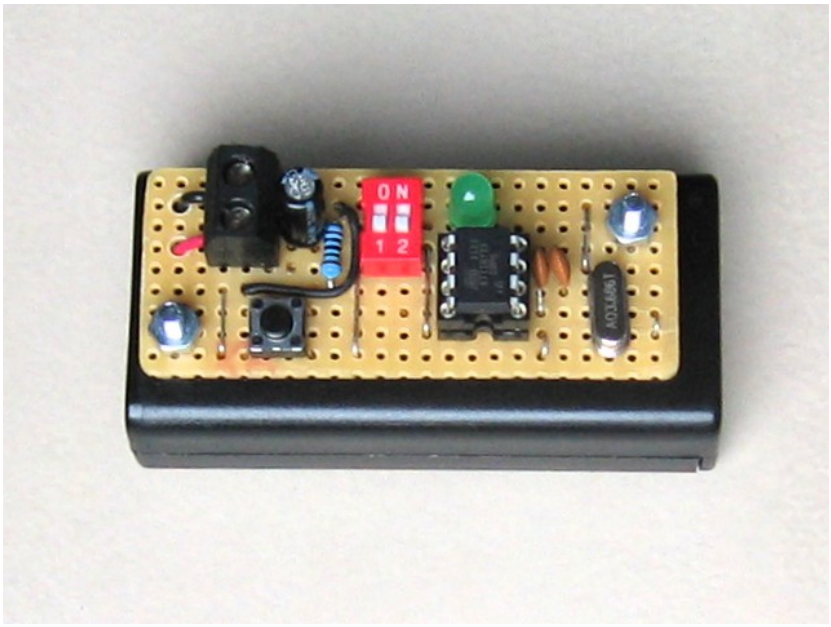
Pillenwarner

Ralf Beesner

19. Februar 2012



1 Überblick

Im Moment muss ich irgendwelche Tabletten schlucken, und da ich das leicht mal vergesse, habe ich mir eine kleine AtTiny-Schaltung gebaut, die mich jeden Morgen solange anblinkt, bis ich das Signal per Tastendruck quittiere.

Die Schaltung und die Software sind sehr einfach gehalten. Im Gegensatz zu einer Lösung mit analogen Timer- und Logik-ICs kann man die Zeiten in einem weiten Bereich sehr einfach per Software ändern und auch sehr lange Zykluszeiten im Wochen- oder gar Monatsbereich bis hin zum jährlichen Hochzeitstag ;-) realisieren - mit Quarzgenauigkeit und so geringem Stromverbrauch, dass die Schaltung aus Batterien betrieben werden kann.

Der Pillenwarner lässt sich also leicht für andere Zwecke abwandeln.

Da noch ein Pin am Mikrocontroller frei war, habe ich eine Wahl zwischen zwei Zykluszeiten vorgesehen.

2 Bedienung

Der Pillenwarner hat 3 Bedienelemente: den Quittungstaster, einen Wahlschalter (der im vorliegenden Beispiel zwischen 24h-Modus und 12h-Modus umschaltet) und den Reset-Schalter, der die Uhr startet.

Ausgabe-Element ist eine LED. Als Lebenszeichen blitzt sie alle 8 Sekunden für 20 ms kurz auf (Heartbeat).

Möchte man ein mal pro Tag (also alle 24 Stunden) an die Tabletten-Einnahme (oder was auch immer) erinnert werden, stellt man Schalter 2 auf "OFF", für zweimaliges Erinnern pro Tag (also im 12h-Rhythmus) auf "ON".

Kurzes Betätigen von Schalter 1 löst einen Reset aus; nach Ende des Resets läuft die Uhr los, allerdings startet sie nicht bei 0, sondern mit 7200 Sekunden (2h) Vorlauf, damit sie künftig immer zwei Stunden vor der Tabletten- Einnahme aktiv wird (falls man mal eher aufsteht).

Überschreitet der Zähler 24h (bzw. 12h), blinkt die LED im Sekundentakt für jeweils 100 ms. Durch Druck auf den Quittungstaster fällt sie wieder in den Heartbeat-Modus zurück.

Vergisst man das Quittieren (und vermutlich auch die Tabletten-Einnahme), geht die LED nach einem weiteren Ablauf der Zykluszeit in Doppelblinken über (zwei mal 100ms pro Sekunde) und mahnt auf diese Weise die "Disziplinlosigkeit" an.

