu nun eine Wiederhol- und Zählschleife mit der Variablen x. Diese Variable x zählt 8 Durchgänge von 1 auf 8 hoch (das Ende 8 steht nach „Hal").

Nutze diese Variable x, um die Pause bei jedem Durchgang der Zählschleife steigern zu lassen (**Tipp:** math. Operation: x * 25).

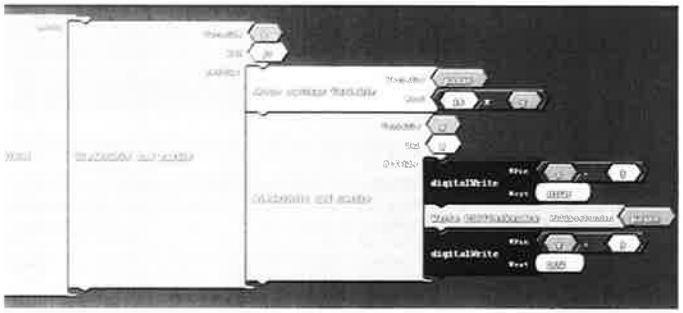


5. Noch schwerer und variabler:
Da du 8 LEDs hast, kannst du auch statt der Variablen x die Variable pin zur Ansteuerung der LED benutzen und so das Programm wesentlich verkürzen.

Dieser Bereich wiederholt sich von Pin 4 bis 11 und kann daher mit einem variablen Pin-Wert ablaufen.

Wir verwenden die Pin 4 bis 11 und können daher die Variable jedoch nicht genauso übernehmen. (**Tipp:** eine math. Operation ist wieder notwendig: Pin + 3 bzw. x + 3).

Die Pause wird durch eine weitere Zählschleife verändert. Hier ein Beispiel:



Probiere dieses kleine Programm aus und verändere es nach Belieben, um dessen Ablauf besser zu verstehen.

Schaue immer wieder nach dem Quellcode, der im Arduino-Fenster erzeugt wird und versuche zu verstehen, wie diese Programme aufgebaut sind.

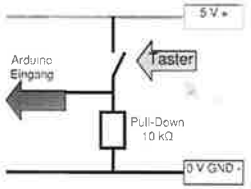
Name:		**Arduino und ArduBlock**	Maßstab:
Klasse:	Datum:	**Aufgabe 3, Teil 2**	Blatt-Nr.:

Die digitale Eingabe

Du hast bisher die digitale Ausgabe umgesetzt: einen Pin entweder HIGH oder LOW gesetzt (1 oder 0 bzw. wahr oder falsch).

Jetzt soll ein Taster abgefragt werden (Input bzw. Read). Ein Taster ist wie ein Schalter, der, solange er gedrückt wird, den Stromkreis schließt (wie bei einer älteren Haustürklingel).

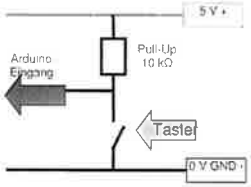
Damit keine elektrischen Störungen ein eindeutiges Signal verhindern können, wird zusätzlich ein sogenannter **Pull-Down-Widerstand** zwischen den Pin und die 5 V+ angeschlossen.



Wird der Taster **gedrückt**, so liegt eine Verbindung zum 5 V+ (**HIGH**) am Pin.

Wird der Taster **nicht gedrückt**, liegt über den Widerstand GND (**LOW**) am Pin an.

Selbstverständlich lässt sich durch Umdrehen der Reihe mit einem **Pull-Up-Widerstand** die Auswirkung umdrehen (invertieren). Dann gilt:



Wird der Taster **gedrückt**, liegt GND (**LOW**) am Pin an.

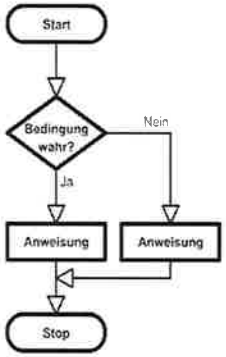
Wird er **nicht gedrückt**, so liegt über den Widerstand eine Verbindung zum 5 V+ (**HIGH**) am Pin.

