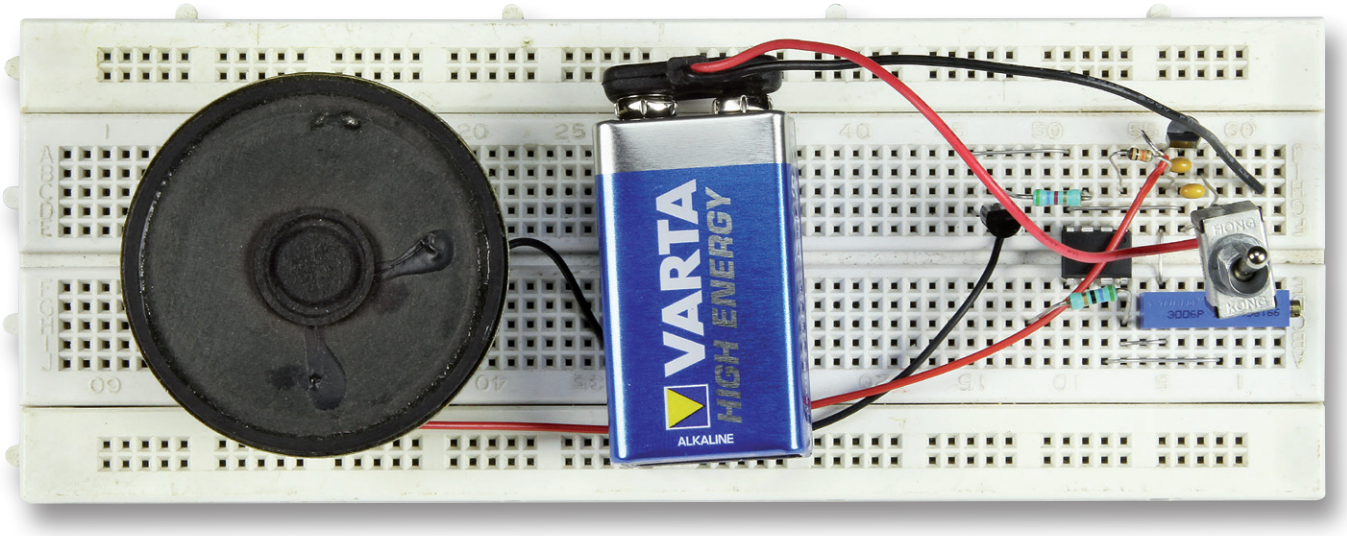
Buhei 2.0

Neuer Lärm mit weniger Aufwand

Von
Friedrich Lischeck
(D)

In der Elektor-Ausgabe vom April 1979 wurde eine Schaltung zum Krachmachen veröffentlicht. Der Autor war zu Recht stolz darauf, dafür nur einen Zähler CD4040, einen Hex-Inverter CD4049 und etwas Kleinkram zu benötigen. Wer damals von so etwas angetan war, kann sich jetzt nochmals freuen, denn dank technischen Fortschritts kriegt man heute das Gejaule mit noch weniger Bauteilen hin!



Was in der Schaltung von **Bild 1** kaum zu übersehen ist: Statt irgendwelcher Zähler und Inverter gibt es heute für die Signalerzeugung einen zeitgemäßen kleinen Mikrocontroller des Typs ATtiny45, der die damalige Funktion eines rückgekoppelten Oszillators übernimmt. Moderne Zeiten haben Folgen: Statt trickreicher Rückkoppelungstechnik gibt es jetzt Software. Zur Hörbarmachung reicht ein einzelner Transistor zur Verstärkung völlig aus. Hinzu kommt noch ein kleiner dynamischer 8-Ω-Lautsprecher. Mit einem Trimmpoti stellt man die Grundfrequenz der Schaltung ein.

