Titel: **Real Time Clock am Raspberry Pi**

Autor(en): DI Karl HARTINGER
Datum: 18.12.2016 – 21.7.2017

Hinweis

1 Real Time Clock am RasPi

1.1 Aufgabe

Der Raspberry Pi („RasPi“) enthält keine Echtzeituhr (Real Time Clock, RTC) sondern nur eine Software-Uhr. Beim Hinunterfahren wird die aktuelle Uhrzeit abgespeichert und beim Hochfahren wieder geladen. Sobald eine Netzwerkverbindung besteht, wird von einem Zeitserver die aktuelle Uhrzeit geholt und damit die Software-Uhr synchronisiert. Besteht nach einem Hochfahren keine Internet-Verbindung, so ist die Software-Uhr soweit hinten, solange der RasPi abgeschaltet war.

Mit Hilfe eines Uhrenbausteins DS3231 (= RTC) soll die Uhrzeit auch bei abgeschaltetem RasPi weiterlaufen und beim Starten geladen werden. Folgende Funktionen sollen realisiert werden:

- Schreiben der RasPi-Zeit in den RTC (Skript date2rtc).
- Holen der RTC-Zeit und Schreiben in die Software-Uhr des RasPi (Skript rtc2date).
- Anzeige der RTC-Zeit im deutschem Format (C-Programm i2c_ds3231_get.c).
- Automatisches Schreiben der RTC-Zeit in die Software-Uhr beim Systemstart.

1.2 Anschluss eines DS3231-Moduls an den RasPi

Die Ansteuerung des DS3231-Moduls erfolgt über den I²C-Bus. Anschluss direkt am RasPi:

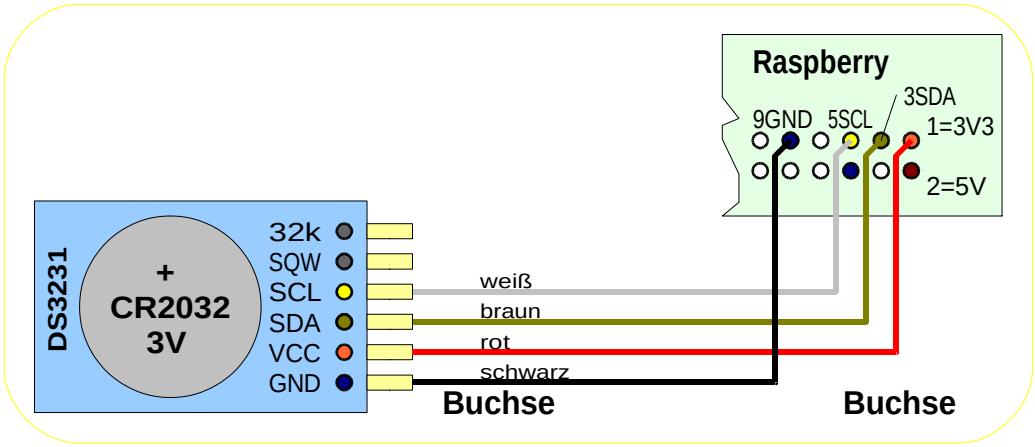

Bild 1: Anschluss des DS3231 direkt am 40poligen RasPi GPIO-Stecker.

Will man mehr als ein Gerät oder ICs mit 5V-Versorgungsspannung am I²C betreiben, so benötigt man ein (Selbstbau-) I2C-Interface. Hier erfolgt der Anschluss folgendermaßen:

