TURN ON YOUR CREATIVITY

FRANZIS MAKER KIT +ARDUINO™

HANDBUCH

FRANZIS

Franzis Maker Kit +Arduino™

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.

Achtung! Augenschutz und LEDs:

Blicken Sie nicht aus geringer Entfernung direkt in eine LED, denn ein direkter Blick kann Netzhautschäden verursachen! Dies gilt besonders für helle LEDs im klaren Gehäuse sowie in besonderem Maße für Power-LEDs. Bei weißen, blauen, violetten und ultravioletten LEDs gibt die scheinbare Helligkeit einen falschen Eindruck von der tatsächlichen Gefahr für Ihre Augen. Besondere Vorsicht ist bei der Verwendung von Sammellinsen geboten. Betreiben Sie die LEDs so wie in der Anleitung vorgesehen, nicht aber mit größeren Strömen.

Liebe Kunden!

Dieses Produkt wurde in Übereinstimmung mit den geltenden europäischen Richtlinien hergestellt und trägt daher das CE-Zeichen. Der bestimmungsgemäße Gebrauch ist in der beiliegenden Anleitung beschrieben.

Bei jeder anderen Nutzung oder Veränderung des Produktes sind allein Sie für die Einhaltung der geltenden Regeln verantwortlich. Bauen Sie die Schaltungen deshalb genau so auf, wie es in der Anleitung beschrieben wird.
Das Produkt darf nur zusammen mit dieser Anleitung weitergegeben werden.

Das Symbol der durchkreuzten Mülltonne bedeutet, dass dieses Produkt getrennt vom Hausmüll als Elektroschrott dem Recycling zugeführt werden muss. Wo Sie die nächstgelegene kostenlose Annahmestelle finden, sagt Ihnen Ihre kommunale Verwaltung.

Autor/Author: Christian Immler
Art & Design: www.ideehoch2.de

Inhalt

Alle Skripte/Quelltexte finden Sie zum Download auf *www.buch.cd*.

BEVOR ES LOSGEHT ...

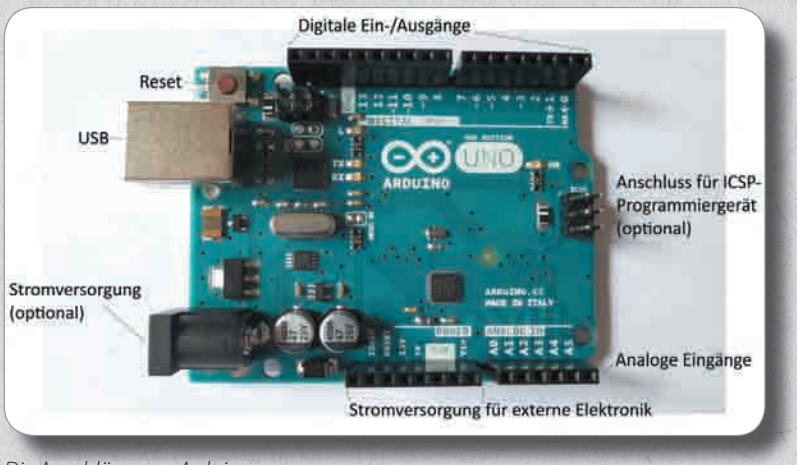
Das Programmieren von Mikrocontrollern war früher nur etwas für Ingenieure und Informatiker. Arduino ermöglicht dank übersichtlicher Hardware und einfach zu verstehender Software auf einmal jedem den Einstieg in die Mikrocontrollertechnik. Die Arduino-Plattform wurde ursprünglich für Bastler und Künstler entwickelt, die damit interaktive elektronische Objekte schaffen können, und ist inzwischen in der Maker-Szene zum allgemeinen Standard für mikrocontrollergesteuerte Hardwareprojekte geworden.

Der Name Arduino

Der Arduino kommt aus Italien und wurde nach dem italienischen König Arduino benannt, der bis ins Jahr 1005 in Ivrea, dem Firmensitz des Arduino-Herstellers, herrschte. Nach König Arduino ist dort heute die Lieblingsbar der Arduino-Entwickler Massimo Banzi und David Cuartielles benannt.

Arduino UNO

Die Arduino-Plattform bietet mittlerweile eine große Vielfalt an Platinen für unterschiedliche Anwendungszwecke. Der Arduino UNO ist die für Einsteiger am besten geeignete und auch eine der leistungsfähigeren Arduino-Platinen. In diesem Maker Kit ist ein solcher Arduino UNO enthalten.



Die Anschlüsse am Arduino.

Auf der Arduino-Platine sind vier LEDs in SMD-Bauweise fest aufgelötet. Diese sind mit Buchstaben auf der Platine deutlich gekennzeichnet:

ON	Leuchtet, wenn der Arduino mit Strom versorgt ist.
L	Frei programmierbar, verbunden mit Pin 13.
TX	Leuchtet, wenn der Arduino Daten über USB oder seriell sendet.
RX	Leuchtet, wenn der Arduino Daten über USB oder seriell empfängt.

Bauteile im Maker Kit

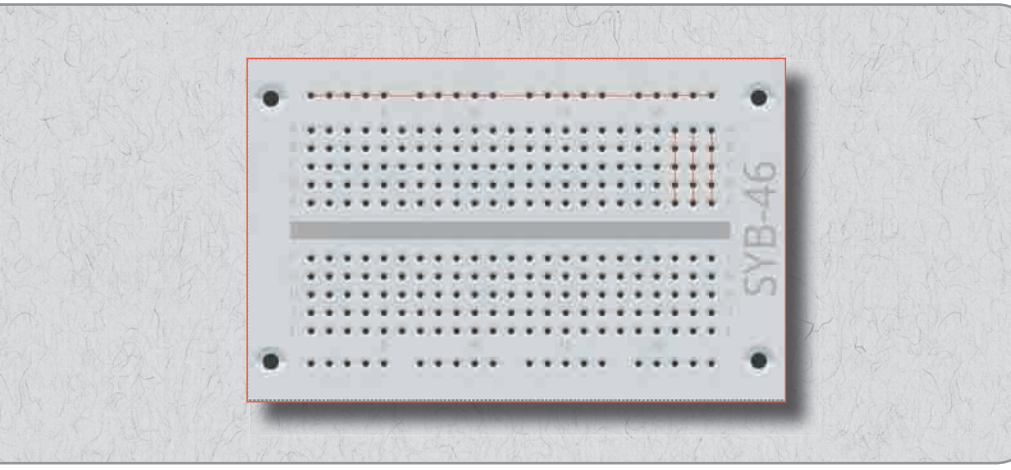
Das Maker Kit enthält diverse elektronische Bauteile, mit denen sich die beschriebenen Experimente (und natürlich auch eigene) aufbauen lassen. An dieser Stelle werden die Bauteile nur kurz vorgestellt. Die notwendige praktische Erfahrung im Umgang damit bringen dann die wirklichen Experimente.

- Arduino UNO
- 2x Steckplatine

- 1x LCD-Anzeige
- 1x Pfostenverbinder (16-polig)
- 7x LED
- 4x Taster
- 4x Widerstand 10 kOhm (Braun-Schwarz-Orange)
- 7x Widerstand 220 Ohm (Rot-Rot-Braun)
- 1x Widerstand 560 Ohm (Grün-Blau-Braun)
- 6x Widerstand 22 MOhm (Rot-Rot-Blau)
- 1x Potenziometer 15 kOhm
- 14x Verbindungskabel (Steckplatine – Arduino)
- Schaltdraht

Steckplatinen

Für den schnellen Aufbau elektronischer Schaltungen, ohne löten zu müssen, sind zwei Steckplatinen im Maker Kit enthalten. Hier können elektronische Bauteile direkt in ein Lochraster mit Standardabständen eingesteckt werden. Bei diesen Platinen sind die äußeren Längsreihen mit Kontakten (X und Y) alle miteinander verbunden. Diese Kontaktreihen werden oft als Plus- und Minuspol zur Stromversorgung der Schaltungen genutzt.



Die Steckplatine aus dem Maker Kit mit, beispielhaft angedeutet, einigen Verbindungen.

In den anderen Kontaktreihen sind jeweils fünf Kontakte (A bis E und F bis J) quer miteinander verbunden, wobei in der Mitte der Platine eine Lücke ist. So können an dieser Stelle größere Bauelemente, wie z. B. Taster, eingesteckt und nach außen hin verdrahtet werden.

Verbindungskabel und Schaltdraht

Die farbigen Verbindungskabel haben auf beiden Seiten einen dünnen Drahtstecker, mit dem sie sich in den Buchsenleisten des Arduino und auf der Steckplatine einstecken lassen.

