#makers
DO IT.

Robert Jänisch
Jörn Donges

Mach was mit Arduino!

Einsteigen und durchstarten mit
Drum Machine, Roboterauto & Co.

HANSER

Jänisch/Donges

Mach was mit Arduino!

Robert Jänisch
Jörn Donges

Mach was mit Arduino!

Einsteigen und durchstarten mit Drum Machine, Roboterauto & Co.

HANSER

Die Autoren:

Robert Jänisch, Köln
Jörn Donges, Hamburg

Bibliografische Information der deutschen Nationalbibliothek:

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet unter http://dnb.d-nb.de abrufbar.

ISBN 978-3-446-45128-5
E-Book-ISBN 978-3-446-45258-9

Lektorat: Julia Stepp
Herstellung: Cornelia Rothenaicher
Umschlagrealisation: Stephan Rönigk
Titelillustration: © Ryan McGuire *(http://gratisography.com/#objects)*
Satz: Kösel Media GmbH, Krugzell
Druck und Bindung: CPI books GmbH, Ulm
Printed in Germany
www.hanser-fachbuch.de

Inhaltsverzeichnis

1 Einführung

Im Zuge der Initiative „Start Coding“ *(http://start-coding.de)* ließ Ranga Yogeshwar folgenden Satz fallen: *„Programmieren ist die Sprache des 21. Jahrhunderts.“* Was genau möchte er uns damit sagen? Er möchte zum Ausdruck bringen, dass die Fähigkeit des Programmierens in unserer vernetzten und digitalen Welt immer wichtiger wird. Wenn wir davon ausgehen, dass Softwareentwickelung eine immer größere Bedeutung gewinnt, dann ist die Verbindung von Software und Hardware die Königsdisziplin.

Um genau diese Verbindung von Software und Hardware wird es in diesem Buch gehen, denn Arduino ist eine aus Soft- und Hardware bestehende Physical-Computing-Plattform. Die hier entwickelten Projekte werden nicht auf deinem Computerbildschirm enden, sondern in der realen Welt erlebbar werden. Wir werden Objekte bauen, die mit deinem Umfeld interagieren können. Das können Gegenstände aus dem alltäglichen Leben, wie z. B. ein Wecker, sein. Das können aber auch solch futuristische Dinge wie ein humanoider Roboter oder eine vollautomatisierte Pflanzenbewässerungsanlage sein. Hast du Lust, zusammen mit uns zum Erfinder zu werden?

1.1 Maker – die Erfinder von morgen

Vielleicht fragst du dich, was dieses Buch mit Erfindungen zu tun hat. Alle großen Erfindungen fingen mit einer kleinen Idee und großer Leidenschaft an. Einige von ihnen haben die Welt grundlegend verändert. Wusstest du, dass viele große Erfindungen gar nicht im Labor eines Unternehmens, sondern durch Menschen wie dich und mich gemacht wurden? Gary Fisher hat zum Beispiel das Mountain Bike erfunden. Ottomar von Mayenburg hat 1907 auf einem Dachboden die Zahnpasta erfunden. Es gibt hunderte von Beispielen faszinierender Dachboden- und Garagen-Erfindungen. Der Zeitungsartikel „50 deutschen

Erfindungen, die die Welt veränderten“[1] gibt dir einen Vorgeschmack. Leider scheinen wir in Deutschland vergessen zu haben, wer der Hauptdarsteller einer Erfindung ist. Du bist es! Erfindungen sind kein Hexenwerk. Du benötigst aber einige Grundlagen, um als Erfinder durchstarten zu können.

Dieses Buch liefert dir das nötige Wissen und die erforderlichen Werkzeuge, um dir das Erfinden und Herstellen von Dingen zu ermöglichen. Das kann eine twitternde Topfpflanze oder ein intelligentes Türschloss sein. Die einzige Grenze ist deine Vorstellungskraft.

Nun fragst du dich sicherlich: Was braucht der Erfinder von morgen? Er benötigt:

- Software (z. B. eine Entwicklungsumgebung), Hardware (z. B. Mikrocontroller, Sensoren, Aktoren etc.) und Werkzeuge (Multimeter, Lötkolben usw.)
- Grundkenntnisse in Programmierung und Elektronik
- Spaß am Experimentieren
- Zugang zu neuester Technologie und neuestem Wissen

Neueste Technologie und frisches Wissen sind das Alphabet des 21. Jahrhunderts. Wenn du neue Technologien beherrschst, konsumierst du Produkte und Services nicht einfach nur, sondern gestaltest deine Welt selbstbestimmt. Du kannst zum Beispiel

- technische Gegenstände reparieren oder verbessern,
- interaktive Dinge für das Internet der Dinge (IoT) bauen,
- kleine Roboter oder Drohnen bauen,
- oder Maschinen bauen, die Maschinen bauen (3D-Drucker, CNC-Fräsen).

