Arduino verstehen und anwenden

Teil 7: Schnittstellenpraxis – Kommunikation mit Laptop und PC

Im siebten Teil der Artikelserie „Arduino verstehen und anwenden" geht es um die Übertragung von Daten über Schnittstellen. Die zentrale Schnittstelle des Arduino ist das USB-Interface, über das jeder aktuelle Arduino verfügt.

Über die USB-Buchse wird der Arduino nicht nur mit Strom versorgt, sondern auch die gesamte Datenkommunikation läuft über diese Schnittstelle. Dabei sind zwei grundlegende Methoden zu unterscheiden:

- Übertragung von Sketchen zum Arduino
- Austausch von Daten zwischen PC und Arduino, während auf dem Prozessor ein Programm abläuft

In diesem Artikel werden die Funktion dieser Schnittstelle und ihre möglichen Anwendungen genauer behandelt.

Serielle Datenübertragung

Wenn Daten zwischen einem Mikrocontroller und einem PC ausgetauscht werden sollen, so ist dafür die serielle Schnittstelle noch immer das Mittel der Wahl. Dies mag in Zeiten der allgegenwärtigen USB-Schnittstelle etwas überraschen, und schließlich ist der Arduino ja auch über eine USB-Schnittstelle mit dem PC verbunden.

Wenn man den Aufbau des Arduino-Boards aber etwas genauer betrachtet, so stellt man fest, dass der Mikrocontroller tatsächlich nur über eine klassische serielle Verbindung kommuniziert. Erst ein weiterer Baustein sorgt für die Umsetzung der RS232-Signale auf ein USB-kompatibles Format.

In den früheren Arduino-Versionen wie dem Duemillenove wurde die Umsetzung von einem Spezialbaustein übernommen, dem FT232RL der Firma Future Technologies.

Da der FT323RL verhältnismäßig teuer ist, wurde dieser Baustein beim aktuellen Arduino Uno durch einen zweiten AVR-Controller, den ATmega16U2, ersetzt. Dieser Controllertyp verfügt über eine integrierte USB-Schnittstelle und kann mittels einer speziellen Firmware USB-Signale in das RS232-Format umwandeln.

Noch einen Schritt weiter ist der Arduino Leonardo. Hier wurde der Controllertyp ATmeg32U4 als Hauptcontroller eingesetzt. Dieser kann, wie der Mega16U4, ebenfalls direkt USB-Signale verarbeiten. Auch im Arduino Micro kommt dieser Chip zum Einsatz. Erst aufgrund der besonders kleinen Gehäusebauform des Chips konnte der Micro auf die Größe eines klassischen DIL-ICs reduziert werden.

Die USB-Schnittstelle ist natürlich auch ein weiterer Grund für die große Beliebtheit der Arduino-Boards. Durch diese Schnittstelle kann der Arduino an praktisch jeden modernen PC oder Laptop problemlos angeschlossen werden. Der Anwender braucht sich um Schnittstellenwandler etc. keine Gedanken mehr zu machen.

Aber auch wenn die RS232-Schnittstelle in der modernen PC-Technik praktisch ausgestorben ist, so ist sie doch in anderen Bereichen noch weit verbreitet. So ist RS232-Kommunikation noch sehr häufig in folgenden Anwendungen zu finden:

- Externe GPS-Module, sogenannte „GPS-Mäuse"
- Seriell angesteuerte LC-Displays
- Messgerätetechnik
- Industrielle Steuerungen etc.

Auch bei der Verbindung von einzelnen Mikrocontrollern untereinander spielt die serielle Schnittstelle noch eine große Rolle.

Außerdem können gerade auch sehr preisgünstig erhältliche oder noch vorhandene ältere PCs mit RS232-Schnittstelle in Kombination mit Mikrocontrollern nutzbringend eingesetzt werden. Da hier meist keine sehr hohen Anforderungen an die Leistungsfähigkeit des Rechners gestellt werden, bieten sich solche Rechner als „Terminaleinheiten" für den Arduino an.

