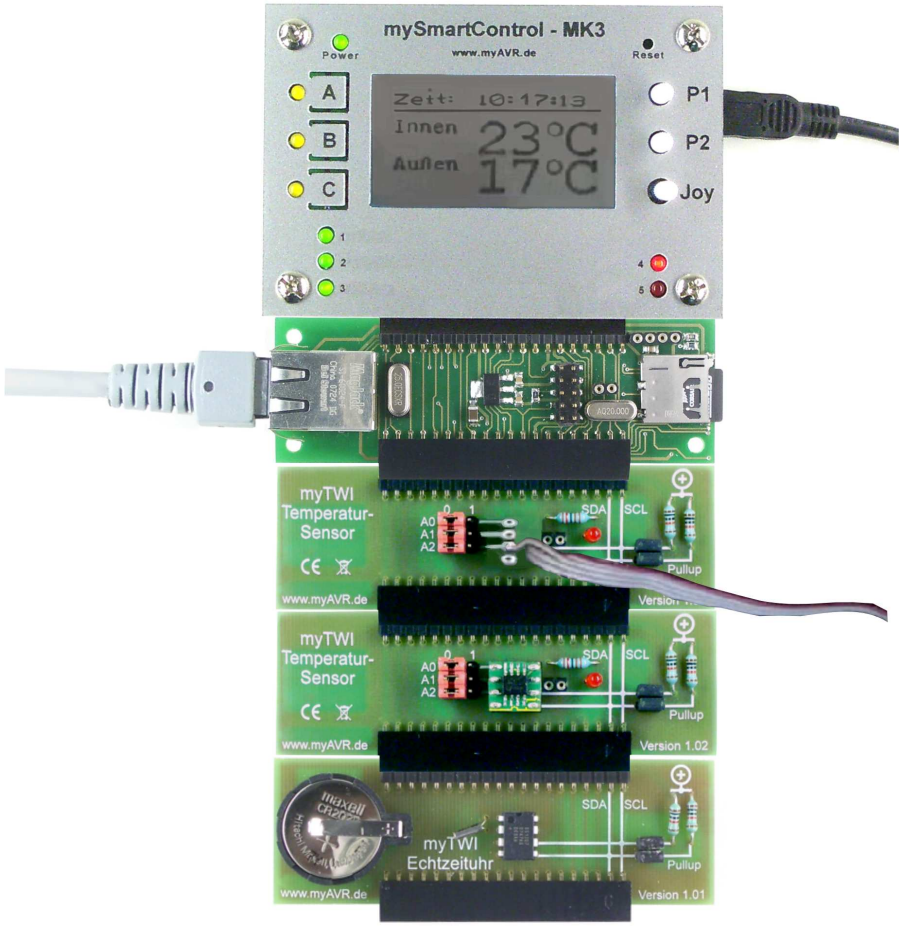


myWeather station

Inhalt

Promenadenring 8
02708 Löbau
Deutschland

www.myAVR.de
support@myavr.de

Tel: ++49 (0) 358 470 222
Fax: ++49 (0) 358 470 233

1 Einleitung

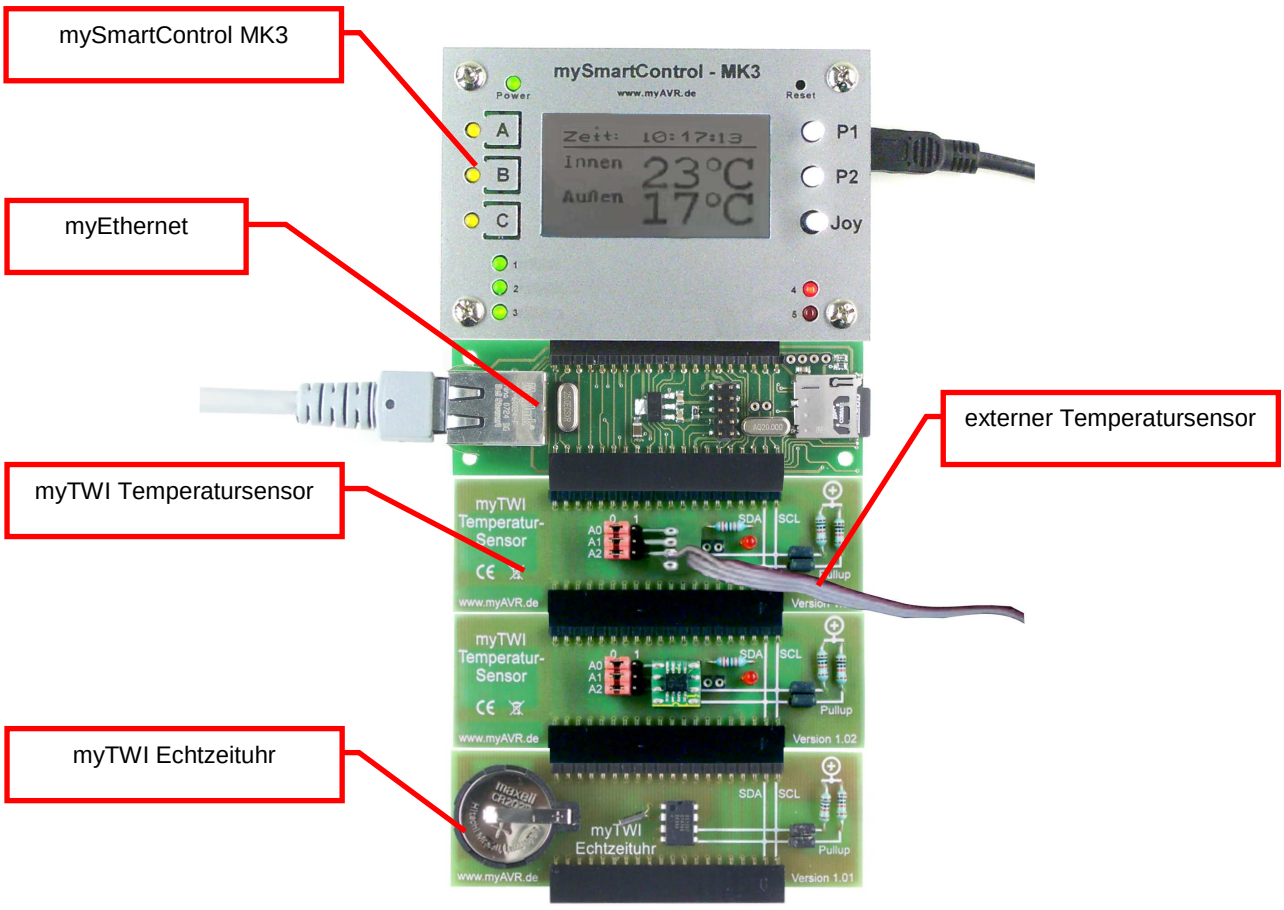
Das Projekt myWeather station zeigt, wie sich Außen- und Innen-Temperaturen erfassen und optisch über ein LCD anzeigen lassen. Für die Ermittlung der Außentemperaturen gibt es einen externen Temperatursensor, der sich z. B. außen auf dem Fensterbrett befinden kann. Alle gesammelten Daten werden zur gesamten Laufzeit in einer Datenbank des www-Servers erfasst. Aus diesen Werten wird dann auch ein Diagramm zur Darstellung des Wetterverlaufs erstellt.

