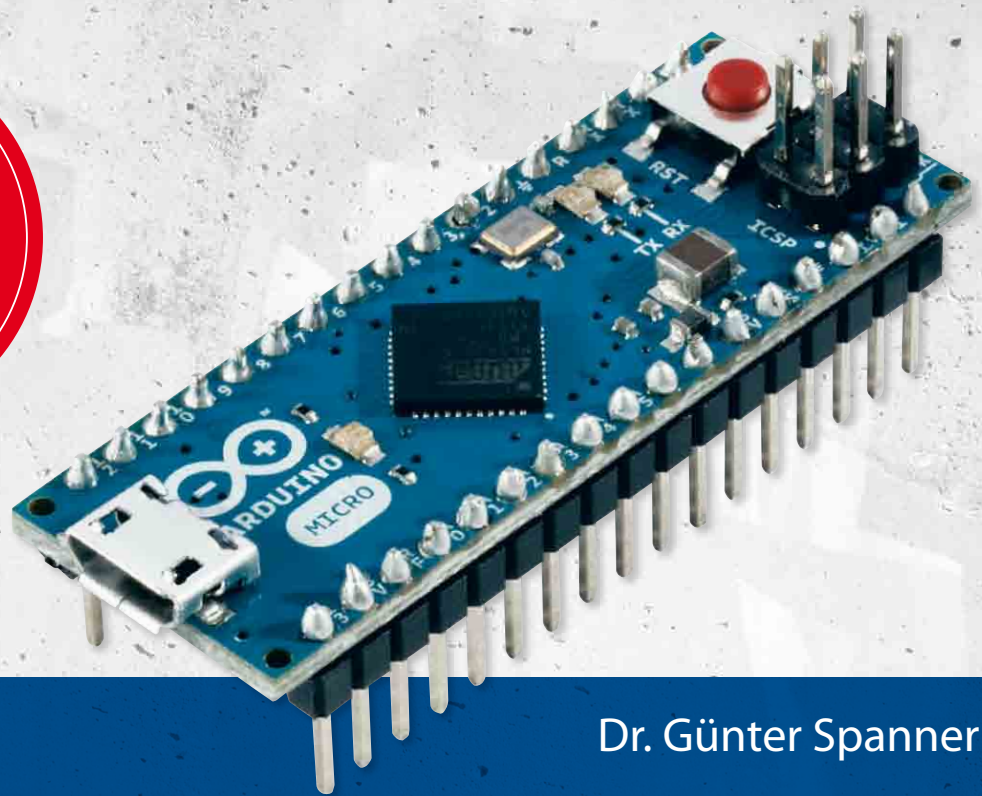


Komplett
in Farbe

Faszinierende
Fotos

Doppelseitige
Infografiken



Dr. Günter Spanner

COOLE PROJEKTE MIT DEM ArduinoTM Micro

- Physical Computing im Projekteinsatz
- Alles für den Einstieg: Platine, ATmega32U4, USB-Zugriff und Programmierung.
- Projekte für die Praxis: Lichteffekte, Designeruhr, Motorensteuerung, Messen, Steuern und Regeln

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.

Hinweis: 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.

© 2014 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.

Programmleitung: Dr. Markus Stäuble
Satz & Layout: DTP-Satz A. Kugge, München
art & design: www.ideehoch2.de
Druck: FIRMENGRUPPE APPL,
aprinta druck GmbH, Wemding

ISBN 978-3-645-65229-2



Vorwort

Der aus dem Leonardo entstandene Arduino Micro zeichnet sich besonders durch seine kompakte Bauform aus. Kaum größer als ein klassisches IC kann der Baustein direkt in ein Breadboard gesteckt werden. Das ist ein erheblicher Vorteil gegenüber den älteren Varianten der Arduino-Familie wie etwa dem UNO oder dem Mega.

Der Micro kann so unmittelbar mit allen gängigen elektronischen Bauelementen verbunden werden. Spezielle Protoboards oder aufsteckbare »Shields« sind nicht erforderlich. Im Rahmen des Buchs kann diese Arduino-Variante daher als nahezu professionelles Modul betrachtet werden, da der Micro das den anderen Arduinos anhaftende Image eines speziellen »Lernsystems« abgelegt hat.

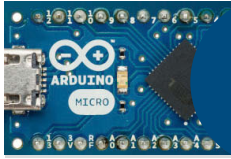
Aufgrund seiner Pinanordnung ist der Micro auch direkt in eine Lochrasterplatte oder sogar in eine IC-Fassung einsteckbar. Ein Schwerpunkt des Buchs liegt daher in der Beschreibung, wie fertig ausgetestete Laboraufbauten schnell und effizient in professionelle und dauerhaft ausgelegte Prototypen oder sogar fertige Kleingeräte integriert werden können.

Trotz seiner kompakten Bauform bringt der Baustein alle erforderlichen Komponenten mit. Über eine integrierte Micro-USB-Buchse kann der Controller mit den gewohnten, vielfach bewährten Arduino-Programmierertools mit Software versorgt werden. Darüber hinaus wird im Buch aber auch die Programmierung in C über die professionelle Tool-Chain des Controllerherstellers Atmel erläutert.

Durch die Integration der USB-Schnittstelle in den Hauptprozessor konnte ein separater USB-zu-RS-232-Wandler eingespart werden. Durch den Wegfall dieses meist recht hochpreisigen Bausteins ist der Micro deutlich günstiger erhältlich als seine Vorgänger. Das trägt dazu bei, dass der Micro direkt, sozusagen als »komplettes elektronisches Bauteil«, in den im Buch beschriebenen Projekten eingesetzt werden kann.

Ein weiterer Vorteil des im Prozessor integrierten USB-Anschlusses ist, dass der Baustein auch als sogenanntes USB-Device/HID (Human Interface Device) konfiguriert werden kann. Damit ist es möglich, den Micro so zu programmieren, dass er sich wie eine USB-Maus oder -Tastatur verhält. Einem direkten Schreiben von Daten in die bekannten Anwenderprogramme wie etwa Word oder Excel steht somit nichts mehr im Weg. Im Buch wird dieser Technik und der Fülle der damit möglichen Anwendungen ein eigenes Kapitel gewidmet.

Korrekturen, Ergänzungen oder Anregungen werden gern unter der Adresse elo@franzis.de angenommen.



1

1	Einführung	12
1.1	Arduino Micro – ein Controllersystem in IC-Größe.....	12
1.2	Der Arduino Micro auf einen Blick	14
1.3	Elektronische Aufbausysteme	15
1.4	Breadboard-Betrieb des Arduino Micro	17
1.5	Die Stromversorgung für den Micro	20
1.6	Grundausstattung eines Arduino-Arbeitsplatzes	23

2

2	Die Technik des ATmega32U4	26
2.1	Rückblick auf die Mikrocontrollerentwicklung.....	26
2.2	Funktionseinheiten des ATmega32U4	27

3

3	Direkter Prozessorzugriff über USB	34
3.1	Bootloader und USB-Schnittstelle	34
3.2	Grundlagen der USB-Technik.....	35
3.3	Die USB-Hardwareschnittstelle des ATmega32U4.....	35
3.4	Enumeration	37

4

4	Programmierung über die Arduino-IDE	38
4.1	Starten und Konfigurieren der IDE	38
4.2	Die Benutzeroberfläche der Arduino-IDE	44
4.3	Die Standardbeispielprogramme der IDE.....	45
4.4	Fehlerbeseitigung.....	45
4.5	Das erste Praxisprojekt: vollautomatische Eieruhr	46
4.6	Vollautomatische Eieruhr mit Batteriebetrieb	49

5

5	Professionelle Programmierung in C.....	52
5.1	Erstellung von C-Programmen mit dem Atmel Studio	52
5.2	Upload von .hex-Dateien auf den Arduino	55
5.3	.hex-Dateien aus der Arduino-IDE.....	58

6

6	Ansteuerung der digitalen Input/Output-Pins	62
6.1	Powerstroboskop mit Leistungstransistor.....	62
6.2	Stroboskop mit variabler Blitzfrequenz	63
6.3	LED-Würfel mit Ausrolleffekt	64



- 7 Die Timer und Interruptfunktionen des ATmega32U4.....70
 - 7.1 Quarzgenaue Timersteuerung 70
 - 7.2 Designerruhr mit LED-Display 71

- 8 Direkte Datenausgabe an Word oder Excel.....78
 - 8.1 Datentransfer via USB 79
 - 8.2 Kommunikation zwischen Host und USB-Gerät 80
 - 8.3 Stromversorgung über USB 82
 - 8.4 Softwarestruktur einer USB-Verbindung 83
 - 8.5 Der Arduino als USB-Tastatur..... 84
 - 8.6 Direkte Ausgabe von Daten an Excel 84
 - 8.7 Mauszeiger außer Kontrolle!..... 87
 - 8.8 Arduino steuert die Computermaus..... 87