Aufgabe 4

Du sollst nun das Lauflicht durch einen Bedarfstaster erweitern. Der soll dazu führen, dass das Lauflicht nur dann abläuft, wenn der Taster nicht gedrückt wird. Wird er gedrückt, sollen alle LEDs kurz aufleuchten und wieder ausgehen (blinken). Der Taster soll am Anfang der Schleife abgefragt werden.

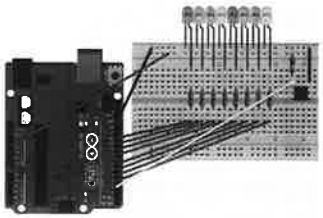
Erstelle dafür einen Programmablaufplan (PAP) – siehe Bild rechts. Die Erklärung für die Symbole findest du hier: http://de.wikipedia.org/wiki/Programmablaufplan

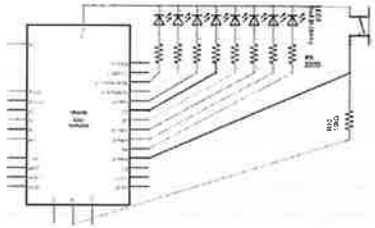
Du kannst so einen PAP mit der Hand, in einem Textverarbeitungsprogramm wie z. B. Word oder mit einem speziellen „Papdesigner" zeichnen (http://www.heise.de/download/papdesigner.html).



Aufgabe 5

Baue für die praktische Lösung von Aufgabe 4 einen weiteren Stromkreis mit einem Taster und einem 10-kΩ-Pull-Up-Widerstand in die Schaltung ein. Beachte, dass du die Plusschiene mit dem Pin 5 V und den digitalen Eingang des Arduino auf Pin 3 mit dem grünen Kabel verbindest (siehe Bilder).





Aufgabe 6

Programmiere diese Schaltung, indem du die Werte des Tasters abfragst. Dies kannst du mit dem Befehl **DigitalRead**, gefolgt von der Pin-Nummer machen. Der abgefragte Wert wird in eine digitale Variable gespeichert (z. B. in „Taster").

Danach kommt ein WENN-DANN-Befehl, der hier mit „**falls**" bezeichnet wird.

Falls also der Taster gleich HIGH ist, dann soll das Lauflicht wie gewohnt ablaufen, ansonsten sollen alle LEDs kurz an- und wieder ausgehen.

Tipp: Verwende einen logischen Operator, um den Tasterwert zu überprüfen:

Fragen und Probleme?

Bei ArduBlock kann man leider keine Fehler erkennen. Sollte das Programm nicht wie gewünscht auf dem Arduino ablaufen, muss man sich das Arduino-Programmierfenster anschauen, das einige Informationen bereithält (siehe Abbildung rechts).

Ganz unten werden in roter Schrift mögliche Fehler in englischer Sprache eingeblendet.
Sollte man mit diesen Meldungen nicht weiterkommen, lohnt sich eine kurze Webrecherche.

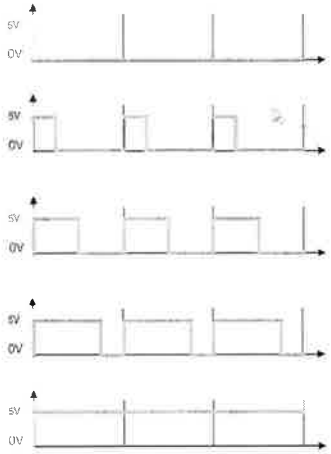
LED-Fading bzw. Pulsweitenmodulation (PWM)

Um LEDs zu dimmen, benutzt Arduino einen Trick. Normalerweise müsste man dafür nämlich die Spannung reduzieren. Arduino kann das nicht. Es ist nur in der Lage, Pins an- und auszuschalten. Das kann es dafür aber sehr schnell. Das Verhältnis der Zeit zwischen An und Aus entscheidet darüber, wie hell die LED leuchtet. Wenn sie lange an- und nur kurz ausgeschaltet wird, dann leuchtet sie heller, als wenn sie lange aus- und nur kurz angeschaltet wird.

