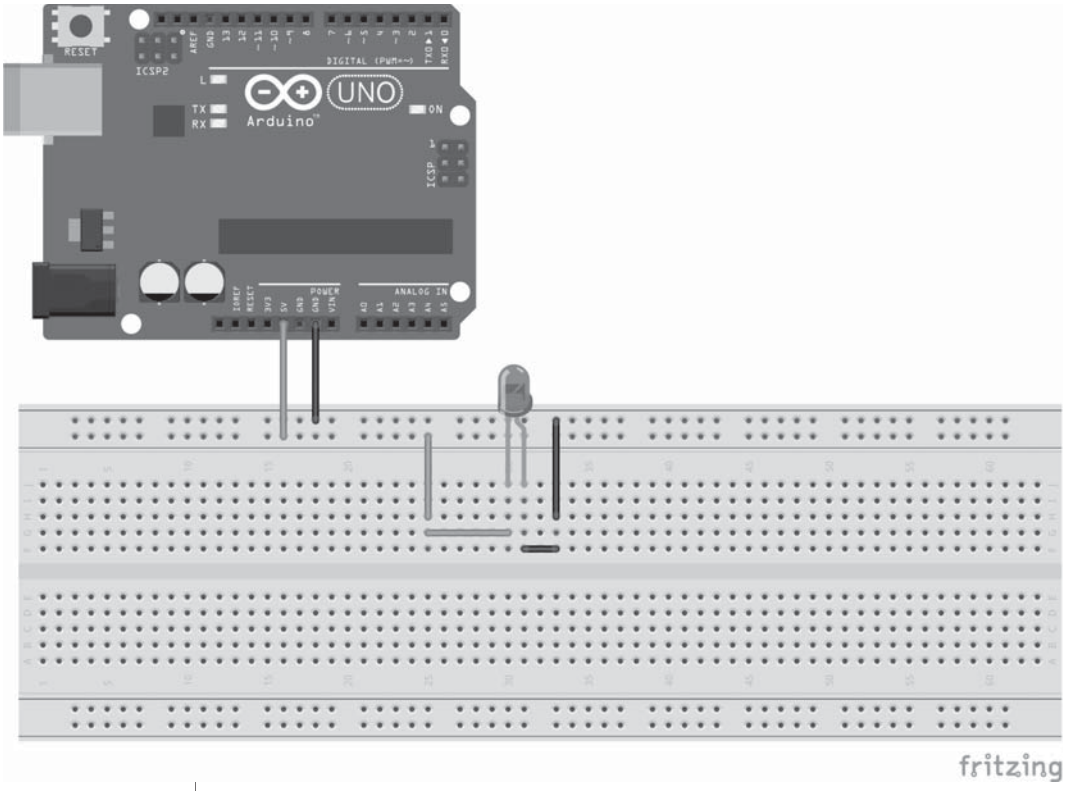
Der erste Anfang

Nun wollen wir aber zum ersten Mal irgendetwas mit dem Arduino machen. Ich schlage vor, zuerst lassen wir einfach eine LED leuchten. Dies können wir auf zwei Arten machen: Entweder wir benutzen den Arduino als Stromquelle 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 rote Kabel kommt an den Pin des Arduino mit der Beschriftung VCC, das schwarze an das Kabel mit GND, also an Plus und Minus. Und zuletzt: Das rote Kabel (der Pluspol) wird an die lange Seite der LED angeschlossen, das kurze Ende an das schwarze Kabel (Minuspol).

Verwende in diesem Beispiel nur (!) die rote LED, für andere LEDs bräuchtest du definitiv einen Widerstand (siehe weiter unten).

Alle Schaltpläne kannst du dir unter *www.mitp.de/580* herunterladen. Dort sind die Kabel auch farbig – im Gegensatz zu den Schaltplänen im Buch.



