Temperaturgivare

Emilia Alfredsson, 900829-2022
Martina Persson Hollsten, 890411-4140
I-09

2012-05-13

Innehållsförteckning

Inledning

Kursen Digitala projekt EITF11 är en konstruktionskurs med syfte att ge grundkunskaper i digitalteknik och elektronik. Studenterna genomför ett valfritt projekt som ska leverera en fungerande prototyp och dokumentation kring projektet.

I detta projekt valdes att konstruera en väderstation. Utan att ha några förkunskaper i digitalteknik fanns ingen tydlig referensram om projektets svårighetsgrad. På grund av en väldigt hög inkörströskel krävdes det till en början mycket arbete och nedlagd tid för att enbart sätta sig in i problemet och förstå tankesättet. Detta resulterade i att vissa av kraven begränsades på grund av otillräckligt med tid för att färdigställa allt, och resultatet blev istället en väl fungerande temperaturgivare.

Kravspecifikation

- Kunna mäta aktuell temperatur.
- Visa meny med aktuell temperatur på en display.
- Visa meny med max- och mintemperatur på en display.
- Kunna byta mellan de två menyerna med ett knapptryck.
- Kunna nollställa max- och mintemperaturerna med ett knapptryck.

Eventuellt:

- Kunna beräkna den aktuella temperaturändringen med hjälp av aktuell temperatur och föregående samplade temperatur.
- Visa med en pil upp eller ner om temperaturen ökar respektive sjunker.
- Med ett knapptryck spela upp röst från ljudfil som meddelar aktuell temperatur.

Teori

De hårdvarukomponenter som har använts i projektet beskrivs nedan.

Processor

Processorn som används är en 8-bitars ATmega16 med en vald klockfrekvens på 8 MHz. Den innehåller en A/D-omvandlare som styrs av registerna ADMUX och ADCSRA. I övrigt har processorn 32 st I/O-pinnar varav 14 st används för att koppla in temperatursensorn, knapparna och displayen. Med hjälp av interfacet JTAG kan processorn programmeras i C på en dator.

Temperatursensor

Temperatursensorn LM335 ger en spänning som är proportionell mot temperaturen i Kelvin. En grad motsvarar 10 mV och på så sätt kan processorn beräkna aktuell temperatur och omvandla detta till Celsius.

LCD display

Displayen som används är en SHARP Dot-Matrix LCD Units. Som namnet beskriver bygger displayen upp varje tecken med en punktmatris. Varje bokstav som ska skrivas ut på skärmen skickas som 8 bitar från processorn.

Knappar

Till temperaturgivaren användes två knappar som fungerar likt två separata strömbrytare. Så fort en knapp trycks ner skickas en spänning in på den port som knappen är kopplad till. Vad som då ska ske är programmerat i processorn. Den röda knappens funktion är att byta mellan menyerna som visar aktuell temperatur respektive min- och maxtemperatur. Den svarta knappen påverkar endast menyn med min- och maxtemperaturen genom att återställa dessa.

Resistorer

En resistor kopplades till varje knapp för att se till att processorn alltid skulle ta emot antingen 1 eller 0 på portarna som var kopplade till knapparna. I annat fall hade processorn fått ”ingenting” vid uppsläppt knapp, vilket processorn inte kan hantera.

Genomförande

Projektet började med att vi tilldelades behövliga komponenter. Därefter var det dags att göra ett kopplingsschema på datorn som vi skulle kunna följa vid kopplandet av hårdvaran.

