Idee und Entwicklung der Arduino-Platine

1

In diesem Kapitel

- Arduino entdecken
- Die Herkunft und Bedeutung des Arduinos
- Die Basisaspekte kurz vorgestellt

Das Arduino besteht als Gesamtlösung aus Hard- und Softwarekomponenten. Dazu gehört zunächst einmal die Arduino-Platine (PCB – Printed Circuit Board) selbst. Diese wurde speziell für einen Mikrocontroller-Chip entwickelt und stellt eine ganze Reihe weiterer Anschlüsse bereit, die sich als Ein- und Ausgänge nutzen lassen. Darüber hinaus befinden sich auf der Platine weitere elektronische Komponenten, ohne die der Mikrocontroller nicht funktionieren würde und/oder die seine Fähigkeiten erweitern.

Mikrocontroller sind letztlich integrierte Kleinstcomputer, die so weit verkleinert wurden, dass sie sich in einem einzigen Chip unterbringen ließen. Sie eignen sich hervorragend für die Programmierung und Steuerung elektronischer Schaltungen. Viele Mikrocontroller genannte Geräte enthalten neben dem Mikrocontroller selbst weitere nützliche Anschlüsse und Komponenten, an die deren Benutzer Ein- und Ausgabegeräte anschließen können. Beispiele für Mikrocontroller-Platinen sind das *Wiring Board*, die *PIC-Boards* und das *BASIC Stamp*.

Mit der Arduino-Entwicklungsumgebung können Sie Software schreiben, die dem Mikrocontroller mitteilt, was er machen soll. Mit einer Programmzeile können Sie eine LED beispielsweise ein- oder ausschalten und sie so blinken lassen. Schließen Sie einen Taster an und fügen eine weitere Codezeile hinzu und schon blinkt die LED nur bei gedrücktem Tastschalter auf. Soll die LED vielleicht nur blinken, wenn der Taster gedrückt gehalten wird? Auch kein Problem. Mit den hier gebotenen Möglichkeiten können Sie elektronischen Systemen schnell und einfach verschiedene Verhaltensweisen beibringen. Ohne Mikrocontroller ließe sich so etwas nur schwer oder überhaupt nicht verwirklichen.

Ähnlich wie konventionelle Rechner können auch Arduinos vielfältige Funktionen übernehmen, bleiben aber auf sich allein gestellt weitgehend nutzlos. Erst mit weiteren Ein- und/oder Ausgängen wird die Platine wirklich nützlich. Dort lassen sich Sensoren anschließen, mit deren Hilfe Computer auf ihre Umwelt reagieren und/oder Dinge in der realen Umwelt steuern können.

Um die nachfolgenden Darstellungen besser verstehen zu können, könnte es nützlich sein, die bisherige Geschichte des Arduinos zumindest in ihren Grundzügen zu kennen.

Wie entstand die Idee für das Arduino?

Die Idee für das Arduino wurde in Italien am IDII (Interaction Design Institute Ivrea) geboren, einer Graduiertenschule für *Interaktionsgestaltung* (interaction design). Das Institut bot Designern weitere Qualifikationsoptionen und hatte sich schwerpunktmäßig auf Mensch-Computer-Interaktionen spezialisiert.

Der Begriff der *Interaktionsgestaltung* (interaction design) wurde Mitte der 1980er von Bill Verplank und Bill Moggridge geprägt. Abbildung 1.1 enthält ein Schaubild von Verplank, das die grundlegende Prämisse der Interaktionsgestaltung verdeutlichen soll. Es zeigt die Funktionsweise des Interaktionsprozesses: Eigene Aktionen führen zu Änderungen und einem wahrnehmbaren Wandel, der selbst von Umweltänderungen angestoßen wurde.

Mit freundlicher Genehmigung von Bill Verplank

Abbildung 1.1: Das Prinzip der Interaktionsgestaltung

Da es sich hier um ein allgemeingültiges Prinzip handelt, sollte besser hinzugefügt werden, dass es bei der Interaktionsgestaltung meist um Interaktionen mit Computern über Peripheriegeräte (beispielsweise Maus, Tastatur und/oder Touchscreen) geht, die der Navigation im digitalen Umfeld dienen, was sich wiederum beispielsweise auf die grafische Bildschirmdarstellung auswirkt.

