



Manuel Di Cerbo
Andreas Rudolf

Android mit Arduino™ Due

- Kommunikation zwischen Android-Geräten und Arduino™ Due
- Steuern Sie Ihren Arduino™ Due mit einem Android-Gerät
- Praxisbeispiele zeigen, wie es geht

Manuel Di Cerbo / Andreas Rudolf
Android mit Arduino™ Due

Manuel Di Cerbo
Andreas Rudolf

Android mit Arduino™ Due

- Kommunikation zwischen Android-Geräten und Arduino™
- Steuern Sie Ihren Arduino™ mit einem Android-Gerät
- Praxisbeispiele zeigen, wie es geht

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.

Alle Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, dass sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung etwaiger Fehler sind Verlag und Autor jederzeit dankbar. Internetadressen oder Versionsnummern stellen den bei Redaktionsschluss verfügbaren Informationsstand dar. Verlag und Autor übernehmen keinerlei Verantwortung oder Haftung für Veränderungen, die sich aus nicht von ihnen zu vertretenden Umständen ergeben. Evtl. beigefügte oder zum Download angebotene Dateien und Informationen dienen ausschließlich der nicht gewerblichen Nutzung. Eine gewerbliche Nutzung ist nur mit Zustimmung des Lizenzinhabers möglich.

Arduino™ ist ein eingetragenes Markenzeichen von Arduino LLC und den damit verbundenen Firmen.

© 2013 Franzis Verlag GmbH, 85540 Haar bei München

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträgern oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt.

Die meisten Produktbezeichnungen von Hard- und Software sowie Firmennamen und Firmenlogos, die in diesem Werk genannt werden, sind in der Regel gleichzeitig auch eingetragene Warenzeichen und sollten als solche betrachtet werden. Der Verlag folgt bei den Produktbezeichnungen im Wesentlichen den Schreibweisen der Hersteller.

Satz: DTP-Satz A. Kugge, München

art & design: www.ideehoch2.de

Druck: C.H. Beck, Nördlingen

Printed in Germany

ISBN 978-3-645-60205-1

Inhaltsverzeichnis

1	Einleitung	7
1.1	Android	7
1.2	Arduino	9
2	Arduino	13
2.1	Installieren der Arduino IDE	13
2.2	Ein erster Sketch	14
2.3	Änderungen am Blink-Sketch	19
2.4	Die Arduino-Programmiersprache	19
2.5	Serielle Verbindung: Host-Computer und Arduino	23
2.6	LED-Intensität steuern über serielle Konsole	29
2.7	LED-Intensität steuern über serielle Konsole und Android Accessory Mode	31
3	Android	33
3.1	Betriebssystem Android	33
3.1.1	Kernel	34
3.1.2	Native Daemons	34
3.1.3	Libraries	35
3.1.4	Command Line Tools	35
3.1.5	System Services.....	35
3.1.6	Applikationen, Provider	36
3.2	Development-Tools	36
3.3	Android-Applikationen	36
3.4	Android Software Development Kit und Eclipse	40
3.5	Hello World in Android	43
3.6	Eclipse-Tricks	59
3.7	Java für Android schreiben	61
3.8	Pattern für Multithreading	61
4	Android und USB	69
4.1	Kommunikation: Arduino als Accessory	70
4.1.1	Hello World mit Android Accessory	78

4.2	Kommunikation: Android als USB-Host.....	94
4.2.1	Funktionsweise der USB-Host-API.....	95
4.2.2	Hello World mit Android-USB-Host.....	100
5	Drucktaster auslesen mit Android und Arduino	111
5.1	Hardware	112
5.2	Arduino-Sketch	113
5.3	Android-Beispiel mit Accessory-API.....	118
5.4	Android-Beispiel mit USB-Host-API	125
6	RGB-LED mit Android ansteuern	133
6.1	Hardware	135
6.2	Arduino-Sketch	137
6.3	Android-Beispiel mit Accessory-API.....	140
6.4	Android-Beispiel mit USB-Host-API	149
7	Rotary Encoder (Drehregler) mit Android auslesen	155
7.1	Hardware	155
7.2	Arduino-Sketch	158
7.3	Android-Beispiel mit Accessory-API.....	160
7.4	Android-Beispiel mit USB-Host-API	165
8	Servo mit Android ansteuern	173
8.1	Hardware	173
8.2	Arduino-Sketch	175
8.3	Android-Beispiel mit Accessory-API & USB-Host-API	176
	Stichwortverzeichnis	177

1 Einleitung

1.1 Android

Android ist ein Betriebssystem, das besonders im Smartphone- und Tablet-Bereich eingesetzt wird. Entwickelt wird es hauptsächlich von der sogenannten *Open Handset Alliance*, die von Google angeführt wird.