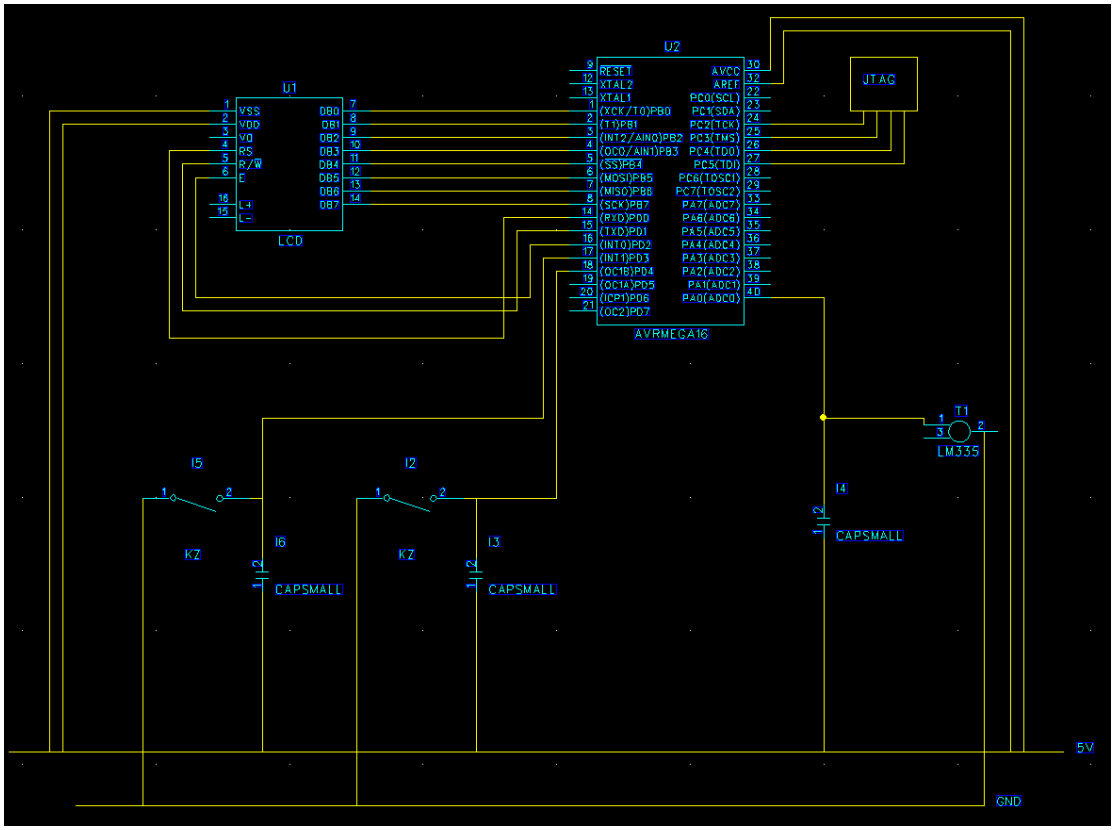

Bild 1. Kopplingsschema

För att förstå hur allt skulle kopplas läste vi igenom databladen för respektive komponent samt googlade en hel del. När schemat var klart virades och löddes alla trådar på plats. För att få processorn till att göra vad vi ville krävdes att den programmerades, vilket vi gjorde i C-kod i programmet AVR Studio 4.

Resultat

Resultatet av vårt arbete blev en väl fungerande temperaturgivare som uppfyller de grundläggande kraven som satts. Displayen kan skriva ut temperaturer med en noggrannhet på varannan grad och fungerar mycket bra då båda menyerna uppdateras korrekt. Den färdiga prototypen visas nedan.

