2 Arduino

Der Arduino ist eine einfache und robuste sowie äußerst zuverlässige Entwicklungsplatine (siehe Abb. 2–1), mit der sich auf einfache Art und Weise programmierbare elektronische Schaltungen aufbauen lassen.

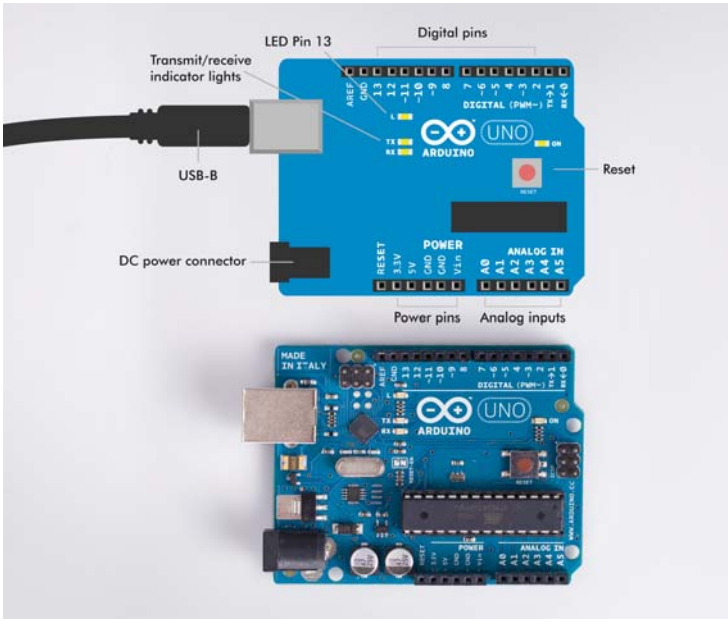

Abb. 2–1 *Anschlüsse am Arduino*

Für den Einstieg in den Arduino ist nicht viel Material erforderlich. Um interessante Dinge zu machen, brauchen Sie lediglich einen Arduino Uno und ein USB-Kabel. Beides zusammen kostet nicht mehr als 35 oder 40 €. Die Software ist kostenlos und quelloffen (der Quellcode steht zur Verwendung, Untersuchung, Änderung und Weitergabe zur Verfügung).

Als Erstes zeigen wir Ihnen, wie Sie die Arduino-Entwicklungsumgebung (Integrated Development Environment, IDE) auf Ihrem Computer installieren. Danach schließen Sie ein Kabel an und laden Ihr erstes Programm hoch (einen

Sketch, wie es im Arduino-Jargon heißt). Auf dem Arduino installieren Sie nur ein Programm, nämlich den Sketch, den Sie ausführen wollen. Ansonsten gibt es keine Software, um die Sie sich kümmern müssten, da der Arduino im Gegensatz zum Raspberry Pi kein Betriebssystem hat. Beteiligt sind nur Sie, Ihr Programm und die Hardware.

In Wirklichkeit gibt es doch noch eine zusätzliche Software. Der Arduino verfügt über einen Bootloader, der einen kleinen Teil des verfügbaren Speicherplatzes auf dem Chip einnimmt. Es handelt sich dabei um ein kleines Programm, das kurz ausgeführt wird, wenn Sie die Platine einschalten oder zurücksetzen, und dafür sorgt, dass Sie Programme einfach über USB laden können, ohne dazu ein Hardware-Programmiergerät zu benötigen.

Der Arduino Uno ist robust. Es ist unwahrscheinlich, dass er Schaden nimmt, selbst wenn Sie ein Kabel falsch herum anschließen sollten. (Werden Sie aber nicht nachlässig, denn wenn Sie den Arduino allzu sehr malträtieren, ist es durchaus möglich, dass ein Pin durchbrennt.)

Den Umgang mit dem Arduino zu erlernen, ist sehr einfach. Schon Anfänger können viele Dinge fertigbringen, indem sie Pins ein- und ausschalten. Anders als beim Raspberry Pi können Sie analoge Widerstandssensoren direkt am Arduino einstecken, da er über einen eingebauten Analog-Digital-Wandler verfügt.

2.1 Grundinstallation des Arduino

Um den Arduino unter Linux, unter Windows und auf dem Mac einzurichten, gehen Sie folgendermaßen vor.

