Arduino verstehen und anwenden

Teil 3: Einführung in die Programmierung

Der dritte Teil der Artikelreihe zum Arduino führt in die Struktur der Programmiersprache „Processing" ein. Die Themen in diesem Beitrag sind:

- Die Grundstruktur eines Processing-Programms
- Praxisbeispiel LED-Ansteuerung: 12-Kanal-Lauflicht

Im Praxisteil wird diesmal bereits ein etwas komplexerer Aufbau realisiert. Mit 12 LEDs entsteht ein flexibel einsetzbares Lauflicht. Als Anwendungsübung können verschiedene Laufgeschwindigkeiten und unterschiedliche Lichtmuster programmiert werden.

Die Grundstruktur eines Processing-Programms

Im Folgenden soll das schon aus dem ersten Beitrag bekannte Programm „Blink" genau analysiert werden.

```
// Blink: Using LED „L"

int ledPin =  13;     // Definitions
// Setup routine for digital port configuration

void setup()
{ // Port as output
   pinMode(ledPin, OUTPUT);
}

// Main program

void loop()
{ digitalWrite(ledPin, HIGH);    // LED on
  delay(1000);                   // wait 1 sec = 1000 ms
  digitalWrite(ledPin, LOW);     // LED off
  delay(1000);                   // wait 1 sec = 1000 ms
}
```

Die erste Zeile

```
// Blink: Using LED „L"
```

stellt einen sogenannten Kommentar dar.

Kommentare werden durch den doppelten Slash „//" gekennzeichnet. Der nach diesem Zeichen stehende Text wird vom Compiler nicht beachtet. Er dient also nur dazu, Programme leichter lesbar zu gestalten.

TIPP I: Nicht zu sparsam mit Kommentaren umgehen! Langfristig sind nur gut kommentierte Programme wirklich nützlich.

Die nächste Zeile

```
int ledPin =  13;     // Definitions
```

definiert den Pin 13 als „ledPin". Damit kann man nun überall im Sketch ledPin schreiben, wenn der Pin 13 gemeint ist.

Es ist sicher leicht einzusehen, dass durch solche Definitionen die Programmierarbeit erheblich erleichtert wird. Kommen im Laufe der Programmentwicklung mehrere LEDs ins Spiel, so kann man sich deren Pin-Nummern kaum mehr merken. Sinnvolle Bezeichnungen dagegen können wesentlich leichter im Kopf behalten werden.

Die Anweisung

```
void setup()
```

ist eine Spezialität von Processing. Hier wird unter anderem festgelegt, welche Aufgaben die einzelnen Arduino-Pins erfüllen sollen.

In unserem Fall wird der oben definierte ledPin als Ausgang genutzt:

```
pinMode(ledPin, OUTPUT);
```

In einem Processing-Sketch muss immer auch die Anweisung void setup() enthalten sein, selbst wenn sie überhaupt keine Definitionen enthält. Durch ihren Aufruf werden nämlich verschiedene interne Zuordnungen ausgeführt, die die Arbeit mit dem Arduino erleichtern.

Mit

```
void loop()
```

startet das Hauptprogramm. Typisch für Mikrocontroller ist, dass das Hauptprogramm praktisch immer innerhalb einer Endlosschleife („loop") abläuft. Da ein Mikrocontroller nicht über ein eigenes Betriebssystem verfügt, würde er am Ende eines Programms einfach stehenbleiben oder undefinierte Aktionen ausführen. Erst nach einem Reset könnte die sinnvolle Abarbeitung von Befehlen wieder aufgenommen werden.

Um dieses Verhalten zu verhindern, muss das Hauptprogramm also in einer Endlosschleife ablaufen, die dafür sorgt, dass der Prozessor so lange arbeitet, bis die Betriebsspannung abgeschaltet wird.

Mit der Anweisung

```
digitalWrite(ledPin, HIGH);
```

wird nun der ledPin auf „HIGH", d. h. hohes Potential, in unserem Fall auf 5 V geschaltet. Damit beginnt die LED zu leuchten.

Mit der Anweisung

```
delay(1000);
```

erhält der Prozessor den Auftrag, 1000 ms = 1 s zu warten.

Schließlich wird mit

```
digitalWrite(ledPin, LOW);
```

die LED wieder ausgeschaltet und das Spiel beginnt nach einer Pause von 1 s von vorne, da wir uns ja in einer Endlosschleife befinden.

