BASIC für PICs (4)
Serielle Eingabe und LC-Anzeige

Von **Wouter Spruit**
(Niederlande)

Die bisherigen Artikel dieser Serie [1] zeigten, wie man einen PICAXE-Chip programmieren und digitale und analoge Ein- und Ausgänge realisieren kann. In Teil 3 machten wir uns in verschiedenen Anwendungen wie Servosteuerung und Sounds die Pulsweitenmodulation (PWM) zunutze. Dieser Artikel schließt nun die Reihe ab, wir werden ein OLED-Display und eine PS/2-Tastatur an den PICAXE anschließen und alles über eine serielle Leitung mit dem PC verbinden.

Bild 1.
Pinbelegung des PICAXE 18M2.

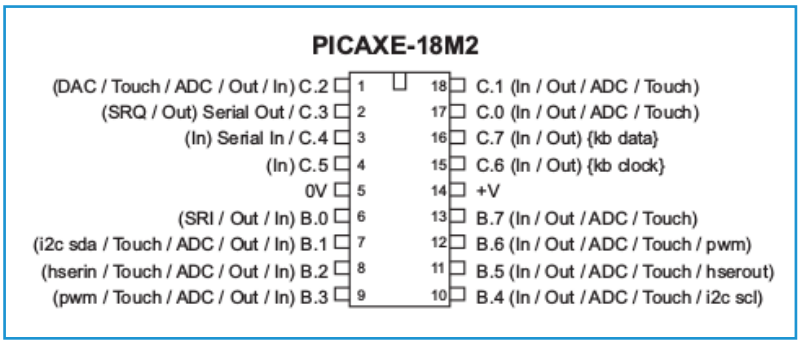
Déjà vu - déjà entendu

Dieses Mal unterscheidet sich der Aufbau des PICAXE deutlich von dem der letzten Folgen. Anstelle des PICAXE 08M2 wird nun ein PICAXE 18M2 verwendet, die Pinbelegung ist in **Bild 1** zu sehen. Aber der Controller wird immer noch von der 5-V-Schiene eines ATX-Netzteils mit Strom versorgt und alles wird auf den gleichen lötfreien Steckboards aufgebaut.

Achten Sie darauf, dass in der Programmier-/Download-Schaltung (Breadboard-Adapter AXE029) die „18er"-Jumper-Einstellung verwendet wird, so dass die Download-Anschlüsse des 18M2 passen. In diesem Artikel wird das USB-Seriell-Kabel (**Bild 2**) für mehr genutzt als nur zur Programmierung des PICAXE – nämlich zur seriellen Kommunikation zwischen PICAXE und einem PC. Der PICAXE ist mit LinAXEpad (Version 1.5.0) für (Arch-) Linux programmiert. Alle Pin-Nummern in den Schaltplänen beziehen sich auf die physikalischen Pin-Nummern des Chips. Pin-Nummern im Code beziehen sich auf interne Pin-Namen nach Bild 1. Beispiele, wie man einen PICAXE-Chip programmiert, finden Sie in den früheren Artikeln [1] oder im PICAXE-Handbuch [2]. PICAXE-Chips und verschiedene Peripheriebauteile sind über den *Revolution Education Webstore* erhältlich [3]. Zur Auswahl von Komponenten für elektronische Schaltungen finden Sie etwas Theorie im zweiten Teil der PICAXE-Artikelreihe.

Serielle Kommunikation

Die meisten Mikrocontroller sind in der Lage, über eine serielle Verbindung zu kommunizieren. Seriell bedeutet, dass die Daten über nur eine Leitung gesendet und als eine Folge von Bits empfangen werden. Ein gemeinsamer Standard für serielle Kommunikation ist beispielsweise RS232 (zum Beispiel die serielle Schnittstelle am PC). Die originalen Spezifikationen von RS232 gehen von Spannungen von ±15 V aus, aber die meisten modernen Mikrocontrollern ignorieren diese in der ursprünglichen Norm spezifizierten Spannungen und nutzen nur das Protokoll für die Codierung und das Timing der seriellen Daten. Die seriellen Befehle des PICAXE kennen separate Modi für eine serielle Kommunikation mit Controller-Spannungspegeln (Baudrate beginnt mit einem „N") und ech-



