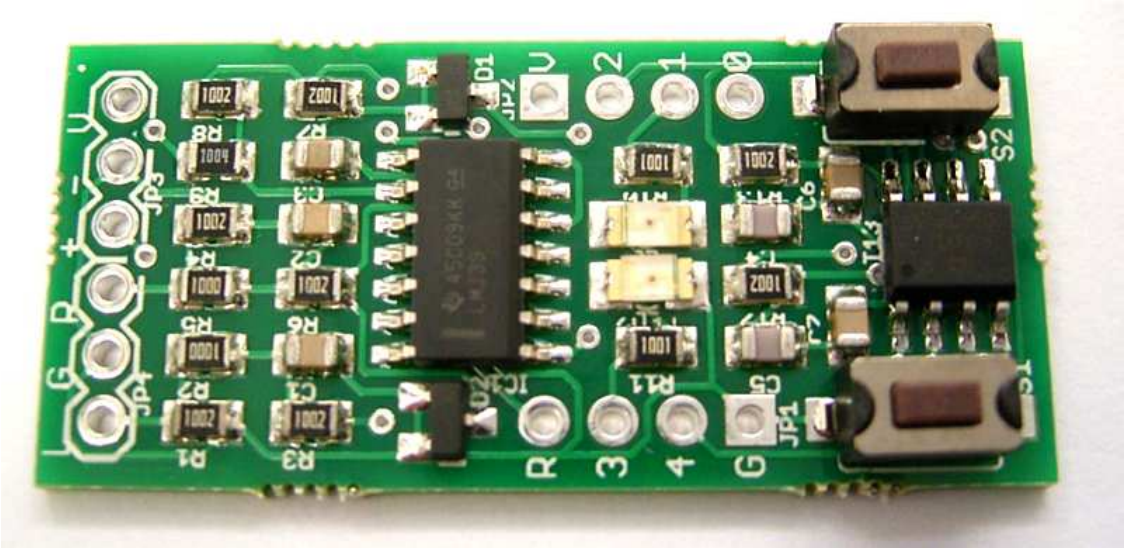
Cheepit Sparrow Bedienungsanleitung



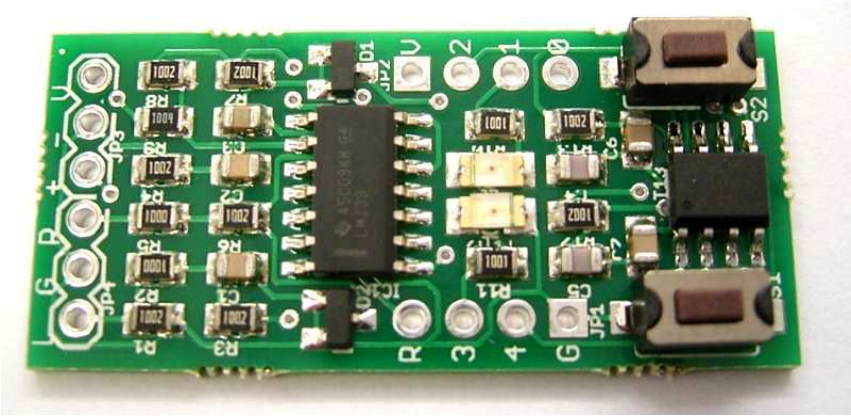
Inhaltsverzeichnis

1 Die ersten Schritte mit dem Sparrow

Cheepit ist ein System zur Programmierung von Mikrocontrollern über den Soundausgang von mobilen Endgeräten. Der linke und der rechte Kanal am Kopfhörerausgang dienen als Daten- und Taktleitung zur ISP-Programmierung. Damit ist es möglich, Mikrocontroller ohne einen PC zu programmieren. Es ist besonders für Einsteiger und für die Ausbildung interessant, weil die gewohnte Hardware wie Smartphones und Tablets verwendet werden können. Kleine Programme (Apps) können direkt aus dem Netz von http://tiny.systems/categorie/cheepit/ in den Controller geladen werden. Selbst entwickelte Software lässt sich problemlos mit andern Nutzern teilen.

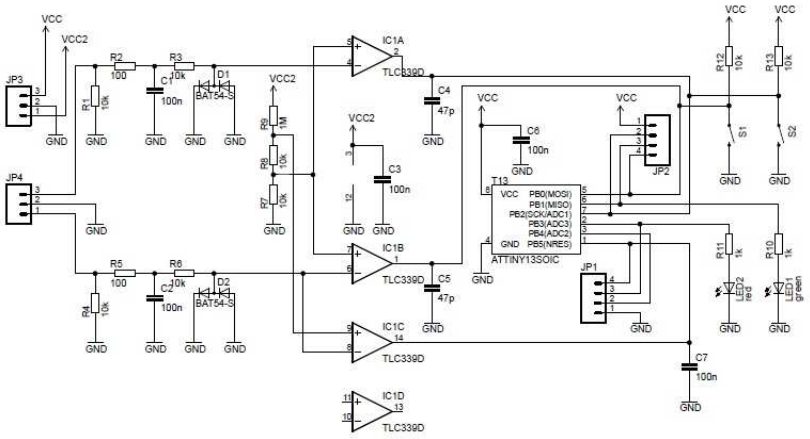
1.1 Die Hardware

Anders als sonst üblich braucht man kein Programmiergerät mehr sondern nur noch ein mobiles Gerät mit einem Webbrowser. Sogar die Programmierung über MP3-Files oder über Klingeltöne ist möglich. Das System eignet sich daher auch für solche Anwendungen, bei denen Programme mobil nachgeladen werden müssen.



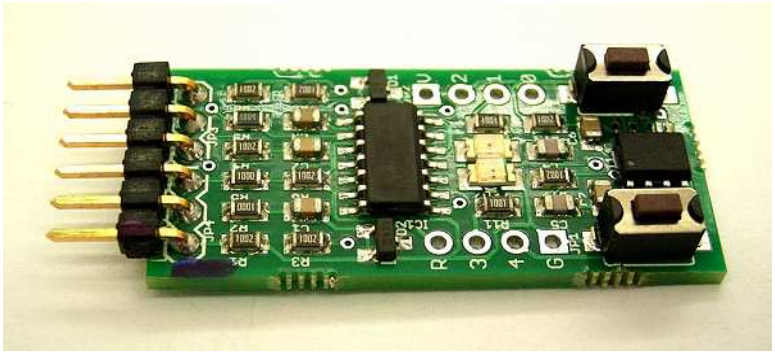
www.ak-modul-bus.de/stat/entwicklungssystem_sparrow.html

Der Sparrow verwendet einen Controller aus der AVR-Familie, den ATtiny13a mit 1 K Flash und acht Anschlüssen. Auf der Platine befinden sich neben dem Programmierinterface zwei LEDs und zwei Tastschalter. Damit wird dieses kleine System zu einem einfachen und vielseitig einsetzbaren Gerät.



Die Schaltung zeigt rechts das eigentliche Mikrocontroller-System mit dem ATtiny13a und den Bedienelementen. Links befindet sich das Programmierinterface. Ein Vierfach-Komparator LM339- Filter und Begrenzer sorgen dafür, dass die Programmierung in einem weiten Bereich möglicher Eingangspegel funktioniert. Ein Programmiersignal versetzt den Mikrocontroller in den Reset-Zustand und steuert die Leitungen MOSI und SCK an. Die erfolgreiche Programmierung erkennt man an einem Signal an MISO und damit am Flackern der LED1 (grün).

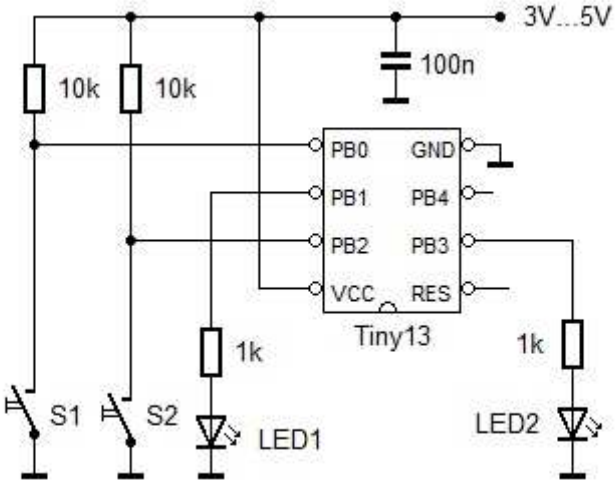
Für die ersten Versuche ohne zusätzliche externe Hardware ist es sinnvoll einen sechspoligen gewinkelten Pfostenstecker einzulöten. Über diesen wird dann die Betriebsspannung zugeführt und das Audiosignal angeschlossen. Die unteren drei Anschlüsse (JP4: L GND R) werden wahlweise an einen Klinkenstecker oder an eine Klinkenbuchse angeschlossen. Der linke und der rechte Kanal dürfen nicht vertauscht werden. An einem Klinkenstecker ist der erste Kontakt (Spitze) der linke Kanal.



Die Betriebsspannung kann im einfachsten Fall von einem Batteriefach für zwei Mignonzellen (AAA) mit zusammen 3 V kommen. Die oberen drei Anschlüsse dienen der Versorgung des Mikrocontrollers (VCC, Anschluss V an Jp3) und des Programmierinterfaces (VCC2, Anschluss + an JP3) mit dem gemeinsamen SNG-Anschluss (G an PJP3). Ein Teilesatz mit allen erforderlichen Steckern und zusätzlichen Bauteilen ist in Vorbereitung.

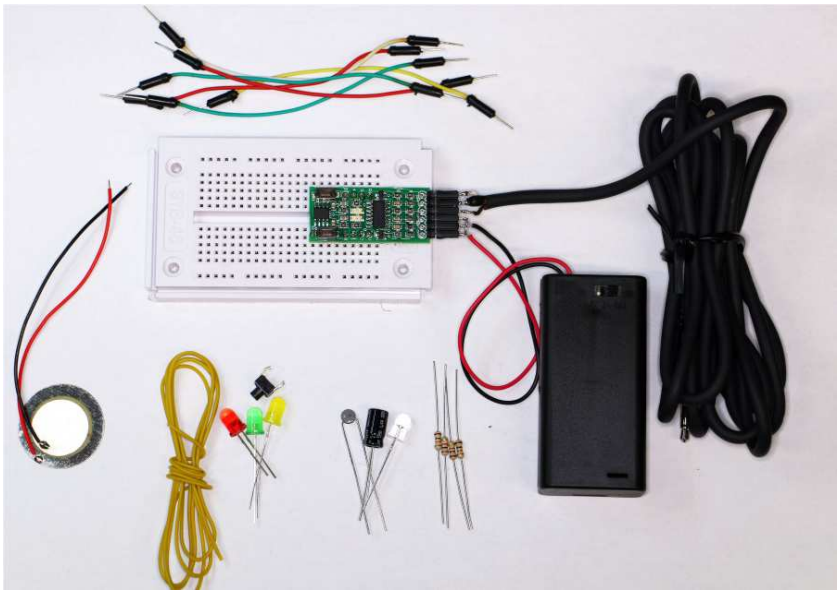
Es existiert bereits eine große Sammlung fertiger Apps (http://tiny.systems/categorie/cheepit/). Die meisten dieser kleinen Programme wurden in Bascom entwickelt, einige auch in Assembler und in C. Anklicken und Übertragen reicht aus, aber wer möchte kann sich auch die Quelltexte ansehen. Es gibt jeweils eine helles und eine dunkle Cheepit-Schaltfläche. Hält man den Mauszeiger länger darauf erscheint ein Hinweis zur Invertierung. Bur die dunkle Schaltfläche startet die invertierte Übertragung und passt zum Sparrow. Die nicht-invertierte Übertragung wird für alternative Schaltungen benötigt, die im Kapitel 10 vorgestellt werden.

Für die Entwicklung eigener Software muss man nur die Grundschaltung des eigentlichen Mikrocontrollers kennen. Mit zwei Tastern und zwei LEDs lassen sich zahlreiche Aufgaben lösen, von Modellversuchen zur digitalen Elektronik bis hin zu praktisch einsetzbaren Geräten für Heim und Hobby. Vielfach kann der Sparrow auch als zentrale Steuereinheit für weitere Elektronik-Projekte dienen. B1 und B3 sind die bevorzugten Ausgänge, deren Zustand dann zugleich über die LEDs angezeigt wird. B4 kann als analoger Eingang ADC2 verwendet werden. Alle fünf Ports lassen sich wahlweise als Eingänge oder als Ausgänge nutzen.



1.2 Komplettpaket SPARROW

Wer nicht löten möchte und lieber alle Bauteile in einem Set haben möchte kann das Komplettpaket bestellen. Alle Stiftleisten sind schon eingelötet und die Anschlusskabel (Audio und Batteriefach) sind vorbereitet. Da kann man gleich loslegen. Das geschlossene Batteriefach hat übrigens eingebauten Schiebeschalter, sodass man alles aufgebaut lassen kann. Die wichtigsten Bauteile für eigene Experimente liegen bei.



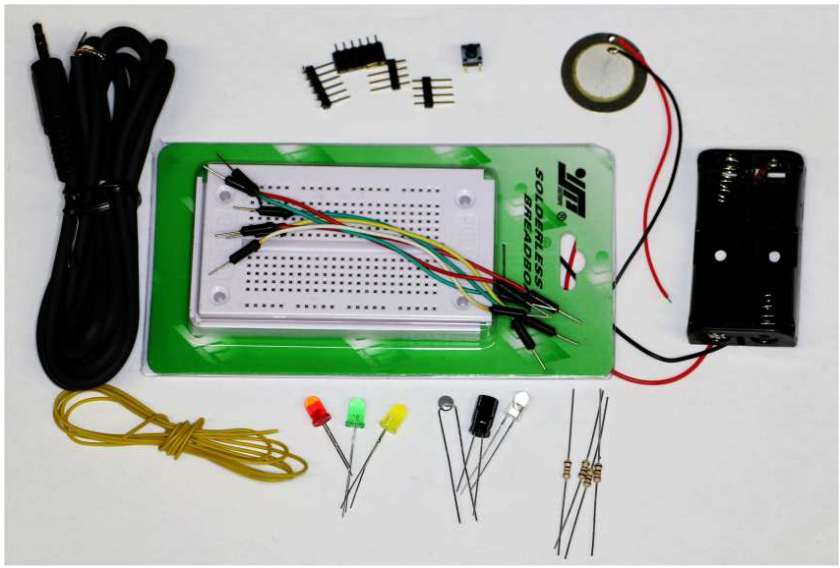
www.ak-modul-bus.de/stat/komplettpaket_sparrow.html

Lieferumfang

* Entwicklungssystem Sparrow mit angelöteten Stiftleisten
* 6pol. Buchsenleiste mit angelötetem Klinkenkabel
* geschlossener Batteriehalter für 2 AA-Mignon-Zellen mit Schalter
* Steckboard mit 270 Steckkontakten
* 3 LEDs rot, grün, gelb 5mm diffus

* Piezosummer
* NTC 10 kΩ
* Fototransistor ELPT333-3C
* 4 Widerstände
* 1 Elko 100µF
* 1 Printtaster
* 6 Steckboard Verbindungskabel
* 1m Schaltdraht

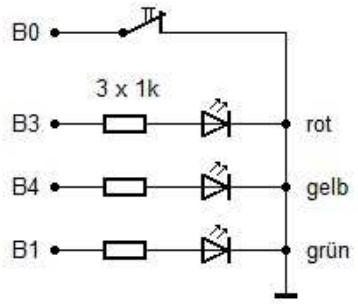
Wer schon einen Sparrow besitzt kann auch ein Bauteileset einzeln bestellen, diesmal mit einem konventionellen Batteriefach. Ansonsten liegen die gleichen Bauteile bei, die so zusammengestellt wurden, dass einige grundlegende Versuche damit durchgeführt werden können.



www.ak-modul-bus.de/stat/bauteileset_fuer_den_sparrow.html

1.3 Schaltungen und Bauvorschläge

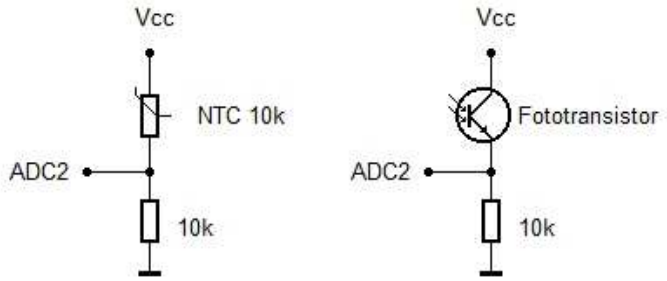
Hier folgen einige Vorschläge, was man mit dem Material tun kann. Eine beliebte Übungsaufgabe in der Schule ist die Entwicklung einer Ampelsteuerung. Die drei Ampel-LEDs werden über Vorwiderstände an die Ports angeschlossen, wobei die rote und grüne LED auf dem Sparrow mit leuchten.



