Arduino verstehen und anwenden

Teil 14: 7-Segment-Displays am Arduino

Neben dem Arduino selbst sind meist nur noch wenige Bauelemente erforderlich, um ein vollständiges Gerät aufzubauen. Wird der Controller beispielsweise mit einer 7-Segment-LED-Anzeige kombiniert, so entsteht eine sehr universell nutzbare Plattform, auf der sich viele interessante Projekte realisieren lassen. Einige Beispiele dazu wie

- eine einzelne 7-Segment-Anzeige
- ein 4-stelliges Display
- eine Digitaluhr

werden im folgenden Artikel ausführlicher beschrieben.

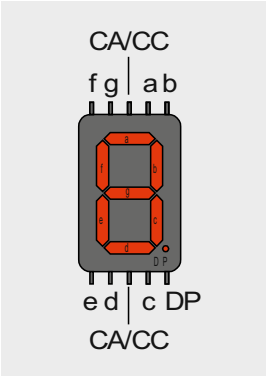

Bild 1: Typische Pin-Belegung einer einzelnen 7-Segment-Anzeige

Einfach und preisgünstig: 7-Segment-Displays

Ziffern können mit sogenannten 7-Segment-Anzeigen einfach und effizient dargestellt werden. Derartige Anzeigen bestehen häufig aus acht LED-Elementen (sieben Segmente plus Dezimalpunkt).

Die LED-Segmente sind in der Form einer Acht angeordnet. Durch entsprechende Aktivierung bestimmter Elemente können so die Ziffern von 0 bis 9 in gut lesbarer Form dargestellt werden.

Darüber hinaus lassen sich auch noch einige Großbuchstaben anzeigen (z. B. A, C, E, F etc.). Möchte man weitere Buchstaben anzeigen, müsste man Sechzehn-Segment-Anzeigen einsetzen. Diese sind aber vergleichsweise wenig verbreitet und daher teuer.

Eine andere Möglichkeit, um Buchstaben und einfache Grafiken anzuzeigen, bieten die sogenannten Punktmatrix-Displays. Diese werden in einem späteren Artikel zu dieser Serie beschrieben.

7-Segment-Anzeigen werden in den verschiedensten Formen, Farben und Anschlussbelegungen hergestellt. Ein weit verbreiteter Typ ist in Bild 1 dargestellt. Die einzelnen Segmente werden üblicherweise mit den Buchstaben a bis g bezeichnet. Die Anschlüsse sind entweder die gemeinsame Anode (Common Anode: CA) oder die gemeinsame Katode (Common Cathode, CC) der Segmente, je nach Displaytyp.

Bild 2 und 3 zeigen, wie die 7-Segment-Anzeige an einen Arduino UNO angeschlossen werden kann. Unter Anwendung des sogenannten Multiplex-Verfahrens kann man hier mit nur einem einzigen gemeinsamen Vorwiderstand für alle LED-Segmente auskommen.

Tabelle 1 verdeutlicht nochmals den Anschluss der Display-Pins an die Arduino-Ausgänge D0 bis D7.