Dieses Verfahren wird Pulsweitenmodulation (PWM) genannt und wird auch z. B. bei Motoren angewendet.

Aufgabe 7

Fülle die Lücken der Tabelle aus.



1. Licht ist **aus**. Impulsdauer **0 %**
   analogwrite(pin,0)

2. Licht ist _____. Impulsdauer _____ %

   analogwrite(pin,____)

3. Licht ist _____. Impulsdauer _____ %

   analogwrite(pin,____)

4. Licht **ist** _____. Impulsdauer _____ %

   analogwrite(pin,____)

5. Licht ist **sehr hell**. Impulsdauer **100 %**
   analogwrite(pin,255)

Arduino kann die Pulsweitenmodulation aber nicht an jedem Pin ausführen, sondern nur an denen, die mit einer Wellenlinie gekennzeichnet sind. Das sind die Pins 3, 5, 6, 9, 10 und 11.

Aufgabe 8

Schließe eine LED mit einem Vorwiderstand an PIN 9 an.

Lasse die Helligkeit über einen Analogwrite-Befehl von 0 auf 255 hochzählen und damit langsam heller werden. Teste und verändere dieses Programm.

Aufgabe 9

Lasse die Helligkeit über einen Analogwrite-Befehl von 255 auf 0 runterzählen – langsam dunkler werden. Variiere dieses Programm.

Tipp:

Mixe beide Aufgaben zusammen: Lasse die LED mal heller und dann wieder langsam dunkler werden.

Der Piezo-Summer

Der Piezo-Schallwandler ist ein kleiner Lautsprecher, der aus einer Metallscheibe und einer aufgeklebten Keramikscheibe, einem sogenannten Piezokristall, besteht. Dazu kommt dann noch eine kleine Platine. Die Keramikscheibe biegt sich etwas, wenn man eine elektrische Spannung anlegt bzw. ihn **HIGH** schaltet. Die Metallscheibe wirkt dann wie eine Membran, die Schallschwingungen in der Luft erzeugt. Ein kleiner Knacker wird hörbar. Da er das sehr schnell hintereinander machen kann, können mit ihm auch Töne erzeugt werden.

Aufgabe 10

Baue an Pin 9 einen Piezo-Summer an. Baue vor den Piezo-Summer einen Taster ein, damit der Summer nur dann lärmt, wenn man den Taster drückt.

Lasse den Summer sehr schnell hintereinander ein- und ausschalten.
(**Tipp:** Mache Pausen.)

Variiere das Programm über die Wartezeiten.

Aufgabe 11

Lasse nun eine kleine Melodie (z.B. Alle meine Entchen) spielen. Suche für andere Melodien nach Noten im Internet. Was ist z. B. g-g-g-dis-b-g-dis-b-g?

Du kannst dazu den Befehl „Ton" verwenden, bei dem du die Tonhöhe bzw. -frequenz und die Tondauer direkt angeben kannst.