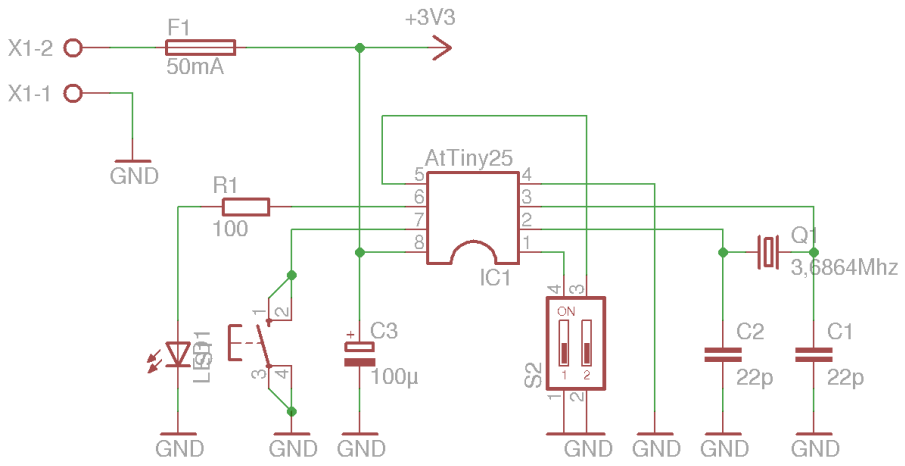


Abbildung 1: Schaltbild

Hardware

Die Schaltung besteht aus einem AtTiny 25 mit 3,6864 MHz-Quarz an PB3 und PB4. An PB1 liegt eine LED mit Vorwiderstand, an PB2 der Quittungstaster. An PB0 liegt der Schalter, mit dem man zwischen den beiden Zykluszeiten wählen kann. Ein weiterer Schalter liegt am Reset- Eingang.

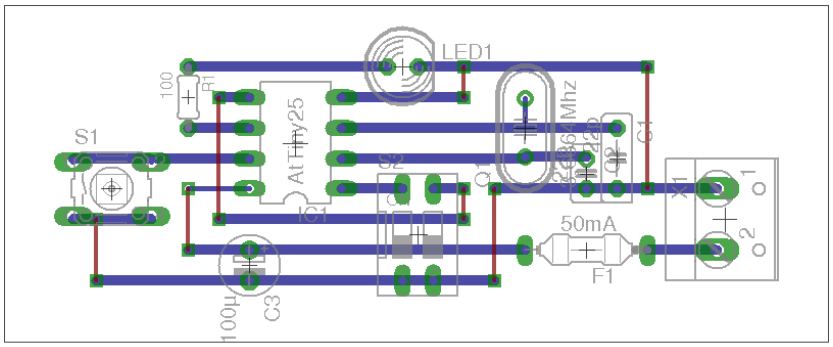


Abbildung 2: Layout

Software

Der AtTiny wird auf Quarzbetrieb gefust. Der Takt wird zur Laufzeit des Programms mit dem Befehl CLKPR durch 64 geteilt, der Controllertakt beträgt also nur 57600 Hz.

Timer 0 erzeugt daraus ein Sekundensignal. Sein Vorteiler wird dazu auf 256 gestellt, und der Timer läuft jeweils nach 225 Schritten über und generiert jeweils einen Interrupt. In der Interruptroutine wird die Sekundenvariable T inkrementiert.

24 Stunden sind 86400 Sekunden, 12 Stunden sind 43200 Sekunden.

Die Hauptschleife prüft ein mal pro Durchlauf, ob 86400 (bzw. 43200) Sekunden ereicht wurden. Ist dies der Fall, wird T auf 0 zurückgesetzt und die Alarm-Variable A inkrementiert.

Anschließend wird geprüft, ob $A = 1$ ist (dann wird für 100 ms geblinkt) oder ob $A > 1$ ist (dann wird doppelt geblinkt). Danach geht der Controller in den Idle-Modus (aus dem er durch den nächsten Sekunden-Interrupt geweckt wird).

Ein Druck auf den Quittungstaster setzt die Alarmvariable A auf 0 zurück.

Stromverbrauch

Da der Quarzoszillator ständig laufen muss, darf man den Mikrocontroller nicht in Powerdown, sondern nur in den Idle- Modus schicken. Bei 3 V Betriebsspan-

nung (2 Mignonzellen) verbraucht er aber nur 100 μA, also knapp 900 mAh pro Jahr.

Die LED verbraucht max. 10 mA, während sie leuchtet; bei einem Tastverhältnis von 100 ms zu 1s also im Mittel 1mA. Ist sie durchschnittlich 3h pro Tag am Blinken, sind das weitere 1095 mAh pro Jahr.

Die Schaltung sollte also mit zwei Mignonzellen deutlich länger als ein Jahr laufen.

3 Flashen des AtTiny

Wer nur die Platine aus dem LP Mikrocontroller als Programmiergerät zur Verfügung hat, kann zwar das Programm mit Burkhard Kainkas LPMikroISP.exe in den Mikrocontroller flashen, aber zusätzlich müssen die Fusebytes verändert werden, um den Controller in den Quarzbetrieb umzuschalten. Dies darf jedoch nicht mit LPMikroISP.exe erfolgen!

Unter www.elektroniklabor.de/AVR/AVRdude.html habe ich beschrieben, wie man die Hardware des LP Mikrocontroller mit dem Kommandozeilentool `avrdude.exe` programmieren kann.

Das erforderliche Fusebyte lautet: `lfuse=0xfd`.

Der komplette Aufruf lautet:

```
avrdude.exe -p t25 -c burkhard -P com1 -U lfuse:w:0xfd:m
```