Väderstation

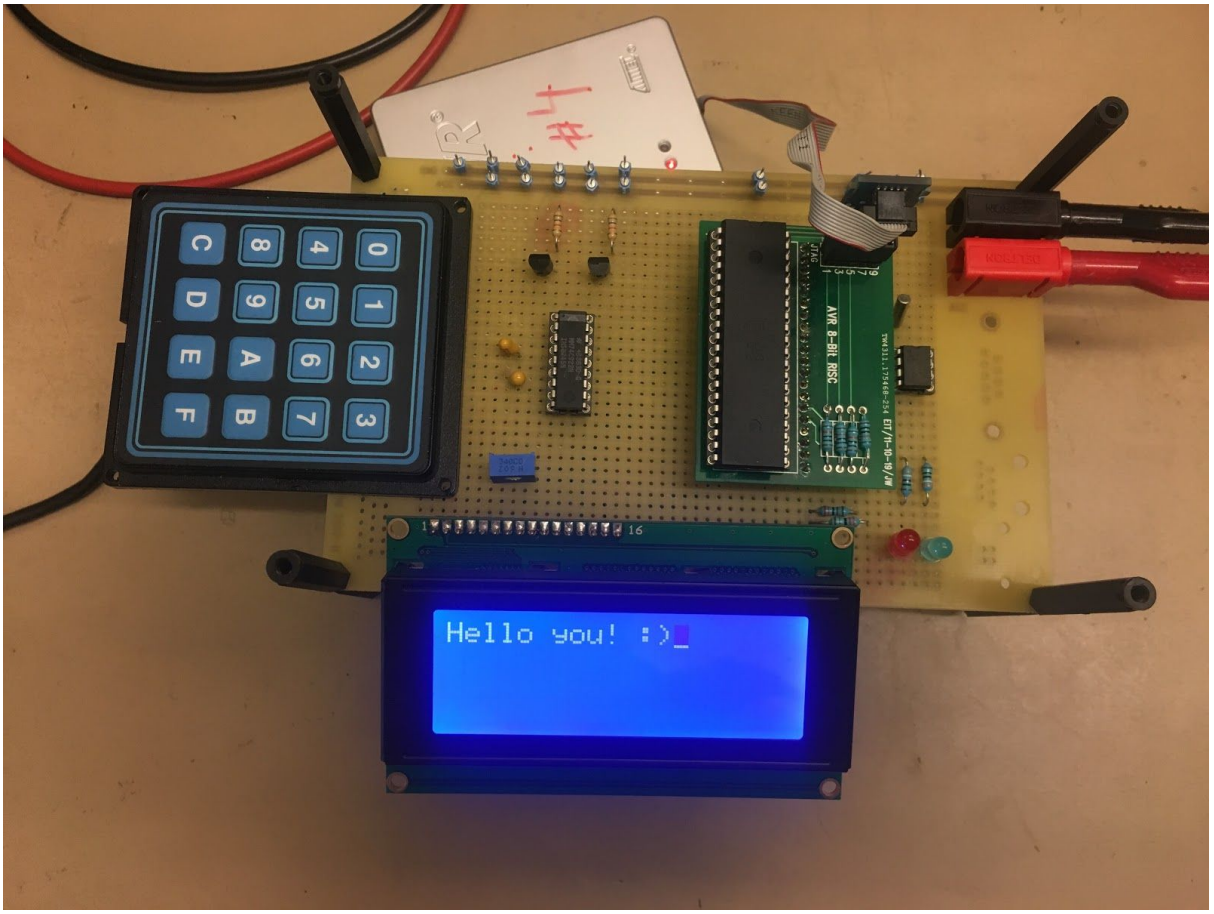
EITF11, Digitala projekt

RAPPORT

Isabella Bergvik, Emma Sandén & Hanna Åberg

Handledare: Bertil Lindvall
Institution: Elektro- och informationsteknik, Lunds tekniska Högskola

2018-05-21



Figur 1. Konstruktionen

Abstract

This project has been the main part of the course Digital projects EITF11. The project consists of the construction of hardware and software for a basic weather station. The weather station is capable of collecting and presenting data from two temperature sensors. It can then, on a display, show current temperature and average day temperature. Aside from this it is also capable of sending out a warning when the users programed max and minimum temperature is exceeded. To do this a diagram of the component was initially made. The diagram was updated under the course of the project continuously developing the construction to fulfill the requirements set. The construction was built parallel to the software being developed. To test that the functions aligned with the requirements the construction was tested using a logic pen and a multimeter to ensure that the right signals were sent. The software was then tested using the debugging function of Atmel studio, were the programming was done, and the JTAG. The project included several challenges, for example broken pieces of the internal construction. The challenges were solved with help from the teachers and led to a large learning process. The result was a construction that met most of the requirement, with small faults in displaying time and reading the temperature continuously.

Innehållsförteckning

1.Inledning

Detta projekt har utgjort den huvudsakliga delen av kursen Digitala projekt (EITF11). Projektet har innefattat att konstruera en hårdvara och programmera mjukvara till en färdigställd produkt som i slutändan ska redovisas till handledare och opponentgrupp. Valet av projekt har bestämts utan gruppen utifrån en samling möjliga alternativ framtagna och kursens ledning.