Die LED blinkt also nun mit einer Periodendauer von 2 s oder einer Frequenz von 0,5 Hz. Im Projekt „Alarmanlagensimulator" im letzten Beitrag wurde ja bereits gezeigt, wie man die Blinkfrequenz und -dauer den gegebenen Erfordernissen anpassen kann.

Praxisbeispiel: Lauflicht für Modellflughäfen oder -eisenbahnen

Wie bereits beim Alarmanlagensimulator werden auch hier externe LEDs verwendet. Nachdem bislang aber immer mit nur einer externen LED gearbeitet wurde, soll der Arduino nun mehrere LEDs steuern.

Mit 12 LEDs kann bereits ein eindrucksvolles Lauflicht aufgebaut werden. Derartige Lauflichter sind z. B. auf den Landebahnen von Großflughäfen oder an Autobahnbaustellen zu finden. Dort soll der laufende

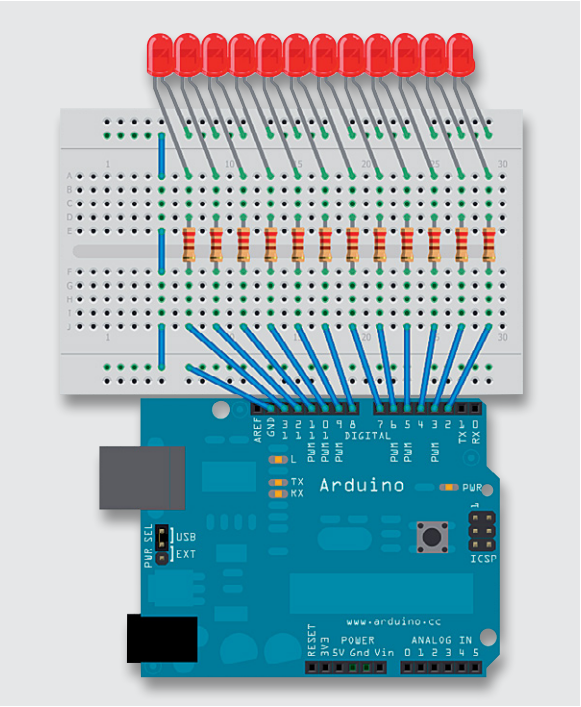

Bild 1: LED-Lauflicht

Lichtpunkt Flug- oder Fahrzeugen den richtigen Weg weisen. Aber auch in Partykellern, Diskotheken oder auf Volksfesten erfreuen sich entsprechende Lichteffekte großer Beliebtheit.

Jede der 12 LEDs bekommt ihren eigenen Vorwiderstand. Prinzipiell wäre es auch möglich, mit einem einzigen Widerstand auszukommen. Dieser müsste dann in die gemeinsame Masseleitung aller Dioden eingesetzt werden. Falls immer nur eine LED aktiv sein soll, ist die Lösung mit nur einem Vorwiderstand durchaus akzeptabel.

Allerdings ergibt sich ein Problem, wenn mehr als eine LED leuchten soll. In diesem Fall müssen sich alle aktiven LEDs sozusagen einen Widerstand teilen. Je mehr LEDs eingeschaltet werden, desto größer wäre der Gesamtstrom durch den Widerstand. Dies würde aber natürlich zu einem immer höheren Spannungsabfall am Widerstand führen, so dass die verbleibende LED-Spannung immer weiter abnehmen würde. Je mehr LEDs also eingeschaltet würden, desto dunkler würde jede einzelne LED werden. Wird für jede LED ein eigener Vorwiderstand verwendet, tritt dieser meist unerwünschte Effekt nicht auf. Weitere Details und Informationen zum Ansteuern von LEDs finden sich auch in [1]. Bild 1 zeigt den Aufbau.

TIP II: Falls eine oder mehrere LEDs nicht leuchten, sind sie wahrscheinlich falsch herum eingebaut. Ein Umpolen der entsprechenden LEDs behebt das Problem.

Vorwiderstände und Low-Current-LEDs

Noch ein Wort zur Berechung der Vorwiderstände. Die meisten einfachen LEDs benötigen einen Strom von ca. 20 mA. Der Spannungsabfall über einer roten LED beträgt immer etwa 1,7 V. Damit ergibt sich:

Rv = (5 V – 1,7 V) / 20 mA = 165 Ω

Mit den gewählten 220-Ω-Widerständen (Farbringfolge Rot-Rot-Braun) liegt man also auf der sicheren Seite.

