

# Konzeption und Entwicklung einer digitalen Funk-, LED-Uhr

## Studienarbeit

Angewandte Informatik  
an der Dualen Hochschule Baden-Württemberg Stuttgart

von

**Tobias Schöneberger, Matthis Hauschild**

Juni 2013

**Abgabe**  
**Kurs**  
**Betreuer**  
**Gutachter**

10.06.2013  
TIT10AID  
Prof. Dr. Karl Friedrich Gebhardt  
Prof. Dr. Karl Friedrich Gebhardt

## **Erklärung**

Wir erklären hiermit ehrenwörtlich:

1. dass wir unsere Studienarbeit mit dem Thema *Konzeption und Entwicklung einer digitalen Funk-, LED-Uhr* ohne fremde Hilfe angefertigt haben;
2. dass wir die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet haben;
3. dass wir unsere Studienarbeit bei keiner anderen Prüfung vorgelegt habe;
4. dass die eingereichte elektronische Fassung exakt mit der eingereichten schriftlichen Fassung übereinstimmt.

Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Stuttgart, Juni 2013

---

Matthis Hauschild

Tobias Schöneberger

# Inhaltsverzeichnis

|  |           |
|--|-----------|
| <b>1. Einleitung</b>   | <b>1</b>  |
| 1.1. Motivation . . . . .  | 1         |
| 1.2. Umfang der Arbeit . . . . .                                   | 1         |
| <b>2. Anforderungen</b>  | <b>1</b>  |
| 2.1. Modularität . . . . .   | 2         |
| 2.2. Zeitempfang und Anzeige der Zeit . . . . .                    | 2         |
| 2.3. Automatische Helligkeitsanpassung . . . . .                   | 2         |
| 2.4. Temperatursensor . . . . .                                    | 3         |
| 2.5. Updatefähigkeit . . . . .                                     | 3         |
| <b>3. Technische Grundlagen</b>                                    | <b>3</b>  |
| 3.1. DCF77 . . . . .   | 3         |
| 3.2. LED Matrix . . . . .  | 5         |
| 3.3. Pulsweitenmodulation . . . . .                                | 6         |
| <b>4. Betrachtung der Komponenten</b>                              | <b>7</b>  |
| 4.1. Mikrocontroller . . . . .                                     | 7         |
| 4.2. DCF77 Empfangsmodul . . . . .                                 | 7         |
| 4.3. LED Matrix . . . . .  | 12        |
| 4.3.1. Schaltung und Hardware . . . . .                            | 12        |
| 4.3.2. Software . . . . .  | 14        |
| 4.4. Helligkeitssensor . . . . .                                   | 16        |
| 4.5. Temperatursensor . . . . .                                    | 17        |
| 4.6. Gehäuse . . . . .   | 19        |
| 4.6.1. Energieversorgung und Verbrauch . . . . .                   | 21        |
| <b>5. Betrachtung des Gesamtsystems</b>                            | <b>21</b> |
| 5.1. Software-Architektur . . . . .                                | 21        |
| 5.1.1. Ansteuern der LEDs - Timer0 . . . . .                       | 22        |
| 5.1.2. Ticken der Uhrzeit - Timer1 . . . . .                       | 22        |
| 5.1.3. Empfangen der Uhrzeit - Timer2 . . . . .                    | 24        |
| 5.1.4. Nichtzeitkritische Funktionen in der main-Methode . . . . . | 25        |
| 5.2. Hardware . . . . .  | 25        |
| 5.2.1. Hauptplatine . . . . .                                      | 25        |

|  |           |
|--|-----------|
| <b>6. Résumé</b>                               | <b>27</b> |
| 6.1. Evaluation . . . . .                      | 27        |
| 6.2. Weiterentwicklungsmöglichkeiten . . . . . | 27        |
| 6.3. Fazit . . . . .                           | 27        |
| <b>A. Quellcode</b>                            | <b>28</b> |

# **1. Einleitung**

## **1.1. Motivation**

Das Projekt einer Digitaluhr stellt sich als sehr vielseitig dar, da sowohl die Hardware als auch die Software für das Projekt erstellt werden muss. Das Themenspektrum umfasst den Funkempfang des Zeitsignals und dessen Auswertung, sowie die Implementation von fehlertoleranten Algorithmen. Die Technik des Zeitmultiplexings war zur Ansteuerung des LED Displays nötig, zudem wurden verschiedene Sensoren verbaut, die mittels 1-Wire-Bus sowie direkt mittels des integrierten A/D-Wandlers ausgelesen werden müssen.

Die begrenzten Ressourcen eines Mikrocontrollers stellen zudem hohe Anforderungen an die Effizienz des Codes. Neben all diesen Punkten war auch handwerkliches Geschick beim Aufbau des Gehäuses von Nöten. Die Arbeit im Team erforderte enge Absprachen und gute Planung.

## **1.2. Umfang der Arbeit**

Dieser Bericht soll die Design-, Konzeptions- sowie Entwicklungsphase der Digitaluhr dokumentieren. Dabei werden zunächst die technischen Grundlagen beschrieben, anschließend wird auf die einzelnen Komponenten eingegangen und darauffolgend das Zusammenspiel der Komponenten betrachtet. An relevanten Stellen wurden die entsprechenden Codeausschnitte angefügt. Es wurde darauf verzichtet, den kompletten Sourcecode in der Arbeit zu erläutern, weil er sich — gut kommentiert — im Anhang befindet. Abschließend wird das Ergebnis des Projekts im Kapitel Résumé diskutiert und kritisch betrachtet.

# **2. Anforderungen**

In der Konzeptionsphase der Digitaluhr wurden die in den folgenden Subkapiteln beschriebenen Anforderungen definiert. Da das Projekt ein rein privat initiiertes ist, sind alle Anforderungen selbst definiert. Damit sich das Projekt jedoch lohnt, wurden durchaus anspruchsvolle Anforderungen mit speziellem Fokus auf Erweiterbarkeit an die Uhr gestellt, weil die Erweiterbarkeit und Flexibilität die größten Vorteile einer Eigenkonstruktion darstellen.

## **2.1. Modularität**

Der modulare Aufbau der Digitaluhr ist eine der wichtigsten Anforderungen. Dabei bezieht sie sich sowohl auf die Hard-, als auch auf die Software.

Es soll darauf geachtet werden, dass möglichst alle externen Komponenten durch Steckverbindungen mit der Hauptplatine verbunden werden. Dies soll dafür sorgen, dass Komponenten leicht ausgetauscht werden können, sei es aus Gründen eines Defekts oder weil sich eine gleiche Komponente eines anderen Herstellers als besser erweist. Zusätzlich sorgt es dafür, dass die Hauptplatine unabhängig von den anderen Komponenten entnommen werden kann.

Auf der Softwareseite soll darauf geachtet werden, komponentenspezifischen Code in extra Funktionen auszulagern, sowie Konfigurationseinstellungen in sinnvoll benannten Konstanten festzuhalten. Damit kann in der Zukunft Funktionalität leicht und übersichtlich geändert oder hinzugefügt werden.

## **2.2. Zeitempfang und Anzeige der Zeit**

Selbstverständlich gehört auch die Anzeige der Zeit zu den wichtigsten Anforderungen an eine Uhr. Die Zeit soll über eine LED-Matrix angezeigt werden. Außerdem soll die Zeit automatisch empfangen werden können. Dafür bietet sich der Zeitzeichensender DCF77 an, der die deutsche Zeit und das Datum via Funk verbreitet. Es soll zusätzlich die Möglichkeit bestehen, die Zeit manuell einzustellen, um die Uhr einerseits auch in einer anderen Zeitzone und andererseits trotz gestörtem Empfang betreiben zu können.

Darüber hinaus soll die Möglichkeit bestehen, das Datum anzeigen (und einstellen) lassen zu können.

## **2.3. Automatische Helligkeitsanpassung**

Die Uhr soll die Umgebungshelligkeit detektieren und die Helligkeit ihrer Anzeige dynamisch anpassen können. Sie kann sich so durch hohe Helligkeit am Tag und Dimmen in der Nacht der Umgebung gut anpassen und sorgt damit für eine optimale Lesbarkeit.

## **2.4. Temperatursensor**

Die Digitaluhr soll in der Lage sein, die Temperatur des Raumes messen und anzeigen zu können.

## **2.5. Updatefähigkeit**

Diese Anforderung bezieht sich speziell auf die Software der Digitaluhr. Da eine einfache Erweiterbarkeit sowie Fehlerkorrektur gewünscht ist, soll sich die Software leicht updaten lassen. Dazu soll das Programmierinterface ohne die fertige Uhr aufschrauben zu müssen, verfügbar sein.

# **3. Technische Grundlagen**

## **3.1. DCF77**

DCF77 ist ein Zeitzeichensender in Mainhausen bei Frankfurt, der seit dem ersten Januar 1959 die Uhrzeit auf der Langwellenfrequenz von 77,5 kHz sendet. Bei Sendeanlagen, die über Ländergrenzen hinaus senden, muss das Rufzeichen in der internationalen Frequenzliste eingetragen sein und das Kennzeichen des jeweiligen Landes enthalten. Deshalb wurde DCF77 gewählt, wobei das D für Deutschland steht. Der Buchstabe C war früher ein Kennzeichen für Langwelle und das F steht für Frankfurt. Da die Sendeanlage in Mainhausen mehrere Sender hat, wurde noch die Zahl 77 als Andeutung auf die Trägerfrequenz (77,5 kHz) gewählt<sup>1</sup>.