2 Eigenschaften

- Eingabegeräte: 3 Taster, Joystick, 2 Potentiometer
- Ausgabegeräte: 8 LEDs, Speaker, Graphik LCD 64x128 mit Hintergrundbeleuchtung
- integrierte Spannungsreglung
- Reset-Taster
- 10 MegaBit Ethernet mit ENC28J60 von Microchip
- ATmega644P 20MHz mit vorinstalliertem Webserver
- MicroSD-Kartenhalter
- TWI (I²C) Temperatursensor Modul
- TWI (I²C) Real Time Clock Modul

3 Entwicklungsumgebung

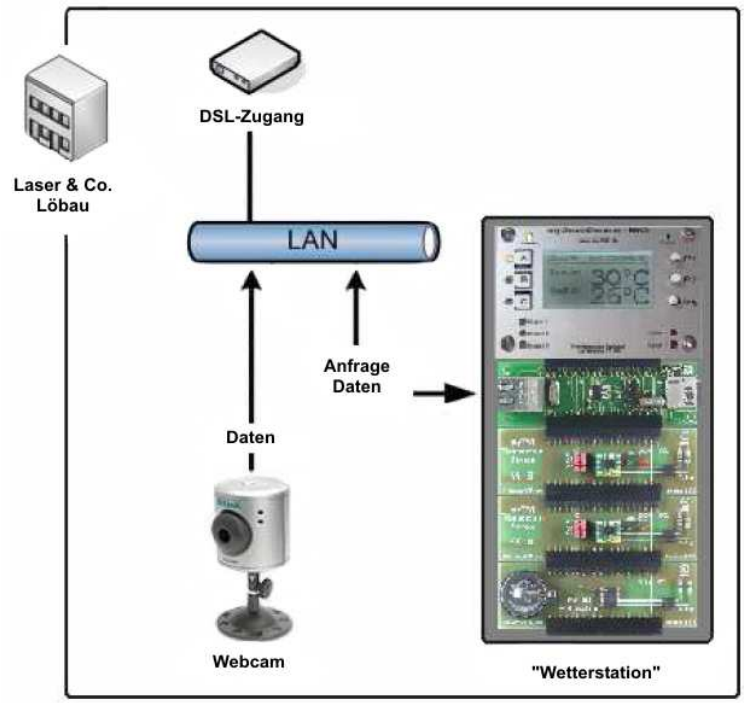
- Anschluss: Mini-USB-Port
- Programmiersoftware: SiSy AVR, myAVR Workpad PLUS



4 Grundlagen

4.1 Allgemein

Um die Temperaturwerte auch außerhalb der Wetterstation verfolgen zu können, wurde eine Webcam mit der Wetterstation kombiniert. Sowohl die Wetterstation als auch die Webcam besitzen einen LAN-Anschluss. Somit können Sie ohne Zusatzgeräte Daten empfangen bzw. senden.



Zusammengesetzt ist die Wetterstation aus 5 Modulen:

- mySmartControl MK3
- myEthernet
- myTWI Temperatursensor (Innentemperatur)
- myTWI Temperatursensor (Außentemperatur)
- myTWI Echtzeituhr

Das Gesamtsystem teilt sich in zwei Untersysteme auf:

1. myEthernet mit Webserverfunktion per LAN-Schnittstelle und
2. mySmartControl MK3 mit Hauptaufgabe als Ausgabeeinheit zur Visualisierung von Daten über ein LCD

Bei Anforderung lädt der Webserver seine Seiten von der MikroSD-Karte und ersetzt verschiedene Platzhalter mit Messwerten, um sie dem Anwender optisch darzustellen. Die Ausgabeeinheit greift auf die gleiche Quelle wie das myEthernet zu und ermittelt autonom alle anzuzeigenden Daten. Beide Untersysteme greifen damit auf denselben TWI –Bus zu.

Datenquellen sind die beiden Temperatursensoren und die Echtzeituhr. Als Kommunikationsprotokoll wird TWI verwendet. Die Daten der Sensoren liegen als Halbgrade vor. Für die Codierung der Daten verwendet die Uhr das BCD-Format.

4.2 myEthernet Webserver

Für die Verbindung mit dem TWI-Bus muss die Konfiguration auf TWI umgestellt sein. Bis zu 8 Sensoren sind anschließbar. Hier werden nur 2 benötigt. Die beiden Sensoren entsprechen dem Platzhalter v200 und v202 im Webseitenquelltext. Mit dem Formatierungsparameter „lm75“ wird der Halbwert in Grad Celsius umgerechnet ausgegeben.

Quelltext mit Platzhaltern:

```
<html>
<head>
</heady>
<body>
      <h3>aktuelle Temperaturen:</h3>
      außen: °v200~lm75° &deg;C<br><br>
      innen: °v202~lm75° &deg;C
</body>
</html>
```

myEthernet Website:

aktuelle Temperaturen:

außen: 25,0 °C
innen: 24,0 °C

Für die Kontrolle des TWI-Busses ist eine gesonderte Anpassung der Firmware notwendig, da es zwei TWI-Master an einem Bus gibt. Bei jedem Zugriff wird überprüft, ob ein Fehlercode existiert und entschieden, ob eine Neuinitialisierung notwendig ist.

4.3 mySmartControl MK3 Software

Auf dem LCD des mySmartControl MK3 soll die Innen- und Außentemperatur sowie die Uhrzeit angezeigt werden. Mit einer Webcam wird ein Bild erzeugt, auf dem alle wichtigen Informationen lesbar sein müssen.

Auf einem Mikrocontroller läuft ein Programm in einer Endlosschleife. Faktoren wie Interrupts oder zeitgesteuerte Ereignisse führen zu Sprüngen in Unterprogramme bzw. Funktionen. So kann eine separate Ausführung von Anweisungen außerhalb der Hauptschleife erfolgen.