Wäre es nicht aufregend, solche Dinge zu bauen und mit der physikalischen Welt interagieren zu lassen? Stell dir vor, was du mit deinen Ideen alles verändern könntest.

Mit diesem Buch möchten wir dich auf deinem Weg zum Erfinder von morgen begleiten. Du wirst erfahren, wie du durch den Einsatz neuester Technologien und deiner Innovationskraft die Welt verändern kannst.

Auf Basis spannender Arduino-Projekte wirst du neue Technologien entdecken, kennenlernen und ausprobieren. Mit dem erworbenen Wissen wirst du in der Lage sein, deine eigenen Projekte zu verwirklichen und diese mit der Maker-Community zu teilen.

„Was ist denn ein Maker?“ fragst du dich nun vielleicht. Maker ist eigentlich nur ein modernes Wort für Erfinder. Früher war es deutlich schwerer, auf gleichgesinnte Erfinder zu treffen. Beflügelt durch das Internet, Maker Faires (Messen) und FabLabs (offene Werkstätten) ist in der Zwischenzeit jedoch eine sehr lebhafte Maker-Community entstanden.

Im Zuge der Digitalisierung wird auch Hobbybastlern der Zugang zu High Tech ermöglicht. Der Maker von heute kann programmieren, löten und kennt sich mit Hardware-Plattformen wie dem Arduino aus. Dennoch ist der Weg zum „smarten“ Maker ein langer und nur wenige schaffen es, diesen erfolgreich zu meistern.

1 *https://www.welt.de/wirtschaft/karriere/leadership/gallery12202607/50-Erfindungen-die-die-Welt-veraenderten.html*

Die Maker-Community ist eine bunt gemischte Truppe. Manche bringen viele Vorkenntnisse mit, andere gar keine. Eines haben sie aber alle gemeinsam: Sie haben eine Idee und wollen diese umsetzen. Diese überproportionale Begeisterung für eine Sache macht den reinen Konsumenten zum Maker, der seine eigenen Produkte und Services bauen möchte. Um den Schritt von der „Zero“ zum „Hero“ zu meistern, müssen zwei wichtige Kriterien erfüllt sein:

- Der Maker muss breiten Zugang zu modernen Werkzeugen und Technologien haben.
- Der Maker muss eine große Auswahl an Möglichkeiten haben, sich frisches Wissen anzueignen.

Beide Kriterien sind in Deutschland noch nicht zur vollsten Zufriedenheit erfüllt, doch die stetig wachsende Maker-Community mit ihren FabLabs und Maker Spaces ist auf einem guten Weg.

Der Wunsch, Dinge zu verbessern, neue Technologien zu nutzen und Wissen mit anderen zu teilen, macht den Maker zum „smarten“ Maker. Um ein „smarter“ Maker zu werden, musst du dich mit anderen Makern austauschen und vernetzen, das heißt, du musst Zugang zu deren Expertise bekommen.

In den FabLabs und digitalen Communities entstehen neue Erfindungen und Innovationen. Wissen wird ausgetauscht und konzentriert sich. Einige dieser Erfindungen finden auch außerhalb der Maker-Community starken Zuspruch. Durch erfolgreiche Crowdfunding-Kampagnen wird ein Zugang zum Markt geschaffen. So kann das Fundament zu einem wirtschaftlichen Unternehmen gelegt werden. Auch wenn nur die Erfindungen einiger mutiger und smarter Maker es auf dieses Level schaffen, so kann die Auswirkung dennoch gigantisch sein. Der italienische Physiker und Elektroningenieur Guglielmo Marconi[2] wurde auf diesem Wege zum Pionier der drahtlosen Telekommunikation.

Wir hoffen, dass dieses Buch dich dabei unterstützen wird, ein „smarter“ Maker zu werden.

1.2 Was dich in diesem Buch erwartet

Auf Basis des Arduino werden wir in diesem Buch die spannende Welt der Mikrocontroller erkunden. Du benötigst dafür keine besonderen Vorkenntnisse. Es reicht, wenn du Interesse an der Mikrocontrollertechnik mitbringst, und Spaß am Aufbau von Schaltungen sowie der Entwicklung von Programmen hast. Aber sonst wärst du ja wahrscheinlich nicht hier, oder?

2 Hier erfährt du mehr über Guglielmo Marconi: *https://www.youtube.com/watch?v=v8qXtu0-__k*

In elf spannenden Projekten lernst du alles, was du über Programmierung und Hardware wissen musst, um deine eigenen Ideen mit dem vielseitigen Mikrocontroller-Board zu verwirklichen. Wir haben Wert darauf gelegt, dass du keine graue Theorie pauken musst, sondern gleich mit dem Aufbau von nützlichen Geräten loslegen und wertvolle Praxiserfahrung erwerben kannst.