Das Zeitsignal wird jede Minute wiederholt und kodiert Zeit- sowie Datumsinformationen. Dabei wird jede Sekunde ein Bit übertragen. Dies geschieht durch Amplitudenmodulation. Jede Sekunde wird die Amplitude der Trägerwelle für 100 ms (logisch 0) oder 200 ms (logisch 1) auf etwa 25 % abgesenkt. Lediglich in Sekunde 59 wird die Amplitude zur Erkennung der neuen Minute nicht abgesenkt. Tabelle 2 zeigt die Bedeutung der gesendeten Bits im DCF77-Signal<sup>2</sup>.

---

<sup>1</sup> [PD04], Seite 349f.

<sup>2</sup> [PD04], Seite 351ff.

| Bit     | Bedeutung  |
|---------|--|
| 0       | Minutenanfang  |
| 1 - 14  | verschlüsselte Wetterinformationen                         |
| 15      | Rufbit: Ankündigung, wenn von Reserveantenne gesendet wird |
| 16      | A1: Wechsel von/zur Sommerzeit bei nächstem Stundenwechsel |
| 17 - 18 | Zeitzonenbits: 01: MEZ, 10: MESZ                           |
| 19      | A2: Schaltsekunde bei nächstem Stundenwechsel              |
| 20      | S: Startbit (immer logisch 1)                              |
| 21      | Minutenbit, Wertigkeit 1                                   |
| 22      | Minutenbit, Wertigkeit 2                                   |
| 23      | Minutenbit, Wertigkeit 4                                   |
| 24      | Minutenbit, Wertigkeit 8                                   |
| 25      | Minutenbit, Wertigkeit 10                                  |
| 26      | Minutenbit, Wertigkeit 20                                  |
| 27      | Minutenbit, Wertigkeit 40                                  |
| 28      | Prüfbit für Minuten, Even Parity                           |
| 29      | Stundenbit, Wertigkeit 1                                   |
| 30      | Stundenbit, Wertigkeit 2                                   |
| 31      | Stundenbit, Wertigkeit 4                                   |
| 32      | Stundenbit, Wertigkeit 8                                   |
| 33      | Stundenbit, Wertigkeit 10                                  |
| 34      | Stundenbit, Wertigkeit 20                                  |
| 35      | Prüfbit für Stunden, Even Parity                           |
| 36      | Kalendertag, Wertigkeit 1                                  |
| 37      | Kalendertag, Wertigkeit 2                                  |
| 38      | Kalendertag, Wertigkeit 4                                  |
| 39      | Kalendertag, Wertigkeit 8                                  |
| 40      | Kalendertag, Wertigkeit 10                                 |
| 41      | Kalendertag, Wertigkeit 20                                 |
| 42      | Wochentag, Wertigkeit 1                                    |
| 43      | Wochentag, Wertigkeit 2                                    |
| 44      | Wochentag, Wertigkeit 4                                    |
| 45      | Kalendermonat, Wertigkeit 1                                |
| 46      | Kalendermonat, Wertigkeit 2                                |
| 47      | Kalendermonat, Wertigkeit 4                                |

| Bit | Bedeutung  |
|-----|--|
| 48  | Kalendermonat, Wertigkeit 8                            |
| 49  | Kalendermonat, Wertigkeit 10                           |
| 50  | Kalenderjahr, Wertigkeit 1                             |
| 51  | Kalenderjahr, Wertigkeit 2                             |
| 52  | Kalenderjahr, Wertigkeit 4                             |
| 53  | Kalenderjahr, Wertigkeit 8                             |
| 54  | Kalenderjahr, Wertigkeit 10                            |
| 55  | Kalenderjahr, Wertigkeit 20                            |
| 56  | Kalenderjahr, Wertigkeit 40                            |
| 57  | Kalenderjahr, Wertigkeit 80                            |
| 58  | Prüfbit für Kalenderjahr, Even Parity                  |
| 59  | Markierung der neuen Minute, keine Amplitudenabsenkung |

Tabelle 2: Erläuterung der Bits im DCF77-Zeitsignal<sup>1</sup>

### 3.2. LED Matrix

Unter LED Matrix versteht man die Ansteuerung von LEDs in Zeilen und Spalten. Dabei werden alle Anoden zu Spalten und alle Kathoden zu Zeilen verbunden.<sup>2</sup> Im Vergleich zu einer Einzelansteuerung bietet die LED Matrix den großen Vorteil, dass bei einer Matrix mit  $N$  Spalten und  $M$  Zeilen nur  $N + M$  statt  $N * M$  Leitungen verwendet werden.

Die LED Matrix wird dann zeilenweise oder spaltenweise im Multiplexbetrieb angesteuert. Das bedeutet, dass nacheinander eine der Spalten mit GND versorgt wird und die anderen Spalten unbeschaltet sind (keine Verbindung zu GND). Nun können in dieser Spalte durch Anlegen von Spannung an den entsprechenden Zeilen LEDs angeschaltet werden. Dieser Vorgang wird für alle Spalten durchgeführt, das bedeutet es leuchten zu einem bestimmten Zeitpunkt immer nur die LEDs einer Spalte. Durch das schnelle Umschalten zwischen den Spalten und die Trägheit des menschlichen Auges

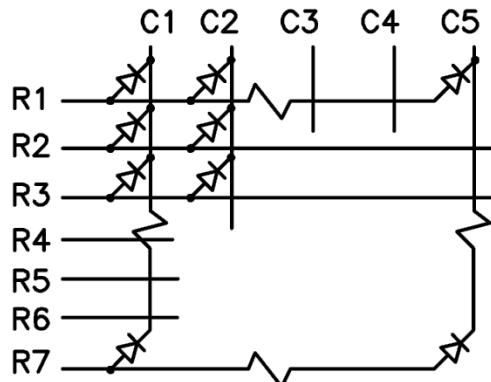


Abb. 1: 5x7 LED Matrix

<sup>2</sup> Es kann natürlich auch die Kathode für Spalten und die Anode für Zeilen verwendet werden.

entsteht die Illusion, dass auf der kompletten LED Matrix die LEDs aktiviert sind. Als Nachteil aus dieser Beschaltung ergibt sich die verringerte Helligkeit, da die LEDs bei  $N$  Spalten nur noch  $\frac{1}{N}$  der Zeit leuchten.

Der Helligkeitsverlust kann durch höheren Stromfluss zum Teil kompensiert werden. Das bedeutet bei  $N$  Spalten werden die LEDs mit einem Pulssstrom von  $N * Nennstrom$  betrieben. Durch die Dunkelphasen kann das aktive Substrat zwischen den Pulsen ausreichend abkühlen. Generell kann dies bis zum ca. zehnfachen Nennstrom (200 mA bei einer gewöhnlichen 20 mA LED) durchgeführt werden.<sup>1</sup>

### 3.3. Pulsweitenmodulation

Pulsweitenmodulation bezeichnet eine Modulationstechnik in der die Weite des Pulses bei einer gleichbleibenden Periode verändert wird (siehe 2). Diese Modulationstechnik erlaubt es, die Leistung von Geräten zu regulieren. Der durchschnittliche Stromfluss wird durch das Verhältniss  $\frac{\text{Pulsweite}}{\text{Periode}}$  definiert. Gilt  $\text{Pulsweite} = \text{Periode}$  so erhält das Gerät 100% der Leistung, gilt  $\frac{\text{Pulsweite}}{\text{Periode}} = \frac{1}{2}$  so wird das Gerät mit halber Leistung versorgt. Die Periode wird in der Regel sehr klein gewählt, zum Beispiel  $\frac{1}{100}\text{s}$  bei LEDs, da das menschliche Auge 100 Hz blinken als konstantes Leuchten wahrnimmt.

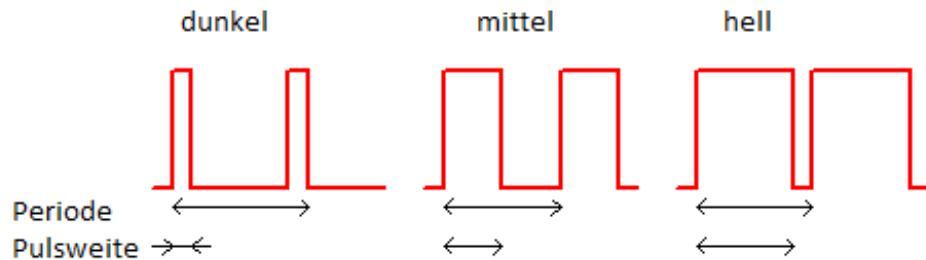


Abb. 2: Schema der Pulsweitenmodulation

---

<sup>1</sup> <http://www.mikrocontroller.net/articles/LED-Matrix>

## 4. Betrachtung der Komponenten

### 4.1. Mikrocontroller

Die zentrale Komponente der Digitaluhr ist der Mikrocontroller ATmega32. Der von Atmel hergestellte 8-bit Kontroller ist im 40-pin DIP<sup>1</sup> Format verfügbar. Dies ermöglicht die einfache Verwendung auf einer Lochrasterplatine mit 2,54 mm Lochabstand. Programmiert wird der ATmega entweder in C oder in AVR-Assembler, die Wahl fiel hier auf die komfortablere Sprache C. Viele der integrierten Komponenten wurden genutzt: Der zur Verfügung gestellte SPI-Bus<sup>2</sup> zur Kommunikation mit den Schieberegistern, der integrierte A/D-Wandler zur Auswertung des Helligkeitssensors und das ISP-Interface zur Programmierung.

|                  |                     |
|------------------|---------------------|
| Bezeichnung      | ATMEGA32 16PU 0926D |
| Hersteller       | Atmel               |
| Architektur      | AVR 8-bit           |
| Geschwindigkeit  | bis 16Mhz           |
| Programmspeicher | 32 KiB Flash        |
| Arbeitsspeicher  | 2 KiB SRAM          |
| EEPROM           | 1 KiB               |
| AD-Wandler       | 8 Kanal / 10 Bit    |
| Bauform          | 40-pin DIP Gehäuse  |

Tabelle 3: Eckdaten des ATmega 32

### 4.2. DCF77 Empfangsmodul

Das in der Digitaluhr verwendete Modul ist die „C-Control DCF-Empfängerplatine“<sup>3</sup> (siehe Abbildung 3) von Conrad.