Zur Programmierung des AVR ATmega2560 auf dem MK3 stehen Bibliotheken bzw. Pakete innerhalb des Casetools SiSy zur Verfügung. In der Steuersoftware wurden die in der Tabelle aufgelisteten Pakete integriert:

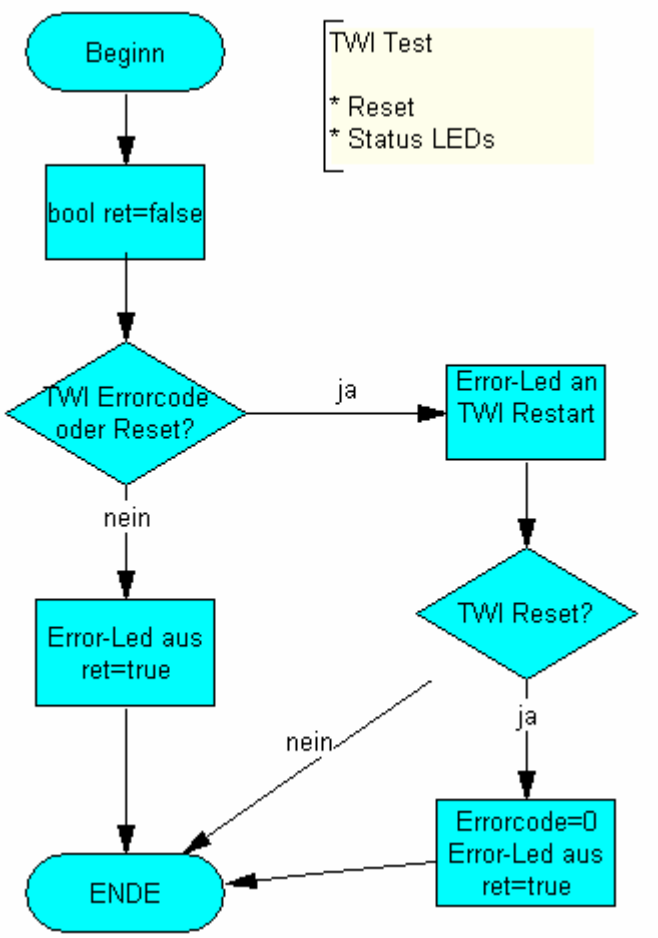
myAVR_Timer	• Schnittstelle zu den 8/16 Bit Timern des ATmega2560 • 2 8Bit Timer • 4 16Bit Timer
myAVR_DigitalOut	• Schnittstelle für digitale E/A des ATmega2560 • LEDs, Buttons usw.
myAVR_MK3Lcd	• Schnittstelle zur Ausgabe auf dem LCD • Text, monochrome Bitmaps, geometrische Formen
myAVR_TWI_Devices	• Schnittstelle zum TWI-Bus über ATmega2560 • Echtzeituhr (DS1307), EEPROM, Temperatursensoren (LM75)

Mithilfe dieser Schnittstellen können zeitgesteuerte Abfragen des TWI-Busses erfolgen. Ein zu kurzes Intervall führte nach ersten Tests zu Ausfällen bzw. undefinierten Antworten seitens TWI. Das Einlesen aller 25 Sekunden für die Temperaturen und alle 5 Minuten für die aktuelle Zeit der Echtzeituhr erwies sich für dieses Vorhaben als ausreichend. Die Uhr läuft über den internen Timer zwischen den Aktualisierungen weiter und synchronisiert sich mit jeder Periode.

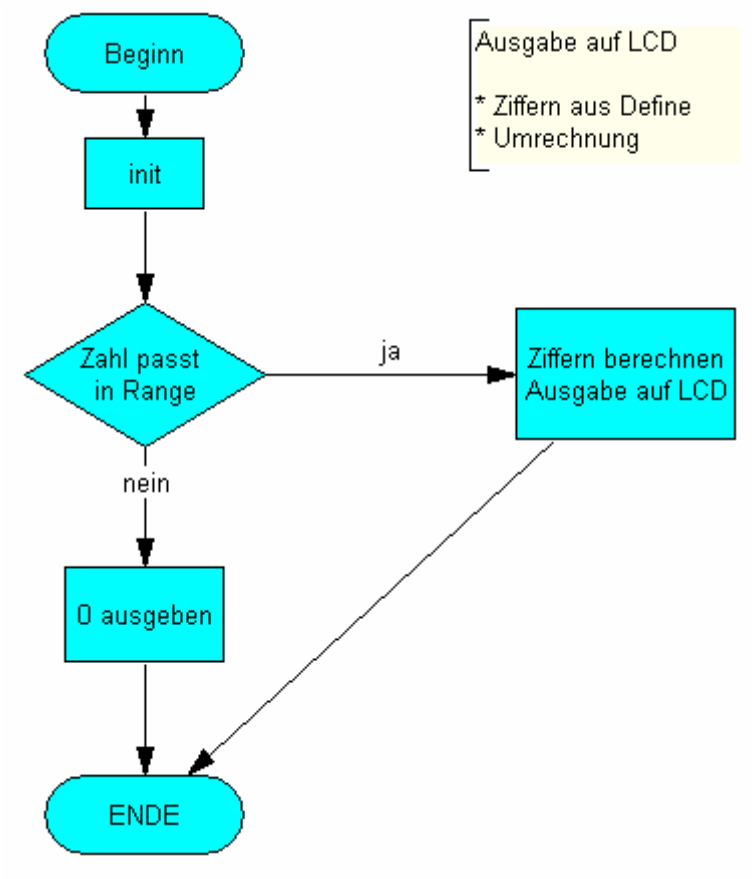
Mit monochromen Bitmaps, die als Struktur beim Übersetzen des Quellcodes implementiert wird, wird die gute Lesbarkeit der Ziffern realisiert. Dadurch kann eine Zahl aus einzelnen grafischen Ziffern dargestellt werden. In der gleichen Art ist das Hintergrundbild implementiert, das während der Laufzeit nicht mit neuen Texten oder Grafiken überzeichnet wird.

Wie beim myEthernet ist eine Überprüfung des Bus-Status notwendig. Für diesen Fall erfolgt eine minimale Fehlerbehandlung mit Anzeige über LEDs sowie Reinitialisierung des TWI.

