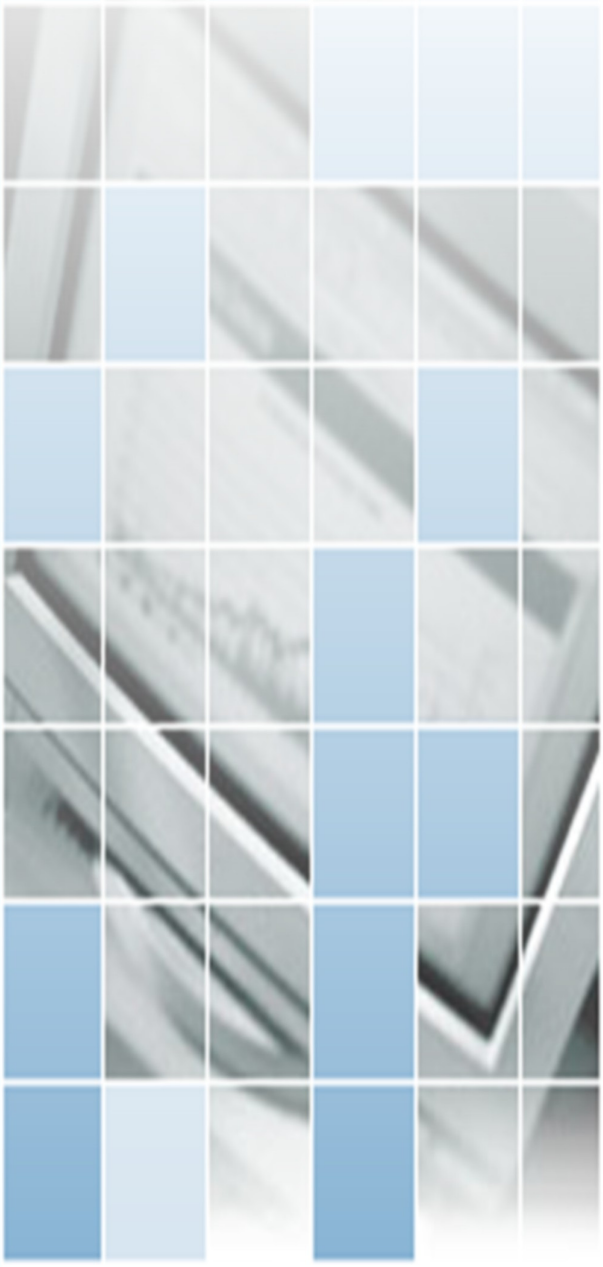
Eva Fancev
0036413308

Fakultet elektrotehnike i računarstva, Sveučilišta u Zagrebu
Zavod za elektroničke sustave i obradbu informacija
Sveučilište u Zagrebu

Sustav za automatsku regulaciju klimatizacije



- Ovaj tekst namijenjen je studentima koji slušaju predmet SPVP
- Za njegovo razumijevanje potrebno je predznanje iz elektronike, digitalne logike i programiranja u C-u
- Nudi informacije o tome kako napraviti sustav za upravljanje klimatizacijom

21. lipanj

Sažetak

Smisao je inteligentne kuće da učini život udobnijim. Sustav za upravljanje klimatizacijom tome uvelike pomaže time što ukućani u svim prostorijama mogu prilagoditi temperaturu svojim željama. Željenu temperaturu zadaju ukućani daljinskim upravljačem, na osnovu čega sustav upravlja klimatizacijom. Trenutnu temperaturu sustav očitava pomoću digitalnih senzora, a ispisuje na kompatibilnom LCD-u. Tako se ispisuje i trenutno stanje klimatizacije, kao i informacija raste li ili pada trenutna temperatura.

