

Facharbeit im Seminarfach Informatik

Dekodierung und Nutzung des DCF77- Zeitsignals

Verfasser: Marvin Thielking

E-Mail: mail@marvin-thielking.de

März 2010

für den Download angepasste Fassung

Inhaltsverzeichnis

1	Einleitung	1
2	Die Plattform.....	1
2.1	Hardware	1
2.2	Software.....	2
3	Das DCF77-Zeitsignal	3
4	Implementierung	5
4.1	DCF77-Dekodierung	5
4.2	Weitere Funktionen	8
4.2.1	Softclock	8
4.2.2	Benutzerschnittstelle	8
4.2.3	Wecker	10
4.2.4	Temperaturanzeige.....	10
4.2.5	Pufferung der Stromversorgung.....	11
4.3	Hardware	12
5	Resümee.....	12
6	Abbildungsverzeichnis	13
7	Literaturverzeichnis	13
8	Anhang	14
8.1	Tabelle: Aufbau des DCF77-Telegramms	14
8.2	Stückliste	15

1 Einleitung

Diese Arbeit beschäftigt sich mit der Dekodierung des DCF77-Signals und der darauf aufbauenden Realisierung eines Weckers. Das Projekt wird auf Basis eines Atmel¹ 8-Bit Mikrocontrollers durchgeführt.

Da die Zielsetzung der Arbeit ein funktionsfähiger Wecker ist, werden neben der Dekodierung noch eine Ausgabemöglichkeit, ein Wecksignal, ein Bedienkonzept sowie Lösungen bei Ausfall des Funkzeitsignals oder der Stromversorgung benötigt.

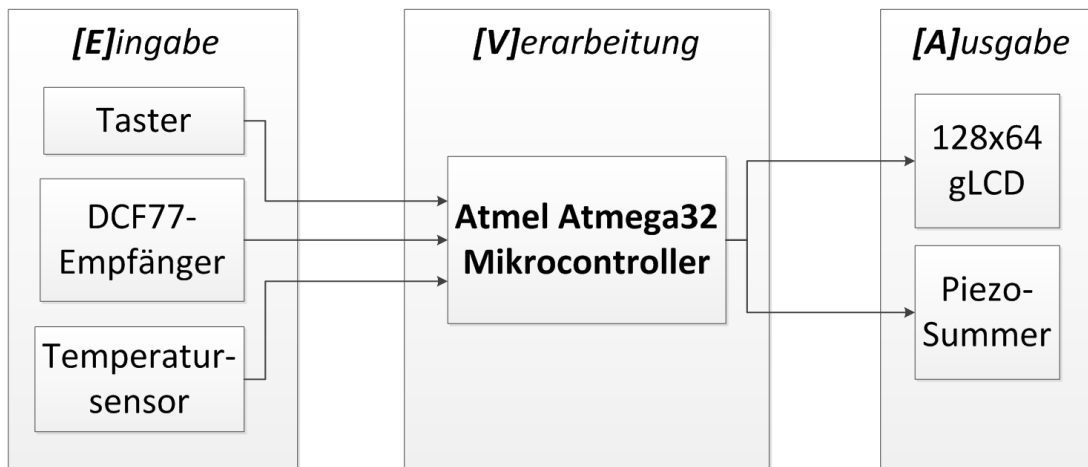


Abbildung 1: Blockdiagramm der Komponenten

Aus Abbildung 1 ergibt sich der grundsätzliche Aufbau, auf den im Folgenden näher eingegangen wird.

2 Die Plattform

2.1 Hardware

Mikrocontroller (μC) finden sich in der heutigen Zeit in tausenden elektronischen Geräten des Alltags. Von MP3-Playern über Roboter bis hin zu Alarmanlagen und Motorsteuerungen bieten sie die Möglichkeit komplexe Aufgaben mit wenigen elektronischen Bauteilen zu realisieren. Im Unterschied zu (Mikro-)Prozessoren sind Komponenten wie Speicher und Ein-/Ausgänge auf einem Chip vereint. Moderne

¹ US-amerikanischer Hersteller von Integrierten Schaltungen

Mikrocontroller haben sogar komplexe Funktionen wie Analog-Digital-Wandler und USB-/RS232 Schnittstellen integriert.

Auf dem Markt ist ein breites Spektrum von Controllern erhältlich. Sie unterscheiden sich in ihrer Bauform, dem Funktionsumfang und dem Preis. Im Hobbyumfeld wäre ein Controller mit 128 Pins im SMD-Gehäuse² zum Beispiel äußerst unpraktisch.

Für den Funkwecker fiel die Wahl letztendlich auf einen 8-Bit-AVR-Controller der Firma Atmel; die Entwicklung begann auf einem ATmega8 und wurde dann aus Pin-Mangel auf einem ATmega32 fortgesetzt. Die Entscheidung zugunsten der ATmega Baureihe war in der geringen Einstiegshürde (es wird kein teures Programmiergerät benötigt), der guten Bauform (PDIP-Gehäuse³) sowie der weiten Verbreitung im Hobby-Sektor und damit einer guten Unterstützung begründet.

