Väderstation

Anna Kjölstad Svedu och Emma Sikander, I09

The overall purpose of this project was to gain further knowledge of the making of digital devices, including the complexity of the process as well as the problems associated with digital product development. More specifically, the project consisted of making a functioning weather station. The requirements were that the weather station should be able to measure two different temperatures and print them to a display, and it should be possible to set an alarm for each temperature. Initially, the components were selected and a wiring diagram was constructed. Then, the different components were wired together and the software was programmed. The result was a weather station which fulfilled all of the requirements established in the beginning.

Digitala Projekt EITF11
Institutionen för Elektro- och Informationsteknik
Lunds Tekniska Högskola

Innehåll

Inledning
Detta projekt gick ut på att skapa en prototyp av en enkel väderstation som mäter två olika temperaturer och som man kan sätta larm på. Det övergripande syftet med projektet är att få insikt i komplexiteten och problematiken att skapa en elektronisk produkt. På vägen har gruppen även fått kunskaper inom C-programmering, elektronik, felsökning, datateknik och digitalteknik.

Krav
Väderstationen ska uppfylla följande krav:

- Ska kunna mäta två temperaturer, inne och ute temperatur, som ska skrivas ut på skärmen.
- Temperaturerna ska uppdateras kontinuerligt.
- Användaren ska kunna ställa in maximum och minimum temperaturer både för inne- och ute temperaturen.
- Då någon av minimum och maximum temperaturerna understigs/överstigs ska det visas ett meddelande på skärmen.

Hårdvara
Till vår produkt användes följande hårdvara:

Processor: AVR-ATmega16
AVR-ATmega16 är en åtta bitars mikroprocessor med 40 pinnar fördelade på 4 portar (A-D) och programmerbart minne på 16 kB.

Värmesensor: LM335
LM335 är en värmesensor som ger avläsning i Kelvin. Sensorn kan hantera temperaturer mellan -40°C och 100°C. I prototypen användes två olika sensorer. En av dem satt på själva ”kortet” och den andra sattes fast i en sladd. Detta för att man samtidigt skulle kunna mäta inne- och utetemperatur. Värmesensorerna kopplades till pinnar i A-porten på mikroprocessorn, eftersom dessa pinnar fungerar som A/D-omvandlare.

J-TAG
För att överföra C-programmet från datorn till processorn användes en J-TAG. Fyra pinnar på mikroprocessorns C-port var reserverade för J-TAG.

Knappsats
En knappsats med 16 knappar användes för att kunna ställa in larmet. Tio av knapparna används för att representera siffrorna 0-9, två knappar fick representera varsitt alarm (ute/inne), en knapp fick indikera att alarmet ska ha en max-gräns, en knapp fick indikera att alarmet skulle ha en min-gräns och en knapp motsvarade att man bekräftade inställningen (OK). Den sextonde knappen var överflödig och användes således inte. För att förbinda knappsatsen med mikroprocessorn användes pinnar på A- respektive C-porten.

Display: SHARP Dot-Matrix LCD Units

Denna display har plats till 80 tecken (80 x 8 bitar). 40 tecken fördelade på två rader syns på skärmen. Displayen kopplades in på både B-porten och D-porten. B-porten användes för att representera de tecken som skulle skrivas, medan pinnarna på D-porten användes för olika kommandon.

Arbetsprocessen

Idespåning och kravlista

Hela arbetsprocessen började med att fundera ut vad som skulle byggas. Det fastställdes att en väderstation skulle utgöra projektet. När detta var bestämt gjordes en kravlista för att tydliggöra produkten som skulle framställas.

Kopplingsschema

Utifrån kraven bestämdes vilka komponenter som skulle användas för att bygga upp väderstationen. Databladen för komponenterna lästes igenom för att få förståelse hur de skulle kopplas på lämpligt sätt, och sedan användes programmet PowerLogic för att rita upp ett kopplingsschema för hur allt skulle kopplas ihop med varandra. Se bilaga 1.

