Arduino

Teil 1

verstehen und anwenden

Einstieg in die Hardware: Arduino und AVR-Mikrocontroller

Dies ist der erste Teil einer Artikelserie zum Thema Arduino. In diesem Beitrag werden zunächst die Kapitel

- Arduino – das erfolgreichste Mikrocontroller-Board der Welt
- Das Arduino-Board als Basis für µC-Anwendungen
- Arduino-Varianten – vom Mikro zum Mega
- Schneller Einstieg mit der Arduino-Entwicklungsumgebung

behandelt.

In weiteren Beiträgen werden dann u. a. die Themen

- Programmierung und Programmierpraxis
- Programmbibliotheken und Shields
- Schnittstellen wie USB, RS232 und I^2C, Ethernet und WLAN
- Sensortechnik und Messwerterfassung
- Interrupts und Polling
- Displaytechnik
- Physical Computing

vorgestellt.

Arduino – das erfolgreichste Mikrocontroller-Board der Welt

Die Idee zum Arduino wurde im Jahr 2005 am Institut für Interaktives Design in Ivrea (Italien) geboren. Die erste Version des Arduinos bestand aus einem Bausatz mit wenigen Einzelkomponenten und dem damals sehr populären ATmega8 als zentralem Mikrocontroller. Nachdem diese ersten Bausätze innerhalb kürzester Zeit verkauft waren, folgten rasch weitere Auflagen. Künstler und Designer aus anderen Regionen nahmen die Idee auf, und das Arduino-Prinzip verbreitete sich über Italien und Europa schließlich in der ganzen Welt.

Das Konzept einer unkomplizierten und preisgünstigen Hardwareplattform zusammen mit einer frei verfügbaren, leicht zu beherrschenden Programmieroberfläche (die sogenannte IDE = Integrated Development Environment) fand auch bei anderen Anwendergruppen wie etwa Hobbyisten oder Technikakademien Anklang. Schließlich erkannten auch Schulen und Hochschulen im wissenschaftlichen und technischen Bereich das enorme Potenzial der Arduino-Idee. Neben immer neuen Hardware-Versionen entstanden auch aufsteckbare Erweiterungsplatinen, sogenannte Shields, und die Erfolgsgeschichte des Arduinos nahm ihren Lauf.

Inzwischen sind die Verkaufszahlen der einzelnen Arduino-Versionen recht beachtlich, und der Arduino kann mittlerweile sicherlich als das erfolgreichste Mikrocontroller-Board aller Zeiten gelten.

Für diesen überragenden Erfolg des Arduinos gibt es mehrere Gründe. Einerseits wird durch das komplett aufgebaute Prozessorboard der Einstieg in die Hardware enorm erleichtert. Die typischen Anfängerprobleme wie falsche Spannungsversorgungen, Probleme mit dem Quarzoszillator oder ungültige Konfigurationsparameter, sogenannte Fuse-Bits, sind beim Arduino unbekannt. Das Board wird einfach mit der USB-Schnittstelle des PCs oder Laptops verbunden und kann innerhalb weniger Minuten programmiert werden.

Die einfache Handhabung führte insbesondere bei Jugendlichen, aber auch bei Angehörigen der älteren Generation, die noch niemals mit Elektronik in Berührung gekommen sind, zu einer ungeahnten Begeisterung für die Mikrocontrollertechnik.

Ein weiterer Erfolgsfaktor ist die zugehörige Programmieroberfläche. Diese steht als kostenlose Open-Source-Version zur Verfügung. Eine langwierige Instal-