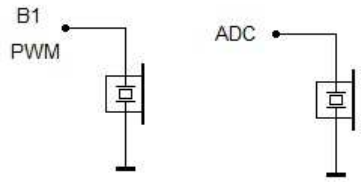
Der Anforderungskontakt liegt an B0 und damit parallel zu S1 des Sparrow. Man hat also die realistische Situation, dass der Steuercomputer seine eigenen Anzeige und Bedienelemente hat. In einem Schaltkasten an der Straßenkreuzung findet man das auch. Der Techniker kann bei Wartungsarbeiten auf S2 drücken,

um die Anlage in einen Notbetrieb mit blinkender gelber Lampe umzuschalten. Auch das wäre eine lohnende Erweiterung des Programms.

Mit Sensoren öffnen sich unbegrenzte Möglichkeiten. Der vorhandene NTC-Temperatursensor und der Fototransistor als Lichtsensor werden in gleicher Weise in einem Spannungsteiler verwendet. Der Eingang B4 (ADC) ist noch frei und eignet sich als Sensoreingang. Eine steigende Temperatur bzw. steigende Helligkeit wird dann an einer steigenden Spannung erkannt. In Ausnahmefällen kann auch B3 als Sensoreingang verwendet werden, wobei die interne rote LED den Sensorstrom mit anzeigt.

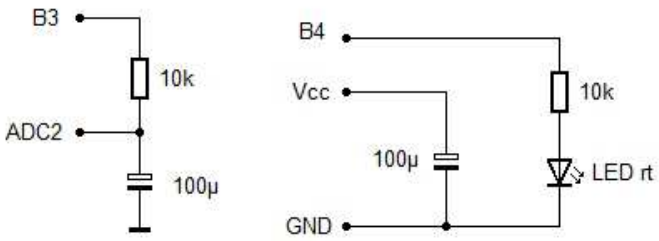


Der Piezoschallwandler kann wahlweise als Lautsprecher oder als Sensor verwendet werden. Zur Soundausgabe eignet sich der Port B2, der auch als PWM-Ausgang dient. Eine fertige App dafür ist das akustische Voltmeter (http://tiny.systems/categorie/cheepit/Voltmeter.html).



Für die Verwendung als Bewegungs- oder Erschütterungssensor oder als Mikrofon muss der Piezowandler an einen AD-Eingang angeschlossen werden. Dies kann der offene Eingang ADC2 (B4) sein oder der Eingang B3 mit der internen roten LED. Die LED kann helfen eine mittlere Spannung einzustellen, wenn man z.B. den Pullup einschaltet und damit einen kleinen Strom vorgibt. In diesem Fall hätte der Sensor eine erhöhte untere Grenzfrequenz, währen ADC2 auch langesame Änderungen zeigen könnte.

Auch mit dem Elko von 100 µF lassen sich ganz unterschiedliche Experimente durchführen. Er kann zum Beispiel in einem Modell eines Regelkreises verwendet werden. Ein Programm soll B3 so steuern, dass möglichst genau die Spannung U am Kondensator eingestellt wird, wobei U vielleicht mit den Tasten eingestellt werden kann.



Eine ganz andere Anwendung ist die Versorgung des Sparrow aus dem Elko und die Entwicklung extrem stromsparender Anwendungen. Ein Beispiel dafür ist der "ewige Blinker" (http://tiny.systems/categorie/cheepit/EwigerBlinker.html), den man aber noch verbessern könnte, wenn eine LED mit größerem Vorwidertand an B4 verwendet würde. Das könnte ein interessanter Wettbewerb in einer Lerngruppe werden. Wer betreibt den Sparrow am sparsamsten und erreicht die längste Betriebsdauer mit einer Kondensatorladung.

2. Einfache Apps, Licht und Sound

Die folgenden einfachen Anwendungen kommen meist noch ohne externe Hardware aus. Die Programme wurden mit Bascom entwickelt, die Quelltexte werden jeweils mit abgedruckt. So können Sie leicht den Überblick gewinnen und sich zu eigenen Anwendungen inspirieren lassen. Bascom kann unter http://www.mcselec.com/ als Testversion bis 4 KB kostenlos geladen werden und eignet sich besonders für den einfachen Einstieg in die Mikrocontroller-Programmierung.

2.1 Einstellbarer Gegentakt-Blinker

Die beiden LEDs blinken im Gegentakt. Die Blinkgeschwindigkeit ist über die Tasten einstellbar. Mit S1 wird das Blinken schneller, mit S2 langsamer. Damit sind alle vier Bedienelemente getestet.

http://tiny.systems/categorie/cheepit/EinstellbarerGegentaktblinker.html

```
'Sparrow_blink2.bas
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
Config Portb = &B000011010
Dim T As Byte
Dim N As Byte

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Led1 = 1
Led2 = 0
T = 20
Do
  N = 0
  Do
    If S1 = 0 Then T = T + 1
    If T > 250 Then T = 250
    If S2 = 0 Then T = T - 1
    If T < 1 Then T = 1
    Waitms 10
    N = N + 10
  Loop Until N >= T
  Toggle Led1
  Toggle Led2
Loop

End
```

2.2 Einstellbare LED-Helligkeit

Die grüne LED1 kann über den PWM-Ausgang PWM0B des Tiny13 gesteuert werden. Das folgende Testprogramm erlaubt die Einstellung der LED-Helligkeit über die Tasten.

http://tiny.systems/categorie/cheepit/EinstellbareLEDHelligkeit.html

```
'Sparrow_PWM.bas
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
Config Portb.1 = 1
Dim D As Byte

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

D = 50
Do
  If S1 = 0 Then D = D + 1
  If D > 254 Then D = 254
  If S2 = 0 Then D = D - 1
  If D < 1 Then D = 1
  Waitms 10
  Pwm0b = D
Loop
End
```

2.3 Elektrofeld-Sensor

Wenn man die rote LED2 nicht verwendet und die Ports B3 und B4 hochohmig lässt hat man an ADC3 einen extrem empfindlichen analogen Eingang. Es reicht ein 5 cm langer Draht an B3 als Antenne, um elektrische Felder zu messen und über den PWM-Ausgang anzuzeigen. Mit dem Spatz in der Hand läuft man über einen Teppich und sieht direkt die elektrische Aufladung bei jedem Schritt an der Helligkeit der grünen LED.

http://tiny.systems/categorie/cheepit/ElektrofeldSensor.html

```
'Sparrow_ADC.bas

$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Dim D As Integer
Ddrb = 2

Config Adc = Single , Prescaler = Auto
Start Adc
Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

Do
  D = Getadc(3)
```

```
  D = D / 4
  Pwm0b = D
  Waitms 18
Loop
End
```

Das Programm enthält eine Verzögerung von 18 Millisekunden, also nahe an der Periodendauer von 20 ms des 50-Hz-Netzes. Alle 50-Hz-Signale werden daher verlangsamt und gut sichtbar an der grünen LED angezeigt. Nähert man sich der Antenne mit der Hand steigt die Amplitude der Wechselspannung deutlich sichtbar an. Sogar Netzleitungen in der Wand kann der Sparrow aufspüren!

Dieses Programm ist fast so nützlich wie ein kleines Oszilloskop. Auch zur Spannungsmessung kann es eingesetzt werden, solange die Messspannung nicht größer als die Betriebsspannung ist. Zur Sicherheit kann man einen Schutzwiderstand von 10 kΩ in die Messleitung legen. Die meisten Messgeräte mitteln die Messung und unterdrücken damit eventuelle überlagerte Wechselspannungen. Der Spatz aber zeigt sie an.

2.4 RS-Schalter

Ein RS-Flipflop hat zwei Eingänge, einen zum Einschalten (Set) und einen zum Ausschalten (RS). Genauso funktionierte diese App. S2 schaltet beide LEDs ein und S1 schaltet sie aus.

http://tiny.systems/categorie/cheepit/RSSchalter.html

```
'Sparrow_RS.bas on/off LEDs
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
Config Portb = &B000011010
Dim T As Byte
Dim N As Byte

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Led1 = 1
Led2 = 1
T = 20
Do
 If S1 = 0 Then
   Led1 = 0
   Led2 = 0
 End If
 If S2 = 0 Then
   Led1 = 1
   Led2 = 1
 End If
Loop
End
```

2.5 Toggle-Schalter

Die Funktion dieser App entspricht der zweier Toggle-Flipflops. Einmal drücken = An, nochmal drücken = Aus. Und das getrennt für beide Taster, die jeweils ihre eigene LED schalten.

http://tiny.systems/categorie/cheepit/ToggleSchalter.html

```
'Sparrow_Toggle.bas on/off LEDs
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
Config Portb = &B000011010
Dim T As Byte
Dim N As Byte

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Led1 = 1
Led2 = 1
T = 20
Do
 If S1 = 0 Then
   Toggle Led1
   Waitms 50
   Do
   Loop Until S1 = 1
   Waitms 50
 End If
 If S2 = 0 Then
    Toggle Led2
   Waitms 50
   Do
   Waitms 50
   Loop Until S2 = 1
 End If
Loop
End
```

2.6 Entspannungslicht

Die grüne LED wird über den PWM-Ausgang wellenartig langsam gedimmt und wieder aufgeblendet. Die Geschwindigkeit ist über die Taster einstellbar. Da das Auge kein lineares Empfinden für die Helligkeit besitzt wird hier durch Quadrieren einer Laufvariablen eine Parabelfunktion realisiert.

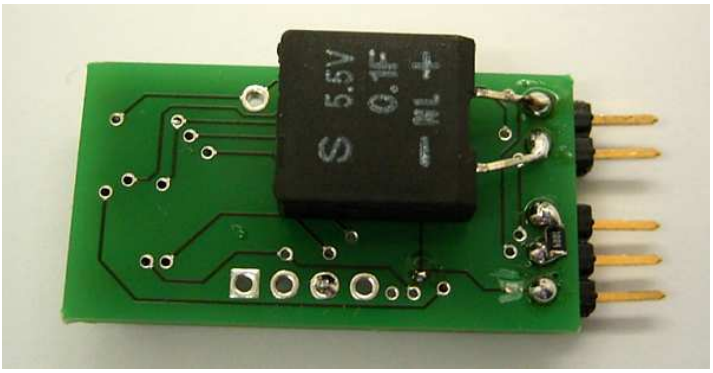
http://tiny.systems/categorie/cheepit/Entspannungslicht.html

```
'Sparrow_Fade.bas
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
```

```
Config Portb.1 = 1
Config Portb.3 = 1
Config Portb.4 = 1
Dim T As Byte
Dim I As Byte
Dim J As Byte
Dim N As Byte
Dim D As Word


Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

N = 50
T = 5
J = 0
Do
  Waitms 10
  I = I + 1
  If I >= T Then
    'Toggle Led2
    If S1 = 0 Then T = T + 1
    If T > 20 Then T = 20
    If S2 = 0 Then T = T - 1
    If T < 1 Then T = 1
    I = 0
    If J = 0 Then N = N + 1
    If N > 250 Then J = 1
    If J = 1 Then N = N - 1
    If N < 50 Then J = 0
    D = N * N
    D = High(d)
    Pwm0b = D
  End If
Loop
End
```

2.7 Der ewige Blinker

Den ewigen Blinker gibt es schon ewig. Mal als reine Elektronik-Schaltung und mal als Mikrocontroller-Anwendung "Der ELO-Flasher". Nun ist er auch auf den Sparrow gekommen. Das Ziel ist extremes Stromsparen. Der Sparrow erzeugt ein regelmäßiges Blitzen an der grünen LED, legt sich aber in der Zwischenzeit schlafen und braucht dann fast keinen Strom. Dass er wieder aufwacht, dafür sorgt der Watchdog-Timer des Tiny13. Der Sparrow wurde hier über einen Speicherkondensator 0,1 F, 5,5 V betrieben. Einmal auf 5 V aufgeladen blinkt er ein bis zwei Stunden lang vor sich hin. Der Spatz ist mit Stromversorgung so klein, dass man ihn sich ans T-Shirt heften kann.

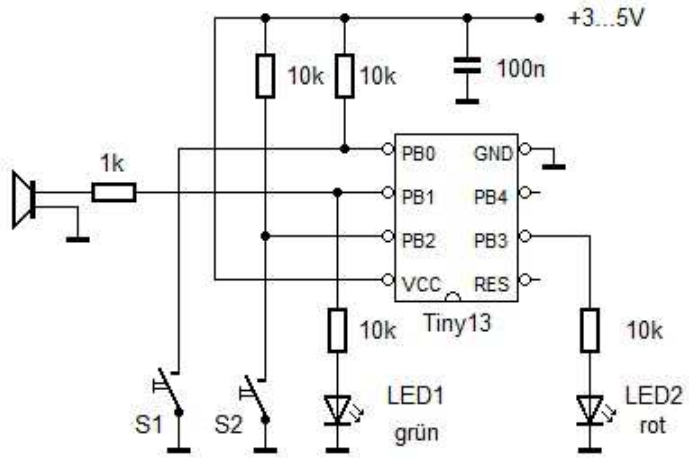


http://tiny.systems/categorie/cheepit/EwigerBlinker.html

```
'ATtiny13 Sparrow Lowpower Flash - Ewiger Blinker
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
Config Portb = 2

Led1 Alias Portb.1
Led2 Alias Portb.3

Config Watchdog = 1024
Start Watchdog

Led1 = 1
Waitms 10
Led1 = 0
Powerdown
End
```

2.8 Das akustische Voltmeter

Wenn man genau aufpasst wohin man die Messkabel hält fällt es manchmal schwer, gleichzeitig auf das Voltmeter zu sehen. Dann braucht man das akustische Voltmeter. An B1 wird ein Piezo-Schallwandler oder ein dynamischer Lautsprecher mit Vorwiderstand angeschlossen.