<sup>1</sup> Dual in-line package

<sup>2</sup> Serial Peripheral Interface

<sup>3</sup> <http://www.conrad.de/ce/de/product/641138/C-Control-DCF-Empfaengerplatine>, Bestellnummer: 641138 - 62

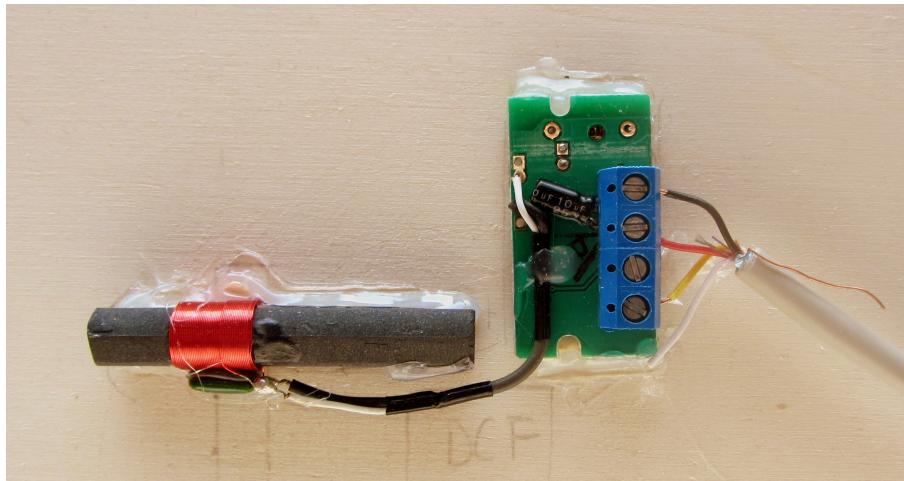


Abb. 3: C-Control-DCF-Empfängerplatine von Conrad

Das Modul besitzt vier Anschlüsse, welche im Anschlussplan rechts beschrieben sind. Es wird der DCF Ausgang invertiert für das Datensignal verwendet. Er ist an Mikrocontrollerpin PB2 angeschlossen. Damit die Uhrzeit auf jeden Fall richtig empfangen wird, ist einerseits die Ausrichtung auf Frankfurt wichtig, sowie andererseits die Verwendung eines fehlertoleranten Codes zur Auswertung der Uhrzeit. Außerdem sollte auf den Abstand zu Störquellen geachtet werden. Als starke Störquellen stellten sich bei der Entwicklung Monitore heraus. In Abbildung TODO ist der Unterschied zwischen einem einwandfreien Signal und einem, durch einen Monitor in 2 m Entfernung, gestörtem Signal abgebildet. Es ist zu erkennen, dass das Signal plötzlich im Mikrosekundenbereich gleichmäßig alterniert, also von dem ursprünglichen Signal nichts mehr zu erkennen ist.

Nachfolgend soll die Funktion `byte conrad_state_get_dcf_data()` erklärt werden. Diese Funktion ist für den Empfang der Daten des DCF77 Moduls verantwortlich. Sie ist als State Machine realisiert und wird beim Empfangen der Daten jede 10 ms aufgerufen.

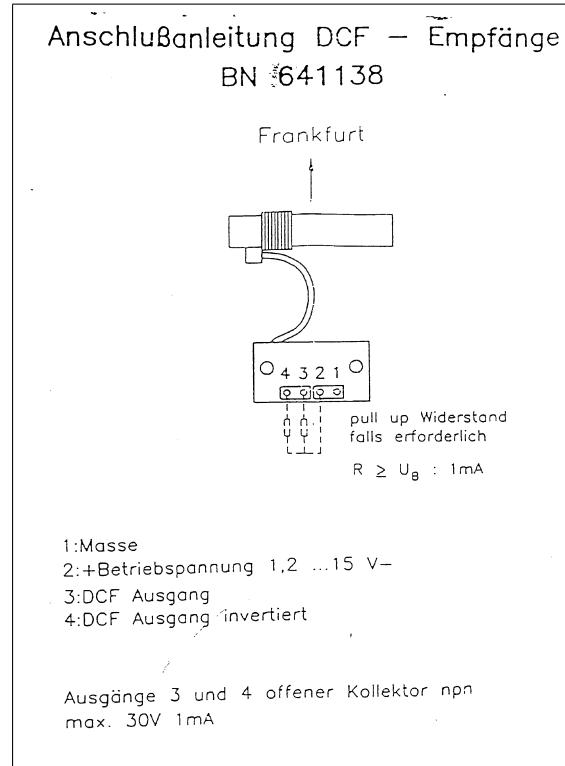


Abb. 4: Anschlussanleitung DCF77 Modul<sup>1</sup>

Dies wird durchgeführt, weil die Funktion sonst sehr lange laufen würde (der vollständige Empfang der Daten dauert im Best Case 60 Sekunden) und andere Funktionen ihre Arbeit nicht mehr verrichten können. Mit diesem Ansatz aber bleibt die Uhr reaktiv und kann beispielsweise weiterhin auf Tasterevents reagieren.

**Listing 1: Empfang des DCF77 Signals - Minutenstart erkennen**

```

1 if (i < 155) {
2     /* DCF Signal unmoduliert (da es invertiert ist, ist es
   standartmaessig 1) */
3     if (DCF_VALUE != 0) {
4         i++;
5         j = 0;
6         DBG_LED_OFF();
7         /* Wenn es moduliert ist (logisch 0) */
8     } else {
9         j++;
10
11     if (j > 7) {
12         i = 0;
13         j = 0;
14         DBG_LED_ON();
15     }
16 }
17 return T2_WAIT;
18 }
```

---

In Listing 1 ist zunächst der erste Teil der Funktion zu sehen. Dabei geht es um das Erkennen des Minutenanfangs. Wie im DCF77 Grundlagenkapitel 3.1 erläutert, wird das Signal beim Übergang von der 59. auf die 60. Sekunde nicht moduliert, es tritt also zwei Sekunden lang keine Amplitudenabsenkung auf.

Die Variable i zählt die Zentisekunden, in denen das Signal unmoduliert ist. Aus Toleranzgründen wird lediglich geschaut, ob i den Wert von 155 übersteigt, statt 200 (= 2 s). Es kann nämlich passieren, dass das Signal zufällig genau zum Messzeitpunkt leicht abfällt. Deshalb werden auch Werte von j, welches die modulierten Zentisekunden misst, von weniger als 70 ms ignoriert. Die durch Tests ermittelten Werte von 155 und 7 sorgen dafür, dass Ungenauigkeiten zuverlässig vermieden werden.

Wie zu erkennen ist, gibt es keine Schleife, sondern lediglich sequentiellen Code, sodass die Funktion auf jeden Fall verlassen wird (siehe Zeile 17). Es gibt die folgenden Return-Codes:

**T2\_WAIT**      Resettet den Counter von Timer2, sodass genau 10 ms gewartet wird

**ERROR**

Bricht den Messvorgang ab, weil ein Fehler aufgetreten ist

**SUCCESS**

Alle 60 Bits wurden gemessen. Dabei wurde kein Fehler erkannt

In Fall des Beispielcodes wird der Returncode T2\_WAIT verwendet, damit die Funktion nach 10 ms nochmals aufgerufen wird und die Messung fortgesetzt werden kann.

**Listing 2:** Empfang des DCF77 Signals - Sekunde analysieren

```
1 if (is_start_of_sec) {
2     /* Pausiere bis zum modulierten Signal */
3     if (DCF_VALUE != 0) {
4         return T2_WAIT;
5     }
6     is_start_of_sec = false;
7 }
8 if (k < 95) {
9     if (DCF_VALUE != 0) {
10        unmodulated++;
11    } else {
12        if (k < 40) {
13            modulated++;
14        }
15    }
16    k++;
17    return T2_WAIT;
18 }
```

Der nächste Schritt (siehe Listing 2) ist das Analysieren jeder Sekunde, also das Zählen der modulierten und nicht-modulierten Signale. Dazu wird zunächst gewartet, bis die Sekunde anfängt, also das erste modulierte Signal auftritt (Zeile 3), anschließend wird 95 % der Sekunde analysiert. Die letzten 5 % sind Toleranz, damit die nächste Sekunde sicher erkannt werden kann. Modulierte Signale treten nur in den ersten 200 ms einer Sekunde auf, deshalb werden sie nur am Anfang gezählt, aber aus Toleranzgründen innerhalb der ersten 400 ms (Zeile 12). Auch hier wird nach jeder Messung die Routine mit T2\_WAIT verlassen (Zeile 17).

