

CONTROLLED CLOCK

Synchronisation der Uhrzeit via DCF77 Signal

Version 1.0

Softwareprojekt 2
Zürcher Hochschule für Angewandte Wissenschaften

Michael Hadorn und Daniel Brun
Informatik 4. Semester

26. Mai 2014

Inhaltsverzeichnis

| | | |
|----------|--|-----------|
| 1 | Einleitung | 1 |
| 1.1 | Aufgabenstellung | 1 |
| 1.2 | Umsetzungsidee | 1 |
| 2 | Projektmanagement | 3 |
| 2.1 | Team | 3 |
| 2.2 | Betreuung | 3 |
| 2.3 | Tools | 3 |
| 2.4 | Planung | 4 |
| 2.4.1 | Meilensteine | 4 |
| | Analyse abgeschlossen | 4 |
| | Server / API umgesetzt | 4 |
| | Auslesen und Interpretation der Daten des Empfängers abgeschlossen | 4 |
| | Client GUI Implementation abgeschlossen | 5 |
| | Umsetzung, Dokumentation und Projekt abgeschlossen | 5 |
| 2.5 | Rückblick | 6 |
| 2.5.1 | 1. Abgabe - 24.03.2014 | 6 |
| | Aktueller Planungsstand | 6 |
| | Ausblick 2. Abgabe | 6 |
| 2.5.2 | 2. Abgabe - 14.04.2014 | 7 |
| | Aktueller Planungsstand | 7 |
| 2.5.3 | 3. Abgabe - 05.05.2014 | 8 |
| | Aktueller Planungsstand | 8 |
| 2.5.4 | 4. Abgabe (finale) - 31.05.2014 | 9 |
| 3 | Analyse | 11 |
| 3.1 | Allgemein | 11 |
| 3.1.1 | Hintergründe | 11 |
| | Kurzer Rückblick in die Geschichte | 11 |
| 3.1.2 | Atomuhr | 12 |
| 3.1.3 | Funkuhr | 13 |
| 3.1.4 | Koordinierte Weltzeit | 13 |

| | | |
|----------|--|-----------|
| 3.2 | Möglichkeiten der Zeitsynchronisation | 13 |
| 3.2.1 | Funk | 13 |
| 3.2.2 | Zeitsignaldienste | 14 |
| 3.2.3 | Zeitzeichensender - Allgemein | 14 |
| 3.2.4 | Sendeanlage | 14 |
| | Codierung & Übermittlung | 14 |
| | Ausgangssignal | 14 |
| | Frequenznormale | 15 |
| | Zeitzeichensender in der Schweiz | 16 |
| 3.2.5 | Zeitzeichensender - DCF77 | 17 |
| | Das Rufzeichen | 17 |
| | Signalerzeugung | 17 |
| | Signalreichweite | 18 |
| | Fehleranfälligkeit (Sender & Empfänger) | 18 |
| | Genauigkeit | 18 |
| | Das Signal | 19 |
| | Codierung | 20 |
| | BCD-Codierung | 24 |
| | Zusatznutzung | 24 |
| 3.2.6 | Fehlererkennung beim Empfänger | 24 |
| 3.2.7 | Fehlererkennung beim Sender | 24 |
| | Network Time Protocol | 24 |
| 3.2.8 | GPS | 24 |
| 3.2.9 | Internet | 25 |
| 3.3 | Zeitprotokoll | 25 |
| | Beispiel Synchronisation | 26 |
| 3.4 | NTP / SNTP | 27 |
| 3.5 | Weitere Zeitsignaldienste | 27 |
| 3.6 | Fazit | 27 |
| | 3.6.1 Codierung der Zeitinformationen einer Minute | 27 |
| | 3.6.2 Fehlererkennung und Fehlerkorrektur | 27 |
| 4 | Dokumentation | 29 |
| 4.1 | Übersicht der Struktur | 29 |
| 4.1.1 | Server Seite | 29 |
| 4.1.2 | Client Seite | 30 |
| 4.2 | Modul-Dokumentation | 30 |
| 4.2.1 | DCF Reader | 30 |
| 4.2.2 | DCF Decoder | 30 |
| 4.2.3 | Clock | 31 |
| 4.2.4 | Simple Web Server | 32 |
| 4.3 | System-Voraussetzungen | 33 |
| 4.4 | Installationsanleitung - DCF77-Treiber | 33 |
| 4.4.1 | OSX | 33 |

| | |
|--------------------------------------|-----------|
| 4.4.2 Windows | 33 |
| 4.5 Betrieb | 33 |
| 4.5.1 Server | 33 |
| 4.5.2 Client | 33 |
| 4.6 Screenshots | 34 |
| 4.6.1 Server | 34 |
| 4.6.2 Client | 35 |
| 5 Schlusswort | 37 |
| 5.1 Aktueller Projektstand | 37 |
| 5.1.1 Fazit | 37 |
| 5.1.2 Reflexion | 37 |
| 5.1.3 Danksagung | 37 |
| Quellenverzeichnis | 39 |

KAPITEL 1

Einleitung

1.1 Aufgabenstellung

Wie funktioniert die Zeitsynchronisation zwischen verschiedenen Geräten? Was für Schwierigkeiten und Probleme gibt es dabei? - Das soll am Beispiel einer computerbasierten Funkuhr gezeigt werden. Ziel dieser Arbeit ist es die Daten von einem DCF77-Empfänger abzugreifen und den Spezifikationen entsprechend zu interpretieren. Die empfangene Zeit wird via einem minimalen Network Time Protocol Server verschiedenen Client-Anwendungen (Web, Desktop) zur Verfügung gestellt.

1.2 Umsetzungsidee

Vom DCF77-Empfänger werden die Daten ausgelesen und gemäss den Spezifikationen interpretiert. Diese Daten werden anschliessend auf einem zu implementierenden Server bereitgestellt. Die Daten können einerseits via einem minimalen Network Time Protocol und andererseits via einem normalen Service abgerufen werden. Dadurch kann die Problemstellung bei der Zeitsynchronisation gezeigt werden. Als optionales Feature könnte auf dem Server eine Möglichkeit geboten werden, eine künstliche Last, bzw. eine Verzögerung, zu simulieren, damit die Problemstellung besser ersichtlich wird.

Auf der Client-Seite werden drei verschiedene Uhren dargestellt. Eine wird via (S)NTP synchronisiert und eine normal, via Web. Die dritte Uhr wird auf Knopfdruck via NTP synchronisiert und lässt sich manuell verstellen. Eine erfolgreiche Synchronisation wird durch eine aufleuchtende LED und einen Timestamp angezeigt. Es findet in regelmässigen Zeitintervallen eine Synchronisation statt. Wurde auf der Server-Seite eine Zeitsynchronisation angezeigt, wird diese auch im Client angezeigt.

Die gesamte Implementation erfolgt in C. Dies hat den Grund, dass gewisse Möglichkeiten in den höheren Programmiersprachen wie Java nicht, bzw. nur sehr umständlich realisierbar sind. Auf der Server-Seite wird eine eigenständige Uhr implementiert, welche sich auf den System-Tic abstützt. Der System-Tic wird ca. 18.2 mal pro Sekunde erhöht, was die Implementation einer sehr präzisen Uhr ermöglicht.

KAPITEL 2

Projektmanagement

2.1 Team

Das Projektteam besteht aus folgenden Personen:

- **Michael Hadorn**
E-Mail: michael.hadorn@gmail.com, ZHAW-Kurzzeichen: hadormic
- **Daniel Brun**
E-Mail: daniel.brun@gmx.net, ZHAW-Kurzzeichen: brundan1

Da unser Projektteam sehr übersichtlich ist legen wir nicht von Anfang an fixe Rollen innerhalb des Projektes fest. Einzelne Rollen werden sich im Laufe der Zeit von selbst herauskristallisieren.

2.2 Betreuung

Unsere Projektarbeit wird durch folgende Personen betreut:

- **Syrus Mozafar**
Bereich: Methodik & Planung
E-Mail: moza@zhaw.ch, ZHAW-Kurzzeichen: moza
- **George Brügger**
Bereich: Technik & Inhalt
E-Mail: irs.and.e@gmail.com

2.3 Tools

Für das Projektmanagement werden folgende Tools eingesetzt:

- Planung: <http://www.comindwork.com>
Projekt: <http://controlledclock.comindwork.com/>
- Versionsverwaltung: <http://www.github.com>
Repository: <https://github.com/MrJack91/controlledClock>
- Projektablage: <https://drive.google.com>

2.4 Planung

Die Grobplanung der Meilensteine und einzelnen Task wurden zu Beginn des Projektes gemacht. Die Detailplanung wird jeweils am Anfang des bearbeiteten Meilensteins und der Arbeitspakete gemacht. So können wir aus gemachten Erfahrungen und Fehlern lernen und die Planung immer dem wirklichen Projektfortschritt anpassen. So haben wir auch die Möglichkeit zu reagieren, falls wir sehen, dass gewisse Aufgaben länger brauchen als ursprünglich geplant. Dann werden wir eine entsprechende Repriorisierung der Tasks und gegebenenfalls der Meilensteine durchführen.

2.4.1 Meilensteine

Es wurden folgende Meilensteine definiert:

- 1. Abgabe (24.03.2014)
 - Analyse abgeschlossen
- 2. Abgabe (14.04.2014)
 - Server / API umgesetzt
 - Auslesen und Interpretation der Daten (Teil 1)
- 3. Abgabe (05.05.2014)
 - Auslesen und Interpretation der Daten des Empfängers abgeschlossen
- 4. Abgabe (31.05.2014)
 - Client GUI Implementation abgeschlossen
 - Umsetzung, Dokumentation und Projekt abgeschlossen

Analyse abgeschlossen

Mit diesem Meilenstein wird die Grundlage für das gesamte Projekt gelegt. Es wird eine detaillierte Analyse der Aufgabenstellung und den Themenbereichen Zeitsynchronisation per Funk und im Internet, Network Time Protocol und weiteren angrenzenden Bereiche durchgeführt. Die Ergebnisse der Analyse dienen als Grundlage für die konkrete Umsetzung der späteren Meilensteine.

Server / API umgesetzt

Der Server, bzw. die API zum Bezug der Daten wird als eigenständiges Arbeitspaket angesehen und somit auch als Meilenstein definiert. Es werden die Schnittstellen der API definiert und umgesetzt. Via API muss das aktuelle Datum und das Datum der letzten erfolgreichen Zeitsynchronisation über den DCF77 abrufbar sein.

Auslesen und Interpretation der Daten des Empfängers abgeschlossen

Dieser Meilenstein ist die Grundlage des gesamten Projektes und beinhaltet das Arbeitspaket zum Auslesen und Interpretieren der Daten, die vom Funkempfänger geliefert werden. Der Empfänger musste zuerst beschafft und getestet werden, daher erfolgte die Abgabe dieses Meilensteins erst relativ spät gegen Ende des Projektes. Ein weiterer Grund ist die Komplexität der Anforderung die Daten via einem mitgeliefertem Treiber von der Schnittstelle auszulesen.

Client GUI Implementation abgeschlossen

Der Meilenstein mit dem Arbeitspaket zur Implementation des Client GUIs wird erst bei der Projektabgabe abgegeben. Für uns ist die Implementation eines perfekten GUIs zweitrangig und wir möchten uns auf den Kern des Projektes fokussieren. Stellen wir im Verlauf des Projektes fest, dass wir uns bei der Aufwandschätzung verschätzt haben oder unvorhergesehene Schwierigkeiten auftauchen, werden wir Abstriche beim Client GUI machen, um die Kernfunktionalität so weit als möglich fertig zu stellen. Auch beim GUI liegt der Fokus zuallererst auf der Funktionalität und erst im Anschluss auf Zusatzfeatures und Effekten.