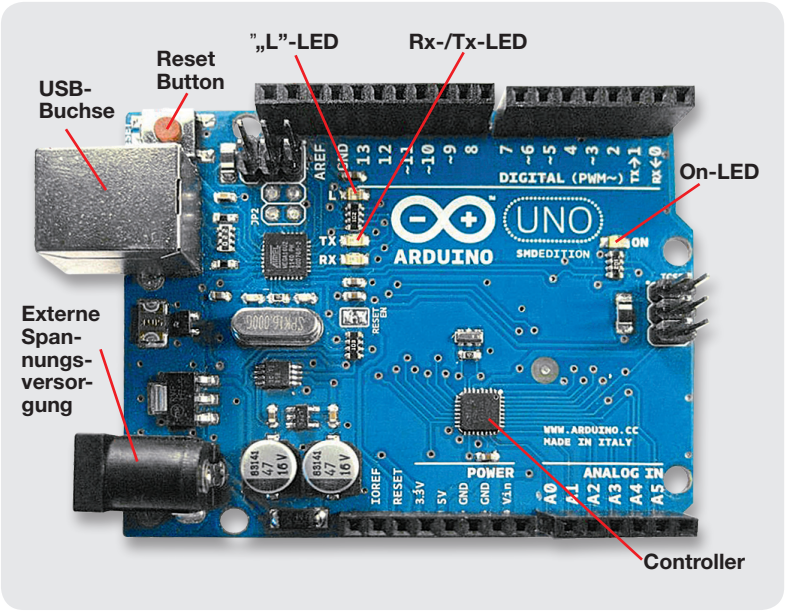

Bild 1: Arduino Uno

lation ist nicht erforderlich, und einfache Einstiegsbeispiele sorgen für den schnellen Erfolg. Komplizierte Parametereinstellungen wie bei anderen Entwicklungstools sind nicht erforderlich. Die ersten Beispielprogramme können so im Handumdrehen auf den Arduino geladen und getestet werden.

Der letzte Grund für die große Popularität des Arduinos findet sich in der ganz eigenen Philosophie des Projekts. Alle Hard- und Softwarekomponenten unterliegen dem sogenannten Open-Source-Prinzip. So werden die Schaltpläne und Layouts aller Arduino-Varianten im Internet veröffentlicht. Gleiches gilt für die Software und die zugehörigen Quellcodes. Weiterhin wird der Arduino-Anwender durch eine Fülle von Software-Bibliotheken unterstützt. Diese nützlichen Libraries werden zum Teil direkt mit der Standard-IDE mitgeliefert. Beispiele hierfür sind Anwendungen zur Thema Ethernet, LC-Displays oder die Ansteuerung von Schrittmotoren und Modellbauservos.

Darüber hinaus hat die Arduino-Community eine nahezu unüberschaubare Fülle von Bibliotheken für alle möglichen Mikrocontroller-Applikationen hervorgebracht. Damit lassen sich auch komplexere Aufgaben wie etwa die Detektion von IR-Fernbedienungssignalen, das Ansteuern von Ultraschallsensoren oder das Auslesen von SD-Speicherkarten problemlos und schnell bewältigen.

Arduino-Boards als Basis für µC-Anwendungen

Das aktuell am weitesten verbreitete Board ist der Arduino Uno (Bild 1). Dieses Board arbeitet mit dem ATmega328 und ergänzt ihn zu einem universellen Mikrocontroller-System.

Neben dem Prozessor selbst weist das Board hierfür noch die folgenden Hauptfunktionseinheiten auf:

Resetschaltung mit Taster
Mit dem Taster kann der Controller manuell neu gestartet werden. Resets sind aber im Gegensatz zu einem Windows-PC nur sehr selten erforderlich.

Quarz für eine zuverlässige und präzise Taktversorgung
Falsch angeschlossene Quarze oder Lastkapazitäten sind die häufigste Ursache bei selbst entwickelten Controllerboards. Durch den fest installierten Quarz entfällt diese Fehlerursache beim Arduino vollständig.

Buchsenreihen für den Anschluss externer elektronischer Bauelemente
Die Buchsen sind in vier Hauptgruppen unterteilt:

a) 6 Analog-Eingänge
b) 14 digitale Ein- und Ausgänge
c) Buchsen für die Stromversorgung (GND, 3,3 V und 5 V)
d) Eingänge für Referenzspannung und Reset

Über diese Buchsenreihen kann auf alle für die Anwendungen relevanten Controllerpins zugegriffen werden.