Im Kern von Android befindet sich der Linux-Kernel, der gänzlich unter der Open-Source-Lizenz *General Public License* (GPL) entwickelt wird. Der Linux-Kernel ist ein Stück Software, um unter anderem Hardware anzusteuern und Speicher zu verwalten. Außerdem werden Prozesse und Rechte zur Verfügung gestellt. Generell kann der Linux-Kernel als »Fundament« des Betriebssystems aufgefasst werden. Bis vor Kurzem wurde Linux vor allem in Server- und Embedded-Systemen verwendet. Dies änderte sich aber schlagartig in den letzten Jahren mit der rasanten Verbreitung von Android.

Linux

Linux ist ein Betriebssystemkern, der vom Hacker und Bastler Linus Torvalds in seiner Studienzeit entwickelt wurde. Mit der Lizenzierung unter GPL ermöglichte er die Verbreitung seines Systems ohne Restriktionen, weil er Entwicklern am System möglichst viel Freiraum geben wollte. Heute verbessern tausende Entwickler den Linux-Source-Code. Der »Vanilla-Kernel«, ein Referenz-Kernel von Linux, wird in regelmäßigen Abständen von Linus Torvalds veröffentlicht. Auch führende Technologiekonzerne wie Intel, ARM, Microsoft und IBM arbeiten am Linux-Kernel und tragen mit »Patches« zu dessen Weiterentwicklung bei.

Wie der Kernel ist der Android-Source-Code auch als »Open Source« lizenziert. Allerdings wird für die Bestandteile von Android, die außerhalb des Kernels liegen, eine weniger restriktive Lizenz (zumeist Apache oder LGPL) verwendet. Dies erlaubt es den Hardware-Vertreibern, Modifikationen vorzunehmen, ohne dazu verpflichtet zu sein, den dazugehörigen Source-Code zu offenbaren. Primär hat dies zum Ziel, den Wettbewerb zwischen den einzelnen Vertreibern anzukurbeln und die Plattform wirtschaftlich für Hardware-Hersteller (meist auch Mitglieder der *Open Handset Alliance*) interessant zu machen.

GPL

GPL ist eine »Copyleft«-Lizenz. Bei der Erweiterung von GPL-lizenziertem Source-Code muss dieser mit der gleichen GPL-Lizenz weitergegeben werden. Dies führt dazu, dass alle Änderungen, die am Source-Code vorgenommen werden, dem originalen Verfasser mitgeteilt werden müssen. Bei einer Apache-Lizenz ist das nicht nötig, allerdings muss bei Weiterentwicklungen eine Notiz vorhanden sein, dass der Source-Code auf einem Apache-lizenzierten Ausgangsprojekt beruht.

Das Hauptinteresse von Google ist es, die Android-Plattform zu verbreiten, sodass Endbenutzer in Zukunft möglichst viele Dienstleistungen über den Suchmaschinen Giganten in Anspruch nehmen. Zusätzlich bemüht sich Google, ein »Entwickler-Ökosystem« zu pflegen, das als zentralen Bestandteil den *Google Play Store* hat. Entwickler sollen in der Lage sein, gewinnbringend Applikationen – die bekannten Android-Apps – zu vermarkten, und Google ist dabei am Erfolg beteiligt (30 Prozent des Preises einer App gehen beim Kauf an Google).

Die »Google Play«-App selbst ist – wie auch die Apps »GMail« und »Google Maps« – nicht Open Source und wird dem Kunden nur zur Verfügung gestellt, wenn die Hardware und die Android-Implementierung den Standards der *Open Handset Alliance* entsprechen. Dies erfolgt aus Gründen der Qualitätssicherung.

Android ist für Entwickler interessant, da es statt klassischer Softwareentwicklung eine nahezu geniale Einbettung von Applikationen in das Betriebssystem erlaubt. Mit akribisch genauer Dokumentation und guten Tutorials für das *Android Software Development Kit* (Android SDK) motiviert Google Entwickler, das Betriebssystem als Basis für ihre nächste große App zu verwenden.

Android SDK

Das SDK ist zentraler Bestandteil der Android-Philosophie. Auch mitgelieferte Apps wie »Telefon«, »E-Mail« und »Browser« basieren auf dem SDK. Regelmäßig veröffentlicht Google neue Versionen des SDKs, die es den Entwicklern erlauben, neue Features in ihre Apps einzubauen.

Als Programmiersprache für Android-Apps wird überwiegend *Java* verwendet. Mit der Möglichkeit, nativen Code in *C/C++* zu schreiben und diesen via *Java Native Interface* (JNI) einzubinden, existiert zusätzlich die Option, zeitkritische Komponenten einer App auszulagern. In diesem Buch werden wir uns allerdings auf die klassische Android-Entwicklung mit *Java* konzentrieren.

Der Open-Source-Charakter des Betriebssystems zieht sehr viele Entwickler an. Allein schon die Möglichkeit, in den Source-Code des Betriebssystems Einblick zu nehmen, um Software-Patterns zu lernen und Bestandteile eines modernen Systems zu erkunden,

begeistert viele Software-Entwickler. Auch der Bestandteil Linux ist ein Faktor, warum Bastler und Hobbyisten gerne mit Android arbeiten. Denn für Linux gibt es Hardware-Treiber wie Sand am Meer – und alle sind frei verfügbar.