Der ATmega32 bietet 32 kB Flash-Speicher, 2 kB RAM sowie 1 kB EEPROM⁴, er operiert bei einer Spannung von 4,5 – 5,5 V mit maximal 16 MHz (in diesem Projekt wird ein 8 MHz Quarz als Taktgeber verwendet). Der μ C besitzt 40 Pins, wovon 32 als Ein- und Ausgänge frei nutzbar sind. Als eingebaute Peripheriefunktionen, die in diesem Projekt genutzt werden, seien zwei 8-Bit und ein 16-Bit Timer genannt. Außerdem wird die PWM⁵ Funktionalität und zwei externe Interrupts benutzt⁶.

2.2 Software

Ein weiterer Vorteil der AVR's ist die hohe Verfügbarkeit von kostenfreien oder verhältnismäßig günstigen Compilern. So ist die Programmierung in Assembler oder den Hochsprachen C und Basic möglich. Die Programmierung in Basic, bzw. genauer gesagt mit der proprietären Software Bascom⁷, führt anfangs zu schnellen Erfolgen, allerdings ohne Kenntnisse über die interne Funktionsweise des Controllers zu gewinnen. Assembler ist das totale Gegenteil dessen, jedes Register⁸ muss direkt

² engl. surface-mounted device, oberflächenmontierbares Bauelement, im Gegensatz zu bedrahteten Bauelementen im Hobbyumfeld komplizierter zu handhaben

³ engl. Dual in-line package (in Plastikvergießung)

⁴ engl. Electrically Erasable Programmable Read Only Memory, nicht flüchtiger Speicher

⁵ Pulsweitenmodulation, ermöglicht das Generieren einer analogen Spannung durch schnelles Ein- und Ausschalten

⁶ Technische Daten: vgl. (4)

⁷ Entwickler: MCS Electronics, URL: <http://www.mcselec.com/>

⁸ 8 Bit breiter Speicherbereich innerhalb des AVR

angesprochen werden – man arbeitet also sehr nah am Maschinencode und benötigt genaue Kenntnis über den Mikrocontroller. Dieses hardwarenahe Programmieren bringt einen Performancegewinn mit sich, außerdem hat der Programmierer die volle Kontrolle. Dies wird jedoch leider bzw. eigentlich natürlich mit einem starken Komfortverlust erkauft. Das gilt vor allem für etwas größere Projekte. Für diese Facharbeit wurde aus genannten Gründen ein kostenloser C-Compiler (WinAVR⁹) genutzt. So wird zwar nicht die maximal mögliche Performance erreicht, aber das ist in diesem Fall auch nicht entscheidend und zugunsten der größeren Übersichtlichkeit vertretbar.

Als Entwicklungsumgebung wird das AVRStudio der Firma Atmel in der Version 4.18 genutzt. Die Programmierung des AVR erfolgt über das MyAVR ProgTool in der Version 1.20 mit dem mySmartUSB MK2 ISP¹⁰.

3 Das DCF77-Zeitsignal

DCF77 ist ein Langwellensender in Mainflingen bei Frankfurt. Seine Bezeichnung ergibt sich aus „D“ für Deutschland, „C“ für Landwellensender und „F“ für Frankfurt, 77 weist auf die Frequenz (77,5 kHz) hin. Der Sender wird von der Physikalisch Technischen Bundesanstalt (PTB) in Kooperation mit der Deutschen Telekom AG betrieben und sendet seit 1973 die genaue Uhrzeit und das Datum aus. Die übermittelte Zeit stellt die offizielle Zeit in Deutschland dar (außerdem werden Informationen über das Wetter und den Katastrophenschutz übermittelt)¹¹. Empfangen werden kann das DCF77 Zeitsignal im Umkreis von 2000 km, also nahezu in ganz Europa.

Die Verbreitung über Langwelle dauert mit mindestens einer Minute für 58 Bit sehr lange und ist störanfällig, diese Nachteile haben Satellitensysteme wie das amerikanische GPS und das europäische Pendant Galileo nicht. Ihr entscheidender Nachteil ist aber der zwingende Sichtkontakt zum Himmel, ein Empfang in Gebäuden ist also nicht möglich. So zeigt sich die PTB zuversichtlich über die

⁹ URL: <http://winavr.sourceforge.net/>

¹⁰ engl. In-System-Programmer, der AVR wird seriell programmiert, ohne ihn aus dem Zielsystem entfernen zu müssen

¹¹ vgl. (13)

Zukunft von DCF77: „Die Zeitübertragung über Satelliten und die Zeitverbreitung auf Langwelle werden sich daher nicht gegenseitig ersetzen sondern ergänzen.“ (1).

Der plötzliche Wegfall des DCF77-Zeitsignals hätte weitreichende Konsequenzen für das öffentliche Leben: So setzen Rundfunk- und Fernsehanstalten ebenso auf DCF77 wie die Bahn, Telekommunikationsdienstleister oder auch Energieversorgungsunternehmen z.B. für ihre Tarifschaltuhren. Die Betreiber scheinen diese Wichtigkeit erkannt zu haben und so wurde jüngst in die Sendeanlagen investiert sowie der Vertrag zwischen PTB und der Telekomtochter T-Systems bis 2013 verlängert¹².

