```c
/*Das Programm dient zur Steuerung einer Kellerlüftung.
 * Der Lüfter wird bei einer Differenz der Taupunkttemperatur von >=5grd
 * zwischen Aussenluft und Kellerluft eingeschaltet. Der Taupunkt der Aussenluft
 * muss niedriger sein, als der Taupunkt im Keller, da es ansonsten im Keller zu einer höheren
 * Luftfeuchtigkeit kommt.
 * Der Lüfter sollte mindestens einmal pro Stunde die Kellerluft austauschen.
 *
 *
 * Prozessor: ATMEGA 328 e-Fuse 01h; h-Fuse DFh; l-Fuse E7h; lock 3Fh
 * Version 0.0 3/19 pkr
 * Version 0.1 11/19 Hysterese geändert und Lüfterabfrage Fehler beseitigt.
 * Sensor HTU21D hinzugefügt. Muss vor dem Compilieren definiert werden.
 * Tabelle auf 0,5grd erweitern? und 16bit?*/


#define F_CPU 4000000UL
//#define HDC
#define HTU

#include <stdlib.h>
#include <string.h>
#include <avr/io.h>
#include <avr/wdt.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#include <avr/eeprom.h>
#include <util/delay.h>
#include <avr/pgmspace.h>
#include "bibliothek/taupunkttabelle.c"
#include "bibliothek/e-paper-2,9.c"
#include "bibliothek/i2c.h"
#include "bibliothek/i2c1.h"    //für zweiten Sensor, weil gleiche Geräteadresse


#define Sec_Faktor 39062    //15625 bei 4,0...MHz 1s (Vorteiler 256); 31250 2s; 45795 3s
                            //39062 10s bei Vorteiler 1024
#define Timer1_start (TIMSK1 |= (1<<TOIE1))
#define Timer1_halt (TIMSK1 &= ~(1<<TOIE1))
#define Zeit2 15    //1ms bei Vorteiler:256 (15); für Timer 2
#define Timer2_start (TIMSK2 |= (1<<OCIE2A))
#define Timer2_halt (TIMSK2 &= ~(1<<OCIE2A))

#define Adresse_HDC1080 0x80    //Schreiben; 0x81: Lesen
#define Adresse_HTU21D  0x80
#define Adresse_write_Register 0xE6
#define Adresse_read_Register 0xE7
#define Adresse_Reset 0xFE
#define Adresse_Temperatur_no_hold 0xF3
#define Adresse_Feuchte_no_hold 0xF5

#define Intervall_werk 60    //Intervall in Minuten nach der der Lüfter für Dauer eingeschaltet wird
#define Dauer_werk 20    //Zeit in Minuten die der Lüfter läuft
#define T_Punkt_Diff_werk 5  //Taupunktdifferenz
#define Temp_Min_Keller_werk 8   //Minimum der Kellertemperatur
#define Hysterese 2 //Hysterese verhindert ein schnelles Aus- und Einschalten des Lüfters

#define Geraeteport PORTB
#define Impuls_A 1  //Drehgeber
#define Impuls_B 0  //Drehgeber
#define Bestaetigung 5  //Taste Drehgeber
#define Messung 2
#define Luefter 4
#define Taster 3    //für Menüauswahl

#define Luefter_aus (Geraeteport &= ~(1<<Luefter))
#define Luefter_an (Geraeteport |= (1<<Luefter))
#define Luefter_abfragen (PINB & (1<<Luefter))
#define Messung_aus (Geraeteport &= ~(1<<Messung))
#define Messung_an (Geraeteport |= (1<<Messung))
#define Taster_ein !(PINB & (1<<Taster))
#define A_Imp (PINB & (1<<Impuls_A))
#define B_Imp (PINB & (1<<Impuls_B))
#define Bestaetigung_ja !(PINB & (1<<Bestaetigung))

//UP
```

```
 76 void Begruessungsbild();
 77 uint8_t Bestaetigungstaste();
 78 void Display_ini();
 79 uint8_t Menueauswahl();
 80 void Hauptanzeige();
 81 void Auswahl_Einstellungen(uint8_t Auswahl);
 82 void Einstellmenue();
 83 void Temperatur_Feuchte_auslesen_HTU();
 84 void Temperatur_Feuchte_auslesen(uint8_t Sensor);
 85 int8_t Taupunkt_ermitteln(int16_t Temperatur, uint8_t Feuchte);
 86 void Luefter_ein_aus();
 87 uint8_t Bedingung_Luefter_ein();
 88 void Anfangswerte();
 89 void Lese_Werte_aus_dem_EEPROM();
 90
 91 //RAM
 92 volatile uint8_t  Merker_Temperatur_auslesen = 0;
 93 volatile uint8_t Merker_Anzeige_aktualisieren = 0;
 94 uint8_t Anzeigewahl;    //Menüwahl 0 oder 1; Hauptmenü, Einstellungen
 95 uint8_t Wert = 1;   //Zähler zum Einstellen der Werte im Menü Einstellungen
 96 volatile uint8_t Takt = 0;   //wird alle 10s um 10 erhöht (im Interrupt 1)
 97 volatile uint8_t Min = 0;    //Minutenzähler im Interrupt
 98 char Platz[] = {"123456"};
 99 char Leerzeichen[] = {"      "};    //5 Leerzeichen
100 int16_t Temperatur_aussen;
101 uint8_t Feuchte_aussen;
102 int16_t Temperatur_Keller;
103 uint8_t Feuchte_Keller;
104 int8_t T_Punkt_A;   //Taupunkt aussen
105 int8_t T_Punkt_K;   //Taupunkt Keller
106 uint8_t Dauer;  //Einschaltdauer Lüfter
107 uint8_t Intervall;  //Periodendauer für das Einschalten des Lüfters
108 uint8_t T_Punkt_Diff;    //Taupunktdifferenz
109 uint8_t Temp_Min_Keller;    //Minimale Kellertemperatur
110
111 //Speicher EEPROM
112
113     //Vorgegebene Einstellungen
114 uint8_t ee_Intervall_werk  EEMEM = Intervall_werk;
115 uint8_t ee_Dauer_werk EEMEM = Dauer_werk;    //Zeit in Minuten die der Lüfter läuft
116 uint8_t ee_T_Punkt_Diff_werk EEMEM = T_Punkt_Diff_werk;  //Taupunktdifferenz
117 uint8_t ee_Temp_Min_Keller_werk EEMEM = Temp_Min_Keller_werk;   //Minimum der Kellertemperatur
118     //Für eigene Einstellungen