Anschluss für eine externe Spannungsversorgung
Eine über eine Hohlbuchse zugeführte externe Versorgungsspannung wird auf dem Board geregelt und geglättet.

Eine USB-Schnittstelle
Die USB-Schnittstelle dient zur Programmierung des Controllers. Darüber hinaus kann diese Schnittstelle auch für die Kommunikation des Mikrocontrollers mit einem PC während der Programmausführung eingesetzt werden.

Der wichtigste Baustein auf dem Arduino ist der Mikrocontroller. Hierbei handelt es sich um einen Halbleiterchip, der neben einer Prozessoreinheit auch verschiedene Peripheriefunktionen in einem einzigen IC vereinigt.

Meist ist auch der Arbeits- und Programmspeicher auf dem gleichen Chip integriert. Ein Mikrocontroller ist somit mit einem kompletten Ein-Chip-Computersystem vergleichbar [1].

Die bekannten Kriterien für Prozessoren bestimmen auch die Einsatzmöglichkeiten eines Mikrocontrollers. Tabelle 1 fasst die wesentlichen Leistungsmerkmale des ATmega328 zusammen.

Auf modernen Mikrocontrollern der ATmega-Reihe finden sich zusätzlich zu den einfachen Eingabe-/Ausgabe-Ports auch komplexere Peripheriefunktionen wie z. B.

- serielle Schnittstellen
- I²C-(Inter-Integrated-Circuit-)Bus
- PWM-Ausgänge
- Analog-digital-Wandler

Damit lassen sich alleine mit dem Controller bereits umfangreiche Praxisprojekte realisieren. Als Beispiel für die Leistungsfähigkeit eines modernen Controllers zeigt Bild 2 eine selbst gefertigte Digitaluhr, die allein mit einem ATmega als integriertem Baustein auskommt. Es sind keinerlei weitere ICs erforderlich [2].

Tabelle 1

ATmega328	
32	KBytes Flash-Speicher
1	KByte EEPROM
2	KBytes SRAM
2	8-Bit-Timer/-Counter
1	16-Bit-Timer
6	Kanäle für 10-Bit-ADC
23	programmierbare I/Os

Die Stromversorgung des Arduino-Boards kann über zwei verschiedene Wege erfolgen:

- Über die USB-Schnittstelle:
  Dabei wird die Versorgungsspannung direkt dem angeschlossenen Computer entnommen.
- Über eine externe Spannungsquelle:
  Ist der Arduino fertig programmiert, kann er auch ohne angeschlossenen PC arbeiten. Dazu muss er nur mit einer externen Spannungsquelle versorgt werden. Diese kann sowohl aus einem einfachen, unstabilisierten Steckernetzteil (Spannungsbereich 7–12 V) oder auch aus einem Batteriesatz (5x 1,5 V oder 9-V-Block) bestehen.

An dieser Stelle noch zwei wichtige Hinweise:
TIPP I: Das Arduino-Board muss stets auf einer elektrisch isolierenden Unterlage liegen. Andernfalls kann es zu Kurzschlüssen kommen, und sowohl der USB-Port des Rechners als auch der Arduino können Schaden nehmen (Bild 3).
TIPP II: Will man den USB-Port des Rechners schützen, kann man einen USB-Hub mit eigener Spannungsversorgung einsetzen. Diese Hubs sind meist gut gegen Überlastung abgesichert, und das „Durchschlagen" eines Kurzschlusses über den Hub hinweg zum Rechner ist sehr unwahrscheinlich.

Arduino-Varianten – vom Mikro zum Mega

Außer dem Klassiker Arduino Uno existiert noch eine Vielzahl weiterer Boardvarianten. Neben dem Uno selbst sollen hier zwei weitere interessante Vertreter dieser Familie genauer besprochen werden.

Eine Nummer kleiner als der Uno ist der Arduino Mikro (Bild 4). Dieser hat den Vorzug, dass er direkt in ein Breadboard gesteckt werden kann. Er eignet sich daher hervorragend für den Aufbau von Prototypen oder fertigen Kleingeräten. Im Rahmen dieser Artikelserie wird der Arduino Mikro noch bei verschiedenen Projekten zum Einsatz kommen.

