```
//Einbinden der benoetigten Bibliotheken
#include <SimpleDHT.h>
#include <PinChangeInterrupt.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal_I2C.h>
#include "HL1606strip.h"

//LCD Display adressieren
LiquidCrystal_I2C lcd(0x27, 20, 4);



//_____________________________________________________________
//Definieren benoetigter Variablen
//Timer
volatile int hours = 0;              //Startzeit Stunde
volatile int minutes = 0;            //Startzeit Minute
volatile int seconds = 0;            //Startzeit Sekunde
volatile int plusTimer = 30;         //Schrittgroeße Timer erhoehen in Minuten
volatile int minusTimer = 30;        //Schrittgroeße Timer verringern in Minuten
volatile bool timerFlag = true;
volatile bool buttonTimerState =true;
volatile bool pause =true;
volatile float value = 49910;


//Temp
volatile int sollTemp = 20;          //Voreinstellung Soll-Temperatur
volatile double istTemp = 0.0;       //Gemessene Ist-Temperatur
volatile int plusTemp = 1;           //Schrittgroesse Temperatur erhoehen
volatile int minusTemp = 1;          //Schrittgroesse Temperatur verringern
volatile int maxTemp = 35;           //Maximale einstellbare Temperatur
volatile int minTemp = 15;           //Minimale einstellbare Temperatur
volatile int notAusTemp = 80;        //Temperatur Heizmatte fuer Not-Aus
volatile bool buttonHeatState = false;
volatile bool heat = false;
volatile bool aus = true;
volatile bool notAus = false;
volatile int zaehler = 0;

//Relais
volatile int relaisState = 0;

//LED
volatile bool buttonLEDState = false;

//Button
//Anschliessen der Elemente aus dem Schaltplan an die entsprechenden Pins (Nummer steht fuer Ein-/Ausgang
auf dem Arduino-Board)
#define PIN_STARTLED 4               //Taster Licht ein/aus
#define PIN_STARTHEAT 5              //Taster Start/Stop Heizung
```

```
#define PIN_INCREASETEMP 6              //Taster Temperatur plus
#define PIN_DECREASETEMP 7              //Taster Temperatur minus
#define PIN_STARTTIMER 8               //Taster Start/Stop Timer
#define PIN_INCREASETIMER 9             //Taster Timer plus
#define PIN_DECREASETIMER 10             //Taster Timer minus
#define PIN_DHT_MATTE 11                //Temperatursensor DS18B20 fuer Temperaturmessung an Heizmatte
fuer Sicherheit
#define PIN_RELAIS 12                 //Datenpin des Relais
#define PIN_DHT_AIR 13                //Temperatursensor DHT22 fuer Lufttemperaturmessung
#define PCINT_MODE RISING

//LED-Streifen
#define STRIP_D 3
#define STRIP_C 2
#define STRIP_L 0
HL1606strip strip = HL1606strip(STRIP_D, STRIP_L, STRIP_C, 2);

//DHT22
SimpleDHT22 dht22(PIN_DHT_AIR);
float temperature = 0.0;
float humidity = 0.0;

//DS18B20
OneWire oneWire(PIN_DHT_MATTE);
DallasTemperature sensors(&oneWire);

//_____________________________________________________________

void increaseTimer(void);
void decreaseTimer(void);
void startTimer(void);
void increaseTemp(void);
void decreaseTemp(void);
void startHeat(void);
void startLED(void);
void initTimer();

//_____________________________________________________________
//Setup
void setup()
{
 lcd.init();             //Initialisierung LCD Display
 lcd.backlight();        //Hintergrundbeleuchtung Display ein
 sensors.begin();

 //Definitionen der Pins als Eingaenge/Ausgaenge und der Interruptfunktionen fuer die Taster
 //Timer_____________________________________________________
 pinMode(PIN_INCREASETIMER,INPUT_PULLUP);
 pinMode(PIN_DECREASETIMER,INPUT_PULLUP);
 pinMode(PIN_STARTTIMER,INPUT_PULLUP);

 attachPCINT(digitalPinToPCINT(PIN_INCREASETIMER), increaseTimer, RISING);
```

```
  attachPCINT(digitalPinToPCINT(PIN_DECREASETIMER), decreaseTimer, RISING);
  attachPCINT(digitalPinToPCINT(PIN_STARTTIMER), startTimer, RISING);

  //Temp_____________________________________________________________
  pinMode(PIN_INCREASETEMP,INPUT_PULLUP);
  pinMode(PIN_DECREASETEMP,INPUT_PULLUP);
  pinMode(PIN_STARTHEAT,INPUT_PULLUP);

  attachPCINT(digitalPinToPCINT(PIN_INCREASETEMP), increaseTemp, RISING);