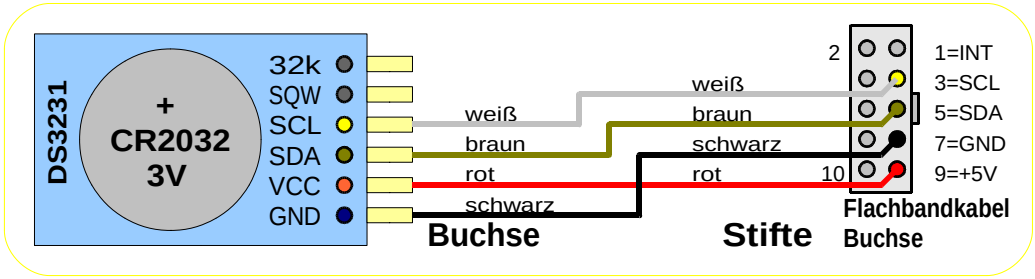

Bild 2: Anschluss des DS3231 am 10poligen (Selbstbau-) I2C-Interface

1.3 DS3231 Hilfsfunktionen

Die Dateien `utils_ds3231.h` bzw. `utils_ds3231.c` enthalten Hilfsfunktionen zur Programmierung einer RTC, wie die Definition von Konstanten, eine einfache Datum-Struktur, Funktionen zur Umwandlung von Binär in BCD und umgekehrt sowie Prüffunktionen für Datum und Uhrzeit und zur Bestimmung des Wochentags zu einem Datum. Sie sollten sich im Verzeichnis befinden, in dem die Dateien zum Lesen und Setzen von Datum und Uhrzeit des DS3231-Moduls kompiliert werden.

```
// utils_ds3231.h, 3.7.2016-19.12.2016, karl.hartinger@gmail.com
#ifndef _UTILS_DS3231_H_
 #define _UTILS_DS3231_H_
 #ifdef  __cplusplus
  extern "C" {
 #endif
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wiringPi.h>

//_____common defines_____________________________________________
#define DEBUG                1
#define I2C_ADDRESS          0x68
#define _OK_                 0
#define ERROR_WIRINGPI       1
#define ERROR_I2C            2
#define NO_ARGS              -1
#define WRONG_DATE           -2
#define WRONG_TIME           -3
#define WRONG_I2C_ADDRESS    -5
#define TOO_MANY_ARGS        -6

//_____date and time as int values________________________________
typedef struct
{
 int yy, mm, dd, nr, hh, ii, ss;
}datetime_t;

//_____convert binary to bcd (binary coded decimal)_______________
// 0 -> 0x00, 10 -> 0x10 (16), 99 -> 0x99 (153)
// Error: return 0xFF
unsigned char bin2bcd(unsigned char bin_);

//_____convert bcd (binary coded decimal) to binary_______________
// 0x00 -> 0x00, 0x10 -> 16, 0x99 -> 99
// Error: return 0xFF
unsigned char bcd2bin(unsigned char bcd_);

//_____Date yyMMdd?_______________________________________________
```

```
int checkdate(char* sdate, datetime_t *datetime);

//_____Time hhmmss?_____________________________________________
int checktime(char* stime, datetime_t *datetime);

//_____Wochentag zu Datum: 1=Sonntag, 2=Montag,...7=Samstag_____
// Rarameter: Tag, Monat, Jahr; Jahr unbedingt vierstellig
// Rueckgabe: 0 = Datum ungueltig z.B. 29.2.2007, 32.12.2007
//           -1 = Jahr kleiner als 1583 oder groesser als 9999
int dayofweek(int d, int m, int y);

//_____Wochentag als Text_______________________________________
// nr_ 1..7 ergibt 1=Sonntag, 2=Montag, ...
// kurz_ungleich 0: Wochentag nur 2 Buchstaben, sonst Langtext
void getweekday(int nr_, char* weekday, int kurz_);

//_____make DateLong YYYY-mm-dd hh:ii:ss_____________________
// Return: 0=OK (string formatted) or 1 on error
int makeDateLong(char* sDateLong, datetime_t datetime);

//_____make date format mmddhhiiyy.ss_____________________
// Return: 0=OK (string formatted) or 1 on error
int makeDateDate(char* sDateLong, datetime_t datetime);

 #ifdef  __cplusplus
 }
 #endif
#endif //_UTILS_LCD_H_
```

