```
//  >> PINout, unvollstaenig  <<
/*

A0
A15

D0
D4  = SS fuer SD-Kart via Ethernet Shield
D10 = SS fuer Ethernet Controller
D20 = SDA I2C
D21 = SCL I2C
D50 = MISO (Ethernet) = ICSP 1
D51 = MOSI = ICSP 4
D52 = SCK
D53 = Hardware SS

*/

//_____________________________________________________________________________
_____
//_____________________________________________________________________________
_____
//_____________________________________________________________________________
_____

// DEFINITIONEN

//_____________________________________________________________________________
_____
//ETHERNET

#include <Ethernet.h>
#include <SPI.h>

char formkey[] = "dFpHMG1zbzAxXXXXXXXXXXXX"; //Replace with your GoogleKey
byte mac[] = {   0x00, 0x6A, 0x43, 0xC0, 0xXX, 0xXX};  //Replace with your
Ethernet shield MAC
byte ip[] = {  192,168,XXX,XXX};  //The Arduino device IP address

//byte subnet[] = { 255,255,255,0};

byte gateway[] = { 169, 254,XXX,XXX}; //Internet des Internetstellenden
Computers

byte server[] = { 173,194,70,139 }; // Google IP
//byte server[] = { 209,85,229,101 }; // Google IP

//Client client(server, 80);
EthernetClient client; //Arduino ist als Client definiert
int zeitpkt_letzte_uebertragung=0; //Hilfsvariable die den Minutenzeitpkt der
letzten ¸bertragung anzeigt
int uebertragungsfrequenz = 120; //H‰ufigkeit der ‹bertragung in sekunden
long vorhin=0;


//_____________________________________________________________________________
_____
//ECHTZEIT und LICHT-RELAY

#include <Wire.h>
#include "RTClib.h"


RTC_Millis RTC;

int minutenUhrzeit;
int Lichtpin = 28;
int Dimmpin = 32;
```

```
//_____________________________________________________________________________
_____
//DEFINITIONEN DER PARAMETER F‹R MESSUNGEN (IN VOLT)MITTELS DES
TEMPERATURF‹HLERS


#include <OneWire.h>

// Temperaturfuehler auf Digital Pin 11
OneWire  ds(11);  // on pin 11


//_____________________________________________________________________________
_____
//DEFINITIONEN DER PARAMETER ZUR ZURODNUNG VON OHM-INTERVALLEN ZU BESTIMMTEN
FALLNUMMERN


// Definition von Temperaturen
int adc_key_val[2] ={24, 27}; //Entspricht in C∞ 27, 24

//Ordnet den Temp-Intervallen eine Bewertung zu.
char msgs[3][18] = {"Kalt",
                    " OK ",
                    "Warm"
                    };

// Variable, die Temp-Intervalle eine Fallnummer zuordnet
int key;


// Variable, die sicherstellt, dass nur dann ein Programm gestartet wird, wenn
sich der gemessene Wertebereich ge‰ndert hat.
//Hilfswert = 6, damit beim ersten Durchlauf die If-Schleife des
Programmabschnittes ("ZUORDNEN BESTIMMTER AKTIONEN ZU DEN DER Temp-INTERVALLEN
ZUGEORDNETEN FALLNUMMERN")
//durchgef¸hrt werden kann.
int oldkey = 6;


// Anzahl der F‰lle (der 6. Fall wird im Unterprogramm "getKey" definiert).
int NUM_KEYS = 3;

//_____________________________________________________________________________
_____
//DEFINITIONEN DER DIGITALEN PINS


// Zuordnung der digitalen Pins entsprechend der Wertebereiche
int digitalPin[6] = {31, 33, 35, 48, 100, 40}; //drei ersten PinNr. f¸r dioden,
drei letzten respektiv die PinNr. f¸r die entsprechenden Relays.
                                                //Pin = 100 steht f¸r einen nicht
vergebenen Pin



//_____________________________________________________________________________
_____
//DEFINITIONEN DER PARAMETER DAS LCD-DISPLAY

#include <LiquidCrystal.h> //Digitalbib f¸r das LCD-Display


/*

  ALTE Pinverteilung, ALLE Digital:
```

```
 * LCD RS pin to digital pin 8
 ** LCD RS pin to digital pin 0
 * LCD Enable pin to digital pin 9
 ** LCD Enable pin to digital pin 1
 * LCD D4 pin to digital pin 4
 ** LCD D4 pin to digital pin 2
 * LCD D5 pin to digital pin 5
 ** LCD D5 pin to digital pin 3
 * LCD D6 pin to digital pin 6
 ** LCD D6 pin to digital pin 4
 * LCD D7 pin to digital pin 2 (vorher 7)
 ** LCD D7 pin to digital pin 5
 * LCD BL pin to digital pin 10 (bzw. 13?)