  attachPCINT(digitalPinToPCINT(PIN_DECREASETEMP), decreaseTemp, RISING);
  attachPCINT(digitalPinToPCINT(PIN_STARTHEAT), startHeat, RISING);

  //Relais___________________________________________________________
  pinMode(PIN_RELAIS, OUTPUT);
  digitalWrite(PIN_RELAIS, LOW);

  //LED______________________________________________________________
  pinMode(PIN_STARTLED, INPUT_PULLUP);
  attachPCINT(digitalPinToPCINT(PIN_STARTLED), startLED, RISING);

  initTimer();
}

//_____________________________________________________________________________

void loop() {


  if (timerFlag)
  {


    if (buttonLEDState)
    {
      colorWipe(WHITE, 40);
    }
    else
    {
      for (uint8_t i=0; i < strip.numLEDs(); i++)
      {
      strip.setLEDcolor(i, BLACK);
      }
      strip.writeStrip();
    }

    relaisState = digitalRead(PIN_RELAIS);
    dht22.read2(&temperature, &humidity, NULL);
    sensors.requestTemperatures();
    float tempMatte = sensors.getTempCByIndex(0);

    //Bildschirmausgabe wenn Not-Aus-Temperatur an der Heizmatte erreicht ist
    if (tempMatte>notAusTemp)
```

```
{
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temperatur der");
  lcd.setCursor(0,1);
  lcd.print("Heizmatte > ");
  lcd.print(notAusTemp);
  lcd.print("\337C ");
  lcd.setCursor(0,2);
  lcd.print("Temp an Matte:");
  lcd.setCursor(0,3);
  lcd.print(tempMatte,1);
  lcd.print("\337C ");
  digitalWrite(PIN_RELAIS, LOW);
  notAus = true;
  timerFlag = true;
}

//Bildschirmausgabe wenn Not-Aus-Temperatur an der Heizmatte erreicht wurde
if (notAus && tempMatte <= notAusTemp)
{
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temperatur der");
  lcd.setCursor(0,1);
  lcd.print("Heizmatte war > ");
  lcd.print(notAusTemp);
  lcd.print("\337C ");
  lcd.setCursor(0,2);
  lcd.print("Temp an Matte:");
  lcd.setCursor(0,3);
  lcd.print(tempMatte,1);
  lcd.print("\337C ");
  digitalWrite(PIN_RELAIS, LOW);
  notAus = true;
  timerFlag = true;
}
else if (notAus == false)
{
  //Regelung --> Relais ein-/ausschalten
  if (buttonHeatState)  //war nicht da
  {
    if((float) temperature >= sollTemp + 0.5)
    {
      digitalWrite(PIN_RELAIS, LOW);
    }

    if((float) temperature <= sollTemp - 0.5)
    {
      digitalWrite(PIN_RELAIS, HIGH);
    }
  }
```

```
  else
  {
    digitalWrite(PIN_RELAIS, LOW);
  }

 //Bildschirmausgabe der Temperatur, Luftfeuchtigkeit und des Zustandes der Heizung
 lcd.setCursor(0,0);
 lcd.print("Ist: ");
 lcd.print((float)temperature,0);
 lcd.print("\337C ");
 lcd.print("Soll: ");
 lcd.print(sollTemp);
 lcd.print("\337C ");
 lcd.setCursor(0,1);
 lcd.print("Luftfeuchte: ");
 lcd.print((float)humidity , 0);
 lcd.print("%  ");
 lcd.setCursor(0,3);
 lcd.print("Regelung ");
 if (heat)
 {
   lcd.print("ein");
 }
 else
 {
   lcd.print("aus");
 }
 timerFlag = false;
 delay (10);

 //Bildschrimausgabe der aktuellen Zeit
 lcd.setCursor(0,2);
 lcd.print("Zeit: ");
 if (hours < 10)
 {
   lcd.setCursor(6,2);
   lcd.print(0);
   lcd.print(hours);
   lcd.print(":");
 }
 else
 {
   lcd.setCursor(6,2);
   lcd.print(hours);
   lcd.print(":");
 }
 if (minutes < 10)
 {
   lcd.setCursor(9,2);
   lcd.print(0);
   lcd.print(minutes);
   lcd.print(":");
```

```
   }
   else
   {
    lcd.setCursor(9,2);
    lcd.print(minutes);
    lcd.print(":");
   }
   if (seconds < 10)
   {
    lcd.setCursor(12,2);
    lcd.print(0);
    lcd.print(seconds);
   }

   else if (seconds >=10)
   {
    lcd.setCursor(12,2);
    lcd.print(seconds);
   }

   delay (10);
 }
 }
delay(10);
}

//_____________________________________________________________________________
//Funktion zur Erfassung einer Sekunde in Echtzeit
void initTimer()
{
 noInterrupts();
 TCCR1A = 0;
 TCCR1B = 0;
 TCNT1 = value;
 TCCR1B |= (1 << CS10)|(1 << CS12);
 TIMSK1 |= (1 << TOIE1);
 interrupts();
}

//Funktion die nach Ablauf einer Sekunde in Echtzeit aufgerufen wird
ISR(TIMER1_OVF_vect)
{
 TCNT1 = value;

 //Wenn Timer 00:00:00 ist, soll Heizung abschalten und keine Berrechnung der Zeit stattfinden
 if (hours == 0 && minutes == 0 && seconds == 0 && aus == false )