**Listing 3:** Empfang des DCF77 Signals - Messung auswerten

```
1 if (unmodulated > 50 && unmodulated < 140) {
2     /* Wenn moduliert zwischen 50 und 140 ms, liegt logisch 0 an
   */
3     if (modulated > 5 && modulated < 14) {
4         dcf_data[secs] = 0;
5         /* Zwischen 150 ms und 240 ms, liegt logisch 1 an */
6     } else if (modulated > 15 && modulated < 24) {
7         dcf_data[secs] = 1;
8     /* sonst ist es ungültig */
}
```

```

9      } else {
10         return ERROR;
11     }
12 } else {
13     return ERROR;
14 }
15 /* Bereite die naechste Sekunde vor */
16 secs++;
17 is_start_of_sec = true;
18 k = 0;
19 modulated = 0;
20 unmodulated = 0;
21
22 return SUCCESS;

```

---

Als letztes werden die Daten ausgewertet, also geprüft ob logisch eine 0 oder 1 gemessen wurde. Dazu wurden Zahlen gewählt, die sich durch Tests bewährt haben (siehe Listing 3). Bei fehlerhaften Daten wird ERROR zurückgegeben (Zeile 10 und 13) und die gesamte Messung abgebrochen. Sind die Daten gültig, wird die State Machine auf den Anfangszustand jeder Sekunde gesetzt (Zeilen 16 – 20) und SUCCESS zurückgegeben und damit die Auswertung der nächsten Sekunde begonnen.

Wie zu sehen ist, werden statt diskreten Werten immer Bereiche angenommen, in denen ein bestimmter Wert liegen muss. Dies ist durch die funkbedingten Ungenauigkeiten unbedingt notwendig. Es erforderte einige Tests, bis die passenden Werte gefunden wurden, um einen passablen Empfang zu erreichen, was letzten Endes jedoch gelang und zu einem korrekt funktionierenden Funkmodul für die Digitaluhr führte.

## 4.3. LED Matrix

### 4.3.1. Schaltung und Hardware

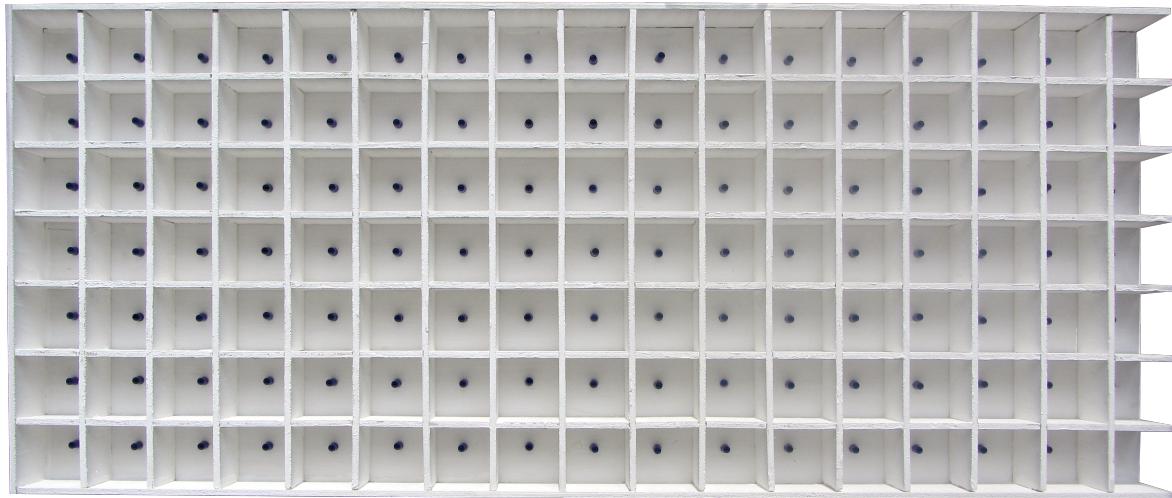


Abb. 5: LED Matrix Draufsicht

Die LED Matrix dient der Uhr als Display. Es werden  $7 * 17 = 119$  diffuse blaue LEDs verwendet. Die LEDs haben einen Abstand von  $45mm$  und werden durch ein Gitter aus Sperrholz voneinander getrennt, so dass Pixel entstehen. Durch das Multiplexen der Zeilen kann jede LED unabhängig gesteuert werden.

Die Anoden der Reihen sind jeweils verbunden und an die Schieberegister angeschlossen. Die Kathoden sind zu Zeilen verbunden und über die Mosfets vom Mikrocontroller geschaltet. Eine Ausnahme bildet die Reihe 17. Da die zwei Schieberegister nur 16 Ausgänge besitzen, wurde diese direkt an den Mikrocontroller angeschlossen.

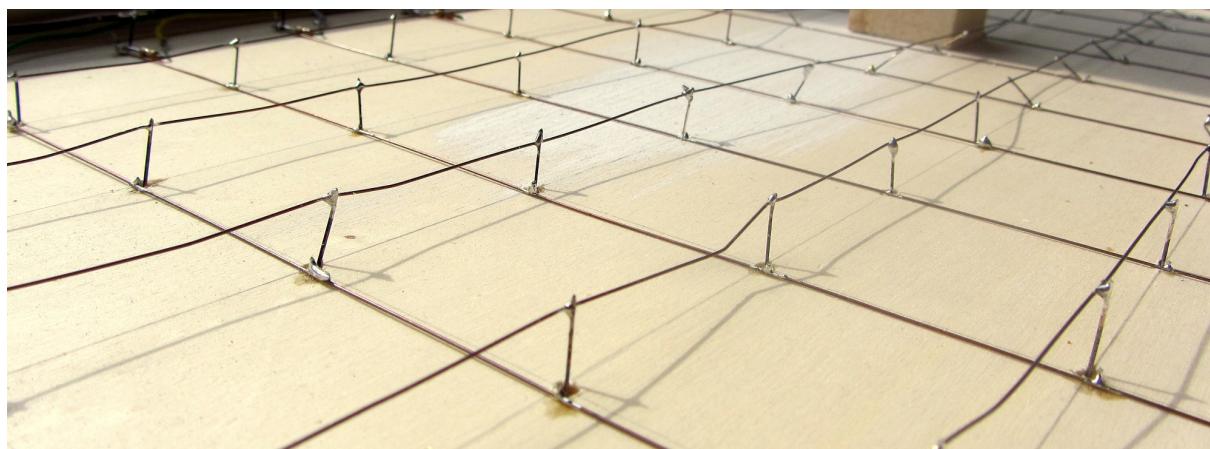


Abb. 6: LED Matrix Verdrahtung, Anoden aufliegend und Kathoden schwebend verbunden

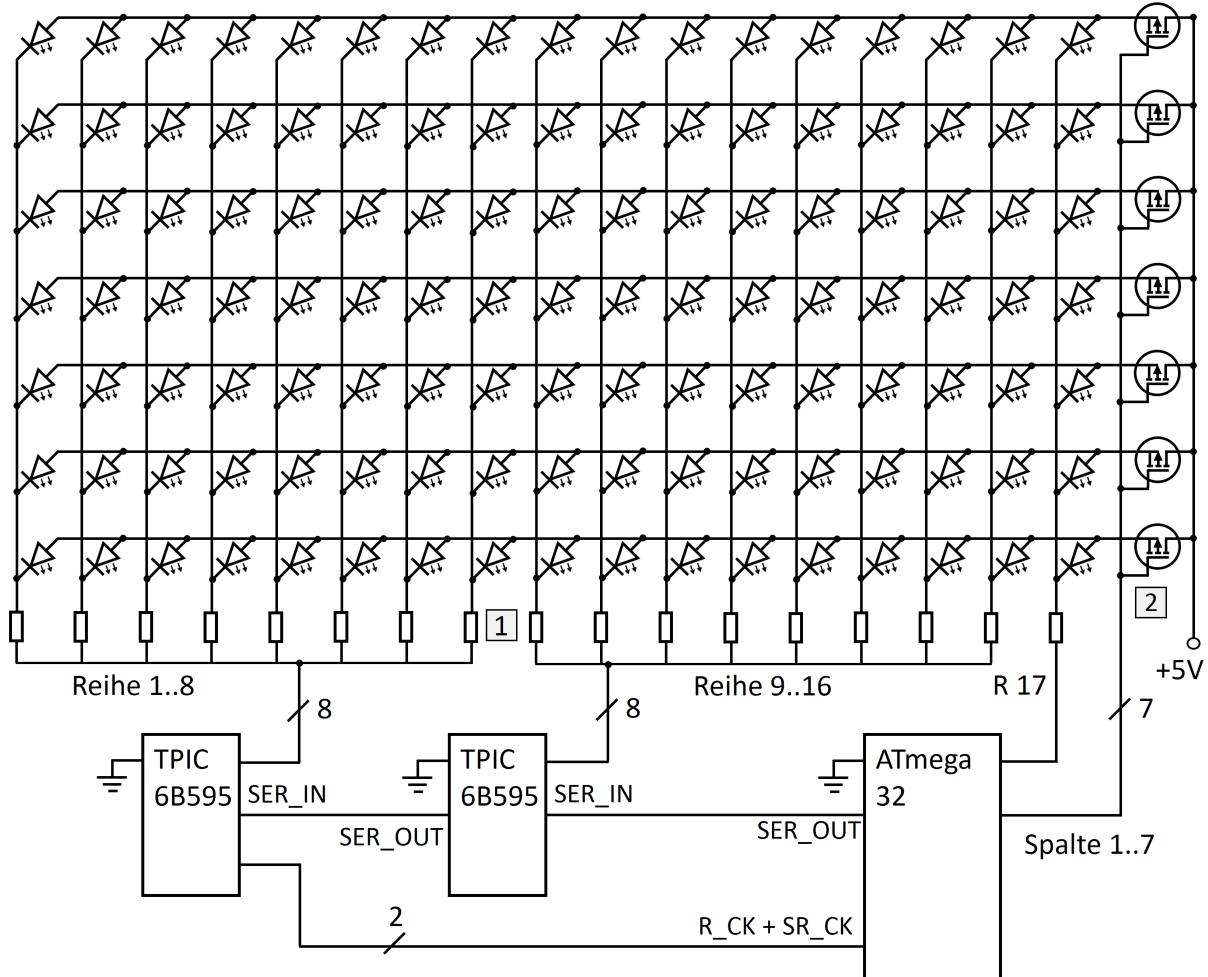


Abb. 7: LED Matrix Schaltplan, 1: LED Vorwiderstände, 2: P-Kanal Mosfets IRLZ24N

Durch die Ansteuerung als Matrix werden nur 24 Leitungen von der LED Anzeige zur Hauptplatine benötigt. Da der Mikrocontroller auf seinen Logikausgängen nicht die benötigte Leistung bereitstellen kann, wird die Spannung durch die P-Kanal Mosfets vom Typ IRLZ24N geschalten.

