

HOCHSCHULE BREMERHAVEN

EMBEDDED SYSTEMS

Software Design Description

*Description of how the software will meet the requirements.
Also describes the rationale for design decisions taken.*

Autoren: Jan Löwenstrom (34937) und Johann Hoffer (34461)
begleitet von
Prof. Dr. Lipskoch

1. Februar 2019

Inhaltsverzeichnis

1	Introduction	3
1.1	Design Overview	3
1.2	Requirements Traceability Matrix	4
2	System Architectural Design	4
2.1	Chosen System Architecture	4
2.2	Discussion of Alternative Designs	6
2.3	System Interface Description	7
3	Detailed Description of Components	9
3.1	DCF77 Datentyp (rawDCF)	9
3.2	Interne Zeit Datentyp (ATime)	10
3.3	Algorithmus zur Analyse und Interpretation des DCF77-Signals	11
3.4	Paritätschecker	11
3.5	Decodierer	12
3.6	ATMega32 mit Evaluationsboard	12
3.7	Löt-Board	15
3.8	Telefondisplay	16
3.9	DCF77 Empfänger	17
4	User Interface Design	18
4.1	Description of the User Interface	18
4.2	Screen Images	20
5	Additional Material	21

Abbildungsverzeichnis

1	Systemübersicht	4
2	Darstellung der parallel ablaufenden Programmteile	5
3	Systembausteine	7
4	Systemkontext	8
5	ATMega32 auf dem Evaluationsboard	13
6	40er Flachbandkabel mit Kabelbrücken	13
7	Indexieren des 40er Flachbandkabels	13
8	Schaltplan Löt-Board	15
9	Löt-Board Vorder- und Rückseite	16
10	Telefondisplay	17
11	5V DCF77 Empfangsmodul von Reichelt	18
12	3,3V DCF77 Empfangsmodul von Pollin	18
13	Display Interface	20
14	Externes Netzteil	21
15	Allgemeiner Aufbau	22
16	DCF-Empfänger mit Anschlusskabeln	22
17	DCF-Empfänger am Fenster	23
18	Teilaufbau	24

Listings

1	Ordnerstruktur	3
2	Deklaration der internen Zeit (ATime)	10

Tabellenverzeichnis

1	Schnittstellen-Mapping	9
2	Pin Beschreibung des Evaluationsboard und deren Aufgabe	14

1 Introduction

1.1 Design Overview

Der Source-Code unterteilt sich zunächst in zwei Teile.

```
src/
  dcf/
  util/
  display/
  hardware/
  internClock/
  util/
  validation/
tests/
```

Listing 1: Ordnerstruktur

Zum einen gibt es einen Test-Teil (zu finden im //tests Ordner), in dem sämtliche Methoden, die der DCF-Dekodierung dienen, ohne Einsatz des ATMegas getestet werden können. Bei der Entwicklung dient dies dem Test-Driven-Developement, mit dem alle Methoden Stück für Stück weiterentwickelt und immer wieder getestet werden. Dies hilft dabei die Methoden so zu entwickeln, dass sie stets korrekt ablaufen können und dabei auch Randfälle abdecken. Dies ist essentiell dafür, dass das System im fertigen Zustand reibungslos laufen kann und gesichert ist, dass auch stets die korrekte Uhrzeit angezeigt wird.

Bei dem anderen Teil handelt es sich um denjenigen Part, der später auf dem ATMega dauerhaft läuft. Dieser enthält alle getesteten Methoden und ist ansonsten genau auf den ATMega zugeschnitten. Das bedeutet, dass dieser Teil nur auf der entsprechenden Hardware lauffähig ist und andernfalls nur mit einer speziellen Software simuliert werden kann. In diesem Fall handelt es sich dabei um das Programm „simavr“, dessen .vcd Output-Dateien mit dem Programm „GTK-Wave“ angezeigt werden können.

1.2 Requirements Traceability Matrix

2 System Architectural Design

2.1 Chosen System Architecture

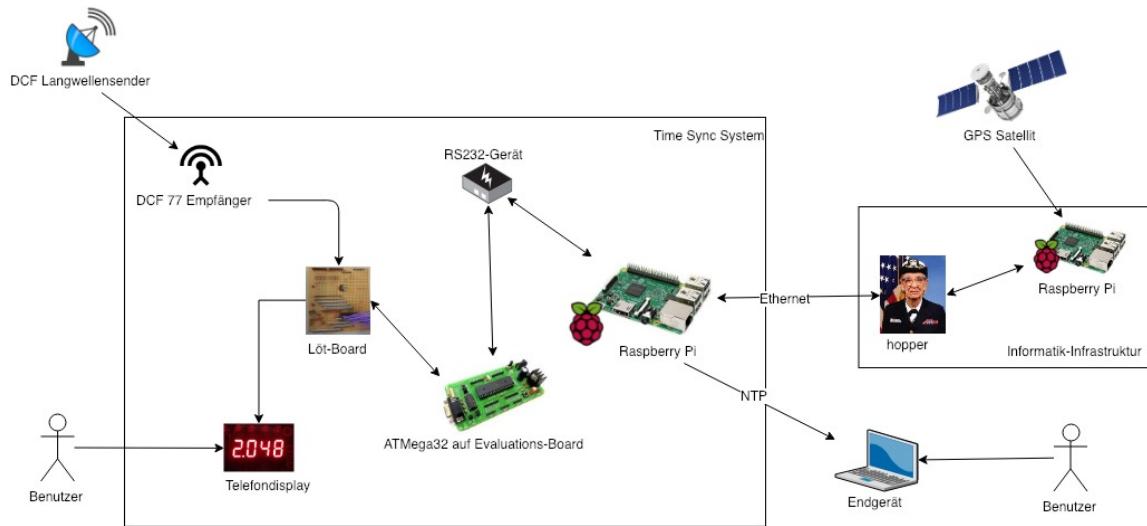


Abbildung 1: Systemübersicht

Für die Core-Features, nämlich das Empfangen des DCF77 Signals, die Auswertung dessen und die Darstellung auf einer Anzeige, wurde ein ATMega32 auf dem Atmel Evaluations-Board 2.0 von Pollin gewählt, der mit einem eigens konstruierten Löt-Board verbunden ist, welches wiederum als zentrale Einheit zur Vermittlung von Informationen zwischen ATMega, dem Telefondisplay und dem DCF77-Empfänger dient. Der ATMega ist somit in der Lage über das Setzen verschiedener Output-Ports dem Display Instruktionen mitzuteilen, oder über das Einlesen eines speziellen Pins die aktuelle Flanke des DCF77-Empfängers zu erfahren. Intern, also softwaretechnisch, ist dieses Einlesen über einen Abtastmechanismus gesteuert, welcher wiederum durch einen internen Interrupt, der jede Millisekunde ausgelöst wird, verwirklicht. Jede Millisekunde wird also überprüft, ob der Datenpin des DCF77-Empfängers HIGH oder LOW ist, wie aus dieser Information Schlüsse über das aktuelle Datum mit Hilfe eines Algorithmus gezogen werden, wird im Kapitel Algorithmus zur Analyse und Interpretation des DCF77-Signals genauer erklärt.