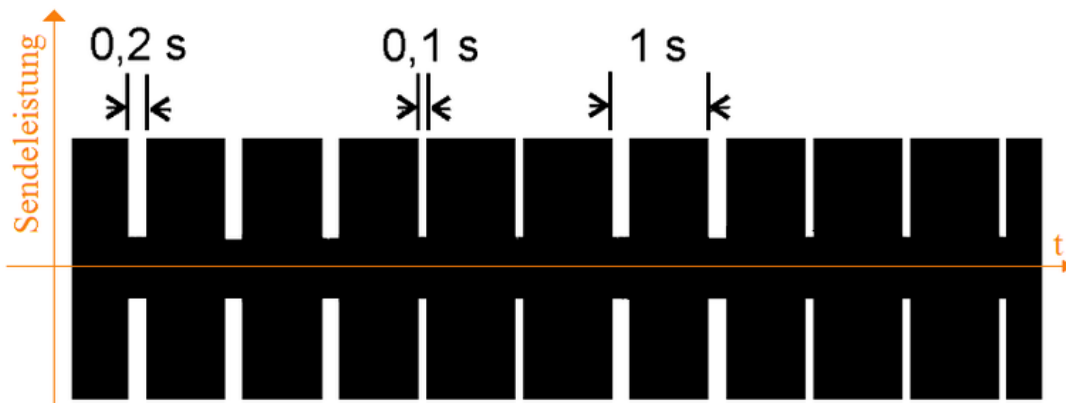


Abbildung 2: DCF77-Signalschema (2)

In Abbildung 2 ist der schematische Aufbau des DCF77-Signals zu erkennen. Im Sekundentakt wird die Sendeleistung auf 25% reduziert, die Dauer dieser Absenkung bestimmt das jeweilige Bit im Datentelegramm (200 ms steht für eine logische Eins, 100 ms für eine Null). Die Absenkung findet jeweils zu Beginn der Sekunden 0 bis 58 statt. Die fehlende Absenkung bei Sekunde 59 ermöglicht eine Synchronisation des Empfängers.

Somit ergibt sich eine Länge von 58 Bit für das Datentelegramm, die Kodierung ist im Anhang näher erläutert.

Es ergibt sich zum Beispiel dieses Telegramm:

```
0xxxxxxxxxxxxxxxx0010100010100100110110001000101000000010001
01234567890123456789012345678901234567890123456789012345678
0          1          2          3          4          5
```

¹² vgl. (5)

Die unteren Zahlen dienen nur zur Orientierung der Bit-Nummern. Es ist Winterzeit, da Bit 18 gesetzt ist. Bei den Minuten ist Bit 24 und 28 gesetzt; $8 + 20 = 28$. Da die Anzahl der gesetzten Bits gerade ist und es sich bei der Parität um eine sogenannte Even-Parity¹³ handelt, ist Bit 28 Null. Die Bits 29, 32 und 33 sind gesetzt, es ist also $1 + 8 + 10 = 19$ Uhr. Da drei Bits und damit eine ungerade Anzahl gesetzt sind, ist das Prüfbit 35 Null.

Es ist also 19:28 MEZ, nach diesem Schema lässt sich das gesamte Telegramm dekodieren.