Projektgruppen valde således att konstruera en väderstation enligt de krav och avgränsningar som finns specificerade i 2.2 *kravspecifikation* samt 1.2 *avgränsningar*.

1.1 Syfte

Syftet med detta projekt har varit att illustrera industriellt utvecklingsarbete.
Detta har gjorts med målsättningar att:

- Analysera och beskriva system av låg och medelhög komplexitet
- Testa och felsöka en konstruktion på ett systematiskt sätt
- Söka upp och tillgodogöra sig relevant information

Vidare har syftet även varit att utveckla följande färdigheter:

- Realisera digitala system av låg och medelhög komplexitet
- Driva ett projekt framåt till en fungerande prototyp
- Formulera sig muntligt och skriftligt

Att nå de ovan nämnda insikterna och samtidigt utveckla grundläggande färdigheter i ämnet har varit de primära drivkrafterna genom hela arbetet.

1.2 Avgränsningar

Projektet har avgränsats till att endast innefatta de kravspecifikationer nämnda i *2.2 kravspecifikation.* Detta har gjort att antalet komponenter kunnat begränsas till de presenterade i *3.2 Huvudkomponenter* med mjukvara formad utefter komponenternas funktioner. Dessa avgränsningar är valda utefter hur kursens upplägg sett ut. Det har också gjorts en uppskattning om hur stor omfattning som är rimlig på ett projekt motsvarande 10 högskolepoäng.

1.3 Process

Projektet har utförts enligt följande process:

1. Idéutveckling och planering
    a. Val av projektarbete
    b. Utformning av kravspecifikation
    c. Val av komponenter till konstruktionen
    d. Planering av arbetsgång och tidsomfattning
2. Hårdvaruutveckling
    a. Studerande av datablad för valda komponenter till konstruktionen
    b. Utformning av kopplingsschema
    c. Praktiskt bygge av konstruktionen utefter kopplingsschemat
    d. Testning av komponenternas basala funktion
3. Mjukvaruutveckling
    a. Studerande av datablad
    b. Kodning varvat med testning
4. Slutleverans
    a. Redovisning av prototypens funktion
    b. Rapport och hemsida

2. Produkt

2.1 Produktbeskrivning

Produkten som har tagits fram är en väderstation med funktionen att samla in och presentera mätdata från två temperatursensorer. Väderstationen är relativt enkel och kan presentera nuvarande temperatur samt genomsnittlig dygnstemperatur. Utöver detta kan den även skicka en varning till användaren då en av användaren förutbestämd minimum- eller maximumgräns för temperaturen överskrids. Varningen skickas ut i form av en lysande lampa.

Produkten består av hårdvara (se *3. Hårdvara*) och mjukvara (se *4. Mjukvara*).

2.2 Kravspecifikation

Konstruktionen specificeras av följande krav

Produkten skall kunna...

- … mäta två analoga signaler [1]
- … avläsa och kontinuerligt presentera temperaturen från temperaturgivare [2]
- … programmera två larmgränser via en knappsats [3]
- .... presentera max- och mintemperatur samt dygnsmedeltemperatur [4]
- … skicka en varning då någon av larmgränserna överskrids [5]
- … presentera mätinsamling med tidsangivelse på dator(månad, dag, timme och minut) [6]

3. Hårdvara

3.1 Kopplingsschema

Innan konstruktionen av hårdvaran kunde inledas togs ett kopplingsschema fram. Kopplingsschemat baserade sig på de valda komponenterna och dess datablad. Under arbetets gång har kopplingsschemat uppdaterats från originalet vid insikter om funktioner som behövts läggas till under arbetets gång. Det slutgiltiga kopplingsschemat för hårdvarukonstruktionen återfinns i appendix A.

3.2 Komponenter

Hårdvaran består av följande komponenter:

- Processor
    - ATMega16 High-Performance AVR 8-bit Microcontroller
    - *Används för att koppla ihop och styra samspelet mellan övriga komponenter*

- Temperatursensorer
    - LM335 Precision Temperature Sensors
    - *Används för att mäta temperaturen*.
- Display
    - SHARP Dot-Matrix, GDM1602K3
    - *Används för att skriva ut mätdata* .
- Knappsats

- *Används för att navigera mellan olika mätdata samt styra varningsgränser samt val av inne- eller utetemperatur i programmet.*

❖ Encoder

- MM54C922/MM74C922 16-Key Encoder
- *Används för att omvandla input från knappsatsen till binära värden som skickas till microprocessorn*.

- RTC
  - MCP7940M - I2C - Real time clock
  - *Används för att visa nuvarande datum och tid*
- JTAG
  - *Används för debugging, felsökning och allmän testning.*

4. Mjukvara

Den största delen av projektet har bestått i att utveckla programvaran. Koden har skrivits i Atmel Studio 7 i programspråket C.

I källkoden återfinns en main-metod som först initierar flöde och värden och därefter främst består i en while-loop som avbryts då avbrott initieras från knappsatsen.

Koden har kontinuerligt testas med komponenten för att analysera att de önskade funktioner uppfylls. Detta har gjorts genom att debugga samt att testa olika fall med hjälp av JTAGen. Källkoden återfinns i appendix B.