 ** LCD BL pin to +5V
 * KEY pin to analogl pin 0

 */

LiquidCrystal lcd(0, 1, 2, 3, 6, 5);

//_____________________________________________________________________________
_____
//DEFINITIONEN DER PARAMETER F‹R DIE LCD-TASTEN

char msgs_taste[5][20] = {"Right Key OK ",
                      "Select Key OK    ",
                      "Up Key OK  ",
                      "Down Key OK  ",
                      "Left Key OK" };

//int adc_taste_val[5] ={50, 200, 400, 600, 800 };
int adc_taste_val[5] ={50, 400, 500, 600, 750 };

int adc_taste_in;

int taste=-1;

int alte_taste=-1;

int NUM_KEYS_TASTEN = 5;

//_____________________________________________________________________________
_____
//DEFINITIONEN DER PARAMETER ZUR MANUELLEN ‹BERBR‹CKUNG DER AKTIONSSSCHLEIFE

int ueberbrueckung_luefter = 0; //‹berbr¸ckungsparameter f¸r den l¸fter
int ueberbrueckung_heizen = 0;//‹vberbr¸ckungsparameter f¸r den Heizstab

//_____________________________________________________________________________
_____
//DEFINITIONEN DER PARAMETER ZUR HERUNTERK‹HLUNG

int temp_krit_kuehlen = 26; //Definiert kritischen wert in Temp, bei dessen
unterschreitung die herunterk¸hlung beendet sein soll. (26 Grad)
int herunterkuehlen; //Pr¸fvariable, die besagt, ob die herunterk¸hlung
eingeschaltet ist.

//_____________________________________________________________________________
_____
//DEFINITIONEN DER PARAMETER ZUM HOCHHEIZEN

int temp_krit_heizen = 25; //Definiert kritischen wert in Temp, bei dessen
unterschreitung das Hochheizen beendet sein soll. (25 Grad)
int hochheizen; //Pr¸fvariable, die besagt, ob das Hochheizen eingeschaltet ist.

//_____________________________________________________________________________
_____
```

```
//DEFINITIONEN DER ZUSTANDSVARIABLEN ZUR ÜBERTRAGUNG INS INTERNET
int binaerstatus_licht;
int binaerzustand_luefter;
int binaerzustand_heizstab;
char an_aus[2][4] ={"AUS", "AN"};//Vektor um den Zustandsbinärcode NULL und EINS
in Worte zu übertragen

//_____________________________________________________________________________
_____
//DEFINITIONEN DER HILFSVARIABLEN

int relay; //Variable um den RelayPin im PinVektor zu errechnen.
//unsigned long value; // zur messung der Durchlaufzeit.

//_____________________________________________________________________________
_____
//_____________________________________________________________________________
_____
//_____________________________________________________________________________
_____



// uint8_t i;
  float average;

void setup(void) {
  //Serial.begin(9600);

  pinMode(digitalPin[0], OUTPUT);
  pinMode(digitalPin[1], OUTPUT);
  pinMode(digitalPin[2], OUTPUT);
  pinMode(digitalPin[3], OUTPUT);
  pinMode(digitalPin[4], OUTPUT); //wird nicht benutzt
  pinMode(digitalPin[5], OUTPUT);
  pinMode(digitalPin[6], OUTPUT);
  pinMode (10, OUTPUT);
  pinMode (53, OUTPUT);


//_____________________________________________________________________________
_____
  //LCD-Display Setup

  lcd.clear();
  lcd.begin(16, 2);
  lcd.setCursor(0,0);
  lcd.print(F("Starting..."));
  delay(1000);
  lcd.clear();


//_____________________________________________________________________________
_____
  //Licht-Steuerung mit Echt-Zeit

  Wire.begin();

    RTC.begin(DateTime(__DATE__, __TIME__));
    pinMode(Lichtpin, OUTPUT);
    pinMode(Dimmpin, OUTPUT);

  //if (! RTC.isrunning()) {
    //Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was compiled
   // RTC.adjust(DateTime(__DATE__, __TIME__));
  //}
```

```
//_____________________________________________________________________________________
_________
  //ETHERNETSETUP

  //Ethernet.begin(mac, ip , gateway);
  Ethernet.begin(mac, ip, gateway);
  delay(1000);

 // Serial.println("connecting...");                              // Auf Clients
warten

}


//_____________________________________________________________________________________
_______
//_____________________________________________________________________________________
_______
//_____________________________________________________________________________________
_______


// HAUPTPROGRAMM


void loop(void) {

  byte i;
  byte present = 0;
  byte type_s;
  byte data[12];
  byte addr[8];
  float celsius;



  if ( !ds.search(addr)) {
//    Serial.println("No more addresses.");
  //  Serial.println();
    ds.reset_search();
    delay(250);
    return;
  }

  Serial.print("ROM =");
  for( i = 0; i < 8; i++) {
   Serial.write(' ');
    Serial.print(addr[i], HEX);
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
   Serial.println("CRC is not valid!");
      return;
  }

  Serial.println();

  // the first ROM byte indicates which chip
  switch (addr[0]) {
    case 0x10:
      Serial.println("  Chip = DS18S20");  // or old DS1820
      type_s = 1;
      break;
    case 0x28:
      Serial.println("  Chip = DS18B20");
      type_s = 0;
      break;
```

```
    case 0x22:
      Serial.println("  Chip = DS1822");
      type_s = 0;
      break;
    default:
      Serial.println("Device is not a DS18x20 family device.");
      return;
  }


  ds.reset();
  ds.select(addr);
  ds.write(0x44,1);         // start conversion, with parasite power on at the
end

  delay(1000);     // maybe 750ms is enough, maybe not
  // we might do a ds.depower() here, but the reset will take care of it.