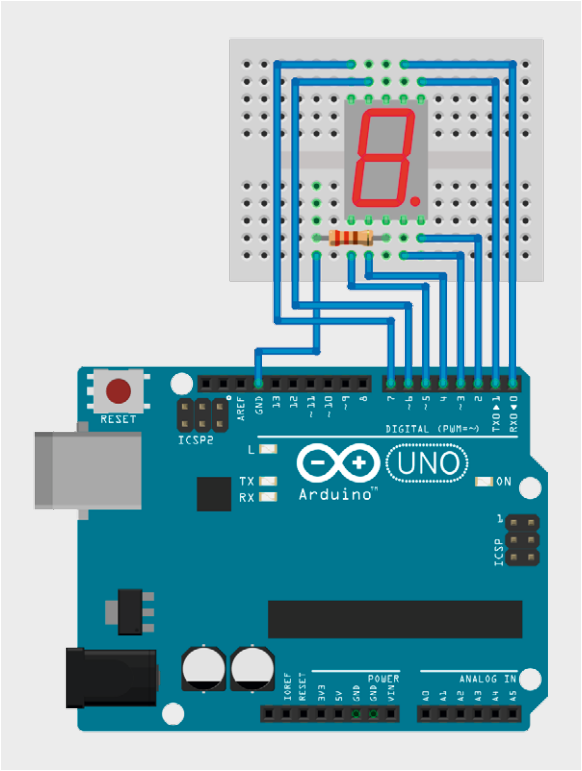

Bild 2: Ansteuerung einer einzelnen 7-Segment-Anzeige

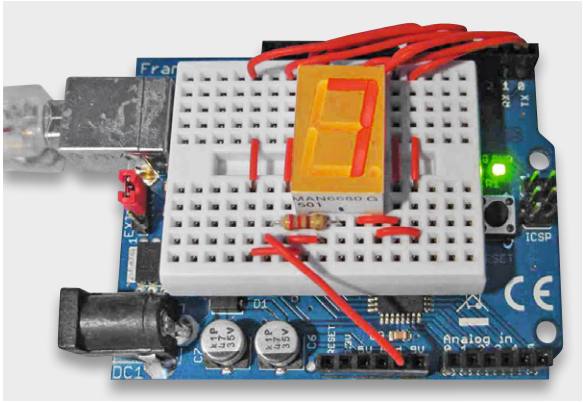

Bild 3: Die 7-Segment-Anzeige zeigt die Ziffer 7.

Tabelle 1

Anschluss der Display-Pins an die Arduino-Ausgänge D0 bis D7

Arduino-Pin	D0	D1	D2	D3	D4	D5	D6	D7
Display-Pin	a	b	DP	c	d	e	f	g

Der Sketch dazu sieht so aus:

```
// 7 segment LED display

// seven segment codes
int numbers[10][8] =
//a b D c d e f g
{{1,1,0,1,1,1,1,0}, // 0
 {0,1,0,1,0,0,0,0}, // 1
 {1,1,0,0,1,1,0,1}, // 2
 {1,1,0,1,1,0,0,1}, // 3
 {0,1,0,1,0,0,1,1}, // 4
 {1,0,0,1,1,0,1,1}, // 5
 {1,0,0,1,1,1,1,1}, // 6
 {1,1,0,1,0,0,0,0}, // 7
 {1,1,0,1,1,1,1,1}, // 8
 {1,1,0,1,1,0,1,1}, // 9
};

void showDigit(int Digit)
{ // clear all segments
  for (int i=0; i<=7; i++)          digitalWrite(i, LOW);
  for(int i=0; i<=7; i++)
  // select one segment after another
  {  if(numbers[Digit][i]==1)
        { digitalWrite(i, HIGH);
          // segment on
          delay(3);
          digitalWrite(i, LOW);
          // segment off
        }
      else delay(3);
      // for time equalization
  }
}

void setup()
{ // pins 0 to 7 as outputs
  for (int i=0; i<=7; i++)
  pinMode(i, OUTPUT);
}

void loop()
{ for (int n=0; n<=9; n++)
  for (int i=0; i<=30; i++)          showDigit(n);
}
```

Hier wird zunächst die Pin-Belegung in einem Array „numbers" festgelegt.

Tipp:
Angaben zur Pin-Belegung einer 7-Segment-Anzeige finden sich im Datenblatt. Unterscheidet sich die Pin-Belegung eines bestimmten Displays von der hier verwendeten Displayversion, dann kann entweder die Pin-Tabelle in der Software (Array „numbers") oder aber die Schaltung angepasst werden.

Im Setup werden alle erforderlichen Pins als Ausgänge aktiviert.