Die gemessene Spannung wird in eine Tonhöhe umgesetzt. Und null Volt ist Stille, damit es nicht dauernd nervt. Der Messeingang ist ADC3 (B3), und der Soundausgang ist B1. Die Signale werden über PWM0B erzeugt. Der Timer wird mit festem Vorteiler betrieben. Aber die Tonhöhe wird über das Osccal-Register verändert, d.h. der interne RC-Oszillator des Tiny13 wird abgestimmt und damit die gesamte Taktrate.
Am Eingang empfiehlt sich ein Schutzwiderstand mit 10 k gegen Überspannungen und ein Widerstand mit ca. 100 k gegen GND, damit offene Messkabel keine Spannung haben.

http://tiny.systems/categorie/cheepit/Voltmeter.html

```
'Sparrow_ADCsound.bas
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Dim D As Integer
Ddrb = 2                                                    'Sound Output
B1

Config Adc = Single , Prescaler = Auto
Start Adc
Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

Do
  D = Getadc(3)                                             'Voltage Input
B3
  D = D / 10
  Osccal = D
  If D < 1 Then Pwm0b = 0 Else Pwm0b = 40
  Waitms 18
Loop
End
```

2.9 Stimmungsindikator

Zwischenmenschliche Beziehungen basieren auf eindeutigen Signalen. Aber nicht immer ist es leicht, die richtigen Worte zu finden. Der Sparrow kann da helfen. Er fungiert als eine Art Stimmungs-Ampel und macht es unnötig, dass Sätze wie "Quatsch mich nicht an!" ausgesprochen werden. Nach dem Start blitzt

der Sparrow nur so vor sich hin und signalisiert seine Bereitschaft. Der Benutzer wählt dann die momentane Stress-oder Entspannungsstufe mit den beiden Tastern (up/down). Was das dann jeweils genau bedeutet, muss man natürlich vereinbaren. Es könnte zum Beispiel so aussehen:

Modus 1: Doppelblitze rot/grün: Maximaler Stress, am besten nicht ansprechen
Modus 2: Einzelblitze rot/grün: Stark belastet, Kopf ist nicht frei
Modus 3: Doppelblitze rot: Versuch der Entspannung läuft
Modus 4: Einzelblitze rot: Gut entspannt, alles easy
Modus 5: Einzelblitze grün: Bereit zu neuen Taten
Modus 6: Doppelblitze grün: Let's go!

http://tiny.systems/categorie/cheepit/Stimmungsindikator.html

```
'ATtiny13 Sparrow Blink3.bas
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
Config Portb = &B000011010
Dim T As Byte
Dim N As Byte
Dim M As Byte
Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Do
  If S2 = 0 Then
    Waitms 50
    If M < 6 Then M = M + 1
    Do
    Loop Until S2 = 1
    Waitms 50
  End If
  If S1 = 0 Then
    Waitms 50
    If M > 1 Then M = M - 1
    Do
    Loop Until S1 = 1
    Waitms 50
  End If

  If M = 0 Then                                          'Standby
    Led1 = 1
    Waitms 10
    Led1 = 0
    Waitms 500
  End If

  If M = 1 Then                                          'very busy
    Led1 = 1 : Led2 = 1
    Waitms 30
    Led1 = 0 : Led2 = 0
    Waitms 100
    Led1 = 1 : Led2 = 1
    Waitms 30
    Led1 = 0 : Led2 = 0
     Waitms 840
  End If
```

```
  If M = 2 Then                                                 'busy
    Led1 = 1 : Led2 = 1
    Waitms 30
    Led1 = 0 : Led2 = 0
    Waitms 970
  End If

  If M = 3 Then                                                 'relaxing
    Led2 = 1
    Waitms 30
    Led2 = 0
    Waitms 100
    Led2 = 1
    Waitms 30
    Led2 = 0
    Waitms 840
  End If

  If M = 4 Then                                                 'relexed
    Led2 = 1
    Waitms 30
    Led2 = 0
    Waitms 970
  End If

  If M = 5 Then                                                 'ready
    Led1 = 1
    Waitms 30
    Led1 = 0
    Waitms 970
  End If

  If M = 6 Then                                                 'go
    Led1 = 1
    Waitms 30
    Led1 = 0
    Waitms 100
    Led1 = 1
    Waitms 30
    Led1 = 0
    Waitms 840
  End If

Loop
End
```

3 Sparrow Werkzeuge

Wer Mikrocontroller an einem PC programmiert, hat meist einen Compiler und ein Programmiergerät am USB. Oft setzt man einen Bootloader ein, wobei das Zielsystem sich selbst programmieren kann. In jedem Fall aber braucht man einen PC mit einer passenden Schnittstelle wie z.B. eine RS232 oder einen USB-Anschluss.

Beim Sparrow ist das anders. Ein Programmiergerät braucht man nicht, denn die erforderliche Hardware ist schon vorhanden. USB oder RS232 braucht man auch nicht, denn die Programmierung läuft über die Soundkarte, genauer gesagt über den Kopfhörerausgang. Spezielle Programmiersoftware braucht man auch nicht, denn es reicht ein Internetbrowser. Ein Klick reicht, dann wird das passende Soundfile übertragen.

So einfach ist es, wenn man fertige Apps verwendet. Aber was ist, wenn man selbst Software entwickeln will? Dann braucht man einen Assembler oder einen Compiler, der ein Hex-File erzeugt. Und dieses kann dann online in das fassende Soundfile umgewandelt und auch gleich übertragen werden.

3.1 Online Hex2Wav Konverter

Fertige Apps sind gut und schön. Aber was hat das alles für einen Nutzen, wenn man nur die Tests aufspielen kann. Das wird schnell langweilig. Aus diesem Grund wurde der Codegenerator für den ATtiny13a zu einer kleinen WebApp aufgebohrt. Mit dieser Anwendung können nun beliebig Programme, unabhängig vom eingesetzten Compiler, auf den Controller geladen werden.

Der Konverter ist sehr einfach zu bedienen. Die folgende Anleitung gilt für den Tiny13 außerhalb eines Sparrow, beim Sparrow entfällt das manuelle Drücken der Reset-Taste:

- Datei auswählen (bitte eine gültige .hex für den AVR ATtiny13a oder ATtiny13)
- Schaltung anschließen
- Lautstärkeregler auf Maximum einstellen
- RESET-Leitung auf Masse ziehen
- "Upload" drücken
- Warten bist das Programm übertragen wurde (blinken)
- RESET-Leitung von Masse entfernen

Bei Misserfolg folgende Punkte überprüfen:

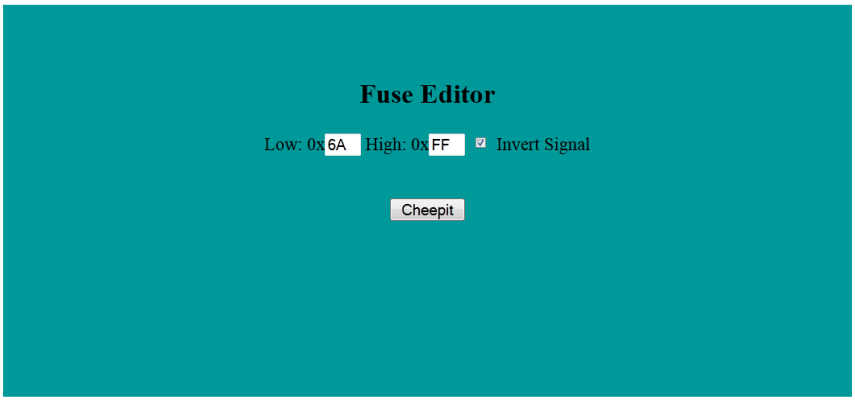
• Lautstärke kontrollieren
• Nochmals Reset auslösen
• Playtaste am eingebetteten Player drücken.

Der Sparrow erwartet die Einstellung "Signal invertieren". Nichtinvertierte Signale kann man verwenden um einen Tiny13 mit zwei Widerständen und einer Reset-Taste zu programmieren(vgl. Kap. 10).

Wenn man genau hinhört, erkennt man eine Art Vorspann des Sounds. Sie dient dazu, den Sparrow in den Reset-Zustand zu versetzen. Der Sparrow hat eine dazu passende Reset-Schaltung. Das ist der eigentliche Grund dafür, dass man bei der Programmierung mit nur zwei Leitungen auskommt. Wer die übliche ISP-Schnittstelle für AVR-Controller verwendet, kennt ja den sechspoligen Anschluss mit MOSI, MISO, SCK, Reset, GND und VCC. Hier wird auf MISO verzichtet, MOSI (linker-Kanal) erzeugt zugleich das Reset-Signal und der rechte Kanal liefert das SCK-Signal.

3.2 Fuse Editor

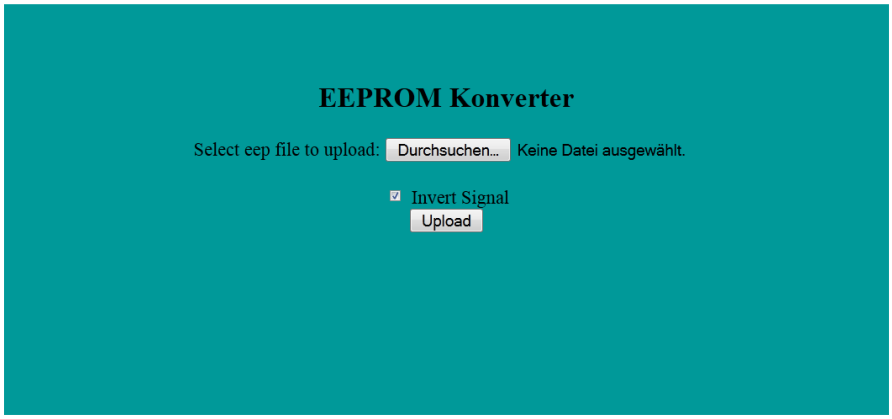
Der Fuse Editor ermöglicht das Schreiben der Low und High Fuses am Sparrow/ATtiny13a. Achtung! Bei der Veränderung der Fuses sollte man genau wissen was man tut. Durch falsche Einstellungen ist unter Umständen eine Programmierung des Controllers nicht mehr möglich.



3.3 EEPROM Konverter

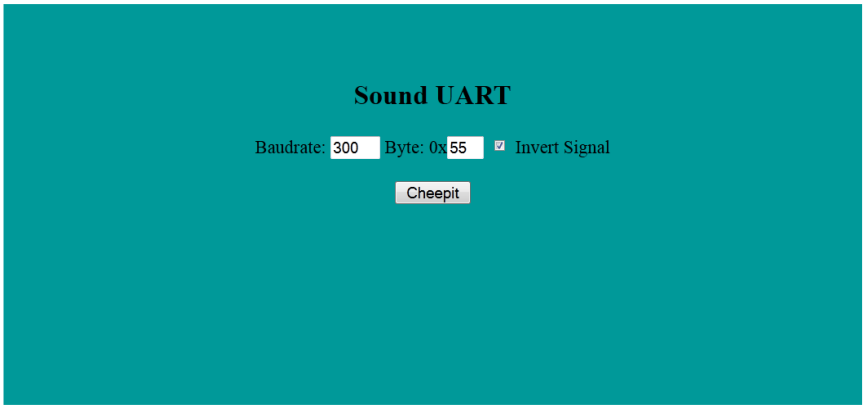
Mit dem EEPROM Konverter kann der 64 Byte große EEPROM des Sparrow beschrieben werden. Einfach ein EEPROM-Image (.eep-Datei) auswählen und analog zum Sound Konverter hochladen.

Der Programmspeicher des Sparrow (ATtiny13a) bleibt dabei unberührt. Bestehende Programme bleiben erhalten. Besteht ein Programm aus Flash-Image (.hex) und EEPROM-Image (.eep) muss zuerst der Flash beschrieben werden (mit dem Sound Konverter) erst danach der EEPROM.



3.4 Sound UART

Die Sound UART Schnittstelle ermöglicht dem Sparrow den Empfang von seriellen Daten (RS232) über den Soundausgang des angeschlossenen Gerätes (Smartphone, Tablet oder PC). Für das Übertragungsverfahren wird der linke Audiokanal verwendet (PB2/Clockleitung am Programmierinterface). Mit dem Konverter können beliebige Bytewerte umgewandelt werden. Das Verfahren liefert dem Mikrocontroller ein echtes serielles Signal mit TTL-Pegel.



Bei der Übertragung wird lediglich der linke Kanal genutzt. Der rechte Kanal (Reset/Datenleitung) bleibt unverändert. Damit ist eine Datenübertragung an PB2 während des Betriebs möglich.

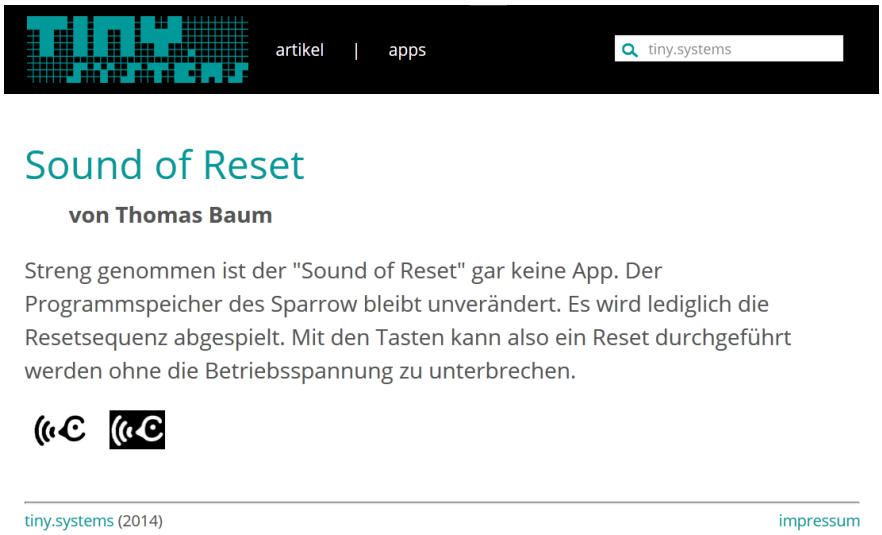
Die voreingestellte Übertragungrate beträgt 300 Baud bei 8 Datenbits, keiner Parität und einem Stoppbit. Die erzeugten Sounddaten werden mit einer Samplingrate von 44100 Hz erzeugt. Hieraus ergibt sich eine theoretisch mögliche Baudrate von maximal 22050. Bei der Konvertierung treten jedoch bedingt durch die

feste Samplingrate Rundungsfehler auf. Der Fehler kann errechnet werden. Ist die Samplingrate ganzzahlig durch die eingestellte Baudrate teilbar, gibt es keine Abweichung.