Um weitere Pins am ATmega einzusparen, werden die Reihen (abgesehen von Reihe 17) mittels Schieberegister gesteuert. Die beiden Schieberegister vom Typ TPIC6B595 können dauerhaft Ströme bis zu 150mA pro Pin und 500mA Gesamtstrom gegen GND schalten.<sup>1</sup> Die Kommunikation zwischen ATmega und den Schieberegistern erfolgt über den SPI Bus, welcher sehr einfach mit Daten gefüllt werden kann und diese unabhängig vom eigentlichen Programmablauf an die Schieberegister schickt.

---

<sup>1</sup> vgl. [Ins05], S. 1

Der SPI Bus wird über drei Leitungen realisiert: SER IN als Datenleitung, SRCK zur Taktübertragung, RCK um die Daten vom Schieberegister auf die Ausgänge zu legen. Das zweite Schieberegister wird parallel mit RCK und SRCK verbunden und der serielle Ausgang des ersten Schieberegisters SER OUT wird an SER IN des zweiten Schieberegisters angeschlossen. Durch diese Schaltung erreicht man quasi ein 16-Bit Schieberegister. Insgesamt benötigt das Display somit 11 Pins am Mikrocontroller: 3 für die Schieberegisteransteuerung, 7 für die Ansteuerung der Mosfets sowie eine Leitung um Reihe 17 zu schalten.

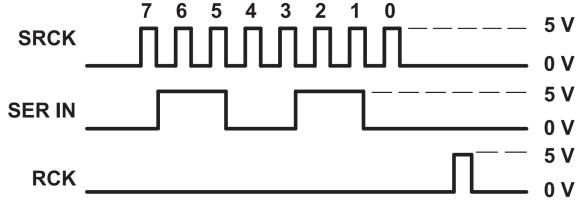


Abb. 8: Füllen des Schieberegisters<sup>1</sup>

### 4.3.2. Software

Im Folgenden wird die Ansteuerung der LED Matrix beleuchtet, diese findet in der Methode `drawWithBrightness (void)` statt. Sie gibt jeweils eine komplette Zeile aus und wird periodisch 7200 mal pro Sekunde aufgerufen (siehe Kapitel 5.1.1). Zu Anfang wird der aktuelle `cmp` Vergleichswert mit der global eingestellten Helligkeit `brightness` verglichen. Wenn `brightness` größer als dieser Wert ist, bleiben die LEDs im Aufruf dunkel, was für die dynamische Helligkeitsanpassung von Nöten ist.

Wenn gezeichnet werden soll, wird auf die Bilddaten im Array `data` zugegriffen und für jedes Pixel überprüft, ob dieses aktiv sein soll. Die Pixel werden durch Schiebeoperationen nacheinander in die `output` Variable geschrieben.

**Listing 4:** Zeichenmethode zur Ansteuerung der LED Matrix: Teil 1

```

1 /*Draws all the pixel with the brigtness of the global brightness
   variable*/
2 static inline void drawWithBrightness(void) {
3     byte output=0;
4     if(brightness>cmp) {
5         byte i=0;
6         /* The first output Byte */
7         for(i=0;i<8;i++) {
8             output = output<<1;
9             if(data[row][i]>0) {
10                 output++;
11             }
12         }

```

Der Inhalt der `output` Variable wird anschließend in das SPDR Register geschrieben, welches den Buffer der SPI Schnittstelle darstellt. Im Hintergrund beginnt nun das SPI Modul des ATmegas die Daten an das Schieberegister zu übertragen. Währenddessen werden die Pixeldaten für Reihe 9–16 verarbeitet. Anschließend wird gewartet bis die Übertragung zum Schieberegister abgeschlossen ist und der neue `output` Wert in das SPDR Register geschrieben.

**Listing 5:** Zeichenmethode zur Ansteuerung der LED Matrix: Teil 2

```

1  /* output to SPI-Register */
2  SPDR = output;
3  output = 0;
4  /* Second output Byte */
5  for(;i<16;i++){
6      output = output<<1;
7      if(data[row][i]>0){
8          output++;
9      }
10 }
11 /* Wait for SPI transmission complete*/
12 while(!(SPSR & (1<<SPIF)));
13 /* output to SPI-Register */
14 SPDR = output;

```

---

Die Ausgabe von Reihe 17 unterscheidet sich deutlich, da der Mikrocontrollerpin auf dem selben Port wie die Mosfets liegt. Außerdem muss hier GND anliegen, wenn die LED aktiviert sein soll. Deshalb wird `output` mit 128 (acht Bit im Byte) initialisiert und wenn das Pixel aktiv ist mit 0 überschrieben.

Im `else`-Fall, wenn alle LEDs aus bleiben sollen, werden Nullen in die Schieberegister geschoben und das achte Bit von `output` mit 1 beschrieben.

**Listing 6:** Zeichenmethode zur Ansteuerung der LED Matrix: Teil 3

```

1  /* Last Bit direct on microcontroller pin */
2  if(data[row][i]>0){
3      output=0;
4  }
5  while(!(SPSR & (1<<SPIF)));
6 }else{
7     /* output to SPI-Register */
8     SPDR = 0;
9     /* Wait for SPI transmission complete*/
10    while(!(SPSR & (1<<SPIF)));
11    SPDR = 0;
12    while(!(SPSR & (1<<SPIF)));
13
14

```

```
15     output=128; //LED 17 aus
16 }
```

---

Zuletzt muss die RCK Leitung der Schieberegister auf Low und anschliessend auf High geschaltet werden, sowie der Mosfet der aktuellen Zeile aktiviert werden. Die Mosfets schalten durch, wenn ein Low Pegel angelegt wird. Um während des Schaltens im Schieberegister Flackern auf der Anzeige zu verhindern, werden zuerst alle Mosfets ausgeschaltet, dann die Schieberegisterinhalte auf die Schieberegisterausgänge übernommen und anschliessend der Mosfet der Zeile aktiviert.

Abschließend wird die Zeilennummer für den nächsten Aufruf der Methode inkrementiert und wenn ein komplettes Bild gezeichnet wurde `row==7`, wird der `cmp` Wert um 16 erhöht.

**Listing 7:** Zeichenmethode zur Ansteuerung der LED Matrix: Teil 4

```
1  /* RCK auf null ziehen */
2  PORTB &= 0b11101111;
3  /* Alle Mosfets aus, PD7 unbelegt */
4  PORTD = 0xFF;//
5  /* RCK auf high */
6  PORTB |= 0b00010000;
7  /* Mosfet wieder an */
8  PORTD = states[row++] + output;
9
10 if(row == 7){
11     row = 0;
12     /* Increment cmp-variable */
13     cmp+=16;
14 }
15 }
```

---

## 4.4. Helligkeitssensor

Ein Fotoresistor (lichtabhängiger Widerstand, kurz LDR für Light Dependend Resistor) bildet einen Teil eines Spannungsteilers. Die innerhalb des Spannungsteilers anliegende Spannung wird dann vom Mikrocontroller mit Hilfe des A/D-Wandlers ausgewertet. Der verwendete Fotoresistor besitzt nominal einen Dunkelwiderstand von  $100k\Omega$ . Um den Widerstandsverlauf besser abschätzen zu können, wurden folgende Messungen vorgenommen.

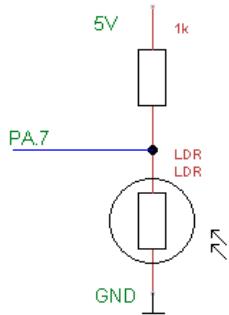


Abb. 9: Spannungsteiler des LDRs

|                                |                    |
|--------------------------------|--------------------|
| Dunkelheit                     | $70\text{k}\Omega$ |
| Zimmerhelligkeit               | $3\text{k}\Omega$  |
| Schreibtischlampe 50cm Abstand | $500\Omega$        |
| Schreibtischlampe 10cm Abstand | $90\Omega$         |
| Schreibtischlampe 2cm Abstand  | $30\Omega$         |

Tabelle 4: Widerstandsmessung des LDR

## 4.5. Temperatursensor

Als Temperatursensor wurde der DS18S20<sup>1</sup> von Maxim Integrated verwendet. Dieser verwendet das 1-Wire®-Protokoll<sup>2</sup>, sodass nur ein Datenpin voneinander trennen ist, worüber Befehle gesendet sowie Daten empfangen werden.

Der 1-Wire-Bus ermöglicht die Verwendung mehrerer Geräte an der Datenleitung, wobei jedes Gerät über seine eindeutige 64-Bit ID angesprochen wird. Da bei diesem Projekt lediglich ein DS18S20 verwendet wird, kann die Ansteuerung der eindeutigen ID weggelassen werden und alle Geräte auf dem Bus angesprochen werden, da keine Kollision entstehen kann.