4 Implementierung

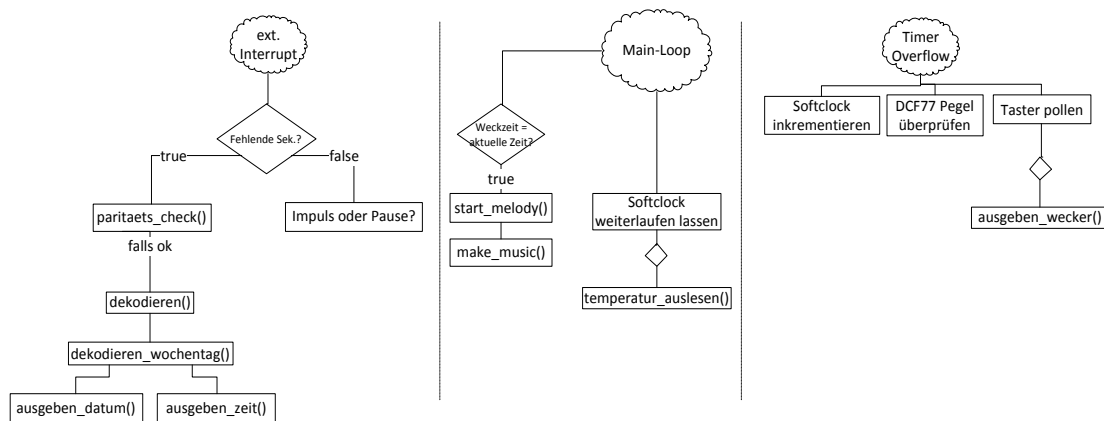


Abbildung 3: Programmstruktur

4.1 DCF77-Dekodierung

Zum Empfang des Langwellensignals wird eine Empfängerplatine der Firma Conrad¹⁴ genutzt. Bei abgesenktem Pegel liegen an ihrem Ausgang 0 V an (low, logisch 0) und bei voller Sendeleistung VCC (5 V, high, logisch 1). Sie bietet außerdem einen invertierten Ausgang. Für den Funkwecker wird der invertierte Ausgang mit dem externen Interrupt 1 (INT1) des AVR verbunden (da es sich um einen sog. Open-Kollektor Ausgang handelt, ist außerdem noch ein Widerstand gegen VCC geschaltet¹⁵). Außerdem wird der 16-Bit Timer so konfiguriert, dass alle 10 ms ein Overflow-Interrupt ausgelöst wird. In der Interrupt Service Routine (ISR)

¹³ Prüfbit ist bei ungerader Anzahl an Einsen im Datenwort Eins

¹⁴ Conrad Electronic SE, deutscher Elektronik-Versandhandel

¹⁵ vgl. (11), mit dem internen Pull-Up-Widerstand des AVR realisiert

des Timers wird je nach Status des DCF-Signals eine Variable hochgezählt: Bei High-Pegel wird `Dcf_impuls` inkrementiert, ansonsten `Dcf_pause`.

Bei fallender Taktflanke am Signalpin ist eine Sekunde vergangen und der externe Interrupt wird ausgelöst. In dessen ISR wird `Dcf_sekunde` inkrementiert und die Variablen werden ausgewertet: War der Impuls ~ 200 ms, und damit `Dcf_impuls` bei ~ 20 ($20 \cdot 10 \text{ ms} = 200 \text{ ms}$), wird eine Eins in den Array `Inhalt[Dcf_sekunde]` vermerkt, ansonsten eine Null. Sollte die Pausenvariable entsprechend hochgezählt sein, signalisiert dies den Minutenwechsel und das aufgezeichnete Telegramm kann ausgewertet werden. Außerdem wird dann `Dcf_sekunde` auf Null gesetzt, um dem Beginn einer neuen Minute Rechnung zu tragen.

Bei der Auswertung werden zuerst die Paritätsbits überprüft. Hierzu wird zunächst die Anzahl der Einsen im Datenwort bestimmt (hier für die Minuten):

```
for (uint8_t i = 22; i<=28; i++) {
    Paritaet = Paritaet + Inhalt[i];
}
```

Um nun zu bestimmen ob, die Variable `Paritaet` gerade oder ungerade ist, wäre eine Modulo-Division mit Zwei möglich. Bei einer geraden Anzahl an Einsen wäre das Ergebnis Null, ansonsten Eins. Aus Performancegründen wird statt dessen der bitweise And Operator (&) genutzt, hier beispielhaft die Funktionsweise:

0101 (5)	0010 (2)	0011 (3)
AND 0001	AND 0001	AND 0001
= 0001	= 0000	= 0001

Das Ergebnis ist also gleich dem des Modulo Operators bei deutlich besserer Performance¹⁶. Im Programm wird der so erzeugte Wert einfach mit dem empfangenen Paritätsbit verglichen:

```
if ((Paritaet & 1) != Inhalt[29])
```

Sollte die Prüfung erfolgreich gewesen sein (also die Bedingungsabfrage false), beginnt die eigentliche Dekodierung. Sollte sich das Datentelegramm beim Paritätscheck als ungültig erweisen, wird es komplett verworfen. Mit der eingesetzten, eindimensionalen Parität ist nur die Erkennung bestimmter Übertragungsfehler möglich (das korrekte Datenbit 0011000 erzeugt die gleiche

¹⁶ vgl. (6)

Parität wie z.B. 0001100, obwohl hier ein Übertragungsfehler vorliegt). Auch eine Korrektur ist leider nicht möglich. Dazu wäre mindestens eine Blockparität, oder besser noch Verfahren wie das des Hamming-Codes nötig. Dann müssten aber mehr Prüfbits übertragen werden, beim Hamming Code zum Beispiel für die sieben Minutenbits zusätzlich vier Paritätsbits.

Die Dekodierung gestaltet sich relativ simpel, hier wieder exemplarisch für die Minuten:

```
for (uint8_t i = 22; i<=28; i++) {
    Dcf_minute = Inhalt[i] * Wertigkeit[ii] + Dcf_minute;
    ii++;
}
```

Der Array `Wertigkeit[]` enthält nur die Wertigkeit der einzelnen Bits (1, 2, 4, 8, 10, 20, 40, 80), im Array `Inhalt[]` ist das eingelesene Datentelegramm gespeichert (im Vergleich zur Tabelle im Anhang jeweils um eine Position verschoben, Bit 22 entspricht Bit 21 in der Tabelle usw.). In der Schleife werden also die Minuten-Bits durchlaufen, ihr Wert (Null oder Eins) mit der Wertigkeit multipliziert und dann zur gesamten Minutenvariable addiert (sie wird an anderer Stelle im Programm natürlich vorher auf Null gesetzt). Parallel zur eigentlichen Schleifenvariable `i` wird auch die Variable `ii` hochgezählt, um die Wertigkeit anzupassen (`Inhalt[22]` hat die Wertigkeit `Wertigkeit[0] = 1`, da LSB¹⁷).