In einem weiteren Bereich geht es um das *Physical Computing* und damit um die Übertragung der Möglichkeiten solcher Computerprogramme, Software oder Systeme und ihrer Erweiterung bis in die reale Welt hinein. Computer können bei Nutzung entsprechender elektronischer Bauteile etwas über ihre Umwelt erfahren und auch physisch darauf einwirken.

In den Bereichen der Interaktionsgestaltung und des Physical Computings braucht man *Prototypen* zur umfassenden Erforschung der Interaktionen. Und das stellte Designstudenten jenseits der technischen Fachbereiche vor nicht unbedeutende Probleme und Hindernisse.

Processing wurde als Projekt 2001 von Casey Reas und Benjamin Fry ins Leben gerufen. Es sollte auch Nichtprogrammierer zur Programmierung hinführen und ihnen schnell und leicht nutzbare Möglichkeiten bieten, mit denen sie Visualisierungen und Grafiken auf den Bildschirm bringen konnten. Das Projekt versorgte Benutzer mit einem digitalen Zeichenblock, mit dessen Hilfe sie ihre Ideen und Experimente mit geringem Zeitaufwand festhalten und ausprobieren konnten. Von diesem Projekt wurde wiederum ein ähnliches inspiriert, das auch Experimente in der realen Welt möglich machte.

Auf denselben Prinzipien wie *Processing* aufbauend begann Hernando Barragán 2003 mit der Entwicklung der Mikrocontroller-Platine *Wiring*, bei der es sich um einen Arduino-Vorläufer handelt.

Wie bei Processing sollten auch hier Künstler, Designer und andere Nichttechniker mit eingebunden und berücksichtigt werden. Wiring sollte seinen Nutzern aber weniger die Programmierung als die Elektronik näherbringen. Das in Abbildung 1.2 abgebildete *Wiring Board* war zwar preiswerter als einige andere Mikrocontroller (beispielsweise *PIC* und *BASIC Stamp*), für Schüler und Studenten aber immer noch eine gewisse Investition.

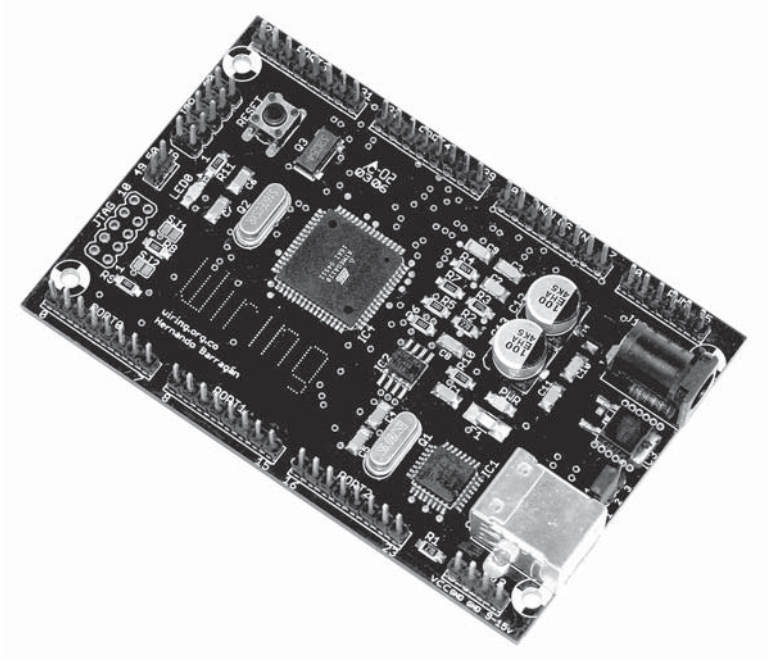

Abbildung 1.2: Eines der ersten Wiring Boards

In einer Reaktion auf die Nachfrage nach erschwinglichen Geräten, die im Rahmen des Studiums der Interaktionsgestaltung leicht in Projekten genutzt werden konnten, wurde 2005 das Arduino-Projekt gestartet. Angeblich sollen Massimo Banzi und David Cuartielles das Projekt zwar nach dem italienischen König *Arduin von Ivrea* benannt haben, aber wie mir aus zuverlässigen Quellen zugetragen wurde, hieß auch ein Lokal in der Nähe der Universität so, das möglicherweise eine bedeutsamere Rolle für das Projekt gespielt haben könnte.