Der Temperatursensor wird wie in Grafik 10 angeschlossen. Die Datenleitung wird über einen  $4,7\text{k}\Omega$  Pullup-Widerstand mit der Spannungsquelle verbunden, sowie an Pin PA6 des Mikrocontrollers angeschlossen. Für das 1-Wire-Protokoll ist ein genaues Timing unabdingbar, deshalb müssen bei jeglicher Kommunikation mit dem DS18S20 alle Interrupts ausgeschaltet werden. Da die längste Kommunikation am Stück der Reset-Pulse mit  $960\mu\text{s}$  ist, ist das Ausschalten der Interrupts für die Funktionalität der Uhr nicht beeinträchtigend.

Das Protokoll sieht vor, dass jede Kommunikation mit einem Reset-Pulse startet. In Abbildung 11 ist das genaue Timing des Reset-Pulses zu erkennen. Zunächst muss der Master, also hier der Mikrocontroller den Bus für mindestens  $480\mu\text{s}$  auf Low ziehen. Nach Freigabe des Bus' sorgt der Pullup-Widerstand dafür, dass der Bus wieder auf High gesetzt wird. Anschließend zieht der Sensor den Bus nach  $15\mu\text{s}$  bis  $60\mu\text{s}$  nach der ansteigenden

<sup>1</sup> <http://www.maximintegrated.com/datasheet/index.mvp/id/2815>

<sup>2</sup> <http://www.maximintegrated.com/products/1-wire/flash/overview/index.cfm>

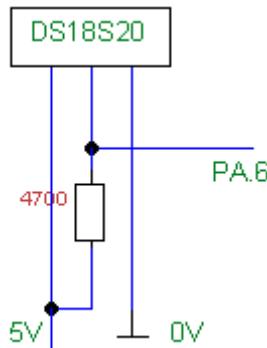


Abb. 10: Schaltung für den 1-Wire Temperatursensor

Flanke für  $60\mu s$  bis  $240\mu s$  auf Low. Dies signalisiert dem Master, dass ein Gerät am Bus hängt und einsatzbereit ist.<sup>1</sup>

In der weiteren Kommunikation wird nun das SKIP ROM Kommando benutzt, welches bewirkt, dass die 64-Bit ID weggelassen werden kann und alle Geräte (hier nur eins) angesprochen werden. Nun kann der DS18S20 mit dem CONVERT TEMP Kommando dazu aufgefordert werden, die Temperatur zu messen und digital zu konvertieren. Dies macht er nicht permanent, um Strom zu sparen. Nach einer Konvertierungszeit von mindestens  $700ms$  kann die Temperatur ausgelesen werden. Dazu muss erneut ein Reset- sowie ein Skiprom-Kommando gesendet werden, um dann mit READ SCRATCHPAD die eigentliche Temperatur zu erhalten.<sup>2</sup>

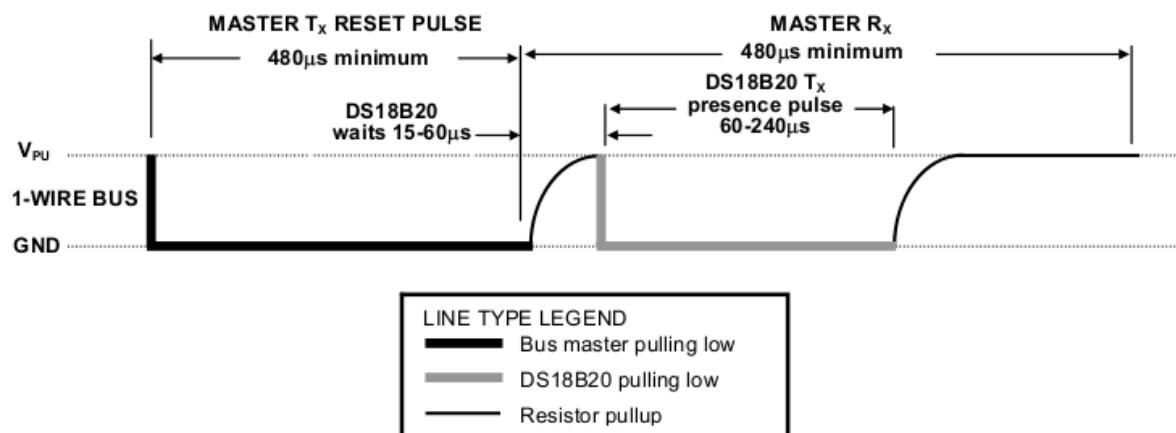


Abb. 11: Reset-Pulse Timing Diagramm<sup>3</sup>

<sup>1</sup> [mi10], Seite 13

<sup>2</sup> Genaue Beschreibung der Kommandos siehe [mi10], Seite 10ff

## **4.6. Gehäuse**

Ziel war es ein optisch ansprechendes, sowie funktionales Gehäuse zu kreieren.

Als problematisch stellte sich die Zielsetzung der flachen Bauart heraus. Denn um eine gleichmässige Lichtverteilung innerhalb eines Pixels zu erreichen, muss ein relativ großer Abstand zur LED gegeben sein. Durch praktisches Testen ergab sich für die verwendeten LEDs ein idealer Abstand von  $33\text{mm}$ . Außerdem muss verhindert werden, dass die offenliegende Verdrahtung der LED-Matrix zu Kurzschlüssen führt. Im Besonderen ist hier das Netzteil zu nennen, da hier eine Spannung von  $230\text{ Volt}$  anliegt und das Netzteil von allen Komponenten mit  $20\text{mm}$  über die höchste Bauhöhe verfügt. Die Bauhöhe der Verdrahtung der LED-Matrix wurde an den kritischen Stellen von  $5 - 6\text{ mm}$  auf ca.  $2\text{ mm}$  verringert. Um Kurzschlüsse innerhalb der LED-Matrix zu verhindern, wurden die Kreuzungspunkte zum Teil isoliert.

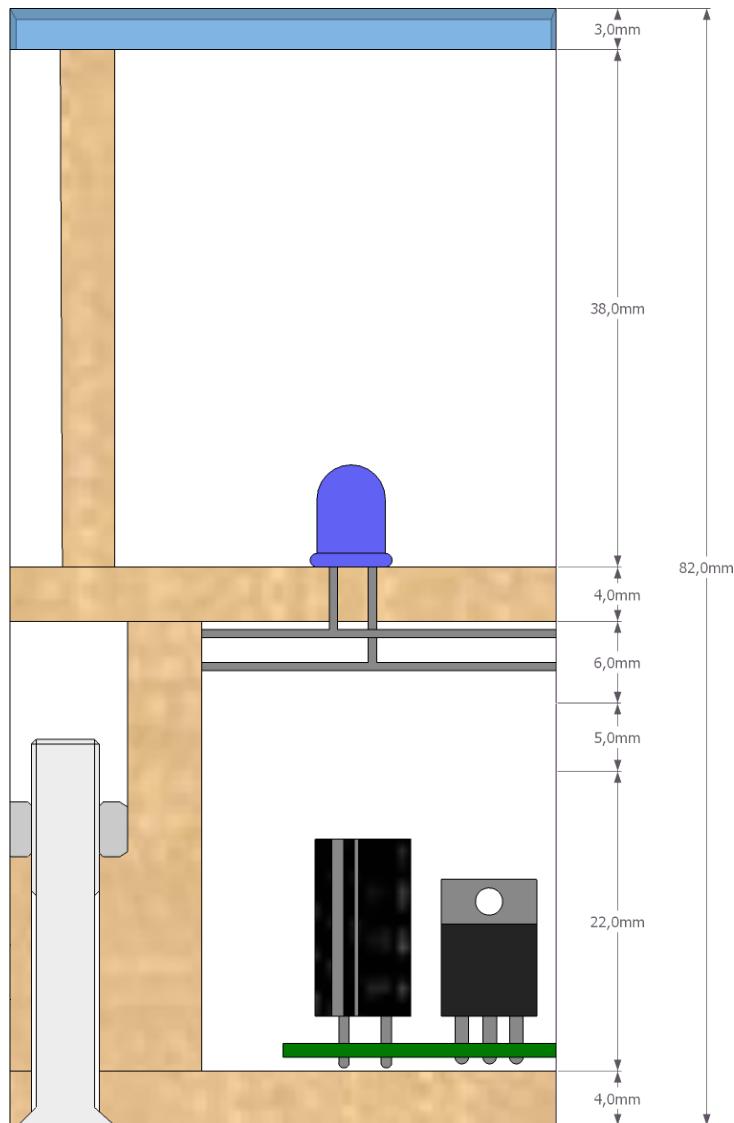


Abb. 12: Schematischer Querschnitt des Gehäuses

Während der Entwicklung war ein einfacher Zugang von großer Bedeutung, deshalb wurde das Gehäuse in zwei Teile aufgeteilt. Die LED-Matrix bildet zusammen mit ihrer Abdeckung und drei Seitenwänden den vorderen Teil des Gehäuses. Auf der Rückwand wurde die Hauptplatine, das Netzteil und der DCF77-Empfänger platziert. An der vierten Seitenwand sind die Sensoren für Helligkeit und Temperatur, die Stromversorgung, die 6 Taster sowie der Debuganschluss (ISP<sup>1</sup>) befestigt (siehe Abbildung 13). Diese Seitenwand wurde an der Rückwand befestigt. Dies ermöglicht ein Öffnen des Gehäuses, indem nur die Schrauben an der Rückwand entfernt werden und die 3 Steckverbinder zur LED-Matrix gelöst werden, die komplette Sensorik und die Taster aber nicht entfernt werden müssen.