5. Utförande

5.1 Hårdvaruutveckling

Efter att ha fastställt att projektet skulle omfatta en väderstation och tagit fram ett kopplingsschema kunde konstruktionen av hårdvaran påbörjas. Konstruktionen utgick från kopplingsschemat och kopplades ihop i enlighet med detta. Några av komponenterna löddes fast på mönsterkortet och virtråd virades runt komponenternas pinnar utefter schemat för att koppla samman konstruktionen. När konstruktionen var färdig genomfördes olika test för att fastställa att konstruktionen var rätt kopplad. För att göra detta användes en logikpenna och en multimeter. samt enklare kod. Då kollades det att

rätt out- och input kom till rätt portar och pinnar. För att säkerställa att hårdvara och mjukvara uppträdde enligt kravspecifikationen användes debugger-funktionen i Atmel Studio 7. Under processens gång upptäcktes exempelvis fel såsom att några av mikroprocessorns pinnar behövde lödas fast ordentligt då den från början varit trasig och att skärmstyrkan på display:en behövde justeras. Det upptäcktes även kontinuerligt under arbetet att det saknades komponenter som påverkade funktionen, exempelvis att två kondensatorer behövde installeras. Vid sådana tillfällen justerades först kopplingsschemat i enlighet med datablad hos komponenterna och sedan justerades komponenten baserat på detta. Efter ett antal justeringar testades konstruktionen igen i enlighet med tidigare testmetoder för att fortsatt säkerställa funktionen.

5.2 Mjukvaruutveckling

Parallellt med att hårdvaran utvecklades påbörjades utvecklingen av mjukvaran i Atmel Studio 7. Detta för att lättare få förståelse för vad som kunde testas i samband med testerna av hårdvaran. Mjukvaran programmerades så att in- och utflöden mellan mikroprocessorn och komponenterna ställdes in för att kunna skicka kommandon från knappsatsen till displayen, omvandla analoga signaler från temperatursensorerna via mikroprocessorn samt konvertera dessa till signaler som kan visas på display:en.

Vid framtagande av koden användes databladen angivna under *7.Referenser*. I dessa framgick vilka portar som skulle vara in- respektive utsignaler(1:or och 0:or) för att få fram önskade kommandon. Kontinuerligt under framtagandet av koden testades den med hjälp av JTAGen och dess debugging funktion. Vid närmande av komplett produkt testades koden direkt genom observation av displayen på väderstationen vid knapptryck. Koden justerades i enlighet med resultatet observationerna gav.

6. Utvärdering

6.1 Resultat

Projektets slutresultat är en väderstation som fungerar i relativ enlighet med kravspecifikationen. Kraven är uppfyllda, utom krav [2] och [6] som där som endast är delvist uppfyllda. Detta ansågs vara överkomliga fel.

Krav [2], att kontinuerligt avläsa temperatur från sensorerna är ej fullständigt uppfyllt då temperaturen endast avläses och uppdateras då konstruktionen sätts på.

Krav [6], att presentera tidsangivelse, är ej fullständigt uppfyllt då datumet ej visas på displayen korrekt samt att startvärden på högre tidsangivelser för timme, minut och sekund ej visas korrekt. Lägre värden visas korrekt. I övrigt fungerar tidsangivelsen som specificerat.

Slutkonstruktionen kan obsererverас i figur 1 och källkoden återfinns i appendix B.
För att korrekt kunna använda konstruktionen hänvisas användaren till manualen i appendix C.

6.2 Diskussion

Arbetet har inneburit flertalet utmaningar men där även bidragit till lärorika insikter.

En av de största svårigheterna under projektets gång var tolkande av databladen för de olika komponenterna. Detta har varit ett återkommande moment under processens gång. Från utformande av kopplingsschema till att testa den sista koden har databladen spelat en vital roll och varit utmanande att hantera korrekt. Detta har samtidigt kontinuerligt gett mer insikt och förståelse för att arbeta med sådana verktyg.

Andra utmaningar under projektets gång har varit felsökning av komponenten och koden. Ett problem som stöttes på var att processorn hade pinnar som inte satt fast korrekt vilket behövde korrigeras. För att upptäcka felet krävdes en nedbrytning av hela processorn, metoder som var svåra att våga sig på på egen hand.

Vidare låg en stor utmaning i utformandet av realtidsklockan. Där fanns svårigheter i både förståelse av dess funktion och framför allt i utformande av korrekt kod för att den skulle fungera som tänkt. Denna utmaning var en som aldrig helt löstes, då resultatet med tidsangivelsen inte var helt korrekt.

Projektet har krävt mycket assistans från kursledare Bertil Lindvall och övningsledare Christofer Cederberg. Detta har innefattat vägledning och genomgång av såväl grunder i elektronik till programmering i C. Detta har varit av yttersta vikt för säkerställande av projektets framfart och genomförande.

Sammantaget har arbetet inneburit flertalet utmaningar och problem att lösa som i slutändan bidragit till stor utveckling och mycket ny kunskap inom ämnet. Det gav även ett sammantaget bra resultat som speglar de tänkta funktionerna nästintill fullständigt. Bristerna i konstruktionen ansågs överkomliga och syftet och målen med arbetet därmed ändå uppfyllda.