Das Arduino-Projekt konnte in vielerlei Hinsicht von den durch Wiring und Processing erworbenen Erfahrungen profitieren. Offensichtlich werden die Einflüsse von Processing beispielsweise bei der von der Arduino-Software verwendeten *grafischen Benutzeroberfläche* (GUI – Graphical User Interface). Diese GUI wurde anfangs einfach von Processing »übernommen«. Heute ähneln sich die beiden Varianten zwar immer noch, aber die heutige GUI ist schon deutlich spezieller auf Arduino zugeschnitten. Mit der Arduino-GUI werden wir uns in Kapitel 4 eingehender befassen.

Arduino hat auch die Processing-Namenskonventionen beibehalten und nennt Programme *Sketch* (Skizze). So wie Processing seinen Benutzern als digitaler Zeichenblock diente, mit dessen Hilfe sie Programme schnell erstellen und testen konnten, bietet Arduino seinen Nutzern Möglichkeiten, ihre Hardware-Ideen mal eben kurz zu »skizzieren«. Im weiteren Verlauf dieses Buches werde ich Ihnen viele Sketche vorstellen, mit deren Hilfe Ihr Arduino außerordentlich vielfältige Aufgaben erledigen kann. Wenn Sie diese Beispielsketche nutzen und variieren, lernen Sie schnell, wie sie funktionieren, und werden rasch eigene erstellen können. Um auch wirklich kein Detail zu vergessen, werden die jeweiligen Sketche und ihre Funktionsweise anschließend zeilenweise erläutert.

Ein Entwicklungsziel der in Abbildung 1.3 dargestellten Arduino-Platine bestand darin, sie robuster und fehlertoleranter als Wiring und die anderen älteren Mikrocontroller zu machen. Bisher war es keineswegs ungewöhnlich, wenn Studenten und Profis (insbesondere mit künstlerischem oder gestalterischem Hintergrund) ihre Platinen bereits nach wenigen Minuten der Nutzung einfach mal eben durch verpolt angeschlossene Leitungen zerstört hatten. Diese Empfindlichkeit konnte nicht nur wirklich teuer werden, sondern war auch hinsichtlich des Erfolgs der Platinen außerhalb technischer Kreise problematisch.

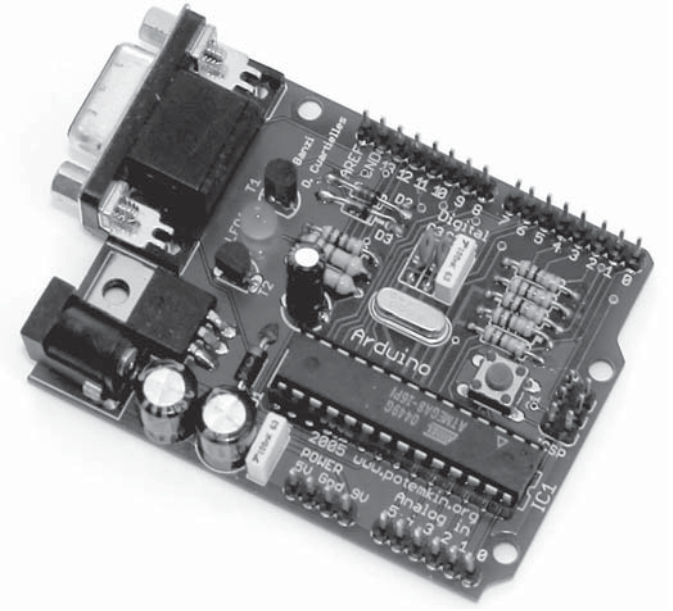

Abbildung 1.3: Das ursprüngliche Arduino Serial

Zudem kann der Mikrocontroller-Chip beim Arduino ausgetauscht werden. Da er gesockelt montiert wird, lässt sich auch nur der Chip anstelle der kompletten Platine wechseln.

Einen weiteren wichtigen Unterschied zwischen Arduino und anderen Mikrocontroller-Platinen bilden die Kosten. Der ebenfalls beliebte Mikrocontroller *BASIC Stamp* kostete 2006 fast das Vierfache wie ein Arduino. Selbst heute kostet das Wiring Board meist noch etwa das Doppelte eines Arduinos, das in der Uno-Variante bei etwa 30 Euro oder auch noch ein wenig darunter liegt. In einem meiner ersten Arduino-Workshops hieß es, dass der Preis für Studenten erschwinglich sein sollte. Und für 30 Euro konnte man damals zwischen einem Projekt und einem netten Essen wählen.