Will man Lichtmuster erzeugen, in welchen alle 12 LEDs gleichzeitig betrieben werden, so ergibt sich ein Gesamtstrom von 12 x 20 mA = 240 mA. Dies stellt bereits eine erhebliche Belastung für den USB-Port dar. In diesem Fall sollte man darüber nachdenken, den Arduino mit einem externen Netzteil zu betreiben.

Eine andere Möglichkeit wäre die Verwendung von sogenannten Low-Current-LEDs. Diese benötigen bei sehr guten Helligkeitswerten nur einen Strom von 4 mA. Die Vorwiderstände ergeben sich in diesem Falle zu:

Rv = (5 V – 1,7 V) / 4 mA = 825 Ω

Mit 1-kΩ-Widerständen erreicht man immer noch eine gute Helligkeit und der Gesamtstromverbrauch bei 12 aktiven LEDs reduziert sich auf unter 50 mA.

Das Steuerprogramm zum Lauflichteffekt

Um den Code etwas kompakter zu gestalten, soll hier kurz auf eine neue Anweisung vorgegriffen werden. Die Anweisung

for(int i=a; i<=b; i++)

sorgt dafür, dass der nachfolgende Befehlsblock wiederholt ausgeführt wird. Zudem kann die sogenannte Laufvariable i in diesem Befehlsblock zu Steuerzwecken verwendet werden. So werden beispielsweise mit

for(int i=1; i<=3; i++)

die nachfolgenden Anweisungen 3-mal ausgeführt. Die genaue Beschreibung, wie diese und andere sogenannte Kontrollstrukturen arbeiten, folgt im nächsten Beitrag zu dieser Artikelserie. Hier genügt es zunächst zu wissen, dass die „for"-Anweisung zum definierten Wiederholen von Befehlen angewendet werden kann.

Damit sollte das Programm zum Lauflichteffekt keine Verständnisprobleme mehr bereiten:

```
// LED chaser

void setup()
{for(int i=2; i<=13; i++)
  pinMode(i, OUTPUT);           // set pin 2 to 13 as outputs
}

void loop()
{for(int i=2; i<=13; i++)
  {  digitalWrite(i, HIGH);     // set the LED on
     delay(100);                // wait for a 1/10 second
     digitalWrite(i, LOW);      // set the LED off
  }
}
```

Im Setup werden zunächst die Ports 2 bis 13 als Ausgänge geschaltet:

```
void setup()
{   for(int i=2; i<=13; i++)
    pinMode(i, OUTPUT);          //....
}
```

Tipp III: Die Digital-Pins 0 und 1 besitzen eine zusätzliche Sonderfunktion. Neben der einfachen Portfunktion dienen sie auch als Ein- und Ausgang für die serielle Schnittstelle. Dies kann bei einigen Anwendungen zu unerwarteten Ergebnissen führen. Es ist daher sinnvoll, diese Ports nicht zu verwenden, solange dies nicht unbedingt erforderlich ist.

Im Hauptprogramm wird dann zunächst mit

```
digitalWrite(i, HIGH);   // set the LED on
```

die erste LED in der Kette eingeschaltet. Nach einer Wartezeit von 100 ms – delay(100); – wird sie wieder ausgeschaltet. Dann folgt die zweite LED, bis sich das Spiel nach der zwölften LED wiederholt. Auf diese Weise entsteht der gewünschte Lauflichteffekt.

Natürlich ist dies nur eine Möglichkeit, um ein Lauflicht zu realisieren. Im nächsten Beitrag zu dieser Artikelreihe lernen Sie die sogenannten Kontrollstrukturen von C kennen. Damit kann das obenstehende LED-Lauflicht-Beispiel elegant und einfach um weitere interessante Lichteffekte ergänzt werden.

Aber auch bei dieser einfachen Version sind wieder verschiedene Varianten realisierbar. So können beispielsweise die folgenden Übungen durchgeführt werden.

1. Verändern Sie die Geschwindigkeit des Lauflichts
2. Programmieren Sie ein sogenanntes Pendel-Lauflicht. Das bedeutet, dass der Lichtpunkt nicht nur in eine Richtung laufen und dann wieder bei der ersten LED starten soll, sondern dass er hin und her pendelt.

LED-Band

Abschließend soll noch eine weitere Programm-Variante vorgestellt werden. Hier wird ein sich in der Länge veränderndes Lichtband erzeugt (Bild 2).