Zur Auswertung der Bits 21 bis 28 aus dem obigen Beispielerogramm 0001010 würde das Durchlaufen der Schleife also $0 * 1 + 0 * 2 + 0 * 4 + 1 * 8 + 0 * 10 + 1 * 20 + 0 * 40 = 28$ ergeben.

Zur Sicherheit wird nun noch die Plausibilität der Werte überprüft (Stunden von 0 bis 23, Minuten von 0 bis 59 usw.). Sollte auch diese Überprüfung erfolgreich sein, werden die dekodierten Werte übernommen und ausgegeben. Um eine noch höhere Sicherheit vor Übertragungsfehlern zu gewährleisten, könnte man das empfangene Telegramm noch mit dem Vorherigen vergleichen. Wenn die Minuten aktuell $0001010 = 28$ sind, müssen die unmittelbar danach empfangenen $1001010 = 29$ sein.

¹⁷ Bit mit der geringsten Wertigkeit steht links

4.2 Weitere Funktionen

4.2.1 Softclock

Das DCF-Signal kann sehr leicht, z.B. durch Oberwellen¹⁸ von Netzteilen, gestört werden (im Versuchsaufbau machte der ca. einen Meter entfernte LCD Fernseher den DCF77-Empfang unmöglich). Aus diesem Grund wurde eine interne, vom Funksignal unabhängig laufende Uhr implementiert, eine sogenannte Softclock. Da bereits für die DCF-Auswertung ein Timer aktiv ist, wird dieser mitgenutzt. In der Interrupt-Routine wird so alle 10 ms eine Variable inkrementiert. Hat sie den Wert 100 erreicht ($1\text{ s}/10\text{ ms}$), ist eine Sekunde vergangen. Über simple Bedingungsabfragen werden dann gegebenenfalls auch Minuten, Stunden und Tage/Wochentage weitergezählt. Auf eine komplett autark laufende Datumsanzeige wurde aus einer Kosten/Nutzen-Überlegung heraus verzichtet. Schließlich hätte man dann auch Schaltjahre usw. implementieren müssen. Sobald ein intaktes DCF-Telegramm empfangen wurde, wird die Softclock mit diesem aktualisiert.

Aber auch bei immer andauerndem Funkempfang wäre eine solche Maßnahme notwendig. Würde man nämlich nur die DCF-Zeit ausgeben, würde die 59. Sekunde fehlen. Dieses Problem wird so also auch umgangen.

4.2.2 Benutzerschnittstelle

4.2.2.1 Bedienkonzept: Eingabe über Taster

Da in diesem Projekt eine Funkuhr entstehen soll, wurde auf eine Eingabemöglichkeit der Uhrzeit verzichtet. Es ist also nur eine Möglichkeit zur Eingabe der Weckzeit und zum Aktivieren/Deaktivieren der Weckfunktion nötig. Für diese Aufgabe wurden drei Taster gewählt, die softwareseitig entprellt¹⁹ werden.

Zwei Taster bieten die Möglichkeit die Stunde bzw. Minute der Weckzeit einzustellen. Das Gedrückthalten der Taster ist möglich. Der dritte Taster stellt bei

¹⁸ Überlagerte Schwingungen. Frequenz ist ein ganzzahliges Vielfaches der Grundfrequenz. Erzeuger von Oberwellen: Transformatoren, Drosseln, Gleichstrommotoren etc.

¹⁹ Die mechanischen Taster prellen beim Drücken und Loslassen für kurze Zeit, d.h. sie schalten sich schnell ein- und aus und nehmen nicht sofort den gewünschten Zustand ein. Ursache ist eine Art mechanisches Nachschwingen/Vibrieren, in dessen Konsequenz es zu softwareseitigen Fehlern kommen kann. Durch Entprellung (hier: softwareseitig) wird das verhindert - vgl. (10)

langem Druck die Weckfunktion ein oder aus. Ein kurzer Druck, während der Wecker klingelt, aktiviert die Snooze-Funktion (nochmaliges Klingeln in 5 Minuten).

4.2.2.2 Ausgabe auf einem grafikfähigen LCD

Zur Ausgabe wurde ein 128x64 px aufgelöstes LCD-Panel der Firma Displaytech²⁰ genutzt. Es hat bereits einen KS0108 kompatiblen Controller integriert. Die Ansteuerung erfolgt über 8 Datenleitungen mithilfe einer, von Andre Fabricius veröffentlichten Library. Die Library wurde so modifiziert, dass auch die Ausgabe von weißer Schrift auf schwarzem Grund möglich ist. Der Hintergrund wurde mit einem Grafikprogramm erstellt und mit Hilfe des Programms Image2GLCD konvertiert. Die Bitmap wird dann in der Init-Routine geladen. Hierbei ergaben sich zunächst

diverse Probleme, da in der ursprünglichen Library nur die gesetzten Pixel ausgegeben werden. Die nicht gesetzten Pixel werden aber nicht gelöscht. Des Weiteren haben die Buchstaben unterschiedliche Breiten. Durch diese beiden Phänomene entstehen so



