Arduino verstehen und anwenden

Teil 15: Multidigit-LED-Displays

Im letzten Beitrag wurden Anwendung und Funktionsweise von 7-Segment-Anzeigen ausführlich beschrieben. Durch den Einsatz der sogenannten Multiplextechnik konnten Anzeigen mit bis zu 4 Digits aufgebaut werden. Hierzu wurde ein Displaytreiber entwickelt, welcher für die Ansteuerung der einzelnen LED-Segmente verantwortlich war.

Wie aber aus früheren Artikeln bereits bekannt ist, stehen für Arduino-Anwendungen häufig umfangreiche Programmbibliotheken zur Verfügung. Dies trifft auch auf die Ansteuerung von LED-Displays zu. Für die Displaysteuerung existieren sogar mehrere Varianten. Eine der umfangreichsten und ausgereiftesten ist die SevenSeg-Library. Wie üblich kann diese unter http://sim.marebakken.com/SevenSeg.zip kostenlos aus dem Internet geladen werden. Mittels dieser Bibliothek können auch komplexe Displays mit bis zu 12 Digits angesteuert werden. Genau genommen ist die Anzahl der Digits theoretisch sogar unbegrenzt, in der Praxis zeigt sich jedoch, dass Anzeigen mit mehr als 12 Stellen zu Problemen führen. Im Rahmen dieses Artikels soll diese Bibliothek etwas ausführlicher diskutiert werden. Natürlich werden auch wieder Praxisbeispiele und Anwendungen vorgestellt, u. a.

- Ansteuerung von Anzeigeeinheiten mit bis zu 12 Digits
- Numerische und alphanumerische Zeichendarstellung
- Digitale Counter und Frequenzzähler mit hoher Auflösung

Multiplexbetrieb größerer Displayeinheiten

Im letzten Artikel wurde das Prinzip des Zeitmultiplex erläutert. Dabei kam auch ein Displaytreiber zum Einsatz, der an verschiedene Anzeigeeinheiten angepasst werden konnte.

Hier soll nun die Anwendung der Bibliotheksfunktionen im Vordergrund stehen. Zunächst kann man mehrstellige 7-Segment-Anzeigen in zwei Gruppen einteilen:

- CA: Bauelemente mit gemeinsamer Anode (für **C**ommon **A**node) – siehe Bild 1 (zugehöriges Schaltbild)
- CC: Bauelemente mit gemeinsamer Katode (für **C**ommon **C**athode) – siehe Bild 2 (zugehöriges Schaltbild)

Den internen Aufbau eines speziellen Displays kann man dem zugehörigen Datenblatt entnehmen. Falls zu einem bestimmten Display keine Informationen erhältlich sind, kann man das Pinning aber auch sehr leicht durch Ausmessen bestimmen. Hierzu benutzt man einfach eine Spannungsquelle mit geeignetem Vorwiderstand (z. B. 1 kΩ) und testet die verschiedenen Pins. Man muss dann nur noch notieren, welches Segment bei einer bestimmten Kontaktierung aufleuchtet.

Die Funktionen der Library

Die SevenSeg-Library gestattet es, verschiedenste Informationsformate zu benutzen. Es werden alle gängigen Arten von Displays unterstützt. Die Bibliothek kann für einfache Aufgaben genutzt werden, sie enthält aber auch Funktionen, die den fortgeschrittenen Anwender ansprechen.

Zu den Leistungsmerkmalen der Library gehören insbesondere:

- Unterstützung von Displays mit Dezimalpunkten, Trennungsdoppelpunkten oder Apostrophen
- Es sind sowohl CC- als auch CA-Displays ansprechbar

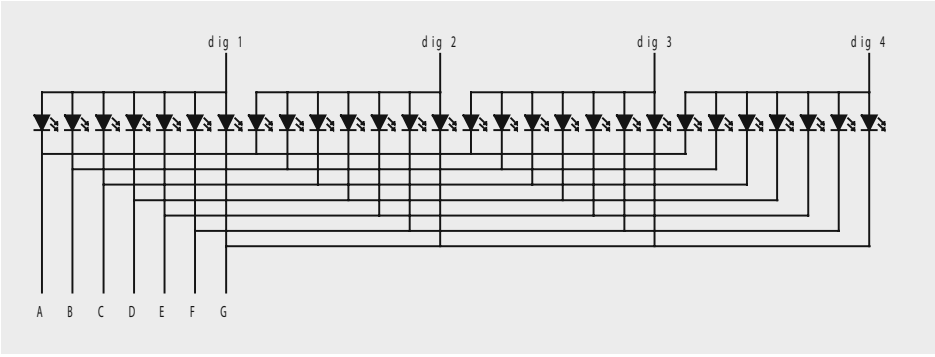


Bild 1: Display mit gemeinsamer Anode (CA)