2.1.1 Ubuntu Linux

Schließen Sie den Arduino über ein USB-Kabel an Ihren Computer an. Der Arduino bezieht seinen Strom direkt über die USB-Verbindung, sodass Sie keine externe Stromversorgung brauchen. Starten Sie die Terminalanwendung.

Sie können das Terminal mit der Kommandozeile auf verschiedene Weisen starten. Im Hauptmenü können Sie es mit *Applications > Accessories > Terminal* öffnen (auf Xubuntu und anderen XFCE-gestützten Distributionen wie Debian mit XFCE). Auf vielen Desktops funktioniert auch die Tastenkombination **(Super)** + **(T)**, also die »Betriebssystem-Taste« (auch »die hässliche Taste« oder »Windows-Taste« genannt) und **(T)**. Wenn Sie Unity in der Ubuntu-Standarddistribution verwenden, suchen Sie nach »Terminal« (obere linke Ecke).

Um die Arduino-IDE verwenden zu können, müssen Sie das Paket `arduino` installieren. Unter Ubuntu Linux gehen Sie dazu wie folgt vor:

```
$ sudo apt-get update
$ sudo apt-get -y install arduino
```

Geben Sie sich selbst die Berechtigung für den Zugriff auf die serielle Schnittstelle über USB. (Das ist notwendig, damit die Arduino-Entwicklungsumgebung funktionieren kann.) Der erste Befehl fügt Sie zur Gruppe `dialout` hinzu, der zweite schaltet auf diese Gruppe um, ohne dass Sie sich abmelden und wieder neu anmelden müssen:

```
$ sudo adduser $(whoami) dialout
$ newgrp dialout
```

Starten Sie nun die Arduino-IDE:

```
$ arduino
```

Die Arduino-IDE wird geöffnet.

Nachdem Sie sich abgemeldet und wieder angemeldet haben, können Sie die Arduino-IDE auch über die Menüs starten.

Nun ist alles bereit, um die Installation zu prüfen. Wie das geht, erfahren Sie in Abschnitt 2.1.4.

2.1.2 Windows 7 und Windows 8

Laden Sie die jüngste Version der Arduino-Software von *http://arduino.cc/en/Main/Software* herunter. Entpacken Sie die Datei in einem Speicherort Ihrer Wahl (z. B. auf dem Desktop oder im Ordner *Downloads*).

Schließen Sie den Arduino über ein USB-Kabel an Ihren Computer an. Der Arduino bezieht seinen Strom direkt über die USB-Verbindung, sodass Sie keine externe Stromversorgung brauchen. Windows startet automatisch einen Installationsprozess für die Arduino-Treiber. Es kann allerdings sein, dass diese Installation nach einer Weile fehlschlägt und eine Fehlermeldung angezeigt wird.

Nehmen Sie die Installation der Treiber in diesem Fall auf folgende Weise vor:

1. Öffnen Sie den Windows Explorer, rechtsklicken Sie auf *Computer* und wählen Sie *Verwalten*.
2. Wählen Sie unter *Computerverwaltung* auf der linken Seite den Punkt *Geräte-Manager*. Suchen Sie in der Geräteliste den Arduino Uno, rechtsklicken Sie darauf und wählen Sie *Treiber aktualisieren*.
3. Wählen Sie *Auf dem Computer nach Treibersoftware suchen*. Wechseln Sie zu dem entpackten Arduino-Ordner, öffnen Sie den Ordner *drivers*, wählen Sie *arduino.inf* aus und klicken Sie auf *Weiter*.
4. Windows installiert den Treiber.

Starten Sie die Arduino-IDE, indem Sie auf das Arduino-Symbol in dem entpackten Ordner doppelklicken. Testen Sie anschließend Ihre Installation wie in Abschnitt 2.1.4 beschrieben.

2.1.3 OS X

Laden Sie die jüngste Version der Arduino-Software von *http://arduino.cc/en/Main/Software* herunter. Entpacken Sie die Datei und kopieren Sie sie in Ihren Programmordner (*/Applications*).