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

Auch wenn das letzte Bauteil eigentlich das einfachste ist, wirst du ohne dieses kaum eine Schaltung so wie im Buch beschrieben nachbauen können. Lediglich beim Löten kann man auf Kabel verzichten, aber auch hier gibt es Ausnahmen, bei denen man ein Kabel braucht.

Widerstände

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

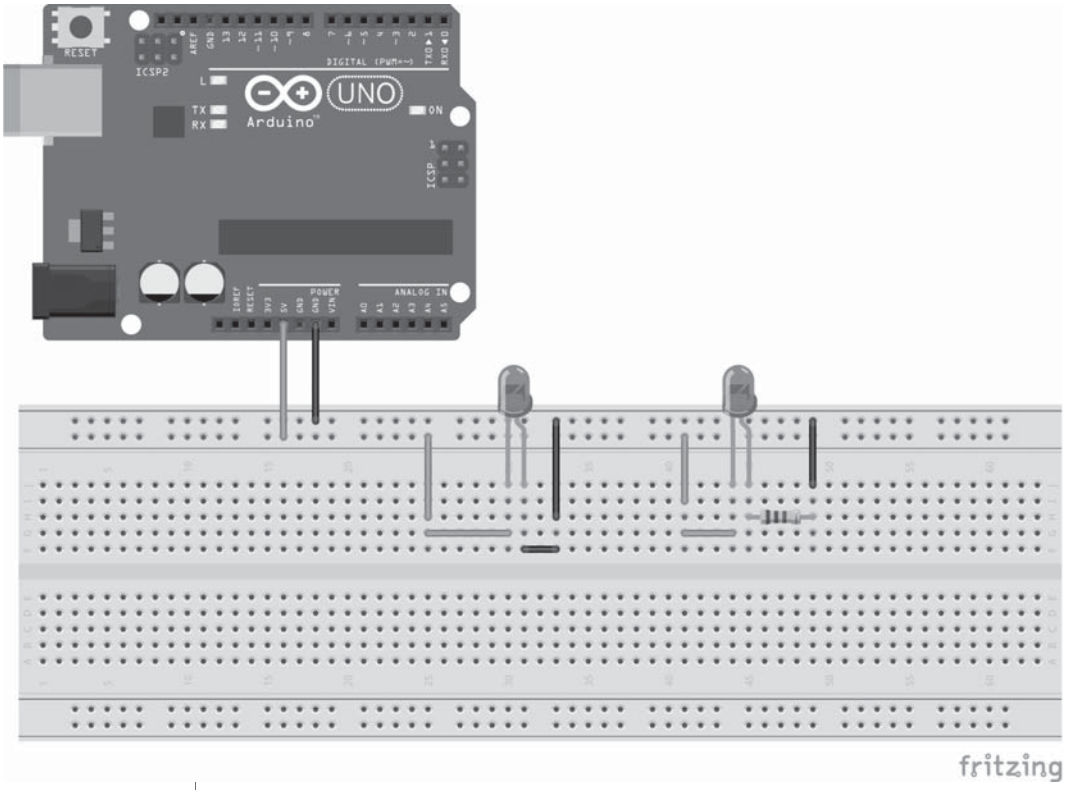
Ein *Widerstand* ist ein kleines Bauteil, das dazu dient, den Strom zu reduzieren, damit empfindlichere Bauteile nicht beschädigt werden. Hierbei ist es egal, wie herum ein Widerstand angeschlossen ist. Er sieht im Prinzip wie eine recht kleine Röhre aus, auf der mehrere farbige Streifen zu finden sind.

Widerstände sind relativ günstig (ungefähr 1 Cent das Stück), sodass du 100 für einen Euro bekommen kannst.

Wenn du jetzt die nächste Schaltung nachbaust, 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.

Im unteren Beispiel wird ein 130-Ohm-Widerstand gefordert.



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 Widerstandes, 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.