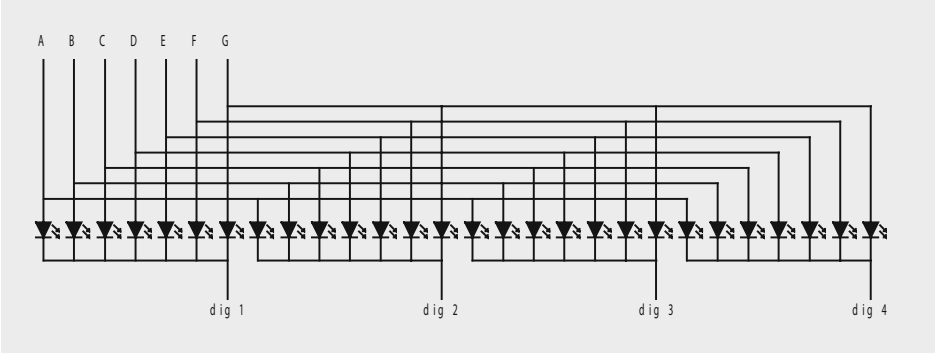


Bild 2: Display mit gemeinsamer Katode (CC)

Bild 3: Nahezu mathematische Genauigkeit: die ersten 12 Ziffern der Zahl Pi

Darüber hinaus sind Routinen enthalten, welche die folgenden Anzeigeformate unterstützen:

- Verschiedene Zahlenformate wie Ganzzahlen, Festkomma- oder Fließkommazahlen
- Textausgabe
- Anzeigen von Uhrzeiten im hh:mm- oder mm:ss-Format
- Automatisches Multiplexing mit einstellbarer Wiederholrate
- Einstellung der Displayhelligkeit über das PWM-Taktverhältnis der Ansteuerung
- Einsatz von Timer-Interrupts in der Multiplexsteuerung
- Vermeidung sogenannter Schattenartefakte

Ansteuerung einer 12-stelligen Anzeige

In einem ersten Anwendungsbeispiel soll eine 12-stellige CA-Anzeige (siehe Bild 3) mittels der Bibliothek angesteuert werden.

Die gemeinsamen Anoden können direkt mit den Arduino-Ports verbunden werden. Die Segment-Pins des Displays werden über geeignete Vorwiderstände mit den Arduino-Ausgängen verbunden. Für erste Experimente können hier 1-kΩ-Widerstände verwendet werden. Falls die Anzeige damit zu dunkel erscheint, können die Widerstandswerte reduziert werden. Wenn ein Datenblatt zum Anzeigemodul vorliegt, kann man die Widerstandswerte natürlich auch aus den erlaubten Maximalströmen berechnen und so die optimale Helligkeit des Displays nutzen. Weitere Details zum Anschluss des Displays an den Arduino finden sich auch im letzten Beitrag zu dieser Artikelserie.

Der Sketch dazu sieht folgendermaßen aus:

```
// 12x7-segment LED display

#include <SevenSeg.h>

// segments         a  b  c  d  e  f  g
SevenSeg disp (7 ,5 ,3 ,1 ,0 ,6 ,4);

const int numOfDigits=12;
// Arduino pins             A4  A3  A2  A5  A0  A1  D12 D13 D8 D9 D10 D11
// digit                    12  11  10   9   8     7     6      5     4    2    2     1
int digitPins [numOfDigits]={ 18, 17, 16, 19, 14, 15,  12,   13,  8,   9,   10,   11};
char str[12];

void setup ()
{ disp.setDigitPins (numOfDigits, digitPins);
}

void loop ()
{ disp.write("314159265359");
}
```

Die Segmente a bis g werden also der Reihe nach an die Digital-Pins D7, D5, D3, D1, D0, D6 und D4 des Arduinos angeschlossen.

Die gemeinsamen Digit-Pins 1 bis 12 des Displays werden entsprechend mit den noch freien Arduino-Ausgängen D8 bis D12 sowie mit den Analog-Ports A0 bis A5 verbunden. Dabei ist zu beachten, dass in diesem Fall die Analog-Pins A0 bis A5 als Digitalausgänge D14 bis D19 geschaltet werden.

