# DCF77 gesteuerte Stationsuhr und NCDXF-Bakenmonitor

## Gerald Moritz, OE3GOD

Zusammenfassung

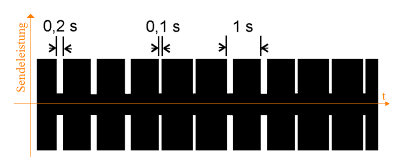
Seit mehr als zehn Jahren gibt es den „Arduino“ am Markt. Dabei handelt es sich um eine Mikrocontrollerentwicklungsplattform, die wegen ihrer einfachen Handhabung immer beliebter wird.

Im Internet findet man dafür vermehrt sehr interessante Anwendungen auch für den Amateurfunk. Als Einstieg wurde nach einem Projekt gesucht, das mich näher an diese Thematik heranführt aber bereits an der Funkstation einen wertvollen Beitrag leistet. Durch meine Beobachtungen des 10m Bandes kam dann rasch auch der Wunsch auf, den Sendeplan der NCDXF-Baken gemeinsam mit der Uhrzeit anzuzeigen.

DCF77

Vom Standort Mainflingen werden auf der Frequenz 77,5 kHz bereits seit 1959 Zeitzeichen gesendet.

Dabei wird das Trägersignal (50 KW) für 0,1 bzw 0,2 Sekunden auf 15% herabgesenkt.

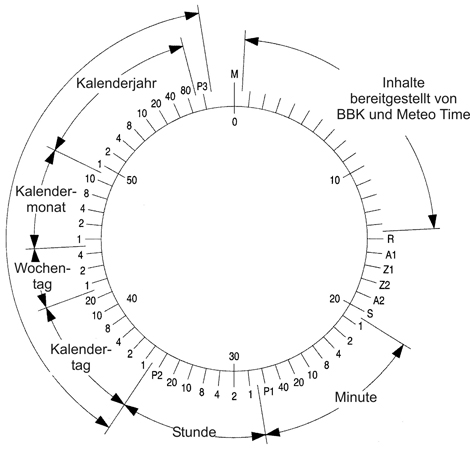


Die binäre Kodierung erfolgt dabei über die Länge der Absenkung.

0,1 Sekunde = binär 0

0,2 Sekunden = binär 1

Die Bedeutung der einzelnen Werte ist in der folgenden Zeittafel abgelegt. Insgesamt werden 59 „Werte“ übertragen, in der Sekunde 59 erfolgt keine Absenkung.

Die Werte von 0 bis 14 enthalten Lizenzpflichtige Wetterdaten bzw. Alarmierungsdaten im Katastrophenfall, zwischen 15 und 20 gibt es die Werte für die Zonenzeit (MEZ/MESZ) und etwaige Schaltsekunden, die eigentliche Zeit- und Datumskodierung beginnt ab Sekunde 20.Dabei werden die Werte immer im Voraus gesendet und mit der Sekunde 0 beginnt diese Minute.

Beispiel Jahr 2016: 01101000

0 \* 1

1 \* 2

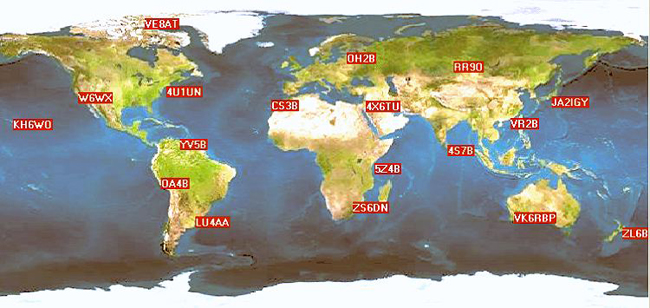
1 \* 4

0 \* 8

1 \* 10

Die NCDXF-Baken

Das Bakennetz der DX-Foundation von Nord Kalifornien (NCDXF) besteht aus 18 zeitsynchronisierten Sendern, welche das Rufzeichen und ein Trägersignal mit unterschiedlichen Ausgangsleistungen senden.



Das Rufzeichen wird mit ca. 22 WpM mit 100 Watt gesendet, danach erfolgt ein 1 Sekunden Träger mit 10 Watt, 1 Watt und 100 mW.