119 uint8_t ee_Intervall EEMEM = 60;    //Intervall in Minuten nach der der Lüfter für Dauer
    eingeschaltet wird
120 uint8_t ee_Dauer EEMEM = 20;    //Zeit in Minuten die der Lüfter läuft
121 uint8_t ee_T_Punkt_Diff EEMEM = 5;  //Taupunktdifferenz
122 uint8_t ee_Temp_Min_Keller EEMEM = 8;   //Minimum der Kellertemperatur
123 uint8_t ee_Version[] EEMEM = {"Version 0.1 11/19 pkr"};
124
125
126 void Begruessungsbild()
127 {
128     char Ueberschrift[] = {"KroTro"};
129     char Beschreibung1[] = {"Lueftersteuerung"};
130     char Beschreibung2[] = {"fuer einen Kellerraum"};
131     char Version[] = {"Version 0.1 pkr 11/19"};
132
133     Schreibe_Text(Ueberschrift, sans_mono_24,  4, 200);
134     Schreibe_Text(Beschreibung1, sans_mono_12,  7, 280);
135     Schreibe_Text(Beschreibung2, sans_mono_12,  9, 280);
136     Schreibe_Text(Version, sans_mono_12,  12, 280);
137
138     Display_Fram(); //Anzeige aktualisieren
139 }
140
141 uint8_t Bestaetigungstaste()
142 {
143
144     if (Bestaetigung_ja)
145         _delay_ms(5);
146     if (Bestaetigung_ja)
147     {
148         while (Bestaetigung_ja);
149         _delay_ms(5);
```

```
150         return 1;
151     }
152     else
153     {
154         return 0;
155     }
156 }
157
158 void Display_ini()
159 {
160     //Der Anfangstext für den Bildschirmhintergrund muss zweimal geschrieben werden,
161     //da ansonsten die beiden Speicher in der Anzeige mit undefinierten Zeichen
162     //beschrieben werden. Wenn der ganze Bildschirm beschrieben wird, eine Pause von 2s
163     //nach dem Schreiben einhalten.
164
165     Schreibe_Bildschirm_voll(weiss);  //0xFF
166     _delay_ms(2000);
167
168     Schreibe_Bildschirm_voll(weiss);  //0xFF
169    _delay_ms(2000);
170
171 }
172
173 uint8_t Menueauswahl()
174 {
175    static uint8_t Menuezaehler = 0;
176
177         if(Taster_ein) //Tastenzähler
178         {
179             _delay_ms(5);   //Entprellung
180             while(Taster_ein);  //warte hier bis Taste losgelassen
181             Display_ini();  //muß gemacht werden, weil Anzeige komplett neu ist
182             Menuezaehler +=1;
183             if (Menuezaehler >= 2) Menuezaehler = 0;  //Zähler rücksetzen
184
185             if(Menuezaehler == 1)   //Einstellungen
186             {
187                 Timer1_halt;
188                 Timer2_start;   //für Drehgeberabfrage
189             }
190             else    //Hauptmenü
191             {
192                 Timer2_halt;
193                 Timer1_start;
194                 Takt = 0;
195                 Min = 0;
196             }
197         }
198     return Menuezaehler;
199 }
200
201
202 void Hauptanzeige()
203 {
204     //X_Wert Oberkante Text(0: oben; 15: unten), Y_Wert Anfang Text(0: rechts; 296: links)
205     #define Anfang_Ausgabe_Aussen 230   //Textanfang Variable für Aussen
206     #define Anfang_Ausgabe_Keller 100   //Textanfang Variable für den Keller
207     int16_t Temp_ganz;
208     uint8_t Temp_nachk;
209     char Temp_String[] = {"-11,5 "};
210     char Feuchte_String[] = {"99%"};
211     char grdC[] = {128,0};  //Zeichen°C
212
213     Schreibe_Text("Aussen", sans_mono_12, 1, 220);
214     Schreibe_Text("Keller", sans_mono_12, 1, 90);
215     Schreibe_Text("T:", sans_mono_24, 6, 290);
216     Schreibe_Text("F:", sans_mono_24, 10, 290);
217     Schreibe_Text("TP:", sans_mono_24, 15, 290);
218
219     //Aussentemperatur ins Display schreiben
220     Temp_ganz = Temperatur_aussen/10;  //Ganzahliger Temperatuwert
221     Temp_nachk = (uint8_t)(Temperatur_aussen - (Temp_ganz * 10)); //Nachkommawert
222
223     Schreibe_Text(Leerzeichen, sans_mono_24, 6, Anfang_Ausgabe_Aussen);   //Anzeige löschen
224     itoa(Temp_ganz, Platz, 10);
```

```
225     strcpy(Temp_String, Platz);
226     strcat(Temp_String, ",");
227     itoa(Temp_nachk, Platz, 10);
228     strcat(Temp_String, Platz );
229     strcat(Temp_String, grdC);
230     Schreibe_Text(Temp_String, sans_mono_24, 6, Anfang_Ausgabe_Aussen);
231
232     //Feuchte aussen ins Display schreiben
233     utoa(Feuchte_aussen, Platz, 10);
234     strcpy(Feuchte_String, Platz);
235     strcat(Feuchte_String, "%");
236     Schreibe_Text(Feuchte_String, sans_mono_24, 10, Anfang_Ausgabe_Aussen);
237
238     //Taupunkt aussen ins Display schreiben
239     Schreibe_Text(Leerzeichen, sans_mono_24, 15, Anfang_Ausgabe_Aussen);  //Anzeige löschen
240
241     if (T_Punkt_A == 100)  //außerhalb des Tabellenbereiches
242     {
243         Schreibe_Text("a.T.", sans_mono_24, 15, Anfang_Ausgabe_Aussen);
244     }
245     else
246     {
247         itoa(T_Punkt_A, Platz, 10);
248         strcat(Platz, grdC);