Umsetzung, Dokumentation und Projekt abgeschlossen

Mit diesem Meilenstein wird die Umsetzung, die Dokumentation und somit das ganze Projekt abgeschlossen und zur Bewertung eingereicht.

2.5 Rückblick

2.5.1 1. Abgabe - 24.03.2014

Während der ersten Iteration haben wir uns intensiv mit der Analyse unserer Aufgabenstellung und der dazugehörigen Themen beschäftigt.

Die Analyse der Themen gestaltet sich als sehr spannend. Bald sahen wir uns auch dem Problem gegenüber, dass wir nicht bei allen Themen alle Details im Rahmen dieses Projektes zusammentragen konnten. Daher mussten wir uns an gewissen Stellen mit einer weniger detaillierten Fassung begnügen. Die zusammengetragenen Informationen haben unser Verständnis enorm verbessert und bilden eine solide Grundlage für die weiteren Meilensteine in unserem Projekt.

Bei den Themen *Network Time Protocol* und *Probleme & Schwierigkeiten* haben wir uns entschieden nicht ganz so viel Zeit zu investieren wie geplant war. Die *Probleme & Schwierigkeiten* wurden zum Teile bereits in anderen Kapiteln abgedeckt und wird auch im Fazit nochmals angeschnitten, daher haben wir das Kapitel nicht so ausführlich gestaltet. Beim *Network Time Protocol* haben wir uns auf eine vereinfachte Variante beschränkt, da es in seiner Gesamtheit sehr komplex ist, würde dies in Kombination mit den übrigen Zielstellungen den Rahmen dieser Projektarbeit sprengen.

Aktueller Planungsstand

Für den ersten Meilenstein stand relativ wenig Zeit zur Verfügung. Nichtsdestotrotz waren wir in der Lage die geplanten Tasks in der zur Verfügung stehenden Zeit erfolgreich abzuschliessen. Auch die geschätzten Aufwände entsprachen etwa den tatsächlichen Aufwänden. Im Bereich NTP haben wir weniger Zeit gebraucht als ursprünglich geplant. Die Gründe dafür wurden oben bereits erläutert.

Ausblick 2. Abgabe

Da wir keine grossen Restanzen aus der ersten Iteration haben, gibt es auch keine direkten Konsequenzen auf die zweiten Iteration. Die Analyse hat einige Knackpunkte offen gelegt, die bei der Umsetzung berücksichtigt werden müssen. Bezüglich Dokument-Layout haben wir noch kleinere offene Punkte, welche wir im Laufe der nächsten Iterationen bereinigen werden.

2.5.2 2. Abgabe - 14.04.2014

Nach Abschluss der ersten Iteration hatten wir mit Syrus Mozafar und George Brügger je ein separates Treffen bezüglich dem aktuellen Projektstand. Aus beiden Treffen konnten wir wertvolle Inputs für den weiteren Projektverlauf mitnehmen.

Beim Treffen mit George Brügger sind wir zum Schluss gekommen, dass wir ein Replanning und weitere Verfeinerung der Aufgabenstellung machen müssen. Die Dokumentation und der Zeitplan wurden entsprechend angepasst. Auch kamen wir zum Schluss, dass eine Implementation des Backend in C unumgänglich ist. Da wir beide noch nie grössere, bzw. komplexere Programme in C geschrieben haben, stellt dies für uns eine weitere Herausforderung dar. Als Konsequenz davon rechnen wir damit, dass wir nicht alle (Zusatz-)Funktionen wie geplant umsetzen können.

Aktuell stellt vor allem das Auslesen des System-Tics eine grosse Hürde für uns dar. Nach sehr intensiven und langwierigen Recherchen und einigen Dutzend ausprobierten Beispielcodes sind wir zum Schluss gekommen, dass der System-Tic nur über eine Interrupt Service Routine ausgelesen werden kann. Die Interrupt Service Routine wiederum muss als Treiber implementiert werden, da auf die entsprechenden Bereiche des Betriebssystems aus dem User-Mode heraus nicht zugegriffen werden kann. Die Ausnahme bildet hier eine allfällige Implementation in Assembler.

Da wir beide über unterschiedliche Notebooks und Betriebssysteme (Windows 7 und Mac OS X) verfügen, haben wir einen Versuch mit einem Raspberry Pi gestartet. Mit dem Raspberry verfügen wir über die gleichen Voraussetzungen und haben im Detail die Kontrolle, bzw. das Wissen, über die verwendete Hardware. So können wir uns bei der Implementation auch auf ein Betriebssystem und eine Hardware beschränken. Niemand von uns hat bisher jemals auf diesen Ebenen programmiert. Dies ist eine spannende Herausforderung, jedoch ist Ungewiss, ob wir unser Ziel auch wirklich erreichen können.

Auch beim „Read-“ und „Decode-Teil“ haben wir uns ein wenig überschätzt. Zur Zeit der ersten Planung wussten wir noch gar nicht, ob wir das Backend in C oder Java umsetzen würden. Das Read und Decode war einiges komplizierter als wir uns das vorgestellt haben. Da war zum Beispiel die fortlaufende Suche nach korrekten Zeitstempeln. Momentan haben wir zur Definition eines korrekten Zeitstempels „nur“ die Kontrolle der Parität eingebaut; deswegen erkennen wir fast viermal mehr Zeitstempel, als es gültige Zeitstempel im Signal hat. Dies muss verbessert werden.

Aktueller Planungsstand

Auch der Termin für die zweite Abgabe des Meilensteins ist schneller da gewesen, als wir uns das gedacht hatten. Vor allem, da wir etwas mit der Programmiersprache zu kämpfen hatten. Vom String-Handling, über Speicherallozierung bis hin zu Problemen mit Libraries, die es nur für bestimmte OS-Versionen, OS-Typen oder Compiler gibt. Die Probleme waren grundsätzlich sehr spannend und herausfordernd. Jedoch mussten wir in die Problemfindung einige Zeit investieren.

2.5.3 3. Abgabe - 05.05.2014

Zu Beginn der Iteration hatten wir ein erneutes Treffen mit Syrus Mozafar. Dabei haben wir festgestellt, dass wir das mindestens geforderte Arbeitsvolumen bereits erreicht haben.

Während dieser Iteration haben wir uns damit beschäftigt, die angefangenen Komponenten soweit als möglich zu finalisieren. Dabei mussten wir feststellen, dass wir aufgrund der unterschiedlichen Plattformen weiterhin Probleme hatten. Einerseits hatten wir von Anfang an mit einem grösseren Aufwand gerechnet und zum Anderen hatten wir während dieser und der letzten Iteration erhebliche Verzögerungen aufgrund von Problemen und Herausforderungen.

In Anbetracht der Auslastung durch das Studium und der aktuellen Projektsituation werden wird diese Woche mit Herrn Brügger zusammensitzen und die Ziele des Projektes nochmals überarbeiten und den Gegebenheiten angepasst. Dabei werden wir wahrscheinlich vor allem im Bereich GUI einige Abstriche in Kauf nehmen müssen.

Nach dem Treffen, haben wir uns definitiv dafür entschieden die eigenständige Uhr mit Hilfe eines Timers o.Ä. zu implementieren. Die Anbindung an den System-Tic wäre zu aufwändig gewesen und hätte den Rahmen des Projektes gesprengt. Die Gründe dafür wurden bereits im Abschlussbericht der letzten Iteration und in der Dokumentation beschrieben.

Die Implementation des WebServers ist abgeschlossen und funktioniert auf allen Plattformen einwandfrei. Die Daten des DCF77-Empfängers können empfangen, decodiert und mit der eigenständigen Uhr synchronisiert werden. Die Zeit der Uhr kann über eine Webservice-Schnittstelle abgerufen werden. Aktuell stehen wir noch vor der Herausforderung alle Komponenten auf beiden Plattformen (Windows und Mac OS X) gleichermassen zum Laufen zu bringen.

Aktueller Planungsstand

Trotz der vielen freien Tage aufgrund von Ostern, Sechseläuten und 1. Mai war auch diese Iteration rasch vorüber. Abgesehen von den Schwierigkeiten mit den verschiedenen Plattformen verlief die Iteration relativ ruhig. Die Planung mussten wir nochmals an die aktuellen Gegebenheiten anpassen. Auch einige Aufwandsschätzungen mussten wir nochmals revidieren.

Während dem Projekt gibt es immer wieder spannende und interessante Herausforderungen zu bewältigen. Auch wenn dies zum Teil etwas zeitaufwändig und nervenaufreibend ist, haben wir immer noch Spass am Projekt und können uns laufend neue und interessante Dinge aneignen.

2.5.4 4. Abgabe (finale) - 31.05.2014

Entgegen der Erwartungen konnten wir in der letzten Iteration noch einige Erfolgserlebnisse verzeichnen. So haben wir es endlich geschafft, dass alle Module auf beiden Plattformen (Windows und Mac OS X) kompiliert und ausgeführt werden können. Die Ursachen lagen in einigen Details bei der Implementation und an einem falschen Include, der nicht bemerkt wurde, da die Fehlermeldung zum einen erst während dem Linken auftrat und zum anderen eine Inkompatibilität (OS X / Architektur) der zu linkenden Treiber-Bibliotheken suggerierte.

Als wir die fertig empfangene Zeit im Client mit einer Atomuhr verglichen, bemerkten wir einen Zwei-Sekunden unterschied. Der beruhte auf einem Fehler der zwischen den verschiedenen Modulen „Decoder“ und „Clock“ hin und her geschoben wurde. Der fertige Zeitstempel für die folgende Minute wird jeweils in der Sekunde 58 decodiert. Nun haben wir die 2 Sekunden Wartezeit nirgends mehr beachtet. Sprich das fertige Signal wurde direkt der Clock als neue, präzise Uhrzeit übergeben. Dies haben wir sofort gefixt.

Beim Client läuft alles nach Planung. Jegliche Basis Features wurden umgesetzt. Ange-dachte Ideen bezüglich SNTP/NTP Vergleich und Server Simulation (Mock up) wurden nicht umgesetzt. Auch haben wir keine Animation beim ändern der analogen Uhrzeit eingebaut.

KAPITEL 3

Analyse

3.1 Allgemein

Wieso braucht es eine Zeitsynchronisation? Was für Probleme und Schwierigkeiten können dabei auftreten? Diese Fragen aus der Aufgabenstellung werden in den nachfolgenden Kapiteln detailliert beantwortet.

3.1.1 Hintergründe

In der heutigen Zeit sind wir es uns gewohnt, dass wir uns darauf verlassen können, dass unsere Uhren die gleiche, oder zumindest beinahe die gleiche Zeit anzeigen. Wie war das früher? Heute würde vieles nicht mehr funktionieren, wenn wir keine Uhren hätten, bzw. die Uhren nicht synchronisiert werden, oder etwa doch?

Kurzer Rückblick in die Geschichte

Seit langem ist die Menschheit an der Messung der Zeit, bzw. der Uhrzeit, interessiert. Seit dem Jahr 300 v. Chr., als die Sonnenuhr in Ägypten erfunden wurde, ergab es sich, dass an zentralen Plätzen solche Sonnenuhren aufgebaut wurden. Wollte man die Uhrzeit wissen, musste man den Weg zur Uhr in Kauf nehmen.¹ Da die Uhrzeit mit Hilfe des Sonnenstandes ermittelt wurde, gab es Abweichungen zwischen der gemessenen Zeit an verschiedenen Orten. Aufgrund dessen war es damals nicht möglich eine einheitliche Zeit für ein ganzes Reich oder Land zu bestimmen. Die Uhrzeit war jeweils nur lokal pro Stadt oder Gebiet mehr oder weniger einheitlich.

Im Jahr 1248 wurde die erste mechanische Uhr in Exeter (England) in Betrieb genommen. Sie hatte eine Genauigkeit von einer Stunde und musste täglich mit Hilfe der Sonne kalibriert werden. Da ein grosses Interesse an der Uhrzeit bestand, wurden in weiteren Städten und Ländern relativ rasch weitere Uhren in Betrieb genommen. Man konnte sich die Uhrzeit abonnieren, das hiess, einmal am Tag kam jemand vorbei und teilte einem die aktuelle Uhrzeit mit, welche von einer mobilen Uhr abgelesen wurde.