3.5 Sound of Reset

Dier Trick ist, nur die MOSI-Leitung anzusprechen, nicht aber die SCK-Leitung. Dann wird ein Reset ausgeführt, ohne eine neues Programm zu übertragen.

http://tiny.systems/categorie/cheepit/Reset.html

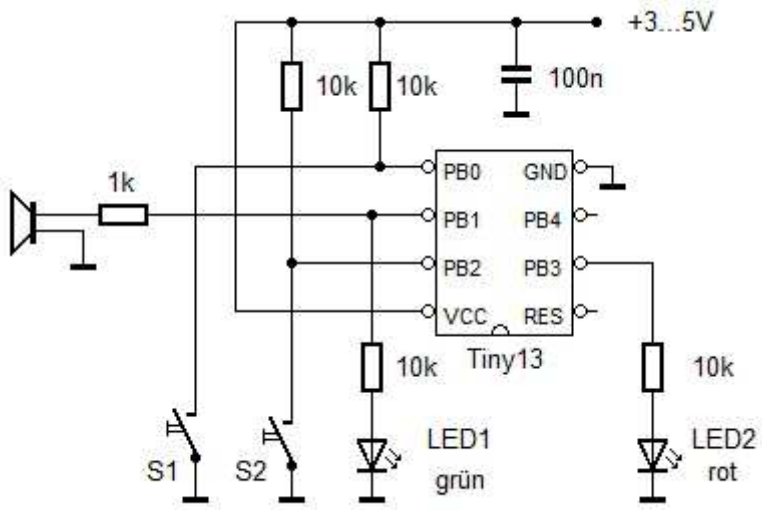


4 Morsen und Binäre Telegraphie

Da der Sparrow nicht übermäßig viele Eingabetasten besitzt, macht es Sinn Daten seriell mit nur einer Taste zu übertragen. Da könnte man natürlich den guten alten Morsecode verwenden. Aber ein Mikrocontroller arbeitet ja intern eher mit Bits und Bytes, die man auch direkt und ohne Umwege über eine Codierung darstellen kann. Der Code sieht so aus:

- Wie beim Morsen werden Punkte und Striche verwendet, Geschwindigkeiten und Strichlängen sind gleich, z.B. 100 ms und 300 ms.
- Punkte stehen für Nullen, Striche stehen für Einsen
- Die Übertragung eine Bytes beginnt mit dem niederwertigsten Bit
- Führende Nullen dürfen unterdrückt werden.

Ein Beispiel: Die Zahl 2 heißt binär 0010 und wird mit dem Low-Bit zuerst als dit dah gesendet. dit dah dit dit oder dit dah dit gehen auch, weil beliebig viele Nullen folgen können ohne das Ergebnis zu ändern.



Video: **http://youtu.be/8-oqc-X2cmo**

Wer morsen kann und die binäre Telegraphie noch nicht gewohnt ist wundert sich vielleicht: a, r und l bedeuten dasselbe, nämlich 2. Die Ziffern von 0 bis 9 hören sich ohne führende Nullen an wie die Morsezeichen e t a m u k w o v x.

Wer nicht morsen kann hat es vielleicht leichter, denn er hört einfach nur Nullen und Einsen. Hier noch ein paar Beispiele für Zahlen bis 4 Bit:

0 = dit
1 = dah
2 = dit dah
4 = dit dit dah

8 = dit dit dit dah
10 = dit dah dit dah
15 = dah dah dah dah

Es geht aber auch für größere Zahlen bis 8 Bit:

255 (&HFF, 1111 1111) = dah dah dah dah dah dah dah dah
100 (&H64, 0110 0100) = dit dit dah dit dit dah dah
128 (&H80, 1000 0000) = dit dit dit dit dit dit dit dah

Man kann die Zeichen auch ohne einen Lautsprecher optisch an der LED 1 nachverfolgen, was allerdings schwieriger ist, weil der Mensch meist schneller hört als er sehen kann. Wenn es aber nur um kleine Zahlen und damit kurze Ein- und Ausgaben geht reicht auch die LED. Das Verfahren kann nach einiger Übung auch zum Debuggen verwendet werden, indem man irgendwelche Zwischenergebnisse ausgibt.

4.1 Sparrow_TeleBin

Die App Sparrow_TeleBin dient zum Ausprobieren und zum Üben der Binärtelegraphie. Am besten schließt man einen kleinen Lautsprecher oder Piezo-Schallwandler an. Dann gibt man über S1 beliebige Binärzahlen ein. Der Sparrow sendet die gleichen Binärzahlen zurück, wobei führende Nullen unterdrückt werden. Gleichzeitig erhält man einen genauen Eindruck von der korrekten Geschwindigkeit mit 100 ms und 300 ms (entspricht 60 Buchstaben pro Minute beim Morsen). Man erhält auf jede Eingabe eine korrigierende Antwort, sodass man nach kurzer Zeit den richtigen Takt drauf hat.

http://tiny.systems/categorie/cheepit/Bin%C3%A4rtelegraphie.html

```
'Sparrow_TeleBin.bas
'Dit 100 ms = 0, Dah 300 ms = 1, Low Bit first

$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010
Dim D As Byte
Dim T As Byte
Dim B As Byte
Dim C As Byte
Declare Sub Receivebinary
Declare Sub Sendbinary

' Timer 0 is configured for Sound at OC0B / PB1
Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

Do
```

```
 Receivebinary
 Waitms 300
 Sendbinary
Loop

Sub Receivebinary
  C = 0
  B = 1
  Do
     Do
     Loop Until S1 = 0
     Pwm0b = 80
     T = 0
     Do
       Waitms 10
       T = T + 1
     Loop Until S1 = 1
     Pwm0b = 0
     Waitms 10
     If T > 15 Then
        C = C + B
     End If
     Shift B , Left
     T = 0
     Do
       Waitms 10
       T = T + 1
     Loop Until S1 = 0 Or T > 25
  Loop Until T > 25
End Sub

Sub Sendbinary
  Do
    B = C And 1
    If B = 1 Then
      Pwm0b = 80
      Waitms 300
    Else
      Pwm0b = 80
      Waitms 100
    End If
    Pwm0b = 0
    Waitms 100
   Shift C , Right
  Loop Until C = 0
End Sub
```

4.2 Ein Code-Schloss

Hier muss der Anwender vier geheime Zahlen in genau der richtigen Reihenfolge eingeben. Die Voreinstellung lautet 1 2 3 4, aber für ernste Anwendungen könnte man sich intelligentere Zahlen ausdenken. Wenn alle vier Eingaben richtig sind geht die rote LED an. Hier könnte z.B. eine Transistor-Schaltstufe mit Relais und Türöffner angeschlossen werden. Für die Anwendung ist es vermutlich besser, keinen Lautsprecher anzuschließen, damit die Zahlenkombination möglichst nicht mitgehört werden kann. Die gewählten Geheimzahlen lauten im Format der binären Telegraphie wie folgt:

dah

dit dah
dah dah
dit dit dah

Und am Ende kann man einmal kurz drücken um die Tür wieder zu schließen. Zur Verwirrung neugieriger Gäste kann man auch ein beliebiges anderes Zeichen eingeben, solange es anders ist als die erste Geheimzahl. Genauso kann man auch beim Öffnen einen langen Vorspann unsinniger Daten senden, bis jeder Lauscher frustriert aufgibt. Erst dann kommen die richtigen Zahlen 1 2 3 4.

http://tiny.systems/categorie/cheepit/Bin%C3%A4rtelegraphieCodeSchloss.html

```
'SparrowTeleBinLock.bas
'Dit 100 ms = 0, Dah 300 ms = 1, Low Bit first
'Secret code = 1 2 3 4

$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010

Dim D As Byte
Dim T As Byte
Dim B As Byte
Dim C As Byte

Declare Sub Receivebinary

' Timer 0 is configured for Sound at OC0B / PB1
Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

Do
C1:
 Receivebinary
 Led2 = 0
 If C <> 1 Then Goto C1
 Receivebinary
 If C <> 2 Then Goto C1
 Receivebinary
 If C <> 3 Then Goto C1
 Receivebinary
 If C <> 4 Then Goto C1
 Led2 = 1
Loop

Sub Receivebinary
  C = 0
  B = 1
  Do
     Do
     Loop Until S1 = 0
     Pwm0b = 80
     T = 0
     Do
```

```
      Waitms 10
      T = T + 1
    Loop Until S1 = 1
    Pwm0b = 0
    Waitms 10
    If T > 15 Then
       C = C + B
    End If
    Shift B , Left
    T = 0
    Do
      Waitms 10
      T = T + 1
    Loop Until S1 = 0 Or T > 25
  Loop Until T > 25
End Sub
```

4.3 Sparrow-Merkhilfe

Lange Zahlenreihen wie z.B. Telefonnummern kann man leicht mal vergessen. Aber im EEPROM des Sparrow sind sie gut aufgehoben, egal ob in Byteform oder als Ziffern 0 bis 9. Beim Start des Geräts wird die zuletzt gespeicherte Zahlenfolge optisch und akustisch telegraphiert. Dann hat man die Möglichkeit eine neue Zahlenfolge einzugeben. Das Ende wird durch eine Zahl 255 (acht Striche) gekennzeichnet. Ohne neue Eingabe bleibt die alte Zahlenfolge erhalten und wird beim nächsten Neustart oder Reset neu ausgegeben.

http://tiny.systems/categorie/cheepit/Bin%C3%A4rtelegraphieMerkhilfe.html

```
'SparrowTeleBinMemory.bas
'Dit 100 ms = 0, Dah 300 ms = 1, Low Bit first
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010

Dim D As Byte
Dim T As Byte
Dim B As Byte
Dim C As Byte
Dim A As Byte

Declare Sub Receivebinary
Declare Sub Sendbinary
' Timer 0 is configured for Sound at OC0B / PB1
Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

S:
A = 0
Do
  Readeeprom C , A
  If C < 255 Then Sendbinary
```

```
  A = A + 1
  Waitms 500
Loop Until C = 255

A = 0
Do
  Receivebinary
  D = C
  Writeeeprom D , A
  A = A + 1
Loop Until D = 255
Goto S


Sub Receivebinary
  C = 0
  B = 1
  Do
     Do
     Loop Until S1 = 0
     Pwm0b = 80
     T = 0
     Do
       Waitms 10
       T = T + 1
     Loop Until S1 = 1
     Pwm0b = 0
     Waitms 10
     If T > 15 Then
        C = C + B
     End If
     Shift B , Left
     T = 0
     Do
       Waitms 10
       T = T + 1
     Loop Until S1 = 0 Or T > 25
  Loop Until T > 25
End Sub

Sub Sendbinary
  Do
    B = C And 1
    If B = 1 Then
      Pwm0b = 80
      Waitms 300
    Else
      Pwm0b = 80
      Waitms 100
    End If
    Pwm0b = 0
    Waitms 100
   Shift C , Right
  Loop Until C = 0
End Sub
```

4.4 Sparrow_Reaktiontest

Noch frisch, oder zu viel gearbeitet? Der Reaktionstest bringt es an den Tag. Es geht darum, sofort nach

einem Lichtsignal auf eine Taste zu drücken. Wenn man kürzer als 200 ms dafür braucht, ist noch alles im Lot. Um die Messung zu starten drückt man kurz auf S1. Nach einer zufälligen kurzen Wartezeit geht die rote LED an. So schnell wie möglich drückt man dann nochmal auf S1. Der Sparrow misst die Verzögerungszeit in Hundertstelsekunden und teilt sie dem Anwender in Form von zwei Ziffern in binärer Telegrafie mit. 2 7 bedeutet 27 * 1/100 s bzw. 270 ms. Ein Schallwandler an B1 ist angenehm, aber man kann es auch optisch über die grüne LED lesen.

http://tiny.systems/categorie/cheepit/Reaktionstest.html

```
'Sparrow_Reaktionstest.bas

$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010

Dim D As Byte
Dim T As Byte
Dim B As Byte
Dim C As Byte

Declare Sub Receivebinary
Declare Sub Sendbinary

' Timer 0 is configured for Sound at OC0B / PB1
Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

Do
 Do
 Loop Until S1 = 0                                          'S1 drücken:
Start
 Waitms 50
 Do
   Waitms 10
 Loop Until S1 = 1
 Waitms 600
 Do
   T = Rnd(250)                                             'Zufällige
Wartezeit
   Waitms T
   Waitms T
 Loop Until S1 = 1
 D = 0
 Led2 = 1                                                   'Startsignal
 Do
   Waitms 10
   D = D + 1                                                'Zeitmessung
 Loop Until S1 = 0
 Led2 = 0
 Waitms 500
 C = D / 10                                                 'Zehner 1/100 s
 Sendbinary
```

```
 Waitms 600
 C = D Mod 10                                                    'Einner 1/100 s
 Sendbinary
 Waitms 300
Loop

Sub Sendbinary
  Do
    B = C And 1
    If B = 1 Then
      Pwm0b = 80
      Waitms 300
    Else
      Pwm0b = 80
      Waitms 100
    End If
    Pwm0b = 0
    Waitms 100
   Shift C , Right
  Loop Until C = 0
End Sub
```

4.5 Der Sparrow-Sputnik

1957 sorgte der erste Sputnik für Aufsehen, der erste Satellit, der einfach nur Tonsignale auf 20 MHz und 40 MHz aus dem All sandte. Der Sender war damals mit Röhren bestückt und stellte seinen Dienst ein als die Batterien leer waren. Genau 57 Jahre später ist der Sparrow-Sputnik gestartet, bestückt mit einem stromsparenden Tiny13 und auf Mittelwelle 520 kHz (Welle 600 m, wie man damals teilweise noch sagte). Der Spatz fliegt nicht so hoch und braucht keine externe Antenne, hat aber auch keine große Reichweite, sondern setzt sich direkt auf das Radio, aus dem er gehört werden möchte. Man legt ihn möglichst nahe an die Ferritantenne. Das Sputnik-Signal erscheint auf etwa 520 kHz und den Oberwellen 1040 kHz und 1560 kHz, alles plusminus ein paar Prozent. Der "HF-Oszillator" besteht aus einer Programmschleife und schaltet LED1 in der höchsten möglichen Geschwindigkeit ein und aus. Nach jeweils 200 Schwingungen kommt eine Pause von 0,8 ms, sodass das ganze Signal mit rund 1 kHz moduliert ist. Und auf diese Weise werden Impulspakete im Abstand von etwa einer Sekunde erzeugt. Die Leitungen zur grünen LED auf der Platine bilden eine Windung der Loop-Antenne und koppeln das Signal magnetisch auf den Ferritstab des Radios. Piep, piep ... cheep, cheep ...