tes RS232 (Baudrate beginnt mit einem „T“). Serielle Kommunikation mit Geräten wie einem PC oder dem RaspberryPi ermöglicht einer PICAXE-Schaltung, nahtlos in andere Projekte integriert zu werden, mit dem PICAXE als Puffer für alle Ein- und Ausgangssignale. Der folgende Abschnitt erläutert, wie Sie eine serielle Verbindung zwischen den Geräten einrichten, aber Sie können auch zunächst die Beispiele ausprobieren, wenn Sie möchten. Die Theorie ist aber nützlich, wenn sie die Verbindungen zwischen den verschiedenen Teilen in Ihrem eigenen Design konfigurieren wollen.

Der PICAXE kennt zwei Arten von seriellen Input- und Output-Befehlen. Die erste Gruppe wird nur verwendet, wenn die üblicherweise für eine serielle Kommunikation genutzten Pins eingesetzt werden: `Sertxd` für die Ausgabe und `serrxd` für die Eingabe. Der zweite Befehlssatz, `serout` und `serin`, ist gedacht für die serielle Kommunikation über andere Pins, die die serielle Kommunikation unterstützen. Das ist auch der Hauptgrund dafür, dass wir den PICAXE 18M2 anstelle des 08M2 einsetzen, da letzterer über keine Pins für `serin/serout` und `kbdata/kbclock` verfügt. Eine serielle Verbindung wird mit einer Reihe von Parametern eingerichtet, die beschreiben, welche Zeichen der Empfänger zu erwarten hat und wie er sie interpretieren soll. Alle Daten werden in Frames (Rahmen) gesendet. Ein Frame besteht (in dieser Reihenfolge) aus: 1 Start-Bit, 5 bis 9 Bits binärer Daten, manchmal einem Parity-Bit und Stopp-Bit(s). Die Parameter einer seriellen Verbindung informieren über die Anzahl der Frames pro Sekunde (Baudrate), die Anzahl der Datenbits pro Rahmen, ob Paritäts-Bits verwendet werden (`n` für nein) und die Anzahl der Stopp-Bits. Die Eigenschaften der seriellen Verbindung von und zum PICAXE sind auf acht Datenbits, keine Parität und ein Stopp-Bit pro Frame festgelegt. Für die `serout/serin`-Befehle muss die Baudrate angegeben werden, für die Kommunikation über die Download-Pins ist sie auf 4800 Bd festgelegt (außer bei X2-Controllern, da müssen es 9600 Bd sein).

Verbindung mit dem PC

Der Anschluss des Download-Kabels wurde schon in einem früheren Artikel dieser Reihe besprochen, falls Sie es verpasst haben, können Sie sich an den Schaltplänen aus dem

Bild 2.
USB-zu-Seriell-Kabel. Sieht vertraut aus, nicht wahr?

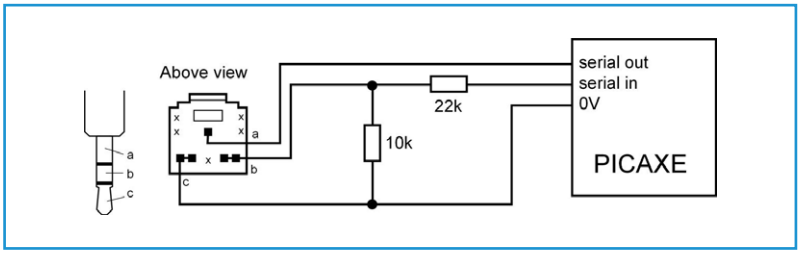


Bild 3.
Runderneuerung: Wie man das Download-Kabel verbindet.