249         Schreibe_Text(Platz, sans_mono_24, 15, Anfang_Ausgabe_Aussen);    //Taupunkt aussen
250     }
251
252      //Kellertemperatur ins Display schreiben
253     Temp_ganz = Temperatur_Keller/10;  //Ganzahliger Temperatuwert
254     Temp_nachk = (uint8_t)(Temperatur_Keller - (Temp_ganz * 10)); //Nachkommawert
255
256     Schreibe_Text(Leerzeichen, sans_mono_24,  6, Anfang_Ausgabe_Keller);   //Anzeige löschen
257     itoa(Temp_ganz, Platz, 10);
258     strcpy(Temp_String, Platz);
259     strcat(Temp_String, ",");
260     itoa(Temp_nachk, Platz, 10);
261     strcat(Temp_String, Platz );
262     strcat(Temp_String, grdC);
263     Schreibe_Text(Temp_String, sans_mono_24, 6, Anfang_Ausgabe_Keller);
264
265     //Feuchte Keller ins Display schreiben
266     utoa(Feuchte_Keller, Platz, 10);
267     strcpy(Feuchte_String, Platz);
268     strcat(Feuchte_String, "%");
269     Schreibe_Text(Feuchte_String, sans_mono_24, 10, Anfang_Ausgabe_Keller);
270
271     //Taupunkt Keller ins Display schreiben
272     Schreibe_Text(Leerzeichen, sans_mono_24, 15, Anfang_Ausgabe_Keller);  //Anzeige löschen
273
274     if (T_Punkt_K == 100)  //außerhalb des Tabellenbereiches
275     {
276         Schreibe_Text("a.T.", sans_mono_24, 15, Anfang_Ausgabe_Keller);
277     }
278     else
279     {
280         itoa(T_Punkt_K, Platz, 10);
281         strcat(Platz, grdC);
282         Schreibe_Text(Platz, sans_mono_24, 15, Anfang_Ausgabe_Keller);    //Taupunkt Keller
283     }
284
285     //Minutenanzeige zum Test
286     utoa(Min, Platz, 10);
287     Schreibe_Text(Platz, sans_mono_12, 15, 23);
288
289     Display_Fram();
290 }
291
292 void Auswahl_Einstellungen(uint8_t Auswahl)
293 //Hier wird die Auswahl der einzustellenden Größe vorgenommen
294 {
295
296     #define Anfang_Text_Variable  30
297
298     switch (Auswahl)
299     {
```

```
300         case 1:
301              Schreibe_Buchstabe('*', sans_mono_12, 6, Anfang_Text_Variable + 10);
302              Schreibe_Buchstabe(' ', sans_mono_12, 8, Anfang_Text_Variable + 10);
303              Schreibe_Buchstabe(' ', sans_mono_12, 10, Anfang_Text_Variable + 10);
304              Schreibe_Buchstabe(' ', sans_mono_12, 12, Anfang_Text_Variable + 10);
305              Schreibe_Buchstabe(' ', sans_mono_12, 14, Anfang_Text_Variable + 10);
306              break;
307         case 2:
308              Schreibe_Buchstabe(' ', sans_mono_12, 6, Anfang_Text_Variable + 10);
309              Schreibe_Buchstabe('*', sans_mono_12, 8, Anfang_Text_Variable + 10);
310              Schreibe_Buchstabe(' ', sans_mono_12, 10, Anfang_Text_Variable + 10);
311              Schreibe_Buchstabe(' ', sans_mono_12, 12, Anfang_Text_Variable + 10);
312              Schreibe_Buchstabe(' ', sans_mono_12, 14, Anfang_Text_Variable + 10);
313              break;
314         case 3:
315              Schreibe_Buchstabe(' ', sans_mono_12, 6, Anfang_Text_Variable + 10);
316              Schreibe_Buchstabe(' ', sans_mono_12, 8, Anfang_Text_Variable + 10);
317              Schreibe_Buchstabe('*', sans_mono_12, 10, Anfang_Text_Variable + 10);
318              Schreibe_Buchstabe(' ', sans_mono_12, 12, Anfang_Text_Variable + 10);
319              Schreibe_Buchstabe(' ', sans_mono_12, 14, Anfang_Text_Variable + 10);
320              break;
321         case 4:
322              Schreibe_Buchstabe(' ', sans_mono_12, 6, Anfang_Text_Variable + 10);
323              Schreibe_Buchstabe(' ', sans_mono_12, 8, Anfang_Text_Variable + 10);
324              Schreibe_Buchstabe(' ', sans_mono_12, 10, Anfang_Text_Variable + 10);
325              Schreibe_Buchstabe('*', sans_mono_12, 12, Anfang_Text_Variable + 10);
326              Schreibe_Buchstabe(' ', sans_mono_12, 14, Anfang_Text_Variable + 10);
327              break;
328         case 5:
329              Schreibe_Buchstabe(' ', sans_mono_12, 6, Anfang_Text_Variable + 10);
330              Schreibe_Buchstabe(' ', sans_mono_12, 8, Anfang_Text_Variable + 10);
331              Schreibe_Buchstabe(' ', sans_mono_12, 10, Anfang_Text_Variable + 10);
332              Schreibe_Buchstabe(' ', sans_mono_12, 12, Anfang_Text_Variable + 10);
333              Schreibe_Buchstabe('*', sans_mono_12, 14, Anfang_Text_Variable + 10);
334              break;
335         default:
336             break;
337     }
338
339     return;
340 }
341
342 void Einstellmenue()
343 {
344     /*In diesem UP könne die Laufdauer des Lüfters in Minuten,
345      * die Wiederholung des Einschaltens in Minuten und die Taupunktdifferenz
346      * zwischen Aussen und Keller eingestellt werden bei der der Lüfter anläuft
347      * */
348
349     #define Anfang_Text  290  //Anfang des Textes