1656 wurde die erste Pendeluhr entwickelt. 1680 war sie so ausgereift, dass sie einen Minutenzeiger erhielt. Das Problem dieser Technologie war, dass eine Pendeluhr nur an

¹ wikipedia.org. *Sonnenuhr*. URL: <http://de.wikipedia.org/wiki/Sonnenuhr> (besucht am 24.05.2014).

ruhigen Standorten funktionierte. Dies stellte besonders für die Schifffahrt ein Problem dar, denn auf See wurde die Uhrzeit zur Berechnung der aktuellen Position verwendet. Ein Fehler von 4 Sekunden entspricht bereits einer Ungenauigkeit von rund 1.8 km. Dies führte dazu, dass die britische Regierung, für eine Genauigkeit von vier Minuten, ein Preisgeld offerierte. Dies entsprach einer Ortungsgenauigkeit von ungefähr 100 km. John Harrison gelang es 1778 einen Chronometer (H4)¹ zu entwickeln, der diese Ansprüche erfüllte.²

Bis ins Jahr 1835 wurde die Zeitübermittlung durch akustische oder optische Signale ermöglicht. Mit dem Aufkommen der Eisenbahn und der somit verbundenen Anforderung, über grössere Strecke die gleiche Zeit zu halten, etablierte sich der Abgleich via elektronische Leitung.

Die ersten Gehversuche der drahtlosen Zeitübermittlung wurden im Jahr 1903 in Washington am United States Naval Observatory gemacht. Für die Seefahrt startete 1904 die erste regelmässige Zeitübermittlung via Funk. Sechs Jahre später wurde das erste Signal für auf dem Land, via Eiffelturm, versendet. Im Jahr 1917 hat der erste Vorläufer der heutigen Zeitzeichensender den Betrieb aufgenommen. Die Grosssendestelle Nauen (Bundesland Brandenburg DE) sendete zweimal im Tag via Langwelle ein Zeitzeichen. Daraufhin entstand im Laufe der Jahre ein weltweites Netz von Zeitzeichensendern.³

Erst im Jahr 1970 wurde der heutige Zeitübermittlungs-Dienst DCF77, in Mainflingen bei Frankfurt, in Betrieb genommen. Kurz darauf, im Jahr 1978, wurden der Betrieb des ersten GPS-Satelliten, mit einer eigenen Atomuhr an Board gestartet.^{4, 5}

3.1.2 Atomuhr

Mit einer Atomuhr wird Folgendes bezeichnet:

Eine Atomuhr ist ein Zeitmesser, dessen Zeitnormal die hochfrequente Schwingungsdauer bestimmter Atome ist (z.B. Caesium, Rubidium), die durch ein elektromagnetisches Feld oder optisches Pumpen zu Schwingungen angeregt werden und einen Quarzgenerator synchronisieren.

Die Atomuhr wurde vom Amerikaner Isidor Isaac Rabi (1898-1988) erfunden. Er wurde 1944 mit dem Nobelpreis belohnt.⁶

1 Die Bezeichnung Chronometer (altgriechisch für „Zeit“ und „Mass, Massstab“) steht für besonders präzise Uhren, welche früher zur Zeitbestimmung und zur Navigation auf Schiffen und Flugzeugen benötigt wurden.

2 wikipedia.org. *Chronometer*. URL: <http://de.wikipedia.org/wiki/Chronometer> (besucht am 24.05.2014).

3 wikipedia.org. *Geschichtliche Entwicklung der Zeitübertragung per Funk*. URL: http://de.wikipedia.org/wiki/Geschichtliche_Entwicklung_der_Zeit%C3%BCbertragung_per_Funk (besucht am 24.05.2014).

4 astro.siggi.de. *Die Zeit aus dem Äther - wie Funkuhren funktionieren*. URL: <http://www.astro-siggi.de/tutorial-funkuhr.html> (besucht am 24.05.2014).

5 wikipedia. *Time signal*. URL: http://en.wikipedia.org/wiki/Time_signal (besucht am 24.05.2014).

6 meinberg.de. *Atom Clock*. URL: http://www.meinberg.de/german/info/atomic_clock.htm (besucht am 24.05.2014).

3.1.3 Funkuhr

Im Gegensatz zur Atomuhr, kann eine Funkuhr nicht selbst die korrekte Zeit berechnen. Sie hat einen Funkempfänger eingebaut, der von einer „echten“ Atomuhr das Funksignal empfängt und daraus die genaue Uhrzeit annäheren kann. Der erstmalige Zeitabgleich dauert, je nach Synchronisationstyp, wenige Minuten. Sobald der Decode Prozess abgeschlossen ist, läuft die Uhr synchronisiert.

Die Genauigkeit einer Funkuhr liegt im Schnitt zwischen 5 und 25 msec.^{1, 2}

3.1.4 Koordinierte Weltzeit

Die koordinierte Weltzeit *Coordinated Universal Time (UTC)* bildet die Grundlage für die Zeitmessung auf der Erde. Total wird die Erde in 24 Zeitzonen unterteilt. Eine Zeitzone wird, basierend auf der Weltzeit, durch hinzufügen oder abziehen von Stunden definiert.

Neben UTC gibt es auch noch *Universal Time (UT1)*, welche im Gegensatz zu UTC, die Unregelmässigkeiten der Erdrotation beachtet. Bei UTC werden diese Schwankungen durch das Einfügen von Schaltsekunden ausgeglichen. Die Schaltsekunden werden in nicht gleichmässigen Abständen, analog der Unregelmässigkeiten der Erdrotation, eingefügt. Diese werden vom *Internationalen Dienst für Erdrotation und Referenzsysteme (IERS)* anhand der Messdaten festgelegt.

Die UTC wird von der *Temps Atomique International - Interkantonale Atomzeit (TAI)* abgeleitet und vom *Bureau International des Poids et Mesures - Internationales Büro für Mass und Gewicht (BIPM)* ermittelt. Als Grundlage dienen Referenzzeiten von ungefähr 250 Atomuhren, die über den ganzen Globus verteilt sind. Bei der Bestimmung des Zeitmittelwertes werden die Instabilitäten der einzelnen Uhren mittels einer Gewichtung berücksichtigt. Dieses Mittel wird als *Echelle Atomique Libre - Freie Atomzeitskala (EAL)* bezeichnet. Die EAL stimmt nicht mit der Definition der SI-Sekunde überein. Daher muss dies durch eine Anpassung der Frequenz korrigiert werden. Die so resultierende Zeit wird als Internationale Atomzeit bezeichnet.

3.2 Möglichkeiten der Zeitsynchronisation

Der heutige Stand der Technik bietet grundsätzlich zwei Möglichkeiten um eine Zeitsynchronisation vorzunehmen. Zum einen über Funk, zum anderen via Internet.

3.2.1 Funk

Elektromagnetische Wellen sind seit jeher allgegenwärtig. Es gilt hier natürliche und von Menschen erzeugte elektromagnetische Wellen zu unterscheiden. Natürliche elektromagnetische Wellen werden z.B. durch die Sonne, Gewitter oder Stürme erzeugt. Im Laufe der Geschichte haben die Menschen diese Wellen entdeckt und damit begonnen diese für ihre Zwecke einzusetzen. (Siehe Kapitel 3.1.1 - [Hintergründe](#)).

Die Zeitinformationen werden von Zeitsignaldiensten ausgesendet und können in der Regel kostenfrei genutzt werden. Um diese Signale auszusenden werden sogenannte Zeitzeichensender eingesetzt.

¹ wikipedia.org. *Funkuhr*. URL: <http://de.wikipedia.org/wiki/Funkuhr> (besucht am 24.05.2014).

² heret.de. *Funkuhr*. URL: <http://www.heret.de/funkuhr/index.htm> (besucht am 24.05.2014).

3.2.2 Zeitsignaldienste

Auf der Erde verteilt gibt es zahlreiche Zeitsignaldienste, die alle miteinander verbunden sind. Die Zeit der jeweiligen Dienste stimmt bis weit unterhalb von Nanosekunden miteinander überein. Jeder Dienst verwaltet die UTC (Coordinated Universal Time, Weltzeit) für ihr Land. Daneben steuern auch Messungen vom Internationalen Dienst für Erdrotation und Referenzsysteme (IERS) zur Berechnung der Weltzeit bei. Die Bestimmung der Weltzeit wird durch das Internationale Büro für Mass und Gewicht koordiniert.

3.2.3 Zeitzeichensender - Allgemein

Zeitzeichensender senden in bestimmten Zeitabständen codierte Zeitinformationen. Funkuhren und entsprechende Geräte können diese Signale empfangen, decodieren und interpretieren. Die Zeit und das Datum auf dem Gerät wird mit denen des Signales abgeglichen. Um eine erfolgreiche Synchronisierung zu erreichen, muss das Signal über eine gewisse Zeit empfangen werden. Vielfach liegt dem Signal mit den Zeitinformationen eine Normalfrequenz (Siehe Kapitel [3.2.4 - Frequenznormale](#)) zu Grunde.

3.2.4 Sendeanlage

Die Zeitzeichensender benötigen nicht zwingend eine spezielle Sendeanlage, vielfach werden die Signale über einen normalen Rundfunksender übermittelt. Jede Sendeanlage verfügt über eine eindeutige Identifikation um eine Zuordnung des Signals zu ermöglichen. Meistens ist das Signal an sich bereits eindeutig und kann einer Sendeanlage zugeordnet werden.

Codierung & Übermittlung

Das Signal wird entweder als Langwelle, Mittelwelle oder als Kurzwelle ausgesendet. An einigen Standorten werden sogar Längstwellen oder Ultrakurzwellen gesendet. Dies ist primär vom Einsatzort und -zweck abhängig. Aufgrund der Reichweite werden vorwiegend Langwellen eingesetzt.

Die Codierung des Signals ist nicht einheitlich, jedoch in den meisten Fällen ähnlich. Das Grundprinzip sieht vor, dass in einer laufenden Minute die Zeitinformationen für die nachfolgende Minute übermittelt werden. Da das Signal in den meisten Fällen im Sekundentakt (Russische Sender senden zum Teil im Zehntelsekundentakt) ausgestrahlt wird, stehen innerhalb einer Minute 60-Bit für die Übermittlung der Informationen zur Verfügung. Der Beginn einer Minute wird in der Regel bei Sekunde 0 / 59 durch eine spezielle Kennung markiert. Die Codierung ist so ausgelegt, dass die Signale maschinell einfach und effizient verarbeitet werden können.

Ausgangssignal

Das Signal wird am Sendestandort oder in der unmittelbaren Nähe von mehreren Atomuhren generiert und abgeglichen. Dadurch wird eine sehr hohe Genauigkeit erreicht. Wird die Distanz und die Signallaufzeit zwischen Sender und Empfänger berücksichtigt, ist die Genauigkeit im Nanosekundenbereich angesiedelt. Pro 300 km beträgt die Verzögerung ca. 1 ms.

Frequenznormale

Bis heute wird oft das Signal von Rundfunksendern (z.B. Radiostationen) als Frequenznormale ausgewertet. Vorallem in den 70er Jahren war der Einsatz der Frequenznormale weit verbreitet, da dies die einzige Möglichkeit für eine „Zeitsynchronisation“ war. Die Frequenznormale ist eine Vorstufe der heutigen Zeitzeichensenders (bzw. dem heutigen Signal). Es wird eine hochgenaue Schwingungsfrequenz eingesetzt (z.B. 150 kHz bei Langwellensendern oder 50 / 60 Hz im Stromnetz). Das Empfangsgerät musste einmal manuell auf die aktuelle Zeit eingestellt werden, anschliessend zählte dieses die Schwingungen des Signals, was jeweils einer Sekunde entsprach. Heute wird die Frequenznormale häufig als Grundlage für digitale Signale verwendet.