Grundlagen der seriellen Kommunikation

Prinzipiell kann man zwei Methoden der Datenübertragung unterscheiden:

- Parallele Kommunikation
- Serielle Kommunikation

Bei der parallelen Datenkommunikation werden mehrere, meist acht Bit, gleichzeitig übertragen. Dies erfordert aber immer eine parallele Verlegung von acht Leitungen bzw. acht Adern in einem Kabel (plus eine Masseverbindung), d. h. einen vergleichsweise hohen technischen Aufwand. Insbesondere bei sehr hohen Datenraten treten erhebliche Probleme mit

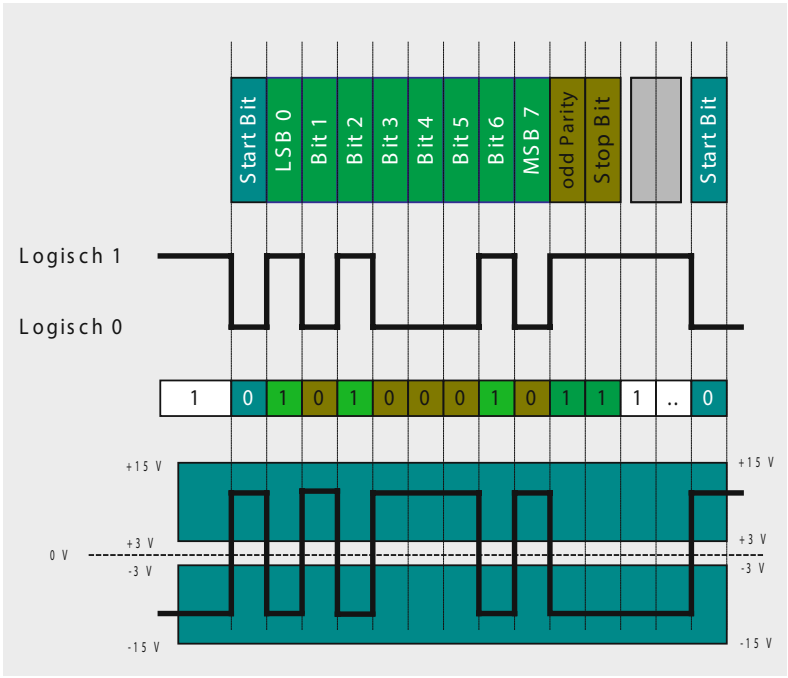

Bild 1: RS232-Protokoll

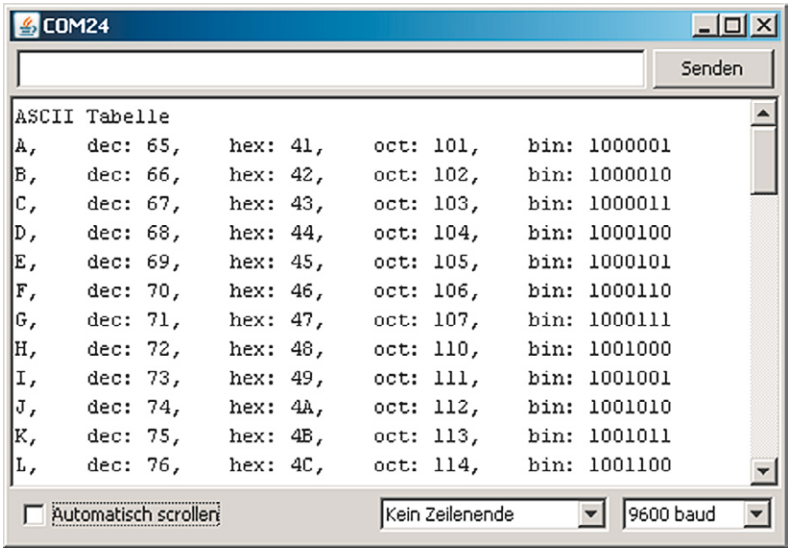

Bild 2: Ausgabe einer ASCII-Tabelle auf die serielle Schnittstelle

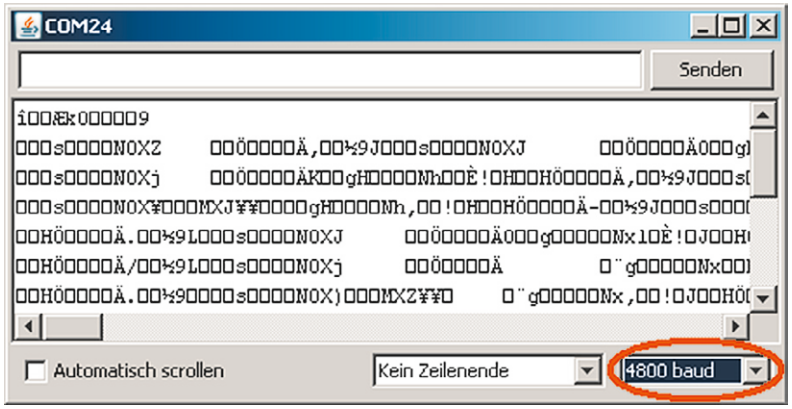

Bild 3: So sieht die Ausgabe mit falsch eingestellter Baudrate aus!

der Synchronisierung der acht Kanäle auf, so dass alle moderneren Übertragungsverfahren, wie etwa auch USB, nur noch seriell ausgeführt werden.