Die Angebotspalette ist heute bei Arduino-Platinen deutlich umfangreicher als 2006. In Kapitel 2 werde ich Ihnen einige der nützlichsten Arduino- und Arduino-kompatiblen Platinen und deren Unterschiede und etliche weitere Produkte vorstellen, die Sie in eigene Projekte integrieren können. In Kapitel 13 wird es dann noch um spezielle *Aufsteckplatinen* (*Shields*) gehen, mit denen Sie Ihr Arduino um zuweilen nützliche und manchmal auch beeindruckende Funktionen erweitern und es beispielsweise zu einem GPS-Empfänger, einem Geiger-Zähler oder sogar einem Mobiltelefon machen können.

Lernen durch Handeln

Menschen haben die Technologie auf vielfältige Weise genutzt, um eigene Ziele zu verwirklichen, ohne sich mit lästigen technologischen Details befassen zu müssen. Nachfolgend werde ich Ihnen einige verwandte Denkansätze vorstellen, die es Interessierten erlauben, sich spielerisch mit Elektronik auseinanderzusetzen.

Patching

Patching nennt man die Vorgehensweise beim experimentellen Umgang mit Systemen. Beim wohl ältesten bekannten Beispiel des Patchings handelt es sich um die aus alten Filmen noch bekannten manuellen Telefonvermittlungsstellen mit den Damen (und gelegentlich auch Herren) vom Amt. Die *Operatoren* mussten zur Herstellung der gewünschten Telefonverbindungen Kabel umstecken. Später war diese Vorgehensweise auch bei den ersten *Synthesizern* im Musikbereich verbreitet. Eines der bekanntesten Beispiele bildet hier der legendäre *Moog-Synthesizer*.

Wenn elektronische Musikinstrumente Töne erzeugen, erzeugen sie dabei eigentlich unterschiedliche Spannungen. Andere Bauteile im Instrument verarbeiten diese weiter, bevor sie schließlich in Form hörbarer Töne ausgegeben werden. Beim Moog-Synthesizer wird dazu der Pfad geändert, über den die Spannung weitergeleitet wird, wobei sie eine Reihe verschiedener Komponenten passiert, die für die unterschiedlichen Klangeffekte sorgen.

Aufgrund der schieren Vielzahl möglicher Kombinationen müssen sich die Experimente der Musiker weitgehend auf Versuch und Irrtum stützen. Bedingt durch die einfache Schnittstelle läuft der Prozess aber extrem schnell ab und erfordert anfangs nur sehr wenig Vorbereitungszeit.

Hacking

Hacking ist heute zu einem verbreitet verwendeten Begriff geworden, der nicht nur auf subversive Elemente im Internet angewendet werden kann. Im allgemeineren Sinne bezieht er sich auf die Erforschung von Systemen und deren umfassende Nutzung oder deren Anpassung oder Umgestaltung, um sie den eigenen Anforderungen anzupassen.

Hacking in diesem Sinne ist sowohl auf der Hardware- als auch auf der Softwareebene möglich. Ein beliebtes Beispiel sind Tastatur-Hacks. Angenommen, Sie wollen mit einem großen roten Schalter zwischen den Bildern einer Diashow wechseln. In den meisten Programmen werden zu diesem Zweck Tastaturkürzel verwendet und in vielen PDF-Betrachtern erfüllt zudem die Leertaste dieselbe Funktion. Wenn Sie dies wissen, wäre es doch praktisch, eine Tastatur zu verwenden, die lediglich aus der Leertaste besteht.

Computertastaturen wurden mittlerweile derart verfeinert, dass sich in einer Standardtastatur nur noch eine kleine Platine befindet, die kaum größer als eine Kreditkarte ist (siehe Abbildung 1.4). Darauf befinden sich in erster Linie eine Menge Kontakte, die bei der Betätigung der verschiedenen Tasten miteinander verbunden werden. Wenn Sie die richtige Kombination kennen, können Sie einfach ein paar Drähte auf der anderen Seite mit den Kontakten eines Druckschalters verbinden. Wenn Sie nun diesen Schalter betätigen, wird dasselbe Signal wie beim Drücken der Leertaste zu Ihrem Rechner übertragen.