Insgesamt senden diese 18 Sender je 10 Sekunden auf den jeweiligen Frequenzen, somit kann man innerhalb von 3 Minuten (180 Sekunden) die Ausbreitungsbedingungen überprüfen.

Der Sendeplan beginnend bei Sekunde Null zur vollen Stunde.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Call |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4U1UN | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| VE8AT |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |
| W6WX |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |  |  |
| KH6RS |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |  |
| ZL6B |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |
| VK6RBP |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |
| JA2IGY |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |
| RR9O |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |  |
| VR2B |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |  |
| 4S7B |  |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |  |
| ZS6DN |  |  |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |  |
| 5Z4B |  |  |  |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |  |
| 4X6TU |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |  |
| OH2B |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 | 5 |
| CS3B | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 2 | 3 | 4 |
| LU4AA | 4 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 2 | 3 |
| OA4B | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 2 |
| YV5B | 2 | 3 | 4 | 5 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 = 14.100 kHz | 2 = 18.110 kHz | 3 = 21.150 kHz | 4 = 24.930 kHz | 5 = 28.200 kHz |

Zu allen anderen Zeiten erfolgt keine Aussendung.

Schaltungsbeschreibung

Taster für Bandwahl

Arduino

(UNO)

DCF77

Empfänger

LCD Display

2 x 16 Zeichen

(I²C)

DS1307

Uhrenmodul

(I²C)

Für den Einstieg in das Thema Arduino hab ich mir den Arduino kompatiblen „SainSmart UNO R3 DIP ATMega328“ ausgesucht, da er zum Zeitpunkt der Anschaffung günstig und lagernd war. Für eigene Projekte kann man den ATMEGA328PU selber programmieren. Um diesen Chip zum Laufen zu bringen braucht man neben der 5V Stromversorgung nur noch einen 16MHz Quarz und 2 x 22 pF Kondensatoren und hat einen platzsparenden Mikrocontroller.

Viele Zeitprojekte arbeiten mit einem günstigen GPS-Modul, für den Beginn war mit die DCF77 Empfangsmöglichkeit etwas leichter zu realisieren. Der DCF77-Empfänger besteht aus einer Ferrit-Antenne und einer Miniaturplatine mit vier Anschlüssen:

1. GND
2. VCC (2,5V – 15V)
3. DCF-Ausgang
4. DCF-Ausgang invertiert

Ausgänge 3 und 4 haben einen offenen Kollektor, ein 10kOhm Pull-up Widerstand wurde verwendet.

Das Uhrenmodul (DS1307) enthält neben der batteriegepufferten Uhr noch ein eEprom für max. 56 Byte.

Beide Teile (Uhr und Speicher) werden über den I²C-Bus angesprochen.

Über diese Zweidrahtleitung ist auch das LCD-Modul angeschlossen, da ansonsten viele der insgesamt 16 Anschlüsse am LCD-Modul zu verdrahten gewesen wären. Egal, ob die anzuzeigenden Daten über den I²C-Bus kommen, oder parallel, der Unterschied liegt nur in der Initialisierung am Beginn des Programms, die Befehle zur Anzeige sind in beiden Versionen gleich. Die ersten Experimente machte ich mit einem (parallelen) blauen LCD-Display mit weißer Schrift. Bei der Verwendung eines I²C LCD-Moduls gibt es den Nachteil, daß die Display-Hintergrund Beleuchtung nicht dimmbar ist. Aber sogar bei hellstem Sonnenschein an der Funkstation ist die Uhrzeit und das aktuelle Bakenrufzeichen wunderbar zu lesen.