Bei der seriellen Übertragung werden die Datenbits zeitlich nacheinander übertragen. Im einfachsten Fall kommt man daher mit einer Signalleitung aus. Für eine bidirektionale Verbindung sind zwei Leitungen ausreichend (plus Masse).

Um eine korrekte Interpretation der übertragenen Daten zu gewährleisten, müssen Datensender und -empfänger ein gemeinsames Protokoll benutzen. In diesem Protokoll müssen die wichtigsten Parameter festgelegt sein wie:

- Übertragungsgeschwindigkeit
- Bitdefinition
- Start- und Endbedingung für die Übertragung

Auf PC-Seite werden die verfügbaren seriellen Schnittstellen mit COM bezeichnet. So verfügen die meisten PCs intern noch über diese Schnittstelle, auch wenn sie nicht mehr über eine Buchse am Gehäuse zur Verfügung gestellt wird. Bei der Installation von sogenannten „virtuellen" COM-Schnittstellen sind die internen Ports dann aber oft noch als COM1 oder COM2 vorbelegt.

Die serielle Datenübertragung haben wir prinzipiell schon bei der ersten Programmierung des Arduino kennengelernt. Ein Anwendungsprogramm wird nämlich ebenfalls über das serielle Protokoll zum µC übertragen.

Deshalb war es auch notwendig, bei der Installation der Entwicklungsumgebung eine spezielle COM-Schnittstelle auszuwählen.

Bild 1 zeigt die prinzipielle Funktion der seriellen Schnittstelle. Die Übertragung beginnt mit einem sogenannten Startbit. Dann folgen die Datenbits, wobei mit dem niedrigstwertigen Bit, dem sogenannten Least Significant Bit (LSB), begonnen wird. Den Abschluss der Daten bildet entsprechend das höchstwertige Bit (MSB: Most Significant Bit). Abschließend folgen das Parity- und ein oder 2 Stopp-Bits.

Ursprünglich wurden für die Übertragung auf elektrischer Ebene Spannungen zwischen ±15 V verwendet (siehe Bild 1 unten).

Bei Controller-Anwendungen kommt dagegen meist die sogenannte TTL-kompatible Version zum Einsatz. Diese verwendet Spannungen von 0 und +5 V für die Bitübertragung.

Die serielle Schnittstelle am Mikrocontroller

Beim Arduino-Board steht die serielle Schnittstelle an den Pins 0 und 1 zur Verfügung Diese Pins sind deshalb auch mit RX und TX für Receive (Empfangen) und Transmit (Senden) gekennzeichnet. Diese beiden Pins sind auf dem Board direkt mit dem RS232-zu-USB-Wandler verbunden.

Damit wird die Kommunikation des Controllers mit einem PC sehr einfach. Der einzige Parameter, der noch eingestellt werden muss, ist die Übertragungsgeschwindigkeit. Diese wird auch als „Baudrate" bezeichnet.

Folgende Datenraten stehen zur Verfügung:

300	1200
2400	4800
9600	14.400
19.200	28.800
38.400	54.760
115.200	

Die Standard-Einstellung ist 9600 Baud.

Die Befehle zur Steuerung der seriellen Schnittstelle

Für die Steuerung der seriellen Schnittstelle können die folgenden Befehle eingesetzt werden:

```
Serial.begin(baudrate);
```

Hier wird die serielle Schnittstelle initialisiert. Die Baudrate ist der einzige hierfür erforderliche Parameter.

Es ist lediglich darauf zu achten, dass die hier angegebene Baudrate mit der im Empfänger auf der PC-Seite identisch ist.

Mit

```
Serial.print();
```

können dann bereits Informationen an den PC übertragen werden.

```
Serial.println();
```

fügt nach der Ausgabe einen Zeilenneuanfang („Line“) ein.