Weiterhin ist Schaltdraht im Paket enthalten. Damit stellen Sie kurze Verbindungsbrücken her, mit denen Kontaktreihen auf der Steckplatine verbunden werden. Schneiden Sie den Draht mit einem kleinen Seitenschneider je nach Experiment auf die passenden Längen ab. Um die Drähte besser in die Steckplatine stecken zu können, empfiehlt es sich, die Drähte leicht schräg abzuschneiden, sodass eine Art Keil entsteht. Entfernen Sie an beiden Enden auf einer Länge von etwa einem halben Zentimeter die Isolierung.

Widerstände und ihre Farbcodes

Widerstände werden in der Digitalelektronik im Wesentlichen zur Strombegrenzung an den Ports eines Mikrocontrollers sowie als Vorwiderstände für LEDs verwendet. Die Maßeinheit für Widerstände ist Ohm. 1.000 Ohm entsprechen einem Kiloohm, abgekürzt kOhm, 1.000 kOhm entsprechen einem Megaohm, abgekürzt MOhm.

Die Widerstandswerte werden auf den Widerständen durch farbige Ringe angegeben. Die meisten Widerstände haben vier solcher Farbringe. Die ersten beiden Farbringe bezeichnen die Ziffern, der dritte einen Multiplikator und der vierte die Toleranz. Dieser Toleranzring ist meistens gold- oder silberfarben – das sind Farben, die auf den ersten Ringen nicht vorkommen, sodass die Leserichtung eindeutig ist. Der Toleranzwert selbst spielt in der Digitalelektronik kaum eine Rolle. Beim Schaltungsaufbau ist die Richtung, in der ein Widerstand eingebaut wird, egal.

Farbe	Widerstandswert in Ohm			
	1. Ring (Zehner)	2. Ring (Einer)	3. Ring (Multiplikator)	4. Ring (Toleranz)
Silber			$10^{-2} = 0{,}01$	±10 %
Gold			$10^{-1} = 0{,}1$	±5 %
Schwarz		0	$10^{0} = 1$	
Braun	1	1	$10^{1} = 10$	±1 %
Rot	2	2	$10^{2} = 100$	±2 %
Orange	3	3	$10^{3} = 1.000$	
Gelb	4	4	$10^{4} = 10.000$	
Grün	5	5	$10^{5} = 100.000$	±0,5 %
Blau	6	6	$10^{6} = 1.000.000$	±0,25 %
Violett	7	7	$10^{7} = 10.000.000$	±0,1 %
Grau	8	8	$10^{8} = 100.000.000$	±0,05 %
Weiß	9	9	$10^{9} = 1.000.000.000$	

Die Tabelle zeigt die Bedeutung der farbigen Ringe auf Widerständen.

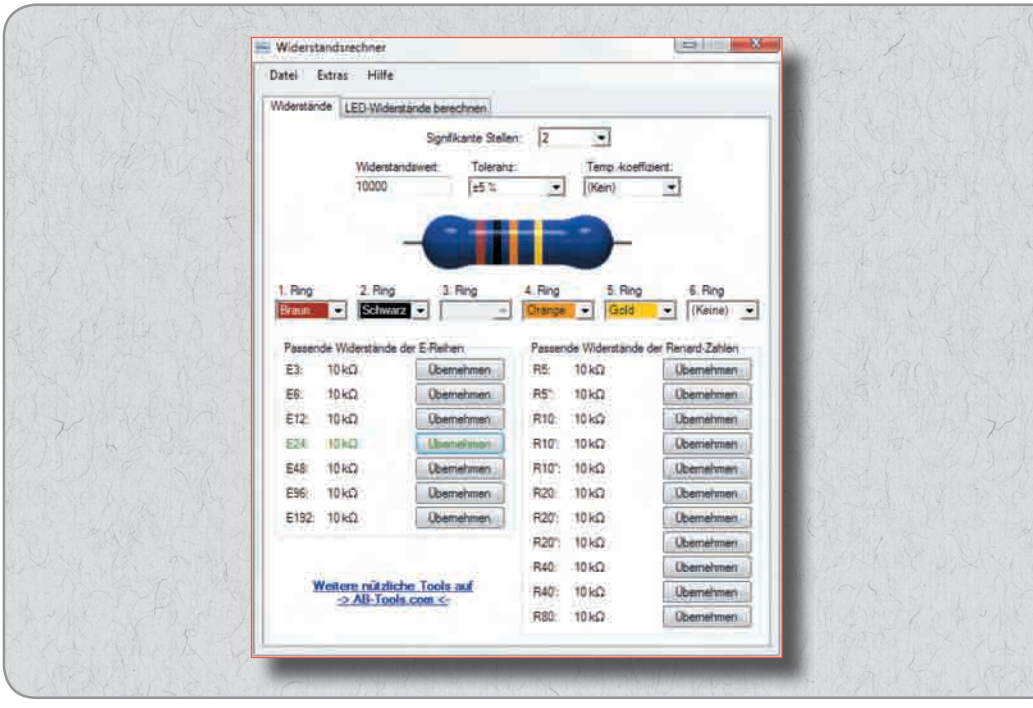
Im Maker Kit sind Widerstände mit vier verschiedenen Werten enthalten:

Wert	1. Ring (Zehner)	2. Ring (Einer)	3. Ring (Multiplikator)	4. Ring (Toleranz)	Verwendung
220 Ohm	Rot	Rot	Braun	Gold	Vorwiderstand für LEDs
560 Ohm	Grün	Blau	Braun	Gold	Vorwiderstand für Stromversorgung der LCD-Anzeige
10 kOhm	Braun	Schwarz	Orange	Gold	Pull-down-Widerstand für Taster
22 MOhm	Rot	Rot	Blau	Gold	Widerstand für Pikey Pikey

Farbcodes der Widerstände im Lernpaket.

Achten Sie besonders bei den 220-Ohm- und 22-MOhm-Widerständen genau auf die Farben. Diese sind leicht zu verwechseln.

Noch einfacher als mit diesen Tabellen ermitteln Sie den Widerstandswert aus einem Farbcode mit der Freeware *Widerstandsrechner* von `www.ab-tools.com/de/software/widerstandsrechner`. Umgekehrt können Sie sich damit auch den Farbcode zu einem beliebigen Widerstandswert anzeigen lassen.



Das Freewaretool Widerstandsrechner.

Potenziometer

Ein Potenziometer ist ein regelbarer Widerstand. Zwischen den äußeren Anschlüssen liegt ein fester Widerstandswert von in unserem Fall 15 kOhm an. Über den Drehregler lässt sich ein kleinerer Widerstand zwischen dem mittleren und einem der äußeren Anschlüsse einstellen.

Wir verwenden im Maker Kit das Potenziometer zur Kontrasteinstellung für das LC-Display sowie für analoge Eingaben.

LEDs

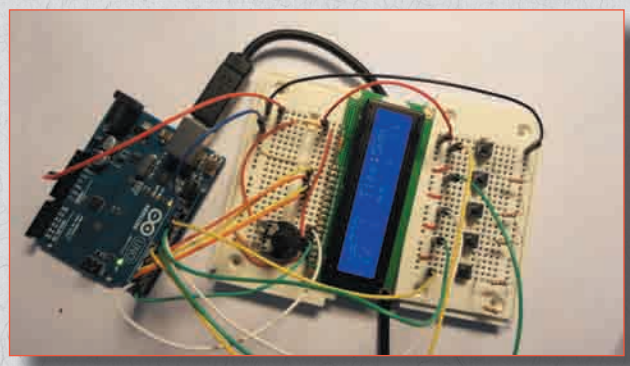
An die digitalen Pins können für Lichtsignale und Lichteffekte LEDs (LED = Light Emitting Diode, zu Deutsch Leuchtdiode) angeschlossen werden. Dabei muss zwischen dem verwendeten Pin und der Anode der LED ein 220-Ohm-Vorwiderstand (Rot-Rot-Braun) eingebaut werden, um den Durchflussstrom zu begrenzen und damit ein Durchbrennen der LED zu verhindern. Zusätzlich schützt der Vorwiderstand auch den digitalen Ausgang des Arduino, da die LED in Durchflussrichtung fast keinen Widerstand bietet und deshalb der Pin bei Verbindung mit Masse schnell überlastet werden könnte. Die Kathode der LED verbindet man mit der Masseleitung des Arduino.

LED in welcher Richtung anschließen?

Die beiden Anschlussdrähte einer LED sind unterschiedlich lang. Der längere ist der Pluspol, die Anode, der kürzere die Kathode. Einfach zu merken: Das Pluszeichen hat einen Strich mehr als das Minuszeichen und macht damit den Draht etwas länger. Außerdem sind die meisten LEDs auf der Minusseite abgeflacht, vergleichbar mit einem Minuszeichen. Leicht zu merken: Kathode = kurz = Kante!