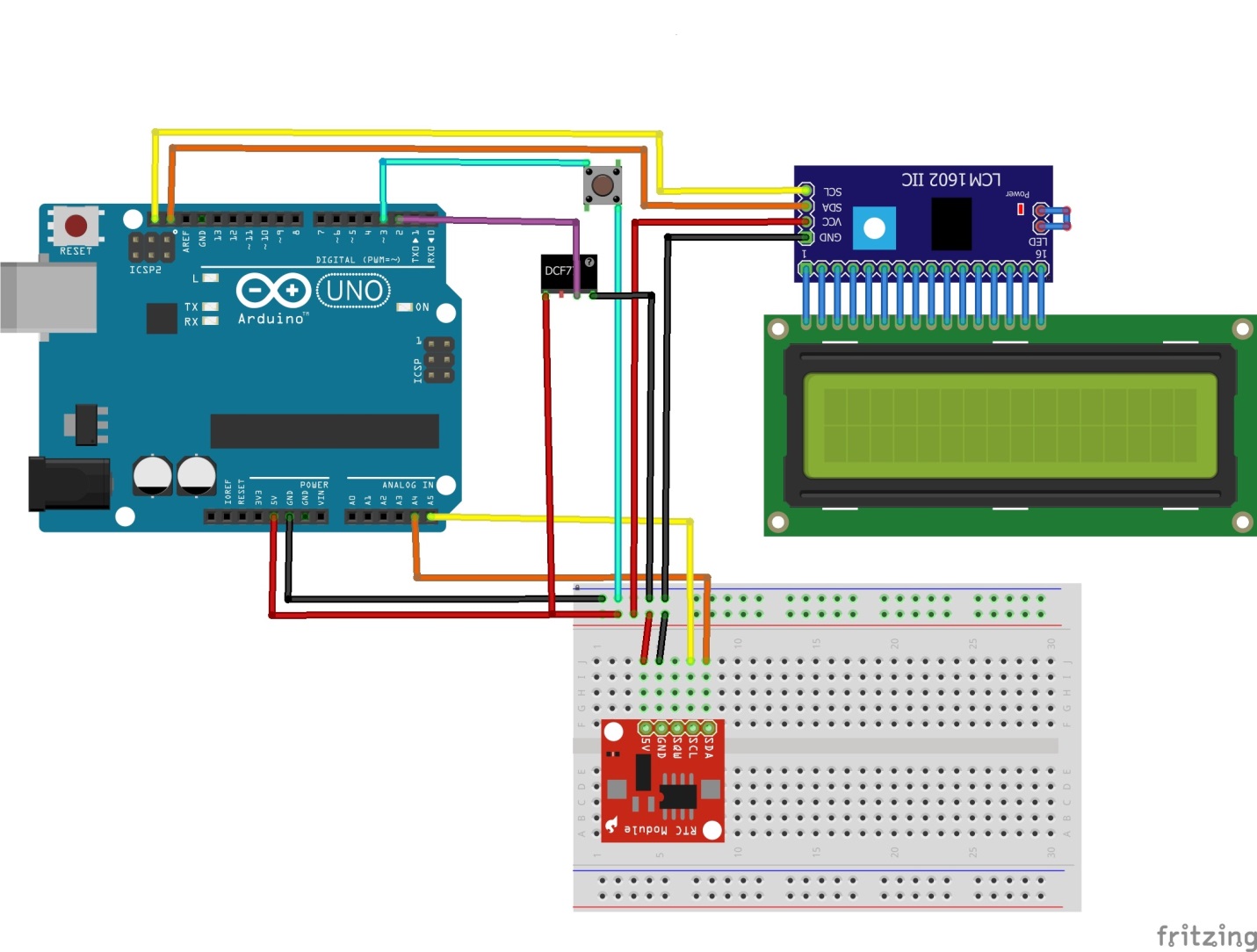
Da die Bakenstationen auf fünf Bändern senden, benötigte ich eine Umschaltung für die jeweiligen Frequenzen. In meiner Bastelkiste hatte ich einen kleinen Taster, mit diesem kann das gewünschte Bakenband ausgewählt werden. Mit einer größeren Anzeige wäre die Darstellung aller Rufzeichen auch ohne Umschaltung möglich.



Beispiel der DCF77 Bakenanzeige

Am Sonntag, 16.10. um 09:18:58 (UTC) sendete VE8AT auf 28.200 kHz.

Das „Blockschaltbild“ ist eigentlich bereits der gesamte Schaltplan, einzig der Pullup-Widerstand am DCF77-Modul ist nicht eingezeichnet. Die einzelnen Module erhalten ihre Spannungsversorgung vom Arduino (rot/schwarz), der I²C-Bus muss nicht von unterschiedlichen Pins zur Peripherie geführt werden, die Geräte können hintereinander geschaltet werden (gelb – SCL, orange - SDA).