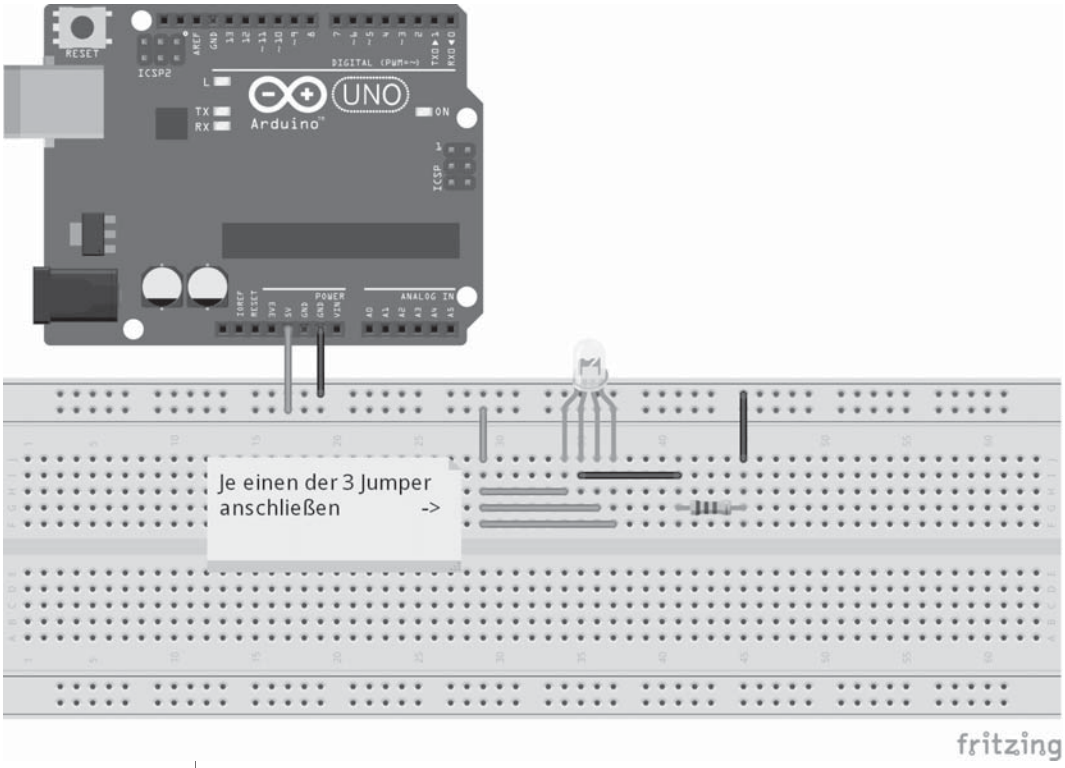
Die Toleranz steht dafür, wie sehr der Wert abweichen darf. Dieser Ring ist meistens goldenfarbig.

Beispielrechnung:

Wenn wir nun 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

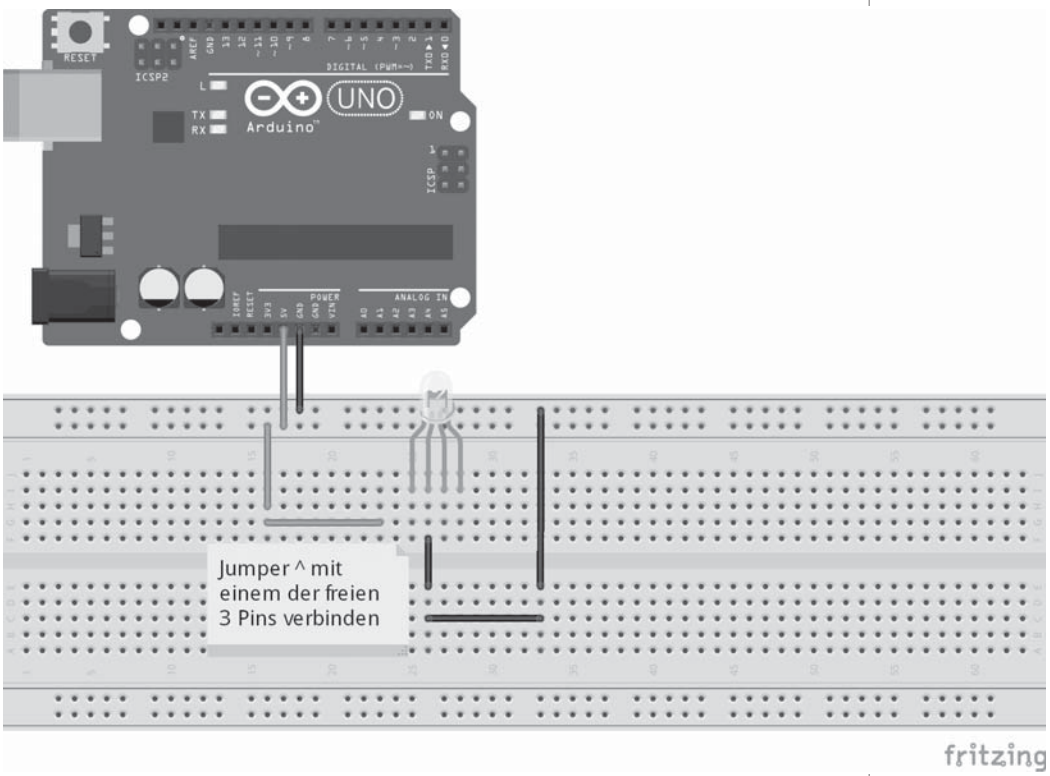
Baue einmal die nächste Schaltung nach.



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 das orangefarbene Kabel nimmst und an einen der freien (LED-)Pins hältst, leuchtet die spezielle LED in einer der drei Farben.

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 einem späteren Kapitel beschreibe.

Statt einer Batterie kannst du auch den Arduino benutzen (VCC->Rot, GND->Schwarz), du musst dann aber eventuell den Widerstand einbauen. Bei 3 V kann man ihn auch teilweise auslassen.



Zusammenfassung

Du weißt nun, ...

- wie man LEDs anschließt
- was ein Widerstand ist
- wie man mehrfarbige LEDs anschließt

Und du weißt ...

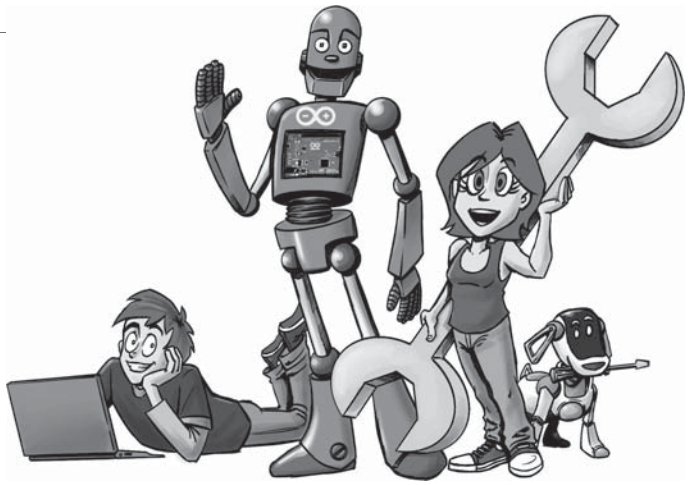
- Wie man eine LED zerstören könnte

Ein paar Aufgaben

1. Schalte drei LEDs in eine Reihe (+ 3 * 330-Ohm-Widerstände).
2. Ersetze den Arduino als Stromquelle durch eine Zungenbatterie und beobachte, wie sich die Lichtstärke der LEDs verändert.

3. Verwende mehrere Dioden, um eine LED leuchten zu lassen, egal wie sie an die Pole angeschlossen ist.
4. Wenn du die obere Aufgabe nicht schaffst, recherchiere nach dem Graetz-Gleichrichter.

