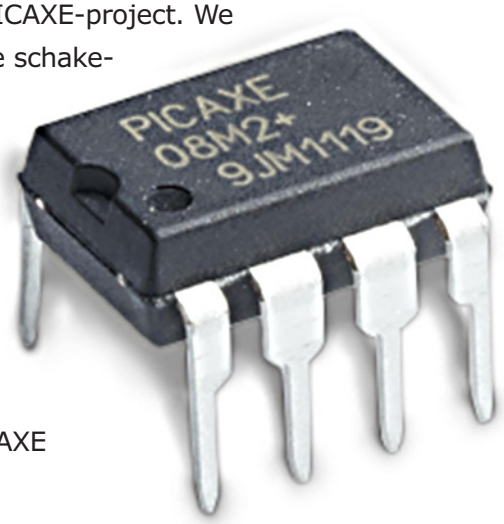
Begraaf de strijdbijl en graaf de AXE op (4)

Tekst weergeven en serieel verzenden

Wouter Spruit
(Nederland)

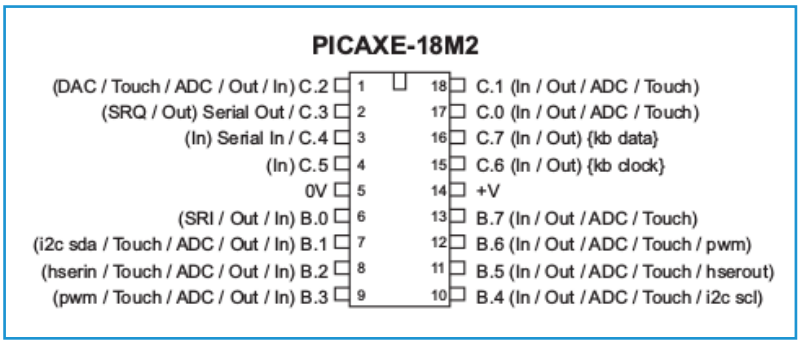
In de vorige artikelen in deze serie [1] hebben we gezien hoe we een PICAXE-chip kunnen programmeren en hoe we interface-schakelingen voor digitale en analoge inputs en outputs kunnen bouwen voor een PICAXE-project. We kunnen de PICAXE nu ook allerlei andere elektronische schakelingen laten besturen en we weten hoe we geschikte elektronische componenten voor onze eigen projecten moeten kiezen. Verder hebben we in deel 3 verschillende toepassingen van pulsbreedtemodulatie beschreven, onder andere servobesturing en het genereren van geluid (hebt u trouwens het liedje herkend?). Met dit artikel sluiten we de serie af. Zoals beloofd gaan we een OLED-display en een PS/2-toetsenbord aansluiten op een PICAXE en we gaan de PICAXE via een seriële verbinding koppelen met een PC.



Figuur 1. PICAXE-18M2-penconfiguratie.

Déjà vu, déjà entendu

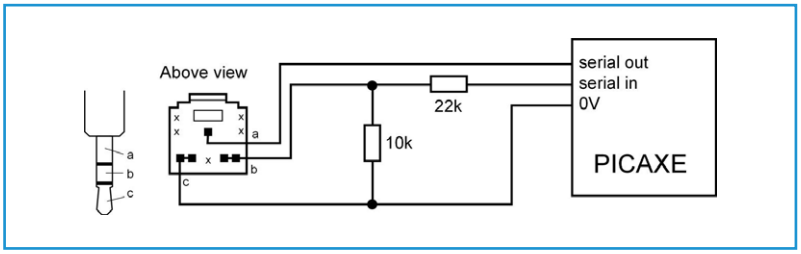
We gaan dit keer uit van een PICAXE-systeem met een andere opbouw dan in de vorige artikelen. We gebruiken nu de PICAXE-18M2 in plaats van de PICAXE-08M2. (De aansluitgegevens zijn te zien in **figuur 1**.) Ook deze chip wordt gevoed uit de 5-V-aansluiting van een ATX-voeding. We bouwen de schakelingen weer op op een breadboard, zonder te solderen. Let er op dat de jumper op de PICAXE-breadboard-adapter (AXE029) in de stand '18' moet staan voor de juiste verbindingen met de download-pennen op 18M2-chips. In dit artikel gebruiken we de USB-naar-serieel-kabel (**figuur 2**) voor meer dan alleen het programmeren van de PICAXE: We zetten hem in als interface voor de seriële communicatie tussen de PICAXE en de PC. De PICAXE wordt geprogrammeerd met de LinAXEpad-software versie 1.5.0 voor (Arch) Linux. Om verwarring te voorkomen: Alle pennummers in de schema's verwijzen naar de fysieke pennummers op de chip en de pennummers in de listings verwijzen naar interne pennamen volgens figuur 1. Zie de vorige artikelen [1], of de PICAXE-handleiding op de website [2], voor voorbeelden van het programmeren van een PICAXE-chip. PICAXE-chips en randapparatuur zijn verkrijgbaar via de webwinkel van Revolution Education [3]. Achtergrondinformatie over het kiezen van componenten voor elektronische schakelingen is te vinden in het tweede artikel van deze serie over de PICAXE [1].