7. Referenser

Datablad

Display - SHARP Dot-Matrix, GDM 1602K
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf

Encoder - MM54C922 - Key - Encoder
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/Other/MM54C922.pdf

MPU - ATMega16 High Performance AVR 8-bit Microcontroller
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf

Temp sensor - LM35 Precision Temperature Sensor
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Sensors/LM35.pdf

RTC - MCP7940M - Real time clock I2C
https://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Periphery/RTC/MCP7940.pdf/

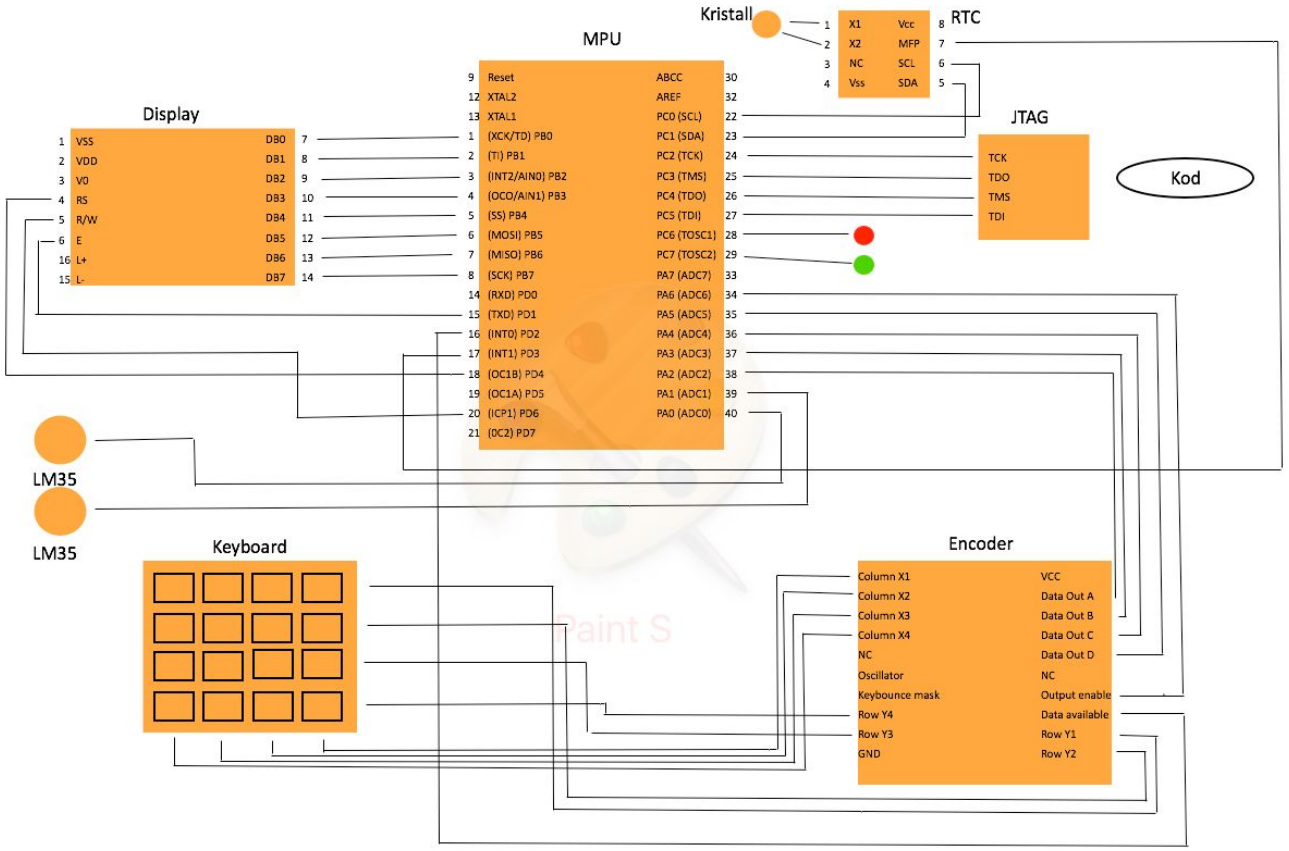
Muntliga källor

Cederberg, Christoffer, Elektro- och informationsteknik, Lunds Tekniska Högskola, 2018