PICAXE-Handbuch in **Bild 3** orientieren. Eine serielle Verbindung zu einem PC ist bereits vorhanden, da sie zum Herunterladen von Programmen auf den PICAXE verwendet wird! Sie können sie natürlich auch verwenden, damit ihre Programme mit dem PC kommunizieren können. Mit dem Befehl `sertxd` senden Sie Daten ohne jegliche Konfiguration zum PC. Zur Anzeige dieser Daten am PC ist ein Terminalprogramm erforderlich. In der LinAXEpad-Software steckt ein solches im Menü PICAXE\Terminal ... (F8). Die in einem PICAXE-Register (zum Beispiel b0) gespeicherten Daten werden als unformatierte binäre Daten gesendet. Diese Daten werden als ASCII-Code [4] interpretiert. Um den Wert in b0 als lesbaren Text, als eine Folge von ASCII-kodierten Zeichen statt als in b0 gespeicherte rohe ASCII-Werte zu senden, verwenden Sie #b0 (das gilt aber bei veralteten PICAXEs nicht immer). Der Code in **Listing 1** weist den PICAXE an, den Wert von einem Register in einer Schleife zu erhöhen, dann den Wert als interpretierte Zahl (das heißt eine lesbare Sequenz von ASCII-Zeichen) und dann als rohe ASCII-Zeichen zu senden. Da nur ein Teil der ASCII-Daten aus lesbarem Text besteht, werden die anderen Zeichen entweder als besondere, nicht druckbare Zeichen oder als Steuersequenzen interpretiert. Die Werte „13“ und „10“ im `sertxd`-Befehl sind rohe ASCII-Daten und werden als Steuersequenzen verwendet, um ein „carriage return“

und „line feed“ zu erzeugen. Dadurch wird der Cursor an den Anfang einer neuen Zeile transportiert.
Der Empfang serieller Daten über die „Download-Pin“-Verbindung erfordert etwas mehr Konfiguration, weil die PICAXE-Firmware diesen Anschluss immer in Betrieb hält und nach neuen Downloads sucht. Um serielle Eingangsdaten zu empfangen, muss der Pin erst von seiner Pflicht entbunden werden. Bis der Pin

Listing 1: PC_OUT

```
main:
b1=0
do
  b1=b1+1
  sertxd(“The value of #b1 is “,#b1,13,10) ‘interpret  number as text
  sertxd(“The value of b1 is “,b1,13,13,10) ‘interpret number as ASCII
  pause 1000
loop
end
```

Listing 2: Loopback

```
init:
disconnect ‘PICAXE no longer scans for program downloads
main:
do
  serrxd [10000,timeout],b0  ‘wait 10 seconds for input, then goto timeout
  sertxd(“character received: “,b0,13,10)
  if b0 = “q” then
    sertxd(“q received, type quit to exit”,13,10)
    serrxd [5000,timeoutmain],(“quit”)
    goto quit
  endif
loop

quit:
reconnect
sertxd(“quit received. program done.” ,13,10)
end

timeout:
reconnect
sertxd(“Input timed out”,13,10)
‘goto main ‘only uncomment this if you know
‘how to reprogram the unconnected device!
sertxd(“Downloading of programs re-enabled.”,13,10)
end

timeoutmain:
sertxd(“no quit received within 5 sec. restarting program.”,13,10)
goto main
```

dann wieder angeschlossen ist (oder ein Soft- oder Hardwarereset durchgeführt wurde), können keine neuen Programme heruntergeladen werden. Ein Timeout für die seriellen Eingangsbefehle ist bei neueren PICAXEs verfügbar. Das Programm in **Listing 2** zeigt, wie die serielle Eingabe über das Download-Kabel freigegeben und gesperrt wird, wie Timeouts verwendet werden und wie man einen Befehl von einem PC-Terminal empfängt. Wir werden jetzt ein Terminal öffnen (auf F8 drücken), um mit dem PICAXE zu kommunizieren.