Abbildung 1.4: Die Innereien einer USB-Tastatur

Diese Vorgehensweise eignet sich hervorragend, um knifflige Hardwareprobleme zu vermeiden und dennoch zu den gewünschten Ergebnissen zu kommen.

Circuit Bending

Circuit Bending setzt sich über alle Prinzipien der traditionellen Ausbildung hinweg und stützt sich vollständig auf spontane Experimente. Kinderspielzeuge bilden zwar meist die Basis der Circuit Bender, aber letztlich besitzen beliebige elektronische Geräte das Potenzial, als Grundlage der Experimente dienen zu können. Wenn Sie ein elektronisches Spielzeug oder Gerät öffnen und seine Schaltung freilegen, können Sie den Weg des Stroms ändern und damit dessen Verhalten verändern. Auch wenn diese Technik der des Patchens ähnlich ist, sind die Ergebnisse doch weit schlechter vorhersehbar. Wenn Sie erst einmal geeignete Kombinationen gefunden haben, können Sie auch Komponenten wie Widerstände oder Schalter hinzufügen oder ersetzen, um Benutzern eine bessere Bedienung der Instrumente zu ermöglichen.

Beim Circuit Bending geht es meist um die Klangerzeugung. Beim fertigen Instrument handelt es sich daher oft um einen rudimentären Synthesizer oder einen Drumcomputer. Zwei der in diesem Bereich beliebtesten Geräte sind der *Speak & Spell* (siehe Abbildung 1.5) und der *Nintendo GameBoy*. Musiker wie das *Modified Toy Orchestra* (`modifiedtoy orchestra.com`) »erforschen das verborgene Potenzial und den latenten Zusatznutzen, der sich in redundanten Technologien verbirgt«. Überlegen Sie es sich also besser doppelt, bevor Sie Ihre alten Spielzeuge bei eBay verhökern!

Abbildung 1.5: Ein vom Modified Toy Orchestra modifizierter Speak & Spell nach dem Circuit Bending (mit freundlicher Genehmigung von Modified Toy Orchestra)

Elektronische Bauteile

Es gibt zwar eine Menge Möglichkeiten des Umgangs mit Technologien, aber irgendwann werden Sie nach Höherem streben und von allem ein wenig mehr haben wollen: mehr Präzision, höhere Komplexität und umfassendere Steuerungsmöglichkeiten. Wenn Sie Elektronik in der Schule hatten, werden Sie möglicherweise gelernt haben, wie Schaltungen aus bestimmten Komponenten erstellt werden können. Diese Schaltungen basieren einzig auf den chemi-

schen Eigenschaften der Komponenten und müssen genau berechnet werden, damit die richtige Menge Strom durch die richtigen Bauteile fließt.

Für derartige Schaltungen finden Sie Bausätze im Fach- oder teilweise auch im Spielzeughandel. Sie erfüllen eine bestimmte Aufgabe wie beispielsweise als Eieruhr oder Warnsirene, die beim Öffnen einer Keksdose ertönt. Diese Bausätze erfüllen zwar ihre jeweilige Aufgaben wirklich gut, lassen sich darüber hinaus aber kaum weiter nutzen, zumal es sich bei den einzelnen Bauteilen heute oft um herstellerspezifische Steckmodule handelt, die auch für Kinder unproblematisch sind.

Hier kommen Mikrocontroller ins Spiel. Dabei handelt es sich um winzige Computer, die zusammen mit elektronischen Schaltungen eingesetzt werden können, um ihre Möglichkeiten zu erweitern. Mikrocontroller lassen sich bei Bedarf auch umprogrammieren und können so unterschiedliche Funktionen erfüllen. Ihr Arduino wurde eigentlich um einen derartigen Mikrocontroller herum entwickelt und soll Ihnen dabei helfen, ihn möglichst umfassend zu nutzen. In Kapitel 2 werden wir uns ein Arduino Uno und seinen Aufbau genauer ansehen und erfahren, was diese kleine Platine zu leisten vermag.

Der Mikrocontroller ist zwar das Gehirn des Systems, benötigt aber Daten, um entweder mehr über seine Umwelt zu erfahren oder selbst auf diese einwirken zu können. Diesem Zweck dienen seine Ein- und Ausgänge.