350     #define Anfang_Text_Variable  30 //Anfang der Ausgabe der Variablen
351
352     char Titel[] = {"Einstellungen"};
353     char Laufzeit_str[] = {"Einschaltzeit in Min:"};
354     char Periode_str[] = {"Periode in Min(1-250):"};
355     char Taup_Diff_str[] = {"Taupunktdifferenz in grd:"};
356     char Temp_Min_Keller_str[] = {"Temperaturminimum Keller:"};
357     char Werkseinstellungen_str[] = {"Werk Einstellungen?"};
358
359     static uint8_t Auswahl = 1;
360     static uint8_t Auswahl_ein = 1;
361     static uint8_t Werteeingabe = 0;
362     static uint8_t Werte_alt = 1;
363
364
365     //Texte für Eingabe
366     Schreibe_Text(Titel, sans_mono_24, 4, 280);
367     Schreibe_Text(Laufzeit_str, sans_mono_12, 6, Anfang_Text);
368     Schreibe_Text(Periode_str, sans_mono_12, 8, Anfang_Text);
369     Schreibe_Text(Taup_Diff_str, sans_mono_12, 10, Anfang_Text);
370     Schreibe_Text(Temp_Min_Keller_str, sans_mono_12, 12, Anfang_Text);
371     Schreibe_Text(Werkseinstellungen_str, sans_mono_12, 14, Anfang_Text);
372
373      //Ausgabe der Variablen
374      Schreibe_Text("   ", sans_mono_12, 6, Anfang_Text_Variable);
```

```
375     utoa(Dauer, Platz, 10);
376     Schreibe_Text(Platz, sans_mono_12, 6, Anfang_Text_Variable);
377
378     Schreibe_Text("   ", sans_mono_12, 8, Anfang_Text_Variable);
379     utoa(Intervall, Platz, 10);
380     Schreibe_Text(Platz, sans_mono_12, 8, Anfang_Text_Variable);
381
382     Schreibe_Text("   ", sans_mono_12, 10, Anfang_Text_Variable);
383     utoa(T_Punkt_Diff, Platz, 10);
384     Schreibe_Text(Platz, sans_mono_12, 10, Anfang_Text_Variable);
385
386     Schreibe_Text("   ", sans_mono_12, 12, Anfang_Text_Variable);
387     utoa(Temp_Min_Keller, Platz, 10);
388     Schreibe_Text(Platz, sans_mono_12, 12, Anfang_Text_Variable);
389
390     if (Werte_alt == 2)
391     {
392         Schreibe_Text("ja ", sans_mono_12, 14, Anfang_Text_Variable);
393     }
394     else
395     {
396         Schreibe_Text("nee", sans_mono_12, 14, Anfang_Text_Variable);
397     }
398
399     if (Auswahl_ein == 1)
400     {
401         Auswahl_Einstellungen(Auswahl);
402         Auswahl = Wert;
403         if (Wert > 5) Wert = 1;
404
405
406         if(Bestaetigungstaste())
407         {
408         Auswahl_ein = 0;
409         Werteeingabe = 1;
410         Wert = 1;
411         }
412     }
413
414     if (Werteeingabe == 1)
415     {
416         switch (Auswahl)
417         {
418             case 1:
419                 Dauer = Wert;
420                 if (Wert > 30) Wert = 1;
421                 if (Bestaetigungstaste())
422                 {
423                     Auswahl = 2;
424                     Auswahl_ein = 1;
425                     Werteeingabe = 0;
426                     Wert = 2;
427                     eeprom_write_byte(&ee_Dauer, Dauer);
428                 }
429                 break;
430
431             case 2:
432                // Wert = Dauer + 1;   //Intervall muss größer als Dauer sein
433                 Intervall = Wert;
434                 if (Wert > 250) Wert = Dauer + 1;
435                 if (Bestaetigungstaste())
436                 {
437                     Auswahl = 3;
438                     Auswahl_ein = 1;
439                     Werteeingabe = 0;
440                     Wert = 3;
441                     Auswahl = 3;
442                     eeprom_write_byte(&ee_Intervall, Intervall);
443                 }
444                 break;
445
446             case 3:
447                 T_Punkt_Diff = Wert;
448                 if (Wert > 15) Wert = 1;
449                 if (Bestaetigungstaste())
```

```
450                 {
451                     Auswahl = 4;
452                     Auswahl_ein = 1;
453                     Werteeingabe = 0;
454                     Wert = 4;
455                     eeprom_write_byte(&ee_T_Punkt_Diff, T_Punkt_Diff);
456                 }
457                 break;
458 
459             case 4:
460                 Temp_Min_Keller = Wert;
461                 if (Wert > 15) Wert = 1;
462                 if (Bestaetigungstaste())
463                 {
464                     Auswahl = 5;
465                     Auswahl_ein = 1;
466                     Werteeingabe = 0;
467                     Wert = 5;
468                     eeprom_write_byte(&ee_Temp_Min_Keller, Temp_Min_Keller);
469                 }
470                 break;
471 
472      /*Stellt die alten Werte für Einschaltzeit, Periode, Taupunktdifferenz
473      * und Temperaturminimum wieder her.*/
474             case 5:
475                 Werte_alt = Wert;
476                 if (Wert > 2) Wert = 1;
477                 if (Bestaetigungstaste())
478                 {