1.2 Arduino

Die Marke *Arduino* hat vieles gemeinsam mit Android. Arduino ist eine Entwicklerplattform für Hardware. Neben den Hardware-Boards, die mit leistungsfähigen Mikrocontrollern ausgestattet sind, ist auch die Softwareplattform *Arduino IDE* (IDE = Integrated Development Environment) als Entwicklungsumgebung ein wichtiger Teil des Projekts. Beide Bestandteile, Hardware und Software, sind als Open Source verfügbar. Der Name *Arduino* ist allerdings als Marke eingetragen und geschützt.

Arduino-Boards

Originale Arduino-Boards werden in Italien hergestellt und weltweit vertrieben. Die Erfinder von Arduino arbeiten konstant an Verbesserungen bestehender Boards und an neuen Shields, die es ermöglichen, die Arduinos mit Peripherie zu erweitern.

Während der letzten zwei Jahre erlebte die Hardware-Branche mit Arduino etwas Unvorstellbares. Open-Source-Hardware, die jeder kopieren und vertreiben kann, bricht als Geschäftskonzept alle Regeln der Betriebswirtschaft. Mit einem Mix aus Markenstärke, durchdachtem Vertrieb und Weiterentwicklung der Arduino-Plattform schaffte es das Produkt in praktisch jedes Zimmer und jede Garage der passionierten Hobbytüftler. Wenn man heute einen Arduino kauft, dann kauft man Qualität sowie Kompatibilität, und dafür steht die Marke *Arduino*.

Für dieses Buch werden wir uns mit dem *Arduino Due* befassen. Er ist ein Entwickler-board, das mit einem *Cortex-M3* als Mikrocontroller bestückt ist und Aus- und Eingänge für Peripherie besitzt, z. B. serielle Schnittstelle, SPI, D/A-Wandler, A/D-Wandler und CAN.

Die Kommunikation mit Android ist ohne Zukauf von Modulen über USB möglich. Dies kann beim Arduino Due auf zwei Arten geschehen:

1. Arduino als USB-Device

Der Arduino wird über den USB-Seriell-Wandler (USB-Serial-Konverter) von Android angesprochen, dazu übernimmt Android die Rolle des USB-Hosts; je nach Gerät erfolgt dies mit einem USB-OTG-Konverter (USB On-The-Go). Gewisse Tablets verfügen über einen herkömmlichen USB-A-Port, wie er vom Desktop-PC bekannt ist. Der Arduino wird dabei vom Android-Gerät mit Strom versorgt und muss nicht extern gespeist werden.

2. Arduino als USB-Host

Der Arduino übernimmt die Rolle des USB-Hosts und versorgt das Android-Gerät mit Energie. Hierzu wird auf der Seite von Android die *Android Accessory API* verwendet und der Arduino als sogenanntes »Accessory« betrieben.

Auf beide Varianten wird in den kommenden Teilen des Buchs näher eingegangen.

Der Arduino kann so die Schnittstelle zur »realen Welt« werden und externe Hardware ansteuern. Vom Toaster-Regulator bis zur Rasenmäher-Steuerung sollte so alles möglich sein.

Tipp

Auf der Webseite von Arduino werden regelmäßig die neuesten Erfindungen präsentiert: <http://arduino.cc/en/Tutorial/HomePage>. Suchen Sie hier z. B. unter dem Punkt *Examples*.

In Verbindung mit einem Arduino kann ein Android-Tablet oder ein Smartphone also in eine »Controller-Steuerung« verwandelt werden. Im Speziellen können dazu die Vorteile des Android-Betriebssystems ausgenutzt werden, z. B. Wi-Fi und Bluetooth oder Beschleunigungssensoren und Touchscreen.

Bluetooth und Wi-Fi

Natürlich könnte man auch den Arduino mit einen Bluetooth- oder Wi-Fi-Modul ausstatten. Allerdings ist es wesentlich einfacher, diese Bestandteile dem schon konfigurierten Smartphone zu übergeben und sich nicht um das Debuggen dieser Kommunikationsprotokolle auf niedriger Ebene kümmern zu müssen. Gerade wenn es darum geht, einen Netzwerk-Stack zu implementieren, greift man gerne auf Embedded-Systeme zurück, welche bereits Kernel-Treiber zur Verfügung haben.

Der *Universal Serial Bus (USB)* ist zentraler Bestandteil der Kommunikation zwischen Android und Arduino. In diesem Buch werden die zwei verschiedenen Varianten *Accessory API* und *USB Serial* genauer unter die Lupe genommen. Die Variante *USB Serial* ermöglicht übrigens auch die Kommunikation mit anderen Boards, die eine serielle Schnittstelle haben, wie z. B. dem *Arduino Uno*.