Video: http://youtu.be/aczOE3Z5WxY

http://tiny.systems/categorie/cheepit/Sputnik.html

```
'Sparrow_Sputnik.bas
$regfile = "attiny13.dat"
$crystal = 9600000
$hwstack = 8
$swstack = 4
$framesize = 4

Dim N As Byte
Dim M As Byte

Config Portb = &B00011111
Led1 Alias Portb.1
Led2 Alias Portb.3

Clkpr = 128                                                      '9,6 MHz
Clkpr = 0
Clkpr = 0
```

```
Do
  Led2 = 1
  For M = 1 To 40
    For N = 1 To 200                                        'ca. 520 kHz
     Led1 = 1
     Led1 = 0
    Next N
    Waitus 800                                              'ca 1 kHz AM
  Next M
  Led2 = 0
  Waitms 900
Loop
End
```

4.6 Der Sparrow-Morse-Sputnik

Zwar hat sich die binäre Telegraphie bei der Eingabe von Zahlen bewährt, aber wenn es um Textzeichen geht, hat die Morse-Telegraphie weiterhin ihre Bedeutung. Und da der Sparrow gleich auch sein eigener Hochfrequenzsender ist, ist diese Art der Datenübertragung eine praktische Sache. Das grundlegende Verfahren der Zeichenerkennung und -Kodierung wurde bereits mit anderen AVR-Controllern vorgestellt.

Das Programm erzeugt ein mit ca. 1 kHz moduliertes HF-Signal bei ca. 520 kHz. Wie beim Sparrow-Sputnik kann das Signal auch auf Oberwellen bei ca. 1 MHz und bei 1,5 MHz gehört werden. Das Morsesignal hat eine Geschwindigkeit von etwa 60 BpM. Es werden zufällige Fünfergruppen erzeugt, wie man sie gern beim Üben verwendet. Der Sparrow wird dazu einfach auf ein Mittelwellenradio gelegt und koppelt dann induktiv auf die Ferritantenne. Es werden alle gesetzlichen Grenzwerte für die induktive Übertragung eingehalten, und die Nachbarn hören nichts davon.

http://tiny.systems/categorie/cheepit/MorseSputnik.html

```
'Sparrow_Morsen_Sputnik.bas

$regfile = "attiny13.dat"
$crystal = 9600000
$hwstack = 8
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010

Dim D As Byte
Dim T As Byte
Dim B As Byte
Dim C As Byte
Dim N As Byte
Dim M As Byte
Dim I As Byte
Dim J As Byte
```

```
Clkpr = 128                                                    '9,6 MHz
Clkpr = 0
Clkpr = 0

Declare Sub Receivemorse
Declare Sub Sendmorse

Do
  For I = 1 To 5                                               'Zufällige
Fünfergruppen A...Z
    C = Rnd(24)
    C = C + 2
    If C = 19 Then C = 27
    If C = 21 Then C = 28
    Sendmorse
    Waitms 300
  Next I
  Waitms 1500
Loop

Sub Receivemorse
  C = 0
  Do
     Do
     Loop Until S1 = 0
     T = 0
     Do
       For M = 1 To 10
         For N = 1 To 250                                      'ca. 520 kHz
           Led1 = 1
           Led1 = 0
         Next N
         Waitms 1                                              'ca 1 kHz AM
       Next M
       T = T + 1
     Loop Until S1 = 1
     Waitms 10
     If T > 15 Then
        C = C + B
     End If
     Shift B , Left
     T = 0
     Do
       Waitms 10
       T = T + 1
     Loop Until S1 = 0 Or T > 25
  Loop Until T > 25
End Sub

Sub Sendmorse
  J = 1
  Do
    Shift C , Left
    B = C And 128
    J = J + 1
  Loop Until B = 128
  Do
    Shift C , Left
    J = J + 1
    B = C And 128
    If B = 128 Then
      For T = 1 To 240
        For N = 1 To 250                                       'ca. 520 kHz
```

```
          Led1 = 1
          Led1 = 0
        Next N
        Waitms 1                                          'ca 1 kHz AM
      Next T
    Else
      For T = 1 To 80
        For N = 1 To 250                                  'ca. 520 kHz
          Led1 = 1
          Led1 = 0
        Next N
        Waitms 1                                          'ca 1 kHz AM
      Next T
    End If
    Waitms 100
  Loop Until J = 8
End Sub
```

4.7 Sparrow-Morse-Memory

Da möchte man sich etwas notieren, aber der Bleistift ist abgebrochen. Dann bleibt nur der Sparrow, was sonst. Die Notiz wird einfach hineingemorst und kann nach einem Reset wieder abgerufen werden. Auch hier geht es wieder um echte Morsezeichen (nicht um Zahlen im binären Telegraphie-Code). Aber trotzdem sucht man die Morsezeichentabelle im Programm vergeblich. Eigentlich kann der Sparrow gar nicht morsen, er merkt sich einfach nur Punkte und Striche sowie Zeichenpausen, und die gibt er dann wieder. Mit einer Ausnahme, er kennt das Ende-Zeichen .-.-. und setzt dann eine 255 in das EEPROM. So weiß er im Abspielmodus, dass die Nachricht an ihr Ende gekommen ist.

Der Sparrow gibt alles wieder, mit allen Sonderzeichen, die man vielleicht gerade erst erfunden hat, und mit allen Fehlern, die man eventuell gemorst hat. Das Programm eignet sich daher auch zum Üben. Am Anfang braucht es vielleicht mehrere Versuche, bis die Notiz richtig drin ist. Aber nach einiger Zeit geht es immer besser. Dann kann man natürlich auch einen Standard-CQ-Ruf einspeichern und damit einen Sender steuern. Oder einfach auf Mittelwelle abhören, denn auch dieses Programm bildet wie der Sputnik einen kleinen AM-Sender. Wer mehr auf NF steht kann aber auch einen Lautsprecher mit 1-k-Vorwiderstand an B1 anschließen.

http://tiny.systems/categorie/cheepit/MorseMemory.html

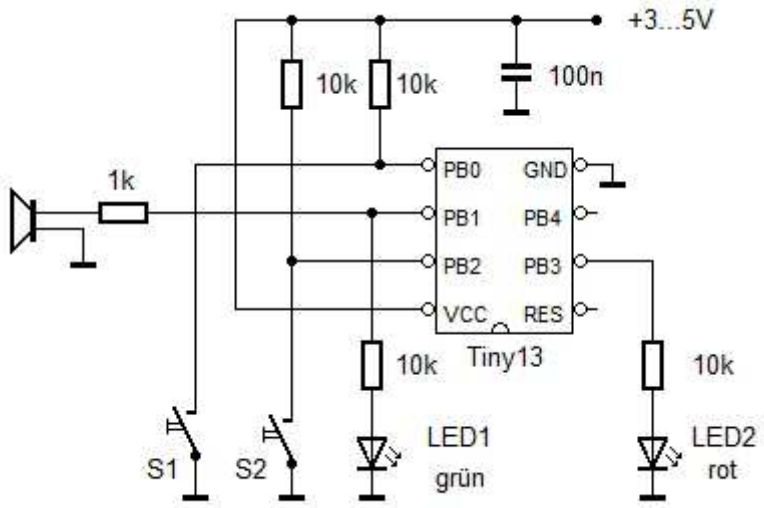
```
'Sparrow_Morse_Memory.bas

$regfile = "attiny13.dat"
$crystal = 9600000
$hwstack = 8
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010

Dim D As Byte
Dim T As Byte
```

```
Dim B As Byte
Dim C As Byte
Dim N As Byte
Dim M As Byte
Dim I As Byte
Dim J As Byte
Dim A As Byte

Clkpr = 128                                                    '9,6 MHz
Clkpr = 0
Clkpr = 0

Declare Sub Receivemorse
Declare Sub Sendmorse

Do
  A = 0
  Do
    Readeeprom C , A
    If C < 255 Then Sendmorse
    A = A + 1
    Waitms 300
  Loop Until C = 255
  A = 0
  Do
    T = 0
    Do
       Waitms 10
       T = T + 1
       If T > 30 Then T = 30
    Loop Until S1 = 0
    If T = 30 Then
      D = 1
      Writeeeprom D , A
      A = A + 1
    End If
    Receivemorse
    D = C
    Writeeeprom D , A
    A = A + 1
    If C = &B00101010 Then
      D = 255
      Writeeeprom D , A
    End If
  Loop Until D = 255
Loop

Sub Receivemorse
  C = 1
  Do
     Shift C , Left
     Do
     Loop Until S1 = 0
     T = 0
     Do
       For M = 1 To 10
         For N = 1 To 250                                      'ca. 520 kHz
           Led1 = 1
           Led1 = 0
         Next N
         Waitms 1
       Next M
       T = T + 1
```

```
     Loop Until S1 = 1
    ' Pwm0b = 0
     Waitms 10
     If T > 15 Then
        C = C + 1
     End If
     Shift B , Left
     T = 0
     Do
       Waitms 10
       T = T + 1
     Loop Until S1 = 0 Or T > 25
  Loop Until T > 25
End Sub


Sub Sendmorse
If C = 1 Then
  Waitms 300
Else
  J = 1
  Do
    Shift C , Left
    B = C And 128
    J = J + 1
  Loop Until B = 128
  Do
    Shift C , Left
    J = J + 1
    B = C And 128
    If B = 128 Then
      For T = 1 To 240
        For N = 1 To 250                                    'ca. 520 kHz
          Led1 = 1
          Led1 = 0
        Next N
        Waitms 1                                            'ca 1 kHz AM
      Next T
    Else
      For T = 1 To 80
        For N = 1 To 250                                    'ca. 520 kHz
          Led1 = 1
          Led1 = 0
        Next N
        Waitms 1                                            'ca 1 kHz AM
      Next T
    End If
    Waitms 100
  Loop Until J = 8
End If
End Sub
```

5 Mikro-TPS für den Sparrow

Nachdem mit der binären Telegraphie eine passende Eingabemethode für den Sparrow entwickelt wurde kam die Frage auf, ob man damit nicht wenigstens eine ganz kleine Tastenprogrammierbare Steuerung (TPS) realisieren könnte. Einschränkungen ergeben sich aus den wenigen Portleitungen und aus dem begrenzten Speicher. Und tatsächlich hat sich gezeigt, dass der Programmspeicher des Tiny13 schnell voll wird. Hier eine erste Version der Mikro-TPS.



Video: **http://youtu.be/MKM-CHtdWxg**

5.1 Sparrow_TPS1

Wie gewohnt hat das Programm eine Tonausgabe für die binäre Telegraphie. Der Anwender drückt beim Start auf S1 um den Programmiermodus zu starten. LED2 geht an um die Eingabe des ersten Befehls anzuzeigen. Man gibt zuerst das 4-Bit Kommando ein, dann die 4-Bit Daten. Während der Datenausgabe ist LED2 aus, danach geht sie für die nächste Befehls-Eingabe wieder an. Das erleichtert die Orientierung. Nach der letzten Eingabe tippt man FF ein um den Programmiermodus zu beenden. Nach der Eingabe oder nach dem Start ohne Eingabe gibt die TPS das eigegebene Programm zunächst akustisch aus, damit man eine Kontrolle hat ob alles richtig eingegeben wurde. danach wird das eingegebene Programm ausgeführt. In dieser Version ist nur der Ausgang B3 (1 8 = LED2 an) nutzbar, weil LED1 als Sound-Ausgang konfiguriert wurde.

Es existieren nur drei Befehle:
1, Portausgabe an Port B
2, Wartezeit 1ms bis 60 s
3, Sprung zurück

Direkt laden: http://tiny.systems/categorie/cheepit/TPS1.html

```
'Sparrow_TPS1.bas
'Dit 100 ms = 0, Dah 300 ms = 1, Low Bit first
regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
```

```
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010

Dim D As Byte
Dim T As Byte
Dim B As Byte
Dim C As Byte
Dim A As Byte
Dim Addr As Byte
Dim Eebyte As Byte
Dim Dat As Byte
Dim Kom As Byte
Dim Adrhi As Byte
Dim Adrlo As Byte
Dim Adrret As Byte
'Dim Prog As Byte
Dim Dd As Word
Dim Times(16) As Word
Dim Delaytime As Word


Dta:
Data 1% , 2% , 5% , 10% , 20% , 50% , 100% , 200% , 500% , 1000% , 2000% ,
5000% , 10000% , 20000% , 30000% , 60000%



Declare Sub Receivebinary
Declare Sub Sendbinary

' Timer 0 is configured for Sound at OC0B / PB1
Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

If S1 = 0 Then
 Receivebinary
 A = 0
 Do
 Led2 = 1
 Receivebinary
 D = C * 16
 Led2 = 0
 Receivebinary
 C = C And 15
 D = D + C
 Writeeeprom D , A
 A = A + 1
 Loop Until D = 255

'Else
 ' Dat = &H12 : Writeeeprom Dat , 0 'Dout=2
 ' Dat = &H29 : Writeeeprom Dat , 1 '1000 ms
 ' Dat = &H18 : Writeeeprom Dat , 2 'Dout=8
 ' Dat = &H29 : Writeeeprom Dat , 3 '1000 ms
 ' Dat = &H34 : Writeeeprom Dat , 4 'Adr = Adr - 4
End If

A = 0
```

```
Do
 Readeeprom Eebyte , A
 If Eebyte < 255 Then
 Led2 = 1
 C = Eebyte / 16
 Sendbinary
 Waitms 500
 Led2 = 0
 C = Eebyte And 15
 Sendbinary
 A = A + 1
 Waitms 500
 End If
Loop Until Eebyte = 255



 Addr = 0
 Do
 Readeeprom Eebyte , Addr
 Addr = Addr + 1
 Dat = Eebyte And 15
 Kom = Eebyte / 16
 If Kom = 1 Then '1: Direkte Portausgabe
 Portb = Dat Or &B11100101
 End If
 If Kom = 2 Then
 Restore Dta
 Delaytime = Lookup(dat , Dta)
 Waitms Delaytime '2: Wartezeit
 End If
 If Kom = 3 Then '3: Sprung - relativ
 Addr = Addr - 1
 Addr = Addr - Dat
 End If
 If Kom = 4 Then
 A = Dat
 End If
 Loop


Sub Receivebinary
 C = 0
 B = 1
 Do
 Do
 Loop Until S1 = 0
 Pwm0b = 80
 T = 0
 Do
 Waitms 10
 T = T + 1
 Loop Until S1 = 1
 Pwm0b = 0
 Waitms 10
 If T > 15 Then
 C = C + B
 End If
 Shift B , Left
 T = 0
 Do
 Waitms 10
 T = T + 1
```

```
 Loop Until S1 = 0 Or T > 25
 Loop Until T > 25
End Sub

Sub Sendbinary
 Do
 B = C And 1
 If B = 1 Then
 Pwm0b = 80
 Waitms 300
 Else
 Pwm0b = 80
 Waitms 100
 End If
 Pwm0b = 0
 Waitms 100
 Shift C , Right
 Loop Until C = 0
End Sub
```

5.2 Sparrow_TPS2

Die zweite Version der Mikro-TPS kennt mehr Befehle, hat dafür aber eine sehr spartanische Eingabe. Diesmal wurde auf die Tonausgabe und auf das Auslesen des Programms verzichtet. Man muss das Programm also mit voller Konzentration "blind" eingeben, was aber wegen der einfachen 4-Bit-Eingaben gelingt. Der eingesparte Speicherplatz wurde für weitere Befehle genutzt. Die Ausgabe-Bits wurden umsortiert, sodass man nun drei Ausgabeports hat.