Keyboard-Eingabe

PS/2-Tastaturen senden Daten über eine serielle Verbindung, aber sie benötigen auch ein Taktsignal. Die Belegung eines PS/2-Tastaturanschlusses ist in **Bild 4** gezeigt. Diverse PICAXEs haben Anschlüsse, die mit „kb data“ und „kb block“ bezeichnet sind (Pins 16 und 15 am 18M2), ideal für die Verbindung mit einer PS/2-Tastatur. Der `kbin`-Befehl wartet auf Daten der Tastatur (mit Zeitlimit), während der `kbled`-Befehl die Steuerung der ScrollLock-, NumLock- und CapsLock-LEDs auf der Tastatur ermöglicht. Beachten Sie, dass der `kbled`-Befehl die Programmausführung stoppt, bis eine Tastatur angeschlossen ist. Keyboard-Daten werden als Scancodes gesendet. Eine Übersicht darüber finden Sie im Kapitel zum `kbin`-Befehl in Teil 2 des PICAXE-Handbuchs [5]. Die meisten Zeichen werden als 8-bit-serielle Daten gesendet.

Jetzt schließen wir die Tastatur gemäß Schaltplan in **Bild 5** an. Da PS/2-Tastaturen nicht mehr up to date sind, könnten sie in Computergeschäften nicht mehr erhältlich sein, aber dafür findet man sie gebraucht im Internet oder auf Flohmärkten umso besser zu einem Spottpreis. Wenn Sie keinen PS/2-Steckverbinder für das Breadboard finden, können Sie auch den Stecker vom Tastaturkabel abknipsen und die vier Drähtchen „einfach so“ anschließen. Das Programm in **Listing 3** empfängt Eingaben von der Tastatur und sendet sie über das Download-Kabel an einen PC. F8 öffnet eine Terminal-Schnittstelle innerhalb (meiner Version) der PICAXE-Programmiersoftware.

OLED-Display hinzugefügt

In diesem Beispiel wird ein „AXE033Y serial/I2C OLED“-Display mit 16x2-alphanumerischen Zeichen an den hserial-Pin des PICAXE angeschlossen. Das AXE033Y-Display besitzt einige zusätzliche Funktionen (sieben Speicherplätze für vorprogrammierte Texte, Funktionen als Stand-alone-Uhr mit optionalem Uhrenmodul), aber jetzt wollen wir es nur als ein einfaches Zeichen-Display einsetzen. Andere serielle Zeichen-Displays dürften deshalb auch gut in diesem Beispiel funktionieren. Daten werden über eine serielle Verbindung zum Display gesendet, wobei wir es wieder mit einer ASCII-Zeichencodierung zu tun haben. Mehrere ASCII-Codes werden auch hier nicht als Zeichen, sondern als Display-Befehle interpretiert. Eine Übersicht der Zeichencodes und Befehle finden Sie im PICAXE-Handbuch Teil 3 [6] auf den Seiten 32 und 41.

Verbinden Sie die serielle Anzeige entsprechend den Schaltungen in **Bild 6**. Vor der Programmierung des PICAXE mit dem relativ komplizierten Display-Code müssen wir sicherstellen, dass das angeschlossene Display auch wie erwartet funktioniert. **Listing 4** zeigt, wie das Display vorzubereiten ist und wie man es anstellt, dass ein „Hello Elektor!“ erscheint. Das Display nutzt Steuersequenzen, um den Cursor zu positionieren. Jedes Mal, wenn ein Zeichen zur Anzeige gesendet wird, rückt der Cursor automatisch weiter. Wenn aber das Display Rückmeldung über die Tastatureingabe geben soll, müssen Sonderzeichen wie „Backspace“ anders codiert werden. Beachten

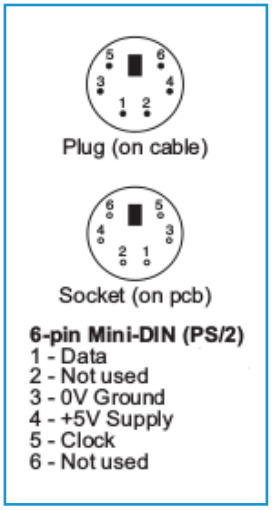


Bild 4.
Pinbelegung eines PS/2-Tastatursteckers.