Die „for"-Schleife in der Loop zählt die Ziffern 0 bis 9 ab. Die Darstellung der Ziffern erfolgt in der Routine „showDigit". Dort werden die Segmente einzeln aktiviert. Je nachdem, ob im „numbers"-Array eine Null oder eine Eins steht, wird das aktuelle Segment eingeschaltet oder nicht.

Obwohl man bei einer Einzelziffer noch mit einer statischen Ansteuerung arbeiten könnte, wurde hier bereits ein Multiplexverfahren implementiert.

Das bedeutet, dass zu einem gegebenen Zeitpunkt immer nur ein Segment aktiv ist. Erst durch die rasche Abfolge der Segmentumschaltung entsteht der Eindruck einer stehenden Ziffer.

Das 4x7-Segment-LED-Display

Zu den Hauptvorteilen von 7-Segment-LED-Anzeigen zählt, dass sie auch aus größerer Entfernung noch gut ablesbar sind. Durch die Lichtemission der LEDs sind die Anzeigen selbst bei Dämmerung und Dunkelheit gut erkennbar. Ein vierstelliges Display mit zusätzlichen Dezimalpunkten ist dabei sehr universell einsetzbar.

Zum einen lassen sich damit gut Stunden und Minuten für eine Uhr oder Minuten und Sekunden für einen Timer darstellen. Andererseits ist auch eine Anzeige mit zwei Vor- und zwei Nachkommastellen realisierbar. Damit lassen sich Spannungen oder Stromstärken gut anzeigen. Für eine Temperaturanzeige in Grad Celsius kann man sogar einen einfachen Trick anwenden, um ein Grad-Zeichen darzustellen.

Ein übliches 4x7-Segment-Display enthält 32 LED-Elemente. 4 x 7 = 28 LEDs sind für die Zifferndarstellung erforderlich. Dazu kommen dann noch beispielsweise 4 Dezimalpunkte.

Meist sind jedoch entweder alle Anoden oder alle Katoden jeweils einer Ziffer zusammengeführt. Darüber hinaus werden auch alle Anoden bzw. Katoden der jeweiligen Segmente verbunden. Ein solches Display weist dann nur 12 Pins auf:

- 7 Pins für die Katoden der Displaysegmente a, b, c, d, e, f und g
- 1 Pin für die Dezimalpunkte (DP)
- 4 Anoden für die Einer- (E), Zehner- (Z), Hunderter- (H) und Tausender-Stelle (T)

Je nachdem, ob die Anoden oder Katoden zusammengefasst sind, spricht man auch hier von Common-Anode- (CA) oder Common-Cathode-Displays (CC).

Neben der Einsparung von Pins hat die gemeinsame Herausführung von Anschlüssen noch den Vorteil, dass die betreffenden Anzeigen sehr leicht im sogenannten Zeitmultiplex betrieben werden können.

Bild 4 zeigt wieder das Pinning einer gängigen Displayversion, diesmal jedoch mit 4 Digits. Das Bauelement selbst ist in Bild 5 zu sehen.

Bild 6 zeigt, wie mit diesem Display-Typ Uhrzeiten bzw. Messwerte oder Temperaturen, inklusive des Sonderzeichens „°", dargestellt werden können.

4-stellige 7-Segment-Displays als Basis für Messgeräte und Uhren

Mit einer einzelnen 7-Segment-Anzeige kann man bereits verschiedene Betriebszustände, etwa einer Wasch- oder Spülmaschine, anzeigen oder längere

Zahlenreihen sequenziell ausgeben. Für die meisten Anwendungen sind aber mehrere Ziffern erforderlich.

Mit 4-stelligen Displays sind viele praxistaugliche Geräte wie Volt- und Amperemeter, Ohm-Meter oder auch Temperaturanzeigen realisierbar.