Zeitzeichensender in der Schweiz

In der Schweiz war zwischen 1966 und 2011 der Zeitzeichensender HBG in Prangins (VD) in Betrieb. Aufgrund mangelnder Nachfrage wurde der Sendebetrieb Ende 2011 eingestellt und die Sendeanlage gesprengt und abgebaut. Der Sender wurde vom Eidgenössischen Institut für Metrologie betrieben und sendete mit einer Frequenz von 75 kHz.



Fig. 3.1: Zeitzeichensender in Prangins (VD)
(Quelle: Eidgenössisches Institut für Metrologie (METAS); http://www.metas.ch/root_legnet/Web/Fachbereiche/Zeit_Frequenz/dissemination/HBG)

3.2.5 Zeitzeichensender - DCF77

DCF77 bezeichnet einen Zeitzeichensender in Deutschland (Mainflingen in Mainhausen) der von der Physikalisch-Technischen Bundesanstalt in Braunschweig (PTB) entwickelt wurde. Der Betrieb erfolgt durch die T-Systems Media Broadcast GmbH, einer Tochtergesellschaft der Deutschen Telekom AG.



(a) Antennenfeld (Quelle: Wikipedia, Patrick Kempf; <http://de.wikipedia.org/wiki/DCF77>)



(b) Sendemast (Quelle: Physikalisch-Technische Bundesanstalt; <http://www.ptb.de/de/aktuelles/archiv/presseinfos/pi2011/pitext/pi111107.htm>)

Abbildung 3.2: Sendeanlage in Mainflingen (Mainhausen, DE)

Das Rufzeichen

Die Bezeichnung setzt sich aus D für Deutschland, C für Langwellensender, F für Frankfurt (Mainhausen liegt in der Nähe von Frankfurt) und 77 für die Sendefrequenz (77.5 kHz). Früher wurde das Rufzeichen des Senders während der 20. bis 32. Sekunde der Minuten 19, 39 und 59 mitgesendet. Es wurde aber festgestellt, dass dies eine Verschlechterung des Signal-zu-Rausch-Abstandes zur Folge hatte. Seither wird das Signal ohne Rufzeichen ausgesendet. Das Signal des DCF77 kann auch ohne Rufzeichen eindeutig identifiziert werden.

Signalerzeugung

Das Signal wird von drei unabhängigen Steuereinheiten mit je einer Caesium-Atomuhr erzeugt (1 Hauptkanal, 2 Reservekanäle). Weicht das Signal des Hauptkanals von den Signalen der Reservekanäle abfindet automatisch eine Umschaltung auf einen Reservekanal statt. Sind auch diese beiden Signale nicht mehr identisch wird der Vorgang unterbrochen und es wird kein Signal ausgesendet. Jede der drei Einheiten verfügt über eine separate

Stromversorgung. Die drei Steuereinheiten sind in einem Speziellen Raum am Sendestandort untergebracht. Darin enthalten ist auch eine weitere Rubidium-Atomuhr, welche als Ersatz und Referenz zur Verfügung steht. Die gesamte Anlage kann via Fernzugriff gesteuert und gewartet werden.¹

Signalreichweite

Das DCF77 Signal hat je nach Wetterlage, Tages- und Jahreszeit eine Reichweite von 2'000 km. Es wird unterteilt in Bodenwellen, welche sich entlang der Erdoberfläche ausbreiten, und Raumwellen, welche sich durch Reflexion an der ionosphärischen D-Schicht ausbreiten. Bis zu einer Distanz von 500 km vom Sendeort ist die Bodenwelle sehr stabil und überwiegt den Anteil an Raumwellen. Im Bereich von 500 - 1'100 km ist das Verhältnis von Raum- und Bodenwellen etwa gleich gross. Dabei besteht die Gefahr der gegenseitigen Auslöschung. Ab einer Entfernung von 1'100 km nimmt der Anteil an Raumwellen laufend zu.

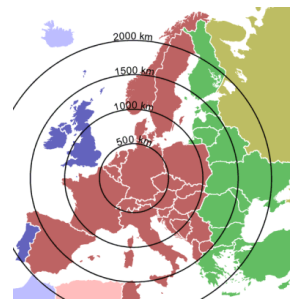


Fig. 3.3: Reichweite des Signals (Quelle: Compuphase; http://www.compuphase.com/mp3/h0420_timecode.htm)

Fehleranfälligkeit (Sender & Empfänger)

Signale welche mit Amplitudenmodulation können relativ einfach gestört werden, da die Amplitude verhältnismässig leicht von aussen beeinflusst werden kann.

Das Signal kann einerseits unterwegs oder am Empfangsort durch Gewitter, Stürme, Motoren oder elektronische Geräte gestört werden. Andererseits kann auch bereits beim Aussenden des Signals ein Fehler entstehen. Bei heftigen Gewittern, Stürmen oder starken Winden wird die Antenne teilweise vorübergehend ausser Betrieb gesetzt, da durch die Schwingung der Antenne eine messbare Phasenmodulation bei den Empfängern entsteht.

Genauigkeit

Das Signal erreicht beim Empfänger, bei optimierter Hardware und optimiertem Decodierungsalgorithmus, eine Genauigkeit von 100 μsec . Bei handelsüblichen Geräten wird jedoch in der Regel nur eine Genauigkeit von 0.1 Sekunden erreicht. Für den Hausgebrauch ist dies vollkommen ausreichend. In der Industrie ist diese Genauigkeit häufig nicht tragbar. Durch eine Erhöhung der Bandbreite des Empfängers kann die Abweichung reduziert werden.

¹ hopf.com. *Genauigkeitsvergleich DCF77 und GPS*. URL: http://www.hopf.com/de/dcf77-gps_de.html (besucht am 24.05.2014).

Dabei wird jedoch auch der Anteil an empfangenen Störsignalen grösser. Diese müssen von der Elektronik verarbeitet und herausgefiltert werden.

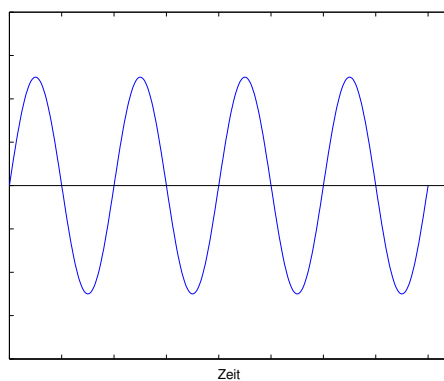
Für die Bestimmung der Zeit, erreichen die Atomuhren einen relativen Fehler von weniger als $2 \cdot 10^{-12}$ Sekunden. Als Referenz dienen dabei die Primären Atomuhren der PTB. Diese bestimmen die aktuelle Uhrzeit mit einem Fehler der zwischen $0.7 \cdot 10^{-14}$ bis $1.2 \cdot 10^{-14}$ Sekunden liegt.¹

Das Signal

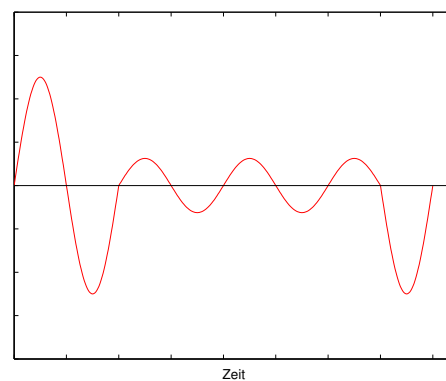
Seit 1973 wird neben der Normalfrequenz (77.5 kHz) ein digitales Signal gesendet. Das digitale Signal wird durch negative Modulation (Absenkung Trägeramplitude auf 25 %) auf die Trägerfrequenz aufgebracht und enthält die Informationen zu Datum und Uhrzeit (Mittleuropäische Zeit, Mittleuropäische Sommerzeit). Das Signal codiert innerhalb von einer Minute die Zeit- und Datumsinformationen für die darauffolgende Minute.

Der Beginn einer Sekunde (eines Bits) wird durch die Absenkung der Trägeramplitude markiert. Die Ausnahme bildet hier die letzte Sekunde, bei welcher keine Absenkung stattfindet. Je nach Dauer der Amplitudenabsenkung kennzeichnet dies entweder den binären Wert 0 (Absenkung für 100 ms) oder für den Wert 1 (Absenkung für 200 ms).

Das Signal wird anschliessend über einen 50-kW-Halbleitersender ausgesendet. Als Reserve dient der 50-kW-Röhrensender, der 1998 durch den 50-kW-Halbleitersender abgelöst wurde.



(a) ohne Amplitudenmodulation



(b) mit Amplitudenmodulation

Abbildung 3.4: Beispielhaftes DCF77 Signal

¹ Ein Fehler von 10^{-14} entspricht ca. einer Milliardstel Sekunde pro Tag (10^{-14}) oder einer Sekunde in 2.7 Mio. Jahren.

Codierung

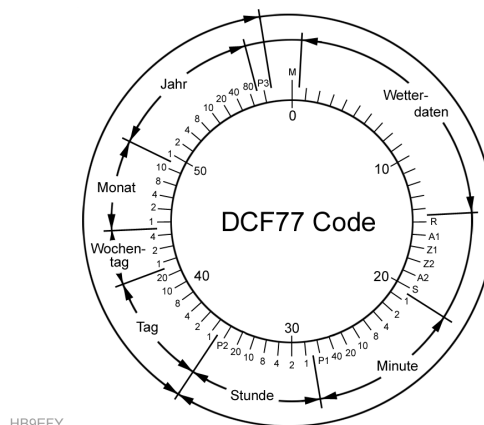
Die Codierung der Zeitinformation erfolgt anhand der untenstehenden Tabelle.

- Wochentag: Beginnend bei Montag (1).
- Monat: Beginnend bei Januar (1).
- Jahr: Es werden nur die Einer- und Zehnerstellen der Jahreszahl übermittelt (Bsp: 2014, Es wird nur 14 codiert).

| Bit | Bedeutung |
|--|---|
| 0 | Start einer Minute |
| 1-14 | Wetterinformationen und Katastrophenschutz |
| <i>Informationen zu Unregelmässigkeiten im Sendebetrieb, Zeitzone, Beginn / Ende Sommerzeit, Schaltsekunden</i> | |
| 15 | Rufbit |
| 16 | Wenn Wert 1: Umstellung MEZ/MESZ am Ende der Stunde |
| 17 | Wenn Wert 0: MEZ, Wenn Wert 1: MESZ |
| 18 | Wenn Wert 0: MESZ, Wenn Wert 1: MEZ |
| 19 | Wenn Wert 1: Schaltsekunde am Ende der Stunde. |
| <i>Zeitinformation der nachfolgenden Minute in BCD-Zahlen (Start mit LSB). Gerade Parität für Fehlererkennung.</i> | |
| 20 | Beginn der Zeitinformation (immer 1) |
| 21 | Minute (Einer) - Bit für 1 |
| 22 | Minute (Einer) - Bit für 2 |
| 23 | Minute (Einer) - Bit für 4 |
| 24 | Minute (Einer) - Bit für 8 |
| 25 | Minute (Zehner) - Bit für 10 |
| 26 | Minute (Zehner) - Bit für 20 |
| 27 | Minute (Zehner) - Bit für 40 |
| 28 | Parität Minute |
| 29 | Stunde (Einer) - Bit für 1 |
| 30 | Stunde (Einer) - Bit für 2 |
| 31 | Stunde (Einer) - Bit für 4 |
| 32 | Stunde (Einer) - Bit für 8 |
| 33 | Stunde (Zehner) - Bit für 10 |
| 34 | Stunde (Zehner) - Bit für 20 |
| 35 | Parität Stunde |
| 36 | Kalendertag (Einer) - Bit für 1 |
| 37 | Kalendertag (Einer) - Bit für 2 |
| 38 | Kalendertag (Einer) - Bit für 4 |
| 39 | Kalendertag (Einer) - Bit für 8 |
| 40 | Kalendertag (Zehner) - Bit für 10 |
| 41 | Kalendertag (Zehner) - Bit für 20 |
| 42 | Wochentag - Bit für 1 |
| 43 | Wochentag - Bit für 2 |
| 44 | Wochentag - Bit für 4 |
| 45 | Monatsnummer (Einer) - Bit für 1 |
| 46 | Monatsnummer (Einer) - Bit für 2 |
| 47 | Monatsnummer (Einer) - Bit für 4 |
| 48 | Monatsnummer (Einer) - Bit für 8 |
| 49 | Monatsnummer (Zehner) - Bit für 10 |
| 50 | Jahr (Einer) - Bit für 1 |

| Bit | Bedeutung |
|-----|----------------------------|
| 51 | Jahr (Einer) - Bit für 2 |
| 52 | Jahr (Einer) - Bit für 4 |
| 53 | Jahr (Einer) - Bit für 8 |
| 54 | Jahr (Zehner) - Bit für 10 |
| 55 | Jahr (Zehner) - Bit für 20 |
| 56 | Jahr (Zehner) - Bit für 40 |
| 57 | Jahr (Zehner) - Bit für 80 |
| 58 | Parität Datum |
| 59 | Keine Sekundenmarke |

Idealerweise sollte ein Gerät das Signal mindestens zwei Minuten ungestört empfangen können, um die Zeit eindeutig einzustellen. Auf jeden Fall werden die ersten 36 Sekunden und anschliessend ein Minutenübergang (Sekunden 58 und 59) benötigt.