- 9 Periphere Komponenten und Physical Computing 92
 - 9.1 Displaytechnik 93
 - 9.2 Gleichstrommotorsteuerung..... 96
 - 9.3 Vollautomatische Ventilatorsteuerung..... 97
 - 9.4 Schrittmotoren 99
 - 9.5 Prinzipieller Aufbau eines Schrittmotors..... 100
 - 9.6 Elektrische Ansteuerung von Schrittmotoren 102
 - 9.7 Softwaresteuerung für Schrittmotoren..... 106
 - 9.8 Schrittmotorgesteuerter Teeautomat 108
 - 9.9 Solarzellennachführung 110
 - 9.10 Servomotoren 112
 - 9.11 Servoansteuerung..... 112
 - 9.12 Die Servobibliothek 114
 - 9.13 Megadisplay mit Servomotor..... 115

- 10 Analog-digital-Konverter und Komparatoren 120
 - 10.1 Erfassung von Analogmesswerten..... 120
 - 10.2 ADC-Wandlerverfahren 121
 - 10.3 Messung einer Potenziometerposition 122
 - 10.4 Temperaturmessung 123
 - 10.5 Maussteuerung via analogem Joystickmodul 126
 - 10.6 Elektronische Wasserwaage 129



	11	Hightech-Applikationen für den Arduino Micro	134
	11.1	Entfernungsmesser mit Lasertargetindikator	134
	11.2	Excel-Datenlogger für Umweltmessdaten	137
	11.3	Datenlogger für Temperaturwerte	138
	11.4	Thermograf mit Megadisplay	140
	12	Arduino Micro in professionellen Entwicklungsprojekten	144
	12.1	Gehäuse	144
	12.2	Lochrasterplatten	144
	13	Include-Bibliotheken	146
	13.1	Ansteuerung von 7-Segment-LED-Anzeigen	146
	14	Befehlsreferenz für Processing	150
	14.1	Strukturen in Processing und C.....	150
	14.2	Syntaxelemente im Überblick.....	151
	14.3	Operatoren.....	152
	14.4	Konstanten.....	154
	14.5	Definition von Variablen	155
	14.6	Variablenfelder.....	156
	14.7	Kontrollstrukturen	156
	14.8	Spezielle Funktionen.....	160
	14.9	Zeitliche Steuerung von Programmabläufen	160
	14.10	Mathematische und trigonometrische Funktionen.....	161
	14.11	Befehle für Maus- und Tastatursteuerung	161
	14.12	Eigene Funktionen.....	163
	14.13	Parameterübergabe	163
	15	Embedded C	164
	15.1	Bitmanipulation und Bitmasken.....	164
	15.2	Ansteuerung von IO-Ports.....	166
	15.3	Bitnummern für GPIO-Register	167
	15.4	Analoge Messwerterfassung	168
	15.5	Warteschleifen (delay.h).....	170
	15.6	Interrupts	171
	15.7	Counter und Timer.....	173

16 **Fehlersuche**..... 174
16.1 Allgemeine Hardwareaufbauten 174
16.2 Mikrocontrollerschaltungen..... 174
16.3 Programmentwicklung und Programmierung..... 175



17 **Literatur** 180
18 **Bezugsquellen** 180
19 **Quellcodes**..... 181
Index..... 182



Tip

Die verschiedenen Versionen der Arduino-IDEs sind nicht immer zu 100 % kompatibel. Es kann durchaus vorkommen, dass ein Sketch, der mit einer Version einwandfrei läuft, mit einer späteren Version nicht mehr ordnungsgemäß arbeitet. Man sollte daher immer im Auge behalten, welcher Sketch mit welcher IDE-Version erstellt wurde.

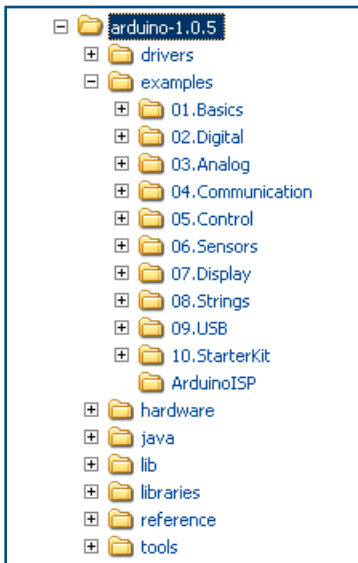


Abb. 4.1: Typisches Arduino-Verzeichnis



Abb. 4.2: Über dieses Icon wird die Arduino-IDE gestartet

4 Programmierung über die Arduino-IDE

Die einfachste Möglichkeit, den Arduino Micro zu programmieren, besteht in der Verwendung der Arduino-IDE. Sie kann in der jeweils neuesten Version von der Internetseite <http://arduino.cc/en/Main/Software> geladen werden.

Die in diesem Buch vorgestellten Sketche wurden alle mit der Version 1.0.5 erstellt. Aus Kompatibilitätsgründen ist es also vorteilhaft, mit dieser Version zu arbeiten. Natürlich kann man auch neuere Versionen verwenden. Falls es aber zu unvorhergesehenen Problemen kommen sollte, ist es immer eine gute Idee, den fraglichen Sketch zunächst mit der Version 1.0.5 zu testen. Oftmals klären sich so offene Fragen von selbst.

4.1 Starten und Konfigurieren der IDE

Für alle gebräuchlichen Betriebssysteme (Windows, MAC OS und Linux) steht eine eigene IDE-Variante zur Verfügung. Unter Ubuntu wird bei der Systeminstallation sogar ein Downloadicon im Apps-Bereich zur Verfügung gestellt. Die Bedienungsoberfläche der Ubuntu-IDE ist dann weitgehend identisch mit der Windows-Version, sodass die hier beschriebene Vorgehensweise auch für Ubuntu anwendbar ist.

Für Windows liegen die vollständigen Programmpakete als komprimierte Ziparchive vor und können in ein beliebiges Verzeichnis extrahiert werden. Eine langwierige Installation ist nicht erforderlich.

Alternativ werden bei neueren IDEs auch fertige Installationspakete (z. B. *arduino-1.0.5-windows.exe*) angeboten. Natürlich kann man auch damit die IDE problemlos auf einem Rechner installieren. Jeder Anwender kann selbst entscheiden, welche Vorgehensweise er wählt.

Nach dem Entpacken und der Installation ist auf der Festplatte ein vollständiges Arduino-Verzeichnis vorhanden.

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. Das Startprogramm für die IDE »*arduino.exe*« findet sich im Verzeichnis `...\\arduino-1.0.x`.

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

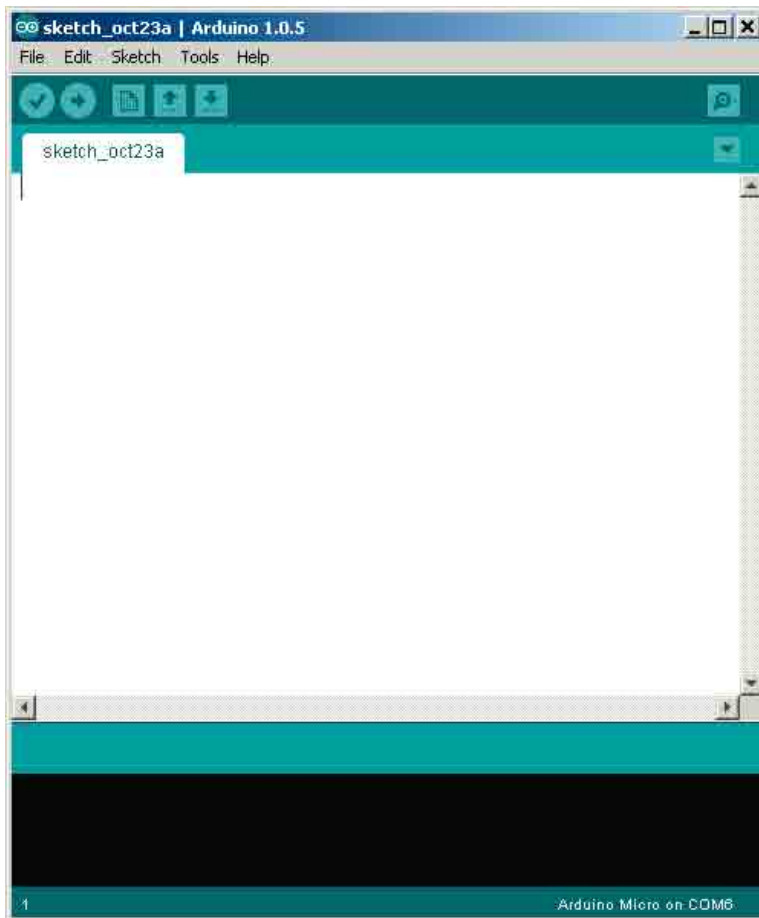


Abb. 4.3: Nach dem Start erscheint ein leeres IDE-Fenster

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

Der Arduino meldet sich mit einer Aufforderung zum Installieren eines Treibers. 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 im Arduino-Forum Hinweise zur Fehlerbehebung.

Tipp

Das USB-Kabel sollte nicht zu lang sein. Kabellängen von mehr als 0,5 m können zu Übertragungsproblemen führen. Der Arduino Micro reagiert auf längere Kabel besonders empfindlich, da die USB-Schnittstelle direkt mit dem Controller verbunden ist. Ist das Kabel zu lang, werden elektromagnetische Störungen eingefangen. Der Micro verweigert unter Umständen den Verbindungsaufbau.