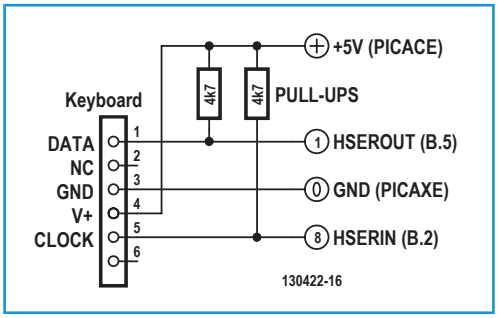


Bild 5.
So wird die Tastatur angeschlossen.

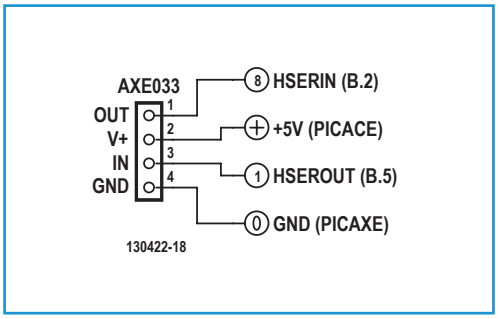


Bild 6.
Anschluss des Displays.

Listing 3: Data from keyboard to PC

```
main:
do
  kbin b1' get keyboard input
  sertxd("Received scancode from PS/2 keyboard: ",#b1,13,10) 'interpret
number as text
loop
end
```

Listing 4: Hello ~~World~~ Elektor!

```
pause 500         'allow screen to initialize first
serout B.5,N2400,(254,1)   'send clear command to screen
pause 30        'wait for the screen
serout B.5,N2400,(254,128)   'move screen cursor to line 1, char 1
serout B.5,N2400,("Hello Elektor!") 'send text to display
end
```

Sie, dass Scancodes und ASCII nicht direkt kompatibel sind! Ein Beispiel, wie man Tastatureingaben konvertiert, um sie auf dem Display darzustellen, ist in **Listing 5** gezeigt. Die Umwandlung muss für alle verwendeten Zeichen/Tasten erfolgen. Die Codes sind nicht sequentiell kompatibel. Schließen Sie wie beschrieben die Anzeige und Tastatur an. Als einzige Tasten sind die Zeichen „a", „b", „c", „h", „i" und „Space" aktiviert; es gibt eine „Backspace"-Funktion und die Tasten „Enter" und „Escape" starten respektive beenden das Programm.
Der Display-Speicher besteht aus zwei Zeilen à 40 Zeichen, aber nur die ersten 16 Zeichen jeder Zeile werden angezeigt. Der übliche Laufeffekt kann durch Verschieben des Sichtfensters relativ zum gespeicherten Text erzielt werden, wie in der Funktion „quit" in Listing 5 gezeigt. Serielle Displays sind mit verschiede-