Codierung der Hilfs-Funktionen:

```
// utils_ds3231.c, 3.7.2016-30.10.2016, karl.hartinger@gmail.com
#include "utils_ds3231.h"

//_____convert binary to bcd (binary coded decimal)_____________
// 0 -> 0x00, 10 -> 0x10 (16), 99 -> 0x99 (153)
// Error: return 0xFF
unsigned char bin2bcd(unsigned char bin_)
{
 unsigned char z_, e_;
 if((bin_<0)||(bin_>99)) return 0xFF;
 z_ = (int) (bin_/10);
 e_ = bin_ - 10*z_;
 return z_*16 + e_;
}

//_____convert bcd (binary coded decimal) to binary_____________
// 0x00 -> 0x00, 0x10 -> 16, 0x99 -> 99
// Error: return 0xFF
unsigned char bcd2bin(unsigned char bcd_)
{
 unsigned char z_, e_;
 z_= bcd_ >> 4;
 e_= bcd_ & 0x0F;
 if((z_>9)||(e_>9)) return 0xFF;
 return (z_*10 + e_);
}

//_____Date yyMMdd?_____________________________________________
int checkdate(char* sdate, datetime_t *datetime)
{
 int mtag[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
 int y4=0,y,m,d;                  // leapyear?, year, month, day
 char sTemp[3]="00";
 if(strlen(sdate)!=6) return WRONG_DATE;
 //-----check year------------------------------------------
 sTemp[0]=sdate[0];
 sTemp[1]=sdate[1];
 y=atoi(sTemp);
 if ((y%4==0 && y%100!=0) || y%400==0) y4=1; //Schaltjahrcheck
 //-----check month-----------------------------------------
 sTemp[0]=sdate[2];
 sTemp[1]=sdate[3];
```

```
 m=atoi(sTemp);
 if((m<1)||(m>12)) return WRONG_DATE;
 //-----check day-----------------------------------------------
 sTemp[0]=sdate[4];
 sTemp[1]=sdate[5];
 d=atoi(sTemp);
 if ((d<1) || (d>mtag[m]+y4*(m==2))) return WRONG_DATE;
 datetime->yy=y;
 datetime->mm=m;
 datetime->dd=d;
 return _OK_;
}

//_____Time hhmmss?_____________________________________________
int checktime(char* stime, datetime_t *datetime)
{
 int h,m,s;
 char sTemp[3]="00";
 //-----check hour----------------------------------------------
 sTemp[0]=stime[0];
 sTemp[1]=stime[1];
 h=atoi(sTemp);
 if((h<0)||(h>23)) return WRONG_TIME;
 //-----check minute--------------------------------------------
 sTemp[0]=stime[2];
 sTemp[1]=stime[3];
 m=atoi(sTemp);
 if((m<0)||(m>59)) return WRONG_TIME;
 //-----check second--------------------------------------------
 sTemp[0]=stime[4];
 sTemp[1]=stime[5];
 s=atoi(sTemp);
 if((s<0)||(s>59)) return WRONG_TIME;
 datetime->hh=h;
 datetime->ii=m;
 datetime->ss=s;
 return _OK_;
}

//_____Wochentag zu Datum: 1=Sonntag, 2=Montag,...7=Samstag_____
// Rarameter: Tag, Monat, Jahr; Jahr unbedingt vierstellig
// Rueckgabe: 0 = Datum ungueltig z.B. 29.2.2007, 32.12.2007
//           -1 = Jahr kleiner als 1583 oder groesser als 9999
int dayofweek(int d, int m, int y)
{
 int s=0;                                    //Tage pro Monat:
 int mtag[13]={0,31,28,31,30,31,30,31,31,30,31,30,31};
 if ((y%4==0 && y%100!=0) || y%400==0) s=1; //Schaltjahrcheck
 if (y<1583 || y>9999) return(-1);          //Jahr gueltig?
 if (m<1 || m>12) return(0);                //Monat gueltig?
 if (d<1 || d>mtag[m]+s*(m==2)) return(0);  //Tag gueltig?
 if (m < 3) { m += 13; y--; } else  m++;    //Jahresanfang Maerz
 s = d + 26*m/10 + y+y/4-y/100+y/400 + 6;   //Berechnung
 return(s % 7 + 1);                         //Ergebnis anpassen
}

//_____Wochentag als Text_______________________________________
void getweekday(int nr_, char* weekday, int kurz_)
{
 switch(nr_)
 {
  case 1: if(kurz_) strcpy(weekday, "SO");
          else strcpy(weekday, "Sonntag");
          break;
  case 2: if(kurz_) strcpy(weekday, "MO");
          else strcpy(weekday, "Montag");
          break;
  case 3: if(kurz_) strcpy(weekday, "DI");
          else strcpy(weekday, "Dienstag");
          break;
  case 4: if(kurz_) strcpy(weekday, "MI");
          else strcpy(weekday, "Mittwoch");
          break;
  case 5: if(kurz_) strcpy(weekday, "DO");
```

```
          else strcpy(weekday, "Donnerstag");
          break;
  case 6: if(kurz_) strcpy(weekday, "FR");
          else strcpy(weekday, "Freitag");
          break;
  case 7: if(kurz_) strcpy(weekday, "SA");
          else strcpy(weekday, "Samstag");
          break;
  default: if(kurz_) strcpy(weekday, "??");
           else strcpy(weekday, "Fehler!");
  strcpy(weekday, "FEHLER!");
 }
}

//_____make DateLong YYYY-mm-dd hh:ii:ss_____________________________
// Return: 0=OK (string formatted) or 1 on error
int makeDateLong(char* sDateLong, datetime_t datetime)
{
 if(strlen(sDateLong)<19) return 1;
 sprintf(sDateLong, "%4d-%02d-%02d %02d:%02d:%02d",
  2000+datetime.yy, datetime.mm, datetime.dd,
  datetime.hh, datetime.ii, datetime.ss);
 return 0;
}

//_____make date format mmddhhiiyy.ss_________________________________
int makeDateDate(char* sDateLong, datetime_t datetime)
// Return: 0=OK (string formatted) or 1 on error
{
 if(strlen(sDateLong)<19) return 1;
 sprintf(sDateLong, "%02d%02d%02d%02d%02d.%02d",
  datetime.mm, datetime.dd, datetime.hh,
  datetime.ii, datetime.yy, datetime.ss);
 return 0;
}
```