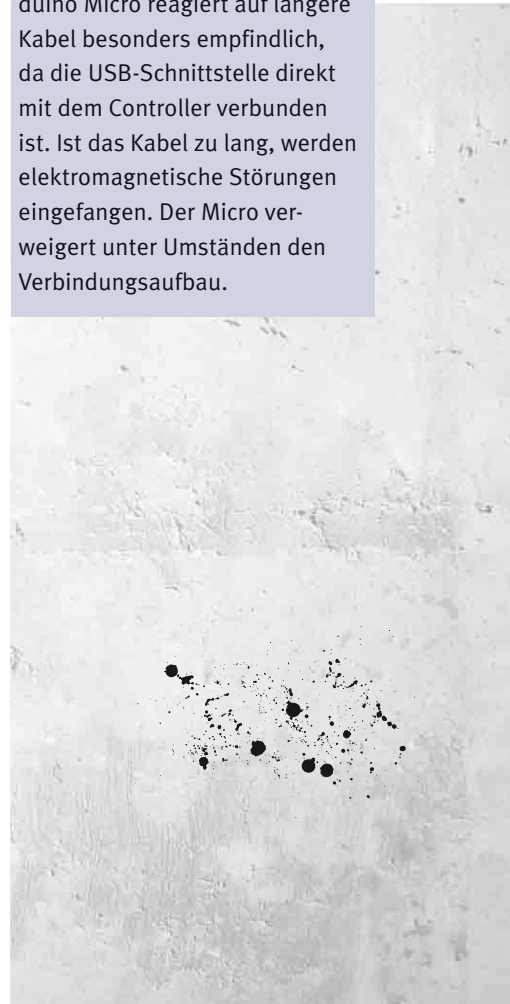




Abb. 4.4: Aufforderung zur Treiberinstallation

Hier ist zu beachten, dass die zweite Auswahl aktiviert wird. Dann wird der Treiber gemäß der folgenden Abbildung ausgewählt.

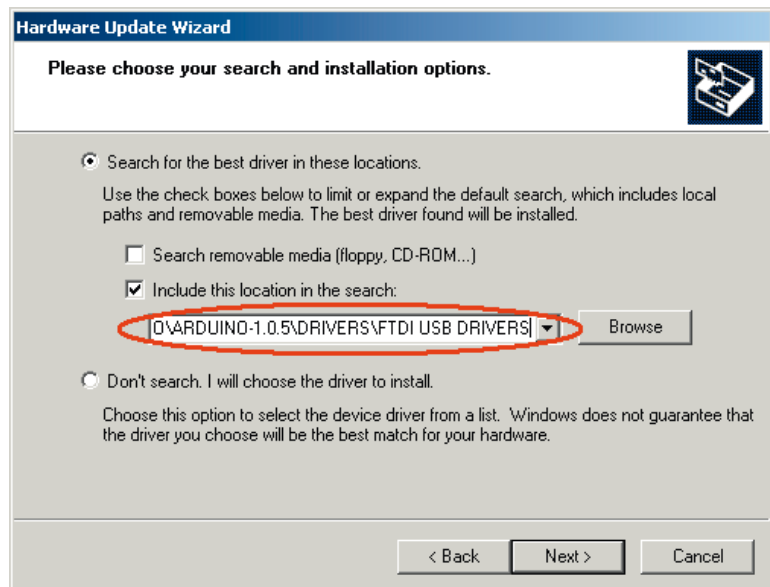


Abb. 4.5: Auswahl des Treibers

Ist der Treiber erfolgreich installiert, kann man im Geräte-Manager überprüfen, unter welchem virtuellen COM-Port der Arduino Micro nun angesprochen werden kann. Es ist vorteilhaft, sich diese COM-Nummer gut zu merken, da sie bei der Konfiguration der IDE wieder benötigt wird.

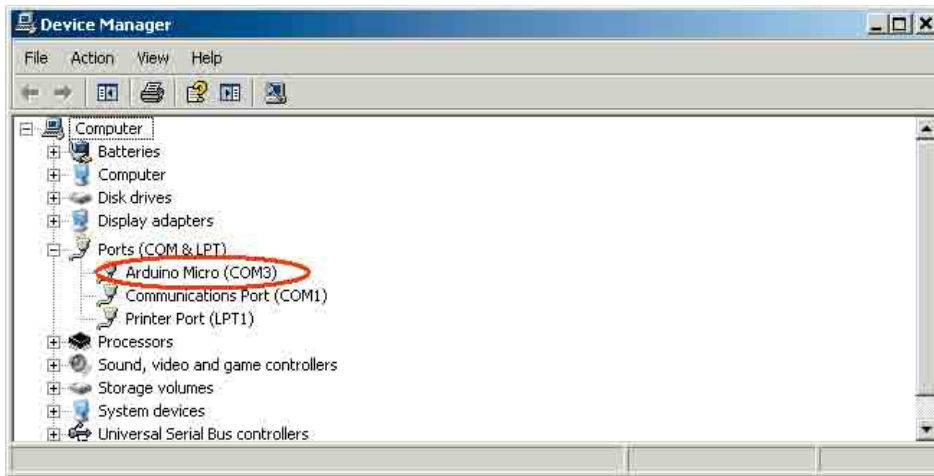


Abb. 4.6:
Arduino Micro im Geräte-Manager

Zunächst muss der richtige Arduino-Boardtyp ausgewählt werden. Das erfolgt unter den Menüeinträgen *Tools* und dann *Board*.

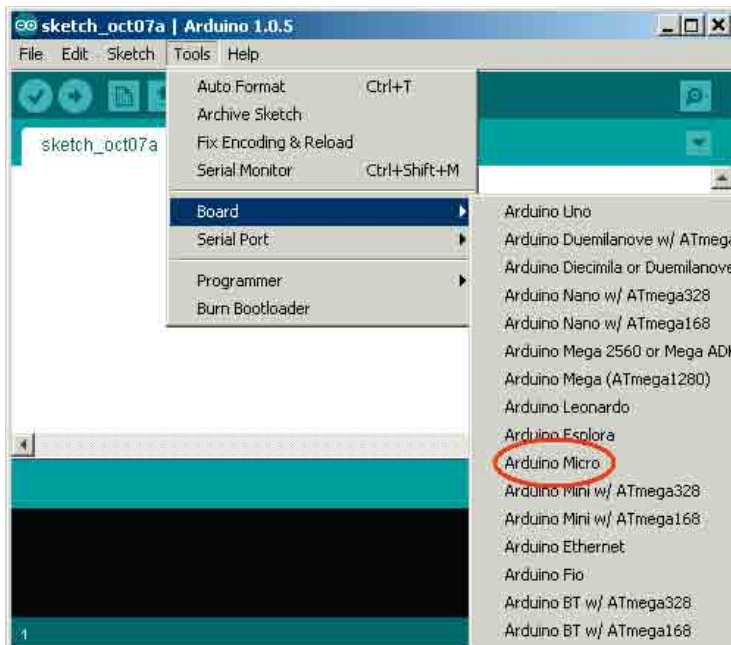


Abb. 4.7: Auswahl des Boardtyps

Dann ist die Selektion des verwendeten virtuellen COM-Ports erforderlich. Meist ist es die letzte Position in der Liste. Falls mehrere Ports angeboten werden und die Nummer nicht aus dem Geräte-Manager bekannt ist, hilft auch einfach das Durchtesten aller angezeigten COM-Nummern.

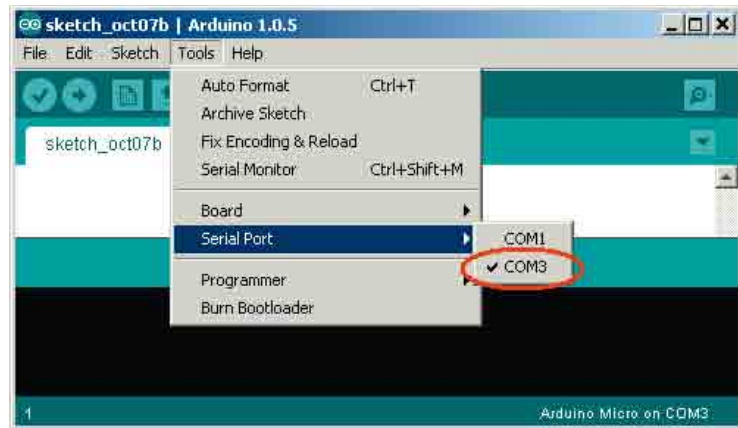


Abb. 4.8: Auswahl des COM-Ports

Alle für die Programmentwicklung notwendigen Steuersymbole finden sich unterhalb der Menüleiste am oberen Rand der IDE. Diese Icons werden noch 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 das Programm *blink.ino*. Nach dem Laden des Programms sollte die IDE so aussehen wie in der folgenden Abbildung.

Über den 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 gelbe LED regelmäßig zu blinken und die beiden RX/TX-LEDs erlöschen.

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 steht nichts mehr im Weg.

The screenshot shows the Arduino IDE interface with the 'Blink' sketch loaded. The code in the editor is as follows:

```

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

  This example code is in the public domain.
  */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
    
```

The status bar at the bottom of the IDE indicates: "1 Arduino Diecimila or Duemilanove w/ ATmega168 on COM27".

Abb. 4.9:
blink.ino ist geladen



Abb. 4.10: LED ist aktiv!