 {
  digitalWrite(PIN_RELAIS, LOW);
  buttonHeatState = false;
  aus = true;
  pause = true;
  buttonTimerState = false;
```

```
   heat = false;
  }

  //Berrechnung der Zeit (eine Sekunde subtrahieren) wenn nicht 00:00:00
  else
  {
   if (pause == false)
   {
    timerFlag = true;

   if (seconds == 0 && minutes > 0)
   {
    seconds = 59;
    minutes = minutes - 1;
   }
   else if (seconds == 0 && minutes == 0)
   {
    seconds = 59;
    minutes = 59;
    hours = hours - 1;
   }
   else if (minutes == 0 && seconds > 0 || seconds > 0 && minutes > 0)
   {
    seconds = seconds - 1;
   }
    aus = false;
   }
  }
  timerFlag = true;

  zaehler = zaehler + 1;
}

//_____________________________________________________________________________
//Interruptfunktion fuer Tastendruck von Taster "Timer erhoehen"
void increaseTimer(void)
{
  if (zaehler <= 1)
  {

  }
  else
  {

   aus = false;
   if ((minutes + plusTimer) >= 60)
   {
    int rest = (minutes + plusTimer)%60;
    minutes = rest;
    hours = hours + 1;
    if (hours >= 24)
    {
```

```
      hours = 24;
      minutes = 0;
      seconds = 0;
    }
   }
   else
   {
    minutes = minutes + plusTimer;
    if (hours >= 24)
    {
     hours = 24;
     minutes = 0;
     seconds = 0;
    }
   }
  }
}
//_____________________________________________________________________________
//Interruptfunktion fuer Tastendruck von Taster "Timer verringern"
void decreaseTimer(void)
{
 if (zaehler <= 1)
 {
 }
 else
 {
  if ((minutes - minusTimer) < 0)
  {
   int rest = 60 + (minutes - minusTimer);
   minutes = rest;
   hours = hours - 1;

   if (hours < 0)
   {
    hours = 0;
    minutes = 0;
    seconds = 0;
   }
  }
  else
  {
   minutes = minutes - minusTimer;
   if (hours < 0)
   {
    hours = 0;
    minutes = 0;
    seconds = 0;
   }
  }
 }
}
//_____________________________________________________________________________
```

```
//Interruptfunktion fuer Tastendruck von Taster "Start/Stop Timer"
void startTimer(void)
{
 if (zaehler <= 1)
 {
 }
 else
 {
  if (hours == 0 && minutes == 0 && seconds == 0)
  {
  }
  else
  {
   if (buttonTimerState || buttonTimerState ==false && pause == true)        //NEU!!-->  || buttonTimerState
==false && pause == true
   {
    initTimer();
    buttonTimerState = false;
    pause = false;
   }
   else
   {
    pause = true;
    buttonTimerState = true;
   }
  }
 }
}
//_____________________________________________________________________________
//Interruptfunktion fuer Tastendruck von Taster "Temperatur erhoehen"
void increaseTemp(void)
{
 if (zaehler <= 1)
 {
 }
 else
 {
  if (sollTemp < maxTemp)
  {
   sollTemp = sollTemp + plusTemp;
  }
 }
}
//_____________________________________________________________________________
//Interruptfunktion fuer Tastendruck von Taster "Temperatur verringern"
void decreaseTemp(void)
{
 if (zaehler <= 1)
 {
 }
 else
 {
```

```
   if (sollTemp > minTemp)
   {
     sollTemp = sollTemp - minusTemp;
   }
  }
}
//_____________________________________________________________________________
//Interruptfunktion fuer Tastendruck von Taster "Start/Stop Heizung"
void startHeat(void)
{
  if (zaehler <= 1)
  {
  }
  else
    {
   if (buttonHeatState)
   {
     buttonHeatState = false;
     heat = false;
   }
   else
   {
     buttonHeatState = true;
     heat = true;
   }
  }
}
//_____________________________________________________________________________
//Interruptfunktion fuer Tastendruck von Taster "LED ein/aus"
void startLED(void)
{
  if (zaehler <= 1)
  {
  }
  else
  {
   if (buttonLEDState)
   {
     buttonLEDState = false;
   }

   else
   {
     buttonLEDState = true;
   }
  }
}

//_____________________________________________________________________________
//Funktion für das Setzen einer Farbe auf dem LED-Streifen
void colorWipe(uint8_t color, uint8_t wait)
{
```

```
  uint8_t i;

  for (i=0; i < strip.numLEDs(); i++) {
     strip.setLEDcolor(i, color);
     strip.writeStrip();
     delay(wait);
  }
}
```