  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);         // Read Scratchpad



  Serial.print("  Data = ");
  Serial.print(present,HEX);
  Serial.print(" ");
  for ( i = 0; i < 9; i++) {           // we need 9 bytes
    data[i] = ds.read();
    Serial.print(data[i], HEX);
    Serial.print(" ");
  }
  Serial.print(" CRC=");
  Serial.print(OneWire::crc8(data, 8), HEX);
  Serial.println();



  // convert the data to actual temperature

  unsigned int raw = (data[1] << 8) | data[0];
  if (type_s) {
    raw = raw << 3; // 9 bit resolution default
    if (data[7] == 0x10) {
      // count remain gives full 12 bit resolution
      raw = (raw & 0xFFF0) + 12 - data[6];
    }
  } else {
    byte cfg = (data[4] & 0x60);
    if (cfg == 0x00) raw = raw << 3;  // 9 bit resolution, 93.75 ms
    else if (cfg == 0x20) raw = raw << 2; // 10 bit res, 187.5 ms
    else if (cfg == 0x40) raw = raw << 1; // 11 bit res, 375 ms
    // default is 12 bit resolution, 750 ms conversion time
  }
  celsius = (float)raw / 16.0;
//  Serial.print("  Temperature = ");
 // Serial.print(" Celsius, ");
//  Serial.print(celsius);
 // Serial.print(fahrenheit);
 // Serial.println(" Fahrenheit");

//_____________________________________________________________________________
_____
// BILDSCHIRMAUSGABE(LCD): Aktuelle Temperatur

 lcd.setCursor(0, 0);
 lcd.print(celsius);
 lcd.setCursor(5,0);
```

```
 lcd.print(F("C"));
// lcd.setCursor(7,0);lcd.print(key);

//_____________________________________________________________________________
_____
//ZEITANZEIGE

    lcd.setCursor (7, 0);
    DateTime now = RTC.now();


// >> Null fuer 1-09 Stunden
if (now.hour() < 10)

  {
    lcd.print('0');
    lcd.print(now.hour(), DEC);
  }

else

  {
    lcd.print(now.hour(), DEC);
  }
        lcd.print(':');

// >> Null fuer 1-09 Minuten
if (now.minute() < 10)

  {
    lcd.print('0');
    lcd.print(now.minute(), DEC);
  }

else

  {
    lcd.print(now.minute(), DEC);
  }
        lcd.print(':');

// >> Null fuer 1-09 Sekunden
if (now.second() < 10)

  {
    lcd.print('0');
    lcd.print(now.second(), DEC);
  }

else

  {
    lcd.print(now.second(), DEC);
  }

//_____________________________________________________________________________
// An-& Ausschalten der Beleuchtung mit Dimmer in Echtzeit
  //  DateTime now = RTC.now();

    minutenUhrzeit=now.hour() * 60 + now.minute(); //Umrechnung Stunden in
Minuten

//Serial.print("MinutenZeit ist ");
//Serial.println(minutenUhrzeit);
    //Serial.println(now.unixtime());
   // lcd.setCursor(0, 1);
   // lcd.print(now.unixtime());
```

```
if (minutenUhrzeit >= 480 && minutenUhrzeit < 840) //Wenn Uhrzeit zwischen 8 Uhr
und 14h00:

{
  digitalWrite(Lichtpin, HIGH);              //Licht anschalten
  binaerstatus_licht = 1;
   //Serial.println(" AN");
}

else if (minutenUhrzeit >= 960 && minutenUhrzeit < 1290) //Wenn Uhrzeit zwisch
16h00 und 21h30:

{
  digitalWrite(Lichtpin, HIGH);              //Licht anschalten
  binaerstatus_licht = 1;
   //Serial.println(" AN");
}

else

{
  digitalWrite(Lichtpin, LOW);              // Sonst Licht ausschalten
  binaerstatus_licht = 0;
   //Serial.println("aus");
}


//_____________________________________________________________________________
_____
//TASTENDRUCK

adc_taste_in = analogRead(0);    // Lese den Wert auf AnalogPin 0
taste = get_taste(adc_taste_in);  // Unterprogramm get_taste(): Ordne gelesenen
Analogwert einer Nummer bzw. Taste zu.