Die wichtigsten Töne (und Halbtöne) mit ihren Frequenzen kannst du aus dieser Tabelle ablesen:

Ton	c	cis	d	dis	e	f	fis	g	gis	a	b/ais	h	c
Frequenz in Hz	523,3	554,4	587,3	622,3	659,3	698,5	740,0	784,0	830,6	880,0	932,3	987,8	1046,5

Die analoge Eingabe: Poti und LDR

Im Alltag gibt es nicht nur digitale Sensoren, die entweder HIGH oder LOW zurückgeben. Wichtiger sind analoge Sensoren, die eine Bandbreite von Werten liefern und somit unseren menschlichen Empfindungen nahe kommen (z. B. sehr heiß – heiß – warm – lauwarm – kalt – eiskalt). Um in diese analogen Eingaben einzusteigen, verwenden wir den Photowiderstand (LDR):

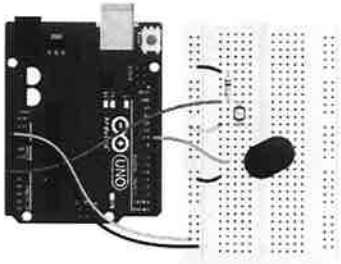
Dunkel: großer Widerstand

Hell: kleiner Widerstand

Dieser LDR ist ein Photowiderstand, der eine Bandbreite von Werten zurückgibt. Über den Befehl AnalogRead kann er Werte zwischen 0 und 1024 liefern.

Aufgabe 12

Baue folgende Schaltung auf.



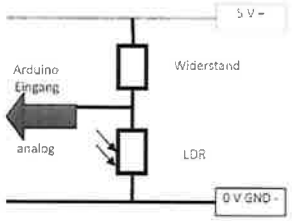
1. Auf der linken Seite bauen wir einen Spannungsteiler (Reihenschaltung) zwischen dem LDR und einem 100-kΩ-Widerstand auf.

Dessen Mitte verbinden wir mit dem grünen Kabel in ANALOG IN A0. Der Piezo wird mit Pin 9 angesteuert. Des Weiteren verwenden wir die Plus- und Minusschiene des Steckbretts.

2. Setze in eine analoge Variable mit analogRead auf Pin 0 den Wert des LDR.

Erzeuge dann einen Ton, der auf Pin 9 ausgegeben wird und die Tonhöhe (Frequenz) der Variablen LDR erhält.

3. Ergänze die Sätze.
Gegeben ist folgender Spannungsteiler:



Wenn es _____ ist, ist der LDR-Widerstand viel größer.
=> relativ große Spannung zwischen GND und Eingang.

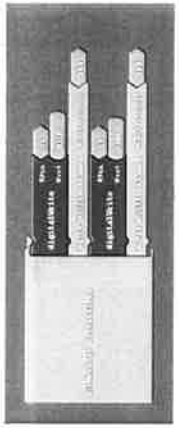
Wenn es _____ ist, ist der LDR-Widerstand viel kleiner.
=> relativ kleine Spannung zwischen GND und Eingang.

Lösungen

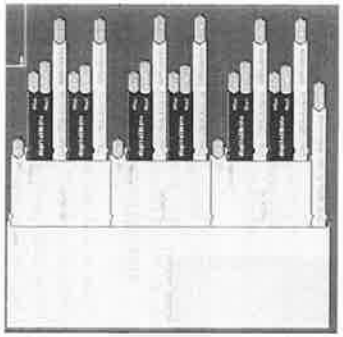
Hier findest du Lösungsvorschläge. Bitte nimm diese Hilfe erst nach reiflichen Überlegungen zur Hand.

Aufgabe 1
$R_V = U / I = (U_0 - U_{LED}) / I_{LED} = (5\,V - 2\,V) / 0{,}02\,A = 3\,V / 0{,}02\,A = 300\,V / 2\,A = 150\,V / A = 150\,\Omega$

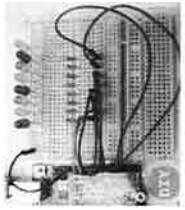
Aufgabe 2
Blink-LED:



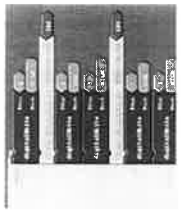
SOS-Programm:



Aufgabe 3
Lauflicht-Aufbau:



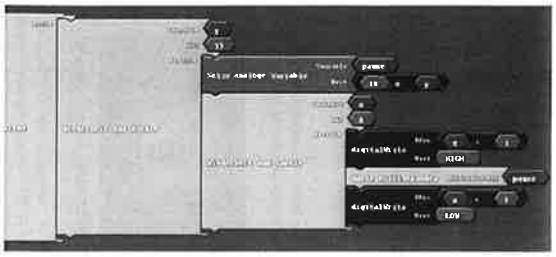
Ohne Variable:



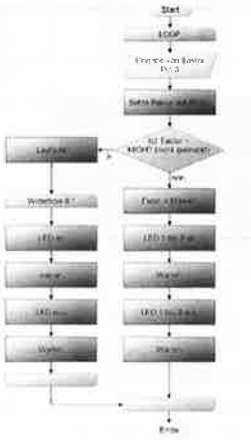
Mit mehreren Variablen:

Wiederhole und Zähl-Schleife:

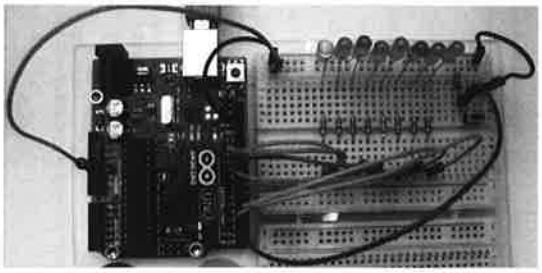
Mit mehreren Variablen:



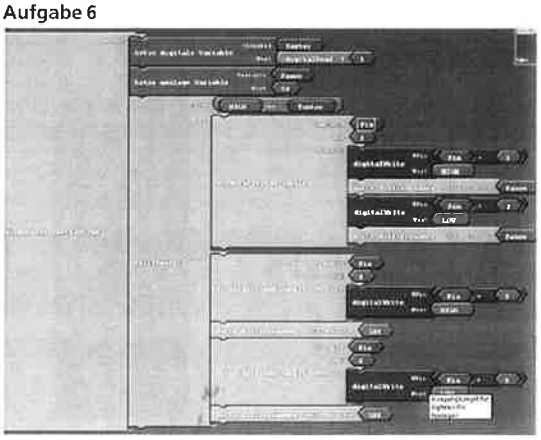
Aufgabe 4
mit PAPDesigner 2,208



Aufgabe 5



Aufgabe 6



Aufgabe 7

Fülle die Lücken der Tabelle aus:

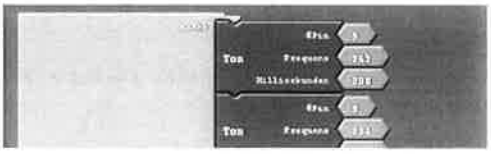
1. Licht ist aus.	Impulsdauer 0 %	analogwrite(pin,0)
2. Licht ist schwach.	Impulsdauer 25 %	analogwrite(pin,63)
3. Licht ist mittel.	Impulsdauer 50 %	analogwrite(pin,127)
4. Licht ist hell.	Impulsdauer 75 %	analogwrite(pin,191)
5. Licht ist sehr hell.	Impulsdauer 100 %	analogwrite(pin,255)

Aufgabe 8

Aufgabe 9

Aufgabe 10

Aufgabe 11
Intro zu Imperial March / Darth Vader / StarWars oder:



usw.

Aufgabe 12
2.

3. Wenn es **dunkel** ist, ist der LDR-Widerstand viel größer
=> relativ große Spannung zwischen GND und Eingang

Wenn es **hell** ist, ist der LDR-Widerstand viel kleiner
=> relativ kleine Spannung zwischen GND und Eingang

Serielle Schnittstelle

Wenn man wissen möchte, welche Werte der LDR zurück gibt, kann man sich diese über die serielle Schnittstelle anschauen. Dazu muss man aber im Programm auch eine Zeile ausdrucken lassen (serial println verbunden mit der Variable LDR):