Nachdem du die Beispiele in diesem Buch aufgebaut und nachprogrammiert hast, wirst du genug über die Arduino-Plattform wissen, um deine eigenen Schaltungsideen planen und umsetzen zu können. Mit diesem Buch im Regal wirst du über den Werkzeugkasten verfügen, den du benötigst, um deiner Kreativität freien Lauf zu lassen. Sicherlich werden dir schon beim Nachbauen der Projekte Ideen kommen, wie man diese erweitern und verändern kann. Fühle dich stets frei, dies zu tun! Die Arduino-Plattform ist genau dafür gemacht. Sie soll Raum zum Experimentieren und Ausprobieren geben, und dich in die Lage versetzen, ohne theoretischen Ballast zur Umsetzung zu gelangen. Im Fachjargon sagt man dazu „Rapid Prototyping“. Das heißt so viel wie „eine Idee schnell mal ausprobieren“. Es wird dann rasch klar, ob die Idee funktioniert, und an welchen Stellen sie noch erweitert oder verbessert werden muss. So kannst du deine Projekte Schritt für Schritt immer weiter optimieren. Und wer weiß, vielleicht treffen wir uns dann irgendwann auf einer Erfindermesse wieder?

1.3 Wie dieses Buch aufgebaut ist

Wenn du noch keine Erfahrung mit dem Arduino hast, dann solltest du die Kapitel nacheinander durcharbeiten, denn sie bauen aufeinander auf. Neue Dinge werden wir immer dann erklären, wenn sie zum ersten Mal auftreten. Daher nimmt der Schwierigkeitsgrad der Projekte im Verlauf des Buches zu.

Solltest du ein Theoriefan sein, kannst du auch zu Beginn auch erst einmal zu Kapitel 17 und 18 springen. Dort findest du die wichtigsten Programmier- und Hardwarekenntnisse, die in die diesem Buch vermittelt werden, in der Zusammenfassung.

Bevor wir die ersten Arduino-Projekte realisieren, erhältst du in den Kapiteln 2 bis 4 erst einmal die wichtigsten Grundlagen für deine Arbeit mit dem Arduino. Dazu zählen u. a. die Installation der Entwicklungsumgebung sowie das Lesen von elektronischen Schaltungen. Auch hier werden wir bereits mit zahlreichen Beispielen arbeiten.

Anschließend stürzen wir uns direkt in die Praxis. In Kapitel 5 bauen wir eine Arduino-gesteuerte Ampelschaltung. Du wirst überrascht sein, wie schnell das geht. Danach bauen

wir einige nützliche Geräte, die dich im Alltag unterstützen. In Kapitel 6 lernst du, wie man eine Weltzeituhr mit Alarmfunktion baut. In Kapitel 7 entwickeln wir eine Mini-Wetterstation mit Analoganzeige. In Kapitel 8 realisieren wir eine temperaturgeregelte Lüftersteuerung.

Wenn Du dich nun fragst, wie du eine ganz individuelle Smart Home-Lösung entwickeln kannst, sind die nächsten beiden Kapitel etwas für dich. In Kapitel 9 erklären wir dir, wie du mit dem Particle Photon, einer Arduino-ähnlichen Hardware-Plattform, mit dem Internet vernetzte Schaltungen entwickeln kannst. Diese nutzen wir in Kapitel 10, um eine automatische Pflanzenbewässerungsanlage zu bauen.

Ein weiteres faszinierendes Gebiet ist die Tonerzeugung mit dem Arduino. In den Kapiteln 11 bis 14 unternehmen wir einen Ausflug in die Welt der Synthesizer und gesampelten Sounds. Dabei lernst du wichtige Hardware-Konzepte wie die Digital-Analog-Wandlung kennen. Im letzten und anspruchsvollsten Sound-Projekt werden wir eine echte Drum Machine bauen (Kapitel 14).

In Kapitel 15 und 16 lernst du die aufregende Welt der Robotik kennen. Wir bauen ein autonom fahrendes Roboterauto und einen humanoiden Roboter.

Wie bereits erwähnt, erhältst du in Kapitel 17 und 18 noch einmal eine Zusammenfassung der wichtigsten Programmier- und Hardwarekenntnisse, die in die diesem Buch vermittelt werden. Diese Kapitel kannst du jederzeit nutzen, um eine bestimmte Sache nachzuschlagen oder dein Wissen aufzufrischen.

Das Buch schließt mit einem Ausblick auf andere Mikrocontroller-Boards sowie spannende Projektideen, die du darüber hinaus noch realisieren kannst.

Die Sketches sämtlicher Projekte aus diesem Buch findest du unter *www.desk-factory.de/mach-was-mit-arduino.* Dort berichten wir auch über neue Projekte und bieten dir Platz für Fragen und Austausch.

So, und nun wünschen wir dir viel Vergnügen bei der Lektüre dieses Buches! Wir hoffen, dass du genauso viel Spaß beim Umsetzen der Projekte hast wie wir.