Das DCF77 Modul (Pin 4) legt im Takt der ausgesendeten Impulse die Versorgungsspannung gegen Masse, daher auch der 10kOhm Widerstand (max. 1mA Schaltstrom). Am Pin3 kann über einen Transistor eine LED betrieben werden, die einzelnen Zeitimpulse (100 bzw. 200 Millisekunden) können damit sichtbar gemacht werden. Gleichzeitig kann aber auch der Empfang kontrolliert werden, im Falle einer Störung sieht man dann nur ein unkontrolliertes Aufflackern.

Für die Dekodierungsarbeit werden fertige Bibliotheken in den Code eingebunden, die man sich frei aus dem Internet holen kann. Zum Entziffern wird ein Interrupt verwendet (also eine Unterbrechung der bestehenden Verarbeitung) wenn das Signal vom Empfänger von High auf Low wechselt.

Nach dem Einschalten benötigt das Programm grundsätzlich mehrere Minuten, bis alle Zeitinformationen eingelangt sind. Wenn jedoch kein Empfang möglich ist (z.B. innerhalb des Hauses oder Störungen in der Umgebung, aber auch wenn der Zeitzeichensender wegen Defekt oder sonstigen Problemen nicht sendet) gibt es auch keine Information über die aktuell aktiven Baken.

Daher wurde die Grundschaltung um das günstige RTC (Real Time Clock) Modul DS1307 erweitert. Nach dem Ausschalten bleibt die Uhrzeit batteriegepuffert erhalten. Teurere Module bieten temperaturkompensierte Oszillatoren, diese sind aber für diese Anwendung nicht notwendig.

Apropos Versorgungsspannung: Die Programmierung inklusive der Spannungszufuhr erfolgt über die USB-Schnittstelle, beim Betrieb muß jedoch ein stabilisiertes Netzteil bzw. Batterie verwendet werden, bei USB-Versorgung hat die Uhr bisher nicht funktioniert. Für diesen Zweck hat die Arduino-Platine einen eigenen Stromanschluss (7-12 Volt).

Der Ablauf

Mit dem Einschalten wird zuerst geprüft, ob das Uhrenmodul vorhanden ist und ob eine Zeitinformation bereits gespeichert wurde. Im Fehlerfall wird eine Meldung angezeigt und mit dem Empfang des DCF77 Signals begonnen. Sollte eine Uhrzeit gespeichert sein, so wird diese in den Speicher geholt und angezeigt. Somit wird auch der aktuelle Bakensender ermittelt.

Abhängig von der Uhrzeit und dem eingestellten Band wird das sendende Rufzeichen berechnet. Mit den 10 Sekunden dauernden Aussendungen der 18 Sender hat man innerhalb von drei Minuten einen Überblick über die aktuellen Ausbreitungsbedingungen.

Das Uhrenmodul hat eine (angebliche) Ungenauigkeit von einer Minute pro Jahr. Aus Beispielen wurde der Code übernommen, daß alle 30 Sekunden die interne Arduino-Uhr an die DCF77 Uhrzeit angepaßt wird. Zu jeder vollen Stunde wird die Uhrzeit im Uhrenmodul gespeichert (falls dieses vorhanden ist).

Somit sollte die Ganggenauigkeit kein Problem mehr darstellen. Ein Betrieb ohne dem RTC ist möglich.

Aktuell gibt es diese Uhr nur als Prototyp, ein Gehäuse ist grundsätzlich in Arbeit, jedoch wird bereits an einer Erweiterung für eine CQ-Rufmaschine gearbeitet (mittels ISD-Sprachspeicher IC). Reizvoll wäre sicher auch eine Landkarte wie auf Seite 2 zu „bauen“, bei der die einzelnen Bakenstandorte mittels LED aktiviert werden (mittels Schieberegister 74HC595).

Anmerkungen zum Source-Code:

Dies ist mein erstes Arduino-Projekt, es kann sicherlich noch Unsicherheiten bei diversen Definitionen geben, diese werden im Laufe der Zeit sicher korrigiert und der Code verbessert werden.

