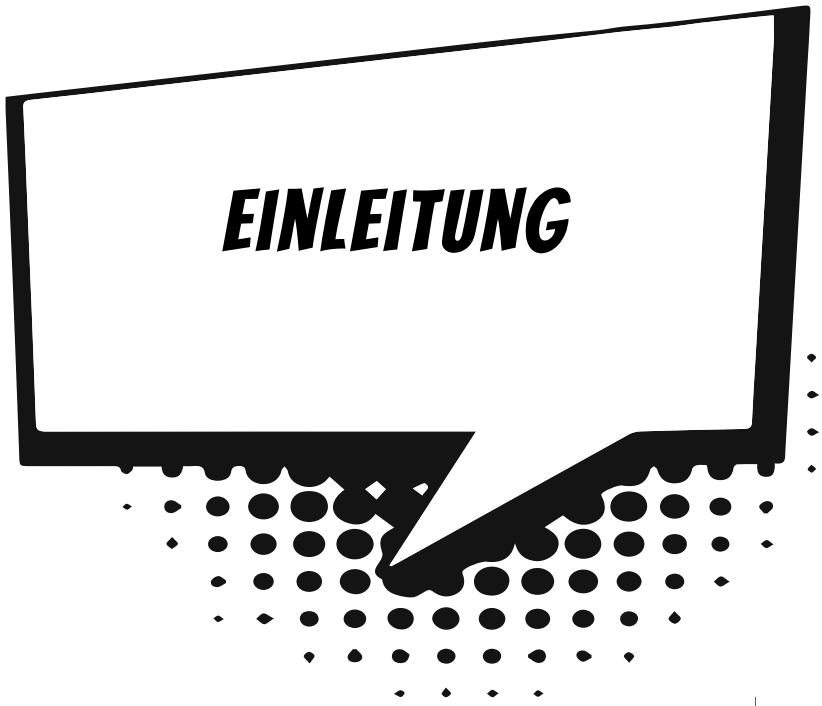
Arduino für Kids

Einfacher Einstieg in die Welt der Mikrocontroller-Programmierung



Du wolltest schon immer programmieren oder mit den Bestandteilen eines Computers arbeiten? Du wirst zwar nach diesem Buch keinen Computer bauen können, aber die Einleitung erklärt einiges, was du später machen kannst:

Dies erfährst du in der Einleitung:

- Was ist ein Mikrochip, ein Mikrocontroller und was ist ein Arduino?
- Wie programmiert man?
- Welche Materialien brauchst du für dieses Buch?

Du siehst, du wirst einige Grundlagen über die Elektronik kennenlernen!

Was ist ein Mikrochip?

Sicherlich hast du schon eine Vorstellung, wie ein Mikrochip aussieht. Wie eine kleine schwarze Fläche, die auf einer Platine, zum Beispiel dem Mainboard deines Computers, angebracht ist. Der Mikrochip, den du programmieren wirst, sieht etwas anders aus als ein solcher Chip. Er ist

rechteckig, nicht quadratisch und besitzt wesentlich weniger Pins (die Metallbeine, die an der Seite herausragen). Außerdem sind diese Pins wesentlich größer, als du sie von »normalen« Mikrochips kennen könntest.

Und was ist jetzt ein Mikrocontroller?

Ein Mikrocontroller ist ein Mikrochip, der alle benötigten Komponenten (sozusagen das Zubehör) bereits in sich trägt. Er sieht genauso aus wie ein Mikrochip. Du kannst ihn erneut mit dem Mainboard eines Computers vergleichen. Dann wäre das Mainboard der Mikrocontroller, und der Arbeitsspeicher wäre dann ein Teil des Zubehörs. Der Arbeitsspeicher ist beispielsweise ebenfalls in den Mikrocontroller integriert.

Merke dir: Wenn in diesem Buch in Zukunft über einen Mikrochip geschrieben wird, ist eigentlich der Mikrocontroller gemeint! Und zur Rechtschreibung: Man kann sowohl Microcontroller mit c als auch mit k (Mikrocontroller) schreiben.

Und was ist dann ein Arduino?

Damit du den Mikrocontroller am Anfang leichter programmieren kannst, gibt es das sogenannte Arduino-Projekt. Beim Arduino-Projekt gibt es fertige Platinen mit dem Mikrocontroller und eigener Software zum Programmieren. Die Platine wird dabei einfach als Arduino bezeichnet. In diesem Buch geht es, wie der Titel schon verrät, um die Programmierung eines Mikrochips auf der Arduino-Platine.