Seit Sommer 2011 und Android 3.1 (*Honeycomb*) ist es möglich, über das Android SDK via USB-Host mit Peripherie zu kommunizieren. Dies ermöglicht den Anschluss von allerlei Hardware an Android-Geräte. Dazu gehören z. B. Maus oder Tastatur, das kann aber auch bis hin zu USB-Spielzeug wie Mini-Raketenwerfern gehen. Von den Autoren wird beispielsweise das Open-Source-USB-Oszilloskop »OsciPrime« entwickelt, welches via USB-High-Speed Gebrauch von der neuen *USB-Host-API* macht. Damit wird das Android-Gerät in ein Oszilloskop inklusive Multitouch-Funktionalität verwandelt.

Viele Arduino-Boards besitzen einen USB-Serial-Konverter. Das ist ein Mikrocontroller, der die Aufgabe hat, USB in »klassische« serielle UART-Kommunikation zu verwandeln. Diese serielle Schnittstelle wird dazu verwendet, den Haupt-Mikrocontroller auf dem Arduino-Board zu programmieren, oder während der Laufzeit mit dem Haupt-Mikrocontroller zu kommunizieren. Die serielle Schnittstelle (RS 232) stellt eines der meistverwendeten Protokolle in der Welt der Mikrocontroller dar. Ihre Einfachheit und Universalität sind Kernfaktoren für die heutige Verbreitung. Da USB wesentlich komplexer ist und spezielle Hardware bei Mikrocontrollern voraussetzt, ist bei vielen Controllern auch heute noch die klassische serielle Schnittstelle anzutreffen.

USB hat einen *Host*, der Ursprung aller Kommunikation ist, und bis zu 127 *Devices*. Dabei diktiert der Host, welches Device gerade etwas senden darf. USB erlaubt den Bezug von maximal 500 mA und läuft bei einer Spannung von 5 V. Maximal werden bei USB-High-Speed Datenraten von 480 Mbit/s erreicht. Dies sind allerdings nur theoretische Werte – in der Praxis ist, je nach Host-Controller, nur etwa die Hälfte erreichbar.

USB-Devices haben eine hierarchische Struktur. So besitzt ein USB-Device verschiedene »Configurations«, welche wiederum verschiedene »Interfaces« zur Verfügung stellen können. Jedes Interface enthält eine gewisse Anzahl »Endpoints«, die jeweils die Richtung »in« (vom Device zum Host) oder »out« (vom Host zum Device) vorschreiben. Die Richtungen *in* und *out* sind also mit Bezug auf den USB-Host definiert. Ausnahme dabei ist der Endpoint »0«, welcher bidirektional betrieben werden kann. Den Endpoints ist jeweils eine Adresse zugeordnet, welche vom Host zum Absenden oder Empfangen von Daten verwendet wird. So kann der Host einen Endpoint eines Device adressieren, welcher dem Interface der aktuellen Konfiguration zugewiesen ist.

Der USB-Standard definiert Geräteklassen. Darunter fallen z. B. Audio, Drucker oder Human Interface Device (HID; wie Maus und Tastatur). Eine Geräteklasse ist auch die »serielle« CDC-Klasse, welche eine virtuelle serielle Schnittstelle emuliert. Das USB-Device kommuniziert jeweils die Geräteklasse, und der USB-Host lädt daraufhin den entsprechenden Treiber. So wird zum Beispiel beim Anschließen einer Maus der passende Maustreiber geladen. Das USB-Device kommuniziert daher auch, um welchen »Typen« es sich handelt, mittels der Parameter »Vendor ID« (VID) und »Product ID« (PID). Für unsere Versuche filtern wir die VID/PID vom USB-Serial-Konverter des Arduinos, um von Android her kommunizieren zu können.

Tipp

»USB in a Nutshell« (<http://www.beyondlogic.org/usbnutshell/usb1.shtml>) bietet eine detaillierte Übersicht des USB-Standards. Hier werden u. a. alle vorhandenen »Geräte-Profile« besprochen.

Ein weiteres wichtiges Feature von USB ist *USB On-The-Go* (OTG). Falls vom Betriebssystem unterstützt, ermöglicht OTG es, ein USB-Device in einen USB-Host

umzuwandeln. So ist es auch möglich, bei gewissen Android-Geräten eine Maus oder Tastatur anzuschließen.

Eine Standard-Typ-A-USB-Buchse wie bei einem Computer gibt es bei Android-Geräten äußerst selten. Deswegen muss für das Anschließen einer Maus oder ähnlicher Geräte in den meisten Fällen ein sogenannter USB-OTG-Adapter verwendet werden. Mit USB OTG ist es auch möglich, externe USB-Devices mit Energie zu versorgen. USB OTG wird allerdings nicht von allen Geräten automatisch unterstützt.

Zwar ist USB OTG hardwaremäßig bei fast allen verbauten Mikrocontrollern für Smartphones und Tablets vorhanden, aber es hängt allein vom Hardware-Vertreiber ab, ob die Software USB OTG unterstützt. Besitzt man ein Android-Gerät, das USB OTG nicht unterstützt, kann man dem immer noch mit einem »Custom ROM« entgegenwirken. Speziell das *CyanogenMod*-Projekt (<http://www.cyanogenmod.org/>) ist bestrebt, dem Benutzer eine Alternative zur vorinstallierten »Interpretation« von Android zu geben.