if (taste != alte_taste)   // Ermittle ob eine Taste gedr¸ckt wurde.

  {
    // ueberpr¸fung ob ermittelte Taste wirklich gedr¸ckt wurde.
    delay(50);  // warte eine gewisse Zeit und...
    adc_taste_in = analogRead(0);    // ...lese den Wert auf AnalogPin 0
    taste = get_taste(adc_taste_in);    // Unterprogramm get_taste(): Ordne
gelesenen Analogwert einer Nummer bzw. Taste zu.
    if (taste != alte_taste)

    {
      alte_taste = taste;
  // Starte Programmaktionen f¸r bestimmte Tasten
  switch(taste)

  {
        case 2: //Key right

        lcd.setCursor(0, 1);
        lcd.print(F("case 2"));
        delay(1000);
        lcd.clear();

       /*

        //taste_nachricht(); //Unterprogramm um entsprechend der gedr¸ckten
taste eine Nachricht auf das LCD-Display zu schicken.
        // Errechnung der Statistik
        temp_int = minuten_vektor[1]; //holt abgespeicherten Wert t-2 aus dem
Speicher
        temp = (float) temp_int;//Konvertierung float in int.
        temp = temp / 100; //Umrechnung in Zahl mit Kommastellen
```

```
        stat_minute = celsius - temp;//Ver‰nderung aktuell im Vergleich zu t-1

        //Ausgabe der Statistik auf das LCD-Display

        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(F("C-Diff t-2 Stde:"));
        lcd.setCursor(0, 1);
        lcd.print(stat_minute);
        delay(2000);
        lcd.clear();
        reset_aktionsschleife(); //Unterprogramm dass sicherstellt, dass die
aktionsschleife wieder aktiv werden kann, auch wenn die temperatur sich nicht
geˆndert hat.

        */

        break;

    case 1: //resetTaste___funktioniert nicht.
    //taste_nachricht(); //Unterprogramm um entsprechend der gedr¸ckten taste
eine Nachricht auf das LCD-Display zu schicken.
        // Errechnung der Statistik

        lcd.setCursor(0, 1);
        lcd.print(F("case 1"));
        delay(1000);
        lcd.clear();

        reset_aktionsschleife(); //Unterprogramm dass sicherstellt, dass die
aktionsschleife wieder aktiv werden kann, auch wenn die temperatur sich nicht
geˆndert hat

        break;

    case 0: //Key Up
        //taste_nachricht(); //Unterprogramm um entsprechend der gedr¸ckten
taste eine Nachricht auf das LCD-Display zu schicken.
        // Errechnung der Statistik
        //temp_int = minuten_vektor[0]; //holt abgespeicherten Wert t-1 aus dem
Speicher
        //temp = (float) temp_int;//Konvertierung float in int.
        //temp = temp / 100; //Umrechnung in Zahl mit Kommastellen
        //stat_minute = celsius - temp;//Ver‰nderung aktuell im Vergleich zu t-1

        //Ausgabe der Statistik auf das LCD-Display
       //lcd.clear();
        //lcd.setCursor(0, 0);
        //lcd.print(F("C-Diff t-1 Stde:");
        //lcd.setCursor(0, 1);
        //lcd.print(stat_minute);
        //delay(2000);
        //lcd.clear();

        //programm zum an/ausschalten der manuellen ¸berbr¸ckung, mit
entsprechendem Hinweis auf dem LCD-bildschirm.

        lcd.clear();
        switch(ueberbrueckung_luefter)
        {case 0:
        ueberbrueckung_luefter = 1;
        lcd.setCursor(0, 1);
        lcd.print(F("Stop Luefter: AN"));
        delay(2000);
        lcd.clear();

        break;
```

```
        case 1:
        ueberbrueckung_luefter = 0;
        lcd.setCursor(0, 1);

        lcd.print(F("StopLuefter: AUS"));
        delay(2000);
        lcd.clear();

        break;

        }

        reset_aktionsschleife(); //Unterprogramm dass sicherstellt, dass die
aktionsschleife wieder aktiv werden kann, auch wenn die temperatur sich nicht
geˆndert hat

        break;

    case 4: //Key down
        //taste_nachricht(); //Unterprogramm um entsprechend der gedr¸ckten
taste eine Nachricht auf das LCD-Display zu schicken.
        // Errechnung der Statistik
        //temp_int = minuten_vektor[2]; //holt abgespeicherten Wert t-3 aus dem
Speicher
        //temp = (float) temp_int;//Konvertierung float in int.