Das Besondere ist dabei, dass das Arduino-Projekt eine fertige Platine mit der passenden Entwicklungsumgebung für einen PC bereitstellt. Frühere Entwickler mussten sich teilweise sogar die Boards selbst basteln, damit sie überhaupt lernen konnten, wie man programmiert.

Wie programmiert man?

Programmieren kann man leider nicht durch reden, sondern nur über Texte am Computer. Diese Texte werden zudem nicht in Deutsch verfasst, sondern die wenigen Wörter sind auch noch auf Englisch. Das sollte dich aber nicht abhalten, programmieren zu lernen! Du lernst programmieren mit der Programmiersprache C++, einer sehr systemnahen und systematischen, aber auch komplexen Programmiersprache. Um es am Anfang

nicht viel zu schwer zu machen, haben die Arduino-Macher einen einfacheren Dialekt (= eine Variation der Programmiersprache) entwickelt. C++ basiert auf wenigen Wörtern, dafür auf vielen teilweise sehr kryptisch wirkenden Zeichen. In den Klammern steht jeweils, wie man dieses Zeichen »aussprechen« würde: zum Beispiel ++ (Inkrement), % (Modulo-Division).

Wenn du erst einmal weißt, was die einzelnen Symbole bedeuten, wirst du sie fehlerfrei anwenden können. Mit folgendem Quelltext zeige ich dir, was mit C++ möglich ist. Diesen Quellcode habe ich für einen selbst gebauten Terrariencomputer entwickelt, der leider noch im Anfangsstadium ist:

```
#include "dimmen.h"
#include "kern_temperatur.h"
#include "terra_temperatur_class.h"
#include "class_cool.h"

void setup() {
    pinMode(13, OUTPUT);  pinMode(12,OUTPUT);
    Serial.begin(9600);  ADMUX = 0xC8;  delay(10);
}

void loop() {
    Cooler cooler(13); delay(100); bool hot = kern_temp(17);
    if (hot)  {
        cooler.start();
    } else {
        cooler.stop();
    }
    delay(500);
}
```

Zum Programmieren brauchst du allerdings noch Software für den PC, die du unter *arduino.cc* im Internet findest. Zur genauen Installation sieh bitte in den Anhang oder in das nächste Kapitel.

Nun sollst du aber etwas über das Gerät lernen, das wir für das Buch benötigen: den Arduino. Der Arduino, den du vermutlich vor dir hast, ist ein Arduino Uno. Du solltest den Arduino erst einmal über USB mit dem Computer verbinden. Du kannst ihn zwar noch nicht programmieren, aber du kannst ihn schon als Stromquelle für die nächsten Schaltungen benutzen.

Der Inhalt dieses Buches basiert auf dem Arduino Uno R3. Einige Kapitel sind aber auch für andere Arduino-Versionen geeignet, was du den Randhinweisen zu Beginn jedes Kapitels entnehmen kannst.