Liefert die Abfrage von

```
Serial.available()
```

einen Wert >0, so stehen Daten an der Schnittstelle bereit.

Diese können dann mit

```
Serial.read();
```

eingelesen werden.

Der Arduino geht auf Sendung: Übertragung von Daten vom Arduino an den PC

Als erstes Praxisbeispiel soll eine ASCII-Tabelle vom Arduino zum PC gesendet werden.

Der Sketch dazu kann wie folgt aussehen:

```
// ASCII table to RS232

void setup()
{  Serial.begin(9600);
   Serial.println("ASCII Tabelle");

   for (int myByte = 65; myByte <= 126; myByte++)
  {  Serial.write(myByte);
     Serial.print(",    dec: ");
     Serial.print(myByte);
     Serial.print(",    hex: ");
     Serial.print(myByte, HEX);
     Serial.print(",    oct: ");
     Serial.print(myByte, OCT);
     Serial.print(",    bin: ");
     Serial.println(myByte, BIN);
  }
}

void loop() {}
```

Neben dem ASCII-Zeichen selbst werden hier noch die zugehörigen Codes in dezimaler (dec), hexadezimaler (hex), oktaler (oct) und binärer (bin) Form dargestellt. Bild 2 zeigt die Ausgabe im Seriellen Monitor der Arduino-IDE.

Einer der häufigsten Fehler bei der Arbeit mit der seriellen Schnittstelle ist, dass die Baudraten auf Sender- und Empfängerseite nicht übereinstimmen.

Bild 3 zeigt das Ergebnis, wenn die Sendebaudrate wie im obigen Code-Beispiel auf 9600 gesetzt ist, die Empfangsbaudrate am Seriellen Monitor jedoch auf 4800 steht.

Sollten Sie also in Zukunft unverständliche Zeichenfolgen auf Ihrem Bildschirm haben, ist es immer eine gute Idee, zu prüfen, ob die Baudraten auf Sender- und Empfängerseite übereinstimmen.

Der Arduino an der kurzen Leine: direkte Steuerung des Controllers mit dem PC

Eine sehr wichtige Aufgabe in der Technik ist die Steuerung von Hardware-Komponenten von einem Rechner aus. Diese Aufgabe ist gewissermaßen die Umkehrung des letzten Praxisbeispiels.

Das folgende Programmbeispiel zeigt, wie man vom PC aus eine LED ein- und ausschalten kann.

Nach dem Laden des Programms kann die LED 13 über den Datenmonitor geschaltet werden. Das Senden einer „1“ schaltet die LED ein. Wird eine „0“ gesendet, so erlischt die LED. Für das Senden der Werte kann wieder der Serielle Monitor eingesetzt werden (Bild 4). Die gewünschten Werte werden dort in das oberste Feld eingetragen und mit dem „Senden“-Button zum Arduino übertragen.

```
// receive serial data

int recData = 0;
int LED = 13;            //on-board LED

void setup()
{ pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
  Serial.begin(9600);
}

void loop()
{ if (Serial.available() > 0)
  { recData = Serial.read();
    if (recData == '0')
      digitalWrite(LED, LOW);
    if (recData == '1')
      digitalWrite(LED, HIGH);
  }
  delay(100);
}
```

Selbstverständlich können nicht nur die logischen Pegel „0“ und „1“ gesendet werden. Auch beliebige andere Werte sind übertragbar. Das folgende Programmbeispiel zeigt, wie ein LED-Dimmer vom PC aus gesteuert werden kann.

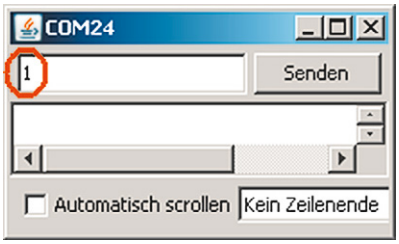

Bild 4: Senden eines ASCII-Wertes zum Arduino

```
// remote controlled LED dimmer

int recData = 0;
int LED = 3;            // PWM-LED

void setup()
{ pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
  Serial.begin(9600);
}

void loop()
{ if (Serial.available() > 0)
  { recData = Serial.read();
    if (recData == '0') analogWrite(LED, 0);
    if (recData == '1') analogWrite(LED, 1);
    if (recData == '2') analogWrite(LED, 2);
    if (recData == '3') analogWrite(LED, 4);
    if (recData == '4') analogWrite(LED, 8);
    if (recData == '5') analogWrite(LED, 16);
    if (recData == '6') analogWrite(LED, 32);
    if (recData == '7') analogWrite(LED, 64);
    if (recData == '8') analogWrite(LED, 128);
    if (recData == '9') analogWrite(LED, 255);
  }
  delay(100);
}
```

Nachdem das Programm geladen ist, können Werte zwischen 0 und 9 zum Arduino gesendet werden.