Sammansättning av komponenter

De olika komponenterna sattes fast på kopplingsplattan ganska så godtyckligt. Därefter kopplades sladdar mellan komponenterna enligt kopplingsschemat. Sladdarna som kopplade ihop komponenterna virades fast på komponenternas pinnar med hjälp av en trådvirare, vilket ger lika bra kontaktyta som vid lödning. Till alla kopplingar som gick direkt till spänningskällan användes en tjockare typ av tråd, som löddes fast. Då det var väldigt många trådar som skulle dras så användes olika färgkoder för att lättare hålla reda på de olika trådarna.

Då det blev en del felkopplingar övades det även på att ta bort befintliga lödningar och vira upp vissa kopplingar. Fortlöpande fick vissa mindre ändringar i kopplingsschemat göras, på grund av misstag av mindre allvarsam art.

Programmering

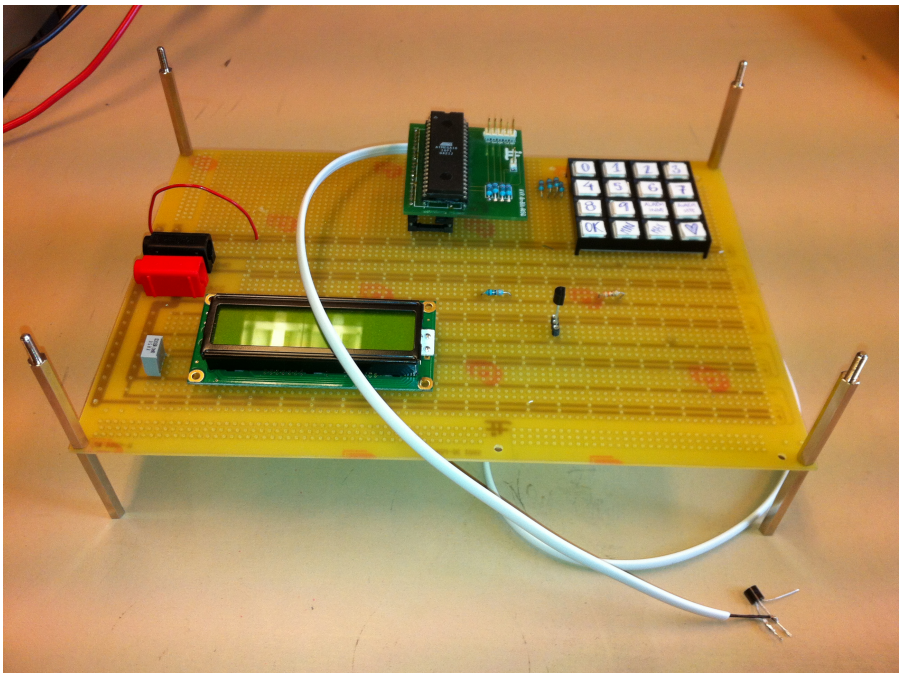
Denna del av projektet var det som tog längst tid.

Innan programmeringen började så testades komponenterna för att se att de var korrekt kopplade och att hårdvaran fungerade. Detta resulterade i att displayen fick bytas ut då den inte fungerade. Den nya skärmen var av en nyare modell och en ”kontrastredigerare” fick läggas till. Att få alla komponenter att kommunicera på ett tillfredställande sätt tog tid och mycket frustration uppstod. Databladen till komponenterna lästes nu väldigt noga i syfte att förstå precis hur de fungerar.