4.2 Die Benutzeroberfläche der Arduino-IDE

Die Bedeutungen der Symbole am oberen Rand der IDE können der folgenden Abbildung entnommen werden.

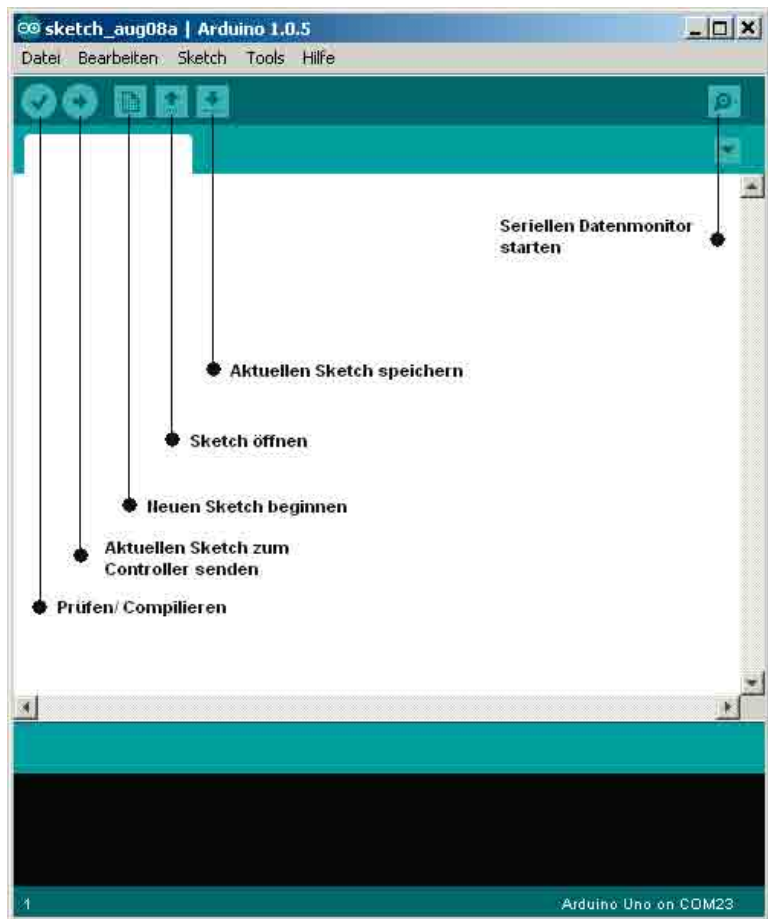


Abb. 4.11: Die Funktionen der Icons

Prüfen/Compilieren: Mit dieser Schaltfläche kann der aktuell geöffnete Sketch kompiliert, d. h. in Maschinensprache übersetzt, werden. Das ist insbesondere nützlich, wenn man ein Programm z. B. auf Syntaxfehler hin testen möchte, ohne dass ein Arduino-Board mit dem PC verbunden ist.

Aktuellen Sketch zum Controller senden: Dieses Icon kompiliert und sendet den aktuellen Sketch direkt zum Arduino. Ein vorheriges separates Übersetzen ist also nicht erforderlich.

Neuen Sketch beginnen: Hier kann ein neuer Sketch begonnen werden.

Sketch öffnen: Mit diesem Icon wird ein bereits abgespeicherter Sketch in die IDE geladen.

Aktuellen Sketch speichern: Das Speichern eines Sketches erfolgt mit diesem Icon.

Seriellen Datenmonitor starten: Hier kann der serielle Datenmonitor gestartet werden. Er dient zur Kommunikation zwischen Controller und PC und wird häufig auch zu Test- und Prüfzwecken eingesetzt.

4.3 Die Standardbeispielprogramme der IDE

Die Standard-IDE enthält bereits eine umfangreiche Sammlung von Beispielprogrammen.

Es ist sicher immer eine gute Idee, diese Beispielprogramme genauer unter die Lupe zu nehmen. Häufig findet sich hier bereits eine geeignete Basis für neue Projekte. Die Erfahrung zeigt sogar, dass auch der fortgeschrittene Programmierer bei der Entwicklung komplexer Projekte häufig einen Einstieg über diese Beispielprogramme findet.

Die unten stehenden ersten eigenen Programme werden auch aus diesen Standardbeispielen abgeleitet. Wie Abbildung 4.1 zeigt, kann man die Beispielprogramme auch problemlos in der Verzeichnisstruktur wiederfinden.

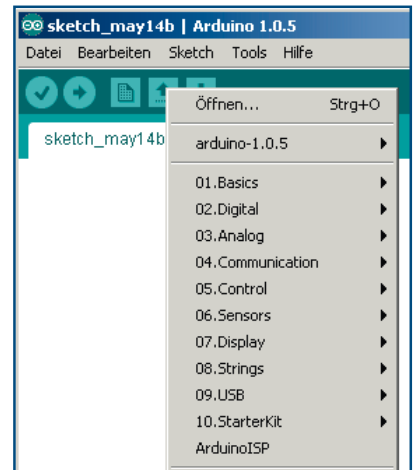


Abb. 4.12: Die Standardbeispielprogramme

4.4 Fehlerbeseitigung

Im Allgemeinen sollte das Übertragen eines Sketchs zum Arduino Micro keine Probleme bereiten. Wurde ein Sketch mit Erfolg übertragen, wird das mit der folgenden Meldung quittiert:

```
Upload abgeschlossen.
Binäre Sketchgröße: 1.084 Bytes (von einem Maximum von 14.336 Bytes)
```

Abb. 4.13: Upload ok

Obwohl die Arduino-IDE benutzerfreundlich ist, können Probleme nicht immer ausgeschlossen werden. Der häufigste Fehler ist die Auswahl des falschen seriellen Ports.

```
Upload abgeschlossen.
Binäre Sketchgröße: 1.084 Bytes (von einem Maximum von 14.336 Bytes)
avrdude: stk500_getsync(): not in sync: resp=0x00
```



Abb. 4.14: Fehlermeldung bei falschem Port

An zweiter Stelle der Fehlerstatistik steht die Auswahl des falschen Boardtyps.

Abb. 4.15:
Fehlermeldung
bei falsch gewähl-
ter Boardversion

```
Es wurde ein falscher Microcontroller gefunden. Haben Sie den richtigen aus dem Menü Tools > Board ausgewählt?
Binäre Sketchgröße: 882 Bytes (von einem Maximum von 7.168 Bytes)
avrdude: Expected signature for ATMEGA8 is 1E 93 07
Double check chip, or use -F to override this check.
```

Tip

Falls der Arduino einmal nicht mehr ansprechbar sein sollte, hilft es meist, die USB-Verbindung zu lösen und das Board erneut mit dem Computer zu verbinden.

Man erkennt, dass die Fehlermeldungen oft nicht sehr spezifisch sind. Allerdings geben Sie in den meisten Fällen brauchbare Hinweise auf die Ursache des Problems.

Neben diesen beiden häufigsten Fehlern treten in der Praxis öfter auch »seltsame« Fehler ohne definierte Ursache auf. Oft hilft es dann, den zuletzt ausgeführten Vorgang erneut zu starten.

4.5 Das erste Praxisprojekt: vollautomatische Eieruhr

Als erste Praxisanwendung entsteht eine vollautomatische Eieruhr. Eine solche Uhr kann in jedem Haushalt nutzbringend eingesetzt werden. Die Ablaufzeit der Uhr kann über die Anpassung des Sketchs in weiten Grenzen variiert werden. So kann man die optimale Zeit für das Frühstücksei individuell einstellen. Da 10 LEDs zur Verfügung stehen, wählt man für das klassische 4-Minuten-Ei eine Zeitspanne von 0,4 Minuten, d. h. also 24 Sekunden, pro LED.

Natürlich ist die Uhr auch für andere Anwendungen einsetzbar. So gibt es etwa zeitkritische Phasen beim Anrühren von bestimmten Klebstoffen. Wenn die Uhr mit entsprechenden Zeitintervallen programmiert wird, kann sie auch hier gute Dienste leisten.

Erforderliche Bauelemente

- 1 x Arduino Micro
- 1 x Breadboard
- 10 x 5-V-LEDs (z. B. 5 x rot, 3 x gelb, 2 x grün)
- 1 x Tiltsensor
- Drahtbrücken

Um die Uhr weitgehend zu automatisieren, kommt ein sogenannter Tilt- oder Kippsensor zum Einsatz. Mithilfe dieses Bauelements kann die Eieruhr ohne weitere Bedienelemente auskommen. Das staubdicht abgeschlossene Gehäuse enthält zwei oder mehr Kontaktstifte und eine leitfähige Metallkugel. Steht der Sensor aufrecht, stellt die Metallkugel eine

leitende Verbindung zwischen den Kontaktstiften her. Wird der Sensor gekippt, rollt die Kugel im Gehäuse nach unten und der elektrische Kontakt wird unterbrochen.

Die Schaltung zur Eieruhr zeigt Abb. 4.20. Wichtig ist zu beachten, dass es sich bei den eingesetzten LEDs um 5-V-Typen handelt. Nur diese dürfen so wie hier gezeigt ohne Vorwiderstand eingesetzt werden. Prinzipiell könnte man natürlich auch gewöhnliche LEDs verwenden, allerdings muss dann jede LED mit einem Vorwiderstand von mindestens 150 Ohm (Ω) versehen werden. Damit wäre aber ein so kompakter Aufbau wie in Abb. 4.21 nicht mehr möglich.