LC-Display

Im Maker Kit ist ein zweizeiliges LC-Display enthalten, das wie die meisten derartigen Displays zum Quasi-Standard HD44780 kompatibel ist. Dabei handelt es sich um die Bezeichnung des in den Displays eingebauten Controllers, der Zeichen in ein bis vier Zeilen darstellt, ohne dass sich der Benutzer um die Ansteuerung der einzelnen Pixel zu kümmern braucht.



Zweizeiliges LC-Display auf einer Steckplatine mit zusätzlichen Steuerungstasten am Arduino.

Was braucht man noch?

In unserem Maker Kit sind bereits alle Bauteile enthalten, die Sie zum Aufbau der Experimente benötigen.

PC

Zur Programmierung des Arduino verwenden wir in den Experimenten einen PC mit Windows. Die Arduino-Software und auch die interaktive Programmiersprache S4A (Scratch für Arduino) werden außer für Windows auch für Linux, Mac OS und sogar für den Raspberry Pi angeboten.

USB-Kabel

Die Verbindung zwischen PC und Arduino erfolgt über ein USB-Kabel mit dem fast quadratischen Steckertyp B auf einer Seite. Sie brauchen sich nicht extra ein solches Kabel zu besorgen, fast alle modernen Drucker verwenden diesen Steckertyp – und während Sie mit dem Arduino experimentieren, werden Sie kaum gleichzeitig drucken.

Arduino noch nicht anschließen

Schließen Sie den Arduino nicht gleich an den PC an, sondern installieren Sie erst die Arduino-IDE. Andernfalls kann es später Schwierigkeiten mit der Treiberinstallation geben.

Im Gegensatz zu klassischen Mikrocontrollerplatinen benötigt der Arduino weder ein externes Programmiergerät noch ein teures FTDI-Kabel. Die Programmierung erfolgt über die USB-Schnittstelle. Es ist aber trotzdem ein optional verwendbarer ICSP-Anschluss auf dem Arduino vorhanden.

Netzteil (optional)

Solange der Arduino am PC angeschlossen ist, wird er auch über die USB-Schnittstelle mit Strom versorgt. Sobald ein Programm auf dem Arduino gespeichert ist, kann dieser in vielen Fällen auch ohne PC laufen. Dazu können Sie entweder ein USB-Steckernetzteil anschließen oder ein einfaches Netzgerät mit einem sogenannten Japan-Stecker, wie von diversen elektronischen Geräten bekannt. Das Netzteil sollte eine Spannung zwischen 7 V und 12 V liefern, der Pluspol muss innen am Stecker liegen. Ein Spannungsregler auf dem Arduino regelt die Stromversorgung dann automatisch.

Arduino-Sketches

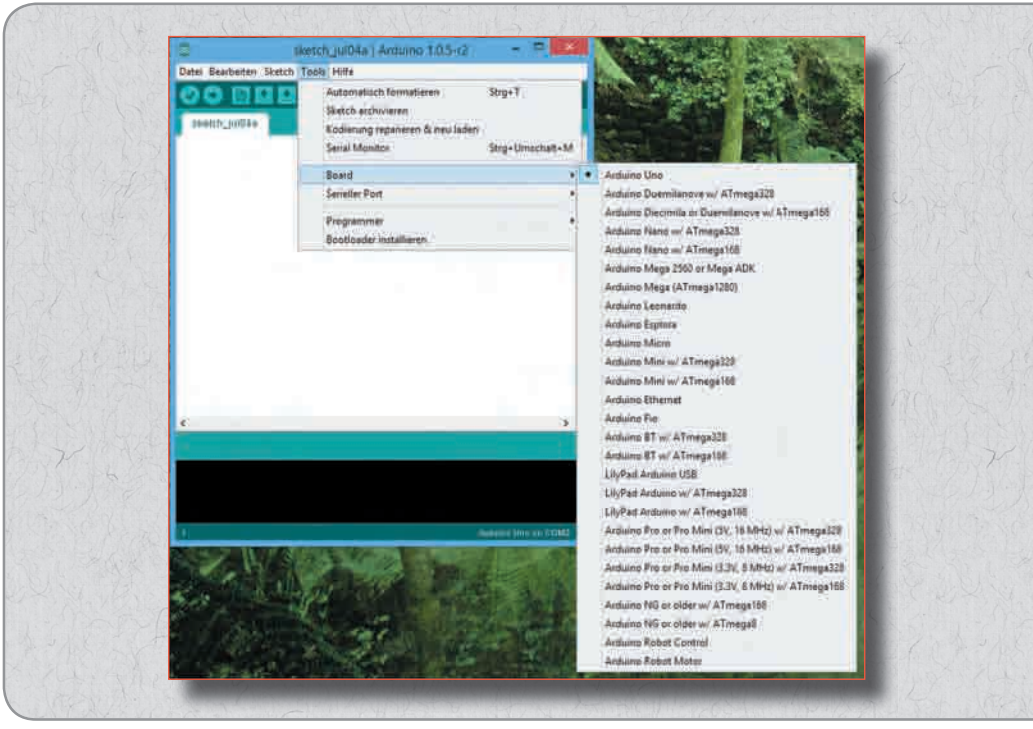
Arduino bezeichnet die Programme, die in der eigenen Programmiersprache Processing geschrieben werden, als Sketches. Diese haben die Dateiendung `.ino`. Auf unserer Website `www.buch.cd` finden Sie alle in diesem Maker Kit verwendeten Sketches zum Download.

Die Arduino-IDE

Für die Programmierung des Arduino liefert der Hersteller eine sehr übersichtliche Entwicklungsumgebung, in der man die Programme, die bei Arduino als Sketch bezeichnet werden, in einer C-ähnlichen Programmiersprache schreiben kann. Diese Arduino-IDE stellt auch die Verbindung zwischen PC und Arduino her.

Laden Sie sich den Windows Installer für die Arduino-IDE 1.0.5 bei `arduino.cc` unter *Downloads* herunter. Hier wird zusätzlich bereits die neue Arduino-IDE 1.5.x angeboten, die aber noch im Betastadium ist. Diese wird für die Platinen Arduino Yún und Arduino Due benötigt, für den Arduino UNO aus dem Maker Kit funktioniert die Version 1.0.5 einwandfrei.

Installieren Sie die Arduino-IDE, bevor der Arduino erstmals an den PC angeschlossen wird. Damit werden die notwendigen Treiber automatisch mit installiert, und der Arduino wird später automatisch erkannt. Andernfalls würde der Arduino als unbekanntes Gerät erkannt. Je nach Windows-Einstellung erscheint möglicherweise eine Warnung wegen eines unsignierten Treibers. Diese können Sie getrost ignorieren und den Treiber trotzdem installieren.



Die Arduino-IDE.

Starten Sie nach der Installation die Arduino-IDE und schließen Sie den Arduino an. Der Treiber wird jetzt automatisch installiert und simuliert über USB einen seriellen Port, mit dem der Arduino verbunden ist. Automatisch öffnet sich ein Fenster, in das Sie Ihren ersten Sketch eintippen können. Die Sketche sind standardmäßig nach dem aktuellen Datum benannt, Sie können aber beim Speichern jederzeit andere Namen vergeben.

Arduino mit PC verbinden

1. Wählen Sie, nachdem der Treiber installiert ist, als Erstes im Menü der Arduino-IDE *Tools/Serieller Port*. Hier wird in den meisten Fällen nur ein einziger serieller Port angezeigt. Setzen Sie dort das Häkchen.

2. Wählen Sie anschließend über den Menüpunkt *Tools/Board* den *Arduino Uno*, wenn dieser nicht bereits automatisch erkannt wurde.