Seriële communicatie

De meeste microcontrollers kunnen communiceren via een seriële verbinding. ‘Serieel’ betekent dat data wordt verzonden en ontvangen als een reeks van bits door één enkele draad. Een gebruikelijke standaard voor seriele communicatie is RS-232 (bijvoorbeeld de seriële poort op een PC). De originele specificaties van RS-232 schreven spanningen van ±15 V voor, maar de meeste moderne microcontrollers houden zich niet aan de gespecificeerde spanningen; ze gebruiken alleen de codering en timing van de data volgens het protocol. Een standaard IC dat ‘echte’ RS232-communicatie, inclusief de spanningen, kan verzorgen, is de MAX232. De seriële commando’s van de PICAXE hebben aparte modes voor gewone seriële communicatie (baudrate begint met een ‘N’), en voor ‘echte’ RS-232 (baudrate begint met een ‘T’).

Figuur 2.
USB-naar-serieel-kabel. Ziet er bekend uit, hè?



Figuur 3.
Opnieuw het aansluiten van de download-kabel.

Dankzij seriële communicatie met apparaten zoals een PC of een Raspberry Pi kan een PICAXE-project naadloos worden geïntegreerd in andere projecten, waarbij de PICAXE kan dienen als buffer voor de aangesloten perifeerieschakelingen. De volgende alinea beschrijft hoe we een seriële verbinding tussen apparaten moeten opzetten. Wie dat te technisch vindt, kan dat gedeelte overslaan en meteen de voorbeelden uitproberen. De theorie is vooral nuttig voor het configureren van verbindingen met zelf ontworpen schakelingen.

De PICAXE kent twee soorten seriële input- en output-commando’s. De eerste groep wordt alleen gebruikt op de pennen die (meestal) verbonden zijn met de seriële kabel: ‘sertxd’ voor output en ‘serrxd’ voor input. De tweede groep, `serout` en `serin`, wordt gebruikt op de andere pennen die seriële communicatie ondersteunen. De belangrijkste reden dat we dit keer de PICAXE-18M2 in plaats van de 08M2 gebruiken, is het ontbreken van de pennen `serin`/`serout` en `kbdata`/`kbclock` op de 08M2.

Een seriële verbinding wordt opgezet met parameters die aangeven wat voor tekens er worden verwacht en hoe die moeten worden geïnterpreteerd. Alle data wordt verzonden in frames. Een frame bestaat uit: 1 startbit, 5 tot 9 bits binaire data, soms een pariteitsbit en één of meer stopbit(s). De parameters van een seriële verbinding vertellen de apparaten het aantal frames per seconde (dat is de ‘baudrate’), het aantal databits per frame, of er pariteitsbits worden gebruikt (‘n’ voor nee/geen) en hoeveel stopbits er zijn. De eigenschappen van de seriële verbinding van en naar de PICAXE liggen vast op 8 databits, geen pariteitsbit en 1 stopbit per frame. De baudrate voor de `serout`/`serin`-commando’s moet worden gespecificeerd. Voor communicatie via de download-pennen ligt de baudrate vast op 4800 (behalve bij X2-chips, daarbij is het 9600).

Verbinden met een PC

Het aansluiten van de download-kabel is al besproken in een eerder deel van deze serie [1], maar wie dat gemist heeft, kan het schema uit de PICAXE-handleiding terugvinden in **figuur 3**. Er is al een seriële verbinding met de PC gemaakt, die wordt gebruikt voor het downloaden van programma’s naar de PICAXE! Deze verbinding kan ook worden gebruikt voor het communiceren met de PC in eigen programma’s. Gebruik het commando `sertxd` om data naar de PC te zenden. Daar is geen extra configuratie voor nodig. Gebruik een terminal-programma op de PC om de data die via deze verbinding verzonden wordt te bekijken. Er zit zo’n programma in LinAXE-pad in het menu PICAXE → Terminal... (F8). Data uit een PICAXE-register, bijvoorbeeld b0, wordt als ruwe binaire data verzonden.