Der Tiltsensor muss so ausgerichtet werden, dass er bei horizontaler Lage der Eieruhr keinen elektrischen Kontakt herstellt. Wird dann die Eieruhr senkrecht gestellt, schließt der Tiltsensor den Kontakt und sie beginnt automatisch zu laufen. Nach Ablauf der Uhr kann sie dann einfach wieder horizontal abgelegt werden. Damit wird sie wieder ausgeschaltet. Auf diese Weise werden keinerlei Bedienelemente wie Taster oder Schalter benötigt.

Die Versorgung der Uhr erfolgt über den USB-Stecker.

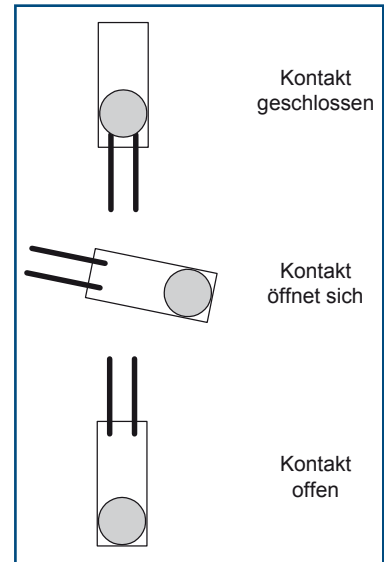


Abb. 4.16: Prinzipieller Aufbau eines Tiltsensors

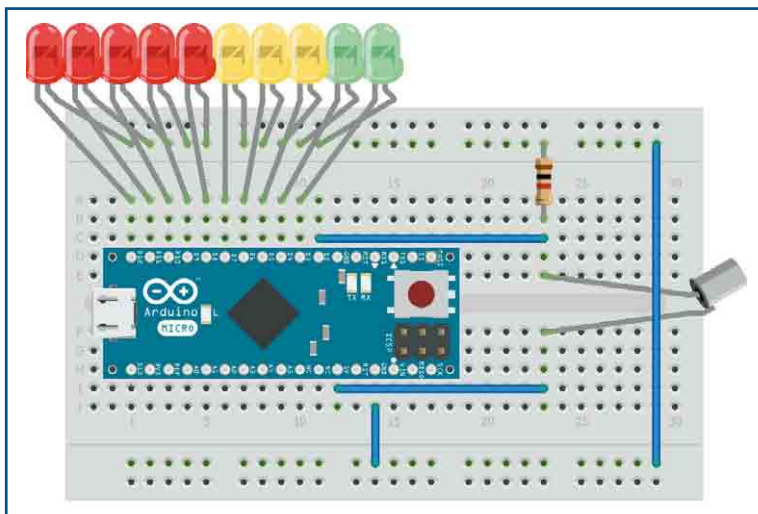


Abb. 4.18: Schaltung zur automatischen Eieruhr

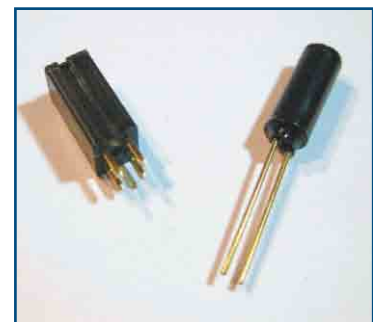


Abb. 4.17: Tiltsensoren in verschiedenen Ausführungen

Das Programm zur Eieruhr ist nachfolgend aufgeführt. Zunächst werden im Setup die Ports für die LEDs als Ausgänge deklariert:

```
for(int i=3 ; i<=12; i++) pinMode(i, OUTPUT);
```



Pin 2 wird für die Abfrage des Tiltsensors als Eingang deklariert:

```
pinMode(2, INPUT);
```

In der Hauptschleife des Programms werden dann die LEDs nacheinander im Abstand von 24 Sekunden eingeschaltet:

```
for(int i=12; i>=3; i--)
{ digitalWrite(i, HIGH); delay(24000);
```

Sketch: automatische Eieruhr (Egg_timer.ino)

```
// egg timer

void setup()
{ // set LED ports as output
  for(int i=3 ; i<=12; i++) pinMode(i, OUTPUT);
  // set tilt sensor port as input
  pinMode(2, INPUT);
}

void loop()
{ // wait for tilt sensor
  while (digitalRead(2)== LOW);

  // set LEDs on bottom to top to left
  for(int i=12; i>=3; i--)
  { digitalWrite(i, HIGH); delay(24000);
  }

  // flash all LEDs when ready
  while (1)
  { for(int i=3; i<=12; i++) digitalWrite(i, HIGH);
    delay(300);
    for(int i=3; i<=12; i++) digitalWrite(i, LOW);
    delay(300);
  }
}
```

Abschließend, d. h. wenn die Zeit von 10×24 Sekunden = 240 Sekunden = 4 Minuten vollständig abgelaufen ist, werden alle LEDs simultan ein- und ausgeschaltet. So ergibt sich ein unübersehbares Zeichen dafür, dass die vorgegebene Zeit abgelaufen ist.

4.6 Vollautomatische Eieruhr mit Batteriebetrieb



In der vorhergehenden Version musste die Eieruhr über das USB-Kabel mit Spannung versorgt werden. Wenn man die Eieruhr unabhängig von einem PC oder Laptop einsetzen will und auch kein passendes USB-Netzteil mit Micro-USB-Stecker zur Verfügung steht, kann die Eieruhr auch völlig autark mit einer Batterie betrieben werden.

Zusätzlich erforderliche Bauelemente

- 1 x 9-V-Blockbatterie
- 1 x 9-V-Batterieclip

Besonders einfach ist die Verwendung einer 9-V-Blockbatterie. Sie kann über einen 9-V-Batterieclip kostengünstig mit dem Arduino Micro verbunden werden. Möglich ist das, da der Micro über einen integrierten Spannungsregler verfügt.

Die Batterie muss folgendermaßen mit dem Micro verbunden werden:

-  Batterie-Minus an GND
-  Batterie-Plus an V1

Warnung

Beim Anschluss der 9-V-Batterie ist mit höchster Sorgfalt darauf zu achten, dass die korrekten Pins beschaltet werden. Wird die 9-V-Spannung auch nur kurzzeitig mit einem falschen Pin verbunden, kann das zur Zerstörung des Arduinos führen!

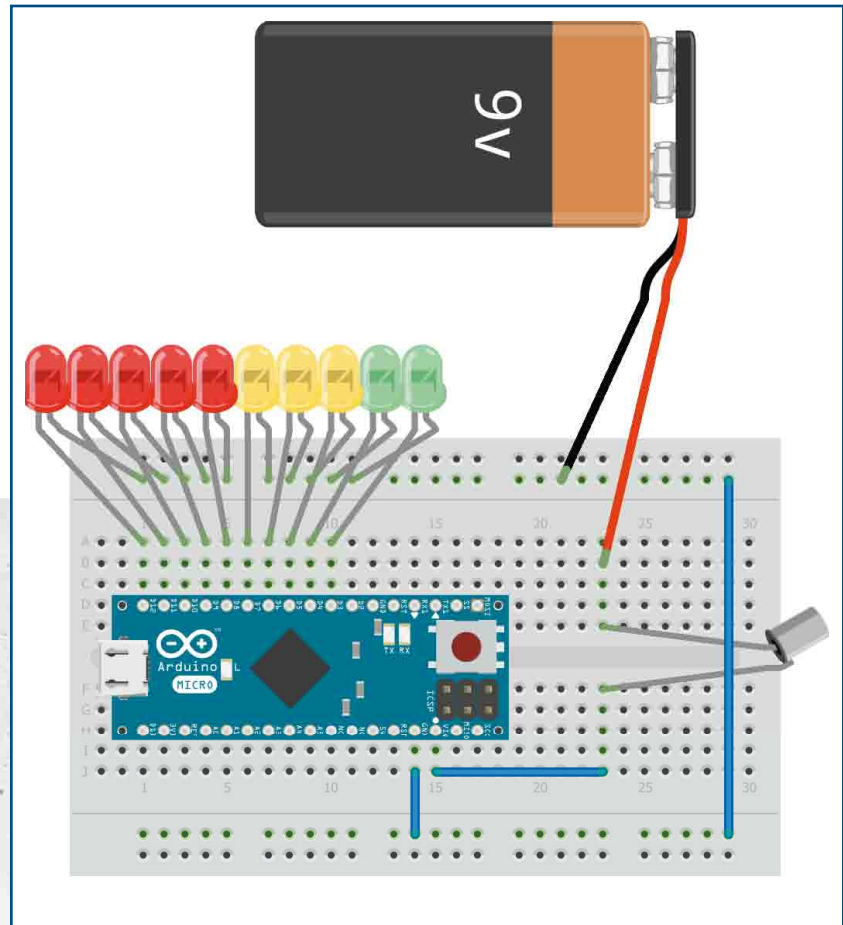


Abb. 4.19: Schaltung zur automatischen Eieruhr mit Batteriebetrieb

Im Unterschied zum vorhergehenden Sketch kann nun auf die Abfrage des Tiltensors verzichtet werden (siehe Sketch *Egg_timer_battery.ino*).