Köln/Hamburg, März 2017

Robert Jänisch
Jörn Donges

2 Willkommen in der Arduino-Welt!

Wenn man vom Arduino spricht, ist meistens die Hardware, also die Platine mit dem Mikrocontroller, gemeint. Zusätzlich zur Hardware wird auch eine Softwareumgebung ausgeliefert, mit der du Programme für den Arduino schreiben kannst. Bevor wir uns in den folgenden Kapiteln in die Praxis stürzen, erhältst du an dieser Stelle erst einmal einen kompakten Schnelleinstieg in die Arduino-Welt.

Wir vermitteln dir die wichtigsten Grundlagen zur Arduino-Hardware und zeigen dir, wie du die dazugehörige Software auf deinem Rechner installierst. Außerdem lernst du, wie du in der Software, auch Entwicklungsumgebung genannt, verschiedene Einstellungen vornehmen kannst, um deinen Arduino zu konfigurieren. Sind alle Einstellungen korrekt gewählt, wird es ein Leichtes für dich sein, ein Programm an die Arduino-Hardware zu übertragen. Am Ende des Kapitels wirst du in der Lage sein, Programme auf deinem Arduino auszuführen.

2.1 Was ist überhaupt der Arduino?

Bei der Arduino-Plattform handelt es sich um eine komplette Experimentier- und Entwicklungsumgebung für die beliebten und weit verbreiteten Mikrocontroller der Firma Atmel. 2016 wurde die Firma Atmel von der Firma Microchip gekauft. Der Mikrocontroller-Chip ist zwar das Hirn und die Schaltzentrale eines jeden Projekts, aber um ihn zum Leben zu erwecken, braucht es noch ein wenig mehr. Der Chip muss mit Strom versorgt werden, und es muss eine Möglichkeit geben, Programme und Daten darauf zu schreiben. Natürlich benötigt man auch noch weitere Ein- und Ausgänge, um zum Beispiel Sensoren auszulesen oder einen Motor bzw. ein Display anzusteuern.

All diese Grundfunktionen stellt die Arduino-Plattform bereit. Sie bettet den Atmel-Chip in eine Umgebung ein, in der alles Nötige schon vorhanden ist, sodass du sofort mit eigenen

Ideen und Projekten loslegen kannst. Dazu gehört auch eine komplette Entwicklungsumgebung, mit der du am PC deine Programme entwickeln und dann auf das Board übertragen kannst.

Es werden verschiedene Boards von Arduino angeboten, die für ganz unterschiedliche Anwendungen geeignet sind. Zum Beispiel ist der **Arduino Mega** mit seinen 54 digitalen Ein- und Ausgabeports ein wahrer Kommunikationskünstler, während der winzige **Ardu-Pilot** dazu gedacht ist, seinen Dienst in kleinen Modellfliegern zu verrichten. In diesem Buch werden wir das Standardmodell, den **Arduino Uno**, benutzen. Der Arduino Uno eignet sich am besten zum Einstieg und ist auch der am weitesten verbreitete Controller der Arduino-Familie. Du bekommst ihn für circa 25 Euro bei den einschlägigen Elektronikhändlern, wie etwa Conrad, Reichelt oder Flikto.