Für einen Neustart des TPS-Programms kann man übrigens die Reset-App verwenden:
http://tiny.systems/categorie/cheepit/Reset.html

Die verfügbaren TPS-Befehle:
1, Portausgabe an Port B, 1=B1, 2=B3, 4 =B4
2, Wartezeit 1ms bis 60 s
3, relativer Sprung zurück
4, A
5 2, C=A
8, AdrHi
9, AdrLo, absoluter Sprung
A, C-mal
C 4, Skip if S1=1
C 5, Skip if S2 =1

Programmbeispiele:
Gegentaktblinker: 11 28 12 28 34
RS-Schalter S1, S2: C5 12 C4 11 34
Zeitschalter S2, 1 min: C5 93 32 13 2F 10 36

http://tiny.systems/categorie/cheepit/TPS2.html

```
'Sparrow_TPS2.bas
'Dit 100 ms = 0, Dah 300 ms = 1, Low Bit first
$regfile = "attiny13.dat"
$crystal = 1200000
```

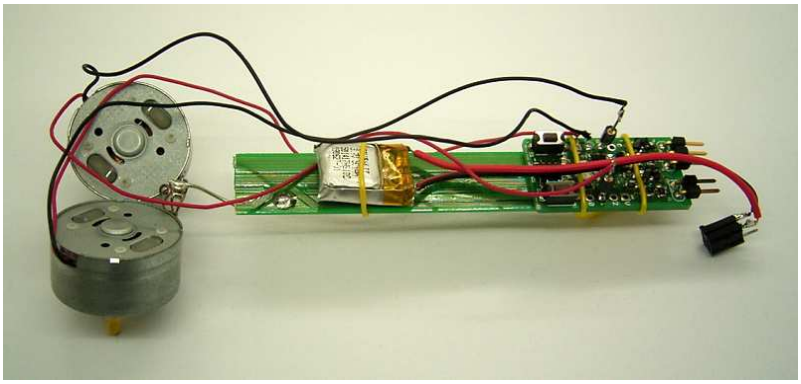
```
$hwstack = 8
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010

Dim D As Byte
Dim T As Byte
Dim B As Byte
Dim C As Byte
Dim A As Byte
Dim Addr As Byte
Dim Eebyte As Byte
Dim Dat As Byte
Dim Kom As Byte
Dim Adrhi As Byte
Dim Adrlo As Byte
Dim Adrret As Byte
'Dim Prog As Byte
Dim Dd As Word
Dim Times(16) As Word
Dim Delaytime As Word

Dta:
Data 1% , 2% , 5% , 10% , 20% , 50% , 100% , 200% , 500% , 1000% , 2000% ,
5000% , 10000% , 20000% , 30000% , 60000%

Declare Sub Receivebinary
Declare Sub Sendbinary

' Timer 0 is configured for Sound at OC0B / PB1
' Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up

If S1 = 0 Then
 Receivebinary
 A = 0
 Do
 Led2 = 1
 Receivebinary
 D = C * 16
 Led2 = 0
 Receivebinary
 C = C And 15
 D = D + C
 Writeeeprom D , A
 A = A + 1
 Loop Until D = 255
End If

 Addr = 0
 Do
 Readeeprom Eebyte , Addr
 Addr = Addr + 1
 Dat = Eebyte And 15
 Kom = Eebyte / 16
 If Kom = 1 Then '1: Direkte Portausgabe
 Led1 = Dat.0
 Led2 = Dat.1
 Portb.4 = Dat.2
```

```
 End If
 If Kom = 2 Then
 Restore Dta
 Delaytime = Lookup(dat , Dta)
 Waitms Delaytime '2: Wartezeit
 End If
 If Kom = 3 Then '3: Sprung - relativ
 Addr = Addr - 1
 Addr = Addr - Dat
 End If
 If Kom = 4 Then
 A = Dat
 End If
 If Kom = 5 Then
 If Dat = 2 Then C = A
 End If
 If Kom = 8 Then
 Adrhi = Dat 'Oberes Nibble der Adresse
 End If
 If Kom = 9 Then
 Addr = Adrhi * 16 'Springe absolut 0...255
 Addr = Addr + Dat
 End If
 If Kom = 10 Then
 C = C - 1
 C = C And 15
 If C > 0 Then 'C-mal
 Addr = Adrhi * 16 'Springe absolut 0...255
 Addr = Addr + Dat
 End If
 End If
 If Kom = 12 Then
 If Dat = 4 Then
 If S1 = 1 Then Addr = Addr + 1
 End If
 If Dat = 5 Then
 If S2 = 1 Then Addr = Addr + 1
 End If
 End If
 Loop

Sub Receivebinary
 C = 0
 B = 1
 Do
 Do
 Loop Until S1 = 0
 ' Pwm0b = 80
 T = 0
 Do
 Waitms 10
 T = T + 1
 Loop Until S1 = 1
 ' Pwm0b = 0
 Waitms 10
 If T > 15 Then
 C = C + B
 End If
 Shift B , Left
 T = 0
 Do
 Waitms 10
 T = T + 1
```

```
 Loop Until S1 = 0 Or T > 25
 Loop Until T > 25
End Sub
```

5.3 Der Sparrow-Bot

Zwei sparsame Motoren wurden direkt an Port B1 (grün) und B3 (rot) angeschlossen. Auf die Achsen wurden kurze Abschnitte weicher Drahtisolierung geschoben. Das sind nun die Räder, ein getriebefreier Antrieb. Hinten wird der Robbi von einer Kufe aus Draht geführt.



Alles zusammen mit dem Li-Akku auf ein Stückchen Schrottplatine gebunden, fertig ist der Sparrow-Bot. Die Software ist ganz einfach und wurde mit Bascom geschrieben:

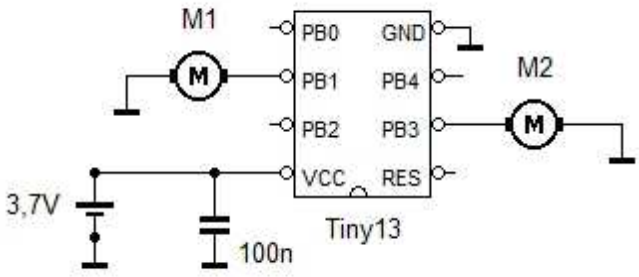
http://tiny.systems/categorie/cheepit/SparrowBot.html

```
'ATtiny13 Sparrow_Bot1.bas
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Config Portb = &B000011010

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Led1 = 0
Led2 = 0
Do
 Led1 = 1
 Led2 = 1
 Waitms 500
 Led1 = 0
 Led2 = 0
 Waitms 500
```

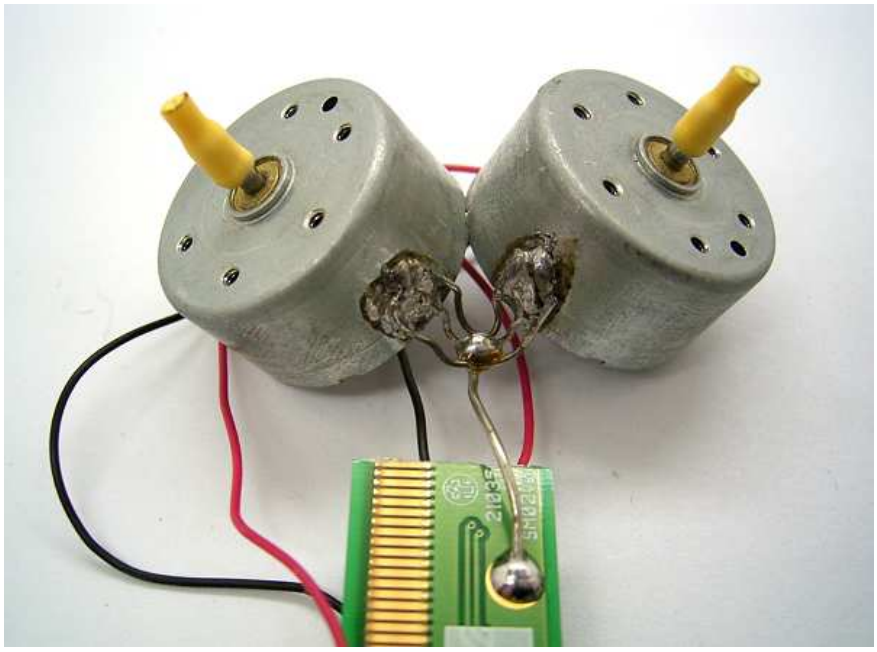
```
 Led1 = 1
 Waitms 500
 Led1 = 0
 Waitms 500
Loop
End
```

Aber mit der TPS geht es noch besser und hat den Vorteil, dass man das Programm jederzeit auf dem Parkett verändern kann. Das Programm besteht nur aus vier Ausgaben und vier Wartezeiten. Auf den Rücksprung kann verzichtet werden, denn der gelöschte Speicher enthält durchgehend FF, was nicht ausgeführt wird und am Ende wieder zum Anfang führt:

13 29 10 29 11 29 10 29



Ganz ohne Motortreiber, ist das denn möglich? Es ist! Und zwar wenn die Motoren extrem sparsam sind. Hier handelt es sich um Gleichstrommotoren aus Franzis-Lernpaketen, die in ähnlicher Art auch in Kassettenrekordern und in manchen CD-Laufwerken zu finden sind. Diese Motoren sind sehr leichtgängig und haben einen Leerlaufstrom von nur etwa 10 mA und eine Anlaufspannung von 0,6 V. Das Ohmmeter zeigt einen Gleichstromwiderstand von 60 Ohm. Ein solcher Motor darf direkt an einen Port des Mikrocontrollers angeschlossen werden. Offiziell erlaubt sind laut Datenblatt Ströme bis 40 mA. Bei einer Batteriespannung von 3,7 V käme man bei stehendem Motor theoretisch maximal auf ca. 60 mA (bis 80 mA wurden schon ohne Schaden getestet). In der Praxis beträgt der Motorstrom nur etwa 20 mA.



Auf die Achsen der Motoren wurden kurze Abschnitte einer weichen Kabel-Isolierung geschoben. Das sind nun die Gummiräder des Roboters. Mit ihrem geringen Durchmesser erhält man eine moderate Geschwindigkeit auch bei hoher Drehzahl, sodass die Motoren sehr leicht und mit wenig Strom laufen. Und das ganz ohne Getriebe und ohne den üblichen Lärm, den andere Roboter verbreiten. Allerdings ist dieser Robbi auf einen ebenen Untergrund angewiesen. Die Motoren wurden mit Draht an die Trägerplatine als Fahrzeug-Chassis gelötet. So kann man sie leicht ausrichten.

6 Sparrow-Apps mit Vier-Bit-Anzeigen

Eine LED steht im Normalfall für ein Bit. Mit den beiden LEDs auf dem Sparrow kann man daher eine 2-Bit-Anzeige realisieren und Zahlen zwischen Null und Drei darstellen. Aber es geht auch anders. Zwei Bit pro LED sind machbar, wenn man bestimmte Blinkmuster zusätzlich verwendet: 0 = Aus, 1 = Kurze Lichtblitze, 2 = Langes Blinken, 3 = An. Insgesamt hat man jetzt vier Bit an zwei LEDs und kann Zahlen im Bereich 0...15 darstellen. Damit lassen sich viele nützliche Apps entwickeln.

6.1 Up/Down-Counter 0...15

Oft will man irgendwelche Ereignisse, Dinge oder Personen zählen und sich dabei nicht allein auf den eigenen Kopf verlassen. Wenn man dann teilweise auch noch rückwärts zählen muss wird es noch schwieriger. Wie viele Leute sind in den Raum gegangen, und einige kamen zwischendurch wieder raus. Eine Sparrow-App kann helfen. Die Anzeige nutzt zwei Bits pro LED (0 = Aus, 1 = kurze Lichtblitze, 2 =langes Blinken, 3 = An). Der Zähler wird mit S2 um eins erhöht und mit S1 um eins verringert. Die beiden LEDs zeigen den momentanen Zählerstand im Bereich 0...15. Beispiel: LED2 zeigt kurze Blitze (1 x 4), LED1 zeigt Dauer-An (3), der Zählerstand ist 7. Nach dem Neustart lautet der Zählerstand übrigens 15 (beide LEDs an), einfach nur damit man sieht, die App ist aktiv. Ein Druck auf S2 erzeugt einen Zähler-Überlauf und setzt damit den Zählerstand auf Null. Der Up/Down-Counter ist bereit.

http://tiny.systems/categorie/cheepit/SparrowCounter.html

```
'ATtiny13 Sparrow Up/Down Counter 0...15
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
Config Portb = &B000011010

Dim N As Byte
Dim H As Byte
Dim L As Byte
Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Do
  If S2 = 0 Then
    Waitms 50
    N = N + 1
    Do
    Loop Until S2 = 1
    Waitms 50
  End If
  If S1 = 0 Then
    Waitms 50
    N = N - 1
    Do
     Loop Until S1 = 1
    Waitms 50
  End If
  L = N And 3                                          'Anzeige N =
```

```
0...15
  H = N
  Shift H , Right , 2
  H = H And 3
  If L = 0 Then Led1 = 0 Else Led1 = 1
  If H = 0 Then Led2 = 0 Else Led2 = 1
  Waitms 50
  If L < 2 Then Led1 = 0
  If H < 2 Then Led2 = 0
  Waitms 100
  If L < 3 Then Led1 = 0
  If H < 3 Then Led2 = 0
  Waitms 100
Loop
End
```

6.2 Sparrow-Watch bis vier Stunden

Hier geht es nicht eigentlich um die Uhrzeit sondern um eine einfache und übersichtliche Zeitmessung. Eine Tätigkeit soll z.B. möglichst nicht länger als zwei Stunden dauern. Auf ein paar Minuten kommt es nicht an, aber deutlich mehr ist nicht geplant. Da reicht eine Auflösung von Viertelstunden. Die Uhr ist ganz einfach abzulesen (0 = Aus, 1 = kurze Lichtblitze, 2 =langes Blinken, 3 = An). Die rote LED zeigt volle Stunden, die grüne LED Viertelstunden. Und es gibt einen Taster, um die Uhr auf Null zu setzen und damit neu zu starten. Nach einem Neustart steht die Uhr zunächst nicht auf Null sondern auf 1 Stunde, einfach damit man sieht, dass das Programm aktiv ist. Ein Druck auf S2 setzt die Uhr zurück. Beide LEDs sind aus. Nach einer Viertelstunde zeigt die grüne LED ihr kurzes Blitzen, nach einer halben Stunde das längere Blinken. Diese Uhr kann man auch auf langwierige Besprechungen mitnehmen. Dabei kann es von Vorteil sein, dass nicht jeder sie lesen kann. Und man ist ja nicht so unhöflich, dauernd auf die Uhr zu schauen. Trotzdem behält man den Überblick. Rot aus, Grün zeigt langes Blinken: Der Chef redet jetzt schon seit einer halben Stunde...