Listing 5: KEYBOARD_DISPLAY

```
init:
pause 500  'allow screen to initialize first
gosub initscreen
serout B.5,N2400,("Enter text:")
gosub setcursor
gosub showcursor
do
  kbin b1
  if b1 = $1C then 'a
    b2=97   'convert to lower case ASCII a
    gosub printcharacter
  elseif b1 = $32 then 'b
    b2=98
    gosub printcharacter
  elseif b1 = $21 then 'c
```

```
    b2=99
    gosub printcharacter
  elseif b1 = $33 then ‘h
    b2=104
    gosub printcharacter
  elseif b1 = $43 then ‘i
    b2=105
    gosub printcharacter
  elseif b1 =  $29 then ‘space
    b2=32
    gosub printcharacter
  elseif b1 = $66 then ‘backspace
    gosub backspace
  elseif b1= $76 then ‘escape quits the loop
    goto quit:
  elseif b1= $5A then ‘enter resets the program
    goto init:
  endif
loop

quit:
gosub initscreen
gosub hidecursor
serout B.5,N2400,(“It is now safe to turn off the PICAXE”)
do
  serout B.5,N2400,(254,24) ‘ move the window left
  pause 100
loop
end

initscreen:
serout B.5,N2400,(254,1) `clear screen
pause 30
serout B.5,N2400,(254,128) `move to start of first line
symbol CURSOR_POS = b3
CURSOR_POS = 0
return

setcursor:
b4=CURSOR_POS+192   ‘192: start of second line
serout B.5,N2400,(254,b4)
return

showcursor:
serout B.5,N2400,(254,14)  ‘turn on cursor
return

hidecursor:
serout B.5,N2400,(254,12)  ‘turn off cursor
```

```
return

printcharacter:
if CURSOR_POS > 15 then return endif  'only use visible character space
serout B.5,N2400,(b2)
pause 200
CURSOR_POS=CURSOR_POS+1
return

backspace:
if CURSOR_POS = 0 then return endif 'already start of line
CURSOR_POS=CURSOR_POS-1
gosub setcursor 'move back one space
serout B.5,N2400,(32) 'overwrite with space character
gosub setcursor 'set cursor to correct position
pause 100
return
```

nen Funktionen und Optionen ausgestattet. Es würde den Rahmen dieses Artikels sprengen, sollten sie alle hier erläutert werden. Deshalb kann ein Blick in die Betriebsanleitung [7] des AXE033-Displasy nicht schaden, um solche Details zu erforschen.

Terminal-Einstellungen

Um PICAXE vollständig in ein vorhandenes oder neues System wie Home-Automation-Anwendungen zu integrieren, sollte der PC (und der Anwender) eine serielle Verbindung nutzen können, um mit dem PICAXE kommunizieren, ohne sich auf das Terminal-Programm des Programmier-Editors verlassen zu müssen. „Minicom“ [8] ist ein Terminal- Programm, das auf Linux und auch auf anderen POSIX-kompatiblen Systemen wie Mac OSX funktioniert, aber dieses Beispiel erfordert Linux. Um es zu nutzen, müssen Sie sicherstellen, dass das (Modul für das) serielle Download-Kabel verwendet wird. LinAXEPad kann Ihnen helfen, das richtige Modul für das USB-Seriell-Kabel (Ansicht\Optionen\Port\AXE027 modprobe) freizugeben und zeigt die Adresse des Geräts (wahrscheinlich „/dev/ttyUSB0“). Um „minicom“ auf Debian-Systemen (einschließlich Raspbian OS auf dem Raspberry Pi) zu installieren, geben wir `sudo apt-get install minicom` oder bei Arch `sudo pacman -S minicom` ein. Um „minicom“ zu konfigurieren, wählt man mit `sudo minicom -s` zunächst den Setup-Modus. Wählen Sie die Option „Serial Port Setup“ (Pfeiltasten, Enter) und geben Sie „A“ ein, um den Gerätepfad auf den Pfad des USB-Downloadkabels (in unserem Fall „/dev/ttyUSB0“) zu ändern. Tippen Sie dann auf „E“, um die serielle Geschwindigkeit zu ändern und verwenden Sie die Tasten „A“ und „B“, um 4800 Bd zu wählen. Stellen Sie sicher, dass das Programm acht Datenbits, keine Parität und ein Stopp-Bit (8N1) erwartet. Mit „Enter“ kehren Sie zum Hauptmenü zurück. Speichern Sie die Einstellungen im standardmäßig eingestellten „dfl“-Format. Nach Auswahl von „Exit“ (NICHT „Exit“ von minicom) können Sie mit dem PICAXE über „minicom“ kommunizieren. Sie können nun auch mit `sudo minicom` starten. Wenn Sie „minicom“ verlassen möchten, verwenden Sie STRG+A, dann X und schließlich YES (Enter). Einige Beispiele zeigen, wie man die serielle Kommunikation mit PICAXE in Programmen und Skripten auf einem PC nutzen kann. Dafür geht man von einer Bash-Shell unter Linux aus, mit einer geprüften und betriebsbereiten seriellen Verbindung zum PICAXE. Alle Befehle benötigen Root-Access. Zuerst müssen die seriellen Verbindungseigenschaften eingestellt werden. Wir geben `stty -F /dev/ttyUSB0 speed 4800 cs8 -cstopb -parenb` ein. Um Daten aus der Verbindung anzuzeigen, verwenden wir „cat“: `cat /dev/ttyUSB0`, um eine einzelne Zeile in eine Variable zu überführen: `read variable < /dev/ttyUSB0`, um die Daten anzuzeigen: `echo $variable` und