Software

Der Gehirnschmalz aktueller Elektronik steckt vielfach weniger in der cleveren Verschaltung der Bauelemente, sondern mehr in der Software für den Mikrocontroller. Dem Programmieren solcher Controller kann man heute selbst bei so einfachen Aufgaben kaum mehr sinnvoll ausweichen. Die für diesen AVR-Controller nötige Software (siehe **Kasten**) wurde mit dem allseits bekannten Bascom-AVR [1] geschrieben. Buhei 1.0 von vor 34 Jahren erzeugte einen anschwellenden Ton, der bei maximaler Frequenz plötzlich wieder tief begann. Die Anstiegsgeschwindigkeit hing mit der eingestellten Frequenz zusammen.
Das gleiche Signal erzeugt nun ein ATtiny45. Mit dem Poti kann man eine Spannung von 0 V bis 5 V am analogen Eingang ADC.1 einstellen. Der ADC des Controllers wandelt diese Spannung dann in korrespondierende Zahlenwerte von 0 bis 1023. Diese Werte dienen der Einstellung der Frequenz „Fg“. Praktisch wird damit Timer0 beeinflusst.
Bei jedem Überlauf löst Timer0 einen Interrupt aus. In der zugehörigen Interrupt-Rou-

tine „Oszillator“ wird zunächst der Timer mit einem neuen Startwert vorbesetzt und der digitale Ausgang B.4 schlicht getoggelt. Damit ergibt sich ein symmetrisches Rechtecksignal. Außerdem inkrementiert die Interrupt-Routine die Variable „Fd“, bis sie einen Wert von 1000 erreicht, worauf sie dann auf 100 zurückgesetzt wird.
Im Hauptprogramm kommt nun zusammen, was zusammen gehört: „Fd“ und der vom Poti abhängige Wert „Fg“ werden addiert. Damit ergibt sich letztlich eine Frequenz zwischen 200 Hz und 4 kHz, die in einem Bereich von 1000 Hz ansteigt und dann wieder mit dem niedrigeren Startwert von neuem zu heulen beginnt.

Startwert

Nun dazu, wie der Startwert des Timers „Preload“ berechnet wird. Es dürfte bekannt sein, dass sich die Zeit „T“ eines 8-bit-Timers wie folgt verhält:

```
T = Tmax – Preload * Tclk
```

Dabei darf der Preload-Wert nur zwischen 0 und 255 liegen. „Tclk“ ist die vom Takt des Controllers abhängige Taktzeit, die gegebenenfalls noch von einem Prescaler (1/8/64/256) beeinflusst wird. „Tmax“ ist die maximal mit dem Timer einstellbare Zeit:

```
Tmax = Prescale * 256 / f
```

Bei „f“ handelt es sich um die Taktfrequenz. Durch Umstellung erhält man also:

```
Preload = (Tmax – T) / Tclk
```

bzw.

```
Preload = (Tmax – 1 / 2 * F) / Tclk
```

Dabei ist F die zu erzeugende Frequenz und T die damit korrespondierende Zeit bzw. deren halbe Periodendauer.

Abschließend

Der Code ist so kurz, dass ein ATtiny45 schon fast zu groß ausfällt. Er war eben gerade zur Hand. Es reicht eigentlich auch das Modell ATtiny25 der Serie mit dem kleinsten Flash-Speicher (2 kB). Zur Kompilierung genügt schon die kostenlose Demo-Variante

