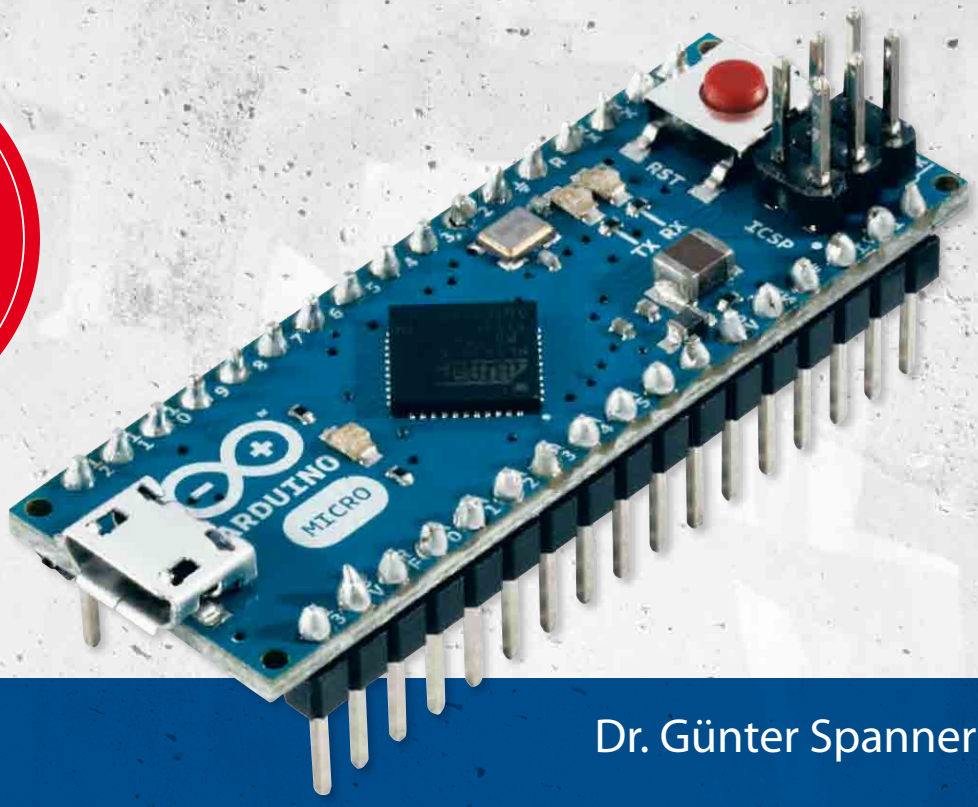
Komplett in Farbe

Faszinierende Fotos

Doppelseitige Infografiken



Dr. Günter Spanner

COOLE PROJEKTE MIT DEM Arduino™ 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

FRANZIS

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.

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.

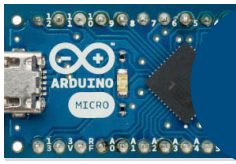
Aufgrund seiner Pinanordnung ist der Micro auch direkt in eine Lochrasterplatine oder sogar in eine IC-Fassung einsteckbar. Ein Schwerpunkt des Buchs liegt daher in der Beschreibung, wie fertig ausgetestete Laboraufbauten schnell und effizient in professionelle und dauerhaft ausgelegte Prototypen oder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

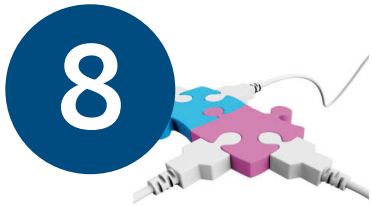
Trotz seiner kompakten Bauform bringt der Baustein alle erforderlichen Komponenten mit. Über eine integrierte Micro-USB-Buchse kann der Controller mit den gewohnten, vielfach bewährten Arduino-Programmiertools mit Software versorgt werden. Darüber hinaus wird im Buch aber auch die Programmierung in C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





Tipp

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.