Und damit hast du das erste Kapitel dieses Buches geschafft. In den nächsten Kapiteln werden wir noch wesentlich interessantere Schaltungen bauen und anwenden. Also noch viel Spaß!



1

Blinke, blinke, kleine LED

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

Genau lernst du Folgendes:

- Installieren der Software, Anschließen des Arduinos
- Wie du LEDs an- und ausschaltest
- Wie du den Arduino pausierst
- Auslesen eines Buttons
- Wie du ein eigenes Projekt planst

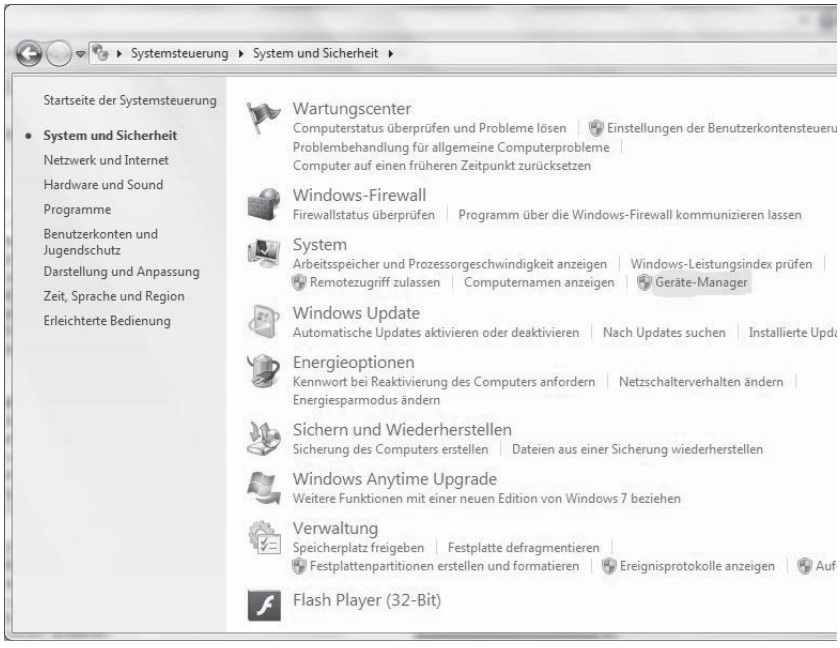
Am Ende des Kapitels werden wir ein kleines Projekt planen und programmieren: eine blinkende Lichterkette mit verschiedenen Funktionen. Im 2. Kapitel werden wir dann die Lichterkette so erweitern, dass wir sie über den PC steuern und die Funktionen austauschen können.

Die Software installieren

Das Installieren der Software ist relativ einfach. Zuerst musst du die Software hier herunterladen: *http://arduino.cc/en/Main/Software*.

Aktuell ist die Version 1.8.5. Das Archiv, das du herunterlädst, entpackst du bitte mit deiner Archivsoftware (beispielsweise Winrar oder 7Zip). Den Ordner kannst du dann zum Beispiel auf dem Desktop speichern. Darin enthalten ist die sogenannte IDE, ein Programm, um einen Quellcode zu entwickeln und diesen dann in »Maschinensprache« zu übersetzen. Im Grunde genommen ist eine IDE eine Textbearbeitung, die Programmierbefehle farbig markiert.