Schließen Sie den Arduino über ein USB-Kabel an Ihren Computer an. Der Arduino bezieht seinen Strom direkt über die USB-Verbindung, sodass Sie keine externe Stromversorgung brauchen. Für OS X müssen Sie keinen Treiber installieren.

Starten Sie die Arduino-IDE, indem Sie auf das Arduino-Symbol im Programmordner doppelklicken. Testen Sie anschließend Ihre Installation wie im folgenden Abschnitt beschrieben.

2.1.4 Hello World

Wenn die Arduino-IDE läuft, können Sie das Arduino-Gegenstück zu einem »Hello World«-Programm ausführen.

Als Erstes sollten Sie sicherstellen, dass Sie in der IDE die richtige Platine ausgewählt haben. Der Arduino Uno ist vorausgewählt. Wenn Sie über eine andere Platine verfügen, z. B. über einen Arduino Mega oder Leonardo, wählen Sie diesen Typ im Menü *Tools > Board* aus.

Nun müssen Sie das Blinktestprogramm laden. Wählen Sie dazu *File > Examples > 1.Basics > Blink*. Klicken Sie auf die Schaltfläche *Upload* (oder wählen Sie *File > Upload*), um das Programm zu kompilieren und auf den Arduino hochzuladen.

Beim ersten Versuch kann es passieren, dass der Arduino eine Fehlermeldung anzeigt: »Serial port COM1 not found (serieller Port COM1 nicht gefunden).« Das liegt daran, dass Sie noch nicht festgelegt haben, welcher serielle Port verwendet werden soll. (Die Verbindung zwischen Computer und Arduino wird durch einen seriellen USB-Port dargestellt.) Wählen Sie den Port aus dem Dropdown-Menü aus. Unter Linux ist das vermutlich `/dev/ttyACM0`, auf dem Mac wahrscheinlich etwas wie `/dev/usbmodem1234` und unter Windows einer der COM-Ports.

Wenn Sie nicht die Aufforderung zur Auswahl eines seriellen Ports angezeigt bekommen, sondern eine andere Fehlermeldung, wählen Sie den seriellen Port unter *Tools > Port* aus. Sollten Sie nicht herausfinden können, an welchem Port der Arduino angeschlossen ist, schauen Sie sich an, welche Ports aufgeführt werden, ziehen Sie dann den Arduino ab und merken Sie sich, welcher Port daraufhin verschwindet. Das muss logischerweise der Arduino-Port sein. OS X führt alle Ports zweimal auf, z. B. als `/dev/cu.usbmodem1234` und als `/dev/tty.usbmodem1234`. Beide Varianten funktionieren.

Während das Programm hochgeladen wird, blinken die TX- und die RX-Leuchte des Arduino (für Transmit/Übertragen bzw. Receive/Empfangen) in rascher Folge. Wenn das Programm läuft, leuchtet die winzige Lampe mit der Beschriftung L.

Wenn die L-LED blinkt, heißt das, dass alles erfolgreich installiert war und Ihr erster Sketch läuft!

Herzlichen Glückwunsch! Merken Sie sich diesen einfachen Trick: Wenn Sie nicht mehr weiterkommen und sich fragen, ob der Arduino überhaupt irgendwelchen Code ausführt, testen Sie ihn mit diesem Hello-World-Beispiel. Wann immer Sie ein neues Programm starten, beginnen Sie mit »Hello World«, um sicherzustellen, dass alles funktioniert.

2.1.5 Der Aufbau eines Arduino-Programms

Zu Anfang eines Arduino-Programms wird der Code innerhalb der Funktion `setup()` einmalig ausgeführt. Anschließend wird der Code in `loop()` endlos wiederholt (zumindest, bis Sie die Stromversorgung unterbrechen). Schauen Sie sich dazu Listing 2–1 an.

```
// blink.ino - Blinken der L-LED, um Entwicklungsumgebung zu testen
// (c) BotBook.com - Karvinen, Karvinen, Valtokari
void setup() {            1
   pinMode(13, OUTPUT);       2
}
void loop() {             3
   digitalWrite(13, HIGH);       4
   delay(1000); // ms        5
   digitalWrite(13, LOW);
   delay(1000);
}
```