3. Unten rechts wird die Verbindung angezeigt, und Sie können loslegen.

Arduino ausschalten

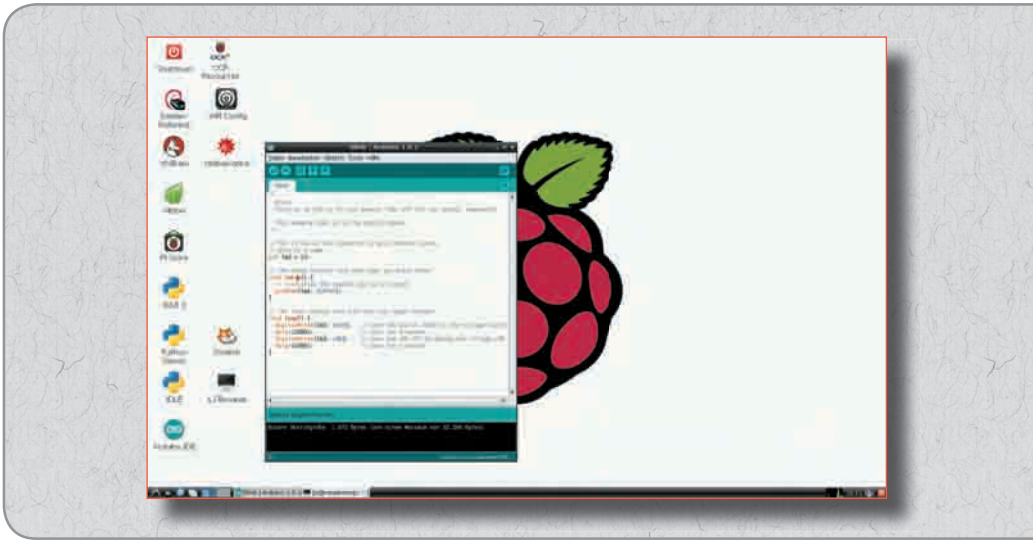
Der Arduino hat keinen Ausschalter. Sie brauchen einfach nur den Stecker zu ziehen, und er schaltet sich ab. Beim nächsten Einschalten startet automatisch der zuletzt gespeicherte Sketch. Das Gleiche passiert, wenn man die Reset-Taste drückt. Der Arduino lässt sich auch ohne PC betreiben, schließen Sie dazu ein USB-Netzteil oder ein Netzteil mit 7 V bis 12 V an der Stromversorgungsbuchse an.

Arduino am Raspberry Pi

Zur Programmierung des Arduino brauchen Sie nicht unbedingt einen Windows-PC – ein Raspberry Pi reicht auch. Um die Arduino-IDE auf dem Raspberry Pi zu nutzen, muss dieser mit der grafischen LXDE-Oberfläche von Raspbian laufen. Installieren Sie die Arduino-IDE in einem Terminalfenster mit:

```
sudo apt-get install arduino
```

Bei der Installation wird automatisch ein Menüpunkt im Startmenü unter *Entwicklung* angelegt. Schließen Sie den Arduino an einen USB-Anschluss des Raspberry Pi an und wählen Sie den seriellen Port über das *Tools*-Menü aus. Die seriellen Ports haben unter Linux andere Bezeichnungen als unter Windows, die Funktionsweise innerhalb der Arduino-IDE ist aber gleich.



Arduino-IDE auf dem Raspberry Pi.

3 S4A – SCRATCH FÜR ARDUINO

Scratch ist eine intuitive Programmierumgebung, mit der Kinder und Programmiereinsteiger schnell Ideen umsetzen können, ohne sich zuerst mit Programmiertheorie auseinandersetzen zu müssen. Scratch ist ein Projekt der Lifelong-Kindergarten-Group am Media-Lab des MIT (`scratch.mit.edu`). Passend für die Zielgruppe der Kinder und Jugendlichen eignet sich Scratch besonders für interaktive Animationen und Spiele. Eine grafische Oberfläche gibt bereits alle Grundlagen vor, sodass man sich um die Programmierung der Benutzeroberfläche des eigenen Programms keine Gedanken zu machen braucht. Die Scratch-Skripte werden nicht als Text geschrieben, sondern aus vorgefertigten Elementen zusammengeklickt.

Interaktion zwischen PC und Arduino

S4A (Scratch für Arduino) ist eine spezielle Version von Scratch, die einen Arduino steuert oder umgekehrt von diesem Eingaben übernimmt.

S4A interagiert in Echtzeit mit dem Arduino. Im Gegensatz zu klassischen Arduino-Sketches werden die Scratch-Programme nicht auf den Arduino hochgeladen und laufen dann eigenständig. Die Verbindung mit dem PC muss die ganze Zeit bestehen. Die Scratch-Oberfläche kann dafür auch jederzeit den Status der Arduino-Pins anzeigen.

1. Laden Sie sich bei `s4a.cat` das Installationsarchiv `vps34736.ovh.net/S4A/S4A15.zip` herunter, entpacken Sie es auf die Festplatte und starten Sie die Installation.

2. Laden Sie sich zusätzlich noch die S4A-Arduino-Firmware `vps34736.ovh.net/S4A/S4AFirmware15.ino` herunter. Dabei handelt es sich prinzipiell um einen normalen Arduino-Sketch, der die Kommunikation zwischen Arduino und PC regelt. Laden Sie diesen Sketch über die Arduino-IDE auf den Arduino. Wenn

Sie zwischendurch andere Arduino-Sketches nutzen, müssen Sie jedes Mal vor dem Start von S4A wieder die S4A-Arduino-Firmware installieren, da auf dem Arduino immer nur ein Sketch laufen kann.

3 Starten Sie jetzt S4A auf dem PC. Die LEDs *TX* und *RX* auf dem Arduino beginnen hektisch zu blinken, da S4A permanent Daten in beide Richtungen überträgt. Scratch startet mit einem viergeteilten Programmfenster:

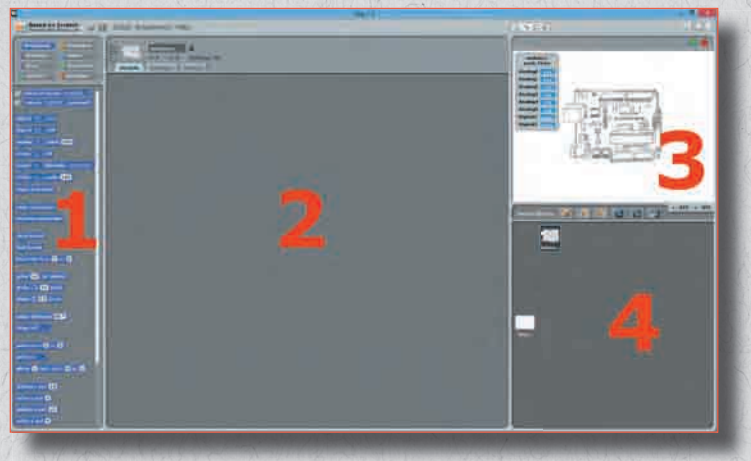


Bild 3.1: Scratch starten.

1 Links befindet sich die Blockpalette mit allen Elementen, aus denen ein Scratch-Skript zusammengesetzt werden kann. Da ausschließlich vordefinierte Blöcke verwendet werden, kann der Benutzer keine Syntaxfehler machen, die beim Einstieg in andere Programmiersprachen sehr ärgerlich und frustrierend sind.

2 Das mittlere Feld ist am Anfang noch leer. Hier entsteht später das Skript.

3 Rechts oben ist die sogenannte Bühne, die Oberfläche, auf der das Skript abläuft. Der Arduino, der dort zu sehen ist, stellt ein spezielles Objekt dar, das mit Scratch animiert werden kann. Hier tauchen in Scratch standardmäßig einfache grafische Symbole wie die typische Scratch-Katze auf. Wundern Sie sich nicht, dass

auf der Bühne ein Arduino Duemilanove dargestellt wird. S4A läuft genauso auch mit dem Arduino UNO.

4 Das Objektfenster unten rechts zeigt die im Skript verwendeten Objekte. Hier können Sie später auch selbst eigene Objekte gestalten oder aus Bibliotheken importieren.

S4A hat die Pins des Arduino für bestimmte Funktionen vorbelegt. Sie können also nicht ganz so frei wie in klassischen Arduino-Sketches verwendet werden, brauchen dafür aber nicht speziell initialisiert zu werden.

Funktion	Pins
Digitale Eingänge	2, 3
Digitale Ausgänge	10, 11, 12, 13
Analoge Eingänge	A0 ... A5
Analoge Ausgänge (PWM)	5, 6, 9
Ausgänge für Servomotoren	4, 7, 8

3.1 | Wechselblinklicht in Scratch

Das erste Skript `1ed02` setzt das Wechselblinklicht aus dem letzten Projekt in Scratch um. Damit haben Sie den direkten Vergleich zwischen den beiden Programmiersprachen. Der Hardwareaufbau ist identisch, es werden die gleichen Pins verwendet.

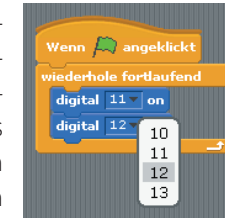
1 Klicken Sie in der Blockpalette oben auf das gelbe Symbol *Steuerung*. Damit werden die Blöcke zur Programmsteuerung angezeigt.