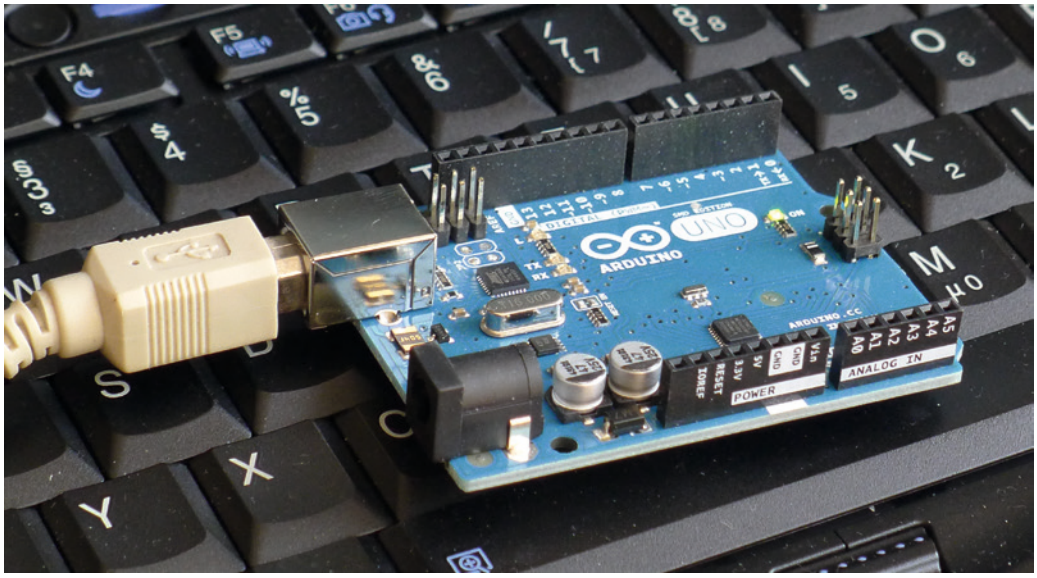


Bild 2.1 Das Arduino Uno-Board

2.2 Los geht's! Installation der Arduino-Software

Im ersten Schritt werden wir die Arduino-Software herunterladen und installieren. Sie sorgt dafür, dass du deine Programme (oder Sketches, wie man beim Arduino sagt) auf dem PC oder Laptop entwickeln und dann übertragen kannst. Danach kannst du den Arduino schon für einen ersten Test in Betrieb nehmen. Hierfür ist noch kein weiterer Schaltungsaufbau nötig.

Für die folgenden Schritte brauchst du:

- ein Arduino Uno- Board
- ein USB-Anschlusskabel
- das Arduino-Softwarepaket

2.2.1 Die Arduino-Software downloaden

Die Arduino-Entwicklungsumgebung und die dazugehörigen Treiber kannst du unter *http://www.arduino.cc* herunterladen. Unter *Downloads* findest du ein zip-Paket, in dem alles, was du brauchst, bereits zusammengestellt ist. Entpacke einfach die zip-Datei an eine beliebige Stelle auf deiner Festplatte. Die Entwicklungsumgebung muss nicht installiert werden, sondern wird einfach durch einen Doppelklick auf die Datei *arduino.exe* gestartet. Dies hat den Vorteil, dass du die Umgebung mit deinen Projektdaten auch mobil von einem USB-Stick aus einsetzen kannst. Du wirst deshalb also keine Probleme haben, dich mit anderen Makern auszutauschen und zusammen an Projekten zu arbeiten.

Die Datei *arduino.exe* startet die integrierte Entwicklungsumgebung (IDE). Hier werden die Programme erstellt, verwaltet und auf den Controller übertragen. Die IDE verwendet als Programmiersprache eine speziell angepasste Version der Sprache C, in die bereits umfangreiche Bibliotheken zur Ansteuerung des Controllers eingebunden sind. Das erleichtert den Einstieg in die Programmierung, da man sich erst einmal nicht näher mit den hardwarenahen Details beschäftigen muss. Der Atmel-Controller selbst versteht natürlich nur Maschinensprache, deshalb müssen die Programme zuerst von einem Compiler übersetzt werden.

2.2.2 Den USB-Treiber unter Windows installieren

Bevor du deinen ersten Sketch übertragen kannst, musst du jedoch erst einmal den USB-Treiber installieren. Wir beschreiben im Folgenden den Vorgang für Windows-Nutzer. Wenn du Linux einsetzt, findest du entsprechende Anleitungen in der *Readme*-Datei des Download-Pakets. Aktuelle Informationen zum Betrieb des Arduino unter den verschiedenen Linux-Distributionen stehen auch unter *http://playground.arduino.cc/Learning/Linux*.

1. Verbinde den Arduino Uno über das USB-Kabel mit deinem Rechner und warte, bis Windows ein *Unbekanntes Gerät* erkennt (Bild 2.2).
2. Öffne die *Systemsteuerung* im Startmenü und wähle dort den Bereich *Geräte und Drucker* aus.
3. Öffne mit einem Rechtsklick auf das Icon *Unbekanntes Gerät* den Dialog *Eigenschaften*.
4. Wähle den Reiter *Hardware* aus und klicke dort auf den Button *Eigenschaften*.

5. Es öffnet sich ein weiterer Dialog. Wähle hier *Einstellungen ändern* aus.
6. Klicke nun auf *Treiber aktualisieren*.
7. Im nächsten Dialog wählst du *Auf dem Computer nach Treibersoftware suchen* aus.
8. Klicke auf *Weiter* und dann auf *Durchsuchen*.
9. Suche nun den Ort aus, an dem du das Arduino-Softwarepaket entpackt hast, und navigiere zum Unterordner *driver*.
10. Klicke auf *Weiter*. Das System findet und installiert jetzt die richtigen Treiber.