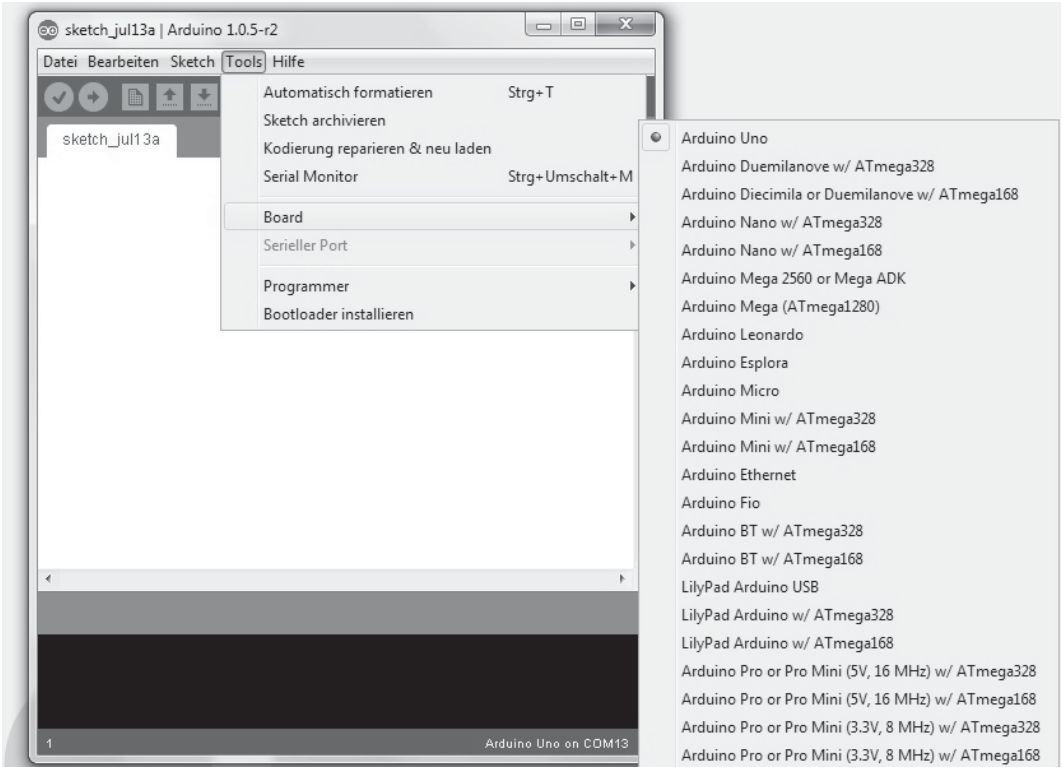
Nun zu dem wirklich schwierigen Teil der Installation, das Installieren der Treiber. Wenn du Windows benutzt, bist du leider darauf angewiesen, Treiber zu installieren, damit du den Arduino programmieren kannst. Stecke den Arduino (Uno) an und warte. Es sollte sich nach kurzer Zeit ein Fenster öffnen, in dem nach Treibern/Updates gesucht wird. Ignoriere dies und öffne die Systemsteuerung. Gehe dann auf SYSTEM & SICHERHEIT. Dort wählst du den Menüpunkt SYSTEM|GERÄTE-MANAGER.



In dem sich neu öffnenden Fenster musst du den ersten Punkt (Anschlüsse) entfalten. Nun sollte dort ein Eintrag mit Arduino sein. Gehe

mit einem Rechtsklick darauf, wähle TREIBERSOFTWARE AKTUALISIEREN und im sich öffnenden Fenster AUF DEM COMPUTER NACH TREIBERSOFTWARE SUCHEN. Nun musst du den Ordner auf dem Desktop suchen, in den du vorhin die IDE entpackt hast. Im Subordner `Drivers` (DESKTOP|ARDUINO|DRIVERS) findest du eine Datei mit dem Namen `ArduinoUno.inf`. Wähle diese aus und die Treiberinstallation ist abgeschlossen. Solltest du ein anderes Board verwenden, musst du dann natürlich einen anderen Treiber (aus dem gleichen Ordner) auswählen.

Hiermit ist die Installation schon fertig. Nun 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.



Danach musst du noch den richtigen Port einstellen: Ziehe den Arduino Uno ab (falls du ihn bereits angeschlossen hattest) und schau dir unter TOOLS|SERIAL PORT die Ports an und notiere 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. Der Code in der Loop wird dagegen in einer Endlosschleife ausgeführt. Der Quellcode, den du selber schreibst, kommt dabei in die geschweiften Klammern. Warum das so ist, kannst du später in diesem Kapitel im Bereich »Wie Funktionen funktionieren« nachlesen.