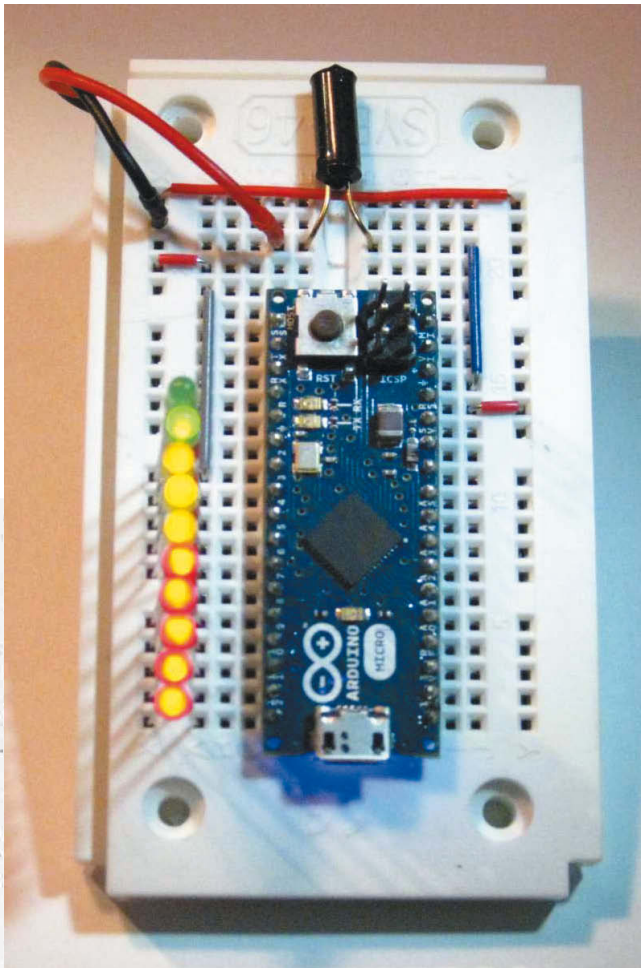


Abb. 4.20: Automatische Eieruhr



KAPITEL **A**

ANHANG

Literatur	180
Bezugsquellen	180
Quellcodes	181
Index	182

17 Literatur

Die folgende Literatur bietet Hilfe, falls Grundlagen im Bereich der Elektronik oder C-Programmierung fehlen. Andererseits bieten sich einige Werke auch als Weiterführung und Ergänzung zum vorliegenden Buch an.

Wer insbesondere die Kapitel der unten genannten Lernpakete sorgfältig durchgearbeitet hat, ist auf den entsprechenden Gebieten kein Anfänger mehr. Er ist durchaus in der Lage, weiterführende Literatur zu studieren und zu verstehen. Zudem können die den Lernpaketen beiliegenden Komponenten für viele Projekte in diesem Buch weiterverwendet werden.

U. Tietze, C. Schenk: Halbleiter-Schaltungstechnik, Springer-Verlag (2002)

P. Horowitz, W. Hill: The Art of Electronics, Cambridge University Press (1989)

Brian W. Kernighan, Dennis M. Ritchie: The C Programming Language, Prentice-Hall Int. (1990)

G. Spanner: AVR-Mikrocontroller in C programmieren, Franzis (2010)

Lernpaket AVR-Mikrocontroller in C programmieren, Franzis, (2012)

18 Bezugsquellen

Elektronische Bauteile und auch Mikrocontroller selbst können bei den folgenden Firmen bezogen werden:

Reichelt Elektronik: www.reichelt.de

Conrad Electronic: www.conrad.de

Speziellere Bauelemente, wie Beschleunigungs- und Tiltensoren oder auch Ultraschallmodule sind beispielsweise erhältlich bei

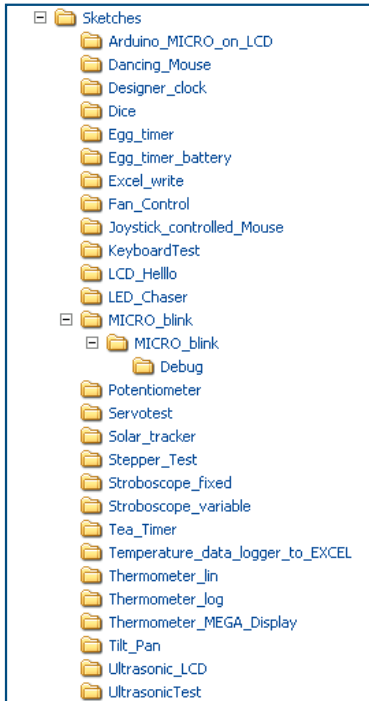
www.watterott.com

www.amazon.de

www.sparkfun.com

19 Quellcodes

Sämtliche Quellcodes zu diesem Buch sind auch auf der Website www.buch.cd zu finden. Geben Sie dort einfach die erforderlichen Ziffern der ISBN ein.



Symbole

1,5-Volt-Batterien 22
 4-Bit-Einheit 27
 4x7-Segmentanzeige 146
 5-Volt-LED 175
 8-Bit-Timer 173
 16-Bit-Timer 173
 16-MHz-Quarz 31
 8051-Controller 27
 _delay_ms 170
 _delay_us 170
 unsigned long millis 161

A

Abblockkondensatoren 174
 Abbruchbedingung 157
 Ablaufgeschwindigkeit 167
 abs 161
 Accelerometer 129
 ADC 32, 120, 168
 Addition 152
 ADR4525 170
 AD-Wandler 120
 AGND 169
 Airbags 129
 Akkus 20, 174
 Akkuzelle 22
 Aktive Konfiguration 82
 Aktiver Hub 20
 Aktuelle Sketch speichern 45
 Aktuelle Sketch zum Controller senden 44
 Alkali-Mangan-Batterie 22
 Alphanumerisches Display 93
 Analog Devices 170
 Analog-digital-Converter 168
 Analog-digital-Umwandler 32
 Analog-digital-Umwandlung 169
 Analoge Eingangswerte 168
 Analogmessgerät 115
 Analogmultiplexer 120
 analogRead 169
 analogReference 169
 Analogsignale 168
 Analogumschalter 168
 Analog-zu-digital-Umsetzer 120
 AND-Operator 153
 Anmeldung eines USB-Geräts 37
 Anode 147
 Ansteuereinheiten 96
 Ansteuermodus 106
 Anwendungsprogramm 34
 Anwendungssoftware 37
 Anzeigeeinheit 72
 Arbeitspause 171
 Arbeitsweise der Brücke 102
 Arduino-Anwendung 83
 Arduino-Boardtyp 41
 Arduino Builder 56
 Arduino Due 13
 arduino.exe 38
 Arduino-IDE 24, 164
 Arduino-Messdaten 84
 Arduino Nano 13
 Arduino NG 13
 Arduino-Verzeichnis 38
 Argument 162, 163
 Arithmetische Operatoren 152
 array 156
 Array 175
 ASCII-Zeichen 162
 AT89-Typen 27
 ATmega8-Controller 13
 ATmega16U2 34
 ATmega32U4 166
 ATmega328 34
 Atmel Studio 173
 attachInterrupt 171
 Aufnahmeleistung 96
 Auskommentieren 151
 Ausrolleffekt 64
 Auswahlvariable 159
 Automatisierung 112
 AVCC 169
 AVRdude 55

AVR-GCC-Compiler 52

B

BASCOM 52
Batteriehalter 174
Batterien 20, 174
Batteriespeisung 22
Bauteilesortiment 23
BD135 98
Bedienelemente 46, 144
Befehlsblöcke 151
Befehlsspeicher 26
Beispielprogramme 38
Beschleunigungssensoren 129
Betriebsgeräusch 108
Bibliotheken 38, 84
Bibliotheksfunktionen 96
Bildstabilisator 129
Binäres AND 164
Binäres OR 164
Binäres XOR 164
Binärschreibweise 165
Binärtechnik 153
Binärvariablen 153
Binärzahl 153
Bipolarer Motor 99
Bitbreite 173
Bitfolge 103
Bitmanipulationen 164
Bitmaske 165
Bitnummern 167
Bitweise Invertierung 153
Bitweise Operatoren 176
Blaue Power-LED 20
Blink 42
blink.ino 42
Boardtypen 56
Boolesche Logikoperationen 153
Boot-Code-Sektor 31
Bootloader 34
Breadboard 15
Breadboard-Aufbau 144

Breadboard-Schaltung 16
break 159, 160
Bulk-Transfer 35
Busschiene 17

C

Case-Anweisung 148
Case-Auswahl 160
cbi 165
CHANGE 171
Codesequenz 173
Compiler 173
Compilerlauf 54
Compound-Ausdrücke 154
Compound-Operatoren 154
Constant 154
Controllerboard 174
Controllerparameter 174
cos 161
Counter 70, 173
CPU-Taktfrequenz 173
C-Strukturen 164
CursorPosition 162