Displaytest

In einem ersten Programm soll nun die Anzeige zunächst getestet werden. Mit Hilfe der Include-Datei display_driver.h fällt das Programm sehr kurz aus:

```
// 4x7 display test
#include "display_driver.h"

void setup()
{ DDRB = 0xFF; DDRD = 0x0F;   // set ports B & D as output
}

void loop()
{ send_int_to_LCD(1234, 2);
}
```

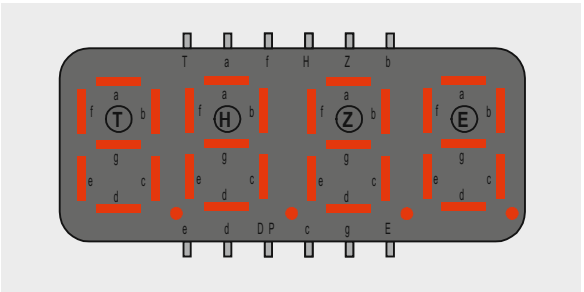

Bild 4: Anschlussbelegung eines typischen 4x7-Segment-LED-Displays

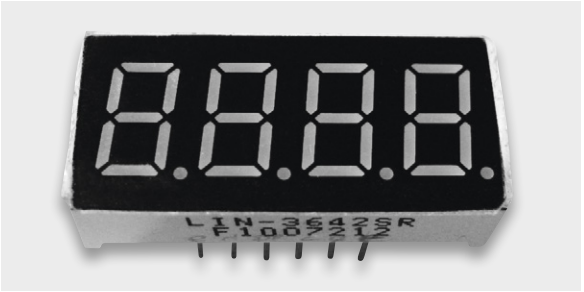

Bild 5: 4x7-Segment-LED-Display

Es müssen lediglich die mit dem Display verbundenen Ports als Ausgänge konfiguriert werden. Dies erfolgt durch die direkten C-Anweisungen

```
DDRB = 0xFF; DDRD = 0x0F
```

da hier auch Pins genutzt werden, die nicht direkt über den klassischen Arduino-Befehl „pinMode()" erreichbar sind.

Dann kann über

```
send_int_to_LCD(1234, 2);
```

eine beliebige Zahl (hier beispielsweise „1234") an das Display gesendet werden. Das 2. Argument dieser Funktion legt fest, an welcher Stelle der Dezimalpunkt gesetzt werden soll. Dabei beginnt die Zählung mit null für die ganz linke Position.

Die Hauptarbeit wird natürlich in der Include-Datei display_driver.h erledigt. Diese Datei wird im übernächsten Abschnitt näher erläutert.

Zunächst soll hier aber noch eine praktische Anwendung des LED-Displays in Form einer Digitaluhr vorgestellt werden.

Bild 6: Die 4x7-Segment-LED-Anzeige als Uhrendisplay oder zur Anzeige von Temperaturen

Digitaluhr mit LED-Display

Eine der klassischen Anwendungen von 4-stelligen 7-Segment-Anzeigen sind Digitaluhren. Der Eigenbau einer solchen Uhr hat den großen Reiz, dass man bei der Gestaltung völlig freie Hand hat. Bild 7 zeigt eine Variante, die unter Verwendung eines Arduino MICRO entstanden ist.

Auf den Boards ist die gesamte Elektronik, einschließlich des Arduino MICRO, untergebracht. Hier zeigt sich wieder der große Vorteil der Bauform des Arduino MICRO. Ein derartig kompakter Aufbau wäre mit einem Arduino UNO nicht möglich.