5 Ablaufplan TWI Test

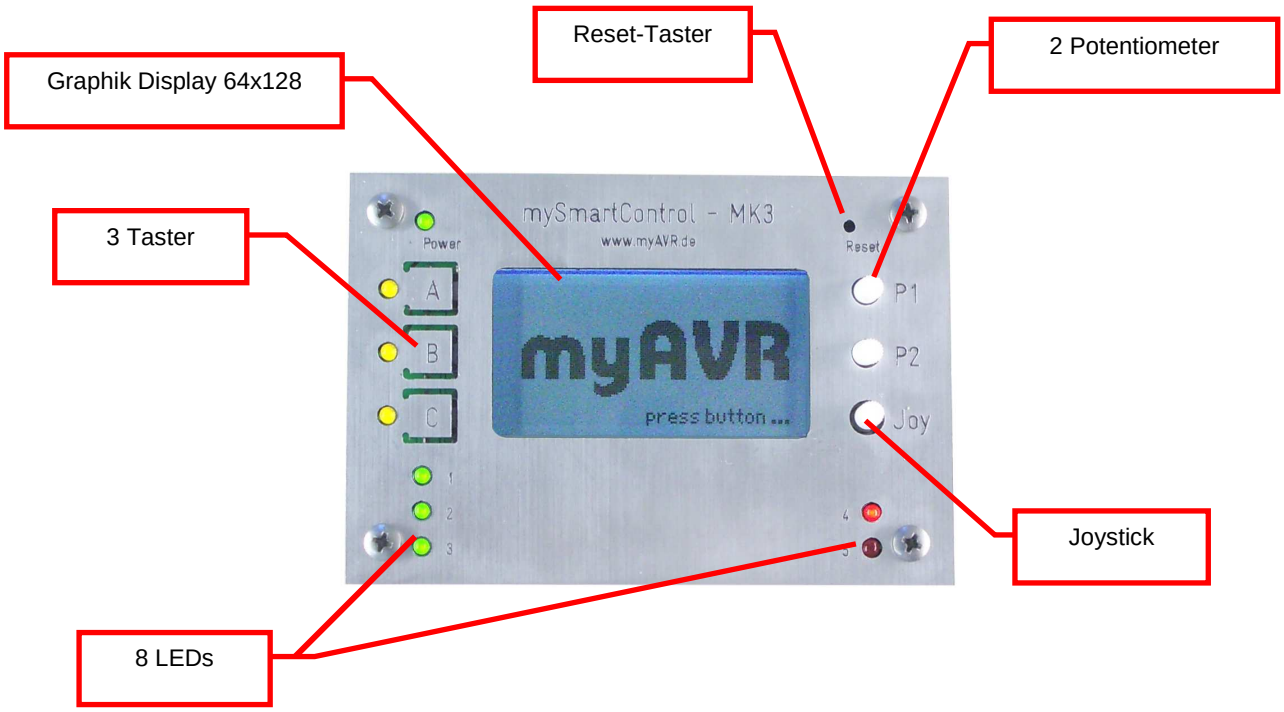


6 Ablaufplan Ausgabe auf LCD

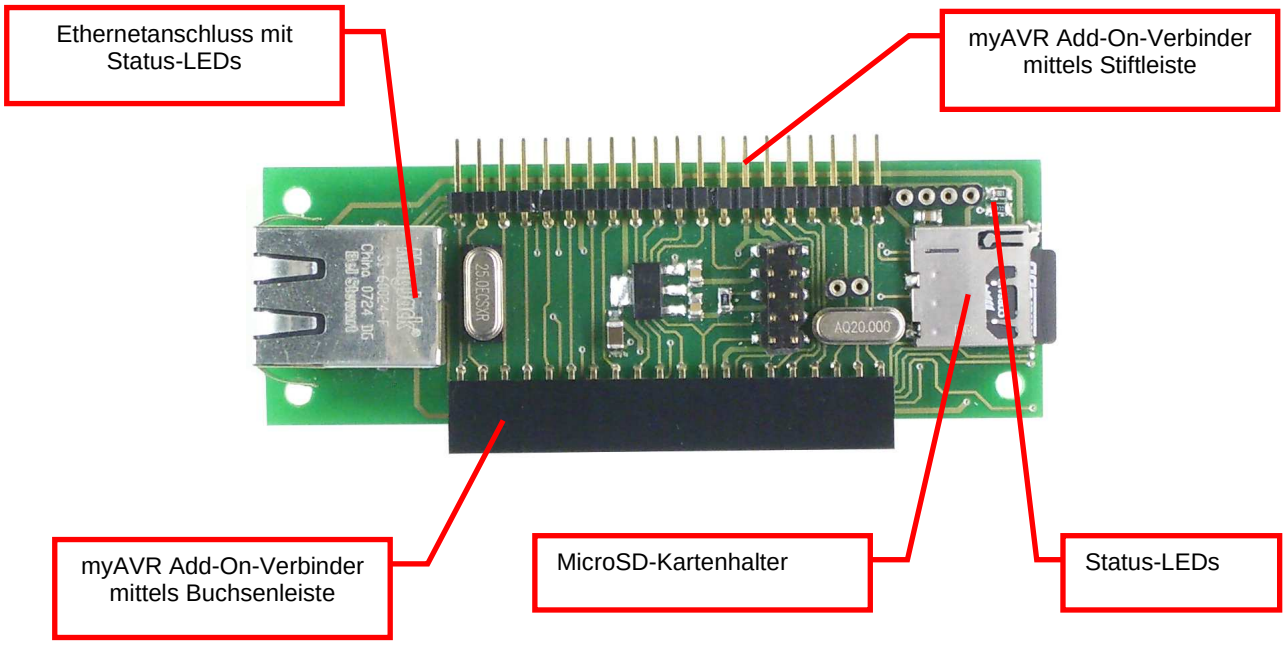


7 Hardware myWeather station

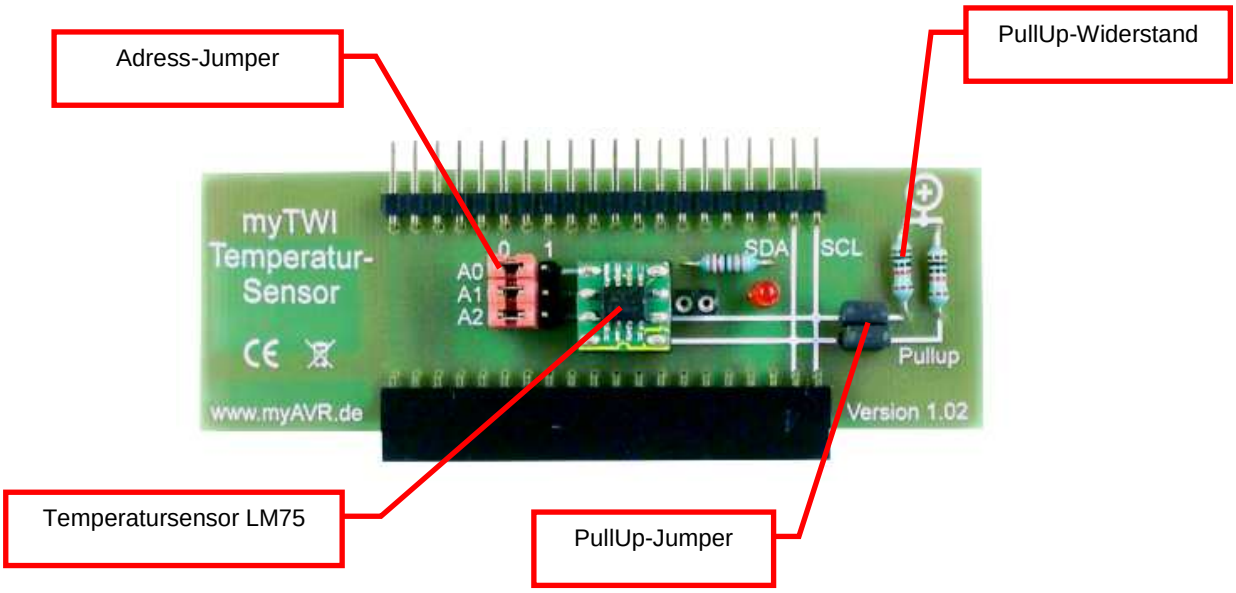
7.1 Hardwareüberblick



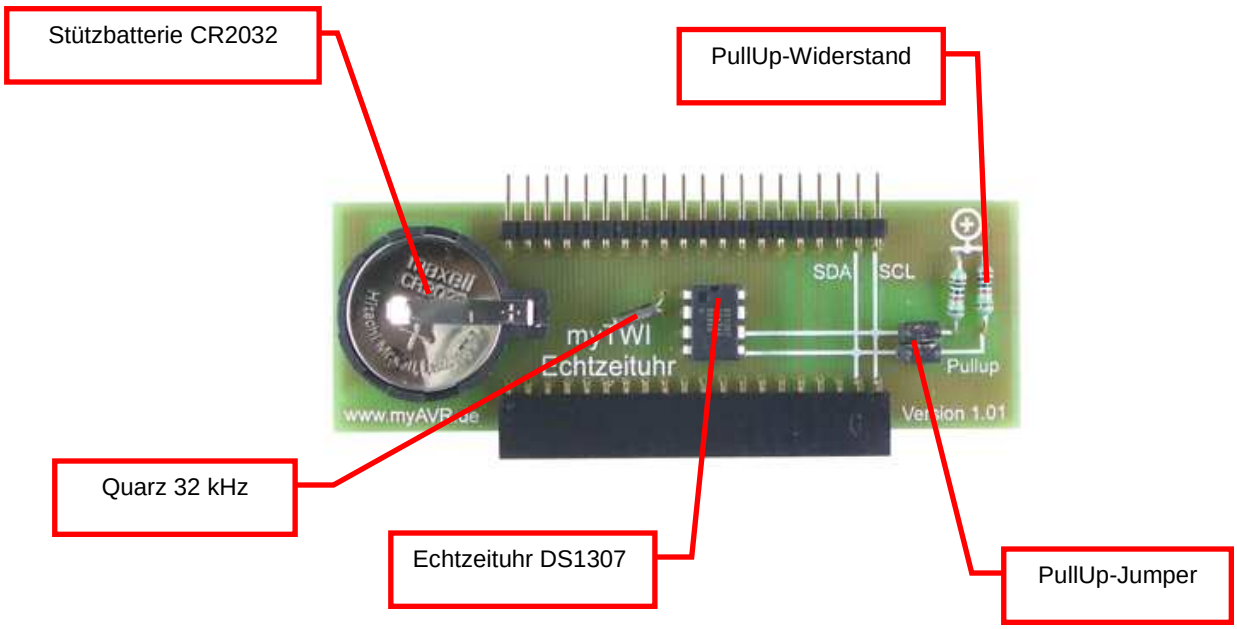
mySmartControl MK3



myEthernet



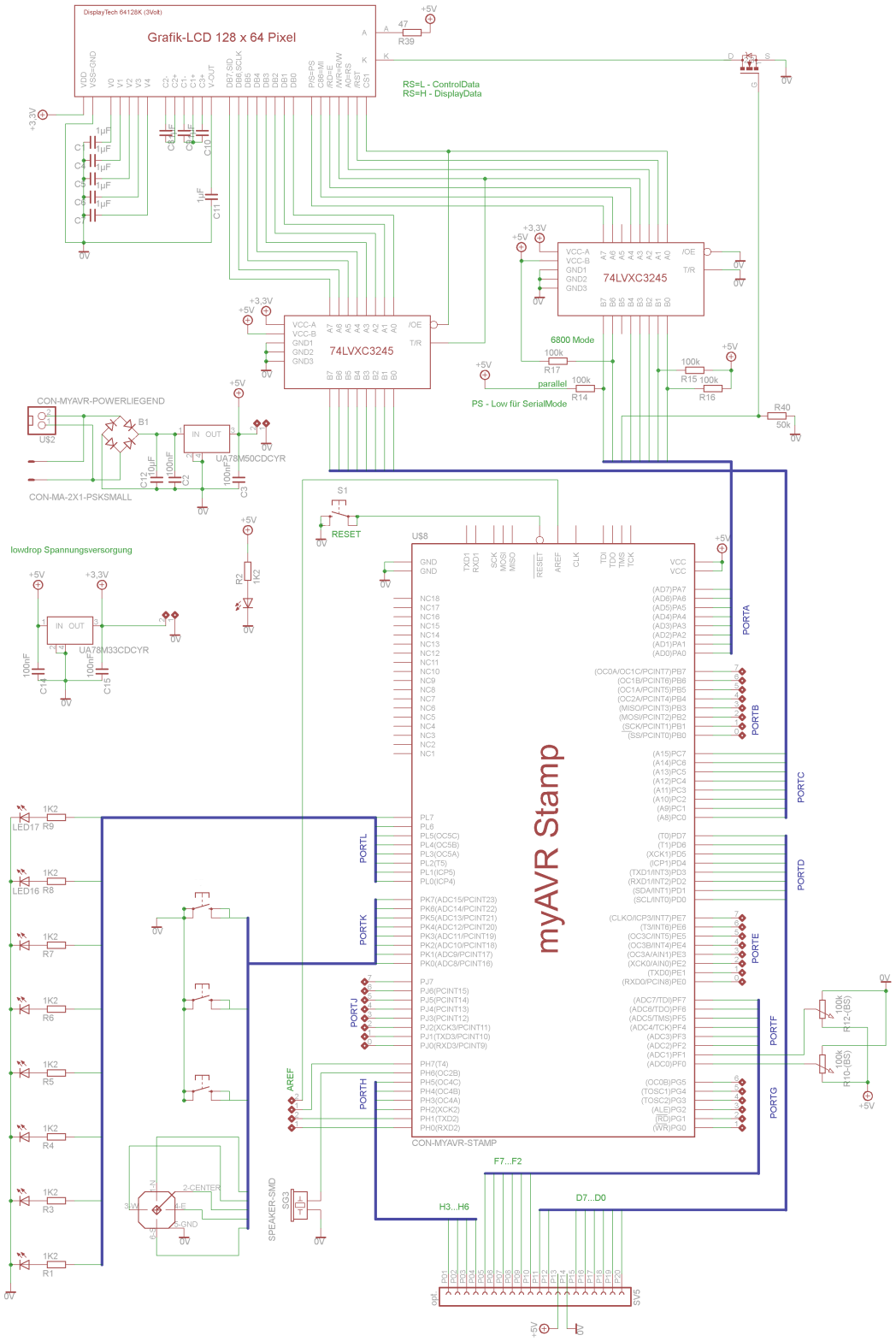
myTWI Temperatursensor



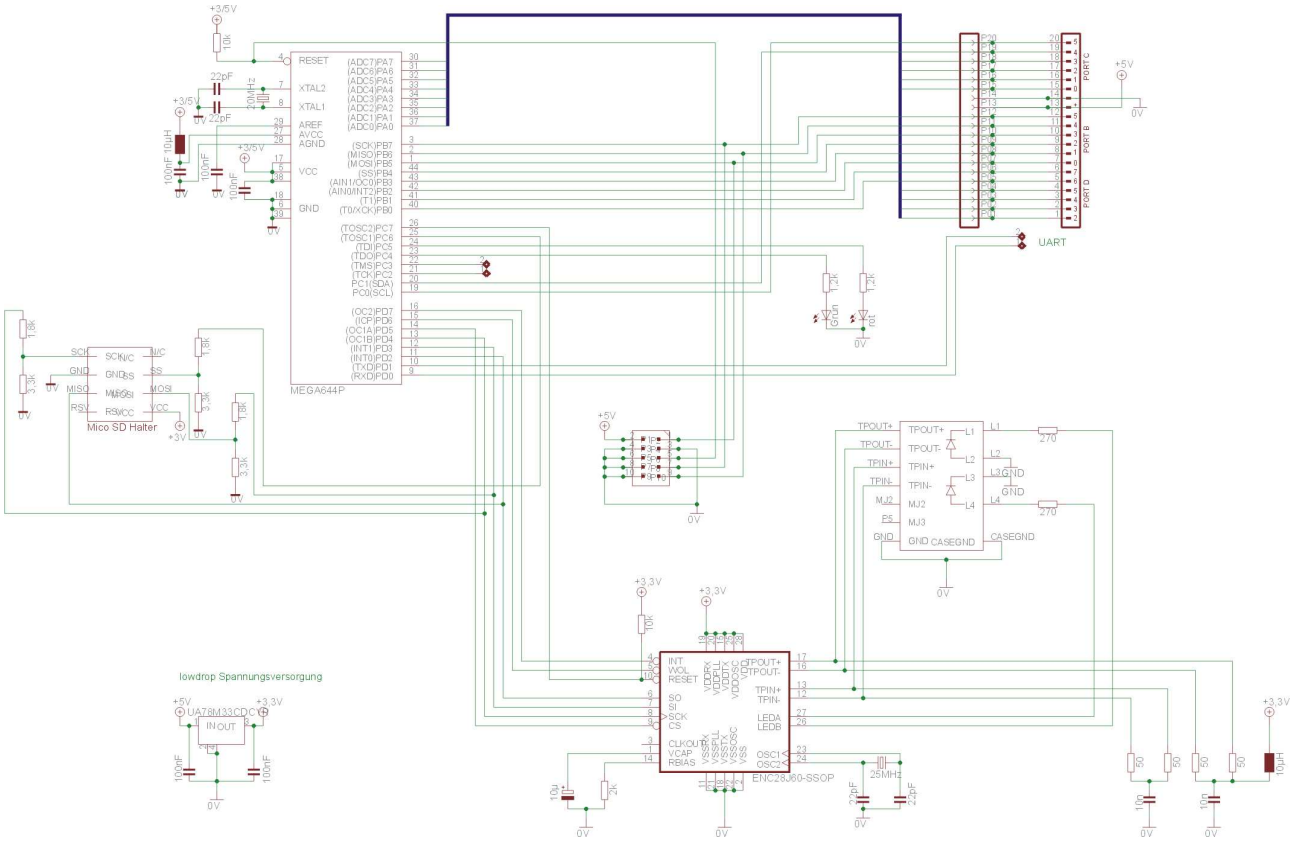
myTWI Echtzeituhr

7.2 Schaltpläne

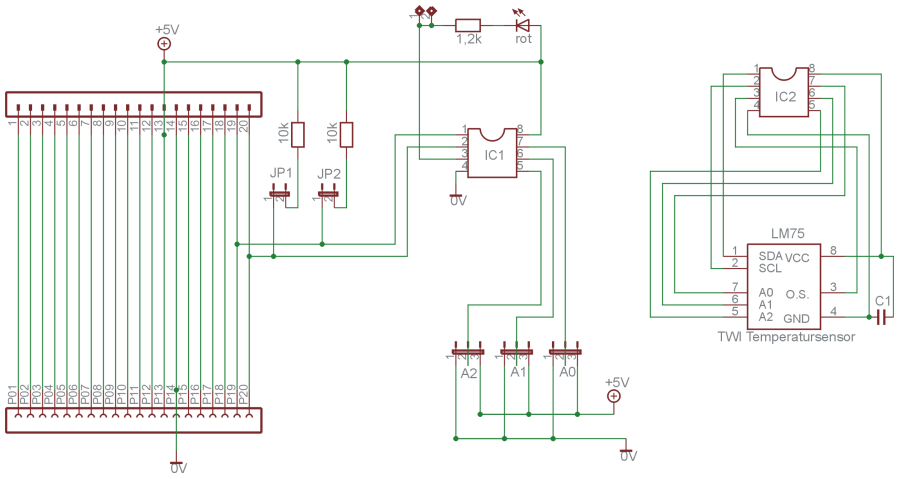
mySmartControl MK3



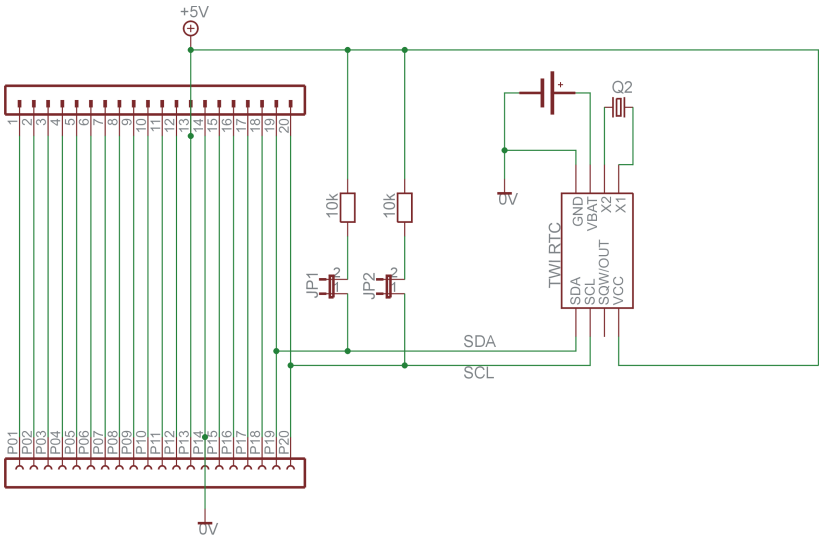
myEthernet


MEGA644P
RESET
XTAL2
XTAL1
AREF
AVCC
AGND
VCC
GND
Mico SD Halter
lowdrop Spannungsversorgung
UA78M33CDCY
ENC28J60-SSOP
25MHz
UART
PORT C
PORT B
PORT D
TPOUT+
TPOUT-
TPIN+
TPIN-
CASEGND

myTWI Temperatursensor

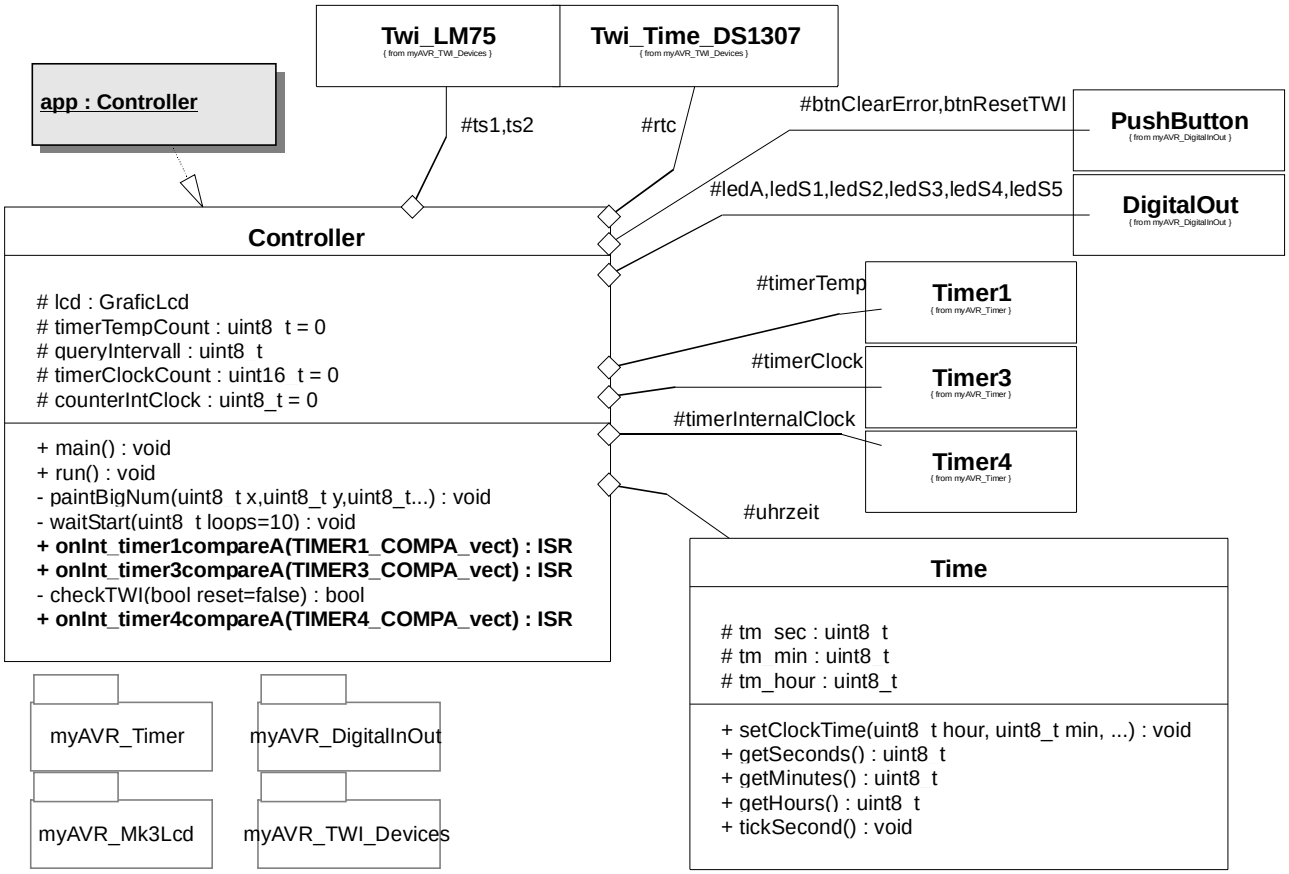


myTWI Echtzeituhr



8 Klassen

8.1 Klassendiagramm



8.2 Methoden der Klasse Controller

```
void Controller::main()
{
	waitMs(1500);				//init wait
	lcd.configLcd(PORTA,PORTC); //LCD
	ledA.config(PORTL,BIT0);	//LEDs
	ledS1.config(PORTL,BIT3);
	ledS2.config(PORTL,BIT4);
	ledS3.config(PORTL,BIT5);
	ledS4.config(PORTL,BIT6);
	ledS5.config(PORTL,BIT7);
	btnResetTWI.config(PORTK,BIT1);		//taster B
	btnClearError.config(PORTK,BIT2);	//taster C
	waitStart();			// auf Ethernet warten mit TWI init
	ts1.config(0x94);		// Tempsensoren
	ts2.config(0x90);
	rtc.config(0xD0);		// Echtzeituhr
	lcd.bitmapFlash(0,0,bg); // Hintergrundbild ins LCD laden
	//Timer für ~1s
	timerTemp.config_compareMatch(Timer1::sourcePrescale1024,19531);
	timerTemp.configInt_compare(true);  // Interrupt an
	//Timer für ~100ms
	timerClock.config_compareMatch(Timer3::sourcePrescale64,31250);
	timerClock.configInt_compare(true); // Interrupt an
	//Timer für ~100ms
	timerInternalClock.config_compareMatch(Timer4::sourcePrescale64,31250);
	timerInternalClock.configInt_compare(true); // Interrupt an
	run();					// Endlosschleife starten
}

void Controller::run()
{
	queryIntervall = 25;   // Abfragezeit in Sekunden für Sensoren
	//in erster Schleife auch Temperaturen abfragen
	timerTempCount = queryIntervall-1;
	lcd.fontStyle = LCD_FONT_WIDE; // LCD Fontstyle (wide) für Uhrzeit
	lcd.setPos(0,0);
	lcd.writeTextRam("Zeit:");
	// Mainloop
	do {
	    waitMs(500);
	    if(btnClearError.isPressed()) {		// Taster C > LEDs aus
	        if(ledS4.getState())		// LED an?
	            ledS4.off();
	        if(ledS5.getState())
	            ledS5.off();
	    }
	    if(btnResetTWI.isPressed()) {  // Taster B > TWI Reset
	        while(!checkTWI(true)) {
	            ledA.toggle();
	            waitMs(500);
	        }
	        ledA.on();
	    }
	}while(1);
}

void Controller::paintBigNum(uint8_t x,uint8_t y,uint8_t zahl)
{
	//Ziffern für Bitmaps
	uint8_t ziff1=0;
	uint8_t ziff2=0;
	if(zahl>=10 && zahl<100)  // 10...99
	{
	    ziff1=zahl/10;
	    ziff2=zahl-10*ziff1;
	    lcd.bitmapFlash(x,y,ziff[ziff1]);
	    lcd.bitmapFlash(x+20,y,ziff[ziff2]);
	}
```

```
       else if(zahl>=0 && zahl<10)      // 0...9
       {
           ziff2=zahl-10*ziff1;
           lcd.bitmapFlash(x,y,ziff[10]); //leer
           lcd.bitmapFlash(x+20,y,ziff[ziff2]);
       }
       else
       {   // eine 0 schreiben
           ledS4.on();
           lcd.bitmapFlash(x,y,ziff[10]);
           lcd.bitmapFlash(x+20,y,ziff[0]);
       }
}


ISR Controller::onInt_timer1compareA (TIMER1_COMPA_vect)
{
    timerTempCount++;

    if(timerTempCount==queryIntervall-1) //Tempsensor 1
    {
        ledS2.on();  // LED an während Abfrageprozess
        ts1.errorcode=0;
        uint8_t temp=ts1.getHalfGrad(); //Temperatur in Halbgrad auslesen
        if(checkTWI())   // TWI check
        {
            temp=temp/2; // Grad berechnen
            paintBigNum(53,16,temp); // Zahl auf LCD anzeigen
        }
        ledS2.off(); // LED aus nach Abfrageprozess
    }
    else if(timerTempCount==queryIntervall) //Tempsensor 2
    {
        ledS3.on();
        ts2.errorcode=0;
        uint8_t temp=ts2.getHalfGrad();
        if(checkTWI())
        {
            temp=temp/2;
            paintBigNum(53,40,temp);
        }
        ledS3.off();
        // Timerzähler für Temperatur zurücksetzen
        timerTempCount=0;
    }
}

ISR Controller::onInt_timer3compareA (TIMER3_COMPA_vect)
{
    if(timerClockCount==0)
    {
        uint8_t hour=0;
        uint8_t min=0;
        uint8_t sec=0;
        ledS1.on();
        rtc.errorcode=0;
        rtc.getClockTime(hour,min,sec); // Zeit von TWI-Clock holen
        if(checkTWI())          // TWI check
            uhrzeit.setClockTime(hour,min,sec); // interne Uhr stellen
        timerClockCount = 3000; // nach 5 Minuten -> neue Zeit holen
        ledS1.off();
    }
    // Uhrzeit ausgeben
    if(timerClockCount%5==0)      // 0.5 Sekunde Intervall
    {
        char rtcbuff[12]={0};   // Puffer RTC
        lcd.setPos(53,0);
         // Uhrzeit in Puffer formatiert speichern
         sprintf(rtcbuff,"%02d:%02d:%02d  ",uhrzeit.getHours(),
              uhrzeit.getMinutes(),uhrzeit.getSeconds());
        lcd.fontStyle=LCD_FONT_WIDE;
         // Uhrzeit ausgeben
```

```
        lcd.writeTextRam(rtcbuff);
    }
    timerClockCount--;  // Timerzähler der internen Uhr dekrementieren
}


ISR Controller::onInt_timer4compareA (TIMER4_COMPA_vect)
{
    // Uhrzeit intern
    counterIntClock++;
    if(counterIntClock%10==0)   // 1s (10 Timerevents je 100ms)
    {
        uhrzeit.tickSecond();   // interne Uhr inkrementieren
        counterIntClock=0;
    }
}


void Controller::waitStart (uint8_t loops=10)
{
    // Startschleife (warten auf myEthernet)
    lcd.clear();
    lcd.writeTextRam("Initialisierung");
    lcd.setPos(0,16);
    for(uint8_t i=0;i<loops;i++)
    {
        ledA.toggle();    // LED A > blinken
        waitMs(500);
    }
    lcd.clear();
    waitMs(1000);
}

bool Controller::checkTWI (bool reset=false)
{
      bool ok=false;
      // Errorcode oder manueller Reset per Taster?
      if(ts1.errorcode || ts2.errorcode || rtc.errorcode || reset)
      {
          ledS5.on();                  // LED5 an > Fehlerstatus
          TWCR=0;
          rtc.config(0xD0);         // RTC
          waitMs(10);
          ts1.config(0x94);         // Tempsensoren
          waitMs(10);
          ts2.config(0x90);
          waitMs(10);
          if(reset)             // manueller Reset
          {
              ts1.errorcode=ts2.errorcode=rtc.errorcode=0;
              ok=true;
              ledS5.off();
          }
      }
      else
      {
          ok=true;
          ledS5.off();
      }
      return ok;
}
```

8.3 Implementationen der Klasse Controller (Ausschnitt)

```
// grafische Ziffern
static const PROGMEM uint8_t ziff[11][63] =
{
    20,24, /* x,y */
    0x00, 0x00, 0x80, 0xE0, 0xF8, 0x7C, ...
    0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, ...
    0x00, 0x00, 0x01, 0x07, 0x1F, 0x3E, ...
    0, /* Ende */
    ...
    0 /* Ende */

}

// grafischer Hintergrund
static const PROGMEM uint8_t bg[] =
{
    128,64, /* x,y */
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, ...
    0x0C, 0x0C, 0x0C, 0x0C, 0x0C, 0x0C, ...
    0x00, 0x00, 0x02, 0x02, 0xFE, 0xFE, ...
    0x00, 0x00, 0x01, 0x01, 0x01, 0x01, ...
  ...
  0 /* Ende */
};
```

8.4 Methoden der Klasse Time

```
void Time::setClockTime (uint8_t hour, uint8_t min, uint8_t sec)
{
       // interne Uhr stellen
       if(hour<24 && hour>=0)
           tm_hour=hour;
       if(min<60 && min>=0)
           tm_min=min;
       if(sec<60 && sec>=0)
           tm_sec=sec;
}

uint8_t Time::getSeconds()
{
       uint8_t mySeconds=tm_sec; // Sekunden zurückgeben
       return mySeconds;
}

uint8_t Time::getMinutes()
{
       uint8_t myMinutes=tm_min; // Minuten zurückgeben
       return myMinutes;
}

uint8_t Time::getHours ()
{
       uint8_t myHours=tm_hour;  // Stunden zurückgeben
       return myHours;
}

void Time::tickSecond()          // interne Uhr 1s erhöhen
{
       if(tm_sec<59) // Sekunden inkrementieren, solange bis Minute erreicht
           tm_sec++;
       else
       {
           tm_sec=0;
           if(tm_min<59) // Minuten inkrementieren
               tm_min++;
           else
           {
               tm_min=0;
```

```
                if(tm_hour<23) // Stunden inkrementieren, ab 24 auf 0 setzen
                    tm_hour++;
                else
                    tm_hour=0;
            }
        }
}
```

9 Stückliste

Bezeichnung	Kategorie Hersteller	Anzahl
mySmartControl MK3	Systemboards / mySmartControl MK3 (http://shop.myavr.de)	1
myEthernet	Add-Ons und Module / myEthernet (http://shop.myavr.de)	1
myTWI Echtzeituhr	Add-Ons und Module / myTWI Echtzeituhr (http://shop.myavr.de)	1
myTWI Temperatursensor	Add-Ons und Module / myTWI Temperatursensor (http://shop.myavr.de)	2
Webcam	Fachhandel	1
externer Temperatursensor	Fachhandel	1
Netzwerkkabel	Fachhandel	1
Mini-USB-Kabel	Zubehör / USB-Kabel (http://shop.myavr.de)	1
myAVR Netzteil	Zubehör / myAVR Netzteil (http://shop.myavr.de)	1