Wem das Abtippen zu mühsam ist, kann den Code auch vie email ([oe3god@gmx.at](mailto:oe3god@gmx.at)) erhalten.

// DCF77 Stationclock and NCDXF Beacon Lister

// by OE3GOD, Oct. 2016

//

// uses:

// Arduino UNO by Conrad.at

// DCF77 Receiver by Conrad.at

// RTC DS1307 by SEMAF Electronics

// LCD I²C Display by SEMAF Electronics

#include "DCF77.h"

#include <Time.h>

#include <TimeLib.h>

#include <LiquidCrystal\_I2C.h>

#include <DS1307RTC.h>

#define DCF\_PIN 2 // Connection pin to DCF 77 device

#define BAND\_PIN 3 // Connection pin to Band Selector

#define DCF\_INTERRUPT 0 // Interrupt number associated with pin

#define BAND\_INTERRUPT 1 // Interrupt number associated with pin

LiquidCrystal\_I2C lcd(0x27, 16, 2);

volatile unsigned long alteZeit=0, entprellZeit=20;

// inverted input on pin DCF\_PIN

DCF77 DCF = DCF77(DCF\_PIN,DCF\_INTERRUPT, false); //bei Betrieb am PIN 4 des DCF77

// DCF77 DCF = DCF77(DCF\_PIN,DCF\_INTERRUPT, true); //bei Betrieb am PIN 3 des DCF77

time\_t prevDisplay = 0;

time\_t time;

// Variables

int reading;

int prevhour=0;

int previous;

byte band, sekm,minm,feld,spalte,i;

// die Tabellen beginnen mit Element "0"

byte matrix[19][18] = {

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{ 1,2,3,4,5,0,0,0,0,0,0,0,0,0,0,0,0,0},

{ 0,1,2,3,4,5,0,0,0,0,0,0,0,0,0,0,0,0},

{ 0,0,1,2,3,4,5,0,0,0,0,0,0,0,0,0,0,0},

{ 0,0,0,1,2,3,4,5,0,0,0,0,0,0,0,0,0,0},

{ 0,0,0,0,1,2,3,4,5,0,0,0,0,0,0,0,0,0},

{ 0,0,0,0,0,1,2,3,4,5,0,0,0,0,0,0,0,0},

{ 0,0,0,0,0,0,1,2,3,4,5,0,0,0,0,0,0,0},

{ 0,0,0,0,0,0,0,1,2,3,4,5,0,0,0,0,0,0},

{ 0,0,0,0,0,0,0,0,1,2,3,4,5,0,0,0,0,0},

{ 0,0,0,0,0,0,0,0,0,1,2,3,4,5,0,0,0,0},

{ 0,0,0,0,0,0,0,0,0,0,1,2,3,4,5,0,0,0},

{ 0,0,0,0,0,0,0,0,0,0,0,1,2,3,4,5,0,0},

{ 0,0,0,0,0,0,0,0,0,0,0,0,1,2,3,4,5,0},

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,3,4,5},

{ 5,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,3,4},

{ 4,5,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,3},

{ 3,4,5,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2},

{ 2,3,4,5,0,0,0,0,0,0,0,0,0,0,0,0,0,1}

};

char\* callsign[19]={

"OE3GOD", " 4U1UN", " VE8AT", " W6WX", " KH6RS", " ZL6B", "VK6RBP", "JA2IGY", " RR9O", " VR2B", " 4S7B", " ZS6DN", " 5Z4B", " 4X6TU", " OH2B", " CS3B", " LU4AA", " OA4B", " YV5B"};

char\* qrg[6] = {"00.000","14.100","18.110","21.150","24.930","28.200"};

void setup() {

pinMode(BAND\_PIN, INPUT); // Pin 2 ist INT0

digitalWrite(BAND\_PIN, HIGH); // interner Pull up Widerstand auf 5V

attachInterrupt(1, Bandwechsel, LOW);