1.4 Lesen von Datum und Uhrzeit aus dem DS3231-Modul

Das folgende C-Programm `i2c_ds3231_get` ermöglicht das Lesen von Datum und Uhrzeit aus dem DS3231-Modul. Der erforderliche Parameter 1 oder 2 gibt das Format an, wie das Datum dargestellt werden soll:

1 ergibt `YYYY-mm-dd hh:ii:ss`
2 ergibt `mmddhhiiyy.ss` (for linux date -u)

(1) Ins Arbeitsverzeichnis wechseln, zB

```
cd /home/pi/src_c/ds3231
```

(2) Texteditor nano starten

```
nano i2c_ds3231_get.c
```

(3) Codierung des Hauptprogramms eingeben:

```
// i2c_ds3231_get.c, 6.7.2016-1.11.2016, Karl Hartinger
// gcc i2c_ds3231_get.c utils_ds3231.c -o i2c_ds3231_get -l wiringPi
#include "utils_ds3231.h"

//_____MAIN FUNCTION_____________________________________________
int main(int argc, char **argv)
{
 int fh_;
 char i2c_address=I2C_ADDRESS;
 char sDateLong_[]="2000-01-01 00:00:00";
 datetime_t datetime_;
 int  ret=NO_ARGS;               // return value
 int  dateformat=0;
 //-----check argument(s)-------------------------------------
 if(argc>1)
 {
  if(argv[1][0]=='1') { dateformat=1; ret=_OK_; }
  if(argv[1][0]=='2') { dateformat=2; ret=_OK_; }
 }
 //-----if params not ok: help text---------------------------
 if(ret!=_OK_)
 {
  printf("Raspberry Pi: Get date and time from RTC DS3231\n");
  printf("Author:   Karl Hartinger, 3.7.2016-30.10.2016\n");
  printf("Usage:    i2c_ds3231_get [-h]\n");
  printf("Format:   1 = YYYY-mm-dd hh:ii:ss\n");
  printf("          2 = mmddhhiiyy.ss (for linux date -u)\n");
  printf("Example:  sudo date +\"%%F %%X\" --set \"$(sudo ./i2c_ds3231_get
1)\"\n");
  return(ret);
 }
 //-----setup wiringPi----------------------------------------
 if (wiringPiSetup () == -1)
 {
  if(DEBUG) printf("ERROR: wiringPiSetup failed\n");
  return ERROR_WIRINGPI;
 }
 if ((fh_=wiringPiI2CSetup(i2c_address)) == -1)
 {
  if(DEBUG) printf("ERROR: wiringPiI2CSetup(%02X) failed\n",I2C_ADDRESS);
  return ERROR_I2C;
 }
 //-----read from RTC-----------------------------------------
 datetime_.yy=bcd2bin(wiringPiI2CReadReg8(fh_, 6));
 datetime_.mm=bcd2bin(wiringPiI2CReadReg8(fh_, 5));
 datetime_.dd=bcd2bin(wiringPiI2CReadReg8(fh_, 4));
```

```
 datetime_.nr=bcd2bin(wiringPiI2CReadReg8(fh_, 3));
 datetime_.hh=bcd2bin(wiringPiI2CReadReg8(fh_, 2));
 datetime_.ii=bcd2bin(wiringPiI2CReadReg8(fh_, 1));
 datetime_.ss=bcd2bin(wiringPiI2CReadReg8(fh_, 0));
 //if(DEBUG) printf("Zahlenwerte: %d.%d.%d %02d:%02d:%02d\n",
datetime_.dd,datetime_.mm,datetime_.yy,datetime_.hh,datetime_.ii,datetime_.ss);
 if(dateformat==2)
  makeDateDate(sDateLong_, datetime_);
 else
  makeDateLong(sDateLong_, datetime_);
 printf("%s\n",sDateLong_);
 return _OK_;
}
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

(4) Programm kompilieren