479                     Auswahl = 1;
480                     Auswahl_ein = 1;
481                     Werteeingabe = 0;
482                     Wert = 1;
483                     if (Werte_alt == 2) Anfangswerte();
484                     Werte_alt = 1;
485                 }
486                 break;
487 
488             default:
489                 break;
490         }
491     }
492 
493     Display_Fram(); //Anzeige aktualisieren
494 }
495 
496 uint8_t CRC_errechnen(uint8_t Wert[])
497 {
498     uint8_t i, n;
499     uint8_t Datum;
500    // uint8_t schluessel = 0x31;
501     uint8_t crc = 0;
502     #define schluessel 0x31
503 
504     for(n=0;n<2;n++)
505     {
506         Datum = Wert[n];
507         for (i=0;i<8;i++)
508         {
509             if ( ((crc>>7) & 1) != ((Datum & (1<<(7-i)))>>(7-i)) )
510                 crc = (crc<<1) ^ schluessel;
511             else
512                 crc = (crc << 1);
513         }
514     }
515     return (crc);
516 }
517 
518 #ifdef HTU
519 void Temperatur_Feuchte_auslesen(uint8_t Sensor)
520 {
521     /*Für den Temperatursensor HTU21D.
522      *Programm liest die beiden Sensoren aus, ermittelt den Taupunkt und schreibt die Werte
523      * Temperatur, Feuchte und Taupunkte in die globalen Variablen (Temperatur_aussen,
524      * Feuchte_aussen, Temperatur_Keller, Feuchte_Keller, T_Punkt_A und T_Punkt_K.)
```

```
525       */
526     uint8_t Temp[3];
527     uint8_t Feuchte_gerundet;
528     int16_t Temperatur_gerundet = -50;
529     uint32_t Temporaer;
530 
531     //Initialisierung Aussen
532     Schreibe_Byte_an_Adresse(Adresse_HTU21D, Adresse_write_Register, 0x03, 1);    //Auflösung Feuchte:8Bit, Temperatur:12Bit
533     //Initialisierung Keller
534     Schreibe_Byte_an_Adresse_1(Adresse_HTU21D, Adresse_write_Register, 0x03, 1);    //Auflösung Feuchte:8Bit, Temperatur:12Bit
535 
536     if (Sensor == 1)    //Aussensensor
537     {
538         //Temperatur auslesen
539         Lese_Byts_ab_Adresse(Adresse_HTU21D, Adresse_Temperatur_no_hold, Temp, 3, 1); //Temperatur auslesen
540         //Temp[0]=Msb, Temp[1]=LSB, Temp[2]=Checksumme
541         Temporaer = 0;
542         Temporaer = (((Temporaer | (uint32_t)Temp[0]) << 8) | (uint32_t)Temp[1]);
543         Temporaer = Temporaer & 0x0000FFFC;   //Letzten beiden Bits zu Null(Status Bit)
544 
545         //CRC überprüfen
546         if (CRC_errechnen(Temp) != Temp[2])
547         {
548             Temperatur_aussen = 999;
549         }
550         else
551         {
552                 Temperatur_aussen = (uint16_t) (((1757*Temporaer) >> 16) - 469);
553                 //Temperatur auf ganze °C runden
554                 Temperatur_gerundet = Temperatur_aussen + 5 - (Temperatur_aussen -5)%10;
555         }
556 
557         //Feuchte auslesen
558         Lese_Byts_ab_Adresse(Adresse_HTU21D, Adresse_Feuchte_no_hold, Temp, 3, 1);
559         Temporaer = 0;
560         Temporaer = (((Temporaer | (uint32_t)Temp[0]) << 8) | (uint32_t)Temp[1]);
561         Temporaer = Temporaer & 0x0000FFFC;   //Letzten beiden Bits zu Null(Status Bit)
562 
563         //CRC überprüfen
564         if (CRC_errechnen(Temp) != Temp[2])
565         {
566             Feuchte_aussen = 0;
567             return;
568         }
569 
570             Feuchte_aussen = (uint8_t) (((125*Temporaer) >> 16) -6);
571             Feuchte_gerundet = Feuchte_aussen +  2 - (Feuchte_aussen - 3)%5;
572 
573             //Taupunkt aussen ermitteln (aus der Tabelle)
574             if (Feuchte_gerundet == 100) T_Punkt_A = Temperatur_aussen;
575             else T_Punkt_A = Taupunkt_ermitteln(Temperatur_gerundet, Feuchte_gerundet);
576     }
577 
578     if (Sensor == 2) //kellersensor
579     {
580          //Temperatur auslesen
581         Lese_Byts_ab_Adresse_1(Adresse_HTU21D, Adresse_Temperatur_no_hold, Temp, 3, 1); //Temperatur auslesen
582         //Temp[0]=Msb, Temp[1]=LSB, Temp[2]=Checksumme
583         Temporaer = 0;
584         Temporaer = (((Temporaer | (uint32_t)Temp[0]) << 8) | (uint32_t)Temp[1]);
585         Temporaer = Temporaer & 0x0000FFFC;   //Letzten beiden Bits zu Null(Status Bit)
586 
587         //CRC überprüfen
588         if (CRC_errechnen(Temp) != Temp[2])
589         {
590             Temperatur_Keller = 999;
591         }
592         else
593         {
594                 Temperatur_Keller = (uint16_t) (((1757*Temporaer) >> 16) - 469);
595                 //Temperatur auf ganze °C runden
```

```
596                 Temperatur_gerundet = Temperatur_Keller + 5 - (Temperatur_Keller -5)%10;
597         }
598
599         //Feuchte auslesen
600         Lese_Byts_ab_Adresse_1(Adresse_HTU21D, Adresse_Feuchte_no_hold, Temp, 3, 1);