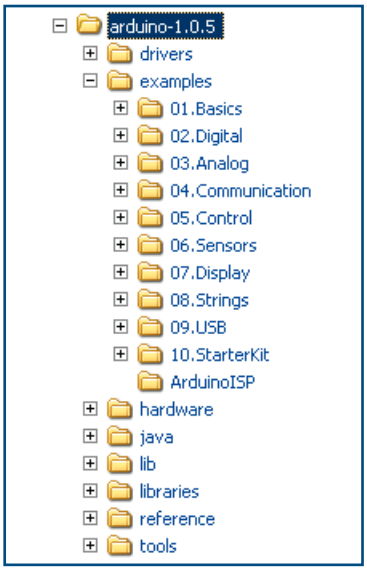


Abb. 4.1: Typisches Arduino-Verzeichnis

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.

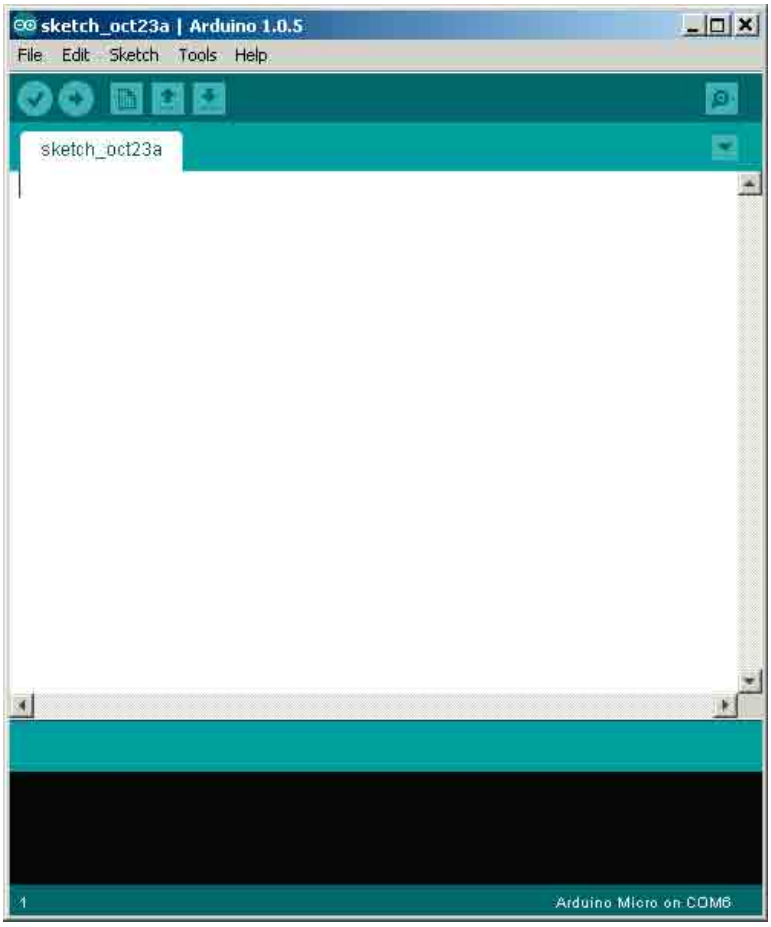


Abb. 4.3: Nach dem Start erscheint ein leeres IDE-Fenster

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.