```
gcc i2c_ds3231_get.c utils_ds3231.c -o i2c_ds3231_get -l wiringPi
```

(5) Programm testen

```
sudo ./i2c_ds3231_get 1
2000-01-01 00:01:07
```

Entweder ist im Modul bereits ein aktuelles Datum+Uhrzeit eingestellt oder es wird die Zeit seit dem Start des Moduls angezeigt.

Sollte eine Fehlermeldung „Unable to determine...“ erscheinen, so muss die wiringPi-Version upgedatet werden (siehe Anhang).

Programm für alle User verfügbar machen (ohne sudo)

```
sudo cp ./i2c_ds3231_get /usr/local/bin
sudo chown root /usr/local/bin/i2c_ds3231_get
sudo chmod 777 /usr/local/bin/i2c_ds3231_get
sudo chmod u+s /usr/local/bin/i2c_ds3231_get
```

Jetzt kann das Programm aus einem beliebigen Verzeichnis heraus von einem beliebigen User ausgeführt werden (ohne sudo und ohne Punkt, aber mit Parameter):

```
cd ..
i2c_ds3231_get 1
```

1.5 Schreiben von Datum und Uhrzeit in den DS3231-Modul

Das folgende C-Programm ermöglicht das Schreiben von Datum und Uhrzeit in das DS3231-Modul durch Aufruf von

```
i2c_ds3231_set yymmdd hhiiss
```

mit yy..year (00..99), mm..month (01..12), dd..day (01..31), hh..hour (00..23), ii...minute (00..59), ss...seconds (00..59).

(1) Ins Arbeitsverzeichnis wechseln, zB

```
cd /home/pi/src_c/ds3231
```

(2) Texteditor nano starten

```
nano i2c_ds3231_set.c
```

(3) Codierung des Hauptprogrammes eingeben:

```
// i2c_ds3231_set.c, 3.7.2016-1.11.2016, Karl Hartinger
// gcc i2c_ds3231_set.c utils_ds3231.c -o i2c_ds3231_set -l wiringPi
#include "utils_ds3231.h"

//_____MAIN FUNCTION_____________________________________________
int main(int argc, char **argv)
{
 int fh_;
 char i2c_address=I2C_ADDRESS;
 char sdate_[10];
 char stime_[10];
 char sday_[11];
 datetime_t datetime_;
 int  ret=NO_ARGS;               // return value
 char *pdata;                     // pointer to arg data
 int nr_;
 int temp_;
 //-----check argument(s)--------------------------------------
 if(argc>1)
 {
  //.....get date.............................................
  strncpy(sdate_,argv[1],7);
  ret=checkdate(sdate_, &datetime_);
  if((ret==_OK_)&&(argc>2))
  {
   strncpy(stime_,argv[2],7);
   ret=checktime(stime_, &datetime_);
  }
  if(argc>3) ret=TOO_MANY_ARGS;
 }
 //-----if params not ok: help text----------------------------
 if(ret!=_OK_)
 {
  printf("Raspberry Pi: Set date and time on RTC DS3231\n");
  printf("Author:   Karl Hartinger, 3.7.2016-1.11.2016\n");
  printf("Usage:    i2c_ds3231_set yymmdd hhiiss\n");
  printf("          yy..year (00..99), mm..month   (01..12), dd..day
(01..31),\n");
  printf("          hh..hour (00..23), ii...minute (00..59), ss...seconds\n");
  printf("Example:  i2c_ds3231_set $(date +\"%%y%%m%%d %%H%%M%%S\")\n");
  return(ret);
 }
 //-----setup wiringPi-----------------------------------------
 if (wiringPiSetup () == -1)
 {
  if(DEBUG) printf("ERROR: wiringPiSetup failed\n");
```