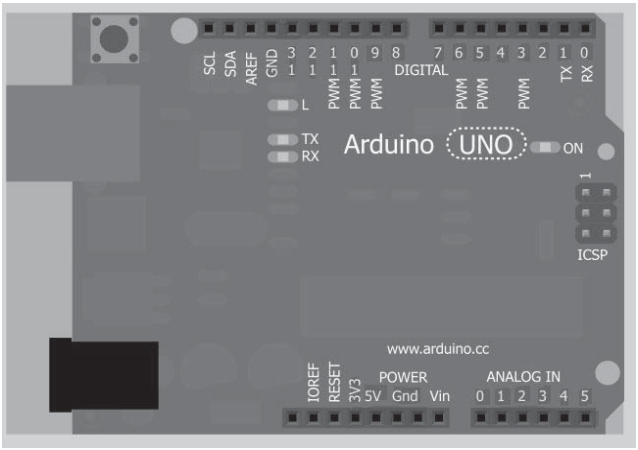
Ein wichtiger Hinweis: Wir verwenden die LED-Schaltung aus der Einführung!

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

```
void setup() {
pinMode(13,OUTPUT); //Pin 13 auf Ausgang
digitalWrite(13,HIGH);//Pin 13 Anschalten
}
void loop() {}
```

Ein wichtiger Hinweis: Alles, was hinter zwei Schrägstrichen steht, wird in einem Programm immer als Kommentar gewertet, das heißt, dass dahinter geschriebener Text keine Funktion hat, sondern nur zur besseren Lesbarkeit dient.

Bei einem Arduino sind alle Ausgänge als sogenannte Pins konfiguriert. Das heißt, dass du jeden Pin einzeln ansteuern kannst. Die Pins sind die kleinen Metallkontakte in den schwarzen Buchsen des Arduino (Uno). Häufig genutzte Pins sind gelb markiert.



Neben jedem Pin ist eine Nummer, über die du den Pin ansteuern kannst. In dem Beispiel verwende ich Pin 13, da an diesem bereits eine LED verbaut ist. Diesen Code musst du wie beim Beispiel auf den Arduino laden und schon leuchtet die LED grün. Oben im Code kannst du erkennen, dass der Befehl `pinMode` eine *Funktion* ist. In den Klammern werden der Funktion *Parameter* übergeben. Parameter kann man als Informationen ansehen, die die Funktion zum Arbeiten benötigt. Hier ist der erste Parameter der Pin, der zweite der Zustand. Dies kann Output (Ausgang) sein oder Input (Eingang), den wir aber erst im Abschnitt »Das Lesen von Eingängen« benötigen.

```
void setup() {
pinMode(13,OUTPUT); //Pin 13 auf Ausgang
digitalWrite(13,HIGH);//Pin 13 Anschalten
}
void loop() {}
```

Ausgänge schalten immer etwas, beispielsweise LEDs, Eingänge lesen immer etwas, z.B. Tasten. Eine Funktion wird immer mit einem Semikolon (;) abgeschlossen, damit der Übersetzer weiß, dass der Befehl zu Ende ist. Der Übersetzer, der sogenannte *Compiler*, übersetzt den Quellcode in Maschinensprache, also in das Dualsystem (dies sind die Einsen und Nullen, mit denen der Computer arbeiteten und von denen du vielleicht schon einmal gehört hast), damit es der Mikrocontroller verstehen kann. Der Befehl `pinMode()` ist notwendig, damit der Arduino weiß, wie der Pin zu benutzen ist. Der zweite Befehl, `digitalWrite()`, schaltet dann den Pin an. Bei dem Arduino Uno liegen 5 Volt an. Bei `digitalWrite()` kannst du bei dem zweiten Parameter entweder `HIGH` angeben, der PIN führt also Strom, oder `LOW`, der PIN ist ausgeschaltet.