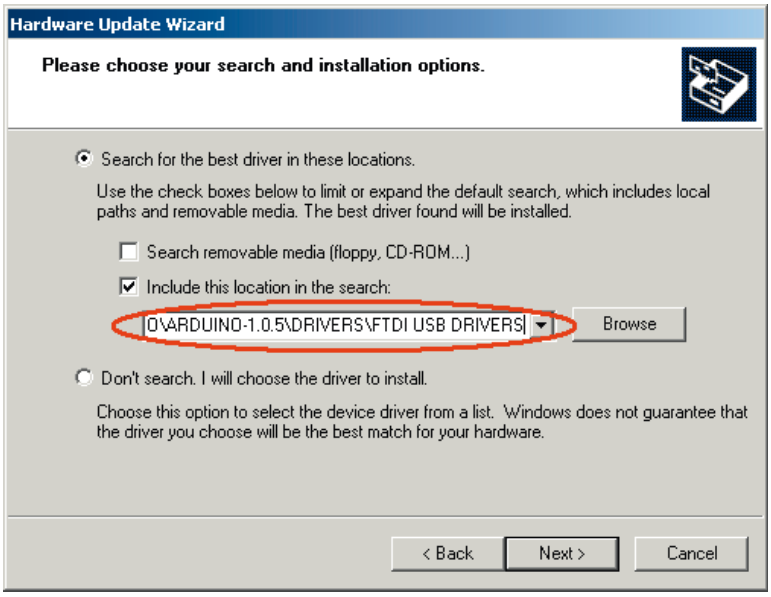


Abb. 4.5: Auswahl des Treibers

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.

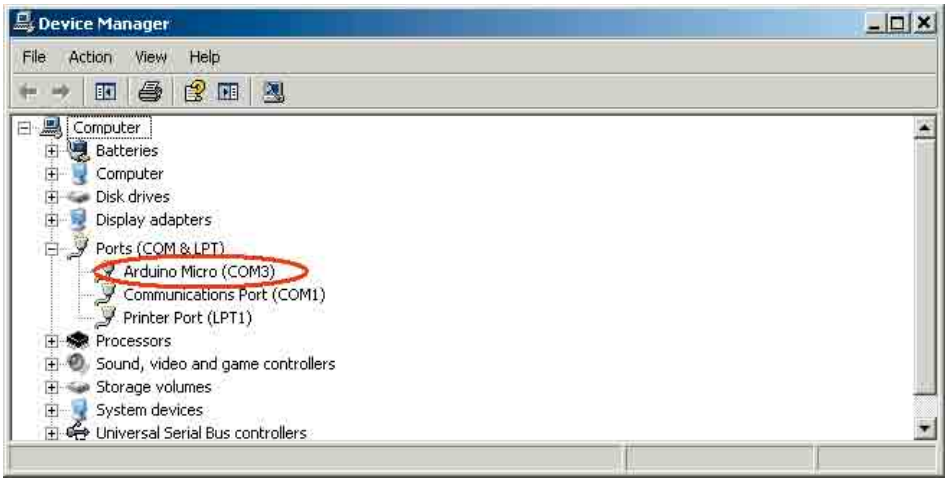


Abb. 4.6: Arduino Micro im Geräte-Manager

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

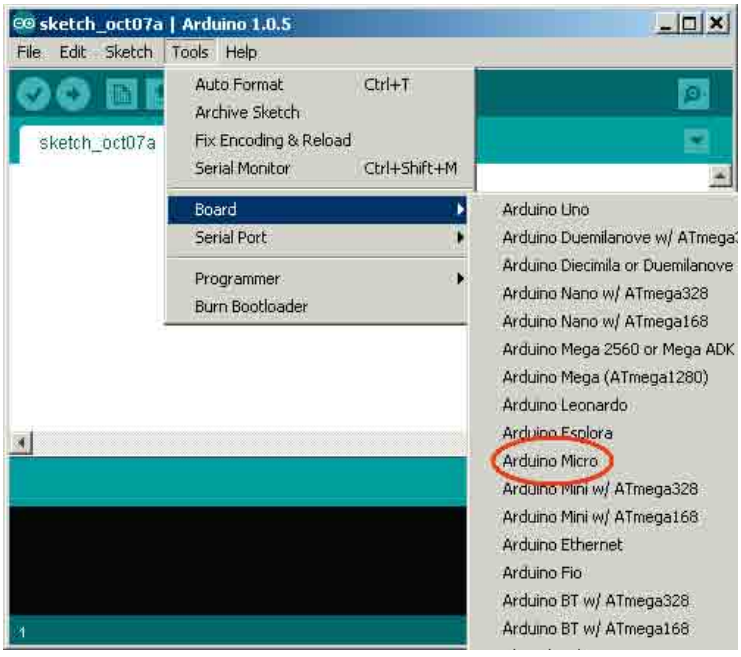


Abb. 4.7: Auswahl des Boardtyps

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.