        //temp = temp / 100; //Umrechnung in Zahl mit Kommastellen
        //stat_minute = celsius - temp;//Ver‰nderung aktuell im Vergleich zu t-3

        //Ausgabe der Statistik auf das LCD-Display

        //lcd.clear();
        //lcd.setCursor(0, 0);
        //lcd.print(F("C-Diff t-3 Stde:");
        //lcd.setCursor(0, 1);
        //lcd.print(stat_minute);
        //delay(2000);
        //lcd.clear();
        //reset_aktionsschleife(); //Unterprogramm dass sicherstellt, dass die
aktionsschleife wieder aktiv werden kann, auch wenn die temperatur sich nicht
geˆndert hat
        //break;


        //programm zum an/ausschalten der manuellen ¸berbr¸ckung_heizen, mit
entsprechendem Hinweis auf dem LCD-bildschirm.

        lcd.clear();
        switch(ueberbrueckung_heizen)
        {case 0:

        ueberbrueckung_heizen = 1;
        lcd.setCursor(0, 1);
        lcd.print(F("Stop Heizen: AN "));
        delay(2000);
        lcd.clear();
        break;

        case 1:
        ueberbrueckung_heizen = 0;
        lcd.setCursor(0, 1);
        lcd.print(F("Stop Heizen: AUS"));
        delay(2000);
        lcd.clear();
        break;

        }

        reset_aktionsschleife(); //Unterprogramm dass sicherstellt, dass die
```

```
aktionsschleife wieder aktiv werden kann, auch wenn die temperatur sich nicht
ge^ndert hat

        break;


    case 3: //Key left

   lcd.print(F("Links"));  //>>> BLAU

         /*lcd.clear();
         lcd.setCursor(2, 1);
         lcd.print(F("Es ist ");
         lcd.print(now.hour(), DEC);
         lcd.print(':');
         lcd.print(now.minute(), DEC);
        */

         delay(1000);
         lcd.clear();

       /*
       //taste_nachricht(); //Unterprogramm um entsprechend der gedr¸ckten taste
eine Nachricht auf das LCD-Display zu schicken.
       // Errechnung der Statistik

        temp_int = minuten_vektor[3]; //holt abgespeicherten Wert t-4 aus dem
Speicher
        temp = (float) temp_int;//Konvertierung float in int.
        temp = temp / 100; //Umrechnung in Zahl mit Kommastellen
        stat_minute = celsius - temp;//Ver‰nderung aktuell im Vergleich zu t-4

        //Ausgabe der Statistik auf das LCD-Display

        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(F("C-Diff t-4 Stde:"));
        lcd.setCursor(0, 1);
        lcd.print(stat_minute);
        delay(2000);
        lcd.clear();
        */
        reset_aktionsschleife(); //Unterprogramm dass sicherstellt, dass die
aktionsschleife wieder aktiv werden kann, auch wenn die temperatur sich nicht
ge^ndert hat


        break;

      }

  }

 }



//_____________________________________________________________________________
_____
// ZUWEISUNG VON Temp-INTERVALLEN ZU BESTIMMTEN FALLNUMMERN.


  // Starte hierzu das Unterprogramm get_key() siehe weiter unten.
  key = get_key(celsius);

//_____________________________________________________________________________
_____
// BILDSCHIRMAUSGABE(SERIELL): BEWERTUNG GEMfl DER DEM OHM-INTERVALL
```

```
ZUGEORDNETEN FALLNUMMER: Beispiel: "Viel zu heiß!"

  //Serial.print(msgs[key]);
  //Serial.println();

//_____________________________________________________________________________
_____
// ZUORDNEN BESTIMMTER AKTIONEN ZU DEN DER Temp-INTERVALLEN ZUGEORDNETEN
FALLNUMMERN

if (herunterkuehlen == 1) //Kriterium der Abschaltung der Herunterkühlung: muss
eingeschaltet

{

 if (ueberbrueckung_luefter == 1) // UND NOTABSCHALTUNG ist aktiviert
 {digitalWrite(digitalPin[0], LOW); //Schalte die Lampe des Falls 0 aus.
  digitalWrite(digitalPin[3], HIGH); // Schalte Relay des Falls 0 aus. (=0 +3).
Relay wird deaktiviert, wenn HIGH-Signal

 herunterkuehlen = 0;

 }


 if (celsius < temp_krit_kuehlen)// ODER kritischer temperaturwert wurde
unterschritten, !!!weil messungen in temp, muss der kritische wert überschritten
sein!!!

 {

 digitalWrite(digitalPin[0], LOW); ////Schalte die Lampe des Falls 0 aus.
 digitalWrite(digitalPin[3], HIGH);// Schalte Relay des Falls 0 aus. (=0 +3).