HB9EFY

Fig. 3.5: DCF77-Codierung als Codierscheibe (Quelle: Endorphino; <http://endorphino.de/projects/electronics/timemanipulator/index.html>)

Die Zeitangabe „12.03.2014 18:15:30“ (MEZ) wird wie folgt als DCF77-Signal codiert:

| | | | | | | | | | | | | | | | | | | | | |
|--------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|
| Bitnr. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| Wert | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| | | | | | | | | | | | | | | | | | | | | |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Bitnr. | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| Wert | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

| | | | | | | | | | | | | | | | | | | | | |
|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Bitnr. | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 |
| Wert | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | - |

BCD-Codierung

Nachfolgend wird kurz die verwendete BCD-Codierung (Binary Coded Decimal) beschrieben:

Jede Ziffer einer Zahl wird einzeln mit 4 Bit (8, 4, 2, 1) codiert. Von den $2^4 = 16$ Kombinationsmöglichkeiten werden nur 10 verwendet; darum spricht man oft von einer Speicherverschwendung.

Zusatznutzung

Das Signal des Zeitzeichensenders wird seit Ende 2006 zusätzlich für Alarmierungen und Wetterdaten eingesetzt. Dafür werden die ersten 14 Bit der Minute verwendet. Im Katastrophenfall werden Details zur Katastrophe und der Region übermittelt. Diese Informationen werden aufgrund ihrer Auswirkungen doppelt gesichert (Paritätsbit und Wiederholung). Die Wetterdaten werden von der Firma Meteo Time GmbH bereitgestellt und ermöglichen eine viertägige Vorhersage für 60 Regionen in Europe. Die Decodierung dieser Wetterdaten ist jedoch lizenzpflichtig, da diese im proprietären Meteo-Time-Protokoll übermittelt werden.

3.2.6 Fehlererkennung beim Empfänger

In die Codierung ist eine einfache Fehlererkennung, anhand einer einfachen Parität, eingebaut. Die Fehlererkennung ist jedoch bei Blockfehlern und Fehlermaskierungen nicht immer gewährleistet. In der Praxis wird häufig auf Redundanz, bzw. Stetigkeit und Kontinuität der gelieferten Zeitinformationen gesetzt.

Dazu ein Beispiel:

In den aktuellen Zeitinformationen wird das Jahr 2014 und der Monat März geliefert. In den vorangehenden Minuten oder Stunden wurde aber immer das Jahr 2014 und der Monat Januar geliefert. Daraus lässt sich schliessen, dass es sich bei der aktuellen Zeitinformation um einen Fehler handelt.

Eine Fehlerkorrektur ist mit einer einfachen Parität nicht möglich. Werden jedoch die Daten der letzten Minuten, Stunden und Tagen hinzugezogen kann basierend auf dem Inhalt eine Fehlerkorrektur durchgeführt werden.

3.2.7 Fehlererkennung beim Sender

Das in Mainflingen ausgesendete Signal wird an Braunschweig ausgewertet, analysiert und bei Bedarf wird ein Alarm ausgelöst. Der Alarm wird ausgelöst, wenn während mehrerer Minuten die gleichen logischen Fehler im Signal erkannt wurden oder wenn auf technischer Ebene eine Unregelmässigkeit festgestellt wurde (z.B. zu kleines Signal über eine längere Zeitdauer).

Network Time Protocol

Wird bei eine NTP-Root-Server (Siehe Kapitel 3.2.9) das DCF77-Signal als Referenz verwendet, wird dieser mit der Kennung „DCFa“ gekennzeichnet.

3.2.8 GPS

Nebst der Zeitsynchronisation über Funk, kann eine genauere Information von den GPS-Satelliten gewonnen werden. Dies obwohl GPS-Systeme hauptsächlich zur Lokalisierung des

Standortes entwickelt wurden. Momentan befinden sich sechs solche Satelliten in ungefähr 200'000 km Höhe. Jeder umrundet die Erde zweimal pro Tag. Auf jedem Satellit sind jeweils zwei Atomuhren vorhanden. Ein Satellit sendet andauernd seine Bahnposition und die genaue Uhrzeit. Durch Signale mehrerer Satelliten, kann der Empfänger seinen genauen Standort berechnen. Anschliessend kann die Laufzeit des Signals zurück gerechnet und die durchs Versenden verstrichene Zeit, der Delay, approximiert werden. Durch dieses Verfahren kann die Uhrzeit mit einer Genauigkeit unter 1 μsec berechnet werden.

Vorteile der Zeit-Synchronisation über GPS sind die weltweite Abdeckung und eine höhere Genauigkeit als die Synchronisation über Zeitzeichensender.

Ein grosser Nachteile der GPS Synchronisation ergibt sich jedoch durch das verwendete Kurzwellen Signal (zwischen 1176,45 bis 1575,42 MHz). Dieses Signal kann praktisch nur unter freien Himmel empfangen werden und eignet sich daher nicht für einen typischen Computer.^{1, 2, 3}

3.2.9 Internet

Will man eine Uhr via Internet synchronisieren, stösst man auf folgendes Problem:

Die genaue Sendedauer eines Datenpaketes, welches die korrekte Uhrzeit beinhaltet, ist weder vorhersehbar, noch kann man die verstrichene Zeit beim Empfänger zurückrechnen.

Da keine Aussage über die Genauigkeit der empfangenen Information möglich ist, verliert sie drastisch an Wert.

3.3 Zeitprotokoll

Um eine brauchbare Synchronisation via Internet zu ermöglichen wurde folgender, vereinfachter Datenaustausch definiert: (Der Client synchronisiert seine Zeit mit der genauen Zeit eines Server. Die Serverzeit entspricht also nicht der Clientzeit.)⁴

1. Der Client schickt eine **Anfrage an den Server** und **merkt** sich seine aktuelle, lokale Zeit (welche mit grosser Wahrscheinlichkeit nicht der korrekten Serverzeit entspricht).
2. Der Server antwortet mit seiner lokalen, korrekten **Zeit des Empfangs und der des (geplanten) Rückversandes**.
3. Der Client merkt sich die Empfangszeit der Antwort.

Nach dem Datenaustausch hat der Client folgende vier verschiedene Zeiten zur Verfügung:

¹ hopf.com, *Genauigkeitsvergleich DCF77 und GPS*.

² mono.rgbtechnology.pl. *GPS Zeitsynchronisation*. URL: <http://www.mono.rgbtechnology.pl/de/faq/gps-zeitsynchronisation.html> (besucht am 24. 05. 2014).

³ emsec.rub.de. *Sichere Zeitsynchronisation in drahtlosen Sensornetzen*. URL: http://www.emsec.rub.de/media/crypto/attachments/files/2010/04/ms_michael_ziaja.pdf (besucht am 24. 05. 2014).

⁴ stackoverflow.com. *How Does the Network Time Protocol Work?* URL: <http://stackoverflow.com/questions/1228089/how-does-the-network-time-protocol-work> (besucht am 24. 05. 2014).

1. Client: lokale Zeit der Anfrage - (A)
2. Server: Empfangszeit - (X)
3. Server: Versandzeit - (Y)
4. Client: Empfangszeit - (B)

Als Annahme wird vereinfacht angenommen, dass der Hinweg des Paketes ungefähr gleich lang wie der Rückweg dauert. Nun kann der Client mit folgender Rechnung die Uhrzeit

gut annähern (inkl. der dazugehörigen Fehlerabschätzung):

- Totale Zeit des Prozesses: $B-A$
- Server Bearbeitungszeit: $Y-X$
- Effektive Datentransferzeit: $B-A-(Y-X)$
- Datentransferzeit Rückweg: $[B-A-(Y-X)]/2$
- Korrekte Zeit; plus Dauer des Rückweges: $Y+[(B-A-(Y-X))/2]$
- als Fehlerabschätzung wird die Hälfte der *Effektive Datentransferzeit* verwendet: $B-A-(Y-X)$

Beispiel Synchronisation

Ein konkretes Beispiel dazu (zur Vereinfachung in Millisekunden).

(Die Clientzeit ist jeweils falsch. Die Serverzeit ist korrekt. Die zwei Zeiten sind nicht synchron und dürfen daher nicht direkt miteinander verrechnet werden.)

- Clientzeit lokal beim Senden: $A = 200$
- Serverzeit des Empfangs: $X = 150$
- Serverzeit Rückversand: $Y = 180$
- Clientzeit Empfang: $B = 300$

Daraus kann auf folgendes geschlossen werden:

- Totale Zeit des Prozesses: $B-A = 300-200 = \mathbf{100}$
- Server Bearbeitungszeit: $Y-X = 180-150 = \mathbf{30}$
- Effektive Datentransferzeit: $[B-A]-(Y-X) = 100 - 30 = \mathbf{70}$
- Nur Rückweg: $(B-A-(Y-X))/2 = 70/2 = \mathbf{35}$
- Korrekte Zeit, plus Dauer des Rückweges: $Y+[(B-A-(Y-X))/2] = 180+35 = \mathbf{215}$
- Fehlerabschätzung: max. 35

Das Worstcase Szenario ist folgendes:

- Client \Rightarrow Server: ≈ 0
- Server \Rightarrow Client: ≈ 70

\Rightarrow **Fehler: 35,**

3.4 NTP / SNTP

NTP (Network Time Protocol) basiert auf dem oben beschriebenen Prinzip, hat aber noch diverse Verbesserungen in sich. NTP befindet sich momentan in der Version 4. Da das Protokoll sehr komplex ist, gibt es auch eine vereinfachte Version, die SNTP (Simple Network Time Protocol) genannt wird. Die genaue Funktionsweise dieses Protokolles wird in dieser Arbeit nicht erläutert.^{1 2}

3.5 Weitere Zeitsignaldienste

Es gibt einige weitere Zeitsignaldienste, welche jedoch in der heutigen Zeit keine grosse Bedeutung mehr haben. Das Zeitsignal wird zum Teil auch als Zusatzinformation auf dem Hörfunk oder im Video- / Teletext von Fernsehsendern übertragen. Früher wurde auch über das öffentliche Telefon ein Zeitdienst angeboten.

