Binäruhr

Karsten Müller
email@kmkcl.de

10. Januar 2016

1 Vorwort

Eine Binäruhr gibt die Zeit genau so wieder, wie sie von digitaler Elektronik verarbeitet wird. Jede elektronische Uhr basiert auf dem Binärsystem und nutzt die gleiche, oder eine sehr ähnliche Zählweise wie diese Uhr. Im Gegensatz zu anderen Uhren, erfolgt bei der Binäruhr die Ausgabe jedoch ohne Umwandlung in ein anderes Format. Stunden, Minuten und Sekunden werden über leuchtende, oder nicht leuchtende LEDs jeweils als Binärzahl wiedergegeben.

Diese elementare Darstellung von Zahlen besitzt vor allem für Technikbegeisterte einen besonderen Reiz. Nicht zuletzt basiert nahezu die komplette Computerverarbeitung auf Binärzahlen und digitalen Zuständen. Es gibt bereits einige Bausätze und Selbstbauprojekte, die Motivation für einen Selbstbau lag jedoch in der völligen Gestaltungsfreiheit. Mir war es dabei zum Beispiel wichtig, die vollen Binärzahlen bis 60 darzustellen und nicht die 10er und 1er Stellen getrennt.

Diese kleine Anleitung ist für all diejenigen, die eine Inspiration suchen, denen genau meine Bauweise gefällt oder die sich eine eigene Konzeption nicht zutrauen. Im Laufe der Zeit habe ich Schaltungsentwurf und Programm mehrmals überarbeitet. Gleichwohl wird es sicher noch elegantere Umsetzungen geben.

Die Anleitung beginnt mit den Überlegungen zum technischen Design und Anmerkungen zum Nachbauen. Weiterhin ist der Schaltplan und die Bedienungsanleitung der Uhr enthalten. Abschließend werden ein paar informative Fakten erwähnt, die sich eher an interessierte Elektronik-Neulinge richten. Der Abschnitt entstand als Geschenkbeilage selbstgebauter Binäruhren.

2 Wahl der Bauteile

Die Uhr besteht zwar nur aus wenigen Bauteilen, dennoch sind einige Überlegungen dazu hilfreich. So ließe sich theoretisch eine Binäruhr auch recht einfach mit reinen Logik-ICs (Zähler) bauen. Der Verdrahtungsaufwand wäre sehr viel größer, der Aufwand für die Software würde jedoch entfallen. Die Mikrocontroller-Lösung bietet aber auch Vorteile in der einfachen Helligkeitssteuerung und einer guten Bedienbarkeit über zwei Taster.

2.1 Zeitgeber

Die einfachste und kostengünstigste Methode besteht in der Verwendung eines Quarzes für den Mikrocontroller. Mit einer guten Kalibrierung lassen sich zum Teil auch höhere Genauigkeiten erzielen als mit anderen Methoden. Ich habe mich jedoch für einen externen Zeitgeber entschieden, einen DS1307. Dieser IC ist eine Uhr, die über den I^2C Bus angeschlossen wird. Er benötigt lediglich einen Quarz und kann den Strom aus einer Pufferbatterie beziehen, wenn die primäre Stromversorgung aus ist, eine Eigenschaft die nicht zwingend notwendig, aber durchaus praktisch ist. Die Verwendung eines externen Zeitgebers vereinfacht die Programmierung, da die Kalibrierung eines genauen Sekundentaktes entfällt. Der Schaltkreis ist zum Teil recht günstig bei ebay zu bekommen, so fand ich ein Angebot mit 10 Stück für 3,50€. Dadurch überwogen die Vorteile deutlich den Nachteil der Bauteilkosten. Ein Kompromiss könnte die Verwendung eines Quarzoszillators sein. Ungenauigkeiten durch nicht exakt abgestimmte Bauteile (Kapazitäten von Quarz, Kondensatoren und Microcontroller) würden dabei vermieden und die Notwendigkeit des Kalibrierens entfallen.