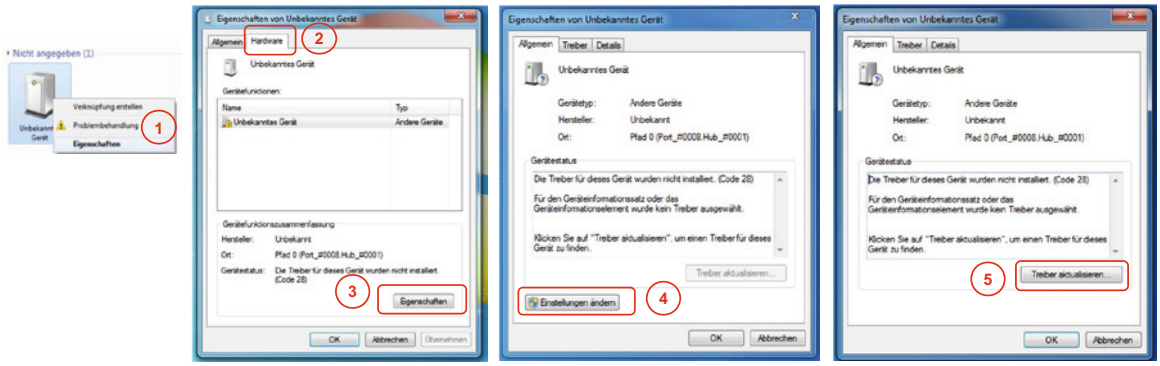


Bild 2.2 Installation des Arduino-USB-Treibers unter Windows 7

2.3 Die Entwicklungsumgebung starten und den ersten Sketch übertragen

Bevor du den ersten Sketch auf den Arduino übertragen kannst, musst du noch einstellen, welches Board und welchen UBS-Anschluss du nutzt. Dies machst du wie folgt: Klicke auf *Werkzeuge* und wähle dort unter *Board* dein Arduino-Board aus. Als Nächstes suchst du unter WERKZEUGE > PORT den UBS-Port aus, an dem dein Arduino angeschlossen ist. Hier sollte eigentlich nur ein Port angezeigt werden.

Jetzt kannst du den ersten Sketch auf den Arduino übertragen. Starte dazu die integrierte Entwicklungsumgebung mit einem Doppelklick auf *arduino.exe*. Wähle DATEI > BEISPIELE > 01 BASICS > BLINK aus. Nun öffnet sich ein Fenster mit einem mitgelieferten Beispiel-Sketch.

Dieses kleine Code-Fragment bringt die auf dem Board verbaute gelbe LED zum Blinken. Um den Sketch zu übertragen, musst du in der oberen Toolbar auf den Button mit dem nach rechts zeigenden Pfeil klicken (in Bild 2.4 rot markiert).

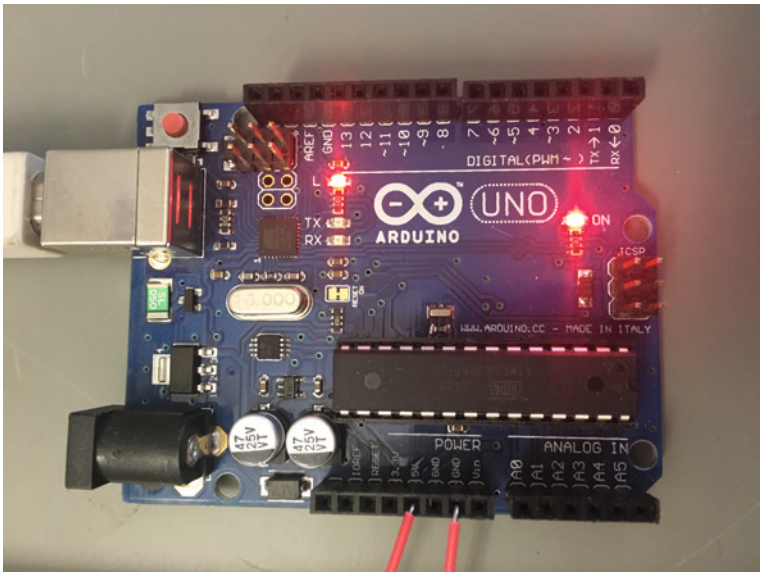


Bild 2.3 Sobald der Arduino am Strom angeschlossen ist, leuchtet er.