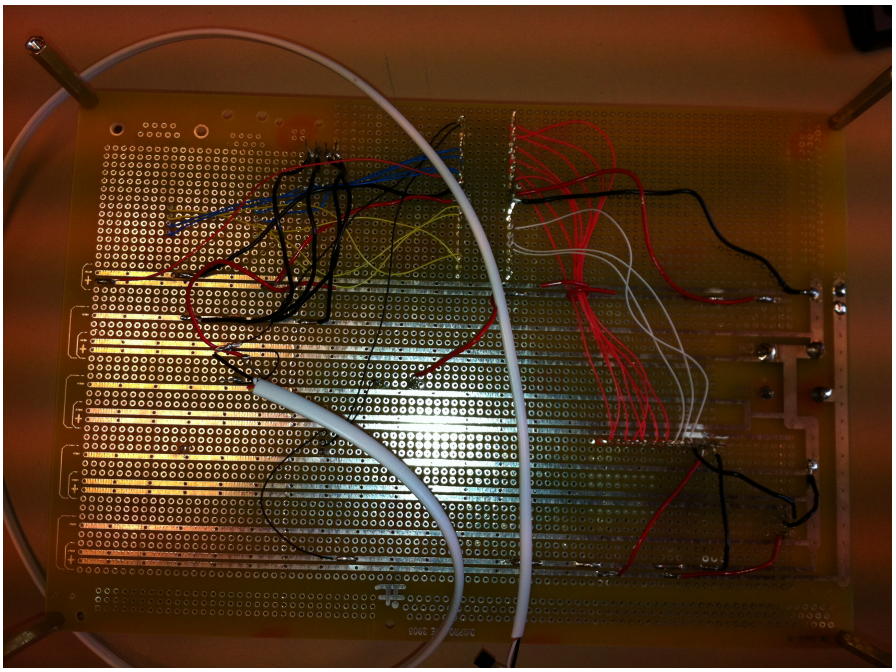
När testningen av komponenterna var klar påbörjades utformningen av mjukvaran. Det uppstod en hel del problem längs vägen och databladen granskades ytterligare en gång för att få ordning på problemen. Då ingen i gruppen hade tidigare erfarenhet av programmering i C, krävdes det även en del litteratur. Se referenslistan i slutet av rapporten för detaljer rörande det använda materialet. Den slutgiltiga källkoden återfinns i Bilaga 2.

Resultat

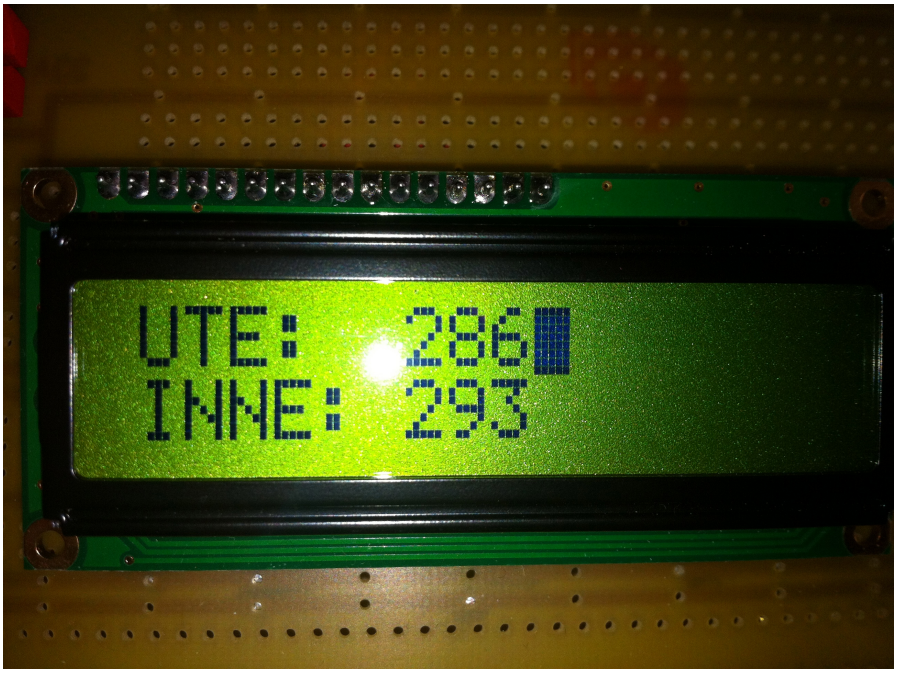
Efter sex veckors arbete stod slutversionen av prototypen klar. Den klarar av att mäta två temperaturer och skriva ut dessa på en display, samt ställa in max- respektive min-alarm för vardera temperatur. Därmed uppfyller den alla de fördefinierade kraven.