601         Temporaer = 0;
602         Temporaer = (((Temporaer | (uint32_t)Temp[0]) << 8) | (uint32_t)Temp[1]);
603         Temporaer = Temporaer & 0x0000FFFC;   //Letzten beiden Bits zu Null(Status Bit)
604
605         //CRC überprüfen
606         if (CRC_errechnen(Temp) != Temp[2])
607         {
608             Feuchte_Keller = 0;
609             return;
610         }
611
612             Feuchte_Keller = (uint8_t) (((125*Temporaer) >> 16) -6);
613             Feuchte_gerundet = Feuchte_Keller +  2 - (Feuchte_Keller - 3)%5;
614
615             //Taupunkt aussen ermitteln (aus der Tabelle)
616             if (Feuchte_gerundet == 100) T_Punkt_A = Temperatur_Keller;
617             else T_Punkt_K = Taupunkt_ermitteln(Temperatur_gerundet, Feuchte_gerundet);
618     }
619     return;
620 }
621 #endif
622
623 #ifdef HDC
624 void Temperatur_Feuchte_auslesen(uint8_t Sensor)
625 {
626     /*Für den Sensor HDC1080
627      * Programm liest die beiden Sensoren aus, ermittelt den Taupunkt und schreibt die Werte
628      * Temperatur, Feuchte und Taupunkte in die globalen Variablen (Temperatur_aussen,
629      * Feuchte_aussen, Temperatur_Keller, Feuchte_Keller, T_Punkt_A und T_Punkt_K.)
630       */
631     uint8_t Temp[4];
632     uint8_t Feuchte_gerundet;
633     int16_t Temperatur_gerundet;
634     uint32_t Temporaer;
635     uint8_t Modus[] = {0x10, 0x00}; //Normalmodus
636     /*
637 #define Software_reset 7    //Normal = 0
638 #define Heizung 5           //Heizung an = 1
639 #define Auslesemodus 4      //1: Temperatur und Feuchte sequentiell;0: Temperatur und Feuchte einzeln
640 #define Temperatur_Aufloesung 2 //0: 14Bit; 1: 11Bit
641 #define Feuchte_Aufloesung_1 1   //00: 14Bit; 01: 11Bit; 10: 8Bit
642 #define Feuchte_Aufloesung_0 0
643    */
644     if (Sensor == 1)    //Aussensensor
645     {
646
647         Schreibe_Byts_ab_Adresse(Adresse_HDC1080, 0x02, Modus, 2, 1); //Modus setzen
648
649         Lese_Byts_ab_Adresse(Adresse_HDC1080, 0x00, Temp, 4, 1); //Auslesen des Sensors
650          //Berechnung der Temperatur mit einer Nachkommastelle
651         Temporaer = 0;
652         Temporaer = (((Temporaer | (uint32_t)Temp[0]) << 8) | (uint32_t)Temp[1]);
653         Temperatur_aussen = 156; //(int16_t)(((Temporaer * 1650) >> 16) - 400);
654         //Temperatur auf ganze °C runden
655         Temperatur_gerundet = Temperatur_aussen + 5 - (Temperatur_aussen -5)%10;
656         //Berechnung der Feuchte
657         Temporaer = 0;
658         Temporaer = (((Temporaer | (uint32_t)Temp[2]) << 8) | (uint32_t)Temp[3]);
659         Feuchte_aussen = 33; //(uint8_t)((Temporaer * 100) >> 16);
660         //Auf 5% Luftfeuchte runden
661         Feuchte_gerundet = Feuchte_aussen +  2 - (Feuchte_aussen - 3)%5;
662         //Taupunkt aussen ermitteln (aus der Tabelle)
663         if (Feuchte_gerundet == 100) T_Punkt_A = Temperatur_aussen;
664         else T_Punkt_A = Taupunkt_ermitteln(Temperatur_gerundet, Feuchte_gerundet);
665     }
666
667     if (Sensor == 2) //kellersensor
668     {
669          Schreibe_Byts_ab_Adresse_1(Adresse_HDC1080, 0x02, Modus, 2, 1); //Modus setzen
670
```

```
671          Lese_Byts_ab_Adresse_1(Adresse_HDC1080, 0x00, Temp, 4, 1); //Auslesen des Sensors
672           //Berechnung der Temperatur mit einer Nachkommastelle
673         Temporaer = 0;
674         Temporaer = (((Temporaer | (uint32_t)Temp[0]) << 8) | (uint32_t)Temp[1]);
675         Temperatur_Keller = (int16_t)(((Temporaer * 1650) >> 16) - 400);
676         //Tempertur auf ganze °C runden
677         Temperatur_gerundet = Temperatur_Keller + 5 - (Temperatur_Keller -5)%10;
678         //Berechnung der Feuchte
679         Temporaer = 0;
680         Temporaer = (((Temporaer | (uint32_t)Temp[2]) << 8) | (uint32_t)Temp[3]);
681         Feuchte_Keller = (uint8_t)((Temporaer * 100) >> 16);
682          //Auf 5% Luftfeuchte runden
683         Feuchte_gerundet = Feuchte_Keller + 2 - (Feuchte_Keller - 3)%5;
684         //Taupunkt Keller ermitteln
685         if (Feuchte_gerundet == 100) T_Punkt_K = Temperatur_Keller;
686         else T_Punkt_K = Taupunkt_ermitteln(Temperatur_gerundet, Feuchte_gerundet);
687     }
688 }
689 #endif
690
691 void Sensor_regenerieren(uint8_t Sensor)
692 {
693     //scheint nicht zu klappen, denn keine Temperaturerhöhung messbar
694     uint8_t Modus[] = {0x20, 0x00}; //Modus heizen
695
696     if (Sensor == 1)
697     {
698         Schreibe_Byts_ab_Adresse(Adresse_HDC1080, 0x02, Modus, 2, 1); //Heizung an
699     }
700 }
701
702
703 int8_t Taupunkt_ermitteln(int16_t Temperatur, uint8_t Feuchte)
704 {
705     /*Der Taupunkt wird anhand einer Tabelle ermittelt.