Ausgabefehler (Abbildung 4). Wie die Abbildung zeigt, überschreiben sich die Zeichen gegenseitig. Erster Lösungsansatz wäre das LCD komplett zu leeren, das Hintergrundbild neu zu laden und dann die jeweilige Ausgabe zu tätigen. Durch die hohe Trägheit des LCDs ist das aber kein gangbarer Weg. Daher wurde die Library so modifiziert, dass bei Ausgabe eines Zeichens alle betreffenden Pixel aktualisiert werden, also bei weißer Schrift auch die Schwarzen und bei schwarzer Schrift auch die Weißen. Die zweite Fehlerquelle, also die unterschiedliche Breite der Buchstaben, wurde beseitigt, indem einfach alle Zeichen die gleiche Breite bekommen haben. Zu diesem Zweck wurde das Programm GLCDFontCreator2.1 genutzt. Im gleichen Arbeitsschritt wurden auch nicht genutzte Zeichen aus der Schriftart entfernt, um Speicherplatz zu sparen. Es wird die Schriftart Arial in fetter Formatierung und in den Größen 14, 22 und 24 px genutzt.

²⁰ DISPLAYTECH Ltd, auf Grafikmodule spezialisiertes Unternehmen mit Sitz in Hongkong

4.2.3 Wecker

Die Weckzeit wird, wie oben beschrieben, über Taster eingegeben. Sollte die Weckzeit mit der aktuellen Zeit übereinstimmen und der Wecker aktiviert sein, wird eine Variable (`Wecker_klingelt`) gesetzt und der Weckton ausgelöst. Das Klingeln kann über einen Taster komplett deaktiviert werden. Eine weitere Funktion ist Snooze. Hierbei klingelt der Wecker in 5 Minuten erneut. Während Snooze aktiv ist, sind die Taster zum Einstellen der regulären Weckzeit gesperrt und auf dem Display wird die Uhrzeit des nächsten Klingelns angezeigt.

4.2.3.1 Melodieklingel

Zur Ausgabe der Melodie wurde ein kleines Script genutzt. Es werden zwei Timer verwendet, wobei einer im CTC-Modus²¹ betrieben wird und so die Frequenz des Tons bestimmt. Der zweite Timer regelt in einer Overflow-Interrupt-Routine die Länge des Tons sowie eventuelle Pausen. Das Script wurde so angepasst, dass die Melodie in einer Endlosschleife läuft. Außerdem wurde die vorgegebene Melodie durch eine eigene Melodie ersetzt („Alle meine Entchen“).

Zur Ausgabe wird ein Piezo-Summer benutzt. Er überzeugt durch eine geeignete Lautstärke bei geringer Größe. Der eingeschränkte Frequenzbereich ist im vorgesehenen Anwendungsszenario kein wirklicher Nachteil.

4.2.4 Temperaturanzeige

Zur Bestimmung der Umgebungstemperatur wird der Temperatursensor LM335 genutzt, er ist günstig und mit einer Genauigkeit von ca. einem Grad Celsius im nicht kalibrierten Zustand auch genau genug. Er verändert die Ausgangsspannung pro Kelvin Temperaturänderung um 10 mV, also linear²². Diese Linearität ist ein weiterer Vorteil gegenüber temperaturabhängigen Widerständen.

Die Auswertung der Spannung am LM335 erfolgt über den Analog-Digital-Wandler (ADC) des AVR. Der ADC kann Spannungen zwischen 0 V und der

²¹ Clear Timer on Compare Match Modus, der Timer löst bei einem vorher definierten Zählerstand einen Interrupt aus und nicht erst bei einem Overflow

²² vgl. (12)

Referenzspannung (hier 5 V) mit einer Genauigkeit von 10 Bit bestimmen²³, d.h. der ADC liefert einen Wert von 0 bis 1024, wobei 0 = 0 V und 1024 = Referenzspannung. Die Genauigkeit seitens des ADC beträgt also hier $\frac{5\text{ V}}{1024} = 0,004883\text{ V}$. Zur Bestimmung der Spannung muss der vom ADC gelieferte Wert also mit 0,004883 multipliziert werden.

