```c
/* Programm zum Neustarten eines Raspi2B mit Hilfe des Uhrenbausteins RV-3029
 * Die Uhr hat keine Backup Batterie, darum wird sie bei jedem Aufruf
 * des Programms mit der Rechnersystemzeit synchronisiert.
 * Die Startzeiten des Raspis sind fest auf 0 und 30min programmiert.
 * Das Herunterfahren des Rechners wird mit einer Batchdatei, die durch cron aufgerufen wird,
 * nach Ablauf der Datenübertragung von der Wetterstation gemacht.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <linux/i2c-dev.h>

int Oeffne_Port(char Adresse)
{
    //Öffnet den Port mit der Adresse, und gibt den Wert device zurück
    int device;

   //Öffne den I2C Port
    if ((device = open("/dev/i2c-1",O_RDWR)) < 0)
    {
        printf("Fehler beim Oeffnen");
        exit (1);
    }

    if (ioctl(device, I2C_SLAVE, Adresse) < 0)
    {
        printf("Fehler in der Adresse");
        exit (1);
    }

    return(device);
}

void Alarm_loeschen(int device)
{
    //Int FLag Alarm muss manuell zurückgesetzt werden
    char Alarm_Flag[] = {0x02, 0x00};    //Int_Flag AF wird zurückgesetzt

    if(write(device, Alarm_Flag, 2) != 2) printf("Fehler beim Alarm Ruecksetzen");

    return;
}

void Alarm_stellen(int device, char Alarm_daten[])
{
    /*
    Stellt den Alarm ein
    Alarm_daten[0] = Registeradresse
    Alarm_daten[1] = Sekunden
    Alarm_daten[2] = Minuten
    */

    char Alarm_Int[] = {0x01, 0x01};    //Alarm Interrupt setzen

    Alarm_daten[1] = Alarm_daten[1] | (1<<7); //Alarm Sek wird gesetzt Bit7 = 1
    Alarm_daten[2] = Alarm_daten[2] | (1<<7); //Alarm Min wird gesetzt Bit7 = 1

    if(write(device, Alarm_daten, 3) != 3) printf("Fehler beim Alarm Stellen");

    if(write(device, Alarm_Int, 2) != 2) printf("Fehler beim Alarm Interrupt");
    return;
}

void Uhr_stellen(int device)
{
    /*
    Stellt das Datum und die Uhrzeit
    char Datum_Zeit_Hex[8];
```

```
 76
 77     Datum_Zeit_Hex[0] = 0x08;    //Register-Adresse
 78     Datum_Zeit_Hex[1] = 0x00;    //Sekunde
 79     Datum_Zeit_Hex[2] = 0x55;    //Minuten
 80     Datum_Zeit_Hex[3] = (0x09 & ~(1<<6));    //Stunden, Bit 6 = 0 (24 Std Modus)
 81     Datum_Zeit_Hex[4] = 0x31;    //Tag
 82     Datum_Zeit_Hex[5] = 0x02;    //Wochentag
 83     Datum_Zeit_Hex[6] = 0x01;    //Monat
 84     Datum_Zeit_Hex[7] = 0x17;    //Jahr
 85     */
 86
 87     char Zeichenkette[15];
 88     char Datum_Zeit_Hex[8];
 89     int n;
 90     time_t t;
 91     struct tm *zeitinfo;    //Dem Zeiger zeitinfo wird die Struktur tm zugeordnet
 92
 93    // setlocale(LC_TIME, "German");
 94
 95     t = time(NULL);    //Aktuelle Zeit abrufen
 96     zeitinfo = localtime(&t); //In Struktur tm überführen
 97
 98     strftime(Zeichenkette, 15, "%S%M%H%d0%w%m%y", zeitinfo);
 99
100     //Wandelt Zeichenkette in Hexzahl um, zum Stellen der Uhr
101     Datum_Zeit_Hex[0] = 0x08;   //Registeradresse Uhrzeit
102     for(n=0;n<7;n++)
103     {
104         Datum_Zeit_Hex[n + 1] = (Zeichenkette[n * 2] << 4) + (Zeichenkette[(n * 2) + 1] & 0x0F);
105     }
106
107     Datum_Zeit_Hex[3] = Datum_Zeit_Hex[3] & ~(1<<6);  //Stunden Bit = 0 (24Std Modus)
108     /*
109     printf("Uhrzeit-hex: %02x:%02x:%02x\n",Datum_Zeit_Hex[3],Datum_Zeit_Hex[2],Datum_Zeit_Hex[1]);
110     printf("Datum-hex: %02x.%02x.%x\n",Datum_Zeit_Hex[4],Datum_Zeit_Hex[6],Datum_Zeit_Hex[7]);
111     */
112     if(write(device, Datum_Zeit_Hex, 8) != 8) printf("Fehler beim Stellen");
113
114 return;
115 }
116
117 char Uhr_Min_auslesen(int device)
118 {
119     //Hier wird nur der Minutenzähler der Uhr ausgelesen, um die nächste Alarmzeit zu bestimmen.
120     char minute[] = {0x09}; //Register 09H (Minuten)
121
122     if(write(device, minute, 1) != 1) printf("Fehler beim Min Schreiben");
123
124     if(read(device, minute, 1) != 1) printf("Fehler beim Min Lesen");
125
126     return(minute[0]);
127 }
128
129 void Uhr_auslesen(int device)
130 {
131     char daten[7];
132
133     //Liest Datum und Uhrzeit aus und zeigt sie an.
134     daten[0] = 0x08; //Register 08H
135     if(write(device, daten, 1) != 1) printf("Fehler beim Schreiben");
136
137     if(read(device, daten, 7) != 7) printf("Fehler beim Lesen");
138     /*
139     printf("Sek: %x\nMin: %x\nStd: %x\nTag: %x\nWTag: %x\nMon: %x\nJahr: %x\n",\
140     daten[0],daten[1],daten[2],daten[3],daten[4],daten[5],daten[6]);
141
142     printf("Uhrzeit: %02x:%02x:%02x\n",daten[2],daten[1],daten[0]);
143     printf("Datum: %02x.%02x.%x\n",daten[3],daten[5],daten[6]+0x2000);
144     */
145     return;
146 }
147
148
149 int main(void)
150 {
151 //int zaehler = 0;
```

```
152 char Adr = 0x56; //Adresse des i2c Busses mit "i2cdetect -y 1" ermittelt
153 int device;
154 char Alarm_daten[] = {0x10, 0x00, 0x00};
155
156     device = Oeffne_Port(Adr);
157
158     Uhr_stellen(device);
159
160     Alarm_loeschen(device);
161
162     //Wechsel zwischen 0 Min und 30 Minuten Alarm, alle 1/2 Stunde.
163     if (Uhr_Min_auslesen(device) >= 0x30) Alarm_daten[2] = 0x00;
164     else Alarm_daten[2] = 0x30;
165
166     Alarm_stellen(device, Alarm_daten);
167
168     Uhr_auslesen(device);
169
170     close(device);
171
172 return;
173 }
174
```