Für eine erhöhte Genauigkeit eignet sich der Uhren-IC DS3231. Die Genauigkeit bei Zimmertemperatur wird mit $\leq$ 2ppm angegeben. Der IC ist mit der aktuellen Software kompatibel.

2.2 Mikrocontroller

Als Mikrocontroller hat sich der Attiny2313 der Firma Atmel als ausreichend erwiesen. Er ist sowohl von dem Programm, als auch von den Ein- und Ausgängen ziemlich ausgelastet, erfüllt aber seinen Job vollkommen. Der Mikrocontroller läuft bei mir mit dem internen 8MHz Takt. Da der Sekundentakt für die Uhr von dem DS1307 kommt, besteht kein Anspruch auf Genauigkeit. Für eine vollautomatische Helligkeitsregelung wird ein Controller mit AD-Wandler benötigt. Dann ließe sich sehr einfach mit einem Fototransistor die Raumhelligkeit erfassen und die Helligkeit der LEDs entsprechend regeln. Um dennoch den unterschiedlichen Helligkeitsbedarf für Tag und Nacht gerecht zu werden unterstützt die Uhr einen Tages- und einen Nachtmodus, der zeitgesteuert oder manuell umgeschaltet werden kann.

2.3 LEDs

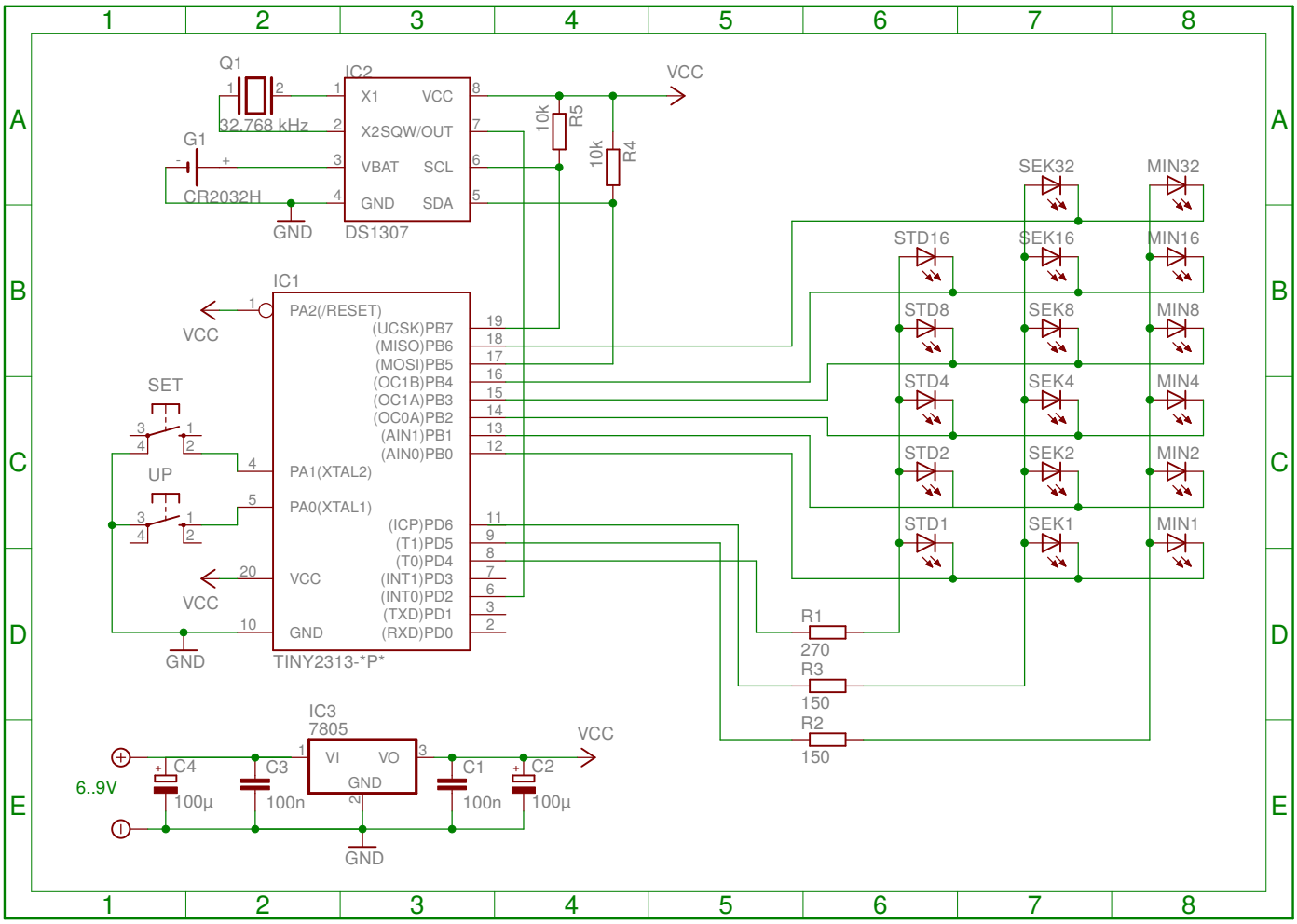
Bei der Auswahl der LEDs ist auf mehre Dinge zu achten. Das wichtigste ist die Helligkeit. Nur wenn die LEDs bei einer kleinen Stromaufnahme (ca. 11mA) schon sehr hell sind, lassen sie sich gut mit dem Multiplexing ansteuern, ohne dass stromverstärkende Transistoren benötigt werden. Konkave LEDs erhöhen die Erkennbarkeit, besonders aus flachen Winkeln und sorgen für ein gleichmäßiges Leuchten. Andere superhelle LEDs hingegen erscheinen beim direkten betrachten mitunter viel zu hell. Die 5mm Bauweise erleichtert den Aufbau auf einer Lochrasterplatine. Im Gegensatz zu vielen 3mm Varianten liegen sie flach auf