http://tiny.systems/categorie/cheepit/SparrowWatch.html

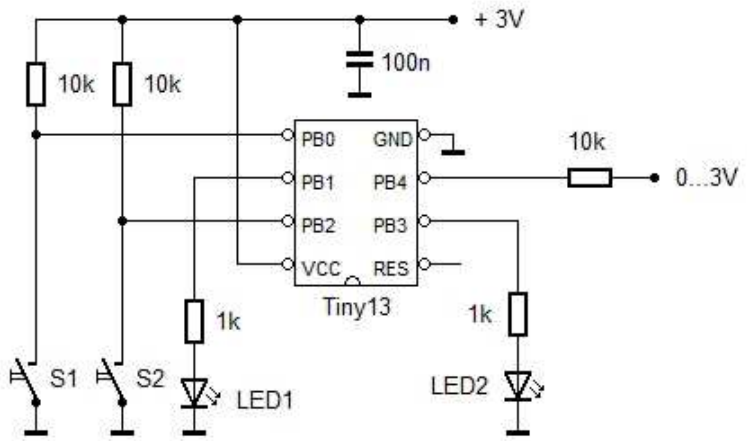
```
'ATtiny13 Sparrow Watch 0...4 h
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4
Config Portb = &B000011010

Dim T As Word
Dim N As Byte
Dim H As Byte
Dim L As Byte

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

N = 4

Do
 For T = 1 To 3600                                          '3600*250 ms =
15 Min
  If S2 = 0 Then                                            'Time Reset
```

```
    Waitms 50
    T = 0
    N = 0
    Do
    Loop Until S2 = 1
    Waitms 50
  End If
  L = N And 3                                              'Anzeige N =
0...15
  H = N
  Shift H , Right , 2
  H = H And 3
  If L = 0 Then Led1 = 0 Else Led1 = 1
  If H = 0 Then Led2 = 0 Else Led2 = 1
  Waitms 50
  If L < 2 Then Led1 = 0
  If H < 2 Then Led2 = 0
  Waitms 100
  If L < 3 Then Led1 = 0
  If H < 3 Then Led2 = 0
  Waitms 100
 Next T
 N = N + 1
Loop
End
```

6.3 Das Sparrow-LED-Voltmeter

Es geht um ein einfaches Voltmeter, also um die Messung und Anzeige von Spannungen. Die Messung selbst ist einfach, denn dafür gibt es ja den 10-Bit AD-Wandler und den freien Pin B4 (=ADC2).

```
Config Adc = Single , Prescaler = Auto
Start Adc
U = Getadc(2)
```



Bleibt das Problem der Anzeige. Es gibt bereits mehrere Lösungen, eine davon ist die Vierbit-Anzeige mit zwei LEDs. Eine LED steht im Normalfall für ein Bit. Zwei Bit pro LED sind machbar, wenn man bestimmte Blinkmuster zusätzlich verwendet: 0 = Aus, 1 = kurze Lichtblitze, 2 = langes Blinken, 3 = An. Insgesamt hat man jetzt vier Bit an zwei LEDs und kann Zahlen im Bereich 0...15 darstellen. Das Ablesen erfordert etwas Übung, weil man praktisch eine Binärzahl liest. Wer das Verfahren vorab auf seine Praxistauglichkeit testen will kann z.B. den Sparrow Up/Down-Counter ausprobieren.

Für das Voltmeter bedeutet die Beschränkung auf 15 Ausgabestufen eine Auflösung von 0,2 V und eine Endspannung von 3 V bei entsprechender Betriebsspannung. Genau genommen muss die Betriebsspannung 3,2 V betragen, was bei zwei ganz frischen Alkalizellen gerade passt. Die Messspannung kann direkt am Anschluss B4 zugeführt werden, besser ist aber ein Schutzwiderstand von 10 kΩ in Reihe zum Messkabel.

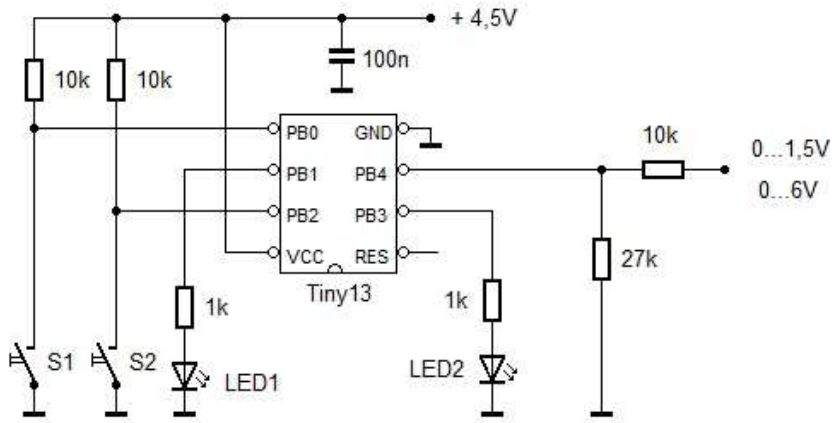
Das folgende Programm zeigt eine erste mögliche Lösung. Der Messwert U wird durch 64 geteilt um die Auflösung auf vier Bit zu reduzieren. In N wird dann ein Wert im Bereich 0 bis 15 weiter verarbeitet und zur Anzeige gebracht. Trotz der geringen Auflösung der Anzeige hat das Messgerät einen praktischen Nutzen. Man kann es nämlich nach etwas Übung aus dem Augenwinkel ablesen, während man auf das Messobjekt und die Messkabel schaut.

```
'ATtiny13 Sparrow Spannung 0...15
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Dim U As Word
Dim N As Byte
Dim H As Byte
Dim L As Byte

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000001010

Config Adc = Single , Prescaler = Auto
Start Adc

Do
  U = Getadc(2)
  U = U / 64
  N = U
  L = N And 3
  H = N
  Shift H , Right , 2
  H = H And 3
  If L = 0 Then Led1 = 0 Else Led1 = 1
  If H = 0 Then Led2 = 0 Else Led2 = 1
  Waitms 50
  If L < 2 Then Led1 = 0
  If H < 2 Then Led2 = 0
  Waitms 100
  If L < 3 Then Led1 = 0
  If H < 3 Then Led2 = 0
```

```
  Waitms 100
Loop

End
```

6.4 Messbereichsumschaltung

Das AD-Wandler des Tiny13 kann wahlweise die interne Referenz von 1,1 V verwenden oder die Betriebsspannung VCC. Mit einem einfachen Spannungsteiler am Eingang kommt man daher auf einen Messbereich von 1,5 V oder bei passender Betriebsspannung von 4,5 V auf einen Messbereich von ca. 6 V. Für mehr Genauigkeit müsste man noch ein Poti einbauen, aber für einfache Messungen und zur Erläuterung des Messprinzips reichen zwei Festwiderstände. Damit hat man zugleich einen Überspannungsschutz. Eine zu große oder eine negative Eingangsspannung würde durch die internen Schutzdioden am Eingang begrenzt.



Für die Umschaltung kann man die beiden Taster S1 (kleiner Bereich) und S2 (großer Bereich) verwenden. Dabei wird jeweils der AD-Wandler neu konfiguriert. Zur Verdeutlichung der Umschaltung soll während des Tastendrucks jeweils eine der beiden LEDs leuchten.

http://tiny.systems/categorie/cheepit/LEDVoltmeter.html

```
'ATtiny13 Sparrow Spannung 1,1V/VCC
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Dim U As Word
Dim N As Byte
Dim H As Byte
Dim L As Byte

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000001010
```

```
Config Adc = Single , Prescaler = Auto
Start Adc

Do
  If S2 = 0 Then
    Config Adc = Single , Prescaler = Auto , Reference = Internal
    Start Adc
    Led1 = 1
    Led2 = 0
    Waitms 300
    Do
    Loop Until S2 = 1
    Waitms 10
    Led1 = 0
  End If
  If S1 = 0 Then
    Config Adc = Single , Prescaler = Auto , Reference = Avcc
    Start Adc
    Led2 = 1
    Led1 = 0
    Waitms 300
    Do
    Loop Until S1 = 1
    Waitms 10
    Led2 = 0
  End If
  U = Getadc(2)
  U = U / 64
  N = U
  L = N And 3
  H = N
  Shift H , Right , 2
  H = H And 3
  If L = 0 Then Led1 = 0 Else Led1 = 1
  If H = 0 Then Led2 = 0 Else Led2 = 1
  Waitms 50
  If L < 2 Then Led1 = 0
  If H < 2 Then Led2 = 0
  Waitms 100
  If L < 3 Then Led1 = 0
  If H < 3 Then Led2 = 0
  Waitms 100
Loop
End
```

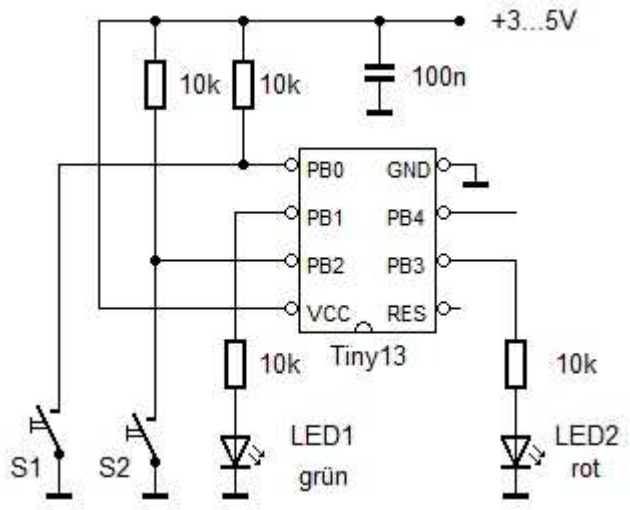
Die Umschaltung der Messbereiche arbeitet wie ein RS-Flipflop. Damit hat man auch einen Berührungspunkt mit der digitalen Elektronik (vgl. Kap 8.2). Aber auch physikalische und technische Aspekte sind enthalten (Spannungsteiler, AD-Wandler, Binärzahlen), sodass das Gerät in verschiedensten Zusammenhängen im Unterricht eingesetzt werden kann. Und nicht zuletzt ist es ein kleines und vielseitiges Messgerät für die verschiedensten Anwendungen von der Batterieüberwachung bis zu Messwertanzeige unterschiedlicher physikalischer Größen.

7 Sparrow-Peripherie und Sensoren

Der Sparrow ist ein komplettes kleines Mikrocontroller-System, das in ganz unterschiedlichen Hardware-Zusammenhängen verwendet werden kann. Besonders interessante Anwendungen ergeben sich mit Sensoren. Der Sparrow kann auf Reize aus der Umwelt reagieren und sinnvolle Dinge ausführen.

7.1 Sparrow_TouchB4

Wenn man B4 berührt geht die rote LED für fünf Sekunden an. Es reicht einen Draht anzuschließen, den man nur außen an seiner Isolierung berührt. Oder man verwendet eine isolierte Metallfläche als kapazitiven Näherungssensor. Das Programm wertet die geringe Kapazitätsänderung am Port B4 aus. B4 wird während der Messung über den hochohmigen Pullup aufgeladen. Sofort nach dem Einschalten des Pullup wird der Eingangszustand des Pins abgefragt.



Die Frage ist, schafft der Pullup eine so schnelle Aufladung, dass schon ein 1-Pegel gelesen wird? Das hängt von der Geschwindigkeit der Abfrage und der Kapazität am Port ab. Wenn hier ein Draht angeschossen ist und man diesen außen an der Isolierung berührt steigt die Kapazität etwas, die Aufladung wird langsamer. Normalerweise reicht das aber noch nicht für einen Null-Zustand beim Lesen. Deshalb wird hier der ganze Controller auf 2,4 MHz hochgetaktet. Damit ist ein eindeutiger Unterschied messbar.

http://tiny.systems/categorie/cheepit/TouchSensor.html

```
'Sparrow_TouchB4.bas
$regfile = "attiny13.dat"
$crystal = 2400000
$hwstack = 8
$swstack = 4
$framesize = 4

Dim D As Byte

  Clkpr = 128              '2,4 MHz
  Clkpr = 2
  Clkpr = 2
  nop
```

```
  nop
  nop
Do
  Ddrb.4 = 1
  Portb.4 = 0
  Ddrb.3 = 0
  Portb.3 = 0
  Ddrb.4 = 0
  Portb.4 = 1
  D = Pinb.4
  Portb.4 = 0
  Ddrb.4 = 1
  If D = 0 Then
    Portb.3 = 1
    Ddrb.3 = 1
    Wait 5
    Portb.3 = 0
    Ddrb.3 = 0
    Waitms 2
  End If
Loop
```

7.2 Sparrow_LichtAlarm

Diese Alarmanlage wertet kleinste Änderungen der Helligkeit aus. Da ist kein direkter Schattenwurf erforderlich, es reicht schon, dass jemand sich irgendwo an einem Fenster vorbeischleicht, eine Tür öffnet oder eine Taschenlampe benutzt. Bei jeder schnellen Änderung der Helligkeit geht für einige Sekunden die grüne LED an. Langsame Änderungen werden dagegen toleriert.

Üblicherweise setzt man für solche Aufgaben einen Fotowiderstand oder einen Fototransistor ein. Hier jedoch wird eine LED auf dem Sparrow zugleich als Lichtsensor verwendet. Eine LED ist zugleich eine Fotodiode und kann wahlweise in Sperrrichtung oder in Durchlassrichtung als Fotoelement verwendet werden.