706      * Der Tabellenbereich geht von 0 bis 33grd, und die Feuchte von 30% bis 95% Ausserhalb des
        Bereiches wird 100 zurückgegeben.
707       */
708     int8_t Taupunkt;
709     if ((Temperatur/10 < 0) || (Temperatur/10 > 33) || (Feuchte < 28))   //Tabellenbegrenzung
710     {
711         Taupunkt = 100;
712     }
713     else
714     {
715         Taupunkt = pgm_read_byte(&taupunkt[Temperatur/10][Feuchte/5 - 6]);  //Lese Taupunkt aus der
        Tabelle
716     }
717
718     return Taupunkt;
719 }
720
721 void Luefter_ein_aus()
722 //Zeitsteuerung des Lüfters
723 {
724     if(Bedingung_Luefter_ein())
725     {
726         if(Min < Dauer)
727         {
728             Luefter_an;
729         }
730         else
731         {
732             Luefter_aus;
733         }
734     }
735     else
736     {
737         Luefter_aus;
738     }
739 }
740
741 uint8_t Bedingung_Luefter_ein()
742 {
743     /* Hier werden die Bedingungen für das Einschalten des Lüfters gesetzt. (Ausser Zeitvorgaben)
```

```
744     Temperatur im Keller > Temp_Min_Keller(8grd) muss sein; ausserdem entweder die Temperatur aussen
        unter 0grd, oder der Taupunkt aussen ist T_Punkt_Diff(5grd) niedriger als im Keller.
745     Ist die Kellertemperatur kleiner als Temp_Min_Keller wird nur bei höherer Aussentemperatur und
        höherer Taupunktdifferenz eingeschaltet.
746     Das Ein- und Ausschalten des Lüfters erfolgt mit Hysterese (2grd), um ein ständiges Ein- und
        Ausschalten zu verhindern.
747     Bei Bedingung erfüllt erfolgt Rückgabewert = 1, ansonsten 0.*/
748
749     uint8_t Freigabe = 0;
750
751    //Aussentemperatur < 0 taupunkt prüfen
752       if (Temperatur_Keller > Temp_Min_Keller)
753      {
754         if (Temperatur_aussen < 0) return 1;   //Bei Temperaturen aussen unter 0 immer freigeben
755
756          //Lüfter mit Hysterese einschalten
757        if(Luefter_abfragen > 0)
758        {
759             if ((T_Punkt_A + T_Punkt_Diff - Hysterese) <= T_Punkt_K) Freigabe = 1;
760             else Freigabe = 0;
761        }
762        else
763        {
764             if ((T_Punkt_A + T_Punkt_Diff + Hysterese) <= T_Punkt_K) Freigabe = 1;
765             else Freigabe = 0;
766        }
767      }
768     else    //Temperatur im Keller unter festgelegte Grenze
769     {
770         if (Temperatur_aussen > Temp_Min_Keller) //Lüfter unter Bedingungen einschalten
771         {
772             //Lüfter mit Hysteres einschalten
773             if(Luefter_abfragen > 0)   //Lüfter eingeschaltet
774             {
775                 if ((T_Punkt_A + T_Punkt_Diff - Hysterese) <= T_Punkt_K) Freigabe = 1;
776                 else Freigabe = 0;
777             }
778             else
779             {
780                 if ((T_Punkt_A + T_Punkt_Diff + Hysterese) <= T_Punkt_K) Freigabe = 1;
781                 else Freigabe = 0;
782             }
783         }
784         else  Freigabe = 0;  //Temperatur aussen zu niedrig
785     }
786
787     return Freigabe;
788 }
789
790
791 //Timer1 Routine alle 10s
792
793 ISR(TIMER1_OVF_vect)
794
795 {
796     TCNT1 = 0xFFFF -Sec_Faktor; //Zähler mit Anfangswert laden für 10s
797
798     Takt += 10; //Taktzeit ist der Sekundenzähler*10
799
800     Merker_Temperatur_auslesen = 1; //Merker Tempetursensoren auslösen setzen
801     Merker_Anzeige_aktualisieren = 1;
802
803     if(Takt >= 60)
804     {
805         Takt = 0;  //Taktzeit wiederherstellen
806
807         Min++;
808         if (Min >= Intervall) Min = 0;
809     }
810  }
811
812 void Anfangswerte()
813 {
814     Dauer = eeprom_read_byte(&ee_Dauer_werk);
815     Intervall = eeprom_read_byte(&ee_Intervall_werk);
```

```
816     T_Punkt_Diff = eeprom_read_byte(&ee_T_Punkt_Diff_werk);
817     Temp_Min_Keller = eeprom_read_byte(&ee_Temp_Min_Keller_werk);
818 }
819
820 void Lese_Werte_aus_dem_EEPROM()
821 {
822     Intervall = eeprom_read_byte(&ee_Intervall);
823     Dauer = eeprom_read_byte(&ee_Dauer);
824     T_Punkt_Diff = eeprom_read_byte(&ee_T_Punkt_Diff);
825     Temp_Min_Keller = eeprom_read_byte(&ee_Temp_Min_Keller);
826 }
827
828 void Ports_init()
829 {
830     //Port D für die Anzeige, Port C für die Sensoren, Port B für Tasten usW.