3.6 Fazit

Mit Hilfe der Analyse konnten folgenden Schwerpunkte und Herausforderungen für die Umsetzung des Projektes, identifiziert werden.

3.6.1 Codierung der Zeitinformationen einer Minute

Pro Minute wird jeweils die Zeitinformation der nächsten Minute empfangen. Einfachheits halber wird vorausgesetzt, dass wir mindestens eine vollständige Minute decodieren können. Dabei müssen die Sekunden-Kennungen gezählt werden, so dass man daraus Rückschlüsse auf die Zeit machen kann. Da der Signal-Takt der Sekunde entspricht (1 Hz), sollte daraus die exakte Sekundendauer angenähert werden. Es muss also eine eigenständige Uhr implementiert werden, die mit diesen Gegebenheiten umgehen kann. Die Uhr muss komplett ohne Signal auskommen und dabei akkurat weiterlaufen. Weiter muss bei erfolgreichem Empfang der Zeit entsprechend reagiert und die Uhr neu justiert werden.

Die Implementation dieser Uhr wird nicht einfach werden, da das Projekt auf einem Multi-Threading-fähigen Rechner umgesetzt wird. Das heisst es gibt keine Echtzeit-Ausführungen. Das Zusammenspiel von Soft- und Hardware gilt als gegeben. Wenn z.B. eine Sekunde mit einer gewissen Anzahl Durchläufe in einer Schleife angenähert wird, so ist nicht garantiert, dass diese Schleife immer die gleiche Ausführungszeit benötigt. Grund dafür ist, dass die verfügbaren Systemressourcen mit anderen Prozessen geteilt werden müssen. Als vorbeugende Massnahme sollte die Implementation so systemnah wie möglich umgesetzt werden, so dass unnötige Verarbeitungszeiten und daraus resultierende Ungenauigkeit vermieden werden können.

3.6.2 Fehlererkennung und Fehlerkorrektur

Das Signal, bzw. die Funk-Wellen des DCF77 können aufgrund ihrer Eigenschaften relativ leicht gestört werden. Deswegen muss unsere Implementation in der Lage sein, fehlerhafte Informationen zu erkennen und zu korrigieren. Dafür muss das „Langzeitverhalten“ gespeichert, analysiert und darauf entsprechend reagiert werden. Dies kann im Besonderen bei der

¹ emsec.rub.de, *Sichere Zeitsynchronisation in drahtlosen Sensornetzen*.

² Spezifikation zum NTP <http://www.eecis.udel.edu/~mills/database/reports/ntp4/ntp4.pdf>

Umstellung zwischen Winter- und Sommerzeit zu Problemen führen. Eine Zeitverschiebung muss korrekt und rasch übernommen werden. Allgemein gilt: Wenn eine Information über längere Zeit gleichbleibend übermittelt wird, muss diese in absehbarer Zeit übernommen werden.

KAPITEL 4

Dokumentation

In diesem Kapitel wird der Aufbau der Implementation mit den verschiedenen Komponenten näher beschrieben.

4.1 Übersicht der Struktur

Die Sourcen, Dokumentation, etc. befinden sich in unserem Github-Repository (Siehe Kapitel [2.3](#))

4.1.1 Server Seite

Das Backend befindet sich im Repository unter `/code/server` und ist in folgende Module aufgeteilt:

- **Main** - `dcf77_main.c`
Das Main-Modul verbindet alle übrigen Module miteinander und stellt die Kommunikation untereinander sicher.
- **DCF77 Reader** - `dcf77_reader.c`
Der Reader liest die Daten von der USB-Schnittstelle zum DCF77 und übergibt diese dem Decoder.
- **DCF77 Decoder** - `dcf77_decoder.c`
Der Decoder decodiert laufend die erhaltenen Daten von der USB-Schnittstelle und generiert pro Minute einen entsprechenden Timestamp. Voraussetzung dafür ist, dass genügend Daten empfangen wurden und keine Fehler aufgetreten sind. Der Timestamp wird anschliessend dem Clock-Modul zur Synchronisation übergeben.
- **Clock** - `Clock.c`
Das Clock-Modul bildet eine eigenständige, laufende Uhr. Wird ein DCF77-Signal erfolgreich decodiert, so werden die Informationen (Datum und Zeit) sekunden-genau übernommen. Anschliessend läuft die Uhr autonom weiter. Als Basis wird die Systemzeit des Server-Hosts verwendet.
- **Server** - `SimpleSocketServer.c`
Der Server bietet eine Web-Schnittstelle via Sockets an, welche den Timestamp,

gemäss ISO 8601¹, als JSON anbietet. Der Timestamp wird aus der aktuellen Zeit aus dem Clock-Modul erzeugt.

4.1.2 Client Seite

Das Frontend befindet sich im Repository unter `/code/client`. Das Frontend ist eine Web-Oberfläche, welche die entsprechenden GUI-Elemente darstellt und via HTTP-Request in bestimmten Zeitintervallen die aktuelle Zeit vom Server abrufen.

Für die Darstellung haben wir folgende Plugins verwendet:

- **jQuery & jQuery UI**
Das wohl bekannteste JS Framework, das einem in jeglichen Bereichen unterstützt und Umsetzungen vereinfacht. jQuery UI ist ein GUI für das Web, welches auf jQuery basiert. (<http://jquery.com/>, <http://jqueryui.com/>)
- **Bootstrap**
Von Twitter eine Struktur-Vorlage um eine moderne Website umzusetzen. Sie bietet gängige Funktionen (Responsive, Informations-Elemente, Navigation, etc.) und grundlegende Styles an. (<http://getbootstrap.com/>)
- **CoolClock**
Die CoolClock ist ein Projekt in dem eine analoge Uhr vollständig mit HTML 5, CSS und JS umgesetzt worden ist. (<http://randomibis.com/coolclock/>)

4.2 Modul-Dokumentation

Die Funktionsweise, der einzelnen Module, wurde innerhalb des Codes dokumentiert und beschrieben. Nachfolgend werden die wichtigsten Aspekte der Umsetzung kurz erläutert.

4.2.1 DCF Reader

Source: `/code/server/dcf77_reader.c`

Via Schnittstelle zum „Expert mouse CLOCK USB II“ wird der Status der Warteschlange solange abgerufen, bis mindestens ein Wert (8-Bit Integer) bereit ist. Sobald ein Wert vorhanden ist, wird dieser ausgelesen und direkt dem Decoder übergeben.

Da das Programm seriell abläuft, ist es wichtig, dass wir pro Verarbeitung nicht länger als eine Sekunde benötigen, denn dann muss das nächste Signal verarbeitet werden können. Dadurch dass die ganze Software in C implementiert wurde, und keine sehr aufwändigen Berechnungen durchgeführt werden müssen, stellt dies kein Problem dar.

4.2.2 DCF Decoder

Source: `/code/server/dcf77_decoder.c`

Der Decoder mappt jedes erhaltene Byte direkt einem Bit zu. Laufend werden aus den verfügbaren Bits, via 4 Paritätsbit, gültige Daten gesucht. Ein vollständiges Datum besteht aus 39 fortlaufenden Bits. Zur Validierung wird das Datum auf grundlegende Datum-Korrektheit überprüft. Als zusätzliche Validierung können Datum-Informationen

¹ Spezifikation: <http://www.iso.org/iso/home/standards/iso8601.htm>

mit der Systemzeit abgeglichen werden (diese Funktion wurde allerdings auskommentiert, da dies nicht einsatzwürdig sei). Wird ein gültiges Datum gefunden, so wird dies direkt der Systemclock übergeben.

4.2.3 Clock

Source: `/code/server/Clock.c`

Die Clock wird zu Beginn mit der aktuellen Systemzeit initialisiert. Anschliessend wird mit Hilfe der POSIX-Alarm Funktion jede Sekunde eine Funktion ausgeführt, welche die Zeit erhöht. Dabei werden die Standardmässigen Regeln für die Zeitzählung angewendet.

- 1 Minute: 60 Sekunden
- 1 Stunde: 60 Minuten
- 1 Tag: 24 Stunden
- - Januar, März, Mai, Juli, August, Oktober, Dezember: 1 Monat: 31 Tage
 - April, Juni, September, November: 1 Monat: 30 Tage
 - Februar:
 - * Aktuelles Jahr ist ein Schaltjahr: 29 Tage
 - * Aktuelles Jahr ist kein Schaltjahr: 28 Tage
- 1 Jahr: 365 Tage (366 Tage in einem Schaltjahr)

Das Schaltjahr wird wie folgt berechnet. Grundsätzlich gilt: Ist die aktuelle Jahreszahl ohne Rest durch 4 teilbar ist dies ein Schaltjahr. Ist dasselbe Jahr jedoch auch restlos durch 100 teilbar handelt es sich trotzdem um kein Schaltjahr, ausgenommen das Jahr lässt sich restlos durch 400 teilen, dann handelt es sich wiederum um ein Schaltjahr.

Die Uhr bietet eine Funktion an, mit welchem die Zeit „synchronisiert“ und auf einen bestimmten Wert gesetzt werden kann. Bei der Synchronisierung wird eine Plausibilisierung der zu synchronisierenden Zeit vorgenommen. Es werden dabei nachfolgende Regeln für die Überprüfung angewendet. Wir haben uns bewusst dafür entschieden die Validierung nur bis auf Stunden-Ebene zu implementieren, da je nach Empfangsqualität nur einige wenige Signale vom DCF77 empfangen werden können.

- Jahr:
 - Jahr entspricht aktuellem Jahr
 - Jahr entspricht dem nächsten Jahr
 - * Monat entspricht dem Januar
 - * Tag entspricht dem 01.
 - Jahr entspricht dem letzten Jahr
 - * Monat entspricht dem Dezember
 - * Tag entspricht dem 31.
- Monat:

- Monat entspricht aktuellem Monat
- Monat entspricht dem nächsten Monat
 - * Tag entspricht dem 01.
- Monat entspricht dem letzten Monat
 - * Tag entspricht dem Letzten des letzten Monats.
- Tag:
 - Tag entspricht aktuellem Tag
 - Tag entspricht dem nächsten Tag
 - * Stunde entspricht 0.
 - Tag entspricht dem letzten Tag
 - * Stunde entspricht 23.
- Stunde:
 - Stunde entspricht aktueller Stunde
 - Stunde ist grösser gleich der vorletzten Stunde.
 - Stunde ist kleiner gleich der übernächsten Stunde.

Das aktuelle Jahr (Monat, Tag, Stunde) bezeichnet die aktuelle Zeit der Clock (vor der Synchronisation).

Um konkurrierende Zugriffe zu verhindern, werden Semaphoren eingesetzt.

Uns ist bewusst, dass die aktuelle Implementation der Uhr je nach Systemlast unterschiedlich stark abweichen kann. Hier mussten wir jedoch aus zeitlichen Gründen einen Kompromiss eingehen. Ursprünglich war gedacht, dass für die Bestimmung einer Sekunde der System-Tic verwendet wird. Dieser generiert 18.2 mal in der Sekunde einen Tic-Interrupt. Die Registrierung einer entsprechenden Interrupt Service Routine oder das explizite Abfragen des Tics erfordert unter allen modernen Betriebssystemen höhere Rechte (Kernel Mode). Dafür müsste entweder ein eigener Geräte-Treiber oder ein Kernel-Modul implementiert werden. Da wir auch nach einigen Anläufen keine funktionierende Lösung realisieren konnten, haben wir uns für die Implementation der vereinfachten Variante entschieden.

4.2.4 Simple Web Server

Source: */code/server/SimpleSocketServer.c*

Aus dem Main-Modul heraus wird ein simpler Socket Server in einem separaten Thread gestartet. Der Server arbeitet nur auf einem Thread. Multithreading muss in C selbst implementiert werden. Dies haben wir aufgrund der Rahmenbedingungen bewusst nicht implementiert. Wird eine Verbindung zum Server hergestellt wird mit Hilfe einer Hilfs-Funktion und der Zeit aus dem Clock-Modul ein JSON-String generiert und an den Client zurückgesendet. Die Implementation des Servers wurde bewusst so minimal als möglich gehalten.