Relay wird deaktiviert, wenn HIGH-Signal

 herunterkuehlen = 0;

  }

 }


if (hochheizen == 1) //Kriterium der Abschaltung des Hochheizen: muss
eingeschaltet

{

 if (ueberbrueckung_heizen == 1) // UND NOTABSCHALTUNG ist aktiviert

  {
   digitalWrite(digitalPin[2], LOW); //Schalte die Lampe des Falls 2 aus.
   digitalWrite(digitalPin[5], HIGH); // Schalte Relay des Falls 2 aus. (= 2
+3). Relay wird deaktiviert, wenn HIGH-Signal

 hochheizen = 0;

  }

 if (celsius >  temp_krit_heizen)// ODER kritischer temperaturwert wurde
überschritten, !!!weil messungen in temp, muss der kritische wert unterschritten
sein!!!

  {

 digitalWrite(digitalPin[2], LOW);//Schalte die Lampe des Falls 2 aus.
 digitalWrite(digitalPin[5], HIGH); // Schalte Relay des Falls 2 aus. (= 2 +3).
Relay wird deaktiviert, wenn HIGH-Signal
```

```
 hochheizen = 0;

  }

 }


if (key != oldkey) // Stellt sicher dass nur dann ein Programm gestartet wird,
wenn sich der gemessene Wertebereich ge‰ndert hat.

{


 if (oldkey == 1) // stellt sicher das die Rote Lampe "sehr heiß" nicht
ungewollt abgeschaltet wird.

  {

    digitalWrite(digitalPin[oldkey], LOW); //Schalte die Lampe der alten
Fallnummer aus.

  }

//_____________________________________________________________________
// Starte Programmaktionen f¸r bestimmte Temperaturbereiche
  switch(key)

  {

    case 0:

    // Fall: "Viel zu warm", Blinke Rot.

    if(ueberbrueckung_luefter == 0) //wenn manuelle ‹berbr¸ckung ausgeschaltet,
dann schalte lampe automatisch an.

    {

    //Serial.print("FALL 0: Blinke Rot");
    digitalWrite(digitalPin[key], HIGH);

    relay = key + 3; //ergibt die relayNr. im Pinvektor

    //Serial.println(digitalPin[relay]);
    digitalWrite(digitalPin[relay], LOW); //damit Relay aktiv wird, muss ein
LOW-Signal gesendet werden.

    herunterkuehlen = 1; //Aktiviert Pr¸fvariable zum herunterk¸hlen

    }


  lcd_anzeige_notstop_an();// Unterprogramm zur Anzeige der Temperaturbewertung
oder hinweis, dass notstopp eingeschaltet ist



    LCD_Bewertung(); //Unterprogramm, dass die Bewertung der Temperatur auf das
LCD-Display schreibt.

    oldkey = key;

        break;


    case 1: // Fall: "Temperatur OK", Blinke Gr¸n.

    //Serial.print("FALL 1, Blinke Gr¸n");
```

```
    digitalWrite(digitalPin[key], HIGH);

    lcd_anzeige_notstop_an();// Unterprogramm zur Anzeige der
Temperaturbewertung oder hinweis, dass notstopp eingeschaltet ist


    //LCD_Bewertung(); //Unterprogramm, dass die Bewertung der Temperatur auf
das LCD-Display schreibt.

    oldkey = key;

    break;


    case 2: // Fall: "Zu Kalt", Blinke Gelb.

    //Serial.print("FALL 2, Blinke Gelb.");

     if(ueberbrueckung_heizen == 0) //wenn manuelle ‹berbr¸ckung ausgeschaltet,
dann schalte lampe automatisch an.

    {

    //Serial.print("FALL 0: Blinke Gelb");

    digitalWrite(digitalPin[key], HIGH);

    relay = key + 3; //ergibt die relayNr. im Pinvektor

    digitalWrite(digitalPin[relay], LOW); //damit Relay aktiv wird, muss ein
LOW-Signal gesendet werden.

    hochheizen = 1; //Aktiviert Pr¸fvariable zum herunterk¸hlen

    }


    lcd_anzeige_notstop_an();// Unterprogramm zur Anzeige der
Temperaturbewertung oder hinweis, dass notstopp eingeschaltet ist


    //LCD_Bewertung(); //Unterprogramm, dass die Bewertung der Temperatur auf
das LCD-Display schreibt.

    oldkey = key;

    break;


    //case 3: // Fall: "In Ordnung", Blinke Gr¸n.
    //Serial.print("FALL 3, Blinke Gr¸n.");
    //digitalWrite(digitalPin[key], HIGH);

    //lcd_anzeige_notstop_an();// Unterprogramm zur Anzeige der
Temperaturbewertung oder hinweis, dass notstopp eingeschaltet ist
    //LCD_Bewertung(); //Unterprogramm, dass die Bewertung der Temperatur auf
das LCD-Display schreibt.