Die interne Zeit (ATime), welche sich, wenn möglich, mit der DCF77-Zeit synchronisiert, wird sekundenweise hochgezählt. Auch dies geschieht durch einen internen

Interrupt, der jede Sekunde ausgelöst wird und neben dem Hochzählen der Sekunden auch das Schreiben in den Display-RAM anstößt. Das Display weist also eine Refreshrate von einem Frame pro Sekunde auf, was völlig ausreichend ist, denn die primären Informationen über die Zeit weisen nur eine sekündliche Veränderung auf. Um die parallel ablaufenden Programmteile und das Zusammenspiel der Interrupts besser zu verstehen, haben wir folgendes Diagramm erstellt:

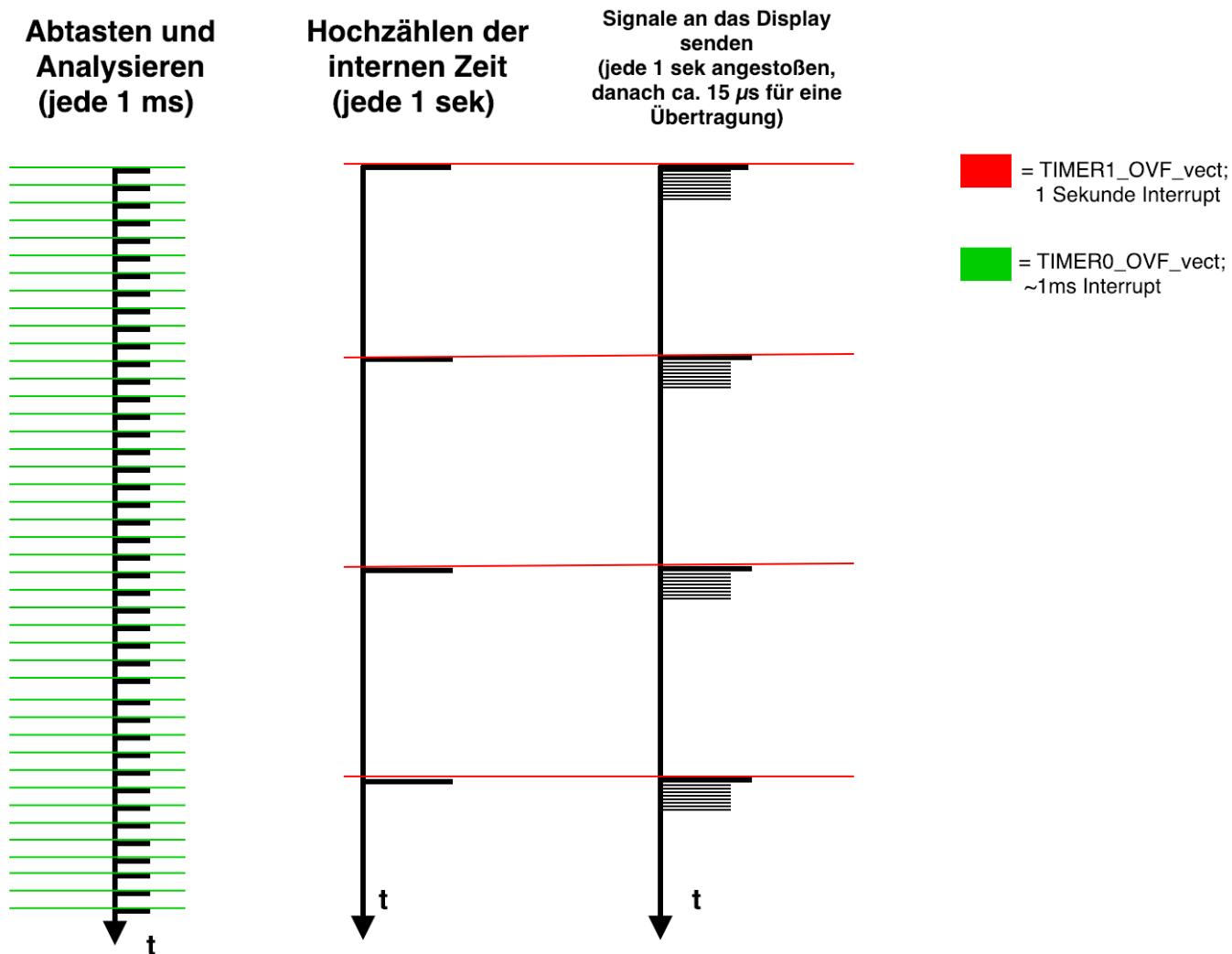


Abbildung 2: Darstellung der parallel ablaufenden Programmteile

Das Zur Umsetzung des Softwareanteils wurde ausschließlich die Programmiersprache C verwendet.

2.2 Discussion of Alternative Designs

Es gab alternative Ansätze für die Speicherung und das Processing der DCF-Folge. Eine Möglichkeit wäre es gewesen, die DCF-Reihenfolge direkt „on-the-fly“ zu untersuchen, um eine Art Stream-processing zu ermöglichen. Dabei würde jede empfangene 1 oder 0 direkt zu ihrer semantischen Gruppe zugeteilt werden. Bit 21 bis 27 wären also direkt mittels Bitshifting an ihre richtige Position des Attributs „minutes“ eines structs platziert worden. Zudem hätte man die Paritätschecks verteilt ansetzen können, nämlich immer dann, wenn ein Block vollständig übermittelt worden ist. Es gibt jedoch keinerlei Vorteile, die sich aus diesem Vorgehen ergeben, aber zugleich würde sich die Komplexität erhöhen. Im aktuellen System werden die Bits der DCF-Folge einfach direkt in ein boolesches Array gespeichert, wobei der Index gleich der Position des jeweiligen Bits in der Reihenfolge ist. Nachdem die Reihenfolge vollständig nach 59 Sekunden übertragen worden ist, wird der Paritätscheck ausgeführt, die einzelnen Datumsinformationen auf Korrektheit überprüft und zum Schluss mit der internen Zeit synchronisiert. Man hat eine komplette Sekunde Zeit, um diese Checks auszuführen, was mehr als ausreichend ist und wodurch es nicht nötig ist, diese Checks zeitlich aufzuteilen. Zudem würde auch kein Speicherplatz gespart werden, denn anstatt eines booleschen Arrays würde ein Struct mit mindestens genauso viel Speicher benutzt werden, wenn nicht sogar mehr, wenn jede Zeitinformation separat in einem 8-Bit Datentyp gespeichert wird.