USB OTG

Galaxy Nexus, Motorola Xoom, Nexus 7 und Galaxy S3 unterstützen USB OTG und auch die Android-USB-Host-API. Zum jetzigen Zeitpunkt sieht es so aus, als würde das Nexus 4 USB OTG nicht unterstützen. Dies kann sich jedoch mit einem Softwareupdate noch ändern – allerdings scheinen weder der Hersteller noch Google daran interessiert zu sein, da kein Statement zur Abwesenheit dieses Features vorhanden ist. Das Projekt <http://usbhost.chainfire.eu/> versucht mit »user-reports«, USB-Host-Support experimentell ausfindig zu machen. Auf der Webseite kann man also nachschauen, ob ein Gerät USB OTG bzw. USB-Host unterstützt.

Dank des Android-ADK (*Accessory Development Kit*) ist es allerdings möglich, mit Android über USB zu kommunizieren, ohne einen USB-Host-Mode vorauszusetzen. Alle Android-Geräte, die einen »Play Store« haben, sind dazu verpflichtet, einen USB-Device-Port zu besitzen. Darüber findet nicht nur Debugging von Apps und Datenaustausch zwischen PC und Android-Gerät statt, sondern auch Kommunikation zu Android-»Accessories«.

Der *Arduino Due* kann dank modernem Mikrocontroller einerseits die Rolle des USB-Hosts übernehmen (mit einem USB-OTG-Adapter) und andererseits auch für ein Android-Gerät mit USB-Host-Support als »serielles« Device verwendet werden. Wird der Arduino an einen USB-Host angeschlossen, so wird er mit Strom versorgt. Unser Android-Tablet oder Android-Smartphone kann also auch als portable Batterie für den Arduino betrachtet werden. Da mit dem *Arduino Due* beide Varianten (USB-Host und USB-Device) möglich sind, werden wir uns in den Programmierbeispielen jeweils mit zwei Lösungsansätzen beschäftigen.

2 Arduino

Arduino ist eine Open-Source-Entwicklungsplattform. Diese Entwicklungsplattform besteht zum einen aus vielen verschiedenen Hardware-Boards und zum anderen aus der *Arduino IDE*, einer Softwareplattform (Entwicklungsumgebung) für die Programmierung der Mikrocontroller. Für dieses Buch verwenden wir das Ende 2012 erschienene *Arduino Due* als Mikrocontroller-Board.

2.1 Installieren der Arduino IDE

Die *Arduino IDE* kann von der offiziellen Website www.arduino.cc heruntergeladen werden. Unter dem Reiter *Download* wählen Sie die entsprechende Version (Windows, Mac OS X, Linux 32/64-bit). Da wir für den Arduino Due entwickeln werden, brauchen wir die Arduino IDE mindestens in der Version 1.5.1. Zur Zeit der Erstellung dieses Buches befindet sich diese Version noch offiziell in Beta-Phase. Entpacken Sie das heruntergeladene Archiv an einen passenden Ort und starten Sie die Applikation. In Ubuntu/Linux könnte dies folgendermaßen geschehen:

```
user@host:~$ cd Downloads
user@host:~/Downloads$ tar -xvf arduino-1.5.1-linux32.tgz
user@host:~/Downloads$ mv arduino-1.5.1 ..
user@host:~/Downloads$ cd ..
user@host:~$ arduino-1.5.1/arduino
```

Auf der Rückseite des Arduino Due sehen wir, dass zwei unterschiedliche USB-Buchsen zur Verfügung stehen. Das ist zum einen der native USB (NATIVE USB -> SAM3X) und zum anderen die Programmierschnittstelle (PROGRAMMING -> ATMEGA16U2). Der native USB-Anschluss ist eine Neuheit des Arduino Due. Dieser native USB-Port ist direkt mit dem Haupt-Mikrocontroller (SAM3X) verbunden. Er kann als USB-Host oder als USB-Device fungieren. Bisherige Arduino-Boards hatten meist nur die Programmierschnittstelle. Dabei handelt es sich um einen USB-Port, der mit einem kleineren Mikrocontroller verbunden ist. Beim Arduino Due ist das der ATMEGA16U; frühere Arduino-Boards verwendeten auch den FTDI232-Chip. Auf diesem kleinen Mikrocontroller läuft eine USB-zu-Seriell-Firmware und die Pins sind mit dem UART-Interface (RX/TX-Pins) des Haupt-Mikrocontrollers verbunden.