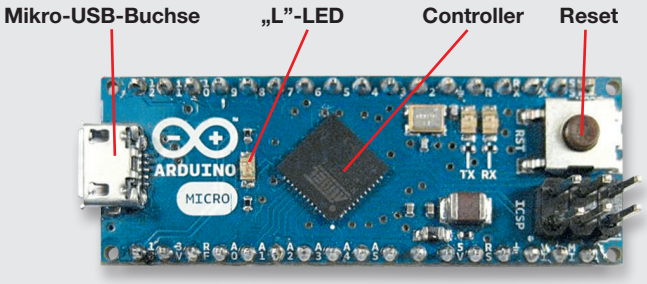

Bild 4: Arduino Mikro

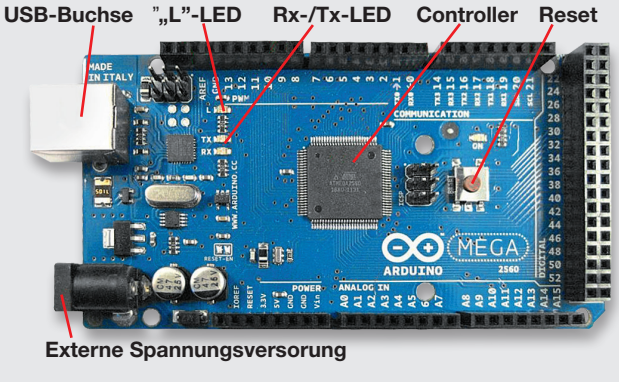

Bild 5: Arduino Mega

Bild 2: Selbst gefertigte Digitaluhr – aufgebaut nur mithilfe eines Mikrocontrollers

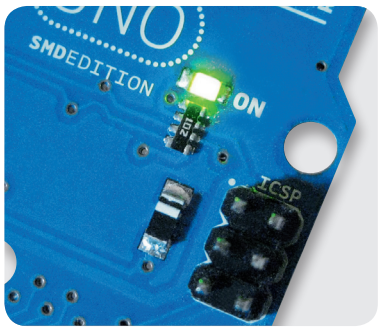

Bild 3: Das Board „steht unter Strom".

Falls mehr Rechenleistung oder eine höhere Anzahl von Ports benötigt wird, kann der Arduino Mega (Bild 5) eingesetzt werden. Dieser ist wesentlich größer als der Uno und bietet zusätzliche Pins für umfangreichere Projekte.

Ansonsten existiert noch eine Fülle weiterer Bauformen, wie etwa das runde Lillypad oder der Esplora, der bereits über mehrere fest verdrahtete Taster, Sensoren etc. verfügt. Weitere Details zu den einzelnen Boards finden sich unter [3].

Schneller Einstieg mit der Arduino-Entwicklungsumgebung

Neben der standardisierten Hardware liegt der große Vorteil des Arduino-Systems in der benutzerfreundlichen IDE. Im Gegensatz zu einer klassischen „Tool-Chain" kann diese ohne umfangreiche Vorkenntnisse bedient werden. Außerdem erfordert sie keine langwierige Installation. Die ersten Programme können so direkt nach dem Start der Entwicklungsumgebung auf den Mikrocontroller geladen werden.

Die aktuelle Version der IDE kann gratis heruntergeladen werden unter: arduino.cc/en/Main/Software

Es stehen verschiedene Versionen für die gebräuchlichsten Betriebssysteme (Windows, Mac OS und Linux) zur Verfügung. Für Linuxfreunde ist die Arbeit unter Ubuntu besonders einfach, da bereits bei der Systeminstallation ein Download-Icon im APPs-Bereich zur Verfügung gestellt wird. Die Bedienoberfläche der Ubuntu-IDE ist weitgehend identisch mit der Windows-Variante, sodass die hier dargestellte Vorgehensweise auch für Ubuntu anwendbar ist.