der Platine auf und müssen nicht aufwändig ausgerichtet werden. Meine Wahl fiel somit auf superhelle 5mm konkav LEDs. Die LEDs sind in einer 3x6 Matrix verschaltet, wobei jede Zeile eine gemeinsame Kathode hat. Dies ist sinnvoll, da ein Port-Pin bei GND-Potential einen höheren Strom verträgt. Angesteuert wird Zeilenweise, somit arbeitet der Mikrocontroller immer innerhalb der Spezifikationen für die Stromaufnahme. Weiterhin kann bei dieser Verschaltung die Stromaufnahme von 3 LEDs nie signifikant überschritten werden, da der Strom hauptsächlich durch die Widerstände begrenzt wird.

2.4 Spannungsversorgung

Die Verwendung des 7805 als Spannungsregler erlaubt eine variable Eingangsspannung von 6-9V. Mit einem kleinen Vorwiderstand oder entsprechender Kühlung sind sogar 12V möglich. Damit wird auch die Verwendung einfacher Gleichstromnetzteile mit schlechter Spannungsstabilisierung möglich, oder wie in meinem Fall der Anschluss an das 12V Netzteil für die RGB-LED Beleuchtung. Ist eine gute 5V Spannung vorhanden, etwa aus einem USB-Ladegerät, kann auch darauf verzichtet werden.

3 Nachbau

3.1 Schaltplan



3.2 Stückliste Bauteile

Stück	Bauteil	Wert
1	IC1	Attiny2313 (Mikrocontroller)
1	IC2	DS1307 (I^2C Echtzeituhr)
1	IC3	7805 (Spannungsregler)
1	Q1	32.768 Hz (Quarz)
5	STD1, STD2, STD4, STD8, STD16	rote LEDs *(superhell, konkav)*
6	MIN1, MIN2, MIN4, MIN8, MIN16, MIN32	grüne LEDs *(superhell, konkav)*
6	SEK1, SEK2, SEK4, SEK8, SEK16, SEK32	blaue LEDs *(superhell, konkav)*
1	R1	270 Ω
2	R2, R3	150 Ω
2	R4, R5	10 kΩ
2	C1, C3	100 nF
2	C2, C4	100 μF
2	UP, DOWN/SET	Taster
1	G1	CR2032 Halterung + Batterie
1		IC-Sockel 20pol
1		IC-Sockel 8pol