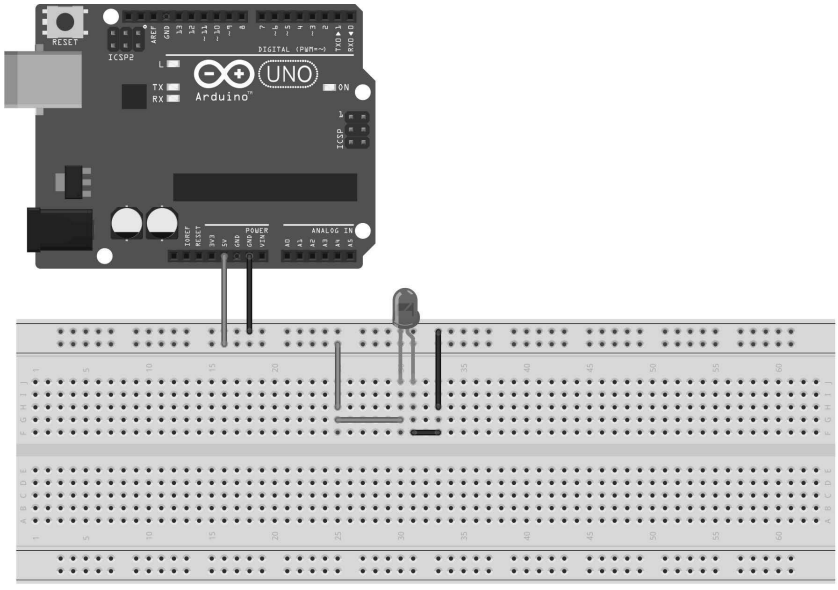
Materialien

Um jetzt in der Einführung weiterzumachen, brauchst du bereits einige Materialien. Dazu gehören eine LED (eine Leuchtdiode, also ein Bauteil, das Licht abstrahlt), Kabel, der Arduino und ein sogenanntes Breadboard.

Sämtliche benötigten Materialien sind übrigens in Anhang C am Ende des Buches aufgelistet.

Der erste Anfang

Nun wollen wir aber zum ersten Mal irgendetwas mit dem Arduino machen. Ich schlage vor, dass wir zuerst einfach eine LED leuchten lassen. Dies können wir auf zwei Arten machen: Entweder wir benutzen als Stromquelle den Arduino oder eine Batterie. Damit du dich gleich mit dem Arduino vertraut machst, baue einmal die Schaltung auf dem ersten Bild in diesem Buch nach. Dabei musst du Folgendes beachten: Das linke der obersten beiden (das hellere) Kabel kommt an den Pin des Arduinos mit der Beschriftung VCC, das das Kabel rechts daneben (das dunklere) an GND, also an Plus und Minus. Und zuletzt: Das helle Kabel (der Pluspol) wird an die lange Seite der LED angeschlossen, das kurze Ende an das dunkle Kabel (Minuspol).



Verwende in diesem Beispiel nur (!) die rote LED, für andere LEDs bräuchtest du definitiv einen Widerstand (sieh im nächsten Abschnitt nach).

Alle Schaltpläne kannst du dir unter *www.mitp.de/0649* herunterladen. Dort sind die Kabel auch farbig – im Gegensatz zu den Schaltplänen im Buch.

Hiermit hast du gleich vier wichtige Schaltungsbestandteile, die immer wieder im Buch vorkommen. Drei davon werden sogar in nahezu jeder Schaltung verwendet: der Arduino, das Breadboard, die LED und zuletzt die Kabel.

Widerstände

Leider geht das obige Beispiel nur mit roten LEDs. Andere würden wegen der Stromstärke durchbrennen. Deswegen braucht man ein weiteres Bauteil: den Widerstand.

Ein *Widerstand* ist ein kleines Bauteil, das dazu dient, den Stromfluss zu reduzieren, damit empfindlichere Bauteile nicht beschädigt werden. Hierbei ist es egal, wie herum ein Widerstand angeschlossen ist. Er sieht im Prinzip wie eine recht kleine Röhre aus, auf der mehrere farbige Streifen sind.

Widerstände sind relativ günstig (ungefähr ein Cent das Stück), sodass du 100 Stück für einen Euro bekommen kannst.

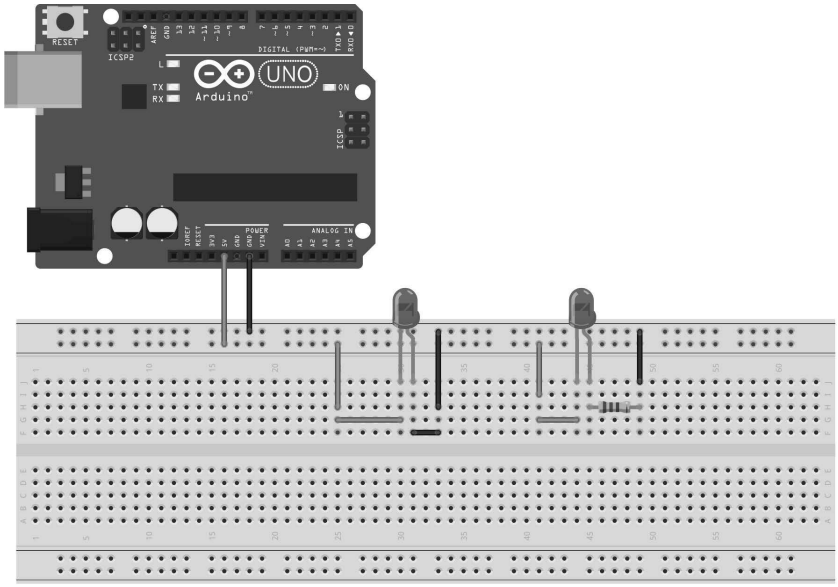
Wenn du jetzt die nächste Schaltung nachbaust (mit einem 130-Ohm-Widerstand), wirst du sehen, dass die LED mit Widerstand schwächer leuchtet, eben weil der Strom reduziert/geschwächt wird.