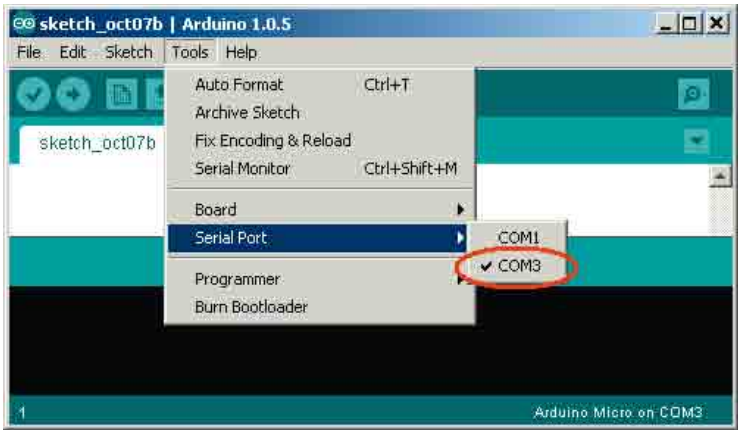


Abb. 4.8: Auswahl des COM-Ports

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.

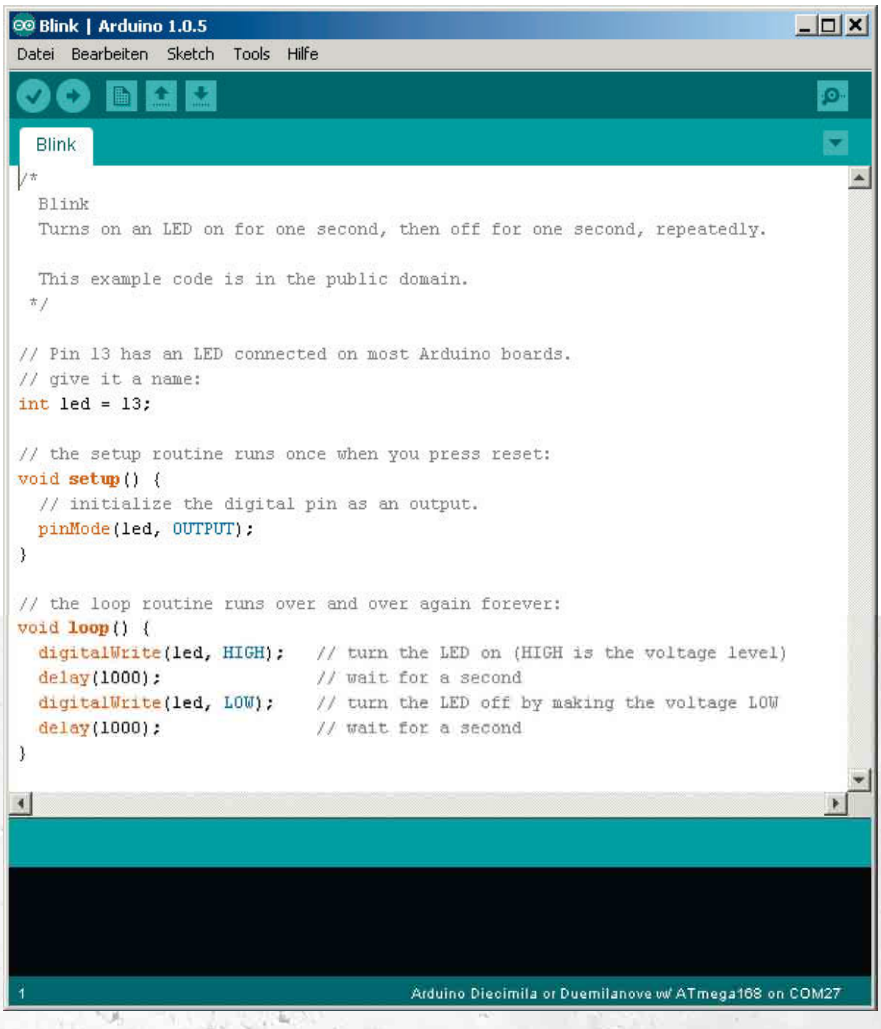


Abb. 4.9:
blink.ino ist geladen

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.