    //oldkey = key;

    //break;


    //case 4: // Fall: "Kalt aber gr¸n", Blinke Gelb.
    // Serial.print("FALL 4, Blinke Gelb");
    //digitalWrite(digitalPin[key], HIGH);

    //lcd_anzeige_notstop_an();// Unterprogramm zur Anzeige der
Temperaturbewertung oder hinweis, dass notstopp eingeschaltet ist
```

```
    //LCD_Bewertung(); //Unterprogramm, dass die Bewertung der Temperatur auf
das LCD-Display schreibt.
    //oldkey = key;

    //break;


    //case 5: // Fall: "Viel zu kalt", Blinke Rot.
    //Serial.print("FALL 5, Blinke Rot");
    //digitalWrite(digitalPin[key], HIGH);

   //lcd_anzeige_notstop_an();// Unterprogramm zur Anzeige der
Temperaturbewertung oder hinweis, dass notstopp eingeschaltet ist

   //LCD_Bewertung(); //Unterprogramm, dass die Bewertung der Temperatur auf das
LCD-Display schreibt.
    //oldkey = key;
    //break;

  }

}



//_____________________________________________________________________________
_____
//Übertragung der Messwerte via Internet

 if (now.unixtime() - vorhin >= uebertragungsfrequenz )//Misst ob genug Zeit
vergangen ist, um Werte erneut ins Internet zu senden (Unixtime ist eine
Fortlaufende Zahl die Gesamtzeit seit 1. Jan. 1970 in sekunden angibt.

  {
  //Serial.println(now.unixtime() - vorhin);
    vorhin = now.unixtime(); //Speichert übertragungszeitpkt ab.


  //Umrechnung, damit die Temperatur mit Komma via der Variable "temperaturwert
übermittelt werden kann.
  int zahl0=(int) celsius;

  zahl0 *= 100;

  float zahl1= celsius *100;

  int zahl2 = (int) zahl1;

  int nachkommastellen = zahl2 - zahl0;

  int vorkommastellen = (int) celsius;

  String temperaturwert; //Erstellt Zeichenkette der Temperatur mit
Kommastellen, die später in die Übertragungszeichenkette eingefügt wird.

  temperaturwert+="";//löscht letzten Inhalt des Strings
  temperaturwert +=vorkommastellen;

  temperaturwert += ",";

  temperaturwert +=nachkommastellen;



  //Ermittlung der Zustände von Lüfter und Heizstab
  binaerzustand_luefter = zustandsermittlung(herunterkuehlen,
ueberbrueckung_luefter);
```

```
  binaerzustand_heizstab = zustandsermittlung(hochheizen,
ueberbrueckung_heizen);

  // Übertragungsstring bzw. -Kette "Data", wird im folgenden mit Informationen
beschrieben, die übermittelt werden sollen.
  String data;

  data+="";//löscht letzten Inhalt des Strings

  data+="entry.0.single="; //Temperatur
  data+=temperaturwert;

  data+="&entry.1.single="; //Lichtstatus
  //data+=an_aus[binaerstatus_licht]; //Überträgt den Status in Worten
  data+=binaerstatus_licht; //Überträgt den Status im Binärcode

  data+="&entry.4.single=";// Heizungsstatus
  //data+=an_aus[binaerzustand_heizstab]; //Überträgt den Status in Worten
  data+=zustandsermittlung(hochheizen, ueberbrueckung_heizen); //Überträgt den
Status im Binärcode

  data+="&entry.9.single=";// Lüfterstatus
  //data+=an_aus[binaerzustand_luefter]; //Überträgt den Status in Worten
  data+=zustandsermittlung(herunterkuehlen, ueberbrueckung_luefter); //Überträgt
den Status im Binärcode

  data+="&submit=Submit";


//Eigentlich Übertragung des Datastrings der die Werte enthält:
  if (client.connect(server, 80)) //Wenn Verbinding zum Googleserver via Port 80
besteht

  {

   // Serial.println("connected");


    client.print("POST /formResponse?formkey=");
    client.print(formkey); //formkey ist die ID der Googletabelle bzw. des
Googleformblattes.

    client.println("&ifq HTTP/1.1");

    client.println("Host: spreadsheets.google.com");

    client.println("Content-Type: application/x-www-form-urlencoded");

    client.println("Connection: close");

    client.print("Content-Length: ");

    client.println(data.length());

    client.println();

    client.print(data);//Übertragung der Zeichenkette mit den Werten.

    delay(1000);//Wichtig  um der Übertragung Zeit zu geben!!!