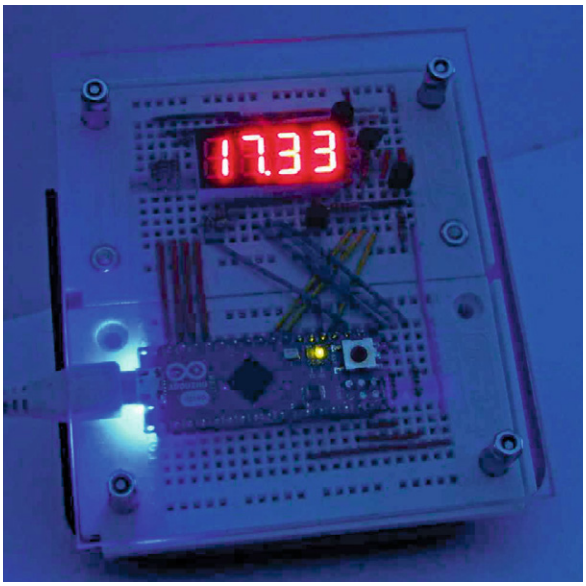

Bild 7: Digitaluhr mit Arduino MICRO

Das Schaltbild zur Digitaluhr ist in Bild 8 zu sehen. Man erkennt, dass neben dem Arduino und dem Display selbst nur noch wenige Bauteile notwendig sind:

4x PNP-Universaltransistoren, z. B. BC557
8x Widerstand 1 kΩ
4x Widerstand 10 kΩ

Diese insgesamt 4 Transistoren und 8 Widerstände reichen aus, um eine hochwertige Uhr aufzubauen.

Wenn man die Beiträge dieser Artikel-Serie bislang aufmerksam durchgearbeitet hat, bietet das Programm zur Uhr (siehe unten) keine Verständnisprobleme. Die drei Variablen sc, mn und hr sind die jeweiligen Zähler für Sekunden, Minuten und Stunden. Ein Zeitzähler „timeCounter" wird zu jeder vollen Sekunde um den Wert eins erhöht. Innerhalb der Sekunde wird die aktuelle Uhrzeit laufend an das Display gesendet:

```
while (sc==tc) send_int_to_LCD(hr*100+mn, dotPos);
```

Erst wenn tc um eins weiterzählt, wird der Dezimalpunkt umgeschaltet:

```
dotPos = 5 - dotPos;
```

Dabei bedeutet dotPos=1, dass der Punkt an der mittleren Position leuchtet, während dotPos = 4 den Dezimalpunkt löscht, da diese Position physikalisch nicht vorhanden ist. Dann wird nur geprüft, ob jeweils 60 Minuten vergangen sind und der Stundenzähler erhöht werden muss bzw. ob nach 24 Stunden der Stundenzähler auf null zurückzusetzen ist.

```
// digital clock

#include "display_driver.h"     // include display driver
#include "TimerOne.h"           // include timer lib
#define intTime 1e6             // interrupt time = 1000000 us = 1.000 s

byte timeCounter = 0;           // timeCounter
byte sc=00, mn=30, hr=17;       // seconds, minutes, hours
byte dotPosition = 1;           // position of decimal dot

void setup()
{ DDRB = 0xFF; DDRD = 0x0F;  // set Ports B & D as output
  Timer1.initialize(intTime);
  Timer1.attachInterrupt(tick);
}



void loop()
{ sc=timeCounter;
  if (sc==60)
    {     timeCounter=0; sc=0; mn++;
      if (mn==60)
          { mn=0; hr++;
            if (hr==24) hr=0;
          }
    }
   while (sc==timeCounter) send_int_to_LCD(hr*100+mn, dotPosition);
   dotPosition = 5 - dotPosition;  // blink decimal dot @ position 1
}

void tick() {timeCounter++;}
```

Der Display-Treiber

Der Vollständigkeit halber wird hier noch der zugehörige Display-Treiber erläutert. Dieses Include-File ist auch wieder ein schönes Beispiel dafür, dass man bei der Programmierung des Arduino nicht ausschließlich mit den Befehlen der Arduino-IDE arbeiten muss. Man kann auch durchaus klassische C-Befehle einsetzen.