Da an dem Sensor 10mV pro Kelvin anfallen, lässt sich die Temperatur nun leicht in Abhängig vom ADC-Rückgabewert (x) bestimmen:

$$^{\circ}\text{C} = \frac{(0,004883 * x) * 1000}{10} - 273 = 0,4883 * x - 273$$

4.2.5 Pufferung der Stromversorgung

Zur Spannungsversorgung ist ein Netzteil gewählt worden, dessen Ausfall abgedeckt werden muss. Schließlich würde sonst ein kurzer, nächtlicher Stromausfall zum Verlust der Weckzeit führen, eine äußerst unschöne Situation. Der AVR hat einen eingebauten EEPROM Speicher, dessen Inhalt bei Spannungsausfall nicht verloren geht (sog. nicht flüchtiger Speicher). Leider ist der EEPROM nur ungefähr 1.000.000-mal beschreibbar²⁴. So ist es also kein gangbarer Weg die Variablen permanent in den EEPROM zu sichern. Mit viel Aufwand lässt sich dieser Wert bei geringen Datenmengen zwar ausbauen, indem man in verschiedene Zellen schreibt (Ring-Puffer), aber eine schöne Möglichkeit ist das immer noch nicht. Um die Variablen nur bei Netzverlust sichern zu müssen, war es vorgesehen die Spannungsversorgung mit einem Kondensator zu puffern. Der Kondensator sollte parallel in die Spannungsversorgung eingesetzt werden. Über eine Diode sollte verhindert werden, dass Strom zurück ins Netzteil bzw. den Spannungswandler fließt. Bricht die Spannung vor der Diode ein, sollte ein externer Interrupt ausgelöst werden und sämtliche relevanten Variablen in den EEPROM gesichert werden (Weckzeit, Uhrzeit, ...). Leider war dieses Vorhaben nicht praktikabel. So betrug die Durchlassspannung an der eingesetzten Halbleiterdiode bei 10 mA über 0,7 V, so dass die Gesamtspannung dadurch unter die Mindestversorgungsspannung des AVR (4,5 V) gefallen wäre. Eine mögliche Lösung wäre hier eine höhere Spannung vor der Diode (mit dem eingesetzten Festspannungsregler nicht möglich) oder evtl. eine

²³ vgl. (4)

²⁴ vgl. (8)

Schottky-Diode gewesen. Weitere Probleme ergaben sich durch den hohen Innenwiderstand des vorgesehenen Goldcap Kondensators.

So wurde das Vorhaben der Pufferung als nicht praktikabel aufgegeben. Statt dessen werden nun die Weckzeit und der Status des Weckers (An, Aus) nach jeder Veränderung an Ihnen in den EEPROM gesichert. Es ergibt sich der gleiche Nutzen bei deutlich geringerem Aufwand.

4.3 Hardware

Wie bereits erwähnt sind durch den Mikrocontroller sehr wenig elektrische Bauteile benötigt worden. Während der Entwicklungsphase wurde ein Steckbrett genutzt. Danach wurde mit der Software Eagle²⁵ ein Schaltplan erstellt. Der finale Aufbau erfolgte auf einer Lochrasterplatine, aus Zeitgründen wurde auf eine geätzte Platine verzichtet. Als Gehäuse war ein Modell aus Aluminium mit Plexiglasfront vorgesehen. Leider stellte sich in der Endphase des Projekts heraus, dass der Aluminiumrahmen den Empfang des DCF77-Signals unmöglich macht. Alternativ wird nun ein Gehäuse aus Polycarbonat genutzt.

5 Resümee

Die eingangs genannte Zielsetzung, ein funktionierender Funkwecker, wurde aus meiner Sicht erfüllt. Während der Arbeit an dem Projekt sind zwar diverse Schwierigkeiten aufgetaucht, welche aber allesamt gelöst werden konnten. Obwohl die Elektronik nicht sehr komplex war, ergaben sich in diesem Bereich die meisten Probleme, vor allem im Zusammenhang mit dem Funkempfang. Gerade diese „Stolpersteine“ (vor allem nicht reproduzierbare Fehler) zwangen einen geradezu intensiv mit der Materie auseinanderzusetzen – und das ist äußerst positiv und lehrreich gewesen.

Bei einem solch praxisnahen Projekt hat sich ein gutes Zeitmanagement als besonders wichtig erwiesen. Verzögerungen sind durch das Nachbestellen neuer Teile mehrmals eingetreten. So sind in der Experimentierphase der DCF-Empfänger und auch das Display defekt gewesen und mussten ersetzt werden.

²⁵ Entwickler: CadSoft Computer GmbH, URL: <http://www.cadsoft.de/>

6 Abbildungsverzeichnis

Abbildung 1: Blockdiagramm der Komponenten	1
Abbildung 2: DCF77-Signalschema (2).....	4
Abbildung 3: Programmstruktur	5
Abbildung 4: Ausgabefehler	9