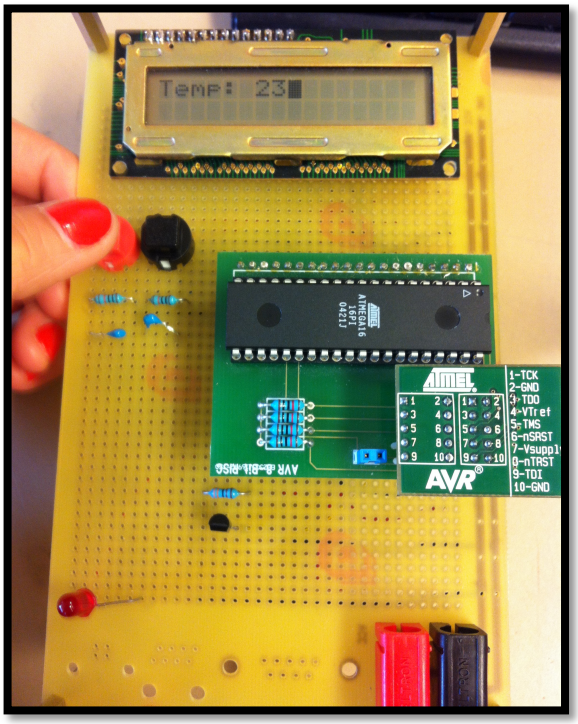

Bild 2. Aktuell temperatur

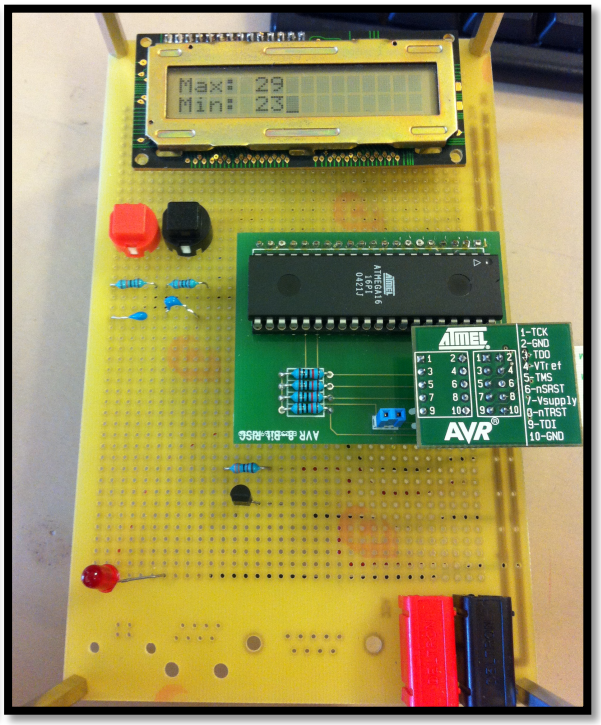

Bild 3. Min- och maxtemperatur

Diskussion

Detta arbete har varit enormt lärorikt. Vi har fått mycket nya kunskaper inom elektronik och datateknik inte bara på en teoretisk nivå utan även lärt oss om de hinder och svårigheter som kan uppkomma i praktiken.
Under ritningen av kopplingsschemat var vi mycket noggranna och läste mycket i datablad. Ändå blev det några smärre fel under själv konstruktionen som ställde till det för oss under testningen av prototypen. Det var inte svårt att lösa då vi kunde koppla om utan större komplikationer men det var lärorikt hur små slarvfel kan påverka mycket för de framtida stegen. Det är viktigt att vara noggrann både då man gör kopplingsschemat och bygger hårdvaran men man bör inte glömma att se över dem ett par gånger igen för att upptäcka eventuella fel.

I kravspecifikationen finns eventuella krav vid mån av tid. Tyvärr var tiden alltför knapp och dessa krav hann ej uppfyllas i denna kursomgång. Om det hade funnits mer tid hade vi gärna färdigställt alla krav samt även utvecklat en väderstation så att den visar datum och tid som kan sparas samt även mätning av vind, tryck, m.m. Då vi lärt oss väldigt mycket under detta projektet, framförallt hur vi ska ta oss an sådana här problem, så tror vi nog att vi hade kunnat få allt detta att fungera. Under projektet har många motgångar slutligen gett resultat och det var en stor glädje när allting väl fungerade.

Referenslista

Datablad, ATmega16 High-performance AVR 8-bit Microcontroller (Complete):
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf
Datablad, SHARP Dot-Matrix LCD Units, Alfanumerisk teckendisplay:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Display/LCD.pdf
Datablad, LM335 Kelvin temperature sensor:
http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Sensors/lm335.pdf