Sadržaj

1. UVOD 3
2. OPIS SUSTAVA 4
1.1. Prikaz informacija na LCD-u 5
1.2. Ispis i nadzor temperature 5
1.3. Podešavanje referentne temperature 6
3. ZAKLJUČAK 8
4. LITERATURA 8
5. POJMOVNIK 9
DODATAK A: OPIS TEMPERATURNOG SENZORA 10
DODATAK B: PROGRAMSKI KÔD 12

Ovaj seminarski rad je izrađen u okviru predmeta „Sustavi za praćenje i vođenje procesa“ na Zavodu za elektroničke sustave i obradbu informacija, Fakulteta elektrotehnike i računarstva, Sveučilišta u Zagrebu.

1. Uvod

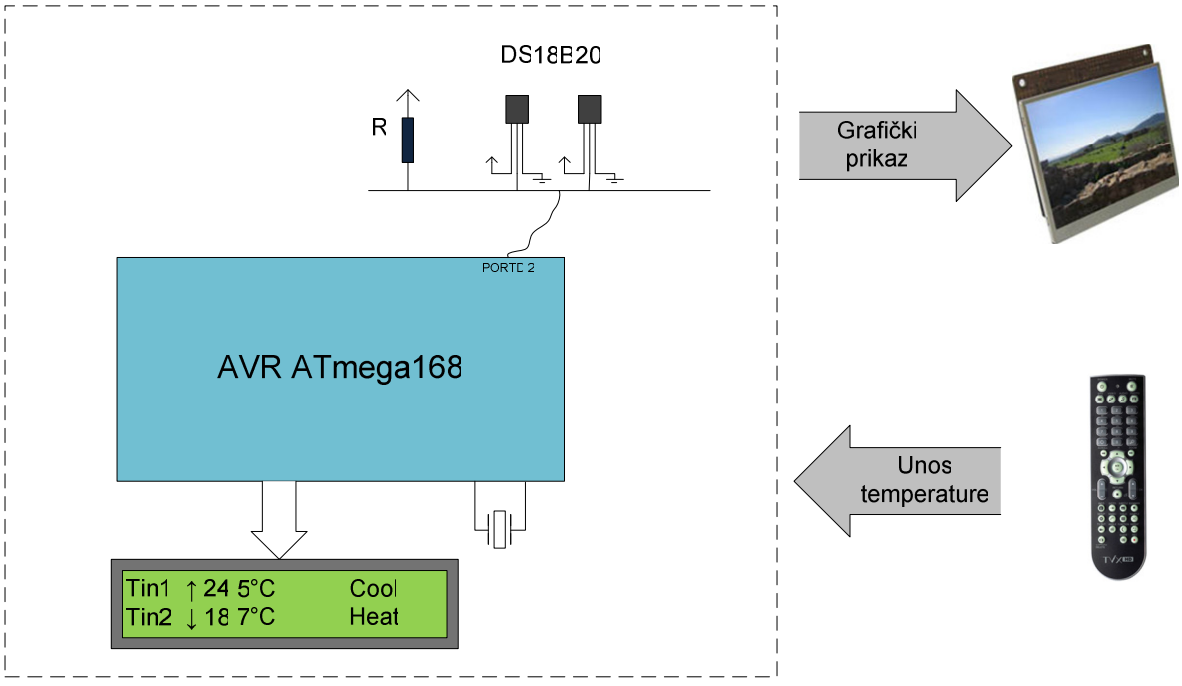
Inteligentna kuća prikuplja informacije o ukućanima, a potom se pokušava prilagoditi njihovim navikama i željama. Temeljna je ideja život učiniti udobnijim. Ona omogućuje obavljanje rutinskih zadataka s puno manje napora kao i odgovarajuće korištenje prirodnih resursa. Najveći je problem, a time i veliki nedostatak, naučiti se koristiti svim pogodnostima koje kuća nudi. No zato su prednosti svakojake. Važno je istaknuti sigurnost, uštedu energije i općenito poboljšanje kvalitete života.