7 Literaturverzeichnis

1. **PTB.** Physikalisch-Technische Bundesanstalt (PTB). [Online] [Zitat vom: 28. 01 2010.] http://www.ptb.de/de/org/4/44/442/pcf77_1.htm.
2. **Weidner, Herbert.** *Amplitudenmodulierte Sendeleistung als Funktion der Zeit.* [Grafik] 2007.
3. **Displaytech Ltd.** *LCD64128 Datenblatt.* [Dokument] 2010.
4. **Atmel.** *ATmega32 / ATmega32L Datenblatt.* [Dokument] 2010.
5. **Linum Software GmbH.** DCF77 - Informationen zum Zeitsignal. [Online] [Zitat vom: 04. 02 2010.] <http://www.dcf77.de/dcf77-information-zum-zeitsignal/>.
6. **Walther, Jan.** Feststellen, ob eine Zahl ungerade/gerade ist: PHP Performance. [Online] 08. 08 2007. [Zitat vom: 14. 02 2010.] <http://phpperformance.de/feststellen-ob-eine-zahl-ungerade-gerade-ist/>.
7. **cplusplus.com.** itoa - C++ Reference. [Online] [Zitat vom: 02. 02 2010.] <http://www.cplusplus.com/reference/cstdlib/itoa/>.
8. **Schwarz, Andreas.** *mikrocontroller.net.* [Online] [Zitat vom: 15. 02 2010.] <http://www.mikrocontroller.net>.
9. **Pressel, Jens.** Grafik LC-Display (kurz GLCD) Bauanleitung mit Schaltplan u.v.m. *OverClocked inside.* [Online] [Zitat vom: 28. 01 2010.] http://www.ocinside.de/index_d.html.
10. **"SprinterSB".** Taster-Abfrage in C: RN-Wissen. [Online] [Zitat vom: 29. 01 2010.] http://www.rn-wissen.de/index.php/Taster-Abfrage_in_C.

11. **Conrad Electronic.** *Anschlussplan DCF Empfängerplatine BN 641138.* [Dokument] 2010.
12. **National Semiconductor.** *LM135/LM235/LM335, LM135A/LM235A/LM335A Precision Temperature Sensors.* [Dokument] 17. 12 2008.
13. **Dirk Piester, Peter Hetzel, Andreas Bauch.** Zeit- und Normalfrequenzverbreitung mit DCF77. *PTB Mitteilungen 114.* 2004, Heft 4.

8 Anhang

8.1 Tabelle: Aufbau des DCF77-Telegramms²⁶

Bit Nr.	Bedeutung
0	Beginn einer neuen Minute (immer 0)
1-14	Informationen des Katastrophenschutzes sowie verschlüsselte Wetterdaten
15	für interne Zwecke der PTB (Rufbit)
16	Kündigt die Umstellung zwischen Sommerzeit (MESZ) und Winterzeit (MEZ) am Ende der laufenden Stunde mit einer 1 an
17-18	18 = 1 und 17 = 0 => MEZ, 18 = 0 und 17 = 1 => MESZ
19	Kündigt eine Schaltsekunde ²⁷ am Ende der laufenden Stunde an
20	Beginn der Zeitübermittlung (immer 1)
21-27	Minute BCD kodiert => 1, 2, 4, 8, 10, 20, 40
28	Paritätsbit (Minute)
29-34	Stunde => 1, 2, 4, 8, 10, 20
35	Paritätsbit (Stunde)
36-41	Tag => 1, 2, 4, 8, 10, 20
42-44	Wochentag, wobei 1 = Montag und 7 = Sonntag => 1, 2, 4
45-59	Monat => 1, 2, 4, 8, 10
50-57	Jahr => 1, 2, 4, 8, 10, 20, 40, 80; nur Zehner und Einer Stelle (2010 => 10)
58	Paritätsbit (Datum)

²⁶ vgl. (5), (1), (13), (8)

²⁷ korrigiert aus der Erdrotation stammende Zeitabweichung

8.2 Stückliste

Anzahl	Bezeichnung	Bestellnummer (Conrad/Reichelt)	Preis (€)
1	Spannungsregler IC 7805 TO 220	179205	0,57
1	DCF Empfängerplatine	641138	10,21
1	mini Potentiometer 100R linear liegend	422380	0,27
1	mini Potentiometer 10K linear liegend	422444	0,27
2	keramischer Kondensator 22pF	457167	0,26
1	Quarz 8.000MHz HC49	155191	0,68
1	Temperatursensor IC LM 335	176656	1,92
4	Kunststoffbolzen 2x Innen M3 20mm	534773	0,88
1	100er Linsenkopfschrauben DIN7985 M3 6mm	815322	1,73
1	Messerleiste mit geraden Lötstifen 10 polig lowprofile	742512	0,80
1	IC-Fassung 40 polig	189588	0,52
1	ATMega AVR-RISC-Controller, DIL-40	ATMEGA 32- 16 DIP	4,30
1	Piezo-Schallwandler	SUMMER BM 15B	0,70
1	Grafik Modul mit Beleuchtung	LCD 64128A LED	16,70
1	Lochrasterplatine, Hartpapier, 160x10	H25PR160	1,90
3	Kurzhubtaster 6x6mm, Höhe: 7,0mm, 12V, vertikal	TASTER 9303	0,36
4	Vielschicht-Keramikkondensator 100N, 20%	Z5U-5 100N	0,16
1	Elko, axial, 10µF/35Volt	AX 10/35	0,13
1	Planar Epitaxial Schaltodiode, DO35, 100V, 0,15A	1N 4148	0,02
1	Kohleschichtwiderstand 1/4W, 5%, 10 K- Ohm	1/4W 10K	0,10
1	Kohleschichtwiderstand 1/4W, 5%, 2,2 K- Ohm	1/4W 2,2K	0,10
SUMME:			<u>40,06</u>
<i>Zusätzlich benötigt: Netzteil, ISP (z.B. Conrad #191510) und Schaltdraht</i>			