Hardware-Varianten

Eine besondere Stärke der SevenSeg-Bibliothek ist die Möglichkeit zur Einstellung verschiedenster Hardware-Varianten. So kann die Belegung der Digit- und Segment-Pins mit

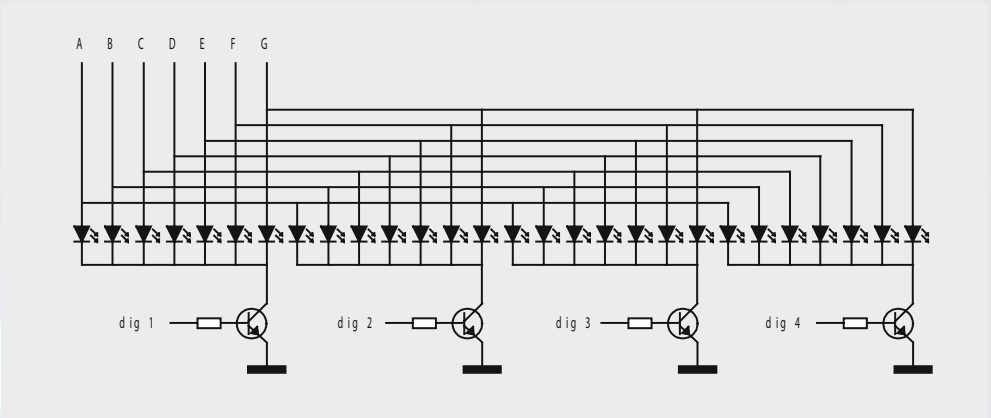


Bild 4: CC-Display mit Transistor-Treibern

der Funktion

SevenSeg(A, B, C, D, E, F, G);

beliebig festgelegt werden. Für A bis G sind lediglich die Port-Pins anzugeben, an welche die Segmente A bis G angeschlossen sind.

Entsprechend kann mit

void setDigitPins(numOfDigits, DigitPins);

die Anzahl der Digits und deren Anschlussreihenfolge festgelegt werden. Die Anzahl der anzuzeigenden Stellen ist dabei in numOfDigits anzugeben, die Pin-Reihenfolge wird durch das Array DigitPins definiert. Über die Funktionen

void setCommonAnode();
void setCommonCathode();

wird die Topologie des Displays festgelegt. Wird keine Topologie angegeben, dann wird ein CA-Display als voreingestellter Wert angenommen.

Die Funktion

void setActivePinState(segActive, digActive);

legt fest, ob die Segmente high- oder low-aktiv sind. Diese Funktionen werden benötigt, wenn Treibertransistoren zum Einsatz kommen. Dies wird erforderlich, wenn ein Arduino-Pin nicht genügend Strom für ein komplettes Digit liefern kann.

Bild 4 zeigt, wie in diesem Fall NPN-Transistoren eingesetzt werden können, um höhere Ströme zu treiben. Während ein Arduino-Pin nur maximal mit 20 mA belastet werden sollte, können mit Kleinsignal-Transistoren (z. B. BC547 o. Ä.) problemlos 100 mA oder mehr geschaltet werden. Hier müsste dann die zugehörige Programmzeile

setActivePinState (HIGH,HIGH);

eingefügt werden.

Darstellung alphanumerischer Zeichen

Wie bereits im letzten Artikel erwähnt wurde, können mit 7-Segment-Anzeigen nicht nur Ziffern, sondern auch einige Buchstaben angezeigt werden. Allerdings hat dieses Verfahren den Nachteil, dass nicht alle Buchstaben gut erkennbar dargestellt werden können. So kann beispielsweise der Großbuchstabe B nicht von der Ziffer 8 unterschieden werden, da in beiden Fällen alle 7 Segmente leuchten müssen. Auch die Buchstaben K oder W lassen sich praktisch nicht in leserlicher Form erzeugen. Für viele System-

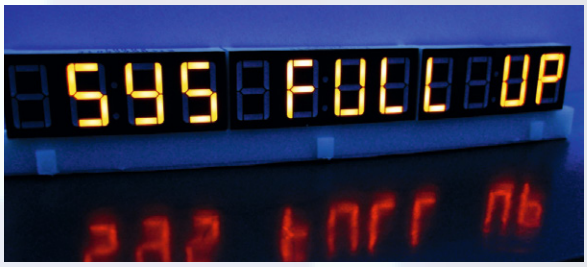


Bild 5: Darstellung von Informationstexten mit 7-Segment-Anzeigen

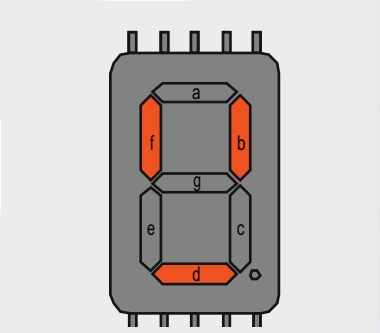


Bild 6: Darstellung des Buchstaben W in der SevenSeg-Library

anzeigen kann man sich allerdings gut behelfen, indem man geeignete Texte wählt.