Es wäre denkbar gewesen, dieses Projekt mit Assembler zu verwirklichen, wenn man bedenkt, dass wir schon Erfahrung mit dieser Art der Entwicklung gesammelt haben. Die Vorteile, die eine höhere Programmiersprache wie C bietet, sind jedoch zu gravierend. Die Nutzung eines Simulators (simavr) und ein Compiler, der das gesamte Speicher- und Registermanagement übernimmt, sind nur zwei von ihnen. Generell führt die Verwendung von C im Vergleich zu Assembler zu einer deutlich erhöhten Entwicklungsgeschwindigkeit und zudem zu einer lesbareren und kleineren Codebase.

Eine weitere Option wäre gewesen, den DCF-Empfänger als externen Interrupt zu verwenden, eventuell mit Zuhilfenahme des Oszilloskops. Jedes Mal, wenn das DCF-Empfangsmodul eine Rising Edge erfährt, feuert es einen Interrupt. Dies hätte jedoch ungewollte negative Konsequenzen haben können. Allgemein ist es in den wenigsten Fällen ratsam, eine externe Quellen seine Interrupts steuern zu lassen und damit seinen Programmfluss unkontrolliert zu beeinflussen. Im schlimmsten Fall, dass z.B. das Signal so gestört ist, dass fast permanent Edges erkannt werden, könnte dies zur Störung bei dem Senden an das Display führen und eine Blockade auslösen, sodass die interne Zeit nicht weiterhin erhöht werden kann oder das Programm gänzlich lahmgelegt wird.