Bild 6: Typisches Arduino-Verzeichnis

Bild 7: Icon für den Start der Arduino-IDE

Bild 8: Leeres IDE-Fenster

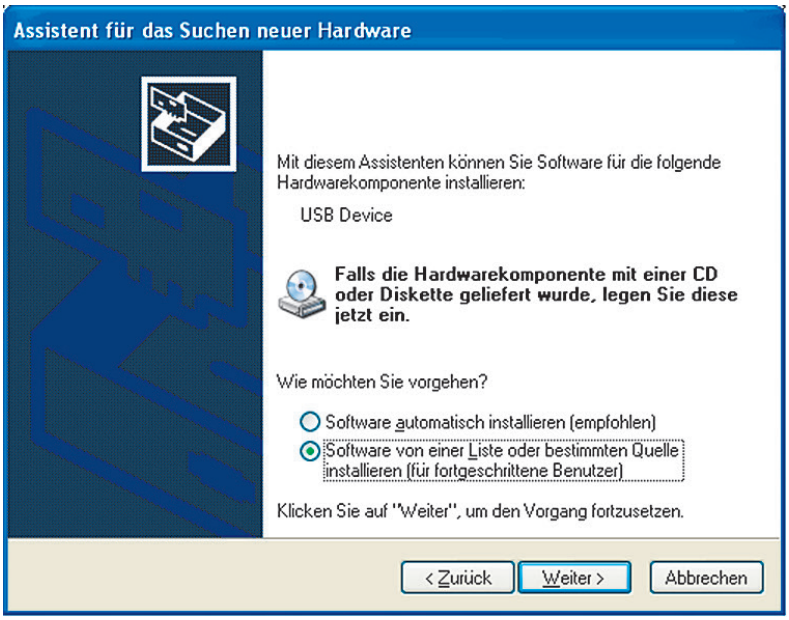

Bild 9: Aufforderung zur Treiberinstallation

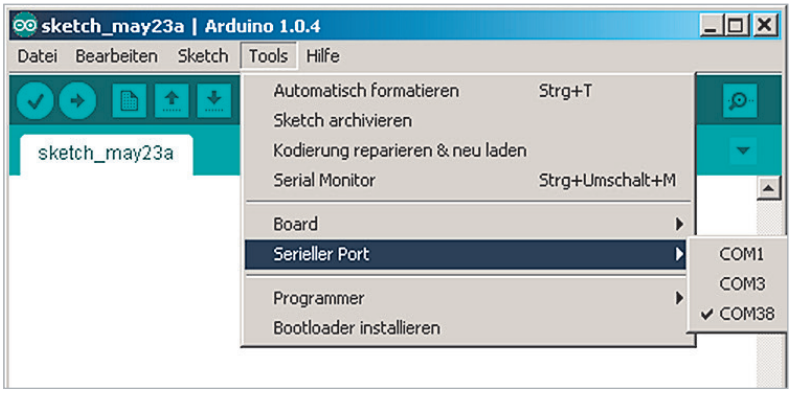

Bild 10: Auswahl des COM-Ports

Unter Windows liegen die vollständigen Programmpakete als komprimierte ZIP-Archive vor und können in ein beliebiges Verzeichnis extrahiert werden (Bild 6).

Das Verzeichnis enthält dann alle zur Programmierung erforderlichen Komponenten. Weiterhin werden unter „examples" noch verschiedene Beispielprogramme mitgeliefert.

Darüber hinaus können die Möglichkeiten des Arduinos mit sogenannten Bibliotheken ergänzt werden. Mehr zu diesem Thema findet sich in späteren Beiträgen dieser Artikelserie.

Nach dem Entpacken des ZIP-Archivs steht im Verzeichnis
..\arduino-1.0.x.
das Startprogramm für die IDE, arduino.exe, zur Verfügung (Bild 7).