D

D- 35
D+ 36
DAC 32
Datenkommunikation 36
Datenlogger 137
Datenrichtungsregister 167
Datenverkehr 42
DDR-Register 167
Debugging 174
default 160
define-Befehl 54, 170
Deklaration 150, 175
delay 160
Delay-Befehl 70
delay.h 54
delayMicroseconds 161
Device-Treiber 82

Dezimalpunkt 149
 Dezimalstellen 148
 Dezimaltrennzeichen 85
 Digital-analog-Umsetzer 32
 digitalRead 160
 Digitalservo 113
 Digital Storage Oszilloskop 24
 Digitaluhr 71
 digitalWrite 52, 160
 Digits 148
 Dioden 174
 Displaytreibersoftware 72
 Display 93
 display_driver.h 72, 149
 Division 152
 Done Uploading 42
 Doppelslash 151
 do while 158
 Drehfeld 102, 105, 107
 Drehgeber 99
 Drehmoment 99
 Drehmomentverlauf 108
 Drehrichtung 113
 Drehwinkel 99
 Duemilanove 13

E

Echtzeit 35
 Eingabe/Ausgabe-Ports 31
 Einstecken der Anschlusspins 18
 Einzeiliger Kommentar 151
 Einzelschritt 99
 Elektromagnete 96
 Elektronikeinheit 112
 Elektroniklabor 23
 Elektronische Wasserwaage 129
 Elkos 174
 Embedded Systems 27
 Empfänger 135
 Encoder 99
 Endanschlagskontrolle 99
 Endlosschleife 54, 150, 158

Endpoint 81
 Energieverbrauch 114
 Entfernen der Arduino-Platine 18
 Entpacken 38
 Enumeration 36
 Erdbeschleunigung 130
 Eurogehäuse 144
 Examples 38
 Excel-Tabelle 137
 Experimentierboard 16
 Exponentielle NTC-Kennlinie 139
 Externe Referenzspannung 169
 Externe Spannungsquellen 20

F

FALLING 171
 Farbringe 174
 F_CPU 170
 Fehlerstatistik 46
 Feldelemente 175
 Fiktive Adresse 80
 Firmware 26
 Firmwareentwicklung 174
 Flashspeicher 26
 Float 155
 for 158
 FPGA 26
 Freilaufdiode 105
 Frequenzstabilität 31
 Frontplatte 144
 Full Speed 36
 Funktionen 163
 Funktionsnamen 163
 Fuse 174
 Fuses 54
 Future Technologies 34

G

Ganzzahlen 155
 Gerätemanager 41
 Gerätetreiber 37, 83
 Geschweifte Klammern 151

Getriebe 104
Getriebemotor 112
Getriebeuntersetzung 104
Gleich 152
Gleichheitsoperator 175
Gleichspannungsmotor 112
Gleichstrommotoren 96
GND-Leitung 113
GPIO-Register 166, 167
Größer 152
Größer gleich 152
Großrechenanlage 27
Grundfrequenz 112
Grundkonstanten 154
Gültige Adresse 80

H

Halbleitermaterialien 124
Halbschrittmodus 106
Hardware 26
Hardwareausstattung 52
hardwaregesteuert 173
Hardwareschalter 85
Hauptdrehachse 112
Hauptprogramm 54, 150
Haushalt 46
H-Brücke 101
HC-SR04 135
HD44780-Treiber 93
Heißleitendes Verhalten 124
hex-Datei 58
HID 36
HIGH 160
HIGH/LOW 155
High-Torque-Motor 99
Hohlbuchse 20
Horizontale Nachführung 110
Horizontallinie 130
Host 35
Hostrechner 176
hot-plug-fähig 35
Human Interface Device 36

Hybridmotor 99

I

ICP-Programmer 57
IDE-Version 38
if 157
if else 157
Include-Datei 54
Increment 158
Index 156, 175
induktive Spannungsspitzen 105
In-Endpoint 81
Informationsfenster 38
Initialisierung 150, 158
Initialwert 155
Inkrementalgeber 112
INPUT/OUPUT 155
Installation 38
Installationspakete 38
Integrierende Methoden 120
Integrierte Peripherie-Einheiten 31
Integrierte USB-Einheit 32
Integrierte USB-Schnittstelle 34
Interne Referenzspannung 169
Interrupt 171
Interrupt-Ablauf 172
Interrupteinheiten 32
interrupt.h 172
Interrupt-Pins 32
Interrupt-Routine 72, 172
Interrupts 171
Interrupt-Transfer 35
I/O-Port 31
Isochron-Transfer 35
ISR 171

J

Joystickmodul 127

K

Kalibrationsfunktion 125
Kalibrationswerte 154

Keyboardanwendungen 84
 Keyboard.begin 162
 Keyboard.end 162
 Keyboardfunktionen 161
 Keyboard.print 162
 Keyboard.write 162
 Kippsensor 46
 Kleiner 152
 Kleiner gleich 152
 Kommentarzeichen 151
 Kompensation 139
 Kondensatorplatten 130
 Konfigurationsdaten 81
 Konstanten 154
 Konstruktionsbaukasten 110
 Kontakt 174
 Kontaktfedern 16, 174
 Kontrollanzeige 88
 Kontroll-Transfer 35
 Kreuzknüppel 127
 Kurzschluss 20

L

L293 102
 L298 105
 L298-Treiber 105
 Lastkapazitäten 174
 Laufzeitmessung 134
 LCD-Bibliothek 93
 LCD-Display 94
 LEDs 174
 Leere Schleife 170
 Leistungstransistor 62, 96
 Leonardo 13, 34
 Lesepuffer 80
 Lilypad 13
 Links schieben 164
 LiquidCrystal 93
 LiquidCrystal.cpp 96
 LiquidCrystal.h 95
 Lochrasterplatten 144

Logarithmische Übertragungsfunktion 139
 Logikblöcke 26
 Long 155
 loop() 150
 Lötfreie Steckplatten 15
 LötKolben 25
 Lötstation 25
 LOW 160, 171
 Low-Power-Modus 82
 Low Speed 36
 Luftfeuchtigkeit 138

M

Magnet 99
 Makro 173
 Map-Funktion 129
 Maschinencode 164
 Mauscursor 162
 Maussteuerung 85
 Maustaste 163
 max 161
 Maximalausschläge 113
 Maximalstrom 102
 Maximalwert 173
 Mega 13
 Megadisplay 140
 Mehrfacherfassung 120
 Mehrzeiliger Kommentar 151
 MEMS-Einheit 130
 MEMS-Technologie 129
 Messung von Versorgungsspannungen 24
 Messwerterfassung 120
 Metallkugel 46
 Metalloxide 124
 Micro-USB-Stecker 49
 Mikrocomputerbereich 27
 Mikroprozessor 27
 Mikroschaltungen 26
 Millisekunden 161
 min 161
 Mini-DSO 24



Mittellanzapfung 100, 101
Mittelstellung 113
Mittenposition 129
Modellbau 112
Modulo 152
Monitor-LED 175
Motoransteuerung 106
Motorpositionierung 105
Motorspule 104
Motortyp 99
Motorumdrehung 104
Motorversorgungsspannung 105
Motorzuleitung 105
Mouse.begin 162
mouseButton 88
Mouse.click 162
mouseDelay 88
Mouse.end 162
Mouse.isPressed 163
MOUSE_LEFT 162
MOUSE_MIDDLE 162
Mouse.move 162
Mouse.press 163
Mouse.release 163
MOUSE_RIGHT 162
Multimediatechnik 35
Multimeter 24
Multiplikation 152

N

Netzteil 20
Neuer Sketch 44
Nicht flüchtige Speichereinheit 26
NOT-Operator 153
NTC-Widerstand 98

O

Oberfläche der IDE 38
Odometrie 104
Offset 125
Operator 152, 176

OR-Operator 153
Oszillatorschaltung 174
Oszilloskop 24
Out-Endpoint 81

P

Parallelport 82
Parallelverarbeitung 172
PC-Applikationen 83
Pegel 160
Pegelwechsel 32, 171
Periodendauer 112
Peripherie-Einheiten 32
Permanentmagnet 115
Physical Computing 12, 92, 164
Physikalische Konstanten 154
physikalische Welt 92
Pinbezeichnungen 167
pinMode 52, 160
Pinmodi 150
Pinverbindungen 95
Pinzuordnung 167
PLL-Baustein 36
Polling 80
Polycarbonat 115
Positionsbestimmendes Element 112
Positionserkennung 99
Positionspotenziometer 129
Positionsrückmeldung 99
Potenziometer 63, 122
Potenziometerspannung 122
pow 161
Praxisanwendung 46
Praxisprojekt 52
Präzision 70, 120
Präzisionsmessungen 169
Prellen 120
Processing-Funktion 163
Produktionstechnik 112
Programmentwicklung 42
Programmer 34

Programmierung 34
 Programmpause 160
 Programmstart 150
 Programm zur Eieruhr 47
 Projektordner 54
 Protokollebene 36
 Prototyping-Shields 14
 Prozessorkern 26
 Prozessortakt 170
 Prozesssteuerung 124
 Prüfbedingung 159
 Prüfen/Compilieren 44
 Pufferspeicher 80
 Pulsfolge, 112
 Pulsweite 112
 PWM-Signale 112