Figur 1: Den färdiga prototypen sedd ovanifrån.



Figur 2: Den färdiga prototypen sedd får undersidan, vilket åskådliggör alla de kopplingar som har gjorts.



Figur 3: Normalläge på displayen.

Figur 4: Hur displayen ser ut då ett min-alarm ställs in.

Figur 5: Hur displayen ser ut då ett max-alarm ställs in.

Slutsats

Vi gick in i projektet med mycket begränsade förkunskaper inom ämnesområdet, men lyckades trots detta bygga en fungerande prototyp. På vägen stötte vi på flertalet problem, men dessa bidrog väsentligen till att vi lärde oss väldigt mycket. Vi har också fått en större insikt i den komplexitet och den problematik som är relaterad till att ta fram en digital produkt, eftersom vi flera gånger under projektets gång har fått ändra kopplingar och tänka om. Vidare har vi även lärt oss vikten av att läsa datablad och manualer noggrant innan man börjar koppla ihop komponenterna.

Referenser

Manualer

Atmel 8-bit AVR Microcontroller with 16K byte In-system programmable flash, ATmega16, ATmega16L.

Datablad för temperaturgivare:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Sensors/lm335.pdf

Datablad för displayen:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf

Manualer för Power Logic:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/PDF_files/PowerLogic/Power_Logic.pdf

http://www.eit.lth.se/fileadmin/eit/courses/edi021/PDF_files/PowerLogic/Powerlogic.pdf