4.3 System-Voraussetzungen

Für die Ausführung gelten folgende Systemvoraussetzungen:

- OS: Windows oder Mac OSX (32- und 64-bit)
- C-Programme Kompilierung und Ausführung (für Windows empfiehlt sich eine Cygwin-Installation)
- Aktueller Browser (Chrome, Mozilla Firefox, Internet Explorer), JavaScript muss aktiviert sein

4.4 Installationsanleitung - DCF77-Treiber

4.4.1 OSX

Für das Auslesen der Daten wird der D2xx Treiber genutzt. Eine genau Anleitung zur Installation ist in folgendem Dokument in Kapitel „3.2 Installing D2xx Drivers“ (Seite 9) beschrieben.

Im Git Repo unter:

/analyse/AN_134_FTDI_Drivers_Installation_Guide_for_MAC_OSX.pdf

4.4.2 Windows

Um die Applikation unter Windows zu kompilieren, muss der Pfad zu den Treiber-Libraries dem Compiler bekannt gegeben werden. Ansonsten sind keine weiteren Schritte notwendig.

4.5 Betrieb

4.5.1 Server

Im Verzeichnis *code/server* können via *make* (Windows: „*make fullWin*“, OSX: „*make fullOst*“) die ausführbaren Dateien generiert werden.

Anschließend kann die ausführbare Datei „./controlledClock.exe“ gestartet werden. Um das DCF77 Signal zu empfangen, muss der Empfänger korrekt installiert und angeschlossen sein. Das Programm ist aber auch ohne Empfänger ausführbar, und kann für die Synchronisation verwendet werden.

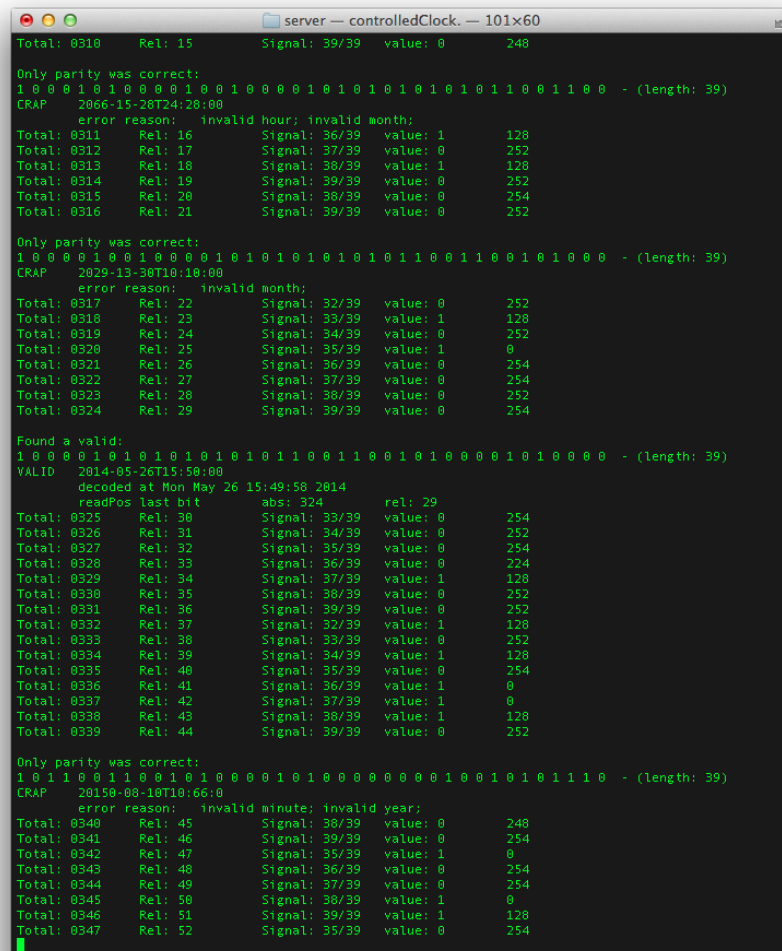
4.5.2 Client

Der Client ist komplett in HTML, JS und CSS geschrieben und kann deshalb ohne Installation ausgeführt werden.

Um den Client zu starten, muss die Datei „index.html“ im Client-Verzeichnis geöffnet werden. Um eine Synchronisation auszuführen, muss der Server laufen.

4.6 Screenshots

4.6.1 Server



```

server — controlledClock. — 101x60
Total: 0310 Rel: 15 Signal: 39/39 value: 0 248

Only parity was correct:
1 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 - (length: 39)
CRAP 2066-15-20T24:20:00
error reason: invalid hour; invalid month;

Total: 0311 Rel: 16 Signal: 36/39 value: 1 128
Total: 0312 Rel: 17 Signal: 37/39 value: 0 252
Total: 0313 Rel: 18 Signal: 38/39 value: 1 128
Total: 0314 Rel: 19 Signal: 39/39 value: 0 252
Total: 0315 Rel: 20 Signal: 38/39 value: 0 254
Total: 0316 Rel: 21 Signal: 39/39 value: 0 252

Only parity was correct:
1 0 0 0 1 0 0 1 0 0 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 0 0 - (length: 39)
CRAP 2029-13-30T10:10:00
error reason: invalid month;

Total: 0317 Rel: 22 Signal: 32/39 value: 0 252
Total: 0318 Rel: 23 Signal: 33/39 value: 1 128
Total: 0319 Rel: 24 Signal: 34/39 value: 0 252
Total: 0320 Rel: 25 Signal: 35/39 value: 1 0
Total: 0321 Rel: 26 Signal: 36/39 value: 0 254
Total: 0322 Rel: 27 Signal: 37/39 value: 0 254
Total: 0323 Rel: 28 Signal: 38/39 value: 0 252
Total: 0324 Rel: 29 Signal: 39/39 value: 0 254

Found a valid:
1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 - (length: 39)
VALID 2014-05-26T15:50:00
decoded at Mon May 26 15:49:58 2014
readPos last bit abs: 324 rel: 29

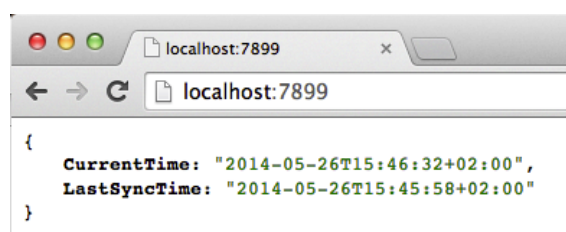
Total: 0325 Rel: 30 Signal: 33/39 value: 0 254
Total: 0326 Rel: 31 Signal: 34/39 value: 0 252
Total: 0327 Rel: 32 Signal: 35/39 value: 0 254
Total: 0328 Rel: 33 Signal: 36/39 value: 0 224
Total: 0329 Rel: 34 Signal: 37/39 value: 1 128
Total: 0330 Rel: 35 Signal: 38/39 value: 0 252
Total: 0331 Rel: 36 Signal: 39/39 value: 0 252
Total: 0332 Rel: 37 Signal: 32/39 value: 1 128
Total: 0333 Rel: 38 Signal: 33/39 value: 0 252
Total: 0334 Rel: 39 Signal: 34/39 value: 1 128
Total: 0335 Rel: 40 Signal: 35/39 value: 0 254
Total: 0336 Rel: 41 Signal: 36/39 value: 1 0
Total: 0337 Rel: 42 Signal: 37/39 value: 1 0
Total: 0338 Rel: 43 Signal: 38/39 value: 1 128
Total: 0339 Rel: 44 Signal: 39/39 value: 0 252

Only parity was correct:
1 0 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 0 - (length: 39)
CRAP 20150-08-10T10:66:0
error reason: invalid minute; invalid year;

Total: 0340 Rel: 45 Signal: 38/39 value: 0 248
Total: 0341 Rel: 46 Signal: 39/39 value: 0 254
Total: 0342 Rel: 47 Signal: 35/39 value: 1 0
Total: 0343 Rel: 48 Signal: 36/39 value: 0 254
Total: 0344 Rel: 49 Signal: 37/39 value: 0 254
Total: 0345 Rel: 50 Signal: 38/39 value: 1 0
Total: 0346 Rel: 51 Signal: 39/39 value: 1 128
Total: 0347 Rel: 52 Signal: 35/39 value: 0 254

```

Abbildung 4.1: Konsolen Output bei laufendem Server (mit DCF77 via USB)



```

localhost:7899
localhost:7899

{
  "CurrentTime": "2014-05-26T15:46:32+02:00",
  "LastSyncTime": "2014-05-26T15:45:58+02:00"
}

```

Abbildung 4.2: HTTP Request auf Webserver

4.6.2 Client

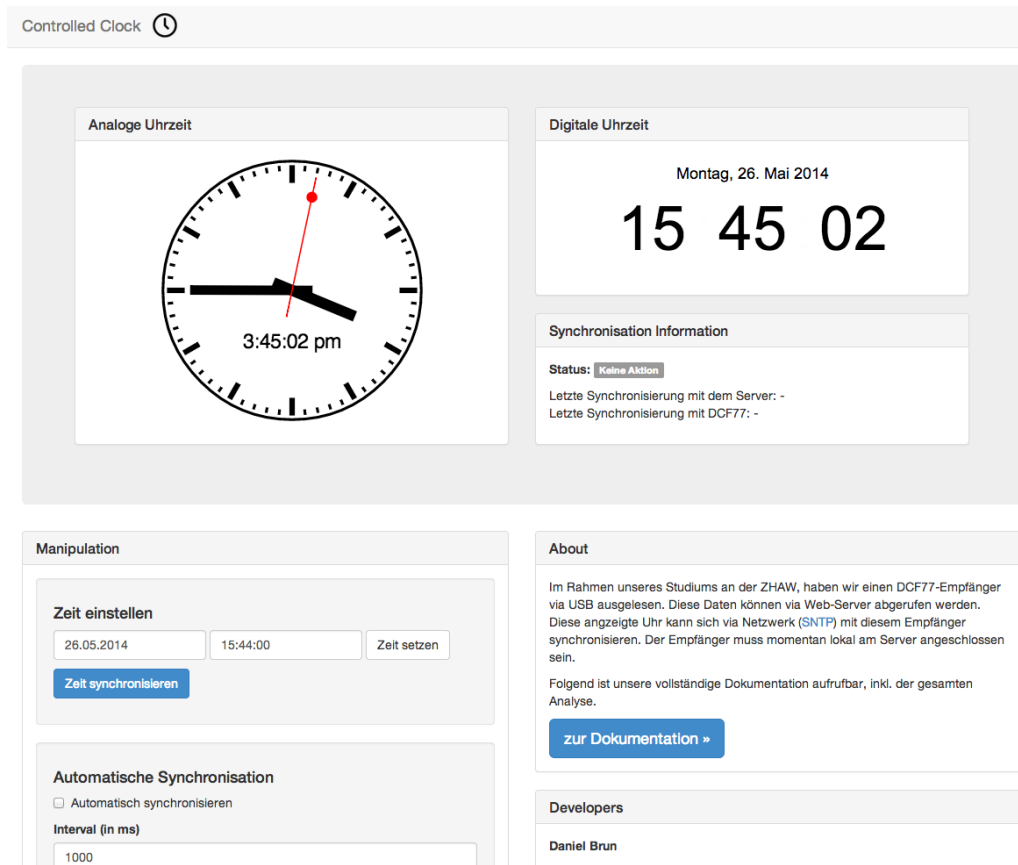


Abbildung 4.3: Client nach Aufruf (vollständig geladen und initialisiert)