Appendix

Kod (C):

```
#include <avr/io.h>

#define F_CPU 8000000UL  //8MHz;

#include <util/delay.h>
#include <avr/interrupt.h>

void set_pin(char port, char pin, char state);

short int res;
int menu;
int temp;
int tempMin;
int tempMax;
short int flagRed;
short int flagBlack;
short int knapp1;
short int knapp2;
float temp1;
int minCount;



//---------------------------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 == '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;
      }
   }
}



//----------------------------INTERRUPTS-----------------------


ISR(ADC_vect) {
           res = ADCH;
           temp1 = (float)(res*1.953);
           temp = (temp1-273);
           ADCSRA = 0b11011100;
}


//-----------------------------DISPLAY-------------------------------


void disp_wait(){
 _delay_ms(100);
}


void write_cmd(short int val) {
 _delay_ms(5);
 PORTB = val;
 set_pin('D', PD0, 0); //RS är 0
 set_pin('D', PD2, 0);
 set_pin('D', PD2, 1);
 set_pin('D', PD2, 0);
 return;
}


void write_data(short int val) {
 _delay_ms(5);
 PORTB = val;
 set_pin('D', PD0, 1); //RS är 1
 set_pin('D', PD2, 0);
 set_pin('D', PD2, 1);
 set_pin('D', PD2, 0);
 return;
}
```

```
void disp_init() {
 write_cmd(0b00000001); //display clear
 disp_wait();
 write_cmd(0b00111000); //function set
 disp_wait();
 write_cmd(0b00001111); //display on
 disp_wait();
 write_cmd(0b00000110); //entry mode set
 disp_wait();
 write_cmd(0b00000010); //cursor home
 disp_wait();
}


void disp_clear(void) {
        write_cmd(0b00000001);
}



//----------------------TEMPERATUR----------------------


int temp_now_ten(int t){
        int rest = t;
        int count = 0;
        while (rest>=10) {
                rest = rest-10;
                count++;
        }
        return count;
}

int temp_now_one(int t) {
        int rest = t;
        while (rest>=10) {
                rest = rest-10;
        }
        return rest;
}



//----------------------SKRIVA UT----------------------


void set_minMax(){
 int t = temp;

 if(t>tempMax){
  tempMax=t;
 }
 if(t<tempMin){
  tempMin=t;
```

```
 }
}


void draw_minMax(){
 set_minMax();
 disp_clear();
 _delay_ms(5);
 write_data('M');
 write_data('a');
 write_data('x');
 write_data(':');
 write_data(' ');
 int ten1 = temp_now_ten(tempMax);
 int one1 = temp_now_one(tempMax);
 write_data(ten1+0x30);
 write_data(one1+0x30);
 write_cmd(0b11000000);
 _delay_ms(5);
 write_data('M');
 write_data('i');
 write_data('n');
 write_data(':');
 write_data(' ');
 int ten2 = temp_now_ten(tempMin);
 int one2 = temp_now_one(tempMin);
 write_data(ten2+0x30);
 write_data(one2+0x30);
}


void reset_minMax(){
 if(menu==1){
 tempMin=temp;
 tempMax=temp;
 disp_wait();
 disp_clear();
 _delay_ms(5);
 write_data('M');
 write_data('a');
 write_data('x');
 write_data(':');
 write_data(' ');
 int ten1 = temp_now_ten(tempMax);
 int one1 = temp_now_one(tempMax);
 write_data(ten1+0x30);
 write_data(one1+0x30);
 write_cmd(0b11000000);
 _delay_ms(5);
 write_data('M');
 write_data('i');
 write_data('n');
 write_data(':');
 write_data(' ');
```

```
 int ten2 = temp_now_ten(tempMin);
 int one2 = temp_now_one(tempMin);
 write_data(ten2+0x30);
 write_data(one2+0x30);
 }
}

void draw_homeMenu(){
 if(minCount==0){
        tempMin = temp;
        minCount++;
 }
 disp_clear();
 _delay_ms(5);
 write_data('T');
 write_data('e');
 write_data('m');
 write_data('p');
 write_data(':');
 write_data(' ');
 int ten = temp_now_ten(temp);
 int one = temp_now_one(temp);
 write_data(ten+0x30);
 write_data(one+0x30);
 _delay_ms(5);
}


void change_menu(){
        if(menu==1){
          _delay_ms(5);
                disp_clear();
                draw_homeMenu();
                _delay_ms(5);
                menu=0;
                _delay_ms(5);
   }
        else {
          _delay_ms(5);
        disp_clear();
                _delay_ms(5);
                draw_minMax();
                menu = 1;
                _delay_ms(5);
        }
}



//---------------------KNAPPAR----------------------


char pressedButtonRed(){
```

```
	if(flagRed==0){
	knapp1=PIND&0x10;
		if(knapp1 != 0x00){
			flagRed=1;
			change_menu();
		}
	}
	if(flagRed==1){
	knapp1=PIND&0x10;
		if(knapp1 == 0x00){
			flagRed=0;
		}
	}
}

char pressedButtonBlack(){

	if(flagBlack==0){
	knapp2=PIND&0x08;
		if(knapp2 != 0x00){
			flagBlack=1;
			reset_minMax();
		}
	}
	if(flagBlack==1){
	knapp2=PIND&0x08;
		if(knapp2 != 0x00){
			flagBlack=1;
		}
		if(knapp2 == 0x00){
			flagBlack=0;
		}
	}

}



//----------------------MAIN-------------------------


void main() {

	DDRA = 0b00000010;
	DDRB = 0b11111111;
//	DDRC = 0b00000000;
	DDRD = 0b00000111;  //sätter vad som ska var in/ut signal på D-porten

	// PORTD = 0b00000000; //sätter vad som ska var 1/0 på D-porten

	set_pin('A', PA1, 1);  //gör att lampan lyser.
	set_pin('A', PA1, 0);  //gör att lampan släcks.
```

```
        disp_init();
        sei();
        menu=1;
        flagRed=0;
        flagBlack=0;


        ADCSRA = 0b11011100;
        ADMUX = 0b01100100;
        SFIOR = 0b00000000;

        while(1){
                pressedButtonRed();
                pressedButtonBlack();
        }
}
```