Ein wichtiger Tipp: Du musst im Buch immer lesen, welchen Widerstand man einfügen soll, denn die Abbildungen enthalten nur einen allgemeinen Platzhalter für den Widerstand.

In diesem Beispiel ist es so, dass die linke LED heller leuchtet als die rechte. Es ist übrigens egal, auf welcher Seite man den Widerstand anschließt. Jetzt aber die entscheidende Frage: Wie bestimmt man den Wert eines Widerstands, denn es gibt ja verschiedene Stärken?

Wenn du genau hinsiehst, hat jeder Widerstand einen Farbcode, also mehrere verschiedenfarbige Ringe. Je nach Farbe hat das Bauteil eine

andere Eigenschaft. Die Stärke des Widerstands wird übrigens in der Einheit Ohm gemessen. Mit der folgenden Tabelle kannst du den Wert eines Widerstands bestimmen.



Ein Tipp: Der goldene oder silberne Ring ist immer rechts, wenn du den Widerstand richtig hältst.

Farbe	1. Ring	2. Ring	Multiplikator	Toleranz
Silber			0,01	10%
Gold			0,1	5%
Schwarz	-	0	1	
Braun	1	1	10	
Rot	2	2	100	
Orange	3	3	1K = 1000	
Gelb	4	4	10K	
Grün	5	5	100K	
Blau	6	6	1M	
Violett	7	7	10M	
Grau	8	8	100M	
Weiß	9	9	1G	

Mit dieser Tabelle kannst du Widerstandswerte nachschlagen. Überall, wo ein »K« steht, musst du dir drei Nullen denken. Denn K steht für Kilo, also Tausend (1K entspricht 1.000, 10K entsprechen 10.000). M(ega) und G(iga) stehen für 1.000.000 bzw. 1.000.000.000.

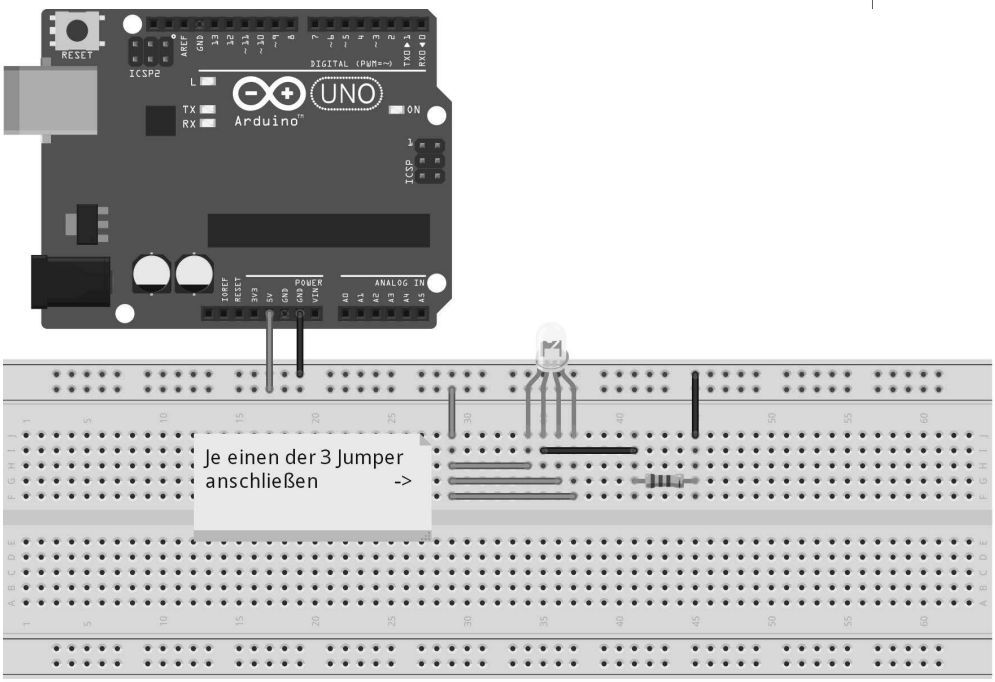
Die Toleranz steht dafür, wie sehr der tatsächliche Wert von dem angegebenen abweichen darf.