Deze data wordt geïnterpreteerd als ASCII [4]. Gebruik #b0 om de in b0 opgeslagen waarde te verzenden als leesbare tekst (een reeks van ASCII-tekens) in plaats van als ruwe waarde. (Bij ouderwetse PICAXE-chips werkt dit helaas niet altijd.) De code in **listing 1** laat de PICAXE de waarde van een register herhaald ophogen in een lus en die waarde dan eerst als een geïnterpreteerd getal (een leesbare reeks van ASCII-tekens) en daarna

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
```

als de ruwe ASCII-waarde verzenden. Omdat niet alle ASCII-tekens leesbare tekst opleveren, worden de andere tekens geïnterpreteerd als bijzondere of niet-afdrukbare tekens of besturingscommando's. De verzonden waarden '13' en '10' in het `sertxd`-commando zijn ruwe ASCII-waarden die werken als de besturingscommando's 'carriage return' respectievelijk 'line feed'. Samen plaatsen die de cursor aan het begin van een nieuwe regel.

Voor het ontvangen van seriële data via de 'download-pen'-verbinding is meer configuratie nodig, omdat die pen altijd in gebruik is in de PICAXE-firmware (hij wacht altijd op een nieuwe download). Om hem te kunnen gebruiken voor het ontvangen van seriële data moeten we de pen eerst 'loskoppelen'. Er kunnen dan geen nieuwe programma's worden gedownload totdat de pen weer wordt aangekoppeld of totdat er een zachte of harde reset wordt gedaan. Bij de seriële input commando's op nieuwere PICAXE-chips is een timeout-functie beschikbaar. In het programma in **listing 2** zien we hoe we de seriële input via de download-kabel kunnen in- en uitschakelen, hoe timeouts worden gebruikt en hoe we een commando van een PC-verbinding kunnen ontvangen. Open nu een terminal (druk op F8) om te communiceren met de PICAXE.

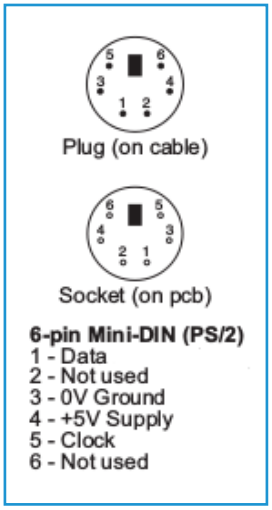
Invoer van een toetsenbord

PS/2-keyboards sturen data over een seriële verbinding, maar ze hebben een kloksignaal nodig. De penconfiguratie van een connector van een PS/2-toetsenbord is te zien in **figuur 4**. Veel PICAXE-chips hebben pennen met de naam 'kb data' en 'kb clock' (pen 16 en 15 op de 18M2), die specifiek bedoeld zijn voor het aansluiten van een PS/2-toetsenbord. Het commando `kbin` wacht op data van een toetsenbord (met timeout) en met het commando `kbled` kunnen we de scroll-, num- en capslock-LED's op het toetsenbord besturen. Let er op dat het commando `kbled` de uitvoering van het programma stopt totdat een toetsenbord is aangesloten. De data van een toetsenbord wordt verzonden in de vorm van scancodes. Een overzicht van de scancodes is te vinden in het hoofdstuk over het commando `kbin` in deel 2 van de PICAXE-handleiding [5]. De meeste tekens worden verzonden als 8-bits seriële data.

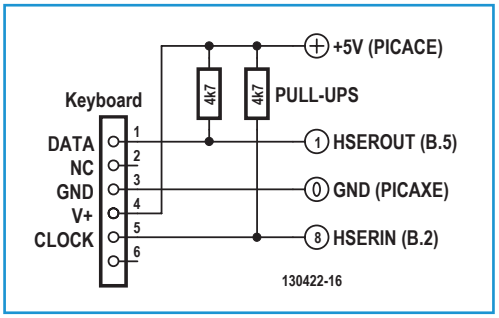
Sluit nu het toetsenbord aan volgens het schema in **figuur 5**. Omdat tegenwoordig meer USB-keyboards dan PS/2-keyboards worden gebruikt, zijn ze bij sommige winkels niet meer verkrijgbaar. Maar bij tweedehands winkels zijn ze nog wel te vinden en meestal zijn ze heel goedkoop. Als er geen PS/2-connector te vinden is, moeten we improviseren, bijvoorbeeld door de PS/2-connector van de toetsenbordkabel af te knippen en de vier aders op de juiste manier aan te sluiten. Het programma in **listing 3** ontvangt input van het toetsenbord en stuurt die via de download-kabel naar een PC. Met F8 openen we een terminal binnen de PICAXE-ontwikkelomgeving.

Een display toevoegen

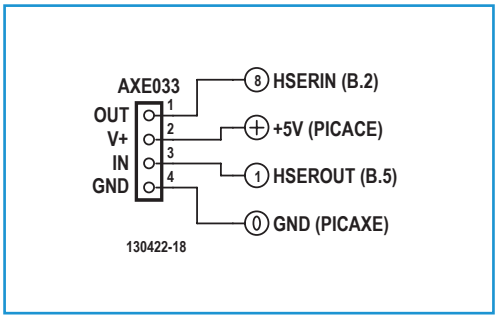
In dit voorbeeld sluiten we een 16x2 alfanumeriek serieel/I²C OLED-display van het type AXE033Y aan op de 'hserial'-pennen van de PICAXE. De AXE033Y beschikt over verschillende extra functies (7 geheugenplaatsen voor voorgeprogrammeerde tekst, functies als standalone-klok met een optionele klokmodule), maar we gebruiken hem nu als een eenvoudig tekstdisplay. Andere seriële tekstdisplays werken waarschijnlijk ook in deze schakeling. De data wordt naar het display verzonden via een seriële verbinding. De tekens worden weer als ASCII-codes



Figuur 4.
Penconfiguratie van een PS/2-toetsenbordconnector.



Figuur 5.
Aansluiten van een toetsenbord.



Figuur 6.
Aansluiten van het display.

Listing 3: Data van het toetsenbord naar de 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
```