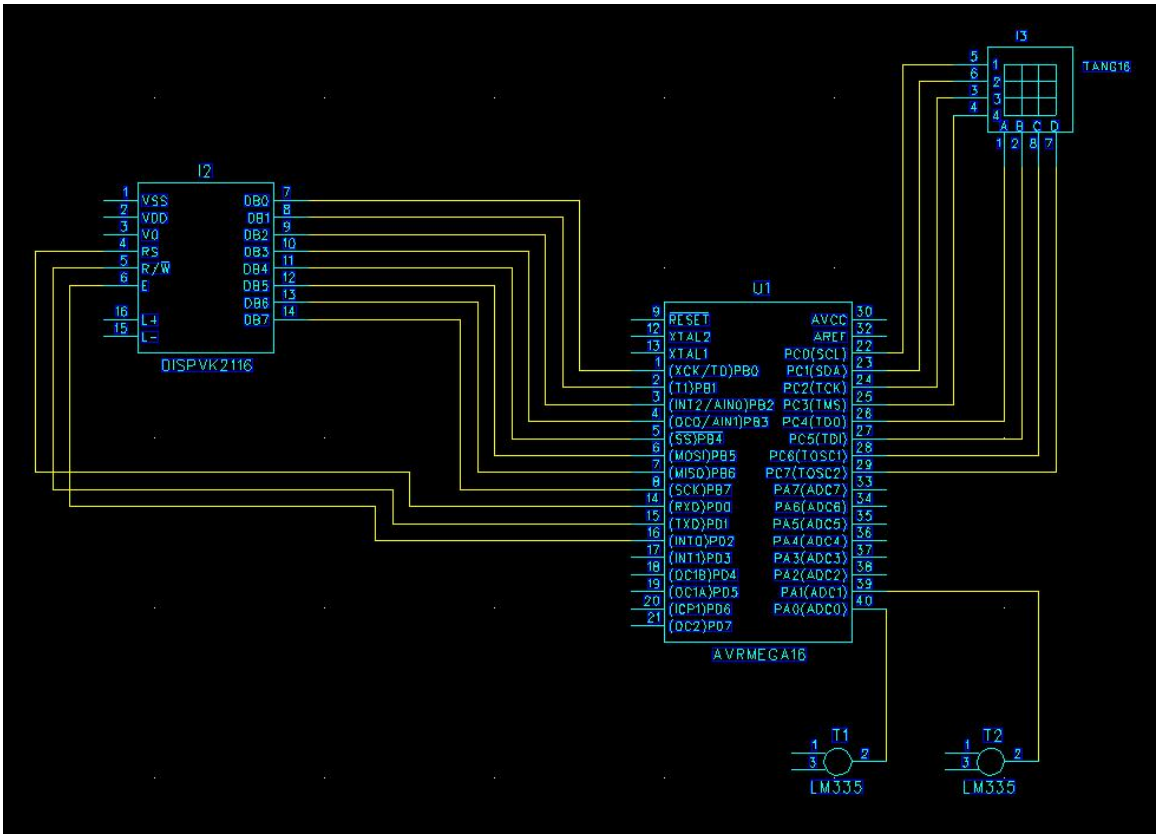
Litteratur

Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, AT&T Bell Laboratories, 1988.

Bilagor

Bilaga 1: Kopplingsschema

Nedan visas det kopplingsschema som gjordes i PowerLogic vid projektets början. Observera att det är viss skillnad mellan detta schema och hur prototypen i slutändan kopplades, på grund av att en del ändringar var tvungna att göras allteftersom.



Bilaga 2: Källkod

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <string.h>
#define F_CPU 8000000UL  // 8 MHz
#include <util/delay.h>

//********************VARIABLER*********************

char val2;
int temp1;
int temp11;
int temp2;
int temp22;
char temp;
int alarm1;
```

```
int alarm2;
int alarm1_type;
int alarm2_type;
char tempString[10];

//********************HJÄLPSATSER*********************

void set_pin(char port, char pin, char state){
	char set = 1 << pin;
	if(port == 'A'){
		set &= PORTA;
		if(set && !state){ //ändra från 1 -> 0
			PORTA ^= set;
		}
		if(set == 0 && state){ //ändra från 0 -> 1
			set = 1 << pin;
			PORTA ^= set;
		}
	}

	else if(port == 'B'){
		set &= PORTB;
		if(set && !state){ //ändra från 1 -> 0
			PORTB ^= set;
		}
		if(set == 0 && state){ //ändra från 0 -> 1
			set = 1 << pin;
			PORTB ^= set;
		}
	}

	else if(port == 'C'){
		set &= PORTC;
		if(set && !state){ //ändra från 1 -> 0
			PORTC ^= set;
		}
		if(set == 0 && state){ //ändra från 0 -> 1
			set = 1 << pin;
			PORTC ^= set;
		}
	}

	else if(port == 'D'){
		set &= PORTD;
		if(set && !state){ //ändra från 1 -> 0
			PORTD ^= set;
		}
		if(set == 0 && state){ //ändra från 0 -> 1
			set = 1 << pin;
```

```
                        PORTD ^= set;
                }
        }
}

//******************DISPLAY**********************

void write_cmd(short int cmd) {
        PORTB = cmd;
        set_pin('D', PD0, 0); //sätter RS till noll. Lyssna på hur den ska bete sig

        //toggla E
        set_pin('D', PD2, 0); //sätter E till noll
        set_pin('D', PD2, 1); //sätter E till ett
        set_pin('D', PD2, 0); //sätter E till noll

        _delay_ms(5);
        return;
}



void write_data(short int d) { //CG RAM/DD RAM Data Write
        PORTB = d;
        set_pin('D', PD0, 1); //satter RS till ett. lyssna på vad den ska skriva

        //toggla E
        set_pin('D', PD2, 0); //sätter E till noll
        set_pin('D', PD2, 1); //sätter E till ett
        set_pin('D', PD2, 0); //sätter E till noll