DCF.Start();

lcd.begin();

lcd.backlight();

lcd.setCursor(0,0);

lcd.print(callsign[0]);

lcd.setCursor(7,0);

lcd.print("NCDXF Uhr");

delay(2000);

setSyncProvider(RTC.get);

if (RTC.chipPresent()) {

if (timeStatus()==timeNotSet) {

lcd.setCursor(0,1);

lcd.print("DS1307 not set !");

delay(2500);

}

}

else {

lcd.setCursor(0,1);

lcd.print("no DS1307 device");

delay(2500);

}

setSyncInterval(30);

setSyncProvider(getDCFTime);

lcd.setCursor(0,1);

lcd.print("init DCF77 ");

band = 1;

while(timeStatus()==timeNotSet) {

delay(1000);}

lcd.clear();

}

void loop() {

if(now() != prevDisplay) {

verarbeiten();

if (prevhour != hour()) {

time\_t t = now();

if (t != 0) {

RTC.set(t); // set the RTC and the system time to the received value

prevhour = hour();

}

}

prevDisplay = now();

digitalClockDisplay();

}

}

void digitalClockDisplay(){

// 1

// 0123456789012345

// "hh.mm.ss 28.200"

// "tt.mm fr OE3GOD"

lcd.setCursor(0,0);

printDigits(hour());

lcd.print(":");

printDigits(minute());

lcd.print(":");

printDigits(second());

lcd.setCursor(10,0);

lcd.print(qrg[band]);

lcd.setCursor(0,1);

printDigits(day());

lcd.print(".");

printDigits(month());

lcd.setCursor(6,1);

switch (weekday()) {

case 1:

lcd.print("SO");

break;

case 2:

lcd.print("MO");

break;

case 3:

lcd.print("DI");

break;

case 4:

lcd.print("MI");

break;

case 5:

lcd.print("DO");

break;

case 6:

lcd.print("FR");

break;

case 7:

lcd.print("SA");

break;

default:

lcd.print("??");

break;

}

lcd.setCursor(10,1);

lcd.print(callsign[i]);

}

void printDigits(int digits){

// utility function for digital clock display: prints leading 0

if(digits < 10){

lcd.print('0');

}

lcd.print(digits);

}

void SerialDigits(int digits){

// utility function for digital clock display: prints leading 0

if(digits < 10){

Serial.print('0');

}

Serial.print(digits);

}

unsigned long getDCFTime() {

// time\_t DCFtime = DCF.getTime(); // MEZ bzw MESZ anzeigen

time\_t DCFtime = DCF.getUTCTime(); // UTC anzeigen

if (DCFtime!=0) {

lcd.clear();

}

return DCFtime;

}

void Bandwechsel() {

if((millis() - alteZeit) > entprellZeit) {

// innerhalb der entprellZeit nichts machen

alteZeit = millis(); // letzte Schaltzeit merken

band++;

if (band > 5) {

band = 1;

}

verarbeiten();

}

}

void verarbeiten() {

minm = minute() % 3; // Minute modulo 3

sekm = int(second()/10); Zehnersekunden

feld = (minm\*60)+(sekm\*10);

spalte = (feld/10);