3.3 Hinweise zu den Bauteilen

Die Widerstände `R1, R2, R3` dienen als Vorwiderstände für die LEDs. Sie sollten so an die verwendeten LEDs angepasst werden, dass ein Strom von 13mA fließen kann. Ein Mikrocontroller-Port ist auf 40mA ausgelegt, da 3 LEDs zur gleichen Zeit angesteuert werden, ergeben sich die 13mA. Eine leichte Überschreitung ist sicher möglich, jedoch kann es dann dazu kommen, dass die LEDs unterschiedlich hell leuchten. Die LEDs sollten dementsprechend hell sein, da jede LED maximal für 1/6 der Zeit leuchten kann. Ich habe mich für konkave LEDs entschieden, deren Lichtstärke mit ca. 150mcd angegeben wurde.

Bei dem Uhrenbaustein DS1307 sollte der Quarz möglichst nah an dem IC sitzen und die Leitungen möglichst kurz sein. Desweiteren sollten sich unter dem Quarz keine weiteren Leitungen befinden. Diese Maßnahmen wirken sich positiv auf die Genauigkeit aus.

Für eine sehr hohe Genauigkeit lässt sich der IC DS1307 durch den wesentlich genaueren DS3231 ersetzen. Die aktuelle Software (ab 10.1.2016) unterstützt diesen. Den IC gibt es kostengünstig als fertiges Modul. Der Anschluss erfolgt wie beim DS1307, insbesondere muss `SQW` mit dem Mikrocontroller-Pin 6 (PD2) verbunden werden.

3.4 Programmierung des Mikrocontrollers

Zum Programmieren des Mikrocontrollers wird ein Programmieradapter und ein passendes PC-Programm benötigt. Unter Linux ist dies zum Beispiel mit dem Programmieradapter *USBasp* und dem Programm *avrdude* möglich. Unter Windows lässt sich unter anderem der *MySmartUSB Light* sehr gut verwenden, da keine Installation notwendig ist und Programm

beiliegt. Es gibt allerdings noch sehr viel mehr Auswahl an Programmieradaptern und Programmen. Soll die Binäruhr ein einzelnes Projekt werden, finden sich im Internet auch Angebote, programmierte Mikrocontroller zu bekommen. Auch ich helfe gerne weiter und programmiere einen Controller gegen einen kleinen Unkostenbeitrag.

Der Mikrocontroller kann mit dem beiliegendem Programm (http://kmkcl.de/binuhr.html) programmiert werden. Neben einer fertig kompilierten Version, liegt auch der Quelltext vor. Damit der Mikrocontroller mit 8 MHz läuft, müssen die entsprechenden Fuses gesetzt sein. Dies geschieht nicht automatisch beim Übertragen des Programmes!

1. `lower Fuse: 0xE4` setzen (`make fuses`)
2. `binuhr.hex` in Mikrocontroller laden (`make program`)

*Die Befehle in Klammern funktionieren mit dem beigelegten Makefile. Voreingestellt ist das Programm **avrdude** und der Programmer **usbasp**.*

3.5 Vorschläge für den Nachbau

Die Anordnung mit den Sekunden in der Mitte sorgt für ein besseres Ablesen im Dunkeln: Durch die permanenten Veränderungen der leuchtenden LEDs werden schnell die Positionen und somit die zugehörigen Stellenwerte deutlich.