Listing 2–1 *blink.ino*

1. Beim Start des Arduino wird `setup()` einmalig ausgeführt.
2. Konfiguriert Digitalpin D13 als Ausgang, sodass Sie ihn von Ihrem Programm aus steuern können.
3. Nachdem `setup()` ausgeführt ist, wird `loop()` aufgerufen. Ist `loop()` durchgelaufen, wird es erneut aufgerufen. Und noch einmal und immer so weiter.
4. Setzt D13 auf `HIGH`, was bedeutet, dass +5 V an den Pin angelegt werden.
5. Während dieser Verzögerungszeit bleiben die Pins im gegenwärtigen Zustand. Hier also bleibt D13 auf `HIGH`, sodass die eingebaute L-LED des Arduino weiterhin leuchtet. Während der nächsten Verzögerungsperiode ist der Pin jedoch auf `LOW` gesetzt, sodass die LED ausgeschaltet bleibt. Die Leuchtdiode ist also jeweils eine Sekunde (1000 Millisekunden) lang eingeschaltet und eine Sekunde lang ausgeschaltet, und das in endlosem Wechsel.

2.1.6 Einfach und vielseitig dank Shields

Shields sind Platinen, die Sie auf den Arduino aufstecken, um seinen Funktionsumfang zu erweitern oder ihn leichter verwendbar zu machen (siehe Abb. 2–2). Es gibt viele verschiedene Shields, von einfachen zur Entwicklung von Prototypen bis zu komplizierten Modellen für einen Ethernet- oder WLAN-Zugang. Ein besonderer Vorteil der Shields besteht darin, dass sie die Kosten für zusätzliche Kabel verringern, da sie auf den Arduino aufgesteckt werden und die Verbindung von Pin zu Pin herstellen, ohne dass Jumperkabel erforderlich wären. Natürlich gibt es nicht für alle Ihre Bedürfnisse passende Shields, aber sie stellen trotzdem eine gute Erweiterungsmöglichkeit dar, die Sie stets im Hinterkopf behalten sollten.

Manche Shields enthalten keine elektronischen Bauteile, sondern dienen nur dazu, Ihnen die Entwicklung von Prototypen zu erleichtern: Meistens führen sie die Pins der Arduino-Anschlussleisten zu einem Steckbrett heraus, sodass Sie dort Bauelemente und Jumperkabel auf einfache Weise anschließen können. Unser Favorit ist der unschätzbare ScrewShield, der an beiden Seiten des Arduino »Flügel« mit Anschlussklemmen bereitstellt. Dadurch werden lose Kabel vermieden, die wahrscheinlich das größte Ärgernis bei der Prototypentwicklung sind.

Sie können auch Ihre eigenen Shields als leicht verwendbare und vielseitige Arduino-Ergänzungsplatinen konstruieren (siehe Abb. 2–3). Einzige Voraussetzung ist, dass Sie die Pinleisten so auf die Platine löten, dass sie der Pinordnung auf dem Arduino entsprechen.

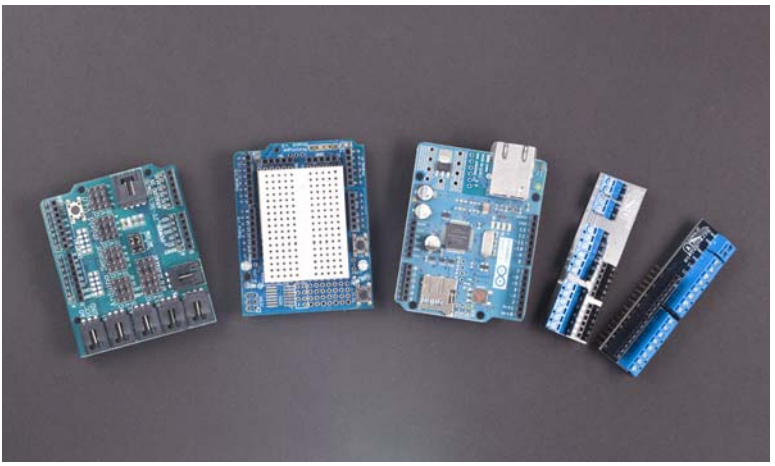

Abb. 2–2 *Shields*

Abb. 2–3 *Shields von Andreas Zingerle*