for (i=0;i<19;i++) {

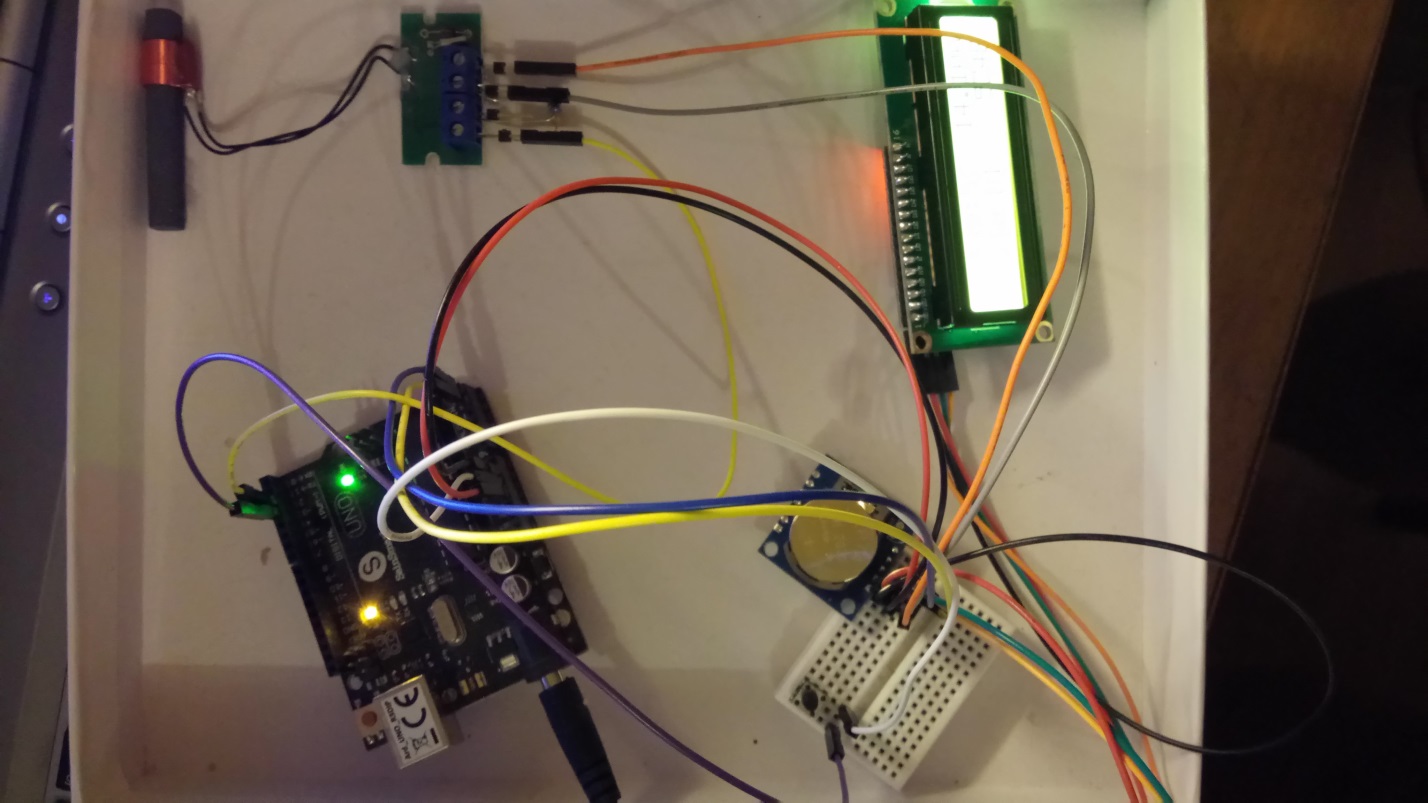
if (matrix[i][spalte] == band) { //Suche, bis das Band übereinstimmt

break;

}

}

}

links oben: Arduino mit externem Stromanschluß

rechts oben: DCF77 Modul

links unten: Breadboard mit Bandwahltaster und RTC DS1307

rechts unten: LCD-Modul

verwendete Bauteile:

SainSmart UNO R3 DIP ATMega328 Conrad Meiselmarkt ( 1359108 – 62) € 18,99

Kabel, Breadboard, Widerstäde, Taster,

LCD-Display (blau/weiß) aus dem Paket

„Arduino Lernpaket Conrad Components

Arduino™ verstehen und anwenden“

Conrad Meiselmarkt (1384145 – 62) € 31,99

DCF Empfänger Modul Conrad Meiselmarkt (641138 – 62) € 13,99

DS1307 Real Time Clock Semaf electronics (ser00171) € 5,95

I²C LCD Modul Semaf electronics (ser00054) € 12,80

Quellen:

<https://www.arduino.cc/>

<http://www.ptb.de/cms/ptb/fachabteilungen/abt4/fb-44/ag-442/verbreitung-der-gesetzlichen-zeit/dcf77/zeitcode.html>

<https://www.ptb.de/cms/fileadmin/internet/fachabteilungen/abteilung_4/4.4_zeit_und_frequenz/4.42/dcf77.pdf>

<https://de.wikipedia.org/wiki/DCF77>

<http://www.bayern-online.com/v2261/artikel.cfm/203/DCF77.html>

<http://www.ncdxf.org/pages/beacons.html>