2 Ziehen Sie den abgebildeten Block aus der Blockpalette in das Skriptfenster. Auf der Scratch-Bühne ist oben rechts ein grünes Fähnchen. Dieses dient üblicherweise dazu, ein Programm zu starten. Das abgebildete Element bewirkt, dass die folgenden Skriptelemente ausgeführt werden, wenn der Benutzer auf das grüne Fähnchen klickt.

3 Ziehen Sie den Block *wiederhole fortlaufend* auf die Bühne und docken Sie ihn unten an den vorhandenen Block an. Der Block *wiederhole fortlaufend* startet eine Endlosschleife, die alle inner-

halb des Blocks befindlichen Blöcke abarbeitet. Diese Schleifen werden vom Skript selbst nie beendet, enden aber automatisch, wenn der Benutzer mit einem Klick auf das rote Stoppsymbol oben rechts auf der Scratch-Bühne das Skript beendet.

4 Innerhalb der Schleife soll eine LED ein- und eine andere ausgeschaltet werden. Nach Scratch-Logik handelt es sich dabei um Aktionen eines Objekts, des Arduino, die auf der Blockpalette *Bewegung* zu finden sind. Klicken Sie oben in der Blockpalette auf das blaue Symbol *Bewegung* und ziehen Sie den Block *digital … on* in das Skriptfenster. Docken Sie ihn innerhalb der Schleife an. Ändern Sie dann noch den vorgegebenen Wert für den digitalen Ausgang auf *11*. Dieser Block schaltet später die LED an Pin 11 ein.



5 Auf die gleiche Weise schalten Sie mit dem Block *digital … off* die LED an Pin 12 aus.

6 Mit dem Block *warte … Sek.* wartet das Skript eine kurze Zeit, bis die LED an Pin 12 ein und die LED an Pin 11 ausgeschaltet wird. Diesen Block finden Sie bei den gelben Blöcken auf der Palette *Steuerung*. Ändern Sie die eingetragene Zeit auf *0.1* Sekunden.

7 Um diese drei Blöcke anschließend zu wiederholen, brauchen Sie sie nicht noch einmal von der Palette zusammenzufügen. Klicken Sie mit der rechten Maustaste auf den ersten Block *digital 11 on* und wählen Sie im Kontextmenü *Duplizieren*. Damit erstellen Sie eine Kopie der drei Blöcke, die außerhalb der Endlosschleife zwischengelagert werden kann.

8 Ändern Sie die beiden Werte auf *digital 12 on* und *digital 11 off*, damit die beiden LEDs umgeschaltet werden, und ziehen Sie dann die kopierten Blöcke wie abgebildet ans Ende in die Endlosschleife.

9 Das Skript sollte jetzt wie abgebildet aussehen. Probieren Sie aus, ob es auch wie erwartet funktioniert. Klicken Sie dazu rechts oberhalb der Bühne auf das grüne Fähnchen. Die LEDs werden abwechselnd blinken. Mit einem Klick auf das rote Stoppsymbol halten Sie das Skript wieder an.

12 SPIELE MIT DEN FINGERN STEUERN

Ein Spiel einfach nur durch Berührung selbst gebauter Sensortasten steuern – mit unserem nächsten Projekt ist das möglich.

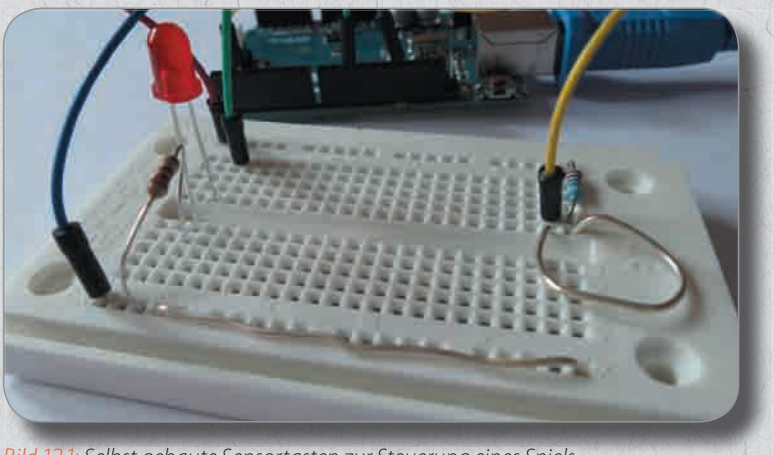


Bild 12.1: Selbst gebaute Sensortasten zur Steuerung eines Spiels.

Das Prinzip der Sensortasten ist einfach: Die verwendeten digitalen Eingänge sind über extrem hochohmige Widerstände (22 MOhm) mit +3,3 V oder +5 V verbunden, sodass ein schwaches, aber eindeutig als `HIGH` definiertes Signal anliegt. Ein Mensch, der nicht gerade frei in der Luft schwebt, ist meistens geerdet und liefert über die elektrisch leitfähige Haut einen `LOW`-Pegel. Berührt dieser Mensch jetzt einen Sensorkontakt, wird das schwache `HIGH`-Signal von dem deutlich stärkeren `LOW`-Pegel der Fingerkuppe überlagert und zieht den entsprechenden digitalen Eingang auf `LOW`-Pegel.

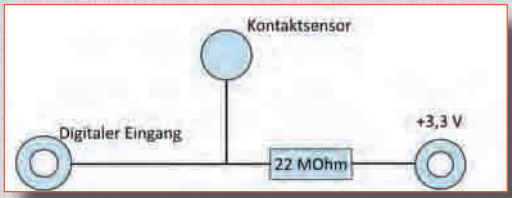


Bild 12.2: Schaltschema der Kontaktsensoren.

Das Spiel ist ein Tennisspiel für zwei Spieler, wie man es von alten Telespielen aus den 70er-Jahren kennt. Jeder Spieler bekommt auf einem Steckbrett eine kleine »Konsole« in die Hand mit einer Sensortaste und einer LED, die die Berührung des Sensors anzeigt.

Bauen Sie die abgebildete Schaltung auf zwei Steckplatinen auf. Dazu brauchen Sie neben isoliertem Schaltdraht noch blanken Draht, im Bild grau dargestellt. Entfernen Sie dazu mit einem scharfen Messer die Isolierung von einem Stück Schaltdraht komplett oder verwenden Sie einfachen Kupfer- oder Silberdraht, der sich in jedem Bastlerhaushalt finden lässt. Biegen Sie aus dem blanken Draht die Kontaktsensoren. Der lange Draht ganz unten quer ist mit 0-V-Masse verbunden. Durch Berühren dieses Drahts können Sie Ihre Hand bei Benutzung der Kontaktsensoren erden, wenn die Sensortaste bei Berührung nicht anspricht.

Benötigte Bauteile: 2x Steckplatine, 2x LED rot, 2x 220-Ohm-Widerstand (Rot-Rot-Braun), 2x 22-MOhm-Widerstand (Rot-Rot-Blau), 8x Verbindungskabel, 4x Draht blank (unterschiedliche Längen)

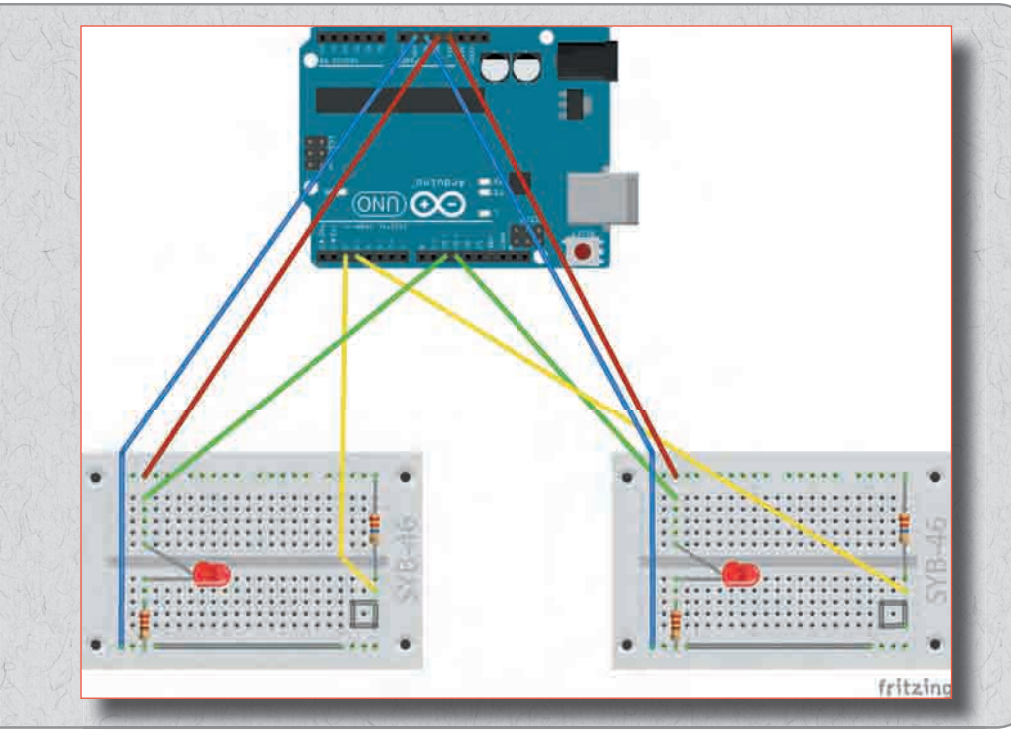


Bild 12.3: Tennisspiel mit Spielkonsolen für zwei Spieler – mit Kontaktsensoren (grau).