Beispielrechnung:

Wenn wir den Widerstand 130 Ohm brauchen, hat er folgende Farben: (1. Ring = Braun, 2. Ring = Orange und 3. Ring = Braun), denn 13 (1. und 2. Ring) + 10 (Multiplikator) gleich (=) 130 (Ohm).

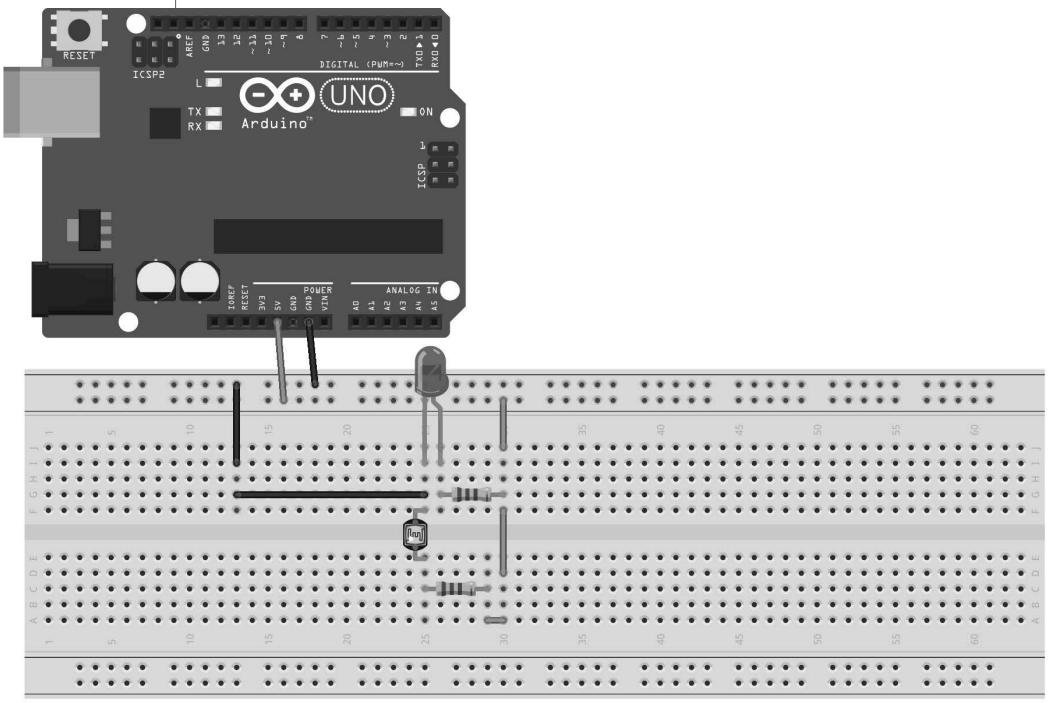
Eine Schaltung mit einer mehrfarbigen LED

Baue einmal die nächste Schaltung nach.



Dies ist eine Schaltung mit einer mehrfarbigen LED. Das heißt, dass eine LED in drei Farben leuchten kann (in der Regel Rot, Grün und Blau). In der obigen Schaltung brauchst du wieder einen 130-Ohm-Widerstand. Wenn du nun einen der drei markierten Jumper anschließt, leuchtet die spezielle LED in einer der drei Farben (aber schließ nie zwei Eingänge gleichzeitig an, davon geht die LED kaputt).

Dabei gibt es einen gemeinsamen Pluspol und für jede Farbe der LED einen Minuspol. Um dieses Kapitel gut abzuschließen, baue einfach die nächste Schaltung nach und sieh, was passiert. Das neue Bauteil ist ein Fotowiderstand, den ich aber erst in Kapitel 3 beschreibe.



Zusammenfassung

Du weißt nun, ...

- was ein Mikrocontroller bzw. ein Arduino ist
- was LEDs und Widerstände sind
- wie man die Stärke eines Widerstands an seinem Farbcode ablesen kann
- und wie man beide Bauteile in einem Stromkreis verwenden kann

Und du weißt ...