Eingänge

Eingänge (*Inputs*) sind die Sinnesorgane Ihres Arduinos. Sie informieren ihn darüber, was in seiner Umwelt vor sich geht. Ganz grundlegend kann es sich bei einem Eingang einfach um einen Schalter wie den Lichtschalter in Ihrem Wohnzimmer handeln. Am anderen Ende des Spektrums finden Sie dann Komponenten wie beispielsweise ein Gyroskop (Lagesensor), das dem Arduino seine genaue Lage im dreidimensionalen Raum mitteilen kann. Mehr über die einfachen Eingabemöglichkeiten erfahren Sie in Kapitel 7, während die anspruchsvolleren Möglichkeiten der Sensoren und ihrer Nutzung in Kapitel 12 behandelt werden.

Ausgänge

Über *Ausgänge* (Outputs) kann Ihr Arduino irgendwie auf seine reale Umwelt Einfluss nehmen. Ein Ausgang kann sich still und diskret melden, wie beispielsweise der Vibrationsalarm eines Mobiltelefons, es kann sich dabei aber auch um einen riesigen Bildschirm an der Außenwand eines Gebäudes handeln, der weithin sichtbar ist. Im ersten Sketch in diesem Buch wird es darum gehen, eine LED zum »Blinken« zu bringen (siehe Kapitel 4). Anschließend können Sie mit der Motorsteuerung weitermachen (Kapitel 8) und letztlich sogar eine große Anzahl von Ausgängen steuern (siehe Kapitel 14 und 15), um sich die Vielzahl der in Ihren Arduino-Projekten nutzbaren Ausgabemöglichkeiten zu erschließen.

Open Source

Open-Source-Software (quelloffene Software), wie insbesondere *Processing*, hatte gewaltigen Einfluss auf die Arduino-Entwicklung. Im Bereich der Computersoftware handelt es sich bei der Open-Source-Bewegung um eine Philosophie, bei der die Einzelheiten der entwickelten Programme offengelegt werden und Dritte dazu ermutigt werden, diese ganz nach Belieben zu nutzen, neu zu kombinieren und weiterzuverbreiten.

Wie die Processing-Software ist auch die Arduino-Software und die zugehörige Hardware Open Source. Software und Hardware des Arduinos können und dürfen also bei Bedarf beliebig angepasst werden. Wahrscheinlich wegen dieser Freigebigkeit seitens des Arduino-Teams begegnet Ihnen der Open-Source-Geist auch in den Nutzergemeinden und den Arduino-Foren.

Nicht nur in den offiziellen Arduino-Foren (`www.arduino.cc/forum/`) haben Arduino-Nutzer mit Gleichgesinnten ihre Quelltexte, Projekte und Fragen weltweit informell präsentiert. Dadurch können alle Interessierten und damit auch erfahrene Techniker, talentierte Entwickler, praxiserfahrene Designer und innovative Künstler ihr Wissen auch blutigen Neulingen in einigen oder all diesen Bereichen bereitstellen. Die Foren ermöglichen es zudem, die Entwicklung der Nutzerinteressen im Auge zu behalten, was dann wiederum gelegentlich zu neuen offiziellen Versionen der Arduino-Software oder gewissen Verbesserungen oder Erweiterungen der Platine selbst führen kann. Auf der Arduino-Website gibt es einen Bereich, der *Playground* (`http://playground.arduino.cc`) genannt wird. Auf dieser »Spielwiese« werden Nutzer dazu eingeladen, ihre Quelltexte hochzuladen, damit die Gemeinschaft sie nutzen, darüber diskutieren und sie bearbeiten kann.

Diese Philosophie hat die relativ kleine Gemeinschaft dazu ermutigt, ihr Wissen in Foren, Blogs und auf Websites zu sammeln und umfassende Ressourcen für neue Arduino-Fans zu schaffen, die nur darauf warten, genutzt zu werden.

Dabei ist es zu dem ein wenig befremdlichen Phänomen gekommen, dass sich ungeachtet des Open-Source-Gedankens eine enorme Loyalität zum Markennamen Arduino herausgebildet hat, der sogar so weit geht, dass eine Arduino-Namenskonvention entstanden ist, bei der *-duino* oder *-ino* an den Namen von Platinen und Zubehör angehängt wird. (Übrigens sehr zum Missfallen der italienischen Mitglieder des Arduino-Teams!)