```
  return ERROR_WIRINGPI;
 }
 if ((fh_=wiringPiI2CSetup(i2c_address)) == -1)
 {
  if(DEBUG) printf("ERROR: wiringPiI2CSetup(%02X) failed\n",I2C_ADDRESS);
  return ERROR_I2C;
 }
 //-----get number day of week---------------------------------
 nr_=dayofweek(datetime_.dd, datetime_.mm,2000+datetime_.yy);
 datetime_.nr=nr_;
 getweekday(nr_, sday_, 0);
 if(DEBUG) printf("Datum  : %02d.%02d.%02d\n",datetime_.dd,
datetime_.mm,datetime_.yy);
 if(DEBUG) printf("Uhrzeit: %02d.%02d.%02d\n",datetime_.hh,
datetime_.ii,datetime_.ss);
 if(DEBUG) printf("Wochentag %d => %s\n", nr_, sday_);
 //-----write to RTC-------------------------------------------
 wiringPiI2CWriteReg8(fh_, 6, bin2bcd(datetime_.yy));
 wiringPiI2CWriteReg8(fh_, 5, bin2bcd(datetime_.mm));
 wiringPiI2CWriteReg8(fh_, 4, bin2bcd(datetime_.dd));
 wiringPiI2CWriteReg8(fh_, 3, bin2bcd(datetime_.nr));
 wiringPiI2CWriteReg8(fh_, 2, bin2bcd(datetime_.hh));
 wiringPiI2CWriteReg8(fh_, 1, bin2bcd(datetime_.ii));
 wiringPiI2CWriteReg8(fh_, 0, bin2bcd(datetime_.ss));
 return _OK_;
}
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

(4) Programm kompilieren

```
gcc i2c_ds3231_set.c utils_ds3231.c -o i2c_ds3231_set -l wiringPi
```

(5) Programm testen

```
sudo ./i2c_ds3231_set 161130 195500
Datum  : 30.11.16
Uhrzeit: 19.55.00
Wochentag 4 => Mittwoch
```

oder Datum+Uhrzeit von der Software-Uhr in die RTC übertragen:

```
sudo ./i2c_ds3231_set $(date +"%y%m%d %H%M%S")
Datum  : 19.12.16
Uhrzeit: 15.43.36
Wochentag 2 => Montag
```

Kontrolle: Abfrage von Datum+Uhrzeit aus der RTC:

```
sudo ./i2c_ds3231_get 1
2016-12-19 15:44:03
```

Programm für alle User verfügbar machen (ohne sudo)

```
sudo cp ./i2c_ds3231_set /usr/local/bin
sudo chown root /usr/local/bin/i2c_ds3231_set
sudo chmod 777 /usr/local/bin/i2c_ds3231_set
sudo chmod u+s /usr/local/bin/i2c_ds3231_set
```

1.6 Script zum Lesen der RTC-Zeit und Schreiben nach date

Mit Hilfe des Skripts `date2rtc` kann das Datum und die Uhrzeit aus der Echtzeituhr (RTC) gelesen und in die Software-Uhr des RasPis übertragen werden (Linux-Befehl date).

(1) Texteditor nano starten

```
sudo nano /usr/local/bin/rtc2date
```

(2) Inhalt der Skript-Datei:

```
sudo date +"%F %X" --set "$(i2c_ds3231_get 1)"
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x
Anmerkung: Die Datei `i2c_ds3231_get` muss sich im Verzeichnis `/usr/local/bin` befinden und für alle User zur Ausführung freigegeben sein.

(3) Skript für alle User verfügbar machen

```
sudo chown root /usr/local/bin/rtc2date
sudo chmod 777 /usr/local/bin/rtc2date
sudo chmod u+s /usr/local/bin/rtc2date
```

(4) Skript testen (zum Testen wird zuerst die Zeit in der Software-Uhr verstellt...)

```
sudo date +"%F %X" --set "000101 0000"
2000-01-01 00:00:00
date
Sam Jän  1 00:00:03 CET 2000
rtc2date
2016-12-19 17:20:17
date
Mon Dez 19 17:20:23 CET 2016
```

1.7 Schreiben der RasPi-Zeit in die RTC (Skript date2rtc)

(1) Texteditor nano starten

```
sudo nano /usr/local/bin/date2rtc
```

(2) Inhalt der Skript-Datei:

```
i2c_ds3231_set $(date +"%y%m%d %H%M%S")
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x
Anmerkung: Die Datei `i2c_ds3231_set` muss sich im Verzeichnis `/usr/local/bin` befinden und für alle User zur Ausführung freigegeben sein.