        _delay_ms(5);
        return;
}


void printString(short int address, char string[]) {
        _delay_ms(5);
        write_cmd(address); //börjar skriva på rätt ställe
        short int pos = 0; //positionen i vektorn
        while (string[pos] != 0x00) {  //går igenom hela vektorn
        write_data((short int)string[pos]); //skriver ut det som är på postionen
pos i vektorn
        pos++;
        }
}

void clear_display(){
        write_cmd(0b00000001);
```

```
}

void cursor_home() {
	write_cmd(0b00000010);
}

void display_on(){
	write_cmd(0b00001111);
}

void function_set(){
	write_cmd(0b00111000); //8-bitar och dual line
}

void entry_mode_set(){
	write_cmd(0b00000110);
}


void cursor_shift_right() {
	write_cmd(0b00010100);
}

void display_setup(){
	function_set();
	display_on();
	entry_mode_set();
	cursor_home();
	cursor_shift_right();
	clear_display();

}

//**********************KEYBOARD**************************


short int checkOutput(short int row, char val) {
	char c = val & 0xF0;
	if (c == 0x10){  //knappen pa forsta raden ar nertryckt
	return (row + 1);
	}
	else if (c == 0x20){ //knappen pa andra raden ar nertryckt
	return (row + 2);
	}
	else if (c == 0x80){ //knappen pa tredje raden ar nertryckt
	return (row + 3);
	}
	else if (c == 0x40){ //knappen pa fjarde raden ar nertryckt
	return (row + 4);
```

```
        }
        else  {
        return 0; //ingen knapp ar nertryckt
        }
}


short int checkY1() {
        set_pin('C', PC7, 0);
        set_pin('C', PC6, 0);
        set_pin('C', PC1, 0);
        set_pin('C', PC0, 1);
        char val = PINA;
        return checkOutput(10, val);
}

short int checkY2() {
        set_pin('C', PC7, 0);
        set_pin('C', PC6, 0);
        set_pin('C', PC0, 0);
        set_pin('C', PC1, 1);
        char val = PINA;
        return checkOutput(20, val);
}

short int checkY3() {
        set_pin('C', PC0, 0);
        set_pin('C', PC6, 0);
        set_pin('C', PC1, 0);
        set_pin('C', PC7, 1);
        char val = PINA;
        return checkOutput(30, val);
}

short int checkY4() {
        set_pin('C', PC7, 0);
        set_pin('C', PC0, 0);
        set_pin('C', PC1, 0);
        set_pin('C', PC6, 1);
        char val = PINA;
        return checkOutput(40, val);
}

short int checkKeys() {
        short int res = 0;
        res = checkY4();
        if (res != 0){
          return res;
        }
```

```
        res = checkY2();
        if (res != 0){
           return res;
        }
        res = checkY3();
        if (res != 0) {
           return res;
        }
        res = checkY1();
        if (res != 0) {
                return res;
        }
        return 0;

}


//**********************ALARMSET*********************


void alarm1_setup() {
        alarm1_type = 1;
        alarm1 = 5000;
}

void alarm2_setup() {
        alarm2_type = 1;
        alarm2 = 5000;
}

void set_alarm1(){
        printString(0b10000000, "                    ");
        printString(0xC0, "                    ");
        alarm1 = 0;
        int i = 1;
        printString(0b10000000, "MIN/MAX?");

        while(i == 1){
                switch(checkKeys()){
                        case 42:
                                alarm1_type = -1; //min-alarm
                                i = -1;
                                break;
                        case 43:
                                alarm1_type = 1; //max-alarm
                                i = -1;
                                break;