Blaue LEDs für die Sekunden haben den Vorteil, dass sie in der Nacht weniger stören, als rot oder grün, da das Auge für blau weniger empfindlich ist. Die Farben der Uhr sind übrigens nach ihren Wellenlängen sortiert. So sind Sekunden die kürzesten, angezeigten Zeiteinheiten und werden mit blau dargestellt (kürzeste verwendete Wellenlänge).

Als “Gehäuse ”hat sich ein Bilderrahmen als praktisch erwiesen. Dazu wird die Platine auf ein für Bilderrahmen übliches Maß gebracht (z.B. 150x100mm) und dann statt der Glasscheibe eingesetzt.

4 Bedienung der Uhr

4.1 Uhr stellen

1. Taste `DOWN/SET` lange drücken.
2. Mit den Tasten `UP` und `DOWN/SET` die **Stunden** einstellen.
3. Taste `DOWN/SET` lange drücken.
4. Mit den Tasten `UP` und `DOWN/SET` die **Minuten** einstellen.
5. Taste `DOWN/SET` lange drücken. Sekunden werden auf 0 gestellt und die Zeit gespeichert.

4.2 Helligkeit einstellen

4.2.1 Tag-/Nachtmodus

Die Uhr besitzt einen Tages- und einen Nachtmodus für den unterschiedlichen Helligkeitsbedarf. Für jeden Modus kann eine individuelle Helligkeit eingestellt werden. Eine automatische Umschaltung erfolgt um **8 Uhr** auf den Tagesmodus, um **22 Uhr** auf den Nachtmodus. Der Modus lässt sich jedoch auch jederzeit manuell umschalten. Zum programmieren ist immer der Tagesmodus aktiv.

- Wechsel Tag-/ Nachtmodus: Taste `DOWN/SET` kurz drücken.

4.2.2 Helligkeiten programmieren

Die Programmierung gilt für den **aktuellen Modus**. Wird die Uhr vom Strom getrennt, sind wieder die Standarteinstellungen aktiv.

1. Taste `UP` drücken.
2. Mit den Tasten `UP` und `DOWN/SET` Helligkeit einstellen.
3. Rückkehr zum Uhrenmodus erfolgt automatisch nach 1-2 Sekunden ohne Eingabe.

Die Anzeige zeigt dabei folgendes an:
linke Spalte:

1. aktuell Nachtmodus
2. aktuell Tagesmodus

mittlere und rechte Spalte:

- Helligkeitsstufe als Binärzahl. (von 1-9)

5 Informatives

5.1 Mikrocontroller

Mikrocontroller sind programmierbare Bauelemente, die eine Vielzahl von Funktionen übernehmen können. Die Besonderheit liegt in der umfangreichen Peripherie, die neben einem Prozessor auf dem Chip integriert ist. Neben den notwendigen Elementen zum Betreiben eines Prozessors, wie Programmspeicher (Flash) und Arbeitsspeicher (SRAM) ist oft auch Elektronik für Takterzeugung und verschiedene Ein- und Ausgänge, wie zum Beispiel Analog-Digital-Wandler, vorhanden. So wird der externe schaltungstechnische Aufwand auf ein Minimum reduziert. Programmiert werden die Mikrocontroller in den gleichen Programmiersprachen wie Computerprogramme mit dem PC. Die dazu nötigen Programmieradapter sind teilweise schon für wenige Euro zu kaufen oder zu bauen. Diese Eigenschaften machen Mikrocontroller besonders im Hobbybereich sehr beliebt. Hier kommt ein Mikrocontroller mit 2kB Programmspeicher der Firma Atmel in Anwendung.

5.2 Programm

Das Programm ist in C geschrieben. Wird der Mikrocontroller gestartet, werden die Einstellungen der Echtzeituhr geprüft und diese bei Bedarf konfiguriert. So ist im Auslieferungszustand die Echtzeituhr abgeschaltet und gibt keinen Sekundentakt ab. Die Uhrzeit wird über I^2C (siehe 5.3) ausgelesen und danach über den Sekundentakt des Uhren-ICs im Mikrocontroller syncron gehalten.