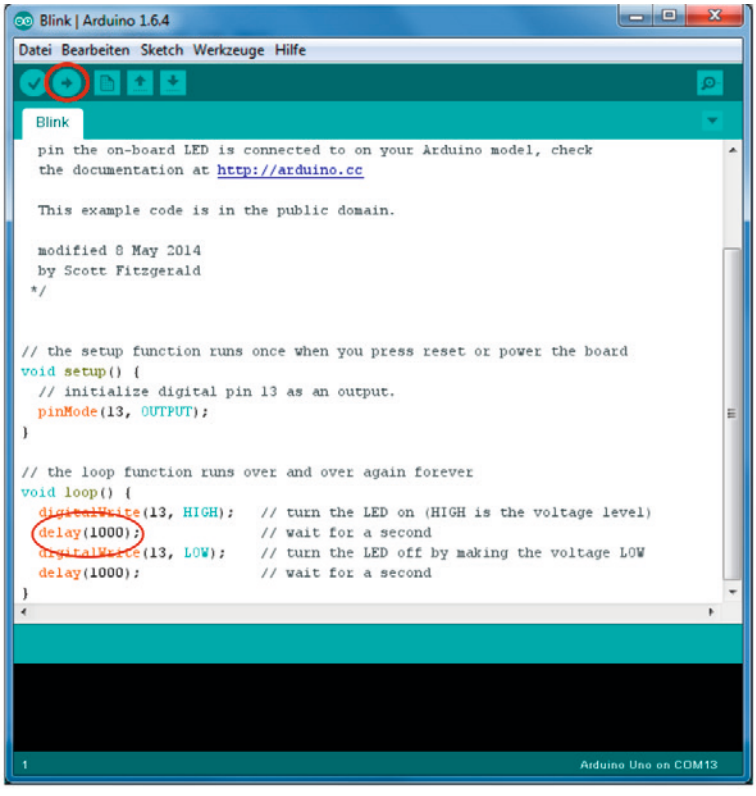


Bild 2.4 Den ersten Sketch übertragen

Wenn alles klappt, flackern die LEDs mit den Bezeichnungen *RX* und *TX* während der Übertragung auf dem Board kurz auf. Sie zeigen an, dass gerade Daten mit dem Computer ausgetauscht werden. Bei *TX* werden Daten gesendet *(Transmit)* und bei *RX* werden sie empfangen (*Receive*). Dann fängt die LED *L* langsam an zu blinken. Aber ist wirklich unser kleines Beispielprogramm für das Blinken verantwortlich? Wie könnten wir das testen? Ganz klar: Wir verändern einfach das Programm an einer Stelle und prüfen, ob und wie sich das auswirkt.

Für diesen Test musst du nur wissen, dass dort, wo im Code die Funktion `delay(milliseconds);` auftaucht, der Controller die angegebene Zeit wartet. In Bild 2.4 ist die entsprechende Code-Stelle rot markiert. Das erste Delay gibt die Leuchtzeit nach dem Einschalten der LED an, das zweite gibt die Wartezeit nach dem Abschalten der LED an. Die Zeiten werden in Millisekunden eingegeben. Der Wert `1000` steht also für eine Sekunde.

Ändere nun die Zahlen und übertrage den Sketch erneut. Du siehst sofort das Ergebnis: Die LED blinkt im von dir gewählten Rhythmus.

An diesem Code-Beispiel kannst du auch den grundsätzlichen Aufbau eines Sketches erkennen. Es gibt zwei Hauptfunktionen, die immer wieder auftauchen werden:

```
1  void setup() { // wird einmal zu Beginn ausgeführt
2  }
3  void loop() { // wird endlos (also immer wieder) ausgeführt
4  }
```

Das Schlüsselwort `void` sagt dem C-Compiler, dass jetzt eine Funktion definiert wird, die keinen Wert zurückliefert. So etwas nennt man auch eine Prozedur, mit der eine bestimmte Tätigkeit ausgeführt wird.

Die Funktionen `setup` und `loop` bestimmen die grundsätzliche Struktur eines Sketches. Die Funktion `setup` wird zu Beginn genau einmal durchlaufen. Hier legst du deine Rahmenbedingungen fest und initialisiert das System. Alles, was zur Vorbereitung der Aufgabe ausgeführt werden muss, gehört hier hinein. Die Funktion `loop` wird immer wiederholt und enthält die eigentlichen Funktionen deiner Anwendung. Auf diese Weise vermeidest du, dass der Arduino in einen undefinierten Zustand gerät. Denn wenn der `loop`-Block zuende ist, wird er gleich wieder von vorne ausgeführt. Daher muss in unserem Blink-Sketch die LED auch nur einmal ein- und ausgeschaltet werden. Die Befehle stehen ja im `loop`-Block, also wird der Arduino endlos blinken.

In den Arduino-Projekten (Kapitel 5 bis 16) wirst du Schritt für Schritt mehr zur Programmierung erfahren. Solltest du dir einen strukturierten Überblick verschaffen wollen, findest du in Kapitel 17 alle Programmiergrundlagen in der Zusammenfassung.

Nachdem du den Arduino nun zum Laufen gebracht hast, kannst du es sicher kaum erwarten, deine erste Schaltung aufzubauen. Damit werden wir uns in Kapitel 3 beschäftigen.

3 Dein erster Schaltkreis

Alle elektrischen Geräte beruhen auf denselben physikalischen Gesetzmäßigkeiten. In einer Schaltung werden diese Gesetzmäßigkeiten gezielt genutzt, um Dinge physikalisch zu verändern. Der einfachste Stromkreis beinhaltet eine Energiequelle und einen Verbraucher. Dies kann zum Beispiel ein mit einer Batterie betriebenes Leuchtmittel wie bei einer Taschenlampe sein. Eine ähnliche Schaltung wirst du in diesem Kapitel kennenlernen und selber bauen.