Nach erfolgreichem Start erscheint kurz ein Informationsfenster mit den üblichen Angaben zu den Eckdaten der verwendeten Programmversion (Bild 8). Anschließend wird die Oberfläche der IDE angezeigt.

Jetzt kann der Arduino über ein USB-Kabel mit dem PC verbunden werden.

TIPP III: Das USB-Kabel sollte nicht zu lang sein. Kabellängen von mehr als 0,5 m können zu Übertragungsproblemen führen. Insbesondere der Arduino Micro reagiert auf längere Kabel sehr empfindlich und verweigert unter Umständen den Verbindungsaufbau.

Der Arduino meldet sich mit einer Aufforderung zum Installieren eines Treibers (Bild 9). Diese Installation ist nach wenigen Schritten abgeschlossen und nimmt nur einige Minuten Zeit in Anspruch. Falls es dabei wider Erwarten zu Problemen kommen sollte, finden sich unter [4] Hinweise zur Fehlerbehebung.

Nach erfolgreicher Treiberinstallation kann mit der IDE weitergearbeitet werden.

Zunächst muss die richtige Arduino-Boardvariante ausgewählt werden. Dies erfolgt unter den Menüeinträgen „Tools" und dann „Board".

Dann ist die Selektion des verwendeten virtuellen COM-Ports erforderlich. Meist ist dies die letzte Position in der Liste. Falls mehrere Ports angeboten werden, hilft das Durchtesten aller angezeigten COM-Nummern (Bild 10).

Alle für die Programmentwicklung notwendigen Steuersymbole finden sich unterhalb der Menüleiste am oberen Rand der IDE. Diese Icons werden im nächsten Beitrag der Artikelserie genauer erläutert. Zunächst sind nur die für das Laden eines ersten Programms erforderlichen Icons wichtig.

Mit dem Symbol „OPEN" wird ein bereits vorhandenes Programm geöffnet. Bitte wählen Sie hierfür den Sketch „blink.ino". Dieser ist unter examples\01.Basics\Blink zu finden.

Nach dem Laden des Programms sollte die IDE so aussehen wie in Bild 11.

Mit dem Button „UPLOAD" wird das Programm in den Speicher des Mikrocontrollers übertragen. Die mit

Rx/Tx bezeichneten LEDs leuchten dabei in unregelmäßigen Abständen auf und zeigen den Datenverkehr auf der seriellen Schnittstelle an. Gleichzeitig wird am PC die Meldung

Uploading to I/O Board

eingeblendet.

Der Abschluss der Übertragung wird mit der Nachricht

Done Uploading

angezeigt. Danach beginnt die LED „L" regelmäßig zu blinken und die beiden Rx-/Tx-LEDs erlöschen (Bild 12).

Nach Abschluss der Datenübertragung wird also ein Reset durchgeführt und automatisch mit der Ausführung des Programms „Blink" begonnen.

Damit wurde bereits ein erstes Programm auf einen Mikrocontroller übertragen, und weiteren Projekten zum Thema Mikrocontroller und Arduino steht nichts mehr im Wege.

TIPP IV: Weitere Details zur Inbetriebnahme und zur Fehlerbeseitigung finden sich auch im Mikrocontroller-Onlinekurs [5].

Ausblick

Im nächsten Artikel zu dieser Serie werden die Details zur Arduino-Entwicklungsumgebung genauer besprochen. Daneben wird das Thema Fehlersuche und -beseitigung im Vordergrund stehen.

Im Praxisteil entstehen dann bereits erste eigene Programme wie ein Alarmanlagensimulator oder eine Morsesignalbake. ELV