Eine an Pin 3 angeschlossene externe LED kann so in ihrer Helligkeit gesteuert werden. Der Einsatz einer externen LED an Pin 3 ist erforderlich, da die interne LED 13 nicht über den analogWrite angesteuert werden kann. Diese Möglichkeit bieten nur die mit einer Tilde (~) gekennzeichneten sogenannten PWM-Ausgänge. PWM steht dabei für Pulsweitenmodulation.

Weitere Details zu diesem Thema werden in einem späteren Artikel zu dieser Serie behandelt.

Betrachtet man das Programm genauer, so fällt auf, dass die Werte im analogWrite-Befehl nicht linear vom übertragenen Datenwert abhängen, sondern exponentiell. Dies hängt damit zusammen, dass das menschliche Auge ein „logarithmischer Empfänger" ist.

Würde man die Helligkeit der LED linear ändern, so würden Helligkeitsstufen bei geringer Helligkeit der LED zunächst sehr groß erscheinen. Bei größeren Helligkeitswerten dagegen könnte das Auge kaum mehr Unterschiede wahrnehmen.

Durch die exponentiell zugeordneten Helligkeitswerte wird das logarithmische Verhalten des Auges dagegen kompensiert und die Helligkeitsstufen erscheinen nahezu gleich groß.

Ausblick

Nachdem in diesem Artikel die klassischen Punkt-zu-Punkt-Schnittstellen behandelt wurden, steht im nächsten Artikel die Netzwerktechnik im Vordergrund.

Der Arduino wird dann über die Ethernet-Schnittstelle in das Heimnetzwerk eingebunden. Dafür ist ein spezielles Shield erforderlich, oder man verwendet gleich eine Arduino-Version mit integriertem Ethernet-Anschluss.

Ist der Arduino erst einmal in das Heimnetzwerk integriert, so kann man mit einem beliebigen Browser wie etwa dem Internet Explorer oder Google Chrome auf den Controller zugreifen. Alle im Artikel „Sensortechnik und Messwerterfassung" vorgestellten Sensoren können dann in das heimische Netzwerk integriert werden.

Dem modernen Thema „Heimautomatisierung" sind so keine Grenzen mehr gesetzt. Der Arduino kann etwa Rollläden in Abhängigkeit von der Sonneneinstrahlung steuern.

Aber auch eine effektive Kontrolle oder sogar die raumindividuelle Steuerung der hauseigenen Heizungsanlage kann so realisiert werden.

Empfohlenes Material

- Arduino Micro,
  Best.-Nr. J6-10 97 74, € 24,95
- Mikrocontroller-Onlinekurs,
  Best.-Nr. J6-10 20 44, € 99,–
- Auch viele Lernpakete von Franzis wie etwa „Elektronik mit ICs" enthalten Materialien wie ein lötfreies Steckbrett, Widerstände und LEDs etc., die für den Aufbau von Schaltungen mit dem Arduino Micro gut geeignet sind.

ELV

Weitere Infos:

- Mikrocontroller-Onlinekurs, Franzis-Verlag, exklusiv für ELV, 2011, Best.-Nr. J6-10 20 44, € 99,–
- G. Spanner: Arduino – Schaltungsprojekte für Profis, Elektor-Verlag 2012, Best.-Nr. J6-10 94 45, € 39,80
- Lernpaket „AVR-Mikrocontroller in C programmieren", Franzis-Verlag 2012, Best.-Nr. J6-09 73 52, € 39,95

Preisstellung Oktober 2014 – aktuelle Preise im Web-Shop

Empfohlene Produkte/Bauteile:	Best.-Nr.	Preis
Arduino Micro	J6-10 97 74	€ 24,95
Arduino-Uno-Platine R3	J6-10 29 70	€ 27,95
Mikrocontroller-Onlinekurs	J6-10 20 44	€ 99,–
NEU: Lernpaket Arduino Projects	J6-11 51 22	€ 79,95

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