831     DDRD &= ~(1<<E_Busy);   //Busy Eingang
832     PORTD |= (1<<E_Busy);   //Pullup Widerstand einschalten
833     DDRD |= (1<<E_Reset) | (1<<E_DC) | (1<<E_CS) | (1<<E_Clk) | (1<<E_DIN); //als Ausgang
834
835     DDRB |= (1<<Luefter) | (1<<Messung);   //Ausgänge
836     DDRB &= ~((1<<Taster) | (1<<Impuls_A) | (1<<Impuls_B) | (1<<Bestaetigung));   //Eingänge
837
838     return;
839 }
840
841 ISR (TIMER2_COMPA_vect)
842 {
843   //Timer für die Abfrage des Drehimpulsgebers (1ms)
844       static uint8_t Imp0=3;    //Bit0(B) und 1(A) vom Drehgeber
845       static uint8_t Imp1=3;    //alter Drehgeberwert
846       uint8_t Imp_EX;   //Exclusiv-oder Imp0 und Imp1 für die Richtung(+ oder -)
847
848     #define Schrittweite  1 //Schrittweite für das Zählen
849
850      if (Wert > 250) Wert = 1; //Begrenzung des Zählers
851
852      Imp0 = (A_Imp<<0) | (B_Imp<<0);    //Setzt die beiden Eingänge in die Variable Imp0, A Bit0, B
    ▒▒▒▒Bit1
853      if (Imp1 == Imp0) return;  //nichts verändert: raus
854      Imp_EX = Imp0 ^ Imp1;  //Exklusiv-Oder bilden
855      if (Imp0 == 3) //Auswertung nur einmal pro Impuls
856      {
857        if (Imp_EX == 1) Wert += Schrittweite;
858        else Wert -= Schrittweite;
859      }
860      if(Wert == 0) Wert= 1;
861      Imp1 = Imp0;   //alten Abfragewert speichern
862 }
863
864 void TMR1_2_init()
865 {
866     //Timer 1 für Zeit
867     TCCR1B |= (1<<CS12) | (1<<CS10);    //Vorteiler 1024
868     TIMSK1 |= (1<<TOIE1);   //Interrupt bei Überlauf
869     TCNT1 = 0xFFFF -Sec_Faktor; //Zähler mit Anfangswert laden für 1s
870
871     //Timer 2 für Drehgeber
872     TCCR2A |= (1<<WGM21) | (1<<COM2A1) | (1<<COM2A0);   //Vergleichsmodus, setzt OC2-Interrupt bei
    ▒▒▒▒Gleichheit
873     TCCR2B |= (1<<CS22) | (1<<CS21);    //Vorteiler 256
874     TCCR2B &= ~(1<<CS20);
875
876     Timer2_halt;    //Timerinterrupt2 stoppen
877     OCR2A = Zeit2;  //Vergleich setzen
878 }
879
880 int main()
881 {
882
883     wdt_disable();
884     Ports_init();
885   //  Anfangswerte();
886     Lese_Werte_aus_dem_EEPROM();
887     Reset();
888
```

```
889     Display_Init(lut_full_update);   //damit der gesamte Bildschirm aktualisiert (2Sekunden?)
890
891     Display_ini();    //Alles löschen (mit weiss überschreiben).
892
893     Begruessungsbild();
894     _delay_ms(5000);
895     Display_ini();    //Alles löschen (mit weiss überschreiben).
896
897     Reset();
898     Display_Init(lut_partial_update);   //damit werden nur die Änderungen aktualisiert
899
900     TMR1_2_init();
901     sei();
902     Timer1_start;
903
904     while(1)
905     {
906         if (Merker_Temperatur_auslesen == 1)    //durch dem Merker werden alle 10s die Sensoren
        ausgelesen
907         {
908        //     Sensor_regenerieren(1);
909         //     _delay_ms(5000);
910             Messung_an;    //LED Anzeige an
911             Temperatur_Feuchte_auslesen(1); //Aussenwerte
912             Temperatur_Feuchte_auslesen(2); //Kellerwerte
913
914             Merker_Temperatur_auslesen = 0; //Merker löschen
915             _delay_ms(500); //damit die LED besser zu sehen ist
916             Messung_aus;    //LED Anzeige aus
917         }
918
919         Anzeigewahl = Menueauswahl();
920
921         if (Anzeigewahl == 1)
922         {
923
924             Einstellmenue();
925             _delay_ms(500);
926         }
927
928        if  ((Merker_Anzeige_aktualisieren == 1) && (Anzeigewahl == 0)) //Anzeige wird alle 10s
        aktualisiert
929        {
930           Merker_Anzeige_aktualisieren = 0; //Merker zurücksetzen
931          // _delay_ms(30);    //warte bis Messwerte eingelesen sind, kann entfallen?
932           Hauptanzeige();
933
934        }
935         Luefter_ein_aus();  //Lüftersteuerung
936
937     };
938
939     return (0);
940 }
941
942
943
```