In der Praxis werden solche Lichtbänder in ähnlicher Form häufig für die Anzeige von analogen Messwerten wie etwa Ladezuständen von Akkumulatoren oder Maschinendrehzahlen eingesetzt.

Der Vorteil der einzelnen Vorwiderstände tritt dabei deutlich zu Tage. Selbst wenn alle 12 LEDs leuchten, ist die Helligkeit jeder LED gegenüber der Helligkeit bei nur einer aktiven LED unverändert.

```
// LED band

void setup()
{for(int i=2; i<=13; i++)
  pinMode(i, OUTPUT);              // set pin 2 to 13 as outputs
}

void loop()
{for(int i=2; i<=13; i++)
  {  digitalWrite(i, HIGH);        // set the LED on
     delay(100);                   // wait for a 1/10 second
  }
 for(int i=2; i<=13; i++)
  {  digitalWrite(i, LOW);         // set the LED off
     delay(100);                   // wait for a 1/10 second
  }
}
```

Ausblick

In diesem Beitrag kam erstmals eine etwas umfangreichere externe Beschaltung des Arduinos zum Einsatz. Die grundlegende Programmstruktur und einige Arduino-spezifische Befehle wurden genauer erläutert.

Der nächste Beitrag in der Reihe wird näher auf die Programmierung eingehen und es werden weitere Befehle im Detail vorgestellt. Damit können dann auch komplexere Programme für eigene Projekte erstellt werden.

Empfohlenes Material

Im Mikrocontroller-Onlinekurs [2] werden die hier vorgestellten Grundlagen weiter vertieft. Das Hardwarepaket zu diesem Kurs enthält ein umfangreiches Bauteilesortiment für den erfolgreichen Einstieg in die Mikrocontrollertechnik.

Auch viele Franzis Lernpakete wie etwa „Elektronik" enthalten geeignetes Material – wie ein lötfreies Steckbrett, Widerstände, LEDs etc. –, das für den Aufbau von Schaltungen mit dem Arduino Micro gut geeignet ist. ELV

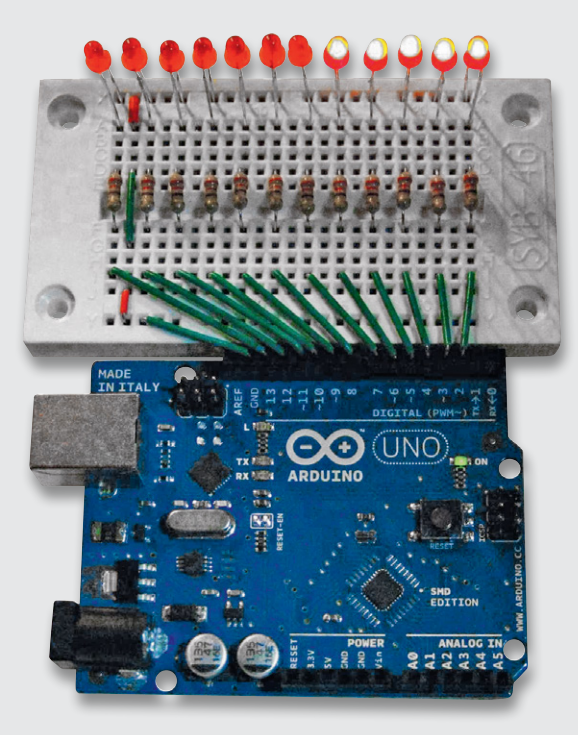

Bild 2: Das variable LED-Band in Aktion

Weitere Infos:

[1] G. Spanner: Arduino – Schaltungsprojekte für Profis, Elektor-Verlag, 2012, Best.-Nr. J2-10 94 45
[2] Mikrocontroller-Onlinekurs, Franzis-Verlag, exklusiv für ELV, 2011, Best.-Nr. J2-10 20 44

Preisstellung Februar 2014 – aktuelle Preise im Web-Shop

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
Arduino-Uno-Platine R3	J2-10 29 70	€ 27,95
USB-2.0-Verbindungskabel A auf B	J2-09 55 56	€ 0,95
Mikrocontroller-Onlinekurs	J2-10 20 44	€ 99,–
LEDs, z. B. Kemo Leuchtdioden, ca. 30 Stück	J2-10 66 60	€ 1,65
Widerstände, Kemo-Sortiment, ca. 200 Stück	J2-10 66 57	€ 1,85

Steckbretter sind in verschiedenen Franzis Lernpaketen enthalten oder im Web-Shop unter www.elv.de/experimentierboards.html erhältlich.

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