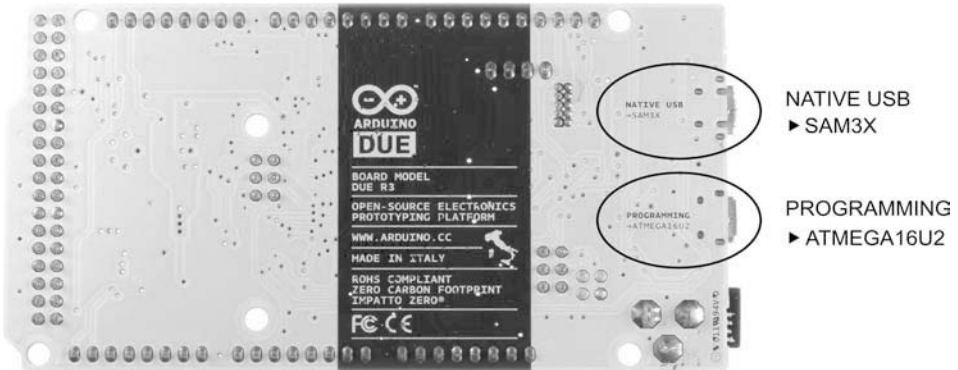


Bild 2.1: Die beiden USB-Buchsen am Arduino-Due-Board.

Verbinden Sie Ihren Host-Computer mit der Programmierschnittstelle (PROGRAMMING -> ATMEGA16U2) via USB-Kabel. Starten Sie auch die *Arduino IDE*, falls sie noch nicht ausgeführt wird.

2.2 Ein erster Sketch

In der Arduino-Entwicklungsumgebung wird ein Programm auch »Sketch« genannt. Standardmäßig sind bereits einige Sketches vorhanden. Wählen Sie deshalb *File > Examples > Basics > Blink*, um einen vordefinierten Sketch zu laden. Am Anfang der Datei steht kurz beschrieben, welche Funktion der Sketch hat:

```

/*
  Blink
  Turns on an LED on for one second,
  then off for one second, repeatedly.

  This example code is in the public domain.
  */

```

Dieses Programm schaltet also die LED periodisch an und ab, jeweils für eine Sekunde.

Stichwortverzeichnis

A

Accessory 70, 73
 Accessory Connected 87
 Accessory Development Kit 12
 accessory_filter.xml 71, 82
 Accessory-API 10, 69, 74
 Accessory-USB-Protokoll 74
 Activity 36, 37, 50
 automatisch starten 82
 activity_main.xml 53, 62, 82, 104, 118, 141, 160
 ADB 41, 44
 adb-Commands 41
 ADK 12, 70
 ADK-Interface 158, 175
 adkLoop 137, 138
 ADK-Thread 92, 93, 123, 125
 ADT Bundle 41, 43
 ADT-Plugin 40, 44
 analogWrite() 29
 Android 7, 33
 Android Accessory API 10
 Android Application Project 47
 Android Debug Bridge 41
 Android Developer Tools 40, 45
 Android Not Responding 64
 Android SDK 8, 36, 43
 Android Services 35
 Android Software
 Development Kit 36, 39
 Android USB Host API 70
 Android-App 36
 Android-Entwickler-Tools 36
 Android-Log-Buffer 59
 Android-Manifest 38, 70
 AndroidManifest.xml 38, 72, 80, 102, 126
 Android-Projekt 47
 Android-Version 38
 Anschlusspins 112
 API 69
 API-Versionen 49
 APK-Paket 37
 App
 Bestandteile 38
 Name 49
 Application is not responding 61
 Application Name 49
 Application Programming Interface 69
 Applikationen 36
 Arduino 9, 13
 Arduino als Accessory 70
 Arduino Due 9, 12, 13, 23, 32, 94
 Arduino IDE 9, 13
 Arduino Servo Library 30, 173
 Arduino Uno 10
 Arduino-Board 9
 ArrayBlockingQueue 89, 141, 147
 ASCII-Format 25
 AsyncTask 68
 ATmega16U2 24
 atoi 138
 Auto Formatting 60
 Autocomplete 60

B

Benutzer 18
 Betriebssystem 7, 33

Bibliotheken 35
 Bidirektionale
 Kommunikation 110, 132
 Blank Activity 50
 Blink 14, 19
 Bluetooth 10
 Board auswählen 15
 Brightness 142
 BroadcastReceiver 38, 89, 92, 100
 Build number 55
 Bulk Transfer 95
 Button 37, 62
 Byte-Code 37
 Bytes in Java 77

C

CAN-Bus 23
 close() 124
 Command Line Tools 35, 41
 Common-Anode-LED 136
 Compile 17
 COM-Port 15, 23, 28, 94
 Configure App 50
 ContentProvider 38
 Content-Provider 36
 Control Transfer 95
 Controller Area Network 23
 Cortex M3 9
 Custom ROM 12

D

Daemons 34
 Dalvik Virtual Machine 35, 37
 Dalvik-Bytecode 35
 Dalvik-Code 37
 Datendurchsatzrate 95