<sup>1</sup> In-system programming



Abb. 13: Taster, Sensoren und ISP-Schnittstelle der Digitaluhr

Als Schrauben kamen Metallschrauben mit M5 Gewinde zum Einsatz. Die Muttern wurden in einem Holzblock befestigt, der anschließend mit der Rückwand verleimt wurde. Diese Lösung zeichnet sich im Gegensatz zu Holzschrauben durch minimalen Verschleis aus und kann oft geöffnet und wieder verschlossen werden ohne Auszufransen.

#### 4.6.1. Energieversorgung und Verbrauch

Als Netzteil wurde ein CE geprüftes 5V/2A Netzteil gewählt. Das kompakte verwendete Schaltnetzteil ist ausreichend dimensioniert um den, mit einem Labornetzteil, ermittelten maximalen Bedarf von ca. 1.8A (alle LEDs aktiv bei maximaler Helligkeit) bereitzustellen.

Als Stromkabel kommt ein zweipoliges Kabel mit Schalter zum Einsatz. Dieses wurde im Inneren des Gehäuses mit Schmelzklebestoff verklebt und in eine Lüsterklemme geführt, so dass bei eventuell auftretenden Zugkräften auf keinen Fall Kräfte auf das Netzteil wirken.

Mittels des Temperatursensors wurde außerdem in einem Testlauf sichergestellt, dass die Temperatur im Inneren der Uhr 50°C (bei einer Raumtemperatur von 22°C) nicht überschreitet.

### 5. Betrachtung des Gesamtsystems

#### 5.1. Software-Architektur

Die Software lässt sich in zwei Einheiten unterteilen:

- zeitkritische Operationen
- nichtzeitkritische Operationen

Zur Realisierung der zeitkritischen Operationen, wie dem Ticken der Uhr, dem Empfangen der Zeit sowie dem Ansteuern der LEDs wurden die Timer des AVR Mikrocontrollers benutzt. Der verwendete ATmega32 besitzt drei Timer, einen 16-Bit- und zwei 8-Bit-Timer.

Die Timer senden in definierten Abständen Interrupts, die zur sofortigen Unterbrechung des Hauptprogrammes führen und den angegebenen Interrupt-Handler ausführen. Dazu wird der momentane Programmkontext auf den Stack gesichert, der Kontext für die Interrupt-Routine geladen, selbige ausgeführt und abschließend der Programmkontext wiederhergestellt, sodass die Ausführung des Programmes an der selben Stelle fortgeführt wird, wo es unterbrochen wurde. Unterbrechbar sind alle nicht-zeitkritischen Funktionen, die in der Endlosschleife der `main`-Methode des Programmes ausgeführt werden.

Der Programmablaufplan 14 beschreibt die `main`-Methode sowie die drei Interrupt-Service-Routinen. In den folgenden Unterkapiteln werden die einzelnen Routinen nochmals textuell erläutert.

### 5.1.1. Ansteuern der LEDs - Timer0

Zum Ansteuern der LEDs wurde der 8-Bit Timer0 im „Interrupt by Overflow“-Modus verwendet. Dies bedeutet, dass er bei jedem Überlauf, also von  $2^8 - 1$  auf 0, einen Interrupt erzeugt. In dem Setup wurde ein 14,7456 MHz Quarz sowie ein Prescaler von 8 verwendet. Dies sorgt dafür, dass nur in jedem 8. Taktschritt der Zähler des Timers inkrementiert wird. Daraus resultiert eine Interruptrate von  $\frac{\text{Clock}}{\text{Prescaler} * \text{Steps}} = \frac{14745600\text{Hz}}{8 * 2^8} = 7200\text{Hz}$ . Dies bedeutet, dass die Funktion `static inline void drawWithBrightness(void)` 7200 mal in der Sekunde aufgerufen wird. Diese Funktion steuert bei jedem Aufruf eine Reihe von LEDs an und inkrementiert den Reihenzähler anschließend. Es entsteht also bei 7 Aufrufen der Funktion ein gesamtes Bild und damit ist die Bildfrequenz  $\frac{7200\text{Hz}}{7} \approx 1028,57\text{Hz}$ . Dies wirkt zunächst viel, wird aber durch die Pulsweitenmodulation (siehe Kapitel 3.3) für die Helligkeitsabstufungen noch deutlich herabgesetzt.

### 5.1.2. Ticken der Uhrzeit - Timer1

Für die zeitlich kritischste Funktion, dem exakten Ticken der Uhrzeit wurde der 16-Bit Timer1 im „CTC“-Modus verwendet. CTC steht für Clear Timer on Compare Match und ist ein Modus, bei dem ein Höchstwert spezifiziert werden kann, bei dem der Timer einen

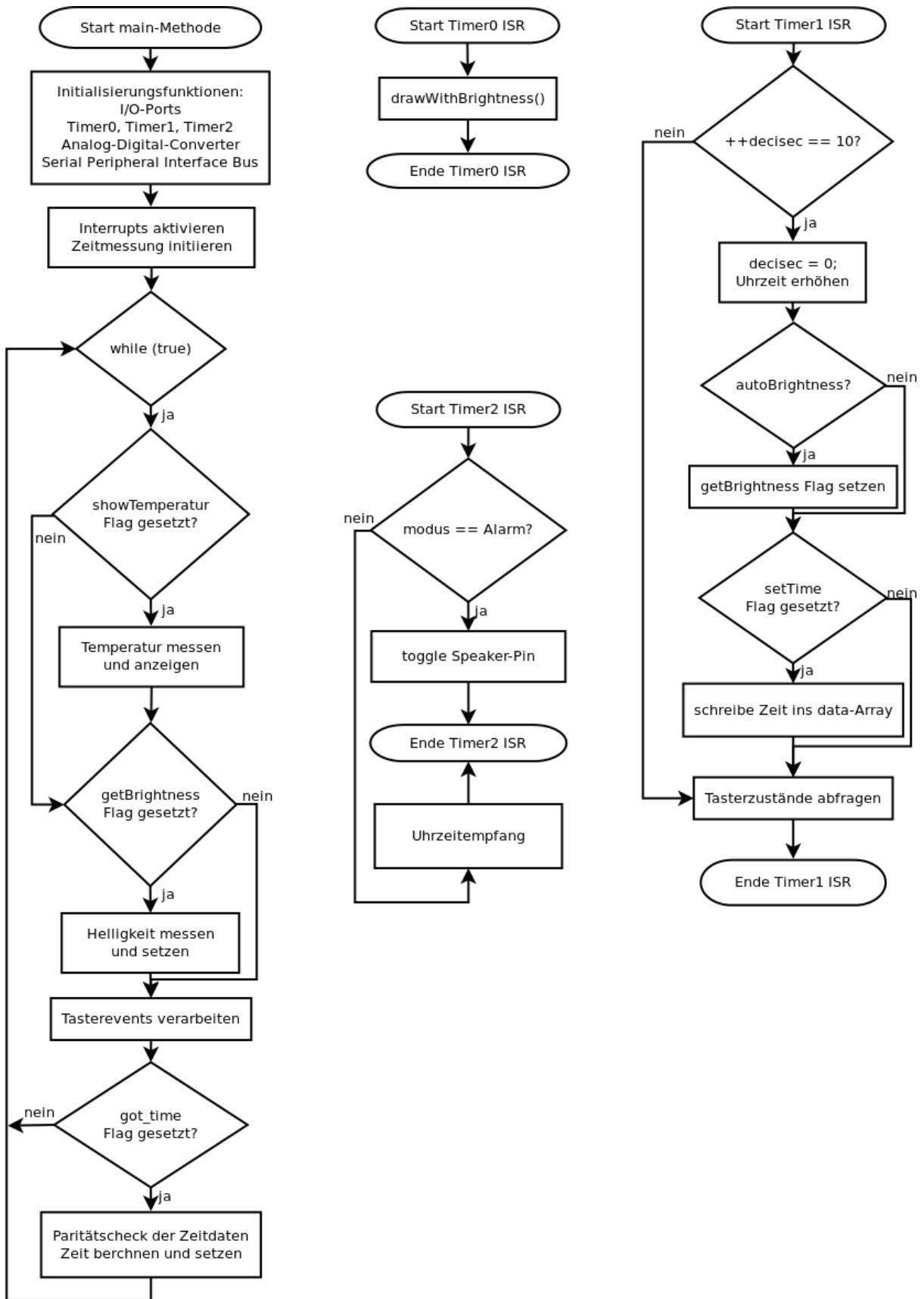


Abb. 14: Programmablaufplan für die main-Methode sowie die drei ISR

Interrupt auslöst und sofort wieder bei 0 zu zählen beginnt. Damit kann ein taktgenauer Timer realisiert werden, wodurch kaum Abweichungen in der Uhrzeit entstehen.

Es wurde sich dafür entschieden, als kleinste Zeiteinheit Dezisekunden zu verwenden. Zur Anzeige der Uhrzeit ist dies allemal ausreichend, aber so kann dieser Timer zusätzlich dazu verwendet werden, die Tasterzustände abzufragen und wäre für eine etwaige Anwendung, wo Dezisekunden benötigt werden (bspw. Stoppuhr) vorbereitet.

Um den Timer also 10 mal pro Sekunde einen Interrupt erzeugen zu lassen, muss der Vergleichswert folgendermaßen gesetzt werden:  $CMP = \frac{Takt}{INT} = \frac{14745600Hz}{10\frac{1}{s}} = 1474560$ . Dieser Wert kann von einem 16-Bit Timer nicht erreicht werden, da  $2^{16} < 1474560$ . Deshalb muss auch hier ein Prescaler verwendet werden. Bei einem Prescaler von 1024 ergibt sich ein Wert von  $\frac{1474560}{1024} = 1440$ , welcher ideal ist, weil er einerseits innerhalb der 16-Bit Grenzen liegt und andererseits eine Ganzzahl ist, sodass genau bei jedem Interrupt eine Zehntelsekunde verstrichen ist.