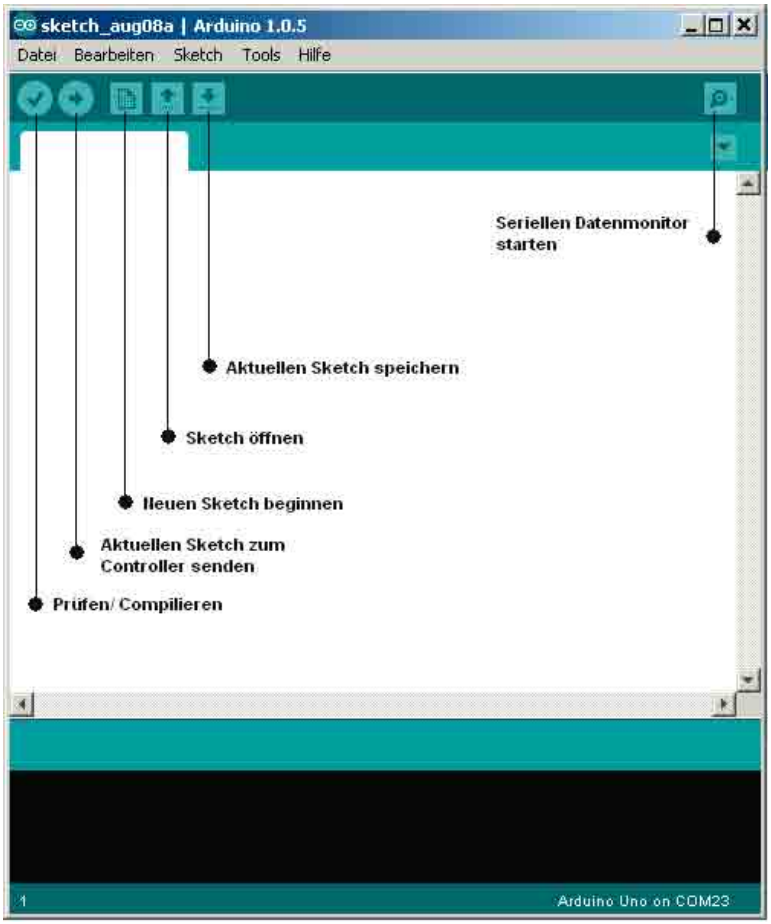


Abb. 4.11: Die Funktionen der Icons

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.

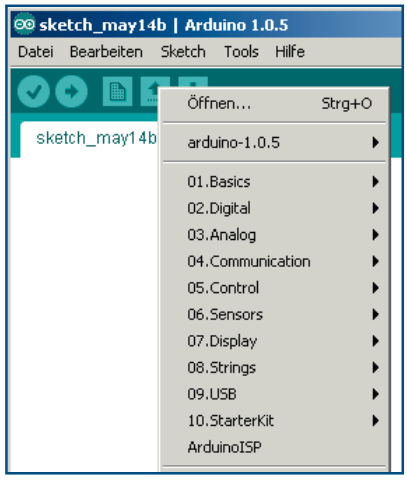


Abb. 4.12: Die Standardbeispielprogramme

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:

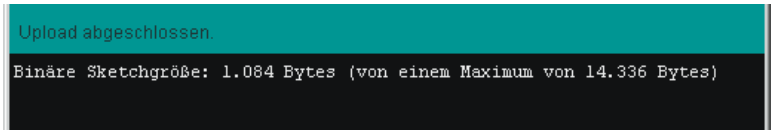


Abb. 4.13: Upload ok

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

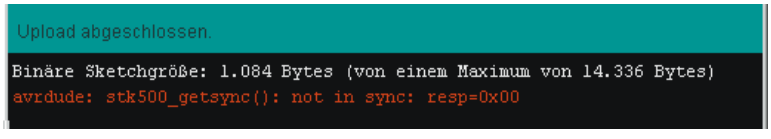


Abb. 4.14: Fehlermeldung bei falschem Port

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

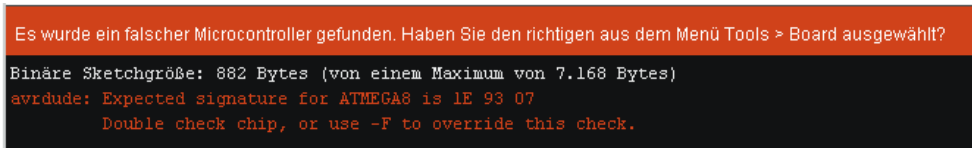


Abb. 4.15: Fehlermeldung bei falsch gewählter Boardversion

Tipp

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 Frühstücks-Ei 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 Wiese werden keinerlei Bedienelemente wie Taster oder Schalter benötigt.

Die Versorgung der Uhr erfolgt über den USB-Stecker.

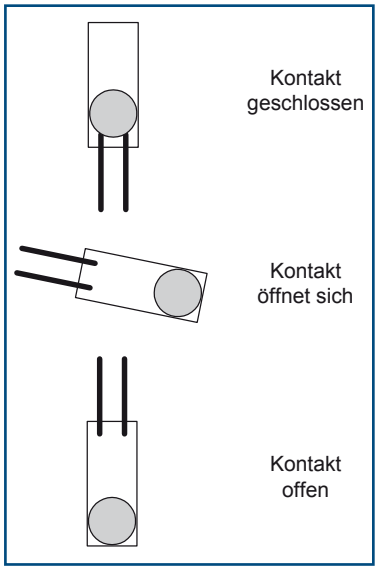


Abb. 4.16: Prinzipieller Aufbau eines Tiltsensors

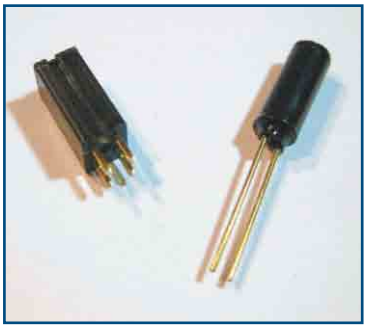


Abb. 4.17: Tiltsensoren in verschiedenen Ausführungen

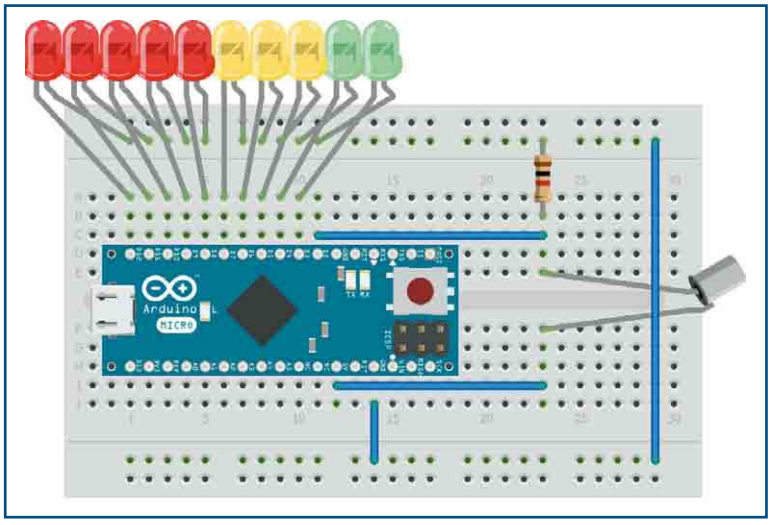


Abb. 4.18: Schaltung zur automatischen Eieruhr

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 x 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 VI

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!