Die beiden Konsolen sind genau gleich aufgebaut. Der Arduino hat genügend GND-Pins, um beide Konsolen getrennt anzuschließen. Leider gibt es nur einen +5-V-Pin, sodass man entweder eine Verbindungsleitung zwischen beiden Konsolen oder ein Y-Kabel (mit drei Enden) benötigen würde. Die +3,3<V reichen aber in diesem Fall ebenfalls aus. Deshalb haben wir in der Schaltung eine Konsole an +5 V und die andere an +3,3 V angeschlossen, was problemlos funktioniert.

Statt mit den mitgelieferten Kabeln können Sie die Steckbretter auch über eigene längere Drähte anschließen, damit sich die Spieler in größerem Abstand zueinander aufhalten können.

12.1 | Das Scratch-Skript zum Tennisspiel

Um sofort loszuspielen, öffnen Sie in S4A das Skript `tennis`. Sie können das Spiel natürlich auch, um jeden Schritt nachzuvollziehen, anhand der Abbildungen selbst erstellen.

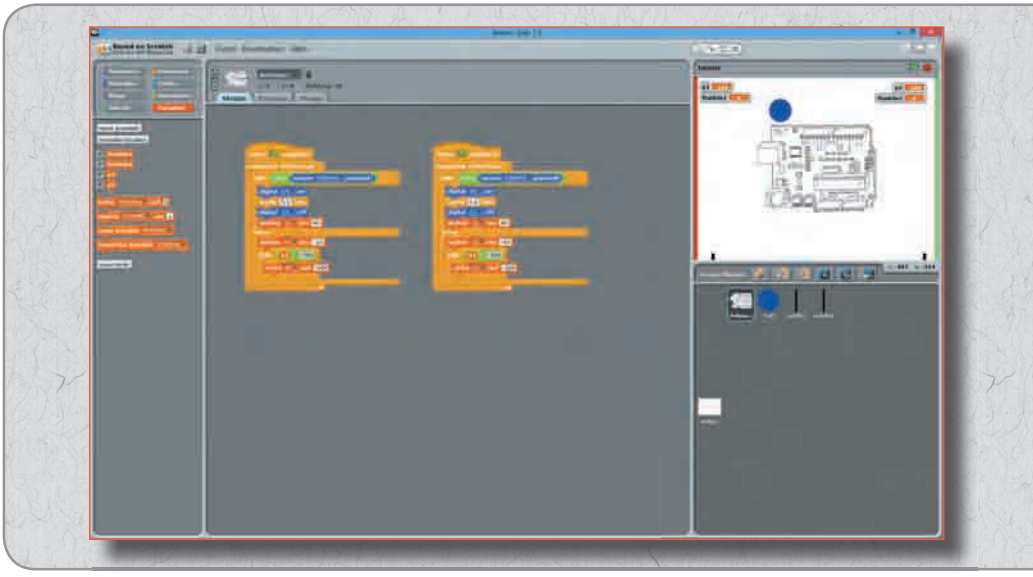


Bild 12.4: Das Tennisspiel in S4A.

Klicken Sie auf das grüne Fähnchen, startet das Spiel, und der Ball fliegt durch den Raum. Jeder Spieler kann sein Paddle durch Antippen der Sensortasten nach oben bewegen. Berührt man den Sensorkontakt eine Weile nicht, fällt das Paddle langsam wieder nach unten. Mit etwas Geschicklichkeit müssen die Spieler versuchen, mit ihrem Paddle den Ball immer wieder zurückzuschlagen, damit dieser nicht gegen die rote bzw. grüne Wand fliegt.

Berührt der Ball die eigene Wand, weil man ihm das Paddle nicht rechtzeitig in den Weg gesetzt hat, bekommt man einen Minuspunkt. Der Ball startet dann wieder an der Ausgangsposition. Durch Klicken auf das rote Stoppsymbol lässt sich das Spiel jederzeit anhalten. Wer weniger Minuspunkte hat, hat gewonnen.

12.1.1 | So funktioniert es

Im Spiel gibt es vier Objekte: den Arduino, der die Kontaktsensoren auswertet und die LEDs schaltet, den Ball, der das eigentliche Spiel steuert und der auch die Minuspunkte vergibt, wenn er an die farbigen Ränder stößt, sowie die Paddles für die beiden Spieler links und rechts.

Die beiden farbigen Streifen an den Rändern sind direkt auf die Bühne gemalt. Wenn Sie diese anklicken, können Sie auf der Registerkarte *Hintergründe* den Hintergrund mit dem in Scratch integrierten Malprogramm bearbeiten.

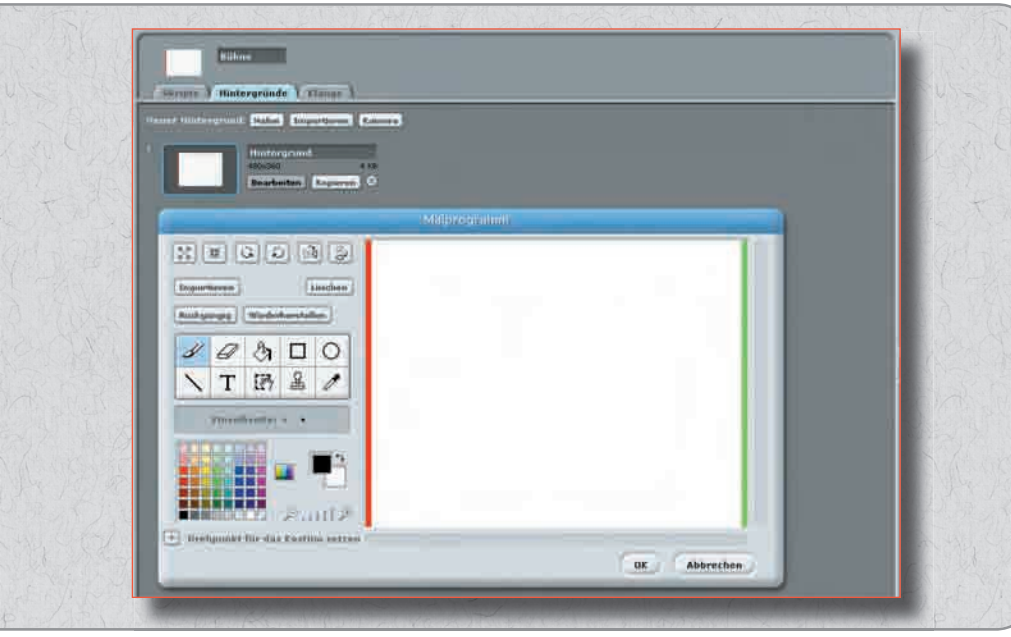


Bild 12.5: Der Hintergrund für das Tennisspiel.

12.1.2 | Der Ball

Der Ball wird durch vier Skripte gesteuert, die alle gleichzeitig laufen und gestartet werden, wenn der Benutzer auf das grüne Fähnchen klickt.

Bild 12.6: Die Skripte und die Variablen für den Ball.

Zunächst werden vier Variablen definiert, die alle auch auf der Bühne in Echtzeit angezeigt werden:

Punkte1	Minuspunkte für Spieler1 (links)
Punkte2	Minuspunkte für Spieler2 (rechts)
y1	y-Koordinate des Paddles von Spieler1 (links)
y2	y-Koordinate des Paddles von Spieler2 (rechts)

Wenn Sie das Skript selbst bauen, schalten Sie nach dem Anlegen der Variablen die Kontrollkästchen links auf der Blockpalette *Variablen* ein, um die Anzeigefelder auf der Bühne zu sehen. Die beiden Anzeigen für die Variablen des rechten Spielers schieben Sie nach rechts außen.