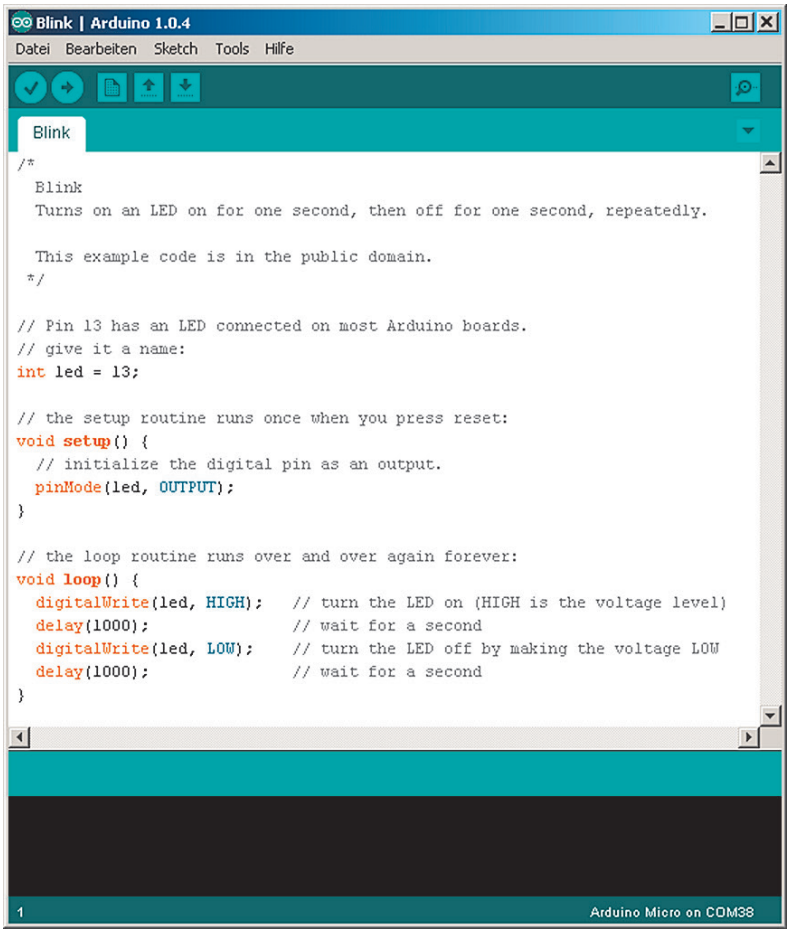

Bild 11: Blink.ino ist geladen.

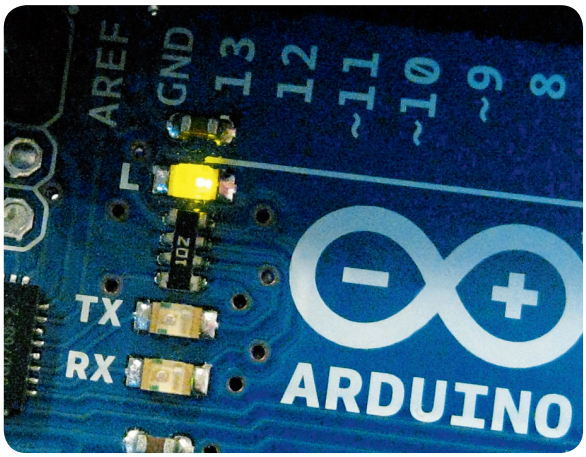

Bild 12: Die LED „L" blinkt im Sekundentakt.

Weitere Infos:

[1] G. Spanner: AVR-Mikrocontroller in C programmieren, Franzis-Verlag, 2010

[2] G. Spanner: Arduino – Schaltungsprojekte für Profis, Elektor-Verlag, 2012

[3] arduino.cc/en/Main/Products

[4] forum.arduino.cc/

[5] Mikrocontroller-Onlinekurs, Franzis-Verlag, exklusiv für ELV, 2011

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
Arduino-Uno-Platine R3	JY-10 29 70	€ 29,95
Arduino Micro	JY-10 97 74	€ 24,95
Arduino-Mega-2560-Platine	JY-10 69 18	€ 49,95
Hama-USB-Hub, 7fach mit Netzteil	JY-08 25 54	€ 24,95
USB-2.0-Verbindungskabel A auf B	JY-09 55 56	€ 0,95

Alle Infos zu den Produkten/Bauteilen finden Sie im Web-Shop.

Preisstellung August 2013 – aktuelle Preise im Web-Shop