verstuurd. Verschillende ASCII-codes worden geïnterpreteerd als besturingscommando's in plaats van weer te geven tekens. Een overzicht van de codes en commando's is te vinden in deel 3 van de PICAXE-handleiding [6] op respectievelijk pagina 32 en 41.

Sluit het seriële display aan volgens het schema in **figuur 6**. Voordat we de PICAXE gaan programmeren met ingewikkelde display-code willen we eerst weten of het aangesloten display wel werkt. In **listing 4** is te zien hoe we het display kunnen initialiseren en de tekst 'Hello Elektor!' weergeven.

Het display werkt met besturingscodes voor het positioneren van de cursor. Telkens als een teken naar het display wordt verzonden, wordt de cursor automatisch opgeschoven. Maar om het display te gebruiken als terugkoppeling

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
```

bij de invoer van een toetsenbord, moeten we bijzondere tekens zoals 'backspace' apart behandelen. Let er op dat scancodes en ASCII niet hetzelfde zijn! In listing 5 zien we een voorbeeld hoe de input van het toetsenbord is om te zetten voor weergave op het display. Alle tekens die we willen gebruiken, moeten worden omgezet. (Ook de volgorde van de codes is niet gelijk aan de volgorde van ASCII-tekens!) Sluit het display en het toetsenbord aan zoals eerder beschreven. De enige toetsen die werken zijn de tekens 'a', 'b', 'c', 'h', 'i' en 'spatie'. Er is ook een 'backspace'-functie en de toetsen 'enter' en 'escape' herstarten en beëindigen het programma.

Het displaygeheugen omvat eigenlijk twee regels van 40 tekens, maar alleen de eerste 16 tekens van elke regel worden weergegeven. Het scroll-effect wordt bereikt door het weergavevenster relatief te verschuiven ten opzichte van de in het geheugen opgeslagen tekst, zoals te zien is in de 'quit'-functie in **listing 5**.

Natuurlijk hebben seriële displays nog allerlei functies en opties, maar er is in dit artikel geen ruimte om die allemaal te beschrijven. Details over het AXE033-display zijn te vinden in de handleiding [7]. Aarzel niet om de mogelijkheden te onderzoeken.

Terminal-instellingen

Om de PICAXE volledig te integreren met een nieuw (of bestaand) systeem (bijvoorbeeld voor domotica toepassingen), moet de PC (en de gebruiker) een seriële verbinding kunnen gebruiken om met de PICAXE te communiceren zonder het terminal-programma van de ontwikkelomgeving. 'minicom' [8] is een terminal-programma dat werkt op Linux en andere POSIX-compatibele systemen (zoals Mac OSX). In dit voorbeeld gaan we uit van Linux. Om het te gebruiken moeten we zeker weten dat de (module voor de) seriële download-kabel actief is. De module voor de USB-naar-serieel-kabel kan met LinAXEpad worden ingeschakeld (kies: view → options → port → AXE027 modprobe). Het adres van de poort (waarschijnlijk '/dev/ttyUSB0') wordt dan weergegeven. Geef `sudo apt-get install minicom` om 'minicom' te installeren op een Debian-systeem (zoals Raspbian OS op de Raspberry Pi). Geef op een Arch-systeem `sudo pacman -S minicom`. We configureren 'minicom' door naar de setup-modus te gaan: `sudo minicom -s`. Kies de optie "Serial port setup" (met de pijltoetsen en enter). Type dan 'A' om het pad van het device te laten wijzen naar de USB-download-kabel (in ons geval: '/dev/ttyUSB0'). Type 'E' om de baudrate te veranderen en gebruik 'A' en 'B' om te kiezen voor 4800 baud. Let er op dat het programma is ingesteld op 8 databits, geen pariteit en één stopbit (8N1). Druk op 'enter' om terug te gaan naar het hoofdmenu en sla de instellingen op als 'dfl' (default). Kies dan 'Exit' (NIET 'Exit from minicom') en de communicatie met