Im Programm werden im Array „numbers" zunächst die einzelnen Ziffern gemäß der nachfolgenden Tabelle 2 codiert:

Tabelle 2

Ziffer	Binär	a	f	b	g	c	DP	d	e
0	0b00010100	0	0	0	1	0	1	0	0
1	0b11010111	1	1	0	1	0	1	1	1
2	0b01001100	0	1	0	0	1	1	0	0
3	0b01000101	0	1	0	0	0	1	0	1
4	0b10000111	1	0	0	0	0	1	1	1
5	0b00100101	0	0	1	0	0	1	0	1
6	0b00100100	0	0	1	0	0	1	0	0
7	0b01010111	0	1	0	1	0	1	1	1
8	0b00000100	0	0	0	0	0	1	0	0
9	0b00000101	0	0	0	0	0	1	0	1

Soll z. B. die Ziffer „0" dargestellt werden, so müssen alle Segmente bis auf das g-Segment leuchten. Da eine Anzeige mit gemeinsamer Anode gewählt wurde, bedeutet dies, dass alle Ausgänge, bis auf den am g-Segment angeschlossenen Pin, auf „low" liegen müssen. Daraus ergibt sich ein Binärmuster für die „0" von:

0b00010100

entsprechend der ersten Zeile der Tabelle. Analog können so auch alle anderen Ziffern codiert werden. Auch andere Zeichen sind auf diese Weise darstellbar. So kann das °-Zeichen binär als

0b00001111

geschrieben und in einer Temperaturanzeige entsprechend eingesetzt werden.

Die Routine

```
void send_int_to_LCD(int n, int dot)
```

verteilt die einzelnen Ziffern einer 4-stelligen Zahl an die Digits des Displays. Danach werden zyklisch die Digits 1 bis 4 für die Einer-, Zehner-, Hunderter- und Tausender-Stelle aktiviert. Dabei wird über die Routine

```
void digit (unsigned int value, unsigned int pin)
```

jeweils eine Ziffer dargestellt.

Der vollständige Treiber sieht damit so aus:

```
// display driver
#include <avr/io.h>

const int numbers [10] =
{ 0b00010100, 0b11010111, 0b01001100, 0b01000101, 0b10000111,
  0b00100101, 0b00100100, 0b01010111, 0b00000100, 0b00000101};

char DX000, D0X00, D00X0, D000X;      // store single digits
char act_dig = 0;

// Write number at indicated position to display
void digit (unsigned int value, unsigned int pin)
{  PORTB = numbers[value]; // send number to output
   PORTD &= ~(1 << pin);    // PinLow => current digit on
}

// set decimal dot, pos = 0 leftmost ... pos = 3 rightmost
// pos > 3 no dot
void set_dot (int pos)
{ if (pos < 4)
  {  PORTB = 0b11111011;
     PORTD &= ~(1 << pos);
  }
}

void send_int_to_LCD(int n, int dot)
{  // distribute number to digits
   DX000 = n/1000; n %= 1000;
   D0X00 = n/100;  n %= 100;
   D00X0 = n/10;   n %= 10;
   D000X = n;
   // activate digits one after another
   PORTD = 0b00001111;      // Digits off
   switch(act_dig)
   { case 0: digit (D000X, 3); break;
     case 1: digit (D00X0, 2); break;
     case 2: digit (D0X00, 1); break;
     case 3: digit (DX000, 0); break;
     case 4: set_dot(dot);
   }
   act_dig++; if (act_dig == 5) act_dig = 0;
}
```