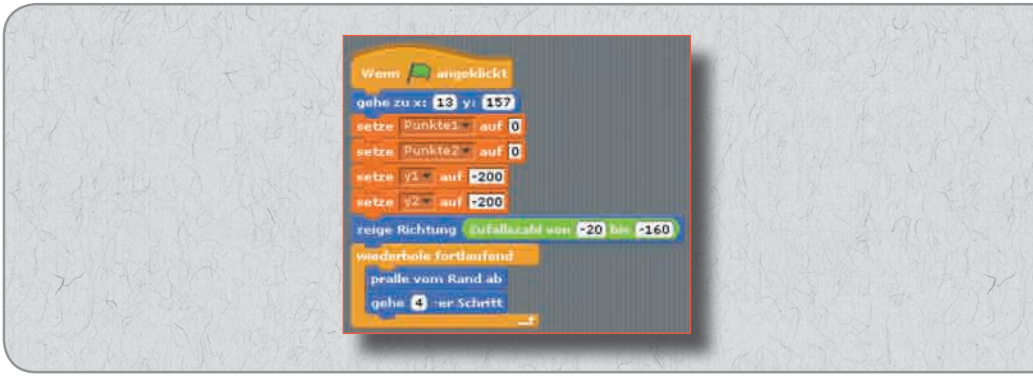


Bild 12.7: Hauptskriptblock für Initialisierung und Bewegung des Balls.

Das Hauptskript schafft beim Klick auf das grüne Fähnchen die Grundvoraussetzungen für das Spiel. Zuerst wird der Ball auf seine Ausgangsposition bei *x:13 y:157* gebracht. Anschließend werden die vier Variablen auf ihre Ausgangswerte gesetzt. Beide Spieler haben *0* Punkte, und die Paddles befinden sich bei der y-Koordinate *–200* am unteren Rand der Bühne.

Der Ball soll in einem zufälligen Winkel losfliegen. Dazu wird die Richtung auf einen zufälligen Wert zwischen *–20* und *–160* gesetzt.

Anschließend wird die Bewegung des Balls fortlaufend wiederholt. Er prallt vom Rand ab, sollte er ihn berühren. Andernfalls fliegt er einen 4er-Schritt in die eingestellte Richtung. Diese Bewegung wiederholt sich theoretisch endlos. Da aber drei weitere Skriptblöcke für den Ball beim Klicken auf das grüne Fähnchen gestartet werden, kann es auch noch zu anderen Bewegungen kommen.

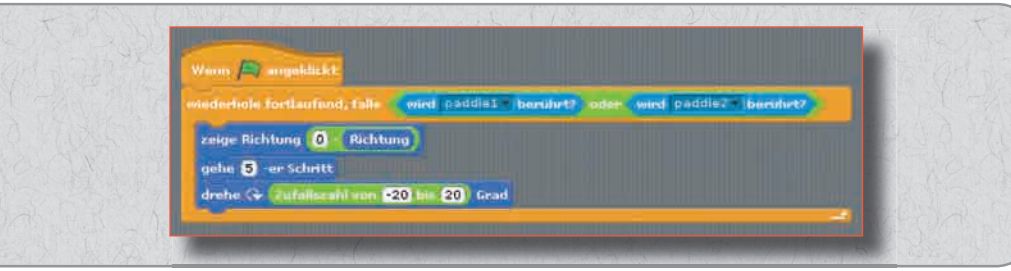


Bild 12.8: Der Ball ändert seine Bewegungsrichtung, wenn er eines der Paddles berührt.

Falls der Ball eines der beiden Paddles berührt, wird die Bewegungsrichtung ins Negative umgekehrt. Der Ball fliegt im gleichen Winkel nach unten weiter, in dem er von oben kam. Um die Bewegung etwas unvorhersehbarer zu gestalten, wird der Ball zunächst einen kleinen 5er-Schritt bewegt, damit danach das Paddle auf jeden Fall nicht mehr berührt wird. Anschließend wird die Flugrichtung gegenüber der bisherigen Richtung um einen zufälligen Wert zwischen *–20* und *20* Grad verändert.

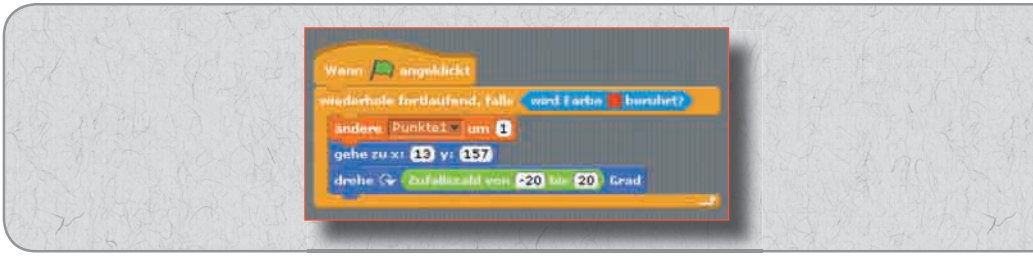


Bild 12.9: Dieser Skriptblock wird ausgeführt, wenn der Ball den linken roten Rand berührt.

Berührt der Ball nicht das Paddle, sondern den roten Balken am linken Rand, bekommt der linke Spieler einen Minuspunkt. Danach wird der Ball wieder auf seine Ausgangsposition gesetzt und um einen zufälligen Wert zwischen *–20* und *20* Grad gedreht, damit er nicht wieder exakt die gleiche Flugbahn nimmt, aber trotzdem ungefähr in die Richtung fliegt, in die er zuletzt geflogen ist.

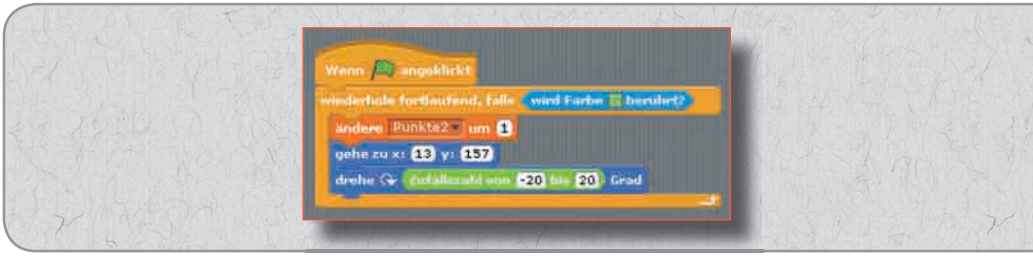


Bild 12.10: Dieser Skriptblock wird ausgeführt, wenn der Ball den rechten grünen Rand berührt.

Das Gleiche passiert auch, wenn der Ball den grünen Balken am rechten Rand berührt, nur mit dem Unterschied, dass diesmal der rechte Spieler den Minuspunkt bekommt.

12.1.3 | Der Arduino

Das Arduino-Objekt wertet die Berührungen der Sensorkontakte aus. Dabei ist zu beachten, dass, sobald ein Spieler einen Sensorkontakt berührt, der entsprechende digitale Eingang mit Masse verbunden, also auf `LOW` gesetzt wird. Ein nicht berührter Sensorkontakt hat, wie ein gedrückter Taster, den logischen Wert `HIGH`.

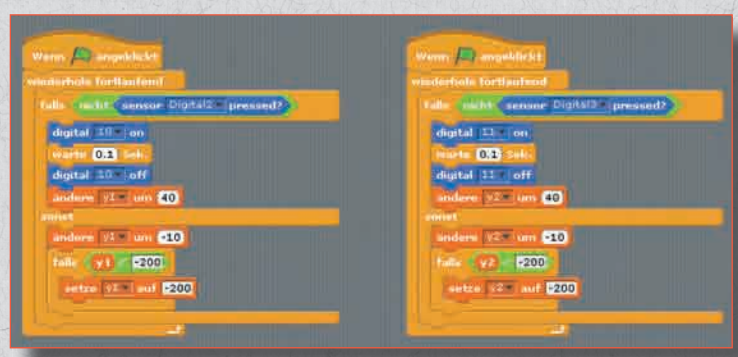


Bild 12.11: Dieser Skriptblock wird ausgeführt, wenn der Ball den linken roten Rand berührt.

Beim Klick auf das grüne Fähnchen werden zwei Skriptblöcke für das Arduino-Objekt gestartet. Diese bestehen aus Endlosschleifen. In der Endlosschleife liegt ein *falls/sonst*-Konstrukt, das jedes Mal eine von zwei möglichen Aktionen auslöst:

- Berührt der Spieler den an Pin *Digital2* angeschlossenen Sensorkontakt, wird dieser auf `LOW` gesetzt, verhält sich also wie ein nicht gedrückter Taster. Durch den Operator *nicht* wird der Logikzustand umgekehrt, also auf *wahr* gesetzt, und die folgenden Anweisungen werden ausgeführt. Die an Pin *Digital10* angeschlossene LED leuchtet für 0,1 Sekunden. Die Variable *y1* wird um 40 erhöht, damit schiebt sich das linke Paddle um 40 Einheiten nach oben.