2.3 System Interface Description

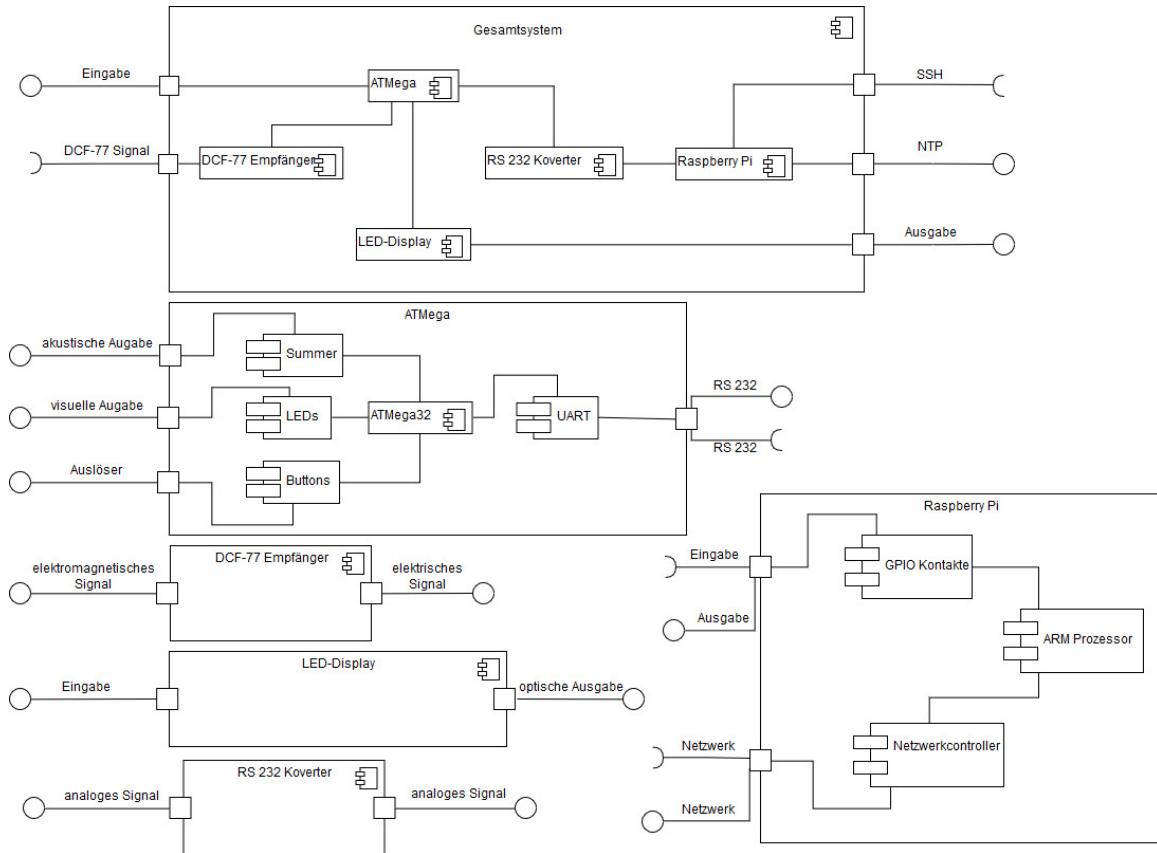


Abbildung 3: Systembausteine

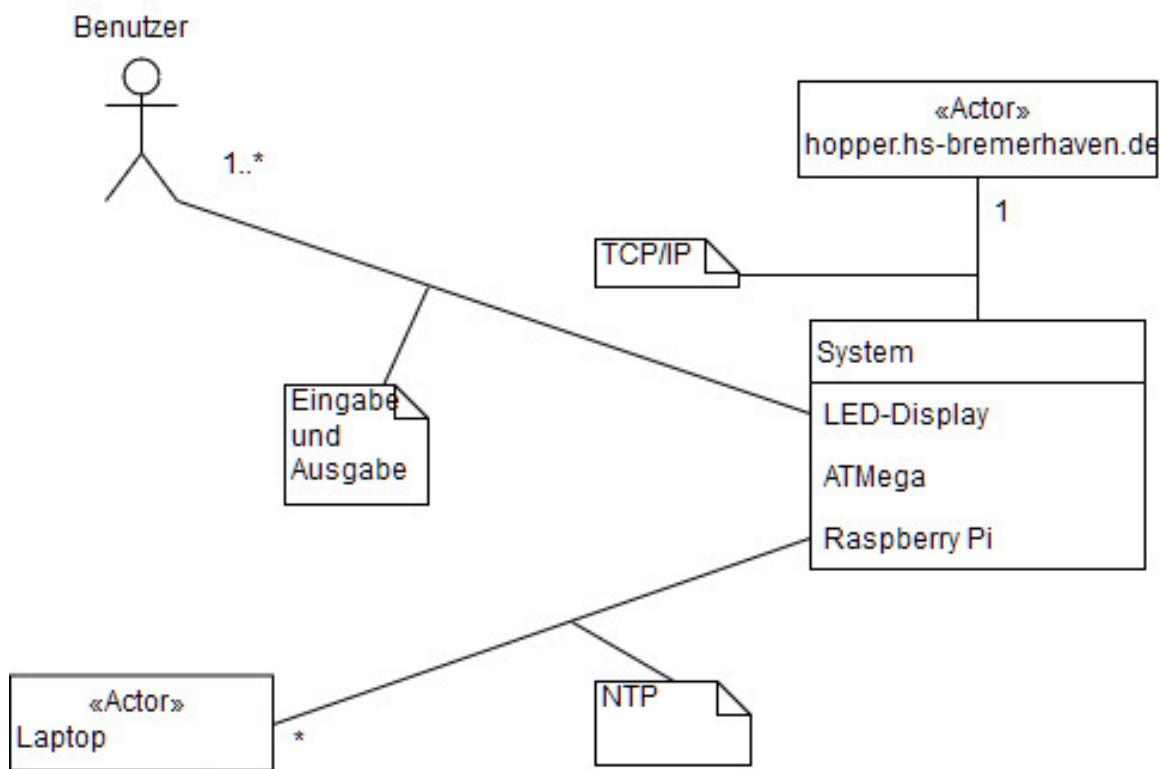


Abbildung 4: Systemkontext

PORT	DisplayInstr	DisplayPin	BoardPin	AtmegaPin	Notiz
/	Vcc	2	1	40	
/	GND	1	2	39	
/	VO Display Vcc	3	/	/	Kontrast Display
PC0	D/I	4	17	17	Auswertung
PC1	R/W	5	16	18	
PC2	Enable	6	15	19	Steuert Clock Cycle
PC3	Chip Select 1	15	5	21	Displayhälften
PC4	Chip Select 2	16	4	22	
PC5	Reset	17	3	23	
PA0	DB0	7	21	1	1x8 Pixel
PA1	DB1	8	20	2	
PA2	DB2	9	19	3	
PA3	DB3	10	18	4	
PA4	DB4	11	9	5	
PA5	DB5	12	8	6	
PA6	DB6	13	7	7	
PA7	DB7	14	6	8	
PD0	/	/	14	26	DCF Data Signal

Tabelle 1: Schnittstellen-Mapping

3 Detailed Description of Components

3.1 DCF77 Datentyp (rawDCF)

Die DCF-Folge wird in einem Bool-Array gespeichert. So können die einzelnen Bits, die in jeder Sekunde übertragen werden, direkt an den entsprechenden Stellen gespeichert werden.

```
#include <stdbool.h>
typedef bool DCF[59];
```

Ist die Folge komplett, d.h. sind alle 59 Bits übertragen, so werden alle Paritäten überprüft. Sind alle Paritätsbits korrekt gesetzt, kann mit der Dekodieren der DCF-Folge begonnen werden. Für die Dekodierung der DCF-Folge können nun direkt aus dem Array die einzelnen Werte aus den entsprechenden Bits an den bekannten Positionen verwendet werden. Jede herausgelesene Zeitinformation wird auf Korrektheit überprüft. Für den Fall, dass alle Daten valide sind, werden die einzelnen Zeitinformationen wie

Minuten, Stunden, Tage, Monate, etc. in ein struct gespeichert, welches die interne, nun synchronisierte Zeit (ATime) repräsentiert.

3.2 Interne Zeit Datentyp (ATime)

```
#include <inttypes.h>

typedef struct {
    uint8_t seconds;
    uint8_t minutes;
    uint8_t hours;
    uint8_t days;
    uint8_t months;
    uint8_t years_tens;
    uint8_t yearsHundreds;
    uint8_t weekdayIndex;
    char* weekdayString;
} AvrDatetime;

volatile AvrDatetime avrDatetime;
volatile AvrDatetime* p_avrDatetime;
```

Listing 2: Deklaration der internen Zeit (ATime)

Die Software definiert eine interne Zeit, die von uns als „ATime“ bezeichnet wird. Dieses struct repräsentiert die aktuelle Zeit, die auch sekundenweise auf dem Display ausgegeben wird. Des Weiteren wird diese Zeit intern jede Sekunde durch einen Interrupt um eine Sekunde erhöht, da das DCF-Signal nur minutenweise synchronisiert. Da wir mit einer 8bit Architektur arbeiten, ist es von Vorteil, alle Informationen als „uint8_t“ darzustellen. Da die Angabe zum aktuellen Jahr z.B. 2019 die Grenze von 256 überschreitet, ist es notwendig diese Information weiter zu unterteilen, nämlich in years_tens und yearsHundreds.

Ein Beispiel: Das Jahr 2019 wird in diesem struct dargestellt durch years_tens=19 und yearsHundreds=20

Das Attribut „weekdayIndex“ gibt an, welcher Wochentag gerade ist. Die Kodierung des Wochentages erfolgt gemäß der Norm ISO 8601 oder DIN EN 28601. Somit ergeben sich folgende Indizes (Darstellung auf dem Display wie nach dem Komma):

- 1 = Montag, Mo
- 2 = Dienstag, Di
- 3 = Mittwoch, Mi

- 4 = Donnerstag, Do
- 5 = Freitag, Fr
- 6 = Samstag, Sa
- 7 = Sonntag, So

3.3 Algorithmus zur Analyse und Interpretation des DCF77-Signals

Das Abtasten des Signals findet in der Datei signalToDCF.c statt. Die Methode evaluateSignal wird jede Millisekunde aufgerufen. Hauptsächlich geht es darum, die 100ms- und 200ms-Erregungen des DCF-Empfänger zu erkennen. Unsere Methode realisiert dies über zwei Zustände, weshalb es 4 Fälle gibt.