Natürlich kann ein Stromkreis auch sehr viel komplexer aufgebaut sein als der einer Taschenlampe. Je nach Anwendung werden in einem Stromkreis verschiedene Komponenten hinzugefügt. Werden bestimmte Schaltungen mehrfach in gleicher Ausprägung benötigt, kann es Sinn ergeben, diese in sogenannte integrierte Schaltungen (Englisch: *integrated circuits*, kurz: IC) zu überführen und als eine Art Baustein verfügbar zu machen. Smartphones bestehen zum Beispiel aus einer Vielzahl integrierter Schaltungen in einem Chip.

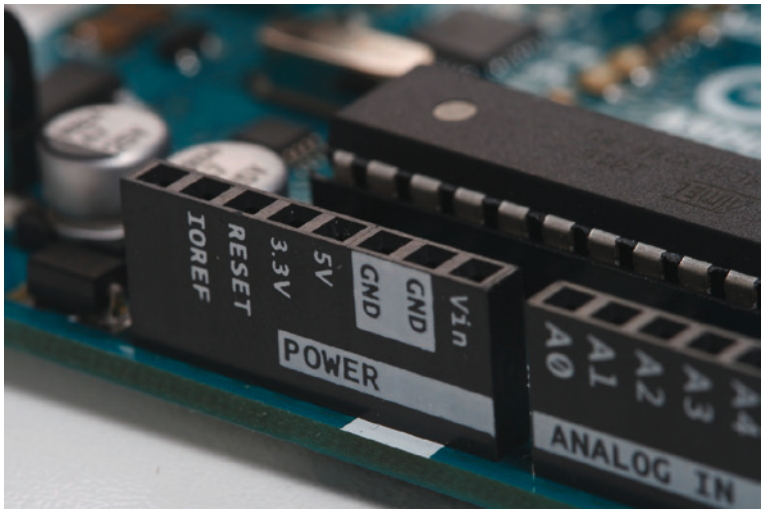


Bild 3.1 Anschlüsse für individuelle Schaltungen beim Arduino Uno

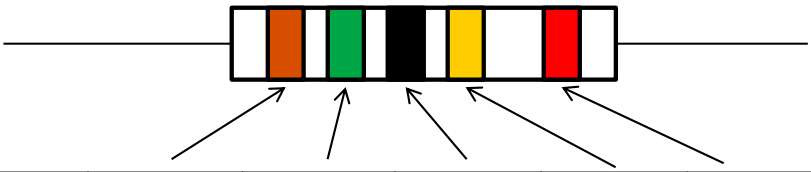
Hast du schon mal vom Mooreschen Gesetz gehört? Das Mooresche Gesetz geht auf Gordon Moore, einen Mitbegründer der Firma Intel, zurück. Dieser prognostizierte in den sechziger Jahren, dass sich die Zahl der Transistoren von integrierten Schaltungen (IC) jährlich verdoppeln würde. Diese Annahme traf er aufgrund der rasanten Entwicklung der Halbleiterindustrie. Später relativierte er die Annahme dahingehend, dass er voraussagte, die aktiven Komponenten eines Chips würden sich etwa alle zwei Jahre verdoppeln. Heute gehen Experten von einer Verdoppelung innerhalb von 18 Monaten aus. Dieser Trend ist ungebrochen und wir bekommen immer kleinere Hardware mit immer mehr Funktionen angeboten.

Doch bevor wir weiter über die Zukunft von technischen Geräten philosophieren, kehren wir zu den Basics zurück und bauen eine erste Schaltung.

Was du dafür brauchst:

- je eine rote, gelbe und grüne Leuchtdiode
- 3 × 220 Ohm-Widerstände
- einen Taster (z. B. Drucktaster T604[1])
- eine Schaltlitze oder einen Klingeldraht (ca. 1 m)
- ein Steckbrett (Breadboard)

Falls du nicht weißt, welchen Widerstand du in der Hand hältst, hilft dir die Übersicht in Bild 3.2 bei der Orientierung.



Farbe	1. Ring 1. Wertziffer	2. Ring 2. Wertziffer	3. Ring 3. Wertziffer	4. Ring Multiplikator	5. Ring Toleranz
farblos	-	-	-	-	-
silber	-	-	-	$x10^{-2}$	-
gold	-	-	-	$x10^{-1}$	-
schwarz	-	0	0	$x10^{0}$	-
braun	1	1	1	$x10^{1}$	±1 %
rot	2	2	2	$x10^{2}$	±2 %
orange	3	3	3	$x10^{3}$	-
gelb	4	4	4	$x10^{4}$	-
grün	5	5	5	$x10^{5}$	±0,5 %
blau	6	6	6	$x10^{6}$	-
violett	7	7	7	$x10^{7}$	-
grau	8	8	8	$x10^{8}$	-
weiß	9	9	9	$x10^{9}$	-

Bild 3.2 Diese Tabelle erklärt dir, wie du den richtigen Widerstand findest.

1 *https://www.conrad.de/de/drucktaster-24-vdc-005-a-1-x-aus-ein-t604-tastend-1-st-700479.html*