- wie man eine LED zerstören könnte (ob beabsichtigt oder nicht)

Ein paar Aufgaben

1. Baue mit drei LEDs und drei 330-Ohm-Widerständen (für den Farbcode sieh in die Tabelle weiter oben) eine Reihenschaltung auf – dies bedeutet, dass der Ausgang einer LED (Pluspol) mit dem Eingang der nächsten LED (Minuspol) verbunden ist und nur die erste bzw. letzte LED jeweils mit dem Arduino verbunden sind.

2. Als Nächstes kannst du dir ansehen, wie die Lichtstärke einer LED vom Widerstand abhängt. Verwende dafür verschiedene Widerstände (beispielsweise solche, die dir in einem Set geliefert wurden) und beobachte jeweils, ob bzw. bis zu welchem Wert die LED noch leuchtet.

3. Wenn du bei einer LED den Plus- und Minuspol vertauscht, geht diese kaputt. Neben den Leuchtdioden gibt es auch »normale« Dioden (diese können nicht leuchten), die den Stromfluss nur in eine Richtung durchlassen (aber nicht beschädigt werden, wenn der Strom falsch herum anliegt).

   Verwende mehrere solcher Dioden, um eine LED so anschließbar zu machen, dass sie unabhängig vom Anschluss der Stromquelle leuchtet (ohne beschädigt zu werden). Wenn du hier nicht weiterkommst, kannst du online nach dem Graetz-Gleichrichter recherchieren.

Und damit hast du schon die ersten Schritte geschafft. In den nächsten Kapiteln werden wir noch wesentlich interessantere Schaltungen bauen und anwenden. Also viel Spaß!

1 BLINKE, BLINKE, KLEINE LED

In diesem Kapitel wirst du lernen, wie du den Arduino programmierst und wie man LEDs (das sind kleine Lampen) mit dem Arduino ansteuert. Zusätzlich lernst du das Benutzen von Buttons (Tasten) und wie du sie mit dem Arduino verwenden kannst.

Uno R3
Uno R4 Minima
Uno R4 WiFi
Leonardo
Nano

Genau lernst du Folgendes:

- Anschließen des Arduinos
- Wie du LEDs mit dem Arduino an- und ausschaltest
- Wie du den Arduino pausierst
- Wie du einen Button (Knopftaste) verwendest
- Wie du ein eigenes, kleines Projekt planst

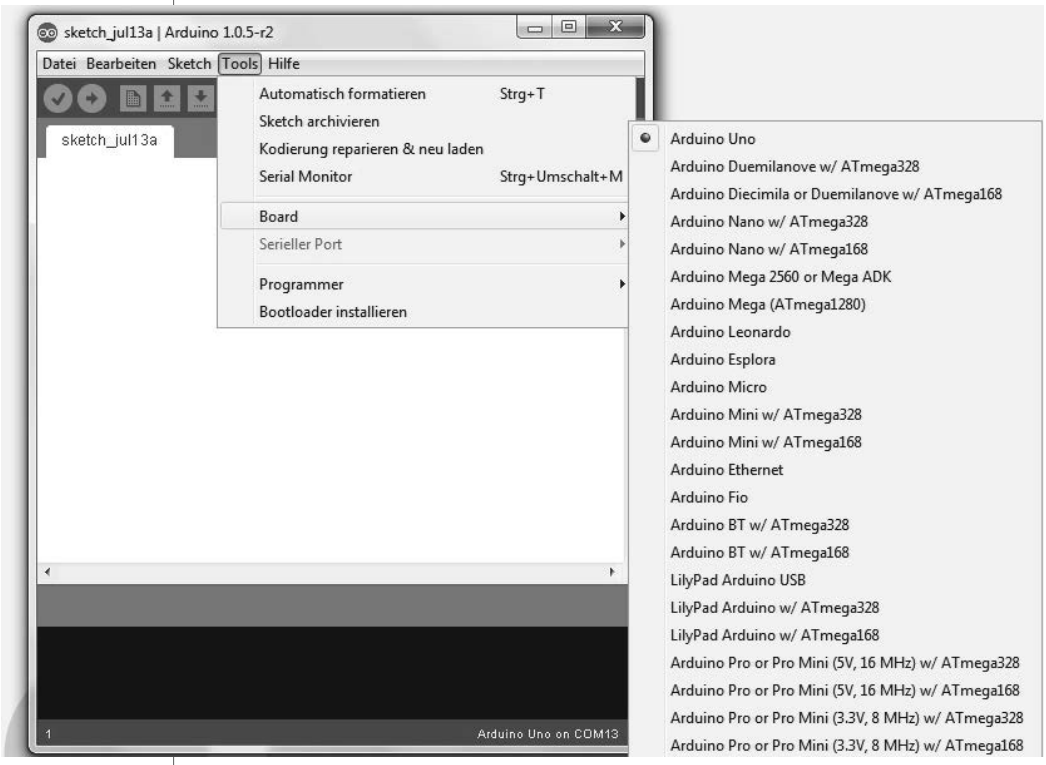
Am Ende des Kapitels werden wir ein kleines Projekt planen und programmieren: eine blinkende Lichterkette mit verschiedenen Funktionen. Im 2. Kapitel werden wir dann die Lichterkette so erweitern, dass wir sie über den PC steuern und die Funktionen austauschen können.