Sustav za upravljanje klimatizacijom samo je jedna od pogodnosti koje nudi inteligentna kuća. Ona omogućuje da nam kompletno okruženje postane središnje i daljinski kontrolirano. Samostalno obavljanje kupovine, štednja energije, upravljanje kućom putem web poslužitelja, automatska regulacija rasvjete, namještanje posebnih scena za romantičnu večeru itd., boravak u inteligentnoj kući zasigurno čine ugodnijim. Osjećaj sigurnosti stanarima pruža zaštita u slučaju požara, video nadzor, simulacija prisutnosti, nadzor vrata i prozora kao i detekcija moguće provale, medicinski alarm i slično. Uz automatsko hranjenje kućnih ljubimaca i navodnjavanje biljki, postaje nemoguće zaboraviti nahraniti psa ili mačku ili pak zaliti cvijeće. Inteligentna kuća može nuditi i pomoć starijim i nemoćnim osobama. Krajnji cilj kuće je poboljšati kvalitetu života, ali uz jednostavnost korištenja za korisnika.

Nekada je bilo nezamislivo sjediti u kući na ugodnoj temperaturi koju smo pritom sami odredili, dok je vani iznad 30 °C. U budućnosti će se automatska regulacija temperature moći oslanjati i na vremensku prognozu i ostale faktore. U nastavku teksta opisan je sustav za automatsku regulaciju klimatizacije.

2. Opis sustava

Blok shema sustava prikazana je na slici 1. Realizirani sustav za automatsku regulaciju klimatizacije odnosi se na dio označen isprekidanom linijom. Temelji se na Atmel AVR mikrokontroleru (ATmega168). Kako bi se omogućio nadzor temperature, mikrokontoler komunicira s dva digitalna temperaturna senzora DS18B20 tvrtke Dallas Semiconductor. Opis i princip rada senzora nalazi se u prilogu A. Komunikacija sa senzorima je potpuno digitalna što eleminira potrebu za analognom elektronikom (npr. AD pretvornicima i slično). Prikaz temperatura i ostalih informacija omogućuje standardni HD44780 kompatibilan LCD ekran. Cijeli softver razvijen je u alatu „Atmel IDE - AVR Studio".



Slika 1. Blok shema sustava

Informacije sa senzora koje odlaze na centralni sustav za praćenje kuće, može iskoristiti protupožarni sustav s ciljem bolje zaštite. Također, te informacije bi mogao koristiti i grafički prikaz kuće kako bi mogao ispisivati trenutne temperature. A predviđeno je da se oslanja i na projekt „Primanje naredbi putem IR signala" kako bi se unos željene temperature (referentne za upravljanje klimatizacijom) mogao obavljati putem

standardnog IR daljinskog upravljača. Sustav s IR mogućnostima može slati željene temperature putem UART-a.

1.1. Prikaz informacija na LCD-u

Slika 2. Prikaz informacija na LCD-u

Slika 2 prikazuje izgled korisničkog sučelja. Svaki od dva retka odgovara jednom senzoru i jednoj klimatizacijskoj jedinici. Na samom početku potrebno je napraviti nekoliko inicijalizacija:

- Inicijalizirati LCD ekran
- Inicijalizirati senzore
- Inicijalizirati UART i omogućiti prekide
- Upisati specijalne znakove (na slici 2 se mogu vidjeti znakovi koji ne pripadaju ASCII tablici te ih je potrebno poslati LCD kontroleru)

1.2. Ispis i nadzor temperature

Cijeli algoritam ispisa i nadzora temperature opisan je sljedećim pseudo-kôdom.

```
for(;;)
{
	if(flag==0)
	{
		//Započni novu konverziju na svim senzorima
		flag=1;
	}

	if(konverzija završena)
	{
		flag=0;
		//Očitaj temperaturu s 1. senzora
		//Podesi strelice
		//Provjeri potrebu za klimatizacijom

		//Ponovi sve isto za drugi senzor
	}
}
```

Zastavica „flag" služi kao značka (engl. *token*) koja onemogućuje nove naredbe za pretvorbu temperature nakon što su one već poslane.