Fall 1: pinC steht auf 1 und Zustand ist 0: Der Zustand wird auf 1 gesetzt. Wurde zuvor mindestens 1700-Mal eine 0 gelesen, wird der Start einer neuen Minute erkannt.

Fall 2: pinC auf 1 und Zustand ist 1: Der Zähler der 1en wird einmal inkrementiert.

Fall 3: pinC auf 0 und Zustand ist 1: Zunächst wird auch eine 1 gezählt. Außerdem wird der 1-missed Zähler erhöht. So wird vermieden, dass in den Zustand 0 gewechselt wird, wenn durch Störungen eine kurze Unterbrechung der Erregung stattfindet. Wurde nun mindestens 10-Mal in Folge eine 0 gelesen, wird in den Zustand 0 gewechselt und ausgewertet, wie oft zuvor eine 1 gezählt wurde. War dies mindestens 150-Mal der Fall, gehen wir davon aus, dass eine 1 übertragen wurde. Wurde weniger häufig eine 1 gelesen, aber noch mindestens 70-Mal, wurde mit sehr hoher Wahrscheinlichkeit eine 0 übertragen. Falls bei einer 0 zuvor ein Minutenstart festgestellt wurde, wird geprüft, ob 60 Sekunden vergangen sind und im positiven Fall die interne Zeit mit der neu empfangenen Zeit und dem Minutenstart synchronisiert. Bei noch weniger 1en handelt es sich vermutlich um eine Signalstörung.

Fall 4: pinC auf 0 und Zustand 0: Der Zähler für die Nullen wird einmal erhöht. Wurden zuvor mindestens 3000-Mal eine 0 gezählt, gehen wir von einem verlorenen Signal aus.

3.4 Paritätschecker

Der Paritätschecker zählt in den jeweiligen Bereichen der DCF-Folge die 1-Bits. Ist die Anzahl gerade, muss das Paritätsbit auf 0 stehen, bei einer ungeraden Anzahl von 1-Bits muss die Parität eine 1 sein. Sind alle Paritätsbits korrekt, so kann mit der Dekodierung begonnen werden.

3.5 Decodierer

Das Bit 0 des DCF-Signals ist immer 0. Mit den folgenden 14 Bits werden Wetterdaten übertragen, für dessen Kodierung eine besondere Lizenz und ein spezieller Chip benötigt werden. Bit 15-19 werden für spezielle Informationen benutzt. Darunter fallen Sommer- und Winterzeit und deren bevorstehende Umstellung. Außerdem kann dort eine Schaltsekunde oder das Vorliegen einer Störung gekennzeichnet werden. Ab Bit 20 werden dann die Zeit- und Dateninformationen kodiert. Jeder Wert wie Stunde oder Minute wird durch eine individuelle Bitfolge dargestellt. Diese werden teilweise in Zehner und Einer unterteilt, bei den Zehnern muss der Wert der Bitfolge mit 10 multipliziert werden.

Da die DCF-Folge eins-zu-eins in das DCF-Array eingefügt wurde, können die Berechnungen direkt mit den jeweiligen Indizes durchgeführt werden. Dies findet im dateExtractor.c statt, der die Bits an die Helpermethoden des bitConverter.c übergibt. Falls es sich um Bits handelt, die Zehner repräsentieren, wird das vom bitConverter.c zurückgelieferte Ergebnis mit 10 multipliziert. Die Ergebnisse werden dann im AvrDatetime-struct gespeichert. Nachdem dieser Vorgang abgeschlossen wurde, findet die Validierung der errechneten Werte statt.

3.6 ATMega32 mit Evaluationsboard

Als Prozessor wurde der ATMega32 genutzt. Dieser läuft mit 5V und einer Taktrate von 16MHz, besitzt 32 Arbeitsregister, 2KB RAM und hat eine 8Bit Architektur. Der ATMega32 ist auf einem Atmel Evaluationsboard 2.0 von Pollin verbaut. Verbindet man das Evaluationsboard mittels eines ISP mit einem Computer, so ist es möglich, sein Programm in den Programmflash zu schreiben. Die ausführliche Dokumentation zum ATMega32 ist aufrufbar unter:

[Link: Datenblatt-ATMega32](#)

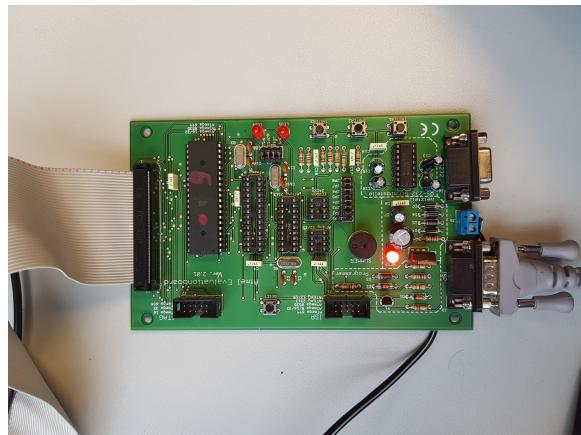


Abbildung 5: ATMega32 auf dem Evaluationsboard

Mit dem passenden 40er Flachbandkabel wird das Evaluationsboard durch den Wannenstecker mit dem Löt-Board verbunden.