Die Installation prüfen

Als Erstes musst du Software installieren, mit der wir gleich arbeiten werden. Sieh dazu bitte in Anhang A, in dem das Vorgehen dazu erklärt wird. In der Installation ist die IDE (Integrated Development Environment) enthalten, ein Programm, um einen Quellcode zu schreiben und diesen dann in »Maschinensprache« zu übersetzen. Im Grunde genommen ist eine IDE eine Textbearbeitung, die Programmierbefehle farbig markiert.

Nachdem du die Installation (Anhang A) durchgeführt hast, kannst du deinen Arduino über USB-Kabel mit dem Computer verbinden und gleich programmieren. Zuerst musst du in deiner IDE den richtigen Arduino einstellen. Wenn du den empfohlenen Arduino Uno benutzt, musst du unter TOOLS|BOARD den Arduino auswählen, siehe dazu folgende Abbildung.

Wenn du ein anderes Arduino-Modell als den Arduino Uno verwendest (mögliche Arduinos werden dir in der Box am Anfang jedes Kapitels genannt), musst du hier das entsprechende Board in der IDE einstellen.



Danach musst du noch den richtigen Port einstellen: Zieh den Arduino Uno ab (falls du ihn bereits angeschlossen hattest), sieh dir unter TOOLS|SERIAL PORT die Ports an und notier sie dir. Danach steckst du den Arduino Uno ein. Den Port, der neu hinzugekommen ist, musst du auswählen. Jetzt kannst du endlich programmieren.

Unser erstes Programm

Unser erstes Programm soll lediglich zeigen, dass der Arduino korrekt verbunden ist.

```
void setup() {}
void loop() {}
```

Diesen Code (den Quellcode) musst du in der IDE eingeben und dann auf den Pfeil oben links klicken. Wenn dann in der schwarzen Textbox Done (schwarzes Feld im unteren Bereich der IDE) steht, ist der Arduino korrekt eingestellt, ansonsten muss du die Anleitung oben wiederholen oder im Anhang A nachsehen. Ein Programm für den Arduino, der sogenannte Sketch, besteht immer aus zwei Teilen: dem »Setup« und der »Loop«. Der Code, der im Setup steht, wird einmal beim Starten beziehungsweise beim Resetten des Controllers ausgeführt.

Bei einem Reset (»resetten« des Arduinos) wird der Arduino auf den Anfangspunkt eines Programms zurückgesetzt (das heißt, alle Variablen werden auf ihren ursprünglichen Wert aus dem Sketch zurückgesetzt und der Code wird von vorne, beginnend mit dem Setup-Teil, neu ausgeführt). Ein Reset findet beispielsweise statt, wenn du den Arduino von der Spannungsversorgung (beispielsweise dem USB-Kabel) abtrennst oder den beschrifteten Reset-Button auf dem Board drückst.

Der Code in der Loop wird dagegen in einer Endlosschleife ausgeführt. Der Quellcode, den du selbst schreibst, kommt dabei in die geschweiften Klammern. Warum das so ist, kannst du später in diesem Kapitel im Abschnitt »Wie Funktionen funktionieren« nachlesen.

Ein wichtiger Hinweis: Wir verwenden die LED-Schaltung aus der Einführung!

Unser erstes Programm mit einer richtigen Funktion soll eine LED ansteuern und sie anschalten, später soll sie blinken.