Kao što se vidi na slici 2 senzor pruža rezoluciju manju od 1°C. Strelica lijevo od ispisa temperature predstavlja informaciju o kretanju temperature. Ukoliko temperatura raste, strelica pokazuje prema gore i obratno. No problem je što i vrlo mali povjetarac tako može utjecati na fluktuaciju temperature. Iz tog razloga napravljena je histereza od jednog stupnja za promjenu stanja strelice. Jasno je da primjerice temperaturu 24.9°C nije dovoljno gledati samo kao cjelobrojnu vrijednost izmjerene temperature. Programski odsječak koji to radi može se pogledati u dodatku gdje se nalazi cijeli kôd.

Na kraju svakog mjerenja izmjerena temperatura uspoređuje se sa željenom i prema tome podešava stanje klimatizacije. Trenutno stanje klimatizacije može biti „Heat", „Cool" ili „Ok". Koriste se engleski nazivi jer hrvatski ne stanu na ekran. Kao i u slučaju strelica ugrađena je histereza koja se programski podešava po potrebi. U suprotnom bi se klima previše palila i gasila što bi znatno skratilo njezin MTBF (engl. *Mean time before failures*). Signal *Heat* može se spojiti na grijače, a *Cool* na sustav za hlađenje.

1.3. Podešavanje referentne temperature

U nastavku je opisan način podešavanja referentne temperature za klimatizaciju. Kao što je već spomenuto, podaci o referentnoj temperaturi dohvaćaju se putem UART-a od sustava koji posjeduje IR mogućnosti. UART radi u prekidnom načinu rada. Dakle svaki primljeni podatak na UART-u uzrokuje prekid.

```
ISR(USART_RXC_vect)
{
	if(UDR0==1 || UDR0==2)
		klima=UDR0;
	else
	{
		if(klima==1)
			reftemp1=UDR0;
		else
			reftemp2=UDR0;
	}
}
```

Referentne temperature su spremljene u dvije globalne varijable dostupne svim funkcijama u kôdu. Globalna varijabla „klima" omogućuje pamćenje o kojoj se klimi radi. Naime, slijed koji drugi sustav šalje uključuje 2 okteta: indeks klime (brojevi 1 ili 2) i referentnu temperaturu. Pretpostavka je da referentna temperatura nikada nije 1 odnosno 2°C. Nakon što prvi oktet podesi indeks klime, drugi se izravno upisuje u odgovarajuću varijablu ovisno o indeksu.

3. Zaključak

Opisanim sustavnom mjeri se i prikazuje temperatura i stanje grijanja odnosno hlađenja. Željenu temperaturu unosi korisnik i tako sam upravlja klimatizacijom. Stoga rezultat projekta ne rješava pitanje same procjene temperature prema navikama ukućana. Također ne prati vanjske trendove temperature što bi primjerice mogao biti korak dalje. Dakle u budućnosti bi se upravljanje klimatizacijom moglo oslanjati i na vremensku prognozu. Rad klima uređaja trebao bi se prilagođavati vanjskoj temperaturi i vlažnosti zraka, a sve u svrhu uštede energije. Temperatura bi se trebala regulirati u takvim granicama koje ne bi degradirale unutarnji ambijent. Klimatizacijom bi se moglo upravljati i resetiranjem temperature klima uređaja na preporučenu, primjerice svakih sat vremena. Rad uređaja mogao bi se regulirati i tako da radi samo određeno vrijeme (korištenjem pauza). Poanta je štedjeti energiju sprečavanjem da se prostorija previše zagrije ili previše ohladi u slučaju da ukućanin zaboravi ugasiti klima uređaj. U današnje se vrijeme klima uređaji upotrebljavaju u većini kućanstava stoga je ušteda energije neophodna radi sprečavanja globalnog zatopljenja.