    client.println();

 //delay(1000);


/*
   Serial.print("POST /formResponse?formkey=");
    Serial.print(formkey);
```

```
    Serial.println("&ifq HTTP/1.1");
    Serial.println("Host: spreadsheets.google.com");
    Serial.println("Content-Type: application/x-www-form-urlencoded");
    Serial.println("Connection: close");
    Serial.print("Content-Length: ");
    Serial.println(data.length());
    Serial.println();
    Serial.print(data);
    Serial.println();
 */

}

  if (!client.connected()) //Wenn keine Verbindung zustande gekommen ist, dann
Abbruch

  {

    //Serial.println();
    //Serial.println("disconnecting.");

    client.stop();

  }

}

}

//_____________________________________________________________________________
_____
//_____________________________________________________________________________
_____
//_____________________________________________________________________________
_____

//DEFINITION VON VERWENDETEN UNTERPROGRAMME

//_____________________________________________________________________________
_____
// UNTERPROGRAMM ZUR ZUORDNUNG VON temp-BEREICHEN ZU BESTIMMTEN FALLNUMMERN

int get_key(float tempw)

{

    int k;
    for (k = 0; k < NUM_KEYS; k++)

    {

      if (tempw < adc_key_val[k])

 {
            return k;

        }

   }

    if (k >= NUM_KEYS)k = 2;  // Groesser als 27c (kleiner als 24 C) erhaelt die
Fallnummer 2

    return k;

}

//_____________________________________________________________________________
```

```
_____
// UNTERPROGRAMM ZUM ANZEIGEN DER BEWERTUNG AUF DEM LCD-DISPLAY


void LCD_Bewertung()

{
    lcd.setCursor(0, 1);
    lcd.print(msgs[key]);
}

//_____________________________________________________________________________
_____
// UNTERPROGRAMM ZUR ZUORDNUNG VON MESSWERTEN ZU BESTIMMTEN TASTENNUMMERN; siehe
Unterprogramm "ZUORDNUNG VON OHM-BEREICHEN ZU BESTIMMTEN FALLNUMMERN"

int get_taste(unsigned int input2)

{

    int k2;
    for (k2 = 0; k2 < NUM_KEYS_TASTEN; k2++)

    {

      if (input2 < adc_taste_val[k2])

 {

            return k2;

        }

   }

    if (k2 >= NUM_KEYS_TASTEN)k2 = -1;  // Wenn keine taste gedr¸ckt wird, dann
nimmt stellt dies hier sicher, dass die selbe taste mehrmals gedr¸ckt werden
darf.
    return k2;

}

//_____________________________________________________________________________
_____
// UNTERPROGRAMM ZUM ANZEIGEN EINER TASTENNACHTICHT AUF DEM LCD-DISPLAY

  void taste_nachricht()

  {
      lcd.clear();
      lcd.setCursor(0, 1);
      lcd.print(msgs_taste[taste]);
      delay(2000);
      lcd.clear();
  }

//_____________________________________________________________________________
_____
// UNTERPROGRAMM UM NACH DEM DRUCK EINER TASTE WIEDER IN DIE AKTIONSSCHLEIFE ZU
KOMMEN

  void reset_aktionsschleife()

  {

    digitalWrite(digitalPin[oldkey], LOW); //stellt sicher dass nach dem reset
des Pr¸fwertes der Aktionsschleife (oldkey)  der digitalPin nicht einfriert.
```

```
    oldkey = 6; //reset des Pr¸fwertes; stellt sicher, dass die Aktionsschleife
aktiv wird, auch wenn sich der temperaturbereich nicht ge‰ndert hat.

  }

//_____________________________________________________________________________
_____
// UNTERPROGRAMM ZUR PERMANENTEN ANZEIGE "NOTSTOP: AN" AUF DEM LCD-DISPLAY, wenn
notstop aus, dann anzeige der normalen bewertung.

void lcd_anzeige_notstop_an()

  {

if(ueberbrueckung_luefter == 1 && ueberbrueckung_heizen == 1)

{
    lcd.setCursor(0, 1);
    lcd.print(F("Alle Stops: AN  "));
}

if(ueberbrueckung_luefter == 1 && ueberbrueckung_heizen == 0)

{
    lcd.setCursor(0, 1);
    lcd.print(F("Stop Luefter: AN"));
}

if(ueberbrueckung_luefter == 0 && ueberbrueckung_heizen == 1)

{
    lcd.setCursor(0, 1);
    lcd.print(F("Stop Heizen: AN "));
}

if(ueberbrueckung_luefter == 0 && ueberbrueckung_heizen == 0)

{
  LCD_Bewertung();//Unterprogramm zum Anzeigen der Temperaturbewertung auf dem
LCD-Display.
}


  }

//_____________________________________________________________________________
_____
// UNTERPROGRAMM ZUr ERMITTLUNG DES AKTIVITÄTSZUSTANDES DES HEIZSTABES ODER
LÜFTERS

int zustandsermittlung(int normal, int notaus)
{

  if (normal == 1 && notaus == 0) {return 1; }
  else {return 0;}
}
```