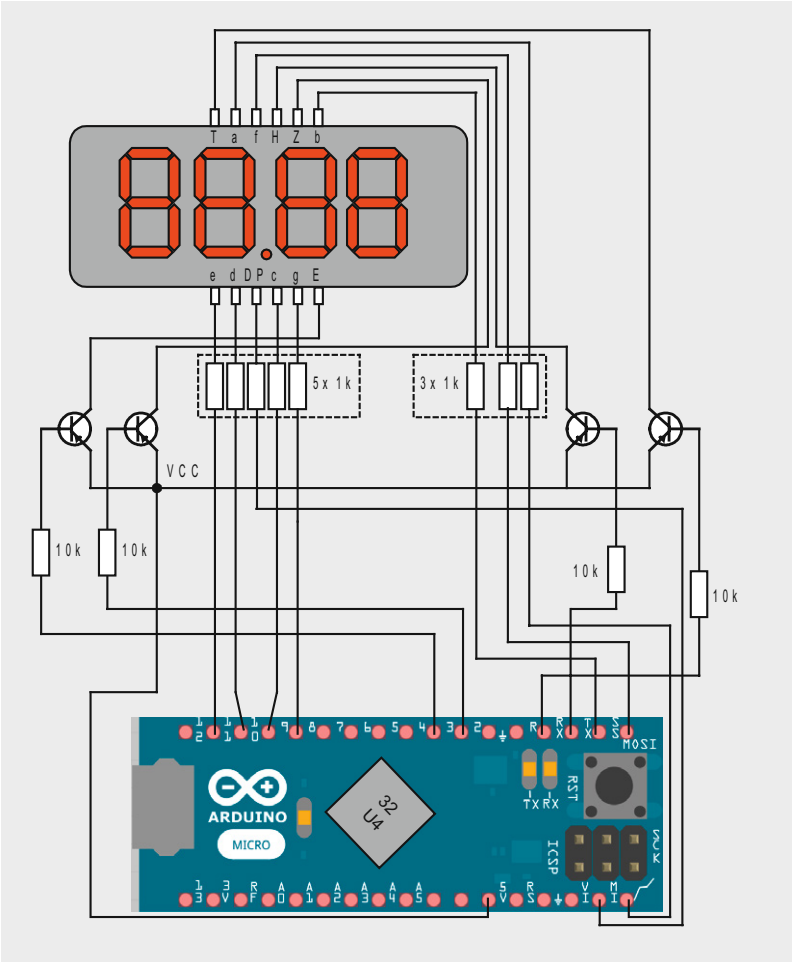

Bild 8: Schaltbild zur Digitaluhr

Ausblick

Nachdem in diesem Beitrag die Anwendung und Ansteuerung von ein- und mehrstelligen 7-Segment-Anzeigen ausführlich diskutiert wurde, geht es im nächsten Beitrag um die Ansteuerung größerer LED-Displays mittels einer flexiblen Library.

Damit lassen sich auch Anzeigeeinheiten mit bis zu 12 Digits ansteuern. Neben Ziffern und Zahlen können dann auch kurze Textnachrichten ausgegeben werden.

Darüber hinaus stellt die Library auch Funktionen für die Ansteuerung von Dezimaltrennungszeichen und Doppelpunkten zur Verfügung. Damit lassen sich praktisch alle gängigen Displayeinheiten effizient und komfortabel ansteuern.

Weitere Infos:

[1] G. Spanner: Arduino – Schaltungsprojekte für Profis, Elektor-Verlag 2012, Best.-Nr. CF-10 94 45, € 39,80

[2] Mikrocontroller-Onlinekurs, Franzis-Verlag, exklusiv für ELV, 2011, Best.-Nr. CF-10 20 44, € 99,–

[3] Grundlagen zur elektronischen Schaltungstechnik finden sich in der E-Book-Reihe „Elektronik!“ (www.amazon.de/dp/B000XNCB02)

[4] Lernpaket „AVR-Mikrocontroller in C programmieren“, Franzis-Verlag 2012, Best.-Nr. CF-09 73 52, € 39,95

Preisstellung Dezember 2015 – aktuelle Preise im Web-Shop

Empfohlene Produkte/ Bauteile:	**Best.-Nr.**	**Preis**
Arduino UNO	CF-10 29 70	€ 27,95
Arduino MICRO	CF-10 97 74	€ 24,95
Mikrocontroller-Onlinekurs	CF-10 20 44	€ 99,–

Alle Arduino-Produkte wie Mikrocontroller-Platinen, Shields, Fachbücher und Zubehör finden Sie unter: www.arduino.elv.de