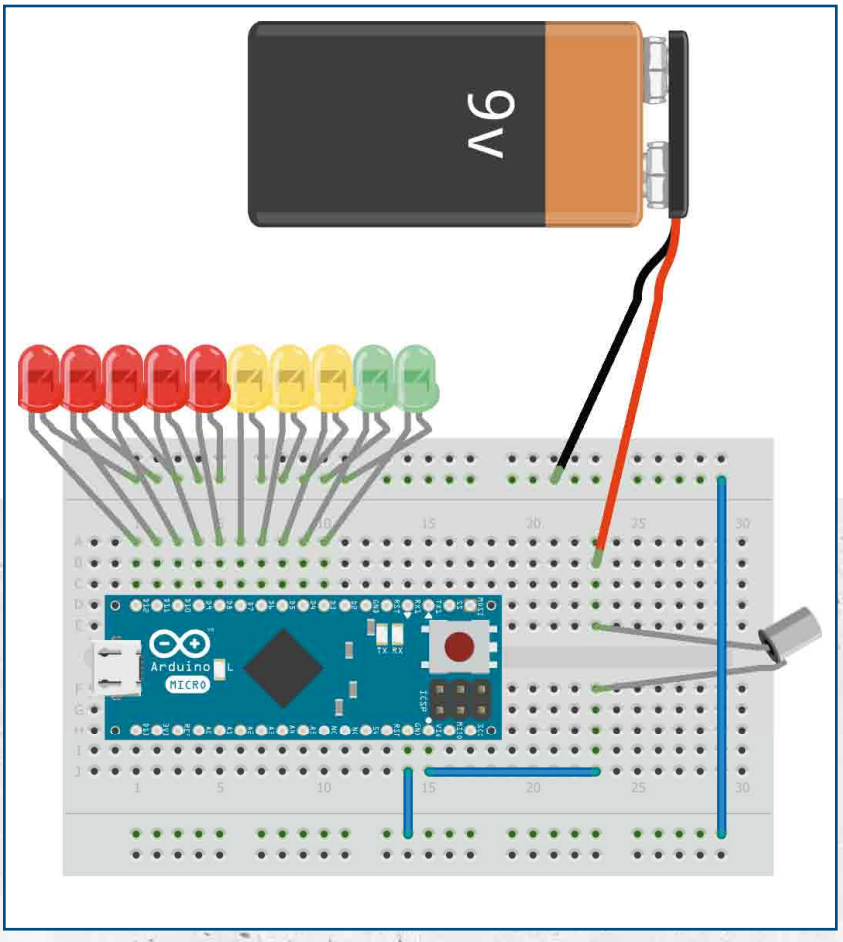


Abb. 4.19: Schaltung zur automatischen Eieruhr mit Batteriebetrieb

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

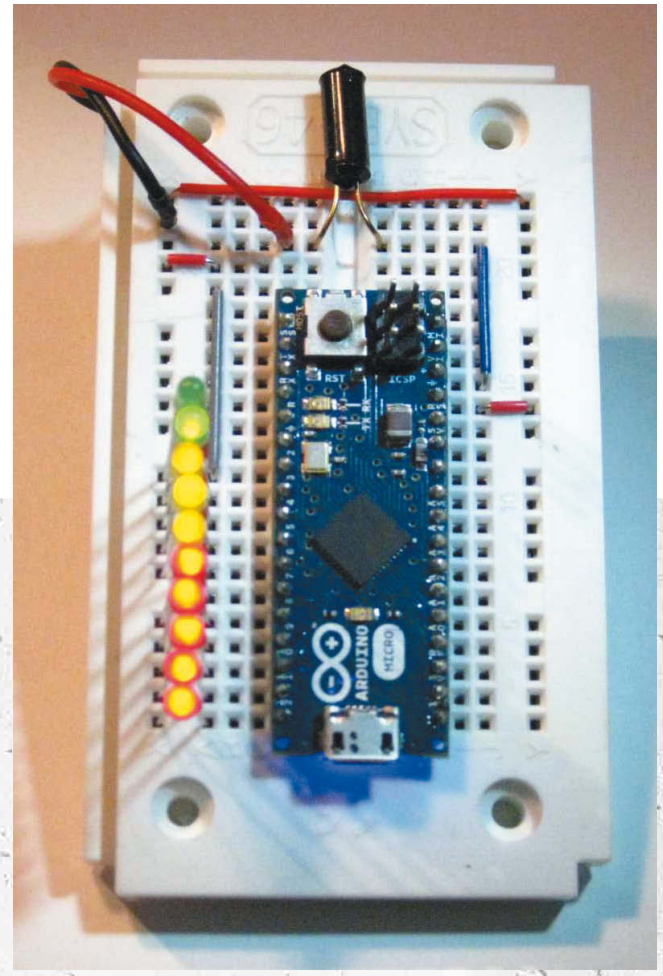


Abb. 4.20: Automatische Eieruhr

KAPITEL A

ANHANG

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 Tiltsensoren oder auch Ultraschallmodule sind beispielsweise erhältlich bei

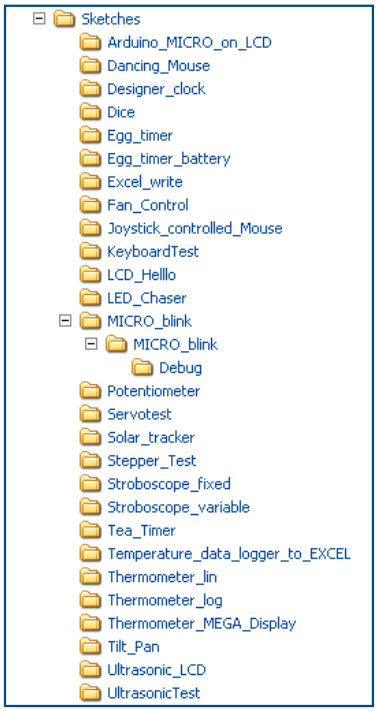
www.watterott.com

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

A