4. Literatura

[1] HD44780 (LCD-II), Dot Matrix Liquid Chrystal Display Controller/Driver, datasheet, Hitachi, 1998.

[2] Atmel: 8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash, 2009.

[3] Dallas Semiconductor: DS18B20 Programmable Resolution 1-Wire Digital Thermometer, Maxim Integrated Products, 2008.

[4] Arduino. URL: http://arduino.cc/en/Main/ArduinoBoardDuemilanove (2005)

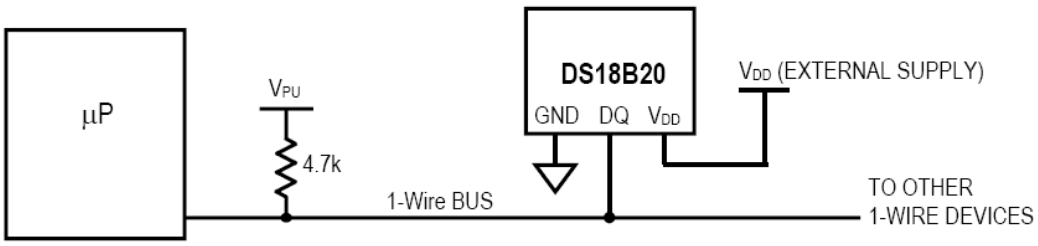
[5] Arduino-shematic. URL: http://www.pdfdownload.org/pdf2html/pdf2html.php?url=http%3A%2F%2Farduino.cc%2Fen%2Fuploads%2FMain%2Farduino-duemilanove-schematic.pdf&images=yes (2009)

5. Pojmovnik

Pojam	Kratko objašnjenje	Više informacija potražite na
LCD	Engl. *liquid crystal display*. Ekran koji se temelji na tehnologiji tekućih kristala.	http://hr.wikipedia.org/wiki/LCD
HD44780	Hitachijev kontroler koji predstavlja jedini standard za sve numeričke LCD ekrane danas	http://en.wikipedia.org/wiki/HD44780_Character_LCD
MTBF	Engl. *Mean time before failures.* Prosječno vrijeme prije kvara.	http://en.wikipedia.org/wiki/MTBF
IDE	Engl. Integrated development environment	http://en.wikipedia.org/wiki/Integrated_development_environment
1-wire	Komunikacijski protokol razvijen u tvrtci Dallas Semiconducter koji definira digitalnu komunikaciju putem samo jedne žice	http://en.wikipedia.org/wiki/1_wire_bus
UART	Engl. *universal asynchronous receiver/transmitter*. Sklop za serijsku komunikaciju i kontrolu priključka.	http://en.wikipedia.org/wiki/UART
IR	Engl. *Infrared.* Infracrveni dio spektra elektromagnetskog zračenja valne duljine od 700 nm do 1 mm.	http://en.wikipedia.org/wiki/Infrared
ASCII	Engl. *American Standard Code for Information Interchange.* Način kodiranja znakova temeljen na engleskoj abecedi.	http://hr.wikipedia.org/wiki/ASCII

Dodatak A: Opis temperaturnog senzora

Digitalni senzor temperature DS18B20 rezolucije 9-12 bita se spaja pomoću *1-wire* protokola koji koristi samo jednu liniju (DQ) za prijenos podataka. Naravno, potrebno je spojiti napajanje što standardno zahtijeva još dvije žice. Zanimljivo je da senzor može raditi i bez spajanja linije napajanja na način da krade naboj s DQ linije u trenucima kada je ona visoko (tzv. „parazitno napajanje") pri čemu se VDD izvod spaja na GND. U projektu je korišteno standardno napajanje preko VDD i GND priključaka. Shema spoja temperaturnog senzora na Arduino prikazana je na slici 3.