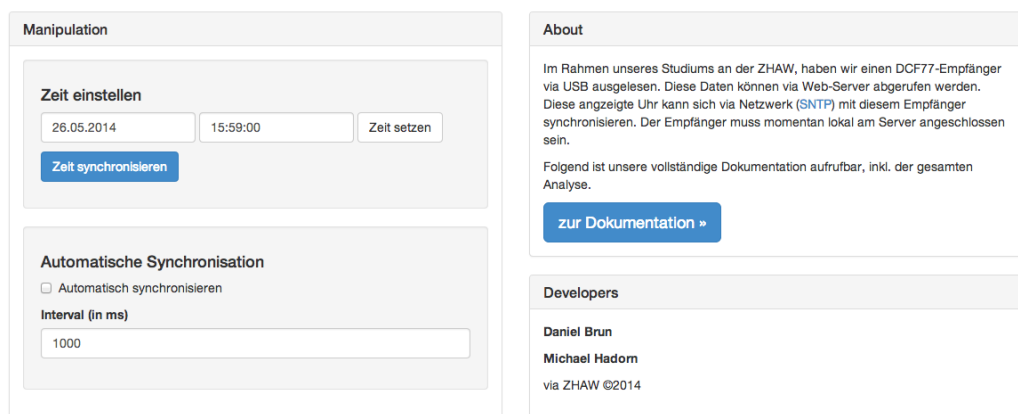


Abbildung 4.4: Einstellmöglichkeiten via Web-GUI

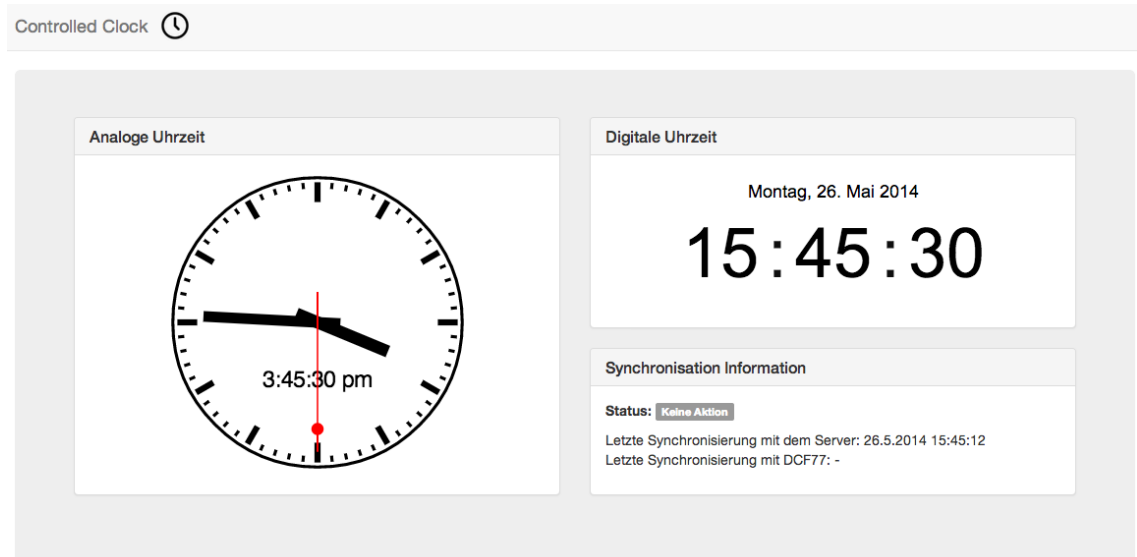


Abbildung 4.5: Synchronisation ohne DCF77 Signal

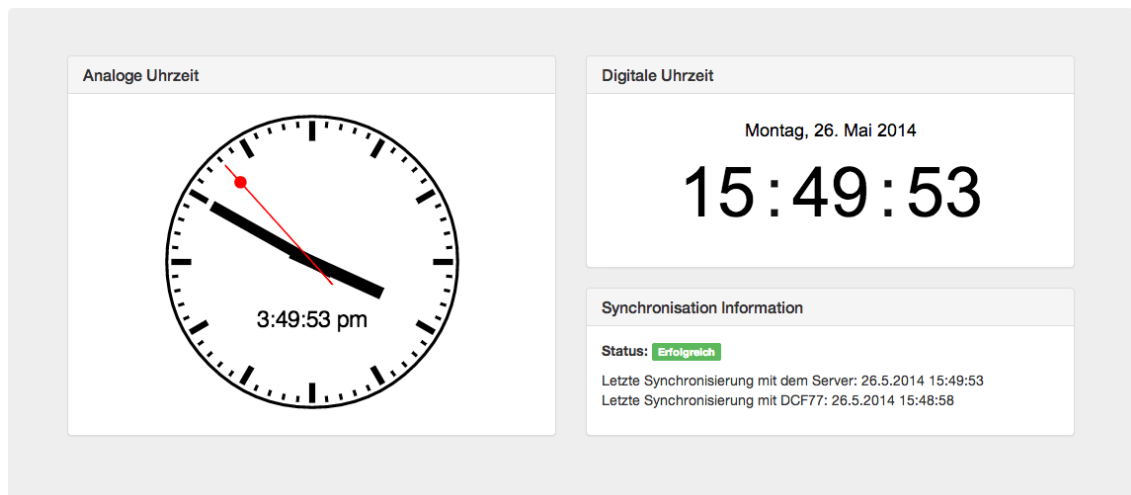


Abbildung 4.6: Synchronisation mit gültigem DCF77 Signal

KAPITEL 5

Schlusswort

5.1 Aktueller Projektstand

Das Projekt kann als funktionierend und abgeschlossen betrachtet werden. Selbstverständlich gäbe es noch weitere nützliche und interessante Erweiterungen, welche das Projektvolumen aber deutlich sprengen würden.

5.1.1 Fazit

Wir sind sehr zufrieden mit unseren Leistungen und dem daraus folgenden Produkt. Es war ein äusserst spannendes, interessante und herausforderndes Projekt. Durch die Möglichkeit die Aufgabenstellung mehrheitlich selbst zu bestimmen, konnten wir eine Umsetzung abschliessen, die ganz in unserem Interesse lag. Dazu konnten wir Technologien verwenden, in denen wir unsere Fähigkeiten verbessern wollten.

Der Aufwand hat uns Spass gemacht und sich für uns schon nur wegen dem neuen Lerninhalt gelohnt.

5.1.2 Reflexion

5.1.3 Danksagung

Wir möchten uns herzlich bei folgenden Personen bedanken:

| | |
|-----------------|---|
| George Brügger | In der Funktion des Auftragsgebers erhielten wir wertvolle Inputs, Tipps, Wünsche und Anregungen wie wir was umsetzen können. Herr Brügger begleitete uns auf der technischen Seite vom Anfang (Grundidee definieren, GUI Ideen) bis zum Schluss. |
| Syrus Mozafar | In der Funktion des Projektkontrolleurs durften wir in regelmässigen Abständen seine Meinung zu unserer Durchführung des Projekt in Anspruch nehmen. |
| Sina Masquiren | Grundidee und Inspiration der Funkuhr-Umsetzung. |
| Jonathan Hadorn | Korrekturlesung |

Quellenverzeichnis

- [1] bmvit.gv.at. *Information der obersten Fernmeldebehörde*. URL: <http://www.bmvit.gv.at/telekommunikation/publikationen/infoblaetter/downloads/042013.pdf> (besucht am 24.05.2014).
- [2] circuitdesign.de. *Leitfaden über Funk*. URL: http://www.circuitdesign.de/products/tech_info/guide2.asp (besucht am 24.05.2014).
- [3] dcf77.de. *Informationen zum DCF77*. URL: <http://www.dcf77.de/> (besucht am 24.05.2014).
- [4] ejpd.admin.ch. *Zeitzeichensender HBG in der Schweiz*. URL: <http://www.ejpd.admin.ch/ejpd/de/home/dokumentation/mi/2009/2009-08-261.html> (besucht am 24.05.2014).
- [5] emsec.rub.de. *Sichere Zeitsynchronisation in drahtlosen Sensornetzen*. URL: http://www.emsec.rub.de/media/crypto/attachments/files/2010/04/ms_michael_ziaja.pdf (besucht am 24.05.2014) (siehe S. 25, 27).
- [6] heret.de. *Funkuhr*. URL: <http://www.heret.de/funkuhr/index.htm> (besucht am 24.05.2014) (siehe S. 13).
- [7] hopf.com. *Genauigkeitsvergleich DCF77 und GPS*. URL: http://www.hopf.com/de/dcf77-gps_de.html (besucht am 24.05.2014) (siehe S. 18, 25).
- [8] meinberg.de. *Atom Clock*. URL: http://www.meinberg.de/german/info/atomic_clock.htm (besucht am 24.05.2014) (siehe S. 12).
- [9] mono.rgbtechnology.pl. *GPS Zeitsynchronisation*. URL: <http://www.mono.rgbtechnology.pl/de/faq/gps-zeitsynchronisation.html> (besucht am 24.05.2014) (siehe S. 25).
- [10] ptb.de. *News zum DCF77*. URL: <http://www.ptb.de/de/aktuelles/archiv/presseinfos/pi2011/pitext/pi111107.html> (besucht am 24.05.2014).
- [11] ptb.de. *Zeitzeichensender DCF77 - Dokumentation*. URL: http://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/4.4_zeit_und_frequenz/4.42/dcf77.pdf (besucht am 24.05.2014).
- [12] astro.siggi.de. *Die Zeit aus dem Äther - wie Funkuhren funktionieren*. URL: <http://www.astro-siggi.de/tutorial-funkuhr.html> (besucht am 24.05.2014) (siehe S. 12).

- [13] stackoverflow.com. *How Does the Network Time Protocol Work?* URL: <http://stackoverflow.com/questions/1228089/how-does-the-network-time-protocol-work> (besucht am 24.05.2014) (siehe S. 25).
- [14] wikipedia. *Time signal*. URL: http://en.wikipedia.org/wiki/Time_signal (besucht am 24.05.2014) (siehe S. 12).
- [15] wikipedia.org. *Chronometer*. URL: <http://de.wikipedia.org/wiki/Chronometer> (besucht am 24.05.2014) (siehe S. 12).
- [16] wikipedia.org. *Funkuhr*. URL: <http://de.wikipedia.org/wiki/Funkuhr> (besucht am 24.05.2014) (siehe S. 13).
- [17] wikipedia.org. *Geschichtliche Entwicklung der Zeitübertragung per Funk*. URL: http://de.wikipedia.org/wiki/Geschichtliche_Entwicklung_der_Zeit%C3%BCbertragung_per_Funk (besucht am 24.05.2014) (siehe S. 12).
- [18] wikipedia.org. *Normalfrequenz*. URL: <http://de.wikipedia.org/wiki/Normalfrequenz> (besucht am 24.05.2014).
- [19] wikipedia.org. *Sonnenuhr*. URL: <http://de.wikipedia.org/wiki/Sonnenuhr> (besucht am 24.05.2014) (siehe S. 11).

Abbildungsverzeichnis

| | | |
|-----|--|----|
| 3.1 | Zeitzeichensender in Prangins (VD) | 16 |
| 3.2 | Sendeanlage in Mainflingen (Mainhausen, DE) | 17 |
| 3.3 | Reichweite des Signals | 18 |
| 3.4 | Beispielhaftes DCF77 Signal | 19 |
| 3.5 | DCF77-Codierung als Codierscheibe | 22 |
| 4.1 | Konsolen Output bei laufendem Server (mit DCF77 via USB) | 34 |
| 4.2 | HTTP Request auf Webserver | 34 |
| 4.3 | Client nach Aufruf (vollständig geladen und initialisiert) | 35 |
| 4.4 | Einstellmöglichkeiten via Web-GUI | 35 |
| 4.5 | Synchronisation ohne DCF77 Signal | 36 |
| 4.6 | Synchronisation mit gültigem DCF77 Signal | 36 |