Das Programm wertet die Spannung an der roten LED aus und mittelt jeweils 64 Messwerte um eventuelles Lampenflackern zu eliminieren. Die schon gemittelten Messwerte werden dann noch einmal wie mit einem Tiefpassfilter geglättet. Wenn sich die tatsächliche Helligkeit nur langsam ändert sind der geglättete und der aktuelle Messwert gleich. Aber bei plötzlichen Änderungen zeigt sich ein Unterschied, der den Alarm auslöst.

http://tiny.systems/categorie/cheepit/LichtAlarm.html

```
'Sparrow_LichtAlarm.bas
'K an PB4, A über 1 k an PB3
'LED1 leuchtet bei schzneller Änderung der Helligkeit

$regfile = "attiny13.dat"
$crystal = 1200000
Baud = 9600
$hwstack = 8
$swstack = 4
$framesize = 4

Dim D As Word
```

```
Dim E As Word
Dim M As Word
Dim N As Byte
Dim I As Byte
Dim T As Word
Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2

Config Portb = &B00010010
'Config Timer0 = Pwm , Prescale = 8 , Compare B Pwm = Clear Up
Config Adc = Single , Prescaler = Auto
Start Adc

Do
  D = M
  Shift D , Right , 4
  M = M - D                    'Tiefpass-Mittelwert * (1 -1 / 16)
  D = 0
  For N = 1 To 64
     D = D + Getadc(3)
     Waitus 160
  Next N
  Shift D , Right , 6          'Gemittelter Messwert
  M = M + D                    'Tiefpass-Mittelwert + Messwert
  E = M
  Shift E , Right , 4
'  Pwm0b = D                   'Ausgabe nur zum Debuggen
  If D > E Then                'Schnelle Änderungen auswerten
    D = D - E
    If D > 2 Then T = 100      'Alarm-Haltezeit
  Else
    E = E - D
    If E > 2 Then T = 100      'Alarm-Haltezeit
  End If
  If T > 0 Then T = T - 1
  If T > 1 Then
    Led1 = 1
  Else
    Led1 = 0
  End If
Loop
End
```

7.3 LED-Leuchte mit IR-Fernbedienung

Da gab es eine LED-Lichterkette mit Batteriefach für eine indirekte Beleuchtung hinter dem Fernseher mit einem großen Nachteil: Den Fernseher kann man mit der Fernbedienung abschalten, aber die Lichterkette nicht. Eine Fernbedienung für die Lichterkette wäre eine feine Sache. Aber da steht auch ein Radio, ebenfalls mit Fernbedienung. Da sind es schon zwei, noch eine weitere Fernbedienung würde stören.

Ideal wäre es, wenn jede beliebige Infrarot-Fernbedienung auch die LED-Lichter ein- und ausschalten könnte. Und das geht! Auf den Sparrow wird ein üblicher IR-Empfänger gelötet. Das Empfangssignal geht an B0, parallel zu S1. Da es ebenfalls aktiv low schaltet passt beides zusammen. Alles wird nun über einen Li-Akku aus einem defekten Smartphone betrieben.

Die Software soll einen langen Tastendruck auf der Infrarot-Fernbedienung auswerten und den LED-Ausgang dann umschalten. Normalerweise wird jede Taste nur kurz gedrückt, darauf reagiert der Sparrow nicht. Man kann aber eine Taste lang drücken, die für den Fernseher oder das Radio gerade keine Funktion hat. Beim Radio verwende ich z.B. eine Taste für den CD-Player, der gerade nicht aktiv ist. Oder der Fernseher ist gerade nicht an, da kann ich ja die Kanal-1-Taste verwenden. Wie lange es genau dauert hängt von der jeweiligen Fernbedienung und ihrem Protokoll ab. Das Programm wartet nämlich einfach nur auf 500 irgendwelche Impulse. Das kommt dann ungefähr auf eine Sekunde. Man drückt einfach so lange, bis die angeschlossene LED an- oder ausgeht. Zusätzlich kann man die LEDs auch mit der Taste S2 schalten. Man kennt das ja, manchmal ist die Fernbedinung gerade unauffindbar. Die Taktrate ist auf 300 kHz herabgesetzt, damit das Gerät auch im Aus-Zustand möglichst wenig Strom braucht. Zusammen mit dem IR-Empfänger kommt man nun auf etwa 1 mA.

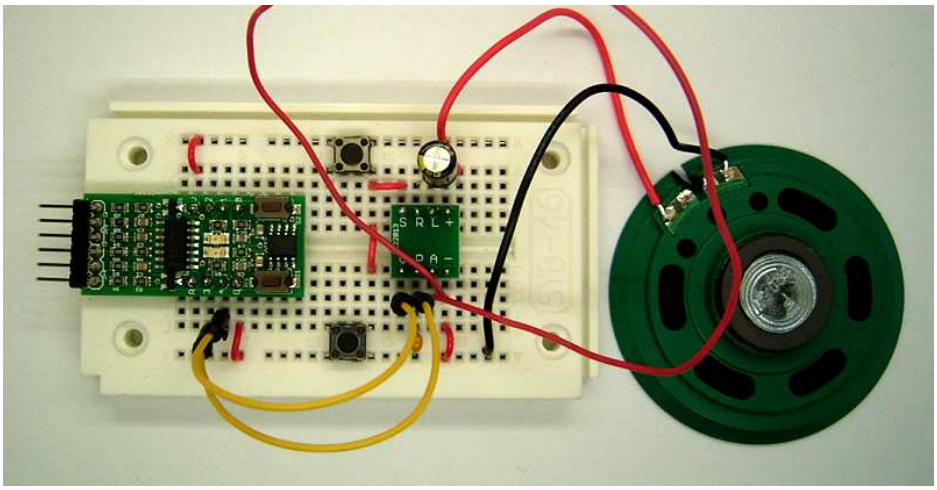
http://tiny.systems/categorie/cheepit/InfrarotLED

```
'Sparrow_IR_Toggle.bas
'TSOP IR-RX an B0
$regfile = "attiny13.dat"
$crystal = 1200000
$hwstack = 8
$swstack = 4
$framesize = 4

Led1 Alias Portb.1
Led2 Alias Portb.3
S1 Alias Pinb.0
S2 Alias Pinb.2
Config Portb = &B000011010

Dim T1 As Word
Dim T2 As Word
Dim N As Word

Clkpr = 128                     '300 kHz
```

```
Clkpr = 5
Clkpr = 5

Led1 = 1

Do
  T1 = 0
  Do
    T1 = T1 + 1
    If T1 > 1000 Then N = 0     '>100 ms Pause = Rücksetzen
    Waitus 100
    If S2 = 0 Then              'Alternative Bedienung mit S2
      Toggle Led1
      Waitms 50
      Do
      Loop Until S2 = 1
      Waitms 50
    End If
  Loop Until S1 = 0             'IR-Puls Start
  Do
    Waitus 100
  Loop Until S1 = 1             'IR-Impuld Ende
  N = N + 1                     'IR-Impulse zählen
  If N > 500 Then               '500 Impulse = lang gesdrückt
    Toggle Led1                 'AUS AN AUS AN
    N = 0
  End If
Loop
```

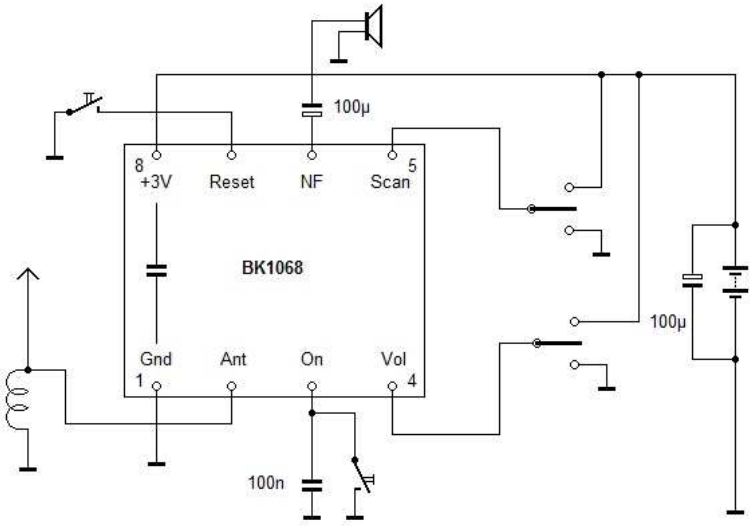
7.4 Das Sparrow-Schlafradio

Das Schlafradio basiert auf dem BK1068-Breakout-Board (http://www.elexs.de/FMradio1.html). Senderabstimmung, Lautstärkeverstellung, Power-Down. NF-Endverstärker, all das ist ja schon vorhanden, sodass ein Radio mit dieser Platine sehr einfach zu bauen ist. Wenn dann noch alles mit einem Mikrocontroller gesteuert wird, hat man ganz andere Möglichkeiten. Hier geht es um das gute Einschlafen. Das Radio soll eine halbe Stunde lang laufen, dann immer leiser werden und schließlich abschalten. So kann es beim Übergang in die Nacht helfen.

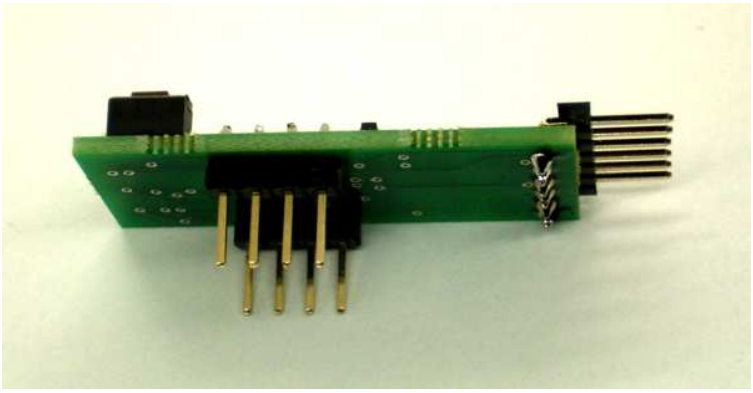


Dass gerade der Sparrow für dieses Projekt ausgesucht wurde hat seinen Sinn. Dieser kleine Controller kann nämlich problemlos mit auf die Platine gesteckt werden. Die wichtigsten Bedienelemente sind schon vorhanden. Man muss nur einmal das Programm aufspielen, dann geht es los.

Das Radio selbst hat zwei Tasten für die Senderwahl. Aber die Lautstärke (Vol) wird nun vom Mikrocontroller bedient, genau wie die Power-Down-Funktion (On).

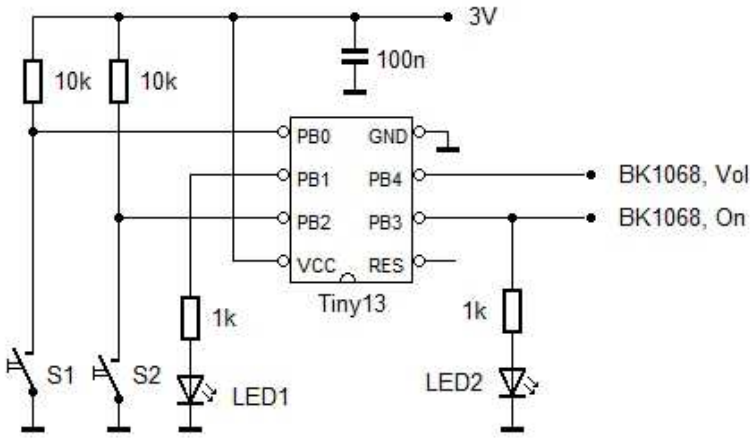


Damit beide Platinen zusammen auf ein Steckboard passen muss der Sparrow zwei Pfostenleisten mit insgesamt acht Anschlüssen bekommen. Alle Anschlüsse des Tiny13A sind dann auf dem Steckboard zugänglich.

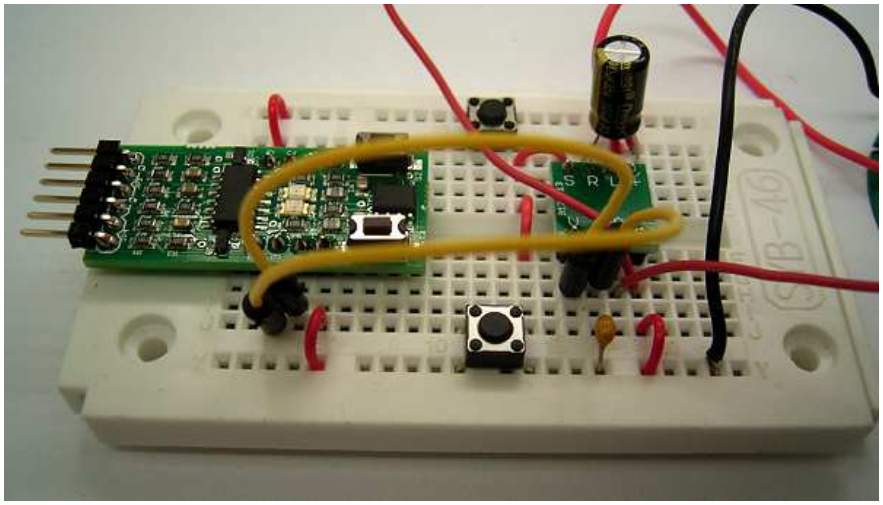


Beide Platinen liegen an derselben Betriebsspannung von 3 V und sind nur über zwei weitere Leitungen verbunden. Der Vol-Eingang ist ein Tristate-Eingang, der im Ruhezustand eine mittlere Spannung von 1,5 V annimmt. Zieht man die Spannung nach oben wird es lauter, zieht man sie nach Masse wird es leiser. Der Port B4 muss daher im Ruhezustand ein hochohmiger Eingang sein, der nur bei einer Veränderung der Lautstärke in den Ausgangszustand umgeschaltet wird und dabei entweder High oder Low wird. Gleichzeitig ist der Anschluss ein AD-Eingang und kann daher messen, ob der Radiochip sich im aktiven Zustand mit ca. 1,5 V befindet. Im Power-Down-Zustand dagegen steigt die Spannung auf 3 V.

Über den Anschluss PB3 wird die Power-Down-Funktion mit einem kurzen Low-Impuls bedient. Praktisch kann das Radio über diesen Pin ein- und ausgeschaltet werden, wobei der gerade vorhandene Zustand am Vol-Pin abgelesen werden kann.



Die Bedienung ist einfach. S2 schaltet das Radio ein und erhöht die Lautstärke in insgesamt 16 Stufen. S1 dagegen reduziert die Lautstärke und schaltet das Radio schließlich ganz ab. Den Betriebszustand des Radios zeigt der Sparrow über die LED1 (grün) an. Die rote LED ist immer an, wird jedoch im Aus-Zustand des Radios in der Helligkeit reduziert, damit beide Platinen zusammen im Ruhezustand nicht mehr als 1 mA brauchen.



Das Steuerprogramm wurde in Bascom geschrieben und erklärt sich weitgehend selbst. Es gibt Prozeduren für die Grundfunktionen Lauter, Leiser, An und Aus. Die Laufzeit wird in T gemessen und beim Druck auf die Lauter-Taste zurückgesetzt. Im Aus-Zustand versetzt das Programm den Port B3 in den hochohmigen Zustand mit Pullup. Der Strom für die LED2 fließt dann über den Pullup-Widerstand im Tiny13 und über den Pullup am On-Pin des Radiochips. Das Ergebnis ist ein schwaches Standby-Leuchten der roten LED, sodass man das Radio auch im Dunkeln leicht findet. Falls man doch mal wieder aufwacht...

Links und Literaturempfehlungen

http://tiny.systems/categorie/cheepit

http://www.elektronik-labor.de/AVR/Sparrow/Cheepit.html

http://www.elexs.de/Sparrow1.html

www.ak-modul-bus.de/stat/entwicklungssystem_sparrow.html

http://www.elektronik-labor.de/Lernpakete/TPS/TPS0.html

Claus Kühnel, Programmieren der AVR RISC Mikrocontroller mit BASCOM-AVR 2010

Burkhard Kainka, Basiskurs BASCOM-AVR , Elektor 2011