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Dr. Günter Spanner

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

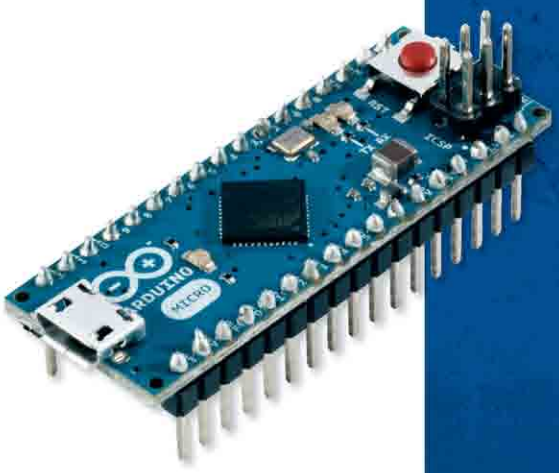
Von Anfang an ist der Bezug zur Praxis da: Lassen Sie es blitzen mit dem Powerstroboskop, bauen Sie den LED-Würfel oder Ihre eigene Wetterstation. Der Autor, Dr. Günter Spanner, baut alle seine Projekte selbst und beweist das zum Beispiel auch in Webinaren zum Thema Arduino™.

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.



9 783645 652292

Besuchen Sie unsere Website:
www.franzis.de

FRANZIS