(3) Skript für alle User verfügbar machen

```
sudo chown root /usr/local/bin/date2rtc
sudo chmod 777 /usr/local/bin/date2rtc
sudo chmod u+s /usr/local/bin/date2rtc
```

(4) Skript testen (zum Testen wird zuerst die Zeit in der RTC verstellt...)

```
i2c_ds3231_set 000101 000000
Datum  : 01.01.00
Uhrzeit: 00.00.00
Wochentag 7 => Samstag
date2rtc
Datum  : 19.12.16
Uhrzeit: 17.09.08
Wochentag 2 => Montag
i2c_ds3231_get 1
2016-12-19 17:09:30
```

1.8 Automatisches Schreiben der RTC-Zeit beim Systemstart

Soll die RTC-Zeit beim Systemstart automatisch in die Software-Uhr des RasPi geschrieben werden, so geschieht dies am Besten durch ein Skript, das beim Systemstart ausgeführt wird. Wenn danach eine Internet-Verbindung besteht, wird die Zeit ohnedies mit dem Zeitserver synchronisiert.

(1) Texteditor nano starten

```
sudo nano /etc/rc.local
```

(2) Eigene Autostart-Datei einhängen (am Schluss des Skripts ergänzen!):

```
#-----Aufruf eines Scripts mit eigenen Befehlen-----
/usr/local/bin/autostart.sh
exit 0
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

(3) Texteditor nano starten

```
sudo nano /usr/local/bin/autostart.sh
```

(4) Inhalt der Skript-Datei:

```
#!/bin/bash
#...Farbe der Schrift auf gelb ändern...
echo -e "\\033[01;33m"
printf "_____autostart.sh____19.12.2016_____Karl Hartinger_____\n"
printf "RTC-Zeit nach date kopieren (rtc2date)\n"
/usr/local/bin/rtc2date &
printf "_____________________________________________________\n"
#...Farbe der Schrift wieder auf weiß ändern...
echo -e "\\033[00m"
exit 0
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Test: Software-Uhrzeit verstellen, RasPi herunterfahren, etwas warten, die Netzwerkverbindung unterbrechen und das RasPi neu starten:

```
i2c_ds3231_get 1
2016-12-19 19:07:46
sudo date +"%F %X" --set "000101 0000"
2000-01-01 00:00:00
date
Sam Jän  1 00:00:03 CET 2000
sudo shutdown -h
```

Kontrolle der Uhrzeit nach dem Neustart mittels Tastatur und Bildschirm ;)

```
date
Mon Dez 19 19:09:05 CET 2016
```

Sollte der 1. Jänner angezeigt werden, dann ist entweder keine RTC angesteckt, die I²C-Verbindung fehlerhaft oder die Batterie der RTC leer. Diesen Fall kann man dadurch testen, dass man die RTC kurz absteckt und mit `i2c_ds3231_get 1` die Zeit abfragt.

2 Anhang

2.1 Update wiringPi-Version

Quelle: http://wiringpi.com/download-and-install/

(1) Check der wiringPi-Version

```
$ gpio -v
gpio version: 2.32
Copyright (c) 2012-2015 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Unable to determine hardware version. I see: Hardware   : BCM2835
,
 - expecting BCM2708 or BCM2709.
...
```

Sollte diese Fehlermeldung erscheinen, so ist ein Update erforderlich ;)

(2) wiringPi entfernen, RasPi-Software updaten, GIT installieren

```
$ sudo apt-get purge wiringpi
$ hash -r
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install git-core
```

(3) wiringPi mit GIT zum ersten Mal herunterladen

```
$ cd ~
$ git clone git://git.drogon.net/wiringPi
```

Erhält man eine Fehlermeldung („fatal: destination path 'wiringPi' already exists and is not an empty directory.“), so wurde wiringPi bereits einmal heruntergeladen und in diesem Fall gilt

```
$ cd ~/wiringPi
$ git pull origin
```

(4) wiringPi bilden

```
$ cd ~/wiringPi
$ ./build
```

(5) wiringPi testen

```
$ gpio -v
$ gpio readall
```