Das eigentliche Programm besteht aus drei Teilen. Es gibt einen Hauptteil und 2 Interrupts. Interrupts unterbrechen kurz das Hauptprogramm und führen ein Stück Code aus. Ein Interrupt sorgt beispielsweise dafür, dass die Variable *Sekunden* immer inkrementiert wird, wenn ein Impuls von dem Uhren-IC kommt.

Das Hauptprogramm besteht aus einer unendlichen **while**-Schleife. Die Grundstruktur der Schleife ist in Abbildung 1 dargestellt. Diese Struktur kann auch Zustandsautomat genannt werden. Ein Zustand (wie Uhrzeit anzeigen) bleibt so lange bestehen, bis eine bestimmte Bedingung eintritt, dann wird in einen anderen Zustand gewechselt.

Der zweite wesentliche Programmteil wird von einem Timer gesteuert. Der Timer ist ein Interrupt, der das Hauptprogramm in regelmäßigen Abständen unterbricht und eine spezielle Funktion aufruft. Hier wird das Hauptprogramm alle 250μs unterbrochen. Die Funktion sorgt zum einen für die Ansteuerung der LEDs (Multiplexing) und zum anderen prüft sie, ob eine Taste gedrückt wurde.

5.3 I^2C

I^2C ist eine spezielle Übertragungsart von Daten. Dabei umfasst die Spezifikation die physische Ebene (wie die Elektronik beschaffen sein muss) und die Protokollebene (wie die Übertragung abläuft). Da mit dieser Art mehr als zwei Bauteile parallel verbunden werden können, ist es ein Datenbus. I^2C wurde in den 80'er Jahren von Phillips entwickelt. Er kommt mit nur 2 Leitungen aus (Taktleitung und Datenleitung) und ist sehr flexibel einsetzbar. Ursprünglich war er nur für kurze Distanzen von wenigen Zentimetern gedacht, mit speziellen Bauteilen lassen sich aber auch mehrere Meter realisieren. Für den I2C Bus gibt es sehr viele Module, wie Sensoren, Echtzeituhren und Speicher.

```
while 1 :
 if modus == 0 :  #Uhrzeit anzeigen
   #...
    if obereTaste:
      modus = 1
    if untereTasteLang:
      modus = 2

 if modus == 1 : #Helligkeit stellen
   #...
    if Sekunden - letzteEingabe > 2 :
      modus = 1

 if modus == 2 : #Stunden stellen
   #...
    if untereTasteLang:
      modus = 3

 if modus == 3 : #Minuten stellen
   #...
    if untereTasteLang:
      modus = 1
```

Abbildung 1: Grundstruktur der **while**-Schleife als Pseudocode in Python Syntax

5.4 Multiplexing

Multiplexing bedeutet eigentlich, dass mehrere Signale über einen Kanal übertragen werden.

Vorallem bei Anzeigen mit LEDs wird Multiplexing verwendet, um nicht für jede LED eine eigene Leitung und einen eigenen Vorwiderstand zu brauchen. Dazu werden die LEDs in einer Matrix angeordnet, wobei ein Anschluss mit der Reihe und der andere Anschluss mit der Spalte verbunden werden.

Bei der Binäruhr werden im festen Rhythmus die Zeilen nacheinander angesteuert, also Minus an jeweils eine Zeile angelegt. Je nachdem, welche LEDs leuchten sollen, wird im richtigen Moment an die entsprechenden Spalten Plus angelegt. Die Frequenz der Anzeige beträgt etwa 650 Hertz. Jede Zeile wird also wird alle 1,5ms angesteuert und hat dabei ein Zeitfenster von 250μs. Um verschiedene Helligkeitsstufen zu erreichen, bleibt eine Zeile jedoch nicht die vollen 250μs eingeschaltet. Die empfundene Helligkeit ergibt sich aus dem Verhältnis von Einschaltzeit und Dunkelzeit. In der dunkelsten Einstellung leuchtet eine LED alle 1,5ms nur für 0,5μs.