                }
```

```
	}

	printString(0b10000000, "SET ALARM IN: "); //alarm1


	while(1){
		_delay_ms(250);
	switch(checkKeys()) {
		case 11: //om man tryckt på knappen på första raden, första
kolumnen
			alarm1 = alarm1*10+0;
			printString(0xC0, itoa(alarm1, tempString, 10));
			break;

		case 12:
			alarm1 = alarm1*10+1;
			printString(0xC0, itoa(alarm1, tempString, 10));
			break;

		case 13:
			alarm1 = alarm1*10+2;
			printString(0xC0, itoa(alarm1, tempString, 10));
			break;

		case 14:
			alarm1 = alarm1*10+3;
			printString(0xC0, itoa(alarm1, tempString, 10));
			break;

		case 21:
			alarm1 = alarm1*10+4;
			printString(0xC0, itoa(alarm1, tempString, 10));
			break;
		case 22:
			alarm1 = alarm1*10+5;
			printString(0xC0, itoa(alarm1, tempString, 10));
			break;

		case 23:
			alarm1 = alarm1*10+6;
			printString(0xC0, itoa(alarm1, tempString, 10))
	break;
		case 24:
			alarm1 = alarm1*10+7;
			printString(0xC0, itoa(alarm1, tempString, 10));
			break;
		case 31:
			alarm1 = alarm1*10+8;
			printString(0xC0, itoa(alarm1, tempString, 10));
```

```
                break;
            case 32:
                alarm1 = alarm1*10+9;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;
            case 41:
                clear_display();
                return;

            }

        }

}


void set_alarm2(){
        printString(0b10000000, "                    ");
        printString(0xC0, "                    ");
        alarm2 = 0;
        int i = 1;
        printString(0x80, "MIN/MAX?");

        while(i == 1){
        switch(checkKeys()){
            case 42:        //Trycker på min knappen
                alarm2_type = -1;
                i = -1;
                break;
            case 43:        //trycker på max
                alarm2_type = 1;
                i = -1;
                break;
        }
}

        printString(0b10000000, "SET ALARM OUT: "); //alarm2


        while(1){
        _delay_ms(250);
        switch(checkKeys()) {
            case 11:
                alarm1 = alarm1*10+0;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;

            case 12:
                alarm1 = alarm1*10+1;
```

```
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;
            case 13:
                alarm1 = alarm1*10+2;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;

            case 14:
                alarm1 = alarm1*10+3;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;

            case 21:
                alarm1 = alarm1*10+4;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;

            case 22:
                alarm1 = alarm1*10+5;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;
            case 23:
                alarm1 = alarm1*10+6;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;

            case 24:
                alarm1 = alarm1*10+7;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;

            case 31:
                alarm1 = alarm1*10+8;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;

            case 32:
                alarm1 = alarm1*10+9;
                printString(0xC0, itoa(alarm1, tempString, 10));
                break;

            case 41:
                clear_display();
                return;

            }
        }
}
```

```
//**********************SENSORER*****************

int convert_to_temp(char in) {
	int res = in;
	return res/2+219;
}

void ADC_startup(){
	temp11=0;
	temp22=0;
	ADMUX = 0b00100000;
	SFIOR = 0b00000000;
	ADCSRA = 0b11011100;
}


ISR(ADC_vect){
	//val2=ADCH;
	//temp = val2 << 2;
	temp = ADCH;
	if (ADMUX == 0b00100001) { //Lyssnar på temperaturgivare
		temp2 = convert_to_temp(temp);
		if (temp2 != temp22){ //temperaturen har ändrats
			temp22=temp2;
			printString(0x80, "                ");
			printString(0x80, "UTE: ");
			printString(0x86, itoa(temp22, tempString, 10));
			//temperaturen har gått under minimum temperaturen och
larmet går igång
			if (alarm2_type == (int)-1 && temp22 <= alarm2){
				clear_display();
				printString(0b10000000, "TEMP UNDERSTIGIT ");
				printString(0xC0, itoa(alarm2, tempString, 10));
				while(1){
					switch(checkKeys()){
						case 41: //tryckt på OK knappen
								clear_display();
								_delay_ms(5);
								alarm2_setup();
								ADCSRA =
0b11011100;

	printString(0b10000000, "UTE: ");
								printString(0x86,
itoa(temp22, tempString, 10));
								printString(0xC0,
"INNE: ");
								printString(0xC6,
itoa(temp11, tempString, 10));
```