Damit kann man bei Messgeräten auch Statusanzeigen oder Systemzustände ausgeben, ohne dass ein zusätzliches Display eingebaut werden muss.

Bild 5 zeigt ein Anwendungsbeispiel für die Ausgabe der Information „System full up" in einem 7-Segment-Display.

Die SevenSeg-Library unterstützt diese Anwendung. Für die nicht eindeutig darstellbaren Zeichen werden dabei Ersatzcodes verwendet. Für den Buchstaben W werden beispielsweise die Segmente B, D und F aktiviert (siehe Bild 6).

Digitale Frequenzzähler

Anwendungen, welche viele Anzeigestellen benötigen, sind beispielsweise Frequenzzähler und Counter. Da Frequenzen durch Verwendung von Quarzen mit hoher Genauigkeit erfasst werden können, ist es sinnvoll, auch entsprechend viele Stellen anzuzeigen. Ein

Voltmeter kommt meist mit 3 bis 4 Stellen aus, da Spannungen mit moderatem Aufwand meist nur bis auf 0,1 % genau gemessen werden können. Entsprechendes gilt für Strom- und Widerstandsmessungen.

Elektronische Quarze liefern dagegen problemlos eine Genauigkeit im ppm-Bereich, d. h. Abweichungen liegen in der Größenordnung von 1:1.000.000 (parts per million). Damit man diese Präzision auch anzeigen kann, sind Displays mit einer entsprechenden Stellenzahl erforderlich.

Die Frequenzmessung selbst wird wiederum durch eine Library vereinfacht. Unter

http://interface.khm.de/index.php/lab/interfaces-advanced/arduino-frequency-counter-library/

kann diese wieder kostenlos aus dem Netz geladen werden.

Die Library erfasst Frequenzimpulse über den Arduino-Pin D5, da dieser in der Sekundärfunktion als T1 dient. T1 ist der Eingang für den 16-Bit-Hardware-Counter des Controllers auf dem Arduino. Mit Hilfe dieses Counters können Impulse mit hoher Präzision erfasst werden.

Weitere Informationen zu den internen Hardware-Timern des Arduino-Controllers finden sich in [5] und im Teil 11: „Timer und Counter" zu dieser Artikelserie (siehe ELVjournal August/September 2015) unter Webcode #1433.

Die Impulszählung erfolgt mit einer Torzeit von 1000 ms = 1,000 s. Die Anzahl der Impulse gibt also direkt die an D5 angelegte Frequenz in Hz wieder. Die Torzeit wird in der Programmzeile

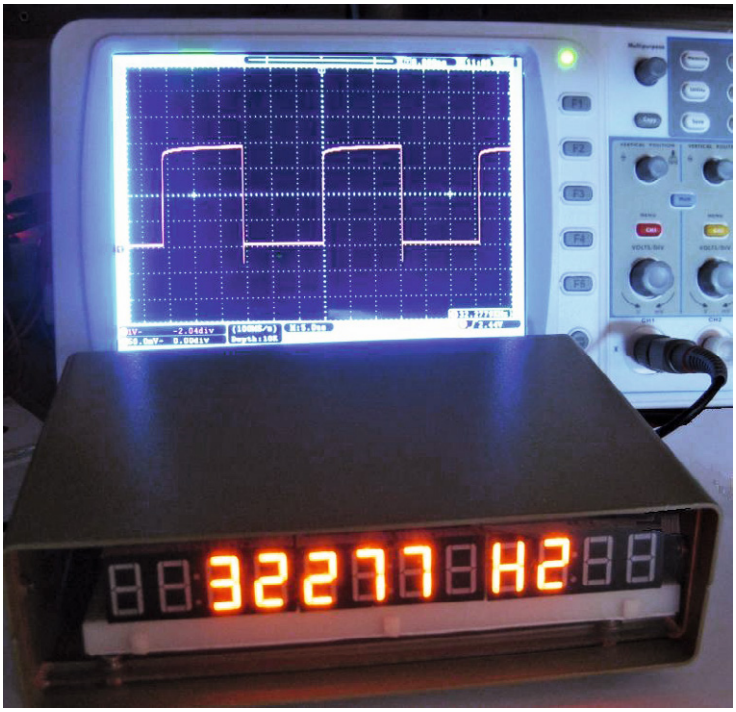
FreqCounter::start(1000) // ...

angegeben.