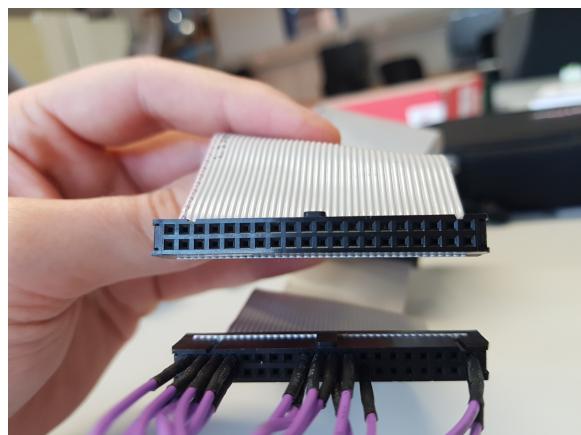


Abbildung 6: 40er Flachbandkabel mit Kabelbrücken



Abbildung 7: Indexieren des 40er Flachbandkabels

Pin-Index	Wohin	Aufgabe
1	PA0	DB0 des Displays steuern
2	PA1	DB1 des Displays steuern
3	PA2	DB2 des Displays steuern
4	PA3	DB3 des Displays steuern
5	PA4	DB4 des Displays steuern
6	PA5	DB5 des Displays steuern
7	PA6	DB6 des Displays steuern
8	PA7	DB7 des Displays steuern
...	-	PORTB wird nicht benutzt
17	PC0	Steuert D/I vom Display. HIGH= Data Input, LOW= Instruction Code Input
18	PC1	Steuert R/W vom Display. HIGH=Data Read, LOW= Data write
19	PC2	Steuert Enable vom Display. Eine Art Clock, die immer von HIGH auf LOW wechselt
20	-	ACHTUNG! Pin 20 wird übersprungen Steuert Chip Select für IC1 vom Display.
21	PC3	Ist permanent auf 1 um die linke Hälfte des Display auszuwählen
22	PC4	Steuert Chip Select für IC2 vom Display. Ist permanent auf 0
		Steuert Reset vom Display.
23	PC5	Wird beim Start kurz auf LOW gezogen, bleibt dann immer HIGH
...	-	Restlichen Pins von PORT C werden nicht benutzt
26	PD0	Einlesen des DCF-Datensignals
...	-	Restliche Pins von PORTD werden nicht benutzt
39	GND	Masse
40	VCC	Versorgungsspannung 5V