Q

Quantisierungsfehler 120
 Quarzbasis 72
 quasi parallel 172
 Quasi-Standard 72

R

RAM 26
 Rechts schieben 164
 Referenzspannung 169
 Regelschleife 100
 Register 164
 Relaissteuerungen 92
 Reset 42, 150
 RISING 171
 Robotertechnik 99
 ROM 26
 Rotor 99
 Routine 52
 RS-232-Interface 82
 RS-232-kompatible Schnittstelle 32
 Rx/Tx 42
 Rx/Tx-LEDs 42

S

sbi 165
 Schaltungsgruppen 19
 Schleifenvariable 158
 Schlüsselwort 154
 Schnittstellen 150
 Schnittstellenbetrieb 36
 Schnittstellenparameter 154
 Schreibposition 96
 Schreibvorgang 85
 Schrittabfolge 101
 Schrittmotor 99
 Schrittmotoren 79
 Schrittposition 106
 Schrittverlust 105, 107
 Schrittverlustfreie Ansteuerung 105
 Schrittweite 101
 Segmente 147
 Selbstrücksetzende Sicherungen 82
 Seltene-Erden-Metall 99
 Semikolon 151
 Sender 135
 Sensorexemplar 125
 Sensormesswerte 125
 Sequenzielle Aktivierung 102
 Seriellen Datenmonitor starten 45
 Servoansteuerung 140
 Servoausgang 114
 Servobibliothek 114
 Servoeinheiten 112
 Servoleistung 114
 Servomotor 112
 Servoobjekt 114
 Servoposition 112, 115
 Servos 92
 Servotechnologie 112
 Servotypen 113
 setSpeed 107
 setup-Funktion 150
 setup-Routine 150

Shields 12
Siebensegmentdisplay 72
sin 161
Sketch 34, 150
Sketch öffnen 45
Softwarearchitektur 36
Softwareroutine 176
Solarmodul 110
Solarstromerzeugung 110
Solarzellenarrays 110
Source-Code 59
Spannungsbereich 20
Spannungsstabilisierung 20
Speichereinheit 26
Speicherort 58
Speicherplatzressourcen 155
Sperrichtung 105
Spezialbauteile 24
Spezialperipherie-Einheit 32
Spezialperipherie im ATmega32U4 33
Sprachelemente 150
sqrt 161
Standardpulsdauer 112
Standardservos 113
Startbedingung 158
Statorspule 99
Steckplatine 18
Steilheit der Kennlinie 125
Stepper 99
Stepperantrieb 99
Stepper.h-Bibliothek 109
Stoppertreiber 102
Steuerimpuls 114
Steuerpulsfrequenz 114
Steuerschritte 105
Steuersignal 99
Steuersymbole 42
Steuerwert 111
Störanfälligkeit 176
Stroboskop 62

Stroboskopblitzer 62
Stroboskopfrequenz 63
Stroboskopscheiben 62
Stromlossschalten 104
Stromreserve 82
Stromversorgung 20
Subtraktion 152
Sukzessive Approximation 121
switch-Anweisung 159
switch/case 159
Syntaktische Elemente 151
Syntaxelemente 151

T

Takt 31
Taktfrequenz 36, 54, 170
Taktrate 31
Taktzyklen 70
tan 161
Tastatur 161
Tastatur-Device 161
Tastatureingabe 161
Temperatur 156
Temperaturbereich 125
Temperaturerfassung 98
Temperaturfühler 123
Temperaturkurven 137
Temperaturlogger 138
Temperaturprotokolle 137
Temperatursensor 124
Temperaturüberwachung 125
tgl 165
Tiltsensor 48, 66
Timer 173
Timer1 71
Timer1.attachInterrupt 71
Timer1.initialize 71
Timerfunktion 71
Timer-Interrupt 71
TimerOne.cpp 71

TimerOne.h 71
 Tintenstrahldrucker 99
 Tool 52
 touring-vollständig 152
 Treiber 41
 TRUE/FALSE 154

U

Übertragungsfehler 36
 Übertragungsfunktion 125
 Ultraschall 134
 Ultrasonic 135
 Umdrehungszahl 62
 Umweltmessdaten 137
 Ungleich 152
 Unipolarer Motor 99
 Unterhaltungselektronik 97
 UPLOAD 42
 Uploading to I/O Board 42
 Uploadtool 55
 US-020 135
 USBasp 57
 USB-Bus 82
 USB-Device 80
 USB-Eingabegeräte 80
 USB-Festplatte 35
 USB-Firmware 83
 USB-Hub 82
 USB-Joystick 126
 USB-Kabel 39
 USB-Leitung 35
 USB-Maus 162
 USB-Maus-Interface 84
 USB-Mouse-Device 87
 USB-Port 20
 USB-Schnittstelle 31, 176
 USB-Speichersticks 35
 USB-Stack 36
 USB-Stromversorgung 82
 USB-Tastatur 84
 USB-Verbindung 78, 176

V

Variablen 155
 Variablenänderung 173
 Variablenfelder 156
 VBUS 36
 Ventilator 97
 Verbose-Option 58
 Vergleichsausdruck 175
 Verknüpfung 176
 Version 1.0.5 38
 Vertikale Nachführung 110
 VI 20
 Vibrationen 16
 Virtuell 176
 Virtueller Com-Port 41
 volatile 173
 Vollschrittbetrieb 106
 Voltage In 175
 Vorwiderstand 47, 175

W

Wackelkontakte 16
 Wandlerbaustein 84
 Wandlungsdauer 120
 Wasserwaage 130
 Wave-Drive 106
 Weicheisenkranz 99
 Werkzeug 24
 wheel 163
 While 157
 Windows-Applikation 83
 Wirkungsgrad 110
 WordPad 96
 Würfelaugenmuster 64

X

x-Achse 127
 xVal 162

Y

y-Achse 127
 yVal 162

Z

- Zähler 70
- Zählerregister 173
- Zählerstand 173
- Zählschleifen 170
- Zahnradgetriebe 111
- Zeitabhängige Steuerung 70
- Zeitintervall 46, 170
- Zeitverzögerung 160
- Zeitzähler 72
- Zink-Kohle-Batterie 22
- Ziparchive 38
- Zuweisung 175



COOLE PROJEKTE MIT DEM Arduino™ Micro

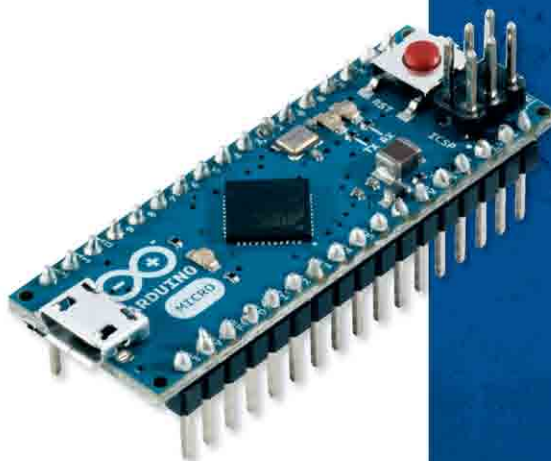
Unterschätzte Platine: Der Arduino™ Micro passt auf jedes Steckboard, steht aber in Sachen Leistung seinen großen Brüdern in nichts nach. Programmierneulinge werden die benutzerfreundliche Arduino™-Oberfläche lieben, Elektronik-Fans werden begeistert sein von den zahlreichen Projekten, die sie mit dem Micro umsetzen können.

Kompaktes Elektronikwissen

Raspberry Pi ist in aller Munde, aber für viele Projekte ist ein Arduino™ die bessere, weil stromsparende Alternative. Haben Sie schon einmal eine Designeruhr gebaut? Oder einen modernen Teeautomaten? All das ist mit dem Micro kein Problem. Auch für das Büro ist der Micro eine Bereicherung: Lassen Sie sich Daten direkt in Excel oder Word ausgeben oder nutzen Sie den Micro als Computermaus.

Praxisprojekte

Von Anfang an ist der Bezug zur Praxis da: Lassen Sie es blitzen mit dem Powerstroboskop, bauen Sie den LED-Würfel oder Ihre eigene Wetterstation. Der Autor, Dr. Günter Spanner, baut alle seine Projekte selbst und beweist das zum Beispiel auch in Webinaren zum Thema Arduino™.



Highlights:

- Kleine Platine ganz groß
- Blitzlichtgewitter: einzigartige Lichteffekte mit dem Micro
- Zeitgeist: der Micro als Uhr
- Fremdgesteuert: alles zum Thema USB-Verbindung
- Bewegung bitte: Ansteuern von Motoren
- Messen, Steuern, Regeln
- Programmierprofi

Über den Autor:

Dr. Günter Spanner ist eine echte Größe auf dem Gebiet der Elektrotechnik. Während seiner Mitarbeit in verschiedenen Forschungs- und Entwicklungsabteilungen, u. a. bei ABB und Siemens, konnte er mehr als zehn Patente auf dem Gebiet der Elektronik, der Umweltsensorik sowie der Biotechnologie und Medizintechnik anmelden. Er ist außerdem Fachdozent für Physik und Elektrotechnik und hat im Laufe seiner Karriere schon zahlreiche Bücher und Fachartikel veröffentlicht. Dank seiner unstillbaren Neugier für neue Technologien hat er auch seine Liebe zum Arduino™ entdeckt.