Im Setup wird eine erste Dummy-Messung ausgeführt, um Fehlmessungen aufgrund von undefinierten Werten im Zählerpuffer zu vermeiden. In der Hauptschleife wird dann kontinuierlich die anliegende Frequenz gemessen und an das 7-Segment-Display ausgegeben.

```
// Frequency display
// using 12x7-segment LED display
// **** for ATmega328, i. e. Arduino UNO ****
// use A0 as D14, A1 as D15, ...

#include <SevenSeg.h>
#include <FreqCounter.h>
float frq;

//             a  b  c  d  e  f  g
SevenSeg disp (7 ,8 ,3 ,1 ,0 ,6 ,4);

const int numOfDigits =10;
int digitPins [numOfDigits]={ 18, 17, 16, 19, 14, 15, 12, 13, 10, 11};

char buf[14];

void setup ()
{ disp.setDigitPins (numOfDigits, digitPins);
  for (int i = 0; i < 300; i++)
  // dry run for settling
  FreqCounter::f_comp=100;   // Cal Value
  FreqCounter::start(1000);  // 1000 ms Gate Time
  while (FreqCounter::f_ready == 0)
    disp.write("counter up");
  delay(10);
}

void loop ()
{ FreqCounter::f_comp=100;
  FreqCounter::start(1000);
  while (FreqCounter::f_ready == 0)
  { frq = FreqCounter::f_freq;
    dtostrf(frq, 7, 0, buf);
    disp.write(buf);
    disp.write("        Hz");
  }
}
```

Der Frequenzzähler in Aktion



Der Frequenz-Messwert „frq“ wird dazu über die Funktionen

```
dtostrf(frq, 7, 0, buf);
```

in eine Puffer-Zeichenkette „buf“ umgewandelt. Beim Befehl „dtostrf“ handelt es sich um eine Standart-C-Funktion, die eine Dezimalzahl in eine ASCII-Zeichenkette umwandelt.

Die Parameter 7 und 0 steuern die Formatierung. Der neue String enthält so mindestens 7 Zeichen und 0 Nachkommastellen. Auf diese Weise wird die Frequenz rechtsbündig im LED-Display dargestellt. Fehlende Stellen werden durch „Blanks“ aufgefüllt, so dass keine unschönen führenden Nullen entstehen.

Weitere Details zu diesen Funktionen finden sich in [5] und [6]. Durch die Kombination dieser beiden Funktionen wird erreicht, dass die Frequenz an der korrekten Position im Display ausgegeben wird.

Der Wert wird dann in

```
disp.write(buf);
```

angezeigt.

Schließlich wird noch über

```
disp.write("      Hz");
```

die Einheit der Frequenzmessung (Hz für Hertz) eingeblendet.

Die maximal messbare Frequenz liegt bei ca. 4 MHz; ein Wert, der für viele Anwendungen vollkommen ausreichend ist. Die Messgenauigkeit wird lediglich durch den 16-MHz-Quarz des Arduino begrenzt. Sie beträgt typischerweise nur wenige ppm und ist somit für die meisten nicht professionellen Anwendungen vollkommen ausreichend.

Ausblick

Nachdem in diesem Beitrag die Anwendung und Ansteuerung von mehrstelligen 7-Segment-Anzeigen ausführlich diskutiert wurde, wird im nächsten Serienbeitrag die sogenannte LED-Punktmatrix-Displaytechnologie im Vordergrund stehen.

Im Gegensatz zu den 7-Segment-Anzeigen erlaubt dieser Displaytyp auch die klar lesbare Darstellung aller Buchstaben des Alphabets.

Darüber hinaus ist sogar die Anzeige einfacher Grafiken möglich. Durch Zusammenfügen mehrerer Matrixeinheiten können auf diese Weise auch größere Displays aufgebaut werden, bis hin zu den bekannten Mega-Werbeflächen auf öffentlichen Plätzen, Flughäfen oder Bahnhöfen. ELV

Weitere Infos:

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

[2] Mikrocontroller-Onlinekurs, Franzis-Verlag, exklusiv für ELV, 2011, Best.-Nr. CG-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. CG-10 68 46, € 129,–

[5] G. Spanner: AVR-Mikrocontroller in C programmieren, Franzis-Verlag 2010, Best.-Nr. CG-09 73 52, € 39,95

[6] G. Spanner: AVR-XMEGA-Mikrocontroller, Elektor-Verlag 2015, Best.-Nr. CG-12 07 62, € 39,80

Preisstellung Februar 2016 – aktuelle Preise im Web-Shop

Empfohlene Produkte	Best.-Nr.	Preis
Arduino UNO	CG-10 29 70	€ 27,95
Mikrocontroller-Onlinekurs	CG-10 20 44	€ 99,–

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