Innerhalb der Timer1 Interruptroutine werden folgende Operationen ausgeführt:

1. Die Uhrzeitvariablen (decisec, sec, min, hour) erhöhen
2. Flag zum Messen der Helligkeit setzen
3. Die Zeit neu in das Array zum Zeichnen schreiben
4. Die Zustände der Taster abfragen

### **5.1.3. Empfangen der Uhrzeit - Timer2**

Der dritte Timer wird primär zum Empfangen der Uhrzeit verwendet, wird aber zusätzlich auch zur Tongenerierung für den Lautsprecher benutzt, wenn ein Alarm aktiv ist.

Auch dieser Timer wird im CTC-Modus verwendet, der Vergleichswert aber je nach Anwendung geändert. Beim Empfangen der Uhrzeit ist die Auflösung 10 ms und damit der Vergleichswert =  $\frac{Takt}{Prescaler * INT} = \frac{14745600Hz}{1024 * 100\frac{1}{s}} = 144$ . Innerhalb der Interruptroutine werden nun statusabhängige Funktionen zum Empfangen der Uhrzeit aufgerufen. Eine genauere Betrachtung dieser Funktionen ist in Kapitel 4.2 zu finden.

Bei der Alarmfunktion wird der Wert abhängig der gewünschten Tonhöhe gesetzt.

#### **5.1.4. Nichtzeitkritische Funktionen in der main-Methode**

Alle Funktionen, bei denen die strikte zeitliche Einhaltung nicht wichtig ist, sind in der Hauptschleife des Programmes, einer `while(1)`-Schleife in der `main`-Methode zu finden.

Dort werden das Messen der Temperatur, sowie der Helligkeit vorgenommen. Außerdem werden dort Events behandelt, die beim Drücken der Taster auftreten sollen, wie das manuelle Umstellen der Zeit oder der Helligkeit und zu guter Letzt die Verifizierung und Konvertierung der in Interruptroutine von Timer2 empfangenen Uhrzeit. Zusammenfassend also alles Funktionen, die für den Betrieb der Digitaluhr wichtig sind, bei denen es aber nicht darauf ankommt, wann genau sie aufgerufen werden, sondern nur, dass sie aufgerufen werden.

## **5.2. Hardware**

### **5.2.1. Hauptplatine**

Das Herzstück der Uhr ist die nachfolgend abgebildete Hauptplatine. Sie enthält den Mikrocontroller, einige wichtige Komponenten sowie die Anschlüsse für alle externen Komponenten. Unter der Abbildung sind die verwendeten Bauteile kurz beschrieben.

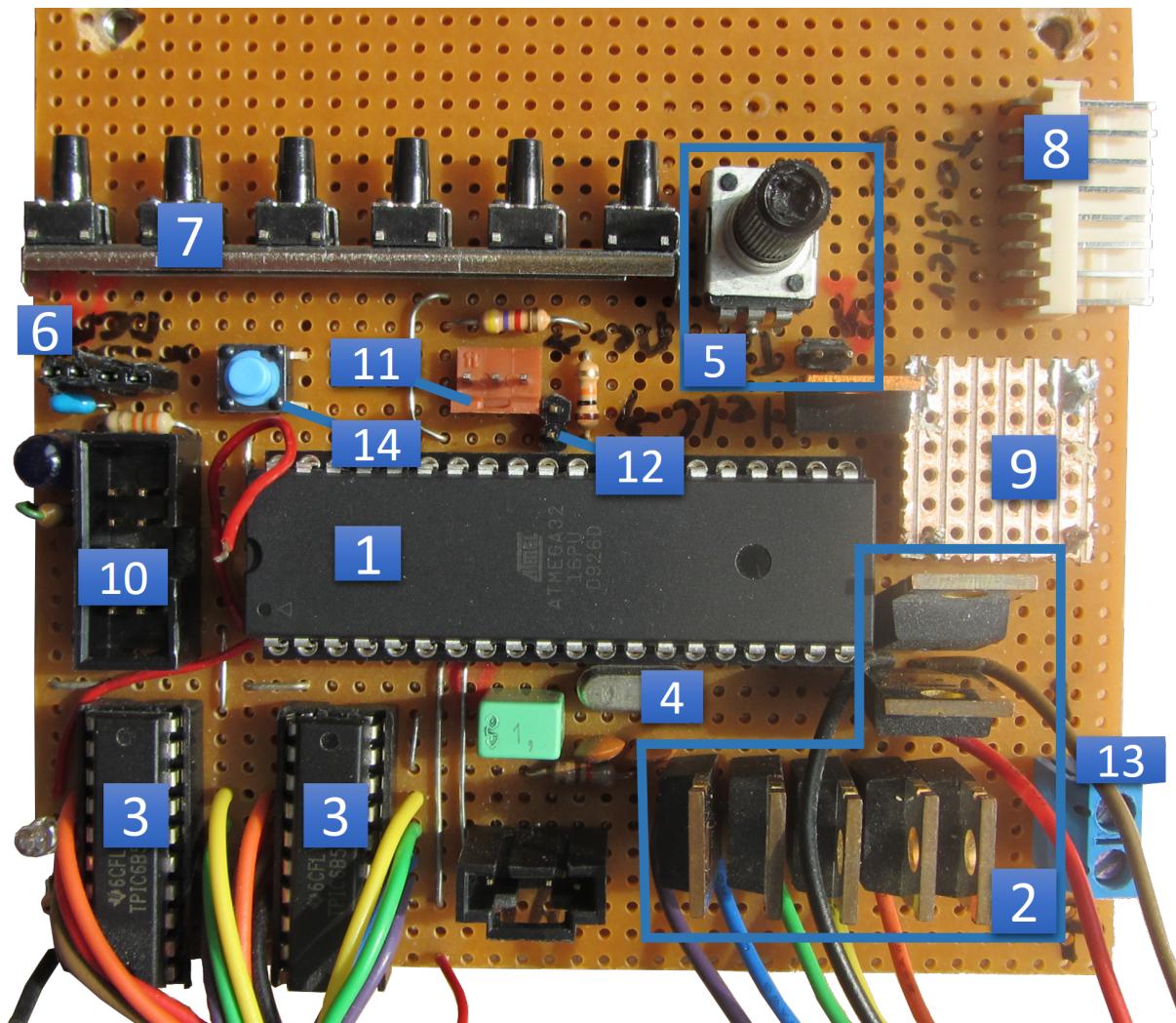


Abb. 15: Hauptplatine der Digitaluhr

- 1. Mikrocontroller ATmega32** Zentrale Steuereinheit
- 2. Mosfet IRLZ24N** zum Ansteuern der Kathoden der LED-Matrix
- 3. Schieberegister TPIC6B595** zum Ansteuern der Anoden der LED-Matrix
- 4. 14,7456 MHz Quarz** als Taktgeber für den Mikrocontroller
- 5. Potentiometer** für die Lautstärkeinstellung der Weckfunktion
- 6. Anschluss DCF77-Empfänger** für den Zeitempfang
- 7. interne Taster** für verschiedene Einstellungsmöglichkeiten (bspw. Helligkeit)

- 8. Anschluss für externe Taster** mit gleicher Belegung wie interne Taster
- 9. Lagesensor** zum Erkennen, wie die Uhr aufgehängt ist
- 10. ISP Schnittstelle** In-system programming zum Programmieren des Mikrocontrollers
- 11. Anschluss Temperatursensor DS18S20**
- 12. Anschluss Fotowiderstand** zum Messen der Helligkeit
- 13. Anschluss Stromquelle** für 5 Volt Gleichspannungsversorgung
- 14. Resettaster** für den vollständigen Reset des Mikrocontrollers

## 6. Résumé

### 6.1. Evaluation

TODO

### 6.2. Weiterentwicklungsmöglichkeiten

Die Grundfunktionalität der Uhr besteht nun und auch einige erweiterte Funktionen, wie beispielsweise Temperatur- und Helligkeitsmessung sind implementiert. Damit stellt das Projekt eigentlich einen abgeschlossenen Zustand dar. Dennoch gibt es natürlich Erweiterungsmöglichkeiten. Es ist vorstellbar, eine zukünftige Version mit Infrarotempfang auszustatten. So könnte man nicht nur die Zeit eingeben, die Helligkeit kontrollieren oder einen Wecker vom Bett aus bedienen, auch einfache Spiele wie das bekannte Pong wären denkbar.

### 6.3. Fazit

Diese Version der Uhr deckt viele Funktionalität ab. Wie aber im Kapitel Weiterentwicklungsmöglichkeiten erläutert wird, gibt es dennoch Raum für Erweiterungen, sodass eine zweite Version der Uhr durchaus denkbar wäre.

## **Literatur**

- [Ins05] INSTRUMENTS, TEXAS: *TPIC6B595 - POWER LOGIC 8-BIT SHIFT REGISTER.* <http://www.ti.com/lit/ds/slis032a/slis032a.pdf>, 2005.
- [mi10] INTEGRATED MAXIM: *DS18S20 - High-Precision 1-Wire Digital Thermometer.* <http://datasheets.maximintegrated.com/en/ds/DS18S20.pdf>, 2010.
- [PD04] PIESTER D., HETZEL P., BAUCH A.: *Zeit- und Normalfrequenzverbreitung mit DCF77.* PTB-Mitteilungen 114, Heft 4, 2004.

## **A. Quellcode**

TODO