```
                                                                return;
                                        }
                                }
                        //temp har gått över maxtemp. Larmet aktiveras
                        }else if(alarm2_type == (int)1 && temp22 >= alarm2){
                                clear_display();
                                printString(0b10000000, "TEMP OVERSTIGIT ");
                                printString(0xC0, itoa(alarm2, tempString, 10));
                                while(1){
                                        switch(checkKeys()){
                                                case 41: //tryckt på OK knappen
                                                                clear_display();
                                                                _delay_ms(5);
                                                                alarm2_setup();
                                                                ADCSRA =
0b11011100;

        printString(0b10000000, "UTE: ");
                                                                printString(0x86,
itoa(temp22, tempString, 10));
                                                                printString(0xC0,
"INNE: ");
                                                                printString(0xC6,
itoa(temp11, tempString, 10));
                                                                return;
                                        }
                                }
                        }
                }
                ADMUX = 0b00100000;
                ADCSRA = 0b11011100;

        }



        else if (ADMUX == 0b00100000) {
                temp1 = convert_to_temp(temp);
                if (temp1 != temp11){//temperaturen har ändrats
                        temp11=temp1;
                        printString(0xC0, "                ");
                        printString(0xC0, "INNE: ");
                        printString(0xC6, itoa(temp11, tempString, 10));
                        if (alarm1_type == (-1) && temp11 <= alarm1){
                                clear_display();
                                printString(0b10000000, "TEMP UNDERSTIGIT :");
                                printString(0xC0, itoa(alarm1, tempString, 10));
                                while(1){
                                        switch(checkKeys()){
```

```
                                        case 41: //tryckt på OK knappen
                                                clear_display();
                                                _delay_ms(5);
                                                alarm1_setup();
                                                ADCSRA =
0b11011100;

        printString(0b10000000, "UTE: ");
                                                printString(0x86,
itoa(temp22, tempString, 10));
                                                printString(0xC0,
"INNE: ");
                                                printString(0xC6,
itoa(temp11, tempString, 10));
                                                return;

                                        }
                                }
                        }else if(alarm1_type == (1) && temp11 >= alarm1){
                                clear_display();
                                printString(0b10000000, "TEMP OVERSTIGIT ");
                                printString(0xC0, itoa(alarm1, tempString, 10));
                                while(1){
                                        switch(checkKeys()){
                                                case 41: //tryckt på OK knappen
                                                clear_display();
                                                _delay_ms(5);
                                                alarm1_setup();
                                                ADCSRA =
0b11011100;

        printString(0b10000000, "UTE: ");
                                                printString(0x86,
itoa(temp22, tempString, 10));
                                                printString(0xC0,
"INNE: ");
                                                printString(0xC6,
itoa(temp11, tempString, 10));
                                                return;

                                        }
                                }
                        }

                }
                ADMUX = 0b00100001;
                ADCSRA = 0b11011100;
        }
```

```
}

//************************MAIN*********************

void main(void){
	DDRA = 0b00000000;//sätter vad som ska var in/ut signal på A-porten
	DDRC = 0b11000011;//sätter vad som ska var in/ut signal på C-porten
	PORTC = 0b11000011;//sätter vad som ska var 1/0 på C-porten
	DDRB = 0b11111111;
	DDRD = 0b00000111;
	PORTD = 0b00000000;

	display_setup();

	alarm1_setup();
	alarm2_setup();

	ADC_startup();

	sei();

	while(1){

		switch(checkKeys()) {

			case 33:
				cli();
				set_alarm1();
				temp11 =0;
				temp22 = 0;
				ADCSRA = 0b11011100;
				sei();
				break;

			case 34:
				cli();
				set_alarm2();
				temp11=0;
				temp22=0;
				ADCSRA = 0b11011100;
				sei();
				break;

		}

	}

}
```