- Debounce 115
 - Debugging 28, 45
 - Debugmeldungen 60
 - Detach-Event 89
 - Development-Tools 36
 - device_filter.xml 103
 - digitalWrite() 23
 - Disconnect 149
 - Drehregler 155
 - Anschlusspins 156
 - Aufbau 156
 - Signalerzeugung 157
 - Terminals 156
 - Drucktaster 111, 112
 - Anschlusspins 112
 - Debounce 115
 - Entprellung 115
 - Duty Cycle 30, 173
 - DVM 37
- E**
- Eclipse 36, 40, 43, 45, 55
 - Shortcuts 60
 - Tricks 59
 - eclipse-Ordner 44
 - Endpoint 96, 109, 125, 132
 - Endpunkte 96
 - Entprellung 115, 158
 - Entwicklungseinstellungen 55
 - Entwicklungs-Tools 36
 - Entwicklungsumgebung 36, 40
 - Enumeration 71
 - Exception 64, 123
- F**
- FileDescriptor 74
- G**
- General Public License 7
 - Gerätefilter 95
 - Google Nexus 7 43
 - GPL 7, 8
 - Graphical Layout 53
- grep** 61
- H**
- Host-Computer 23
 - HSB 133
 - HSV 133, 141
 - HSVToColor 148
 - HTerm 28, 95
 - Hue 142
- I**
- I²C 23
 - Icon 50, 51
 - IDE 36, 40
 - ImageView 119
 - include 173
 - init 34
 - InputStream 118
 - Integrated Development Environment 40
 - IntentFilter 72
 - Intents 38
 - Interfaces 98
 - Interrupt Transfer 96
 - IP-Adresse 42
 - Isochronous Transfer 96
- J**
- Java 8, 61
 - Java Native Interface 8
 - Java Virtual Machine 37
 - Java-Bytecode 35
 - JDK 43
 - Jump to 60
 - JVM 37
- K**
- Kernel 34
 - Klassenrumpf 63
 - Kommandozeile 41
 - Kommentarzeilen 20
 - Kompilierung 17
- L**
- launchMode 81
- LED** 14, 20, 29, 100, 133, 135, 149
- LED-Intensität** 29, 30, 31
- Libraries** 35
- Linux** 7
- Linux-Kernel** 7, 34
- Listener** 147
- Log.d** 59, 61
- LogCat** 60, 61
- loop()** 21
- M**
- mAdkRunnable 164
 - main()-Funktion 21
 - MainActivity.java 55, 59, 62, 83, 104, 119, 126, 142, 149, 161
 - MainActivity-Klasse 59
 - main-Thread 61
 - Main-Thread 92, 93
 - Manifest 38
 - Manifest-Eintrag 71
 - map 175
 - minicom 95
 - minSdkVersion 72
 - Multithreading 36, 61
 - mValueQueue 89
- N**
- Native Daemons 34
 - Native Development Kit 36, 39
 - Nativer USB-Anschluss 13
 - NDK 39
 - Netzwerk-Modus 42
- O**
- offer 89, 148
 - On-board-LED 20, 29, 78
 - OnClick-Handler 62
 - onCreate 59, 62, 87, 88
 - onDestroy 87
 - onNewIntent 87, 93, 124
 - onPause 125
 - onProgressChanged 89

onResume 87, 125
 OnSeekBarChangeListener
 147
 onStop 87, 91, 125
 Open Handset Alliance 7
 Open Source 33
 OpenJDK 43
 Oracle JDK 43
 OsciPrime 10
 OTG 11
 OTG-Modus 94
 OutputStream 90, 93

P

Package Explorer 55
 Package Name 49
 Package-Manager 37
 Paketnamen 38
 Pegel 82
 Permissions 38
 PID 11
 pinMode() 22
 Pin-Nummer 21
 Pipes 110
 Präfix für Variablen 65
 Product ID 11
 product-id 103
 Programmierschnittstelle 13,
 15
 Programmiersprache
 Arduino 19
 C 19, 37
 C++ 19, 37
 Java 37
 Programming Port 15
 Project Name 49
 Pulsweitenmodulation 29, 30
 PWM-Frequenz 29, 173
 PWM-Signal 29, 136, 173

Q

Queue 91, 93

R

read() 124

Refactor 60
 res-Ordner 82
 Ressourcensystem 38, 39
 RGB 135, 141
 RGB-LED 133, 135, 142
 Rotary Encoder 155
 RS-232-Standard 23
 Run 60
 Runnable 62
 RX-Pin 23

S

SAM3X 13, 21, 24
 Sandboxing 37
 Saturation 142
 SDK 39
 sdk-Ordner 44
 SDK-Version 72
 SeekBar 78, 82, 83, 89, 93,
 133, 142, 173
 Serial Monitor 26, 28
 Serial Port 15
 Serial.read() 98
 Serial.write() 98
 serialLoop 137, 138
 Serielle Schnittstelle 11, 23,
 94
 Serielle Verbindung 23
 Service 38
 Servomotor 173
 Anschlüsse 173
 Versorgungsspannung 174
 setup() 21, 32
 Shell 41, 59
 singleTop 81
 Sketch 14, 78, 113, 137, 173,
 175
 Software Development Kit 36
 sp 161
 strings.xml 62
 Suggestion 60
 System Services 35

T

take() 89

Task parallel starten 63
 Template 50
 Terminal-Programme 28
 Textfelder 37
 Textgröße 161
 TextView 155, 160
 Theme 49
 Thread 62, 110
 Synchronisierung 110
 Treiber 11
 TWI 23
 Two-wire Interface 23
 TX-Pin 23