Tabelle 2: Pin Beschreibung des Evaluationsboard und deren Aufgabe

3.7 Löt-Board

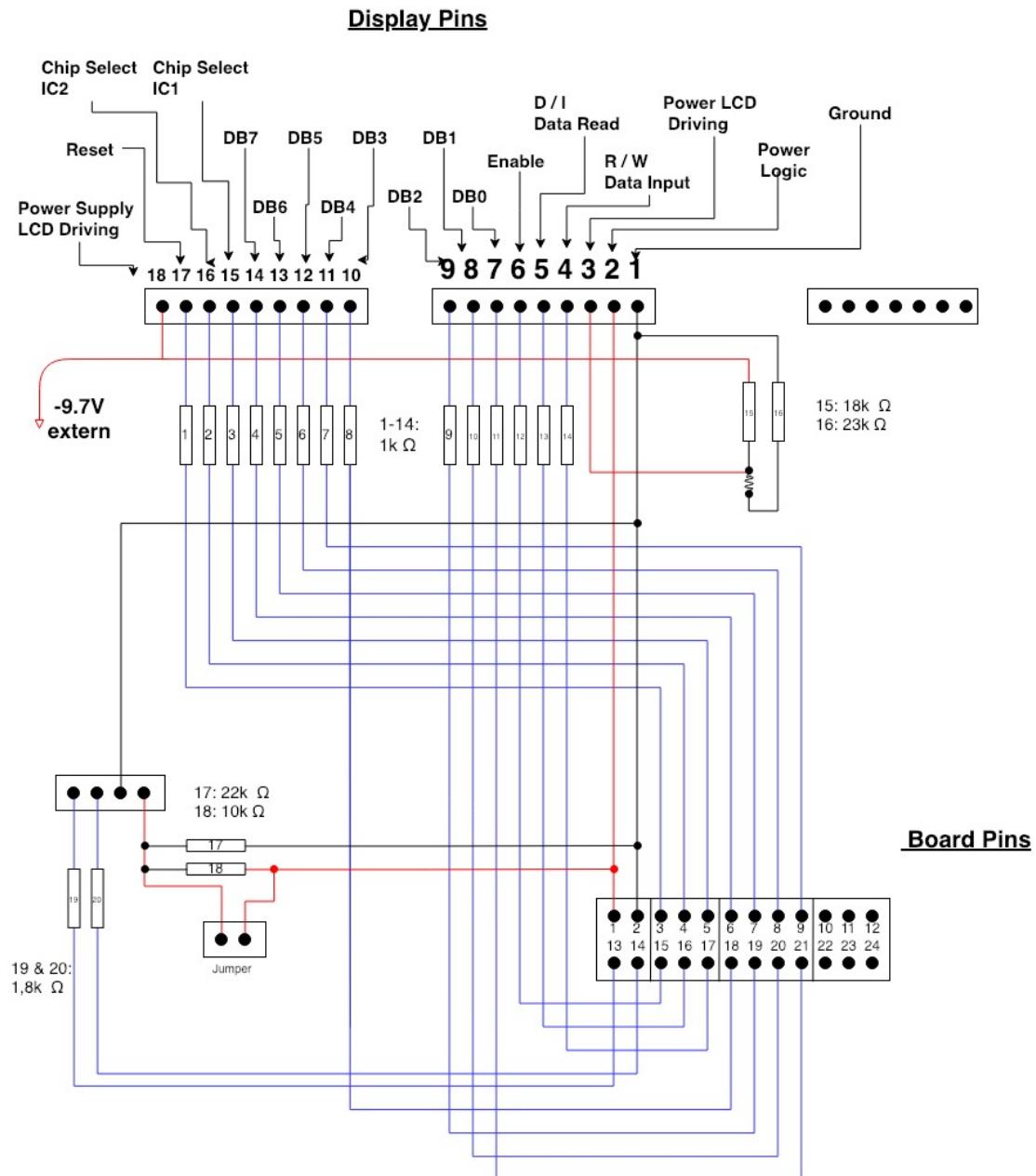


Abbildung 8: Schaltplan Löt-Board

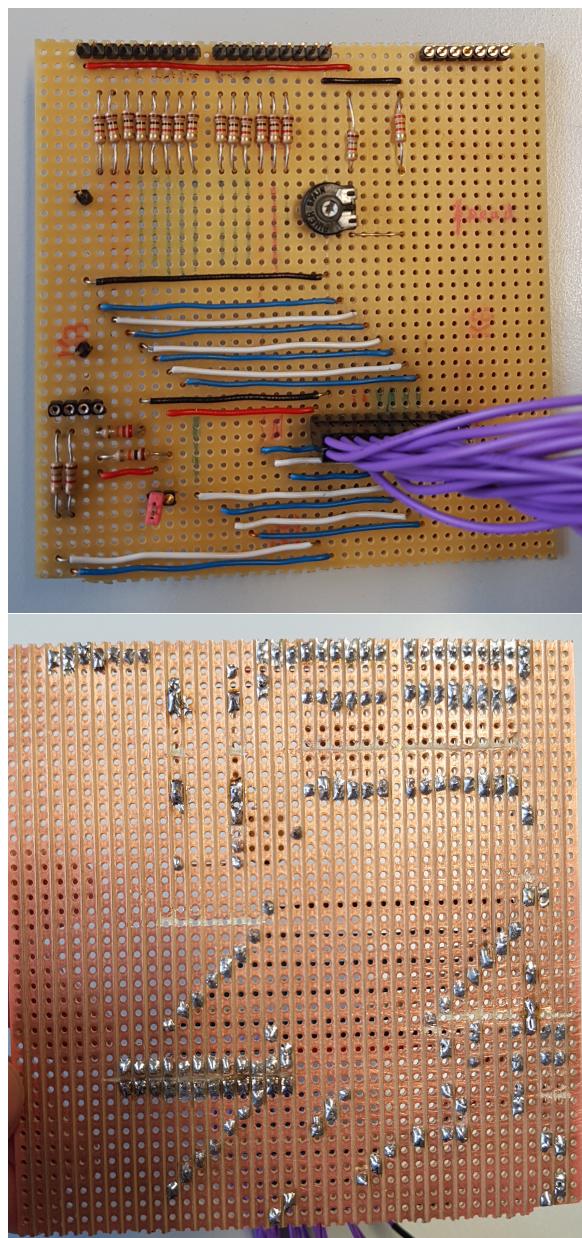


Abbildung 9: Löt-Board Vorder- und Rückseite

3.8 Telefondisplay

Das Display wurde aus einem Telefon ausgebaut. Während der Umsetzung des Projekts stießen wir auf zwei unterschiedliche Modelle. Das eine Modell ist in der Lage die nötige Spannung von 9,7V für den Displaykontrast selbst zu erzeugen. Jedoch ist unser

Modell nicht in der Lage dies zu tun und braucht eine externe Stromversorgung. Durch ein externes Netzteil wird eine Spannung von 9,7V erzeugt und an den Display-Pin mit dem Index 18 und das Potentiometer auf dem Löt-Board angelegt. Für eine detaillierte Beschreibung wie das Display funktioniert und welche Instruktionen es zu welcher Zeit benötigt, empfiehlt sich ein Blick in die Dokumentation des Displays:
Link: Datenblatt-Display



Abbildung 10: Telefondisplay

3.9 DCF77 Empfänger

Es lassen sich zwei unterschiedliche DCF77 Empfängermodule an das Löt-Board anschließen. Zum einen der 5V Empfänger von Reichelt und zum anderen den 3,3V Empfänger von Pollin. Wie in der Beschreibung des Löt-Boards erläutert, reicht es den Jumper zu benutzen oder nicht, um zwischen dem 3,3V und dem 5V Gerät zu tauschen. **Dabei muss der Jumper zwingend gezogen werden, wenn das 3,3V Gerät genutzt werden soll!** (Damit der 10k Ohm Widerstand nicht überbrückt wird) Außerdem ist zu beachten, dass bei den im Projekt eingesetzten Modulen die Pin-Reihenfolgen verschieden sind. Dabei ist zu beachten, dass jeweils die Pins für das Datensignal („Signal“, „TCO“) und das Ein- und Ausschalten („(EIN)“, „PON“) vertauscht sind.

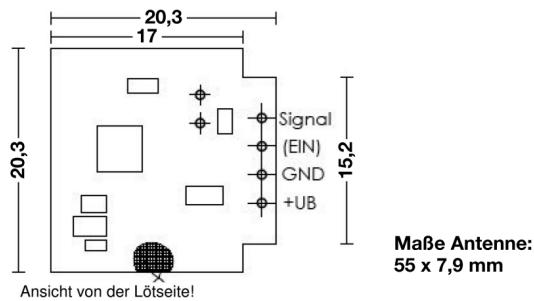


Abbildung 11: 5V DCF77 Empfangsmodul von Reichelt

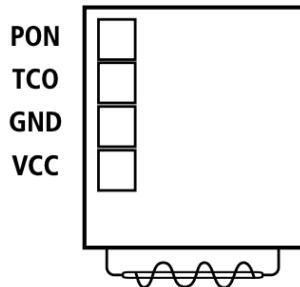


Abbildung 12: 3,3V DCF77 Empfangsmodul von Pollin

[Link: Datenblatt-Reichelt-DCF-Empfangsmodul-5V](#)
[Link: Datenblatt-Pollin-DCF-Empfangsmodul-3,3V](#)

4 User Interface Design

4.1 Description of the User Interface

Die interne Zeit (ATime) wird der Umwelt durch ein altes Telefondisplay sichtbar gemacht. Dabei erscheinen auf der linken des Displays neben den universal verständlichen Datumsinformationen auch noch Zusatzinformationen. Die oberste Zeile des Display hat den Index 0, die unterste den Index 7 und sie stellen folgende Informationen dar:

- Zeile 0: Uhrzeit im Format HH:mm:SS (24-Stunden)
- Zeile 1: Datum im Format dd.mm.yyyy
- Zeile 2: Wochentag (Mo, Di, Mi, Do, Fr, Sa, So)

- Zeile 3: Derzeitiger Index bzw. Position an der die letzte 1 oder 0 in das rawDCF-Array geschrieben worden ist. Im Code ist dies die globale Variable „g_position“. Ist die Uhrzeit synchronisiert und der Empfang des DCF77-Signals gut, so ist der Index im Einklang mit der Sekunde der Uhrzeit. Steigt der Index nicht gleichmäßig um jeweils den Wert 1, sondern springt von z.B. 14 auf 23, so ist davon auszugehen, dass das Signal so gestört ist, dass der Abtastalgorithmus zu viele Einsen oder Nullen pro Sekunde erkennt.

Die Tröte neben dem Index wird dann angezeigt, wenn ein Teammitglied an dem angezeigten Datum Geburtstag hat.

- Zeile 4: „E“steht für Empfang. Wird ein positiver Smiley angezeigt, so ist das Signal nicht unterbrochen. Kommen mindestens 4 Sekunden keine Ausschläge durch den DCF77-Empfänger zustande, so signalisiert ein trauriger Smiley eine Störung des Signals.
- Zeile 5: „MNS“steht für minutes not synced“, also die Zeit, die vergangen ist in Minuten, seitdem die interne Zeit das letzte Mal mit der Frankfurter Zeit synchronisiert worden ist.
- Zeile 6: „SS“steht für SSync Status“. Ist der Wert 1, so bedeutet dies, dass die angezeigte Zeit aktuelle ist und synchronisiert ist. Als Absicherung wird dieser Status auf 0 gesetzt, wenn es seit mindestens einer Stunde zu keiner Synchronisation gekommen ist.