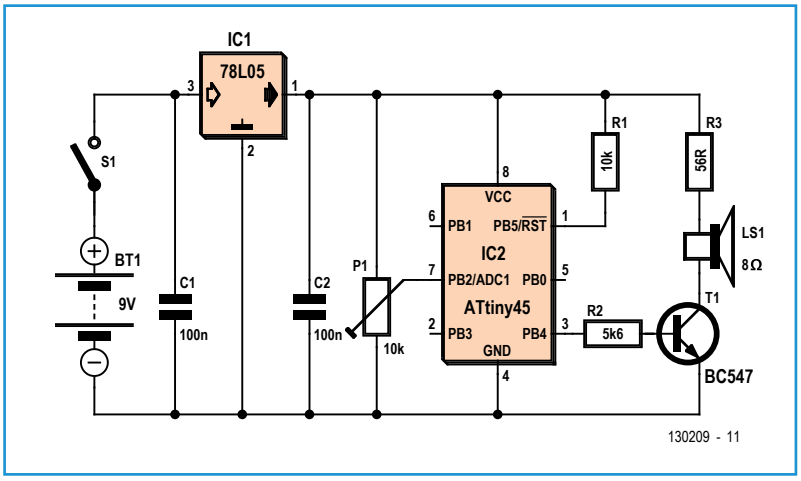
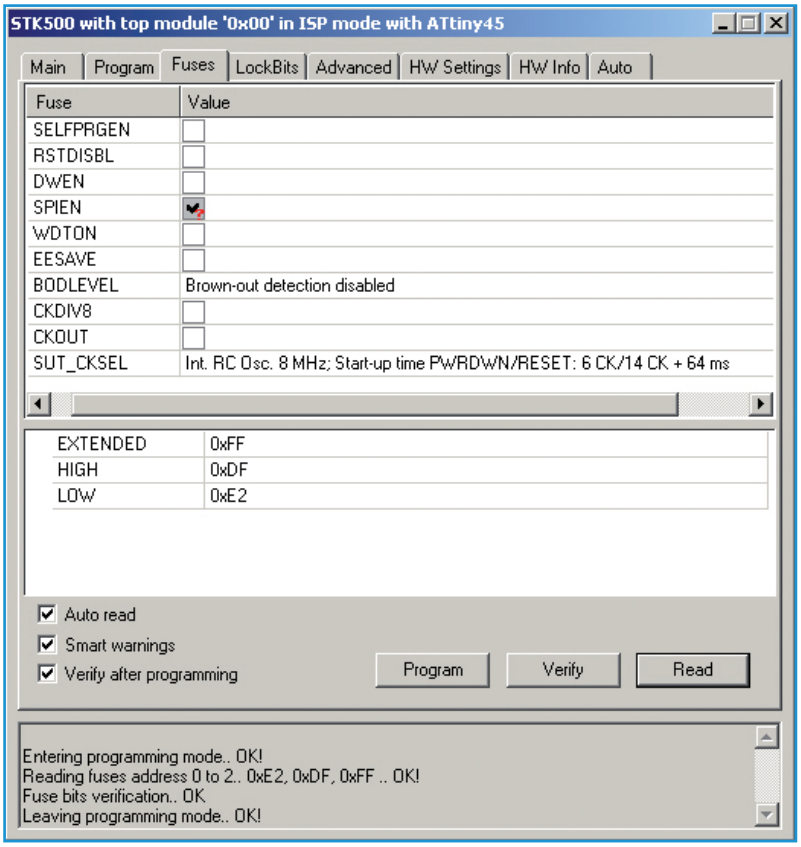
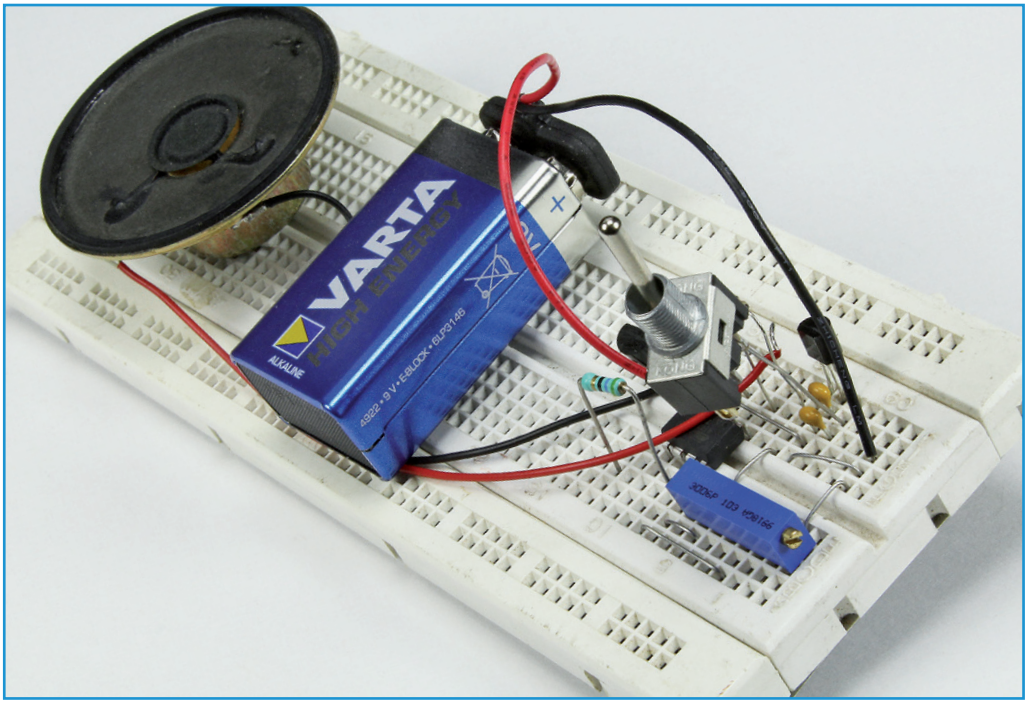