de PICAXE via ‘minicom’ kan beginnen. (Het programma is nu ook te starten met `sudo minicom`). Gebruik CTRL+A, dan X en dan YES (enter) om ‘minicom’ af te sluiten.

Er zijn enkele voorbeelden beschikbaar om het gebruik van seriële communicatie met de PICAXE met programma’s en scripts op een PC te demonstreren. We gaan er bij dit voorbeeld van uit dat we een bash-shell onder Linux gebruiken en dat de seriële verbinding met de PICAXE in orde is. Voor alle commando’s zijn root-rechten vereist. Eerst stellen we de eigenschappen van de seriële verbinding in met: `stty -F /dev/ttyUSB0 speed 4800 cs8 -cstopb -parenb`. We gebruiken ‘cat’ om de data van de verbinding weer te geven: `cat /dev/ttyUSB0`. Om één regel in te lezen in een variabele: `read variable < /dev/ttyUSB0`.

Om de data weer te geven: `echo $variable`. En om data via de verbinding te versturen: `echo q > /dev/ttyUSB0` (Hoewel het quit-commando kan worden verzonden en ontvangen met ‘minicom’, lijkt het niet te werken als het met `echo` vanuit bash verstuurd wordt.) Samenvattend: listing 6 is een bash-script voor het communiceren met een PICAXE, geprogrammeerd met de code uit listing 2, waarbij de regel `goto main` in de subroutine `timeout` actief is gemaakt door de commentaar-tekens te verwijderen. Om het script te draaien moet het uitvoerbaar gemaakt worden, dus geef: `chmod +x scriptnaam.sh`

Aan de slag

Nu weten we hoe we een PICAXE-chip kunnen laten communiceren met periferie via een seriële verbinding! We kunnen nu ook een PICAXE-project een OLED-display laten besturen om data weer te geven en we kunnen input van de gebruiker ontvangen via een PS/2-toetsenbord. Met tweewegcommunicatie met een PC via de USB-naar-serieel-kabel kunnen we PICAXE-programma’s gemakkelijk debuggen met het terminalprogramma in de PICAXE-programmeeromgeving (of equivalente software).

Via een seriële verbinding kan de PICAXE zelfgebouwde elektronica aansturen, eventueel via een web-interface, op een PC of op een Raspberry Pi. Met het PICAXE-systeem kunnen we ingewikkelde systemen, zoals domotica en robotica, heel snel en tegen lage kosten bouwen. De in deze serie getoonde principes zijn, natuurlijk met uitzondering van de code en de functionaliteit van PICAXE-specifieke software, ook van toepassing op andere microcontrollers. Kijk voor verdere ondersteuning ook naar de pagina’s over de PICAXE-serie op

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
```

Elektor.LABS [9], waar onder meer alle listings beschikbaar zijn als afzonderlijke downloads. Tot zo ver deze serie, nu bent u aan zet. Veel succes met uw projecten!

(130422)

Weblinks

[1] ‘Begraaf de strijdbijl en graaf de AXE op’, (1, 2 en 3), Elektor.POST-projecten nr. 8, 16 en 19. www.elektor-magazine.com/nl/extra/post.html

[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