„Sa“steht für das Einfügen einer Schaltsekunde am Ende der Stunde. Wird in einer Stunde mindestens 40 Mal erkannt, dass eine Schaltsekunde folgt, so wird dieser Wert auf 1 gesetzt.

- Zeile 7: „Err“zeigt an, welcher ErrorTyp in der letzten Minute geflogen ist. Mögliche ErrorTypen sind:

- 0: Kein Error
- 1: Es wurden zu schnell, zu viele Einsen und Nullen eingelesen
- 2: Es wurden zu wenige Einsen und Nullen innerhalb einer Minute erkannt
- 3: Der Paritätscheck schlug fehl
- 4: Der Datumscheck schlug fehl

4.2 Screen Images



Abbildung 13: Display Interface

5 Additional Material



Abbildung 14: Externes Netzteil

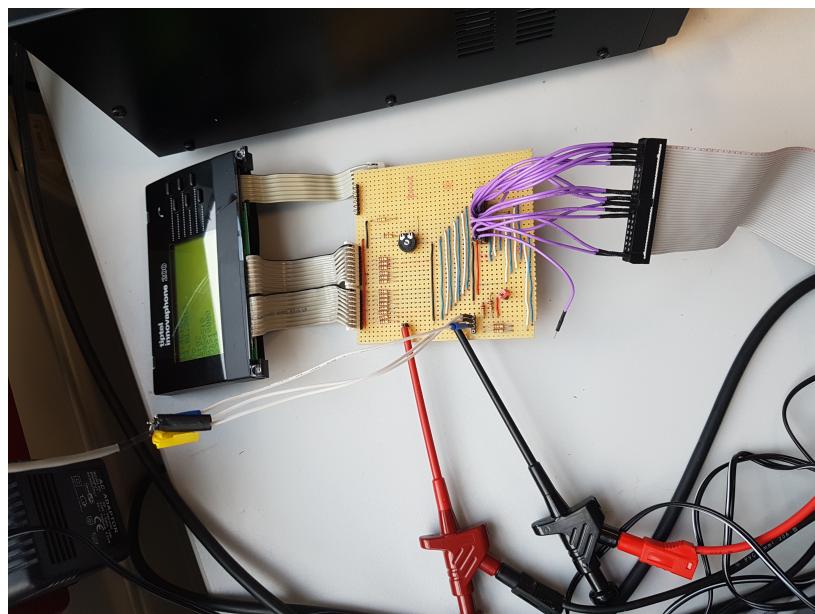


Abbildung 15: Allgemeiner Aufbau

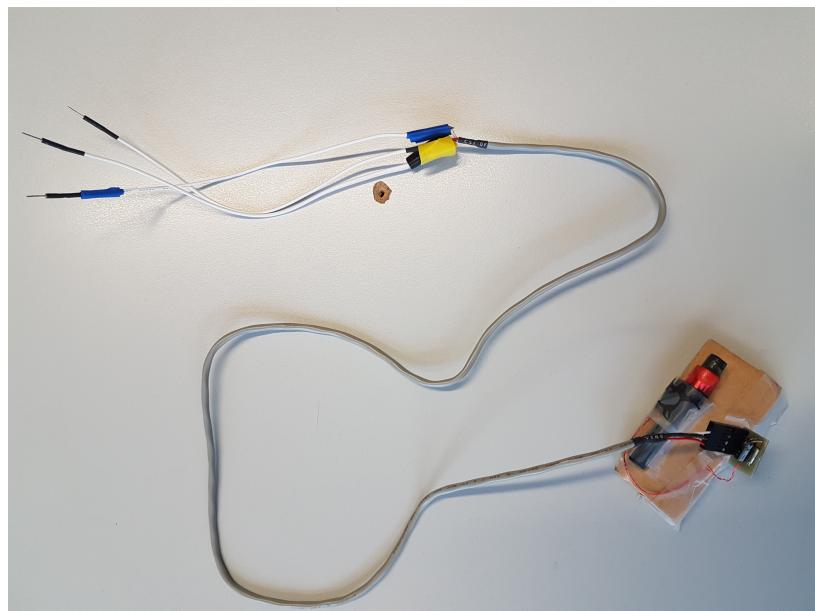


Abbildung 16: DCF-Empfänger mit Anschlusskabeln



Abbildung 17: DCF-Empfänger am Fenster

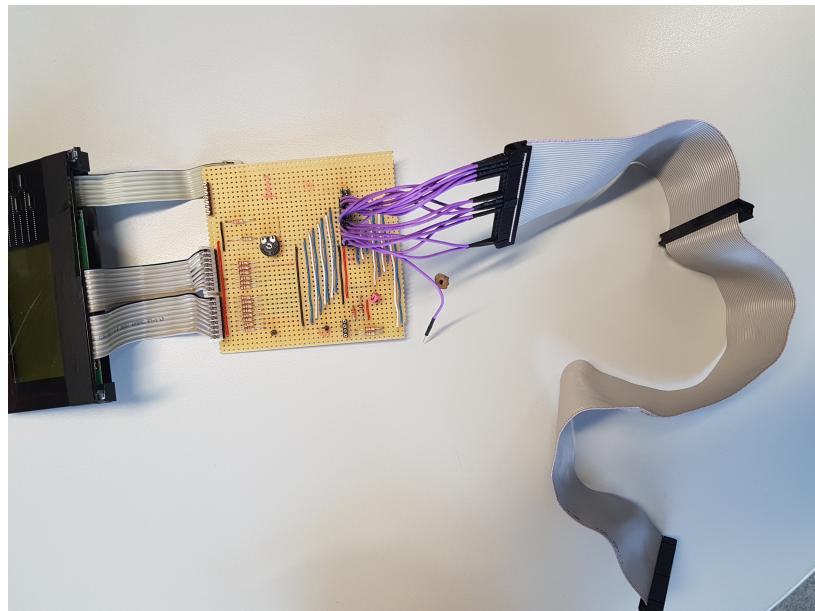


Abbildung 18: Teilaufbau