Bild 1.
Die Schaltung von Buhei 2.0 ist noch einfacher als die historische von 1979.

von Bascom-AVR, denn die kann immerhin bis zu 4 kB an Code erzeugen. Wenn man den erstellten Code mit einem selbst gebauten oder käuflich erworbenen Programmer in den Controller brennt, sollte man zwei Dinge beherzigen: Beim Brennen den richtigen Controller-Typ auswählen und die Fuses nicht vergessen: **Bild 2** zeigt einen Screenshot der Fuses des Programms Atmel Studio. Man erkennt, dass hier der Takt-Teiler „CKDIV8“

Bild 2.
So werden die Fuses des Mikrocontrollers ATtiny45 eingestellt.





Stückliste

Widerstände:
(alle ¼W)
R1 = 10 k
R2 = 5k6
R3 = 56 Ω
P1 = 10 k, Trimmpoti

Kondensatoren:
C1, C2 = 100 n/16 V, keramisch

Halbleiter:
T1 = BC547
IC1 = 78L05
IC2 = ATtiny45, programmiert

Außerdem:
DIL-8-Sockel für IC2
Batterie-Clip für 9-V-Batterie
B = 9-V-Batterie
S1 = Schalter, 1 x ein
LS = Lautsprecher, 8 Ω/0,5 W

deaktiviert und der interne 8-MHz-Oszilator mit langer Startzeit ausgewählt ist. Fehler haben in der Regel zu tiefe oder gleich gar keine Töne zur Folge. Source-Code und fertige Hex-Datei können kostenlos von [2] herunter geladen werden.
Zum Aufbau muss man nicht viel sagen, denn die paar Bauteile kriegt man locker auf einem Stückchen Lochrasterplatine unter, wobei sich für IC2 ein IC-Sockel empfiehlt. Zum Schluss sei noch eine Bemerkung von 1979 zitiert: „Ein Gerät dieser Art ist eine ausgezeichnete Hilfe gegen…ja wogegen eigentlich?“

(130209)

Weblink

[1] Bascom-AVR: www.mcselec.com

[2] www.elektor-magazine.de/130209

Der Code

```
‘Buhei Tiny45
$regfile = “attiny45.dat”
$crystal = 8000000

‘Pin B.0 wird Ausgang:
Ddrb = &B00010000
‘Alles aus:
Portb.4 = 0

Dim Preload As Byte
Dim F As Word
```

```
Dim Fg As Word
Dim Fd As Word
Dim Tmax As Single
Dim Tclk As Single
Dim H As Single

On Timer0 Oszillator
Config Timer0 = Timer , Prescale = 256
Enable Timer0
Enable Interrupts
Preload = &H64:                                    '100 Hz
Timer0 = Preload

Config Adc = Single , Prescaler = Auto , Reference = Avcc
Start Adc
'------------------------------------------------------------
'Init
Tclk = 256 / 8000000
Tmax = 256 * Tclk
Fg = 100
Fd = 100
'------------------------------------------------------------
'Hauptprogramm
'F=200-4000
Do
   Fg = Getadc(1)
   H = Fg + 100
   Fg = Fg + H
   Fg = Fg + H
   F = Fg + Fd
   H = 2 * F
   H = 1 / H
   H = Tmax - H
   H = H / Tclk
   Preload = Int(h)
   H = Preload * Tclk
   H = Tmax - H
   H = 0.5 / H
   F = Int(h)
Loop
End
'------------------------------------------------------------
Oszillator:
   Timer0 = Preload
   Portb.4 = Not Portb.4
   Incr Fd
   If Fd > 1000 Then Fd = 100
Return
'------------------------------------------------------------
```