schließlich, um Daten über die Verbindung zu senden: `echo q > /dev/ttyUSB0`. Obwohl der quit-Befehl mit „minicom“ gesendet und empfangen werden kann, scheint er nicht zu funktionieren, wenn er mit Echo im Bash gesendet wird. Ein Bash-Skript, um mit einem PICAXE zu kommunizieren (programmiert mit dem Code aus Listing 2 mit der unkommentierten `goto main`-Zeile im `Timeout`-Unterprogramm) ist in **Listing 6** dargestellt. Ein Ausführen des Skripts erfordert, dass das Executable-Flag gesetzt wird (also `chmod +x scriptname.sh`).

Und nun zu Ihnen...

Nun wissen Sie, wie sich ein PICAXE mit seiner Peripherie über eine serielle Verbindung unterhalten kann! Jetzt ist auch ein Projekt möglich, in dem ein PICAXE Daten auf einem OLED-Display anzeigt und Anwendereingaben von einer PS/2-Tastatur empfängt. Die Zwei-Wege-Kommunikation mit einem PC über das USB-zu-Seriell-Download-Kabel ermöglicht ein einfaches Debugging von PICAXE-Programmen mit der Terminal-Anwendung im PICAXE-Programmier-Editor (oder entsprechender Software). Über eine serielle Verbindung kann der PICAXE zur Steuerung Ihrer eigenen Schnittstellen-Elektronik herangezogen werden, möglicherweise über ein Web-Interface, einen PC oder einen Raspberry Pi. Das PICAXE-System ermöglicht es uns, komplexe Systeme wie Heimautomatisierung und Robotik sehr schnell und zu geringen Kosten zu bauen. Für zusätzliche Ressourcen können Sie auch die Support-Seiten der PICAXE-Serie bei Elektor.LABS [9] durchsehen, wo unter anderem alle Listings zu Ihrem Komfort in einzelnen Dateien verpackt sind. So, hier endet jetzt unsere Artikelreihe. Jetzt sind Sie dran, viel Glück bei Ihren Projekten!

(130422)

Weblinks

[1] „BASIC für PICs” (1, 2 und 3), Elektor. POST Projekte Nr. 8, 16 und 19. www.elektor-magazine.com/extra/post

[2] www.picaxe.com/Getting-Started/PICAXE-Manuals

[3] www.techsupplies.co.uk/PICAXE

[4] www.asciitable.com

[5] www.picaxe.com/docs/picaxe_manual2.pdf

[6] www.picaxe.com/docs/picaxe_manual3.pdf

[7] www.picaxe.com/docs/axe033.pdf

[8] http://linux.die.net/man/1/minicom

[9] www.elektor-labs.com/PICAXE

Listing 6: BASH_SERIAL

```
#!/bin/bash
#run as root.
export DEVICE='/dev/ttyUSB0'  #the serial device to use
#initialize
echo setting up device $DEVICE
stty -F $DEVICE speed 4800 cs8 -cstopb -parenb
echo waiting for input...
read INPUT < $DEVICE
echo received: $INPUT
sleep 1
echo sending a “q” character
echo q > $DEVICE
echo waiting for response...
read INPUT < $DEVICE
sleep 1
echo response: $INPUT
echo done
```