- Solange der Spieler den Sensorkontakt nicht berührt, ist – durch den Trick mit dem *nicht*-Operator – die Abfragebedingung nicht erfüllt, und die unter *sonst* aufgelisteten Anweisungen werden ausgeführt. Die Variable *y1* wird um 10 Einheiten verringert, das linke Paddle rutscht um 10 Einheiten nach unten. Damit es nicht endlos nach unten aus dem Bild herausfällt, wenn der Spieler den Sensorkontakt länger nicht berührt, wird *y1* automatisch auf die Grundstellung *–200* gesetzt, wenn diese Variable einen geringeren Wert erhalten sollte.

Auf die gleiche Weise steuert der andere Skriptblock über den Sensorkontakt *Digital3* die LED an Pin *Digital11* und mithilfe der Variablen *y2* das rechte Paddle.

Um das Spielfeld besser zu sehen, ist das Sensorboard, das den Zustand der Eingänge des Arduino anzeigt, ausgeblendet. Dies erreichen Sie mit einem Rechtsklick auf den Arduino in der Objektpalette unten rechts. Im Kontextmenü blenden Sie das Sensorboard mit *hide sensor board* aus und mit *show sensor board* wieder ein.

12.1.4 | Die Paddles

Die Paddles nehmen immer die in der Variablen *y1* für das linke bzw. *y2* für das rechte Paddle angegebene y-Koordinate an. Mehr Programmlogik ist hier nicht erforderlich.

Bild 12.12: Skriptblöcke für die Paddles.

Kaffeelöffel oder Bananen als Sensoren

Nicht immer reicht die Erdung über den Fußboden aus, um die Sensorkontakte zu nutzen. Teppich oder Kunststofffußboden und auch manche Schuhsohlen isolieren so gut, dass man ohne Berühren des Massedrahts keinen stabilen `LOW`-Pegel an den Fingern hat. Barfuß auf Steinfußboden oder in feuchtem Gras macht keine Probleme.

Statt der Kontaktsensoren aus blankem Draht können Sie auch Kaffeelöffel, Bananen oder andere leitfähige Gegenstände mit Drähten an die entsprechenden Pins anschließen und in die Hand nehmen, um die Kontakte auszulösen. Damit sind Sie nicht mehr auf den Aufbau auf einer Steckplatine beschränkt.

20 ARDUINO VOM TABLET ODER SMARTPHONE PROGRAMMIEREN

Haben Sie ein Arduino-Projekt irgendwo außerhalb Ihres Arbeitszimmers laufen, brauchen Sie nicht unbedingt ein Notebook dorthin zu schleppen, um etwas daran zu ändern oder einen neuen Sketch hochzuladen. Zwei Apps für Android stellen die Verbindung zum Arduino mit einem Tablet oder Smartphone her, wobei Smartphone-Bildschirme für die sinnvolle Nutzung zu klein sind.

20.1 | ArduinoDroid

ArduinoDroid ist eine komplette Umsetzung der Arduino-IDE mit Editor, Compiler und einer Funktion zum Hochladen der Sketche auf den Arduino. Die App *ArduinoDroid* ist 210 MB groß und muss im Gerätespeicher und nicht auf der Speicherkarte installiert werden.

Im Editor gibt es noch einen praktischen Navigator, der alle Variablen und Konstanten im Sketch anzeigt und schnell an die Stelle springt, an der die jeweilige Variable definiert wurde. Im Gegensatz zur Original-Arduino-IDE müssen die Sketche hier in zwei Schritten erst kompiliert und dann hochgeladen werden. Die Sketche können sich lokal auf dem Tablet befinden, die App hat aber auch Zugriff auf die Dropbox, was die Übertragung der Sketche vom PC deutlich vereinfacht.

Bild 20.1: ArduinoDroid – Arduino-IDE für Android-Tablets.

Die Verbindung zwischen Arduino und Tablet erfolgt mit einem USB-Hostadapterkabel. Das Tablet muss dazu den USB-Hostmodus unterstützen, was bei den meisten aktuellen Geräten der Fall ist.

Bild 20.2: USB-Hostadapterkabel.

20.2 | ArduinoCommander

Mit dem ArduinoCommander lässt sich ein Arduino über ein Tablet oder Smartphone interaktiv steuern. Dabei können Pins als Ausgang geschaltet oder Eingangswerte abgefragt werden.

Die Verbindung zwischen Tablet und Arduino kann auch hier über ein USB-Hostadapterkabel erfolgen. Zusätzlich werden Bluetooth- und Ethernetshields auf dem Arduino zur Verbindung unterstützt.

Bild 20.3: ArduinoCommander auf dem Tablet.

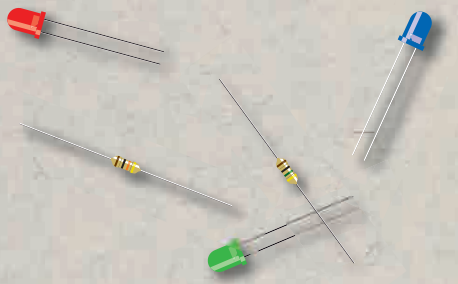
Die App bietet auch die Möglichkeit, Sketche auf den Arduino hochzuladen. Davon muss man als Erstes einmal Gebrauch machen und die mitgelieferte Datei `StandardFirmata` auf den Arduino übertragen. Firmata (`www.firmata.org`) ist ein Arduino-Sketch, der die interaktive Kommunikation mit dem Arduino über eine USB-Verbindung ermöglicht. Verschiedene Programme nutzen Firmata, um Sensoren des Arduino interaktiv abzufragen oder angeschlossene Geräte vom PC zu schalten.

Weitere Android-Apps

Unter dem Suchbegriff *Arduino* finden Sie noch diverse weitere Apps im Google Play Store. Die meisten dieser Apps haben jedoch die einzige Funktion, über angezeigte Werbung dem Entwickler Geld zu bringen. Sie enthalten häufig nur Offlineversionen der Arduino-Website, und das auch noch auf einem alten Stand.

TURN ON YOUR CREATIVITY

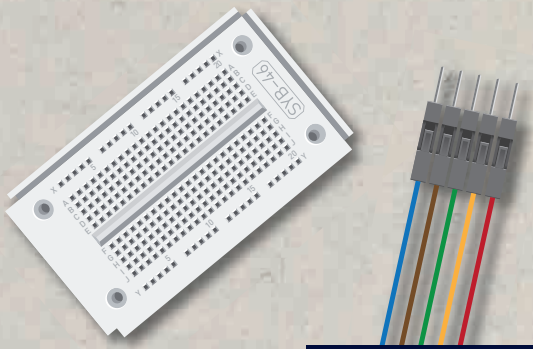
FRANZIS MAKER KIT +ARDUINO™



Haben Sie schon viele Arduino™-Projekte umgesetzt? Egal ob Ja oder Nein, hier sind Sie richtig. Mit Scratch für Arduino™ müssen Sie sich nicht mehr mit der Arduino™-IDE beschäftigen, sondern können den Sketch grafisch erstellen. Das Paket enthält alles, was Sie brauchen, um noch mehr aus dem Arduino™ herauszuholen: 55 Bauteile (inkl. original Arduino™ UNO) und ein 160-seitiges Handbuch: Willkommen bei den Makern!

PROJEKTE:

- LED-Blinklicht
- Wechselblinklicht
- S4A – Scratch für Arduino™
- Fußgängerampel mit Taster
- Fußgängerampel mit Scratch
- Spielwürfel mit LEDs
- LED per Pulsweitenmodulation dimmen
- Analoge Pegelanzeige mit LEDs
- Pong-Spiel mit Arduino™-Tasten steuern
- Scratch mit analoger Eingabe steuern
- Scratch mit dem Smartphone steuern
- Spiele mit den Fingern steuern
- Texte auf dem LCD darstellen
- Uhr auf dem LCD
- Zahlenspiel mit Tasten und LCD
- Eigene Zeichen für das LCD erstellen
- Umlaute auf dem LCD
- Menüsteuerung auf dem LCD
- LCD-Statusanzeige für Windows-PCs
- Arduino™ vom Tablet oder Smartphone programmieren



Für Kinder unter 14 Jahren nicht geeignet.

Zusätzlich erforderlich: USB-Kabel

ISBN 978-3-645-65219-3

FRANZIS