U

UART 23
 UI 36
 Universal Asynchronous
 Receiver Transmitter 23
 Universal Serial Bus 10
 Universal Synchronous
 Asynchronous Receiver
 Transmitter 23
 Upload 17
 USART 23
 USB 10, 11, 34
 Buchsen 13
 Geräteklassen 11
 Nativer Anschluss 13
 OTG 11
 Programming-Port 23
 USB debugging 57
 USB On-The-Go 9, 11
 USB Serial 10
 USB-Bulk-Transfer 94
 UsbConnection 100
 USB-Debugging 55
 UsbDevice 99
 USB-Device 9, 11, 12, 13, 23
 USB-Host 10, 11, 12, 13, 23,
 69, 94, 95
 USB-Host-API 10, 69, 94, 95,
 165
 USB-Hub 110
 UsbManager 74, 95, 98

USB-OTG-Adapter 12
USB-OTG-Konverter 9, 69
USB-Schnittstelle 69, 94
USB-Serial-Konverter 9, 11
USB-Serial-Schnittstelle 94
USB-Seriell-Wandler 9
USB-Slave 95, 109
USB-Transfer 95
USB-zu-seriell-Adapter 23
User Interface 36, 37

V

Variablen 65
Vendor ID 11
vendor-id 103
Verify 17
VID 11
View 37

W

Wi-Fi 10

X

XML-Filter 95
xml-Ordner 82

Y

yield() 32

Z

Zugriffsrechte 19

Manuel Di Cerbo / Andreas Rudolf

Android mit Arduino™ Due

Mit Android und Arduino™ treffen die zwei Platzhirsche in ihrem jeweiligen Segment aufeinander. Android ist in den letzten Jahren zur Nummer eins unter den Smartphone-Betriebssystemen aufgestiegen. Arduino™ zählt zu den wichtigsten Plattformen für eigene Hardwareprojekte und wird weit über die Elektronik hinaus eingesetzt, z. B. auch von Künstlern. Android und Arduino™ haben eines gemeinsam: beide sind Open Source. Aufgrund dieser Eigenschaft haben sich hinter beiden Systemen riesige Communities gebildet. Eine Kombination der beiden Systeme ist deswegen die perfekte Symbiose.

Programmierung des Arduino™ Due

Die Programmierung des Arduino™ Due erfolgt mit der schon vom Arduino™ Uno bekannten Arduino™ IDE. Für Neueinsteiger in die Welt von Arduino™ wird die Installation der Entwicklungsumgebung detailliert beschrieben. Darauf aufbauend werden erste Beispiele mit der Arduino™ IDE umgesetzt, dabei wird auch die Programmiersprache für Arduino™ Due eingeführt.

Android-Apps entwickeln

Bei Android handelt es sich um ein Betriebssystem auf der Basis von Linux. Nach einer Vorstellung der wichtigsten Eigenheiten von Android wird die Installation der Entwicklungsumgebung (IDE) erläutert. Als IDE kommt dabei Eclipse zum Einsatz. Anhand einer ersten Android-App wird schrittweise die Nutzung der IDE gezeigt. Praxis-Tipps für Eclipse erleichtern den Umgang mit der IDE im Alltag.

Android mit Arduino™ Due kombinieren

Mit vier Projektbeispielen, z. B. der Ansteuerung eines Servomotors, wird die Verbindung zwischen Android und Arduino™ Due anhand von Quellcode ausführlich aufgezeigt. Der Quellcode dieser Projekte ist natürlich als Download-Material verfügbar.

Aus dem Inhalt:

- Arduino™ IDE installieren
- Erste Projekte mit der Arduino™ IDE umsetzen
- Einstieg in die Arduino™-Due-Programmierung
- LED steuern über die serielle Schnittstelle
- LED steuern über Android USB Accessory Mode
- Einstieg in das Betriebssystem Android
- Entwicklungsumgebung für Android installieren
- Praxis-Tipps zur Verwendung von Eclipse
- Erste Projekte für Android umsetzen
- USB-Kommunikation über Android
- Einstieg in das Android Accessory Development Kit (ADK)
- Drucktaster auslesen mit Android und Arduino™ Due
- RGB-LED mit Android ansteuern

Über die Autoren:

Manuel Di Cerbo und Andreas Rudolf haben ihr Elektro- und Informations-technikstudium im Sommer 2010 abgeschlossen. Ihre Diplomarbeit „Using Android in Industrial Automation“, die sich mit dem Einsatz von Android auf Embedded Systemen befasst, wurde als beste technische Arbeit der Fachhochschule Nordwestschweiz prämiert. Mit ihrem Startup „Nexus-Computing GmbH“ spezialisierten sie sich fortan auf Android-Software- und Systementwicklung.

Auf www.buch.cd

Der komplette Quellcode des Buchs

Besuchen Sie
unsere Website
www.franzis.de

FRANZIS



9 783645 602051

30,- EUR [D]

ISBN 978-3-645-60205-1