Lindvall, Bertil, Elektro och informationsteknik, Lunds Tekniska Högskola, 2018

Appendix A

Kopplingsschema



Appendix B

Programkod i C

```
/*
 * GccApplication1.c
 *
 * Created: 2018-04-12 09:07:28
 * Author : ine14hab
 */

#define F_CPU 8000000UL
#define F_SCL 100000UL


//LCD macros

#define LCD_ENABLE 1
#define LCD_RW 6
#define LCD_RS 4
#define LCD_DELAY 50

#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <util/twi.h>
#include <string.h>

#include "rtc_mcp7940m.h"
#include "i2c_master.h"



//ATTRIBUTES

int tempIn;
int tempOut;

int maxTempIn;
int minTempIn;
int averageTempIn;

int maxTempOut;
int minTempOut;
```

```
int averageTempOut;

int tempVecIn[100]; // annan typ av deklaration i c tror jag...
int tempVecOut[100];
int tempIndexIn; // håller reda på om hur pass fylld vektorn är
int tempIndexOut; // håller reda på om hur pass fylld vektorn är

int outOrIn; //håller reda på om vi ska mäta ute- eller innetemperatur
int minOrMax; //håller reda på om vi ska uppdatera värde på max eller
min
int interTemp;
int change;

int MAX = 0; // godtyckliga värden
int MIN = 250;

volatile char val;
volatile char knappflagg = 0;
volatile char ticked = 0;

volatile unsigned char albin;
volatile int albin1;

uint8_t set_sec = 0;
uint8_t set_min = 0;
uint8_t set_h = 12;
uint8_t set_day = 5;
uint8_t set_date = 17;
uint8_t set_month = 5;
uint8_t set_year = 18;

uint8_t get_sec;
uint8_t get_min;
uint8_t get_h;
uint8_t get_day;
uint8_t get_date;
uint8_t get_month;
uint8_t get_year;


//METHODS
void initDiods(void);

void displayChar(char);
void displayString(char[]);
void initDisplay(void);
void displayOnOff(void);
void configureDisplay(void);
```

```
void clearDisplay(void);
void setRow(char);

void initThermal(void);
void print(int);
void printTemp(int);
void print_calc_time(uint8_t);
int calcTemp(int);
void warningLow(void);
void warningHigh(void);
void maxTempSetUp(void);
void minTempSetUp(void);
void increaseMaxTemp(void);
void decreaseMaxTemp(void);
void increaseMinTemp(void);
void decreaseMinTemp(void);
void clearVectors(void);
void insertTempIn(int);
void insertTempOut(int);
void printAverageTempIn(void);
void printAverageTempOut(void);

void enable_interupt(void);
void set_up_tick(void);

void setUpKeyboard(void);
void buttonPressed(void);

void printTime(void);


//DIODS

void initDiods() {
      DDRC |= (1<<PC6);
      DDRC |= (1<<PC7);
}

// DISPLAY

//Sätter DDR (Data Direction Registers) för att kontrollera displayen
void initDisplay() {
      DDRD |= (1<<LCD_ENABLE)|(1<<LCD_RW)|(1<<LCD_RS);
      DDRB = 0xFF;
}

//Konfigurerar displayen
void configureDisplay()
```

```
{
	_delay_ms(LCD_DELAY);
	PORTD |= (1<<LCD_ENABLE);
	PORTB = 0x3F;
	PORTD &= ~(1<<LCD_ENABLE);
	PORTD |= (1<<LCD_ENABLE);
}

//Sätter på displayen
void displayOnOff()
{
	_delay_ms(LCD_DELAY);
	PORTD |= (1<<LCD_ENABLE);
	PORTB = 0x0F;
	PORTD &= ~(1<<LCD_ENABLE);
	PORTD |= (1<<LCD_ENABLE);

}

//Rensar display på tecken, se "ClearDisplay"
void clearDisplay()
{
	_delay_ms(LCD_DELAY);
	PORTD |= (1<<LCD_ENABLE);
	PORTB = 0x01;
	PORTD &=~ (1<<LCD_ENABLE);
	PORTD |= (1<<LCD_ENABLE);

}

//Skriver siffror och bokstäver på displayen
void displayChar(char c)
{
	_delay_ms(10);
	PORTD |= (1<<LCD_RS); //E och RS sätts till 1, övriga 0
	PORTD &= ~(1 << LCD_RW); // write mode
	PORTD  |= (1<<LCD_ENABLE);
	PORTB = c;
	PORTD &=~ (1<<LCD_ENABLE);
	PORTD &= ~(1<<LCD_RS);
	PORTD |= (1<<LCD_ENABLE);
	_delay_ms(10);
}

//Skriver meningar på displayen
void displayString(char str[]) {

	 uint8_t i = 0;
```

```
       while(str[i] != '\0') {
             displayChar(str[i]);
             i++;
       }
}

//Bestämmer vilken rad vi ska skriva på
void setRow(char row)
{
      PORTD |= (1<<LCD_ENABLE);
      PORTB = row;
      PORTD &=~ (1<<LCD_ENABLE);
      PORTD |= (1<<LCD_ENABLE);
      _delay_ms(LCD_DELAY);
}


// TEMPERATURE

//Initierar temp sensorer
void initThermal() //se sida 202 i handbok
{
      ADMUX |= 0b01000000;
      ADMUX &=~ (1<<ADLAR);
      ADCSRA |= (1<<ADEN)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
      ADCSRA |=(1<<ADSC);
}

// Interrup för temperatursensor
ISR(ADC_vect)
{
      //int albin;
      if((interTemp > 0)) { //kanal 0
            albin=0;
            albin = ADCL;
            tempIn = (albin-140)/2.75;
            if (tempIn > MAX)
            {
                  MAX = tempIn;
            }
            if (tempIn < MIN)
            {
                  MIN = tempIn;
            }
            ADMUX |= 0b01000001;
            interTemp = 0;
```

```
	} else { //kanal 1
		albin1 = ADCL;
		tempOut = (albin1-140)/2.75;
		 if (tempOut > MAX)
		 {
			MAX = tempOut;
		 }
		 if (tempOut < MIN)
		 {
			MIN = tempOut;
		 }
		ADMUX |= 0b01000000;
		interTemp = 1;
	}
	ADCSRA |= (1<<ADSC); //starta A/D
}

//Skriver ut info om temp, så som varningar etc
void print(int k)
{
	if(outOrIn >0)//Inomhus
	{
		if(k>maxTempIn) // Varning, överskrider högsta temepratur
		{
			printTemp(k);
			_delay_ms(1000);
			displayChar(' ');
			warningHigh();
		}
		if(k < minTempIn) // Varning, underskriver minsta temperatur
		{
			printTemp(k);
			_delay_ms(1000);
			displayChar(' ');
			warningLow();
		} else {
			printTemp(k); //ingen varning
		}
	} else //Utomhus
	{
		if(k>maxTempOut) //Varning, överskrider högsta temperatur
		{
			printTemp(k);
			_delay_ms(1000);
			displayChar(' ');
			warningHigh();
		}
		if(k < minTempIn) // Varning, underskriver minsta temperatur
```

```
            {
                  printTemp(k);
                  _delay_ms(1000);
                  displayChar(' ');
                  warningLow();
            } else {
                  printTemp(k); //ingen varning
            }
      }
}

//Skriver ut temperatur
void printTemp(int t)
{
      int ental = t % 10;
      t= t/10;
      int tiotal = t%10;
      t= t/10;
      int hundratal = t%10;

      if (hundratal!= 0) // ej säkert att det finns hundratal
      {
      int a = calcTemp(hundratal);
      displayChar(a);
      }
      if (tiotal!= 0) // ej säkert att det finns tiotal
      {
      int b = calcTemp(tiotal);
      displayChar(b);
      }
      int  c= calcTemp(ental); // säkert att det finns ett tal
      displayChar(c);

      displayChar(0b11011111); // tecken för celcius
      displayChar('C');
}

//Skriver ut tiden
void print_calc_time(uint8_t h)
{
      int ental = h%10;
      h= h/10;
      int tiotal = h%10;
      h= h/10;
      int hundratal = h%10;

      if (hundratal!= 0)
      {
```

```
            int a = calcTemp(hundratal);
            displayChar(a);
      }
      if (tiotal!= 0)
      {
            int b = calcTemp(tiotal);
            displayChar(b);
      } else
      {
            displayChar('0');
      }
      int  c= calcTemp(ental);
      displayChar(c);
}

//Omvandlar temperaturen x till en siffra som kan generera x i
displayen
int calcTemp(int x)
{
      x += 0b00110000; // lägger till 49 här för att siffran ska stämma
överens med tabellen i displayens siffror. Motsvarar siffran i bits
      return x;
}

//Varningsmeddelande för min-temp
void warningLow()
{
      PORTC|=(1<<PC6);
      setRow(0xC0);
      displayString("WARNING! Low temp!");
      _delay_ms(100);
      PORTC &=~(1<<PC6);
}

//Varningsmeddelande för max-temp
void warningHigh()
{
      PORTC |= (1<<PC6);
      setRow(0xC0);
      displayString("WARNING! High temp!");
      _delay_ms(100);
      PORTC &=~(1<<PC6);
}

//Skriver ut vad maxgränsen för temperaturen är för inne- eller
utemperatur
void maxTempSetUp()
{
```

```
	clearDisplay();
	displayString("Max limit for ");
	setRow(0xC0);

		if (outOrIn > 0)
		{
			displayString("indoor temp is now: ");
			setRow(0x94);
			printTemp(maxTempIn);
		}
		else
		{
			displayString("outdoor temp is now: ");
			setRow(0x94);
			printTemp(maxTempOut);
		}
}

//Skriver ut vad maxgränsen för temperaturen är för inne- eller
utemperatur
void minTempSetUp()
{
	clearDisplay();
	displayString("Min limit for ");
	setRow(0xC0);

		if (outOrIn > 0)
		{
			displayString("indoor temp is now: ");
			setRow(0x94);
			printTemp(minTempIn);
		}
		else
		{
			displayString("outdoor temp is now: ");
			setRow(0x94);
			printTemp(minTempOut);
		}
}

//Ökar maximala temperaturgränsen innan varning skickas med 1 kelvin
void increaseMaxTemp()
{
	if (outOrIn > 0)
	{
		maxTempIn += 1;
	}
	else
```

```
	{
		maxTempOut += 1;
	}
	maxTempSetUp();
}

//Minskar maximala temperaturgränsen innan varning skickas med 1 kelvin
void decreaseMaxTemp()
{
	if (outOrIn > 0)
	{
		maxTempIn -= 1;
	}
	else
	{
		maxTempOut -= 1;
	}
	maxTempSetUp();
}

//Ökar minsta temperaturgränsen innan varning skickas med 1 kelvin
void increaseMinTemp()
{
	if (outOrIn > 0)
	{
		minTempIn += 1;
	}
	else
	{
		minTempOut += 1;
	}
	minTempSetUp();
}

//Minskar minsta temperaturgränsen innan varning skickas med 1 kelvin
void decreaseMinTemp()
{
	if (outOrIn > 0) // sänker gräns inne
	{
		minTempIn -= 1;
	}
	else //sänker gräns ute
	{
		minTempOut -= 1;
	}
	minTempSetUp();
}
```

```
//Raderar mätdata i vektorerna och återställer mätindex
void clearVectors()
{
	int k = 0;
	while (k < tempIndexIn)
	{ //alla värden sätts till 0
		tempVecIn[k]=0;
		k++;
	}
	tempIndexIn=0;

	k=0;
	while (k < tempIndexOut)
	{ //alla värden sätts till 0
		tempVecOut[k]=0;
		k++;
	}
	tempIndexIn=0;
}

//Sparar temperaturavläsningar i två vektorer för inne och ute
void insertTempIn (int j)
{
if (tempIndexIn==99)
{
	tempIndexIn= 0;
}
		tempVecIn[tempIndexIn] = j;
		tempIndexIn+=1;
}

void insertTempOut (int j)
{
	if (tempIndexOut==99)
	{
		tempIndexOut= 0;
	}
		tempVecOut[tempIndexOut] = j;
		tempIndexOut+=1;
}

//Skriver ut den genomsnittliga temperaturen inne
void printAverageTempIn()
{
	int sum = 0;
	int k = tempIndexIn;
	while(k > 0)
	{
```

```
            sum +=tempVecIn[k];
            k -= 1;

      }
      int averageTempIn = sum/(tempIndexIn); //oklart vad vi ska dela
på
      displayString("Average indoor ");
      setRow(0xC0);
      displayString("temp: ");
      printTemp(averageTempIn);
}

//Skriver ut den genomsnittliga temperaturen ute
void printAverageTempOut()
{
      int sum =0;
      int k =tempIndexOut;
      while(k > 0)
      {
            sum +=tempVecOut[k];
            k-= 1;

      }
      int averageTempOut = sum/(tempIndexOut-1);
      displayString("Average outdoor ");
      setRow(0xC0);
      displayString("temp: ");
      printTemp(averageTempOut);
}


//INTERUPT FOR AVERAGE

//Initiera tick
void set_up_tick() {
      GICR |=(1<<INT1);
      MCUCR|= (1<<ISC11) |(1<<ISC10);
}

void enable_interupt() {
      i2c_start(RTC_WRITE);
      i2c_write(0x07);
      i2c_write(0b11000000);
      i2c_stop();
}

ISR(INT1_vect) {
      ticked = 1;
```

```
}

// KEYBOARD

//Konfigurerar inställningarna för knappsatsen
void setUpKeyboard()
{
	MCUCR |= 0b00000011; //Sista ska vara 0 för att interrupt, se
sida 67
	GICR |= 0b01000000; //INT0, bit 6,ska vara satt till 1, sida 67,
de andra ej relevanta
	DDRD &=~(1<<PD2); //Data available i encodern
}

//Interrupt keyboard
ISR(INT0_vect) {
	val=(PINA&0b00111100) >> 2; //Konfigurering av Data Output från
Encodern till MPU. A, B, C, D ska vara 1 ty vi ska läsa av dessa
pinnar, de är dessa som är relevanta
	knappflagg = 1;
}

//En knapp har trycks ned, läs av val
void buttonPressed() {

		if(val==0x03) //Knapp 0, start-sida
		{
			clearDisplay();
			printTime();
		}

		if(val==0x02) //Knapp 1, Skriv ut nuvarande innetemperatur
		{
			clearDisplay();
			displayString("Indoor temp: ");
			print(tempIn);
		}

		if (val==0x01) //Knapp 2, skriver ut nuvarande utetemperatur
		{
			clearDisplay();
			displayString("Outdoor temp: ");
			print(tempOut);
		}

		if (val==0x00) //Knapp 3, visar högst uppmätta temp
		{
			clearDisplay();
```

```
		displayString("Highest recorded");
		setRow(0xC0);
		displayString("temp: ");
		printTemp(MAX);
	}

	if (val==0x07) //Knapp 4, visar lägst uppmätta temp
	{
		clearDisplay();
		displayString("Lowest recorded");
		setRow(0xC0);
		displayString("temp: ");
		printTemp(MIN);
	}

	if (val==0x06) //Knapp 5, visar average temperatur inne
	{
		 clearDisplay();
		 printAverageTempIn();
	}

	if (val==0x05) //Knapp 6 visar average temperaturer ute
	{
		clearDisplay();
		printAverageTempOut();
	}

	if(val==0x04) //Knapp 7, data cleared!
	{
		clearDisplay();
		clearVectors();
		displayString("Data cleared!");
	}

	if(val==0b1011) //Knapp 8, bestäm ute eller inne
	{
		clearDisplay();
		displayString("You will now set: ");
		setRow(0xC0);
		if(outOrIn > 0)
		{
			outOrIn = 0;
			displayString("outdoor limits");
		} else
		{
			outOrIn = 1;
			displayString("indoor limits");
		}
```

```
            }

            if(val==0b1010) //Knapp 9, bestäm om max eller min
            {
                  clearDisplay();
                  _delay_ms(50);
                  displayString("You will now set:");
                  setRow(0xC0);
                  if(minOrMax > 0)
                  {
                        minOrMax = 0; //Sätter den till det den inte var
innan
                        displayString("max limit");
                  } else
                  {
                        minOrMax = 1;
                        displayString("min limit");
                  }
            }

            if(val==0x09) { //Knapp A, öka varningsgränsen med 1 celsius
                  clearDisplay();
                  if(minOrMax > 0)
                  {
                        increaseMinTemp();
                  } else
                  {
                        increaseMaxTemp();
                  }
            }

            if(val==0x08) { //Knapp B, minska varningsgränsen med 1
celsius
                  clearDisplay();
                  if(minOrMax > 0)
                  {
                        decreaseMinTemp();;
                  } else
                  {
                        decreaseMaxTemp();
                  }
            }

            if(val==0b1111) { //Knapp C, växla mellan år, månad och dag
                  clearDisplay();
                  displayString("You will now set:");
                  setRow(0xC0);
                  if (change == 0) {
```

```
                displayString("year");
                change = 1;
          } else if (change == 1) {
                displayString("month");
                change = 2;
          } else if (change == 2){
                displayString("day");
                change = 3;
          } else if (change==3){
                displayString("hour");
                change = 4;
          } else if(change==4){
                displayString("minute");
                change = 5;
          } else { //change är 5
                displayString("second");
                change = 0;
          }
    }

    if(val==0b1110) { //Knapp D, öka tid och datum
          clearDisplay();
          if (change == 1) {
                set_year+=1;
                displayString("Year is now: ");
                print_calc_time(set_year);
          } else if (change == 2) {
                set_month+=1;
                displayString("Month is now: ");
                print_calc_time(set_month);
          } else if (change == 3){
                set_day+=1;
                displayString("Date is now: ");
                print_calc_time(set_date);
          } else if (change==4){
                set_h+=1;
                displayString("Hour is now: ");
                print_calc_time(set_h);
          } else if(change==5){
                set_min+=1;
                displayString("Minute is now: ");
                print_calc_time(set_min);
          } else { //change är 5
                set_sec+=1;
                displayString("Second is now: ");
                print_calc_time(set_sec);
          }
    }
```

```
            if(val==0b1101) { //Knapp E, minska tid och datum
                  clearDisplay();
                  if (change == 1) {
                        set_year-=1;
                        displayString("Year is now: ");
                        print_calc_time(set_year);
                  } else if (change == 2) {
                        set_month-=1;
                        displayString("Month is now: ");
                        print_calc_time(set_month);
                  } else if (change == 3){
                        set_day-=1;
                        displayString("Date is now: ");
                        print_calc_time(set_date);
                  } else if (change==4){
                        set_h-=1;
                        displayString("Hour is now: ");
                        print_calc_time(set_h);
                  } else if(change==5){
                        set_min-=1;
                        displayString("Minute is now: ");
                        print_calc_time(set_min);
                  } else { //change är 5
                        set_sec-=1;
                        displayString("Second is now: ");
                        print_calc_time(set_sec);
                  }
            }

            if(val==0b1100) { //Knapp F, reboot time
                  clearDisplay();
                  //Stop clock
                  i2c_start(RTC_WRITE);
                  i2c_write(0x00);
                  i2c_stop();
                  //Start clock again, with new time values
                  set_clock();
                  displayString("Time is updated!");
                  setRow(0x94);
                  printTime();
            }
}


// TIME

//Skriver ut tid och datum
```

```
void printTime() {
	get_time(); //uppdatera tiden
	displayString("Date: ");
	print_calc_time(get_date);
	displayString("/");
	print_calc_time(get_month);
	displayString("/");
	print_calc_time(get_year);
	displayString(" ");

	setRow(0xC0);
	displayString("Time: ");
	uint8_t hour = 10*((0b00110000 & get_h)>>4) + (get_h &
0b00001111);
	print_calc_time(hour);
	displayString(":");
	uint8_t minutes = 10*((0b01110000 & get_min)>>4) + (get_min &
0b00001111);
	print_calc_time(minutes);
	displayString(":");
	uint8_t seconds = 10*((0b01110000 & get_sec)>>4) + (get_sec &
0b00001111);
	print_calc_time(seconds);

}


// MAINMETHOD
int main(void)
{
	maxTempIn =25;
	minTempIn =15;
	maxTempOut = 35;
	minTempOut = 0;

	val = 0;
	outOrIn = 1;
	minOrMax = 1;
	interTemp = 1;
	change = 0;

	_delay_ms(50);

	initDiods();

	initDisplay();
	displayOnOff();
	configureDisplay();
```

```
	clearDisplay();
	_delay_ms(50);
	displayString("Welcome!");

	setUpKeyboard();

	initThermal();

	i2c_init();
	set_clock();
	set_up_tick();
	enable_interupt();


	sei();

	while (1) {

		if(ticked==1){ //ta ny temperatur för att få medeltemperatur
			insertTempIn(tempIn);
			insertTempOut(tempOut);
			ticked = 0;
		}

		_delay_ms(10);

		if(knappflagg==1)
		{
			buttonPressed();
			knappflagg=0;
		}


	} //stänger while

} //stänger main
```

Appendix C

Manual för knappsatsen

- 0- Displayen sätts på och återställs till grundkonfigurering. Visar nuvarande datum och tid.
- 1- Visar nuvarande temperatur inne
- 2- Visar nuvarande temperatur ute
- 3- Visar högst uppmätta temperatur
- 4- Visar lägst uppmätta temperatur
- 5- Visar genomsnittlig dygnstemperatur inne
- 6- Visar genomsnittlig dygnstemperatur ute
- 7- Rensar medeltemperatursvektorn.
- 8- Bestämmer om man vill ändra temperaturen inne eller ute
- 9- Bestämmer om man vill ändra max- eller mintemperaturen
- A - Höjer varningsnivån med 1 celsius
- B - Sänker varningsnivån med 1 celsius
- C - Bestämmer om man vill väljer att ändra år, månad eller dag
- D - Ökar tid/datum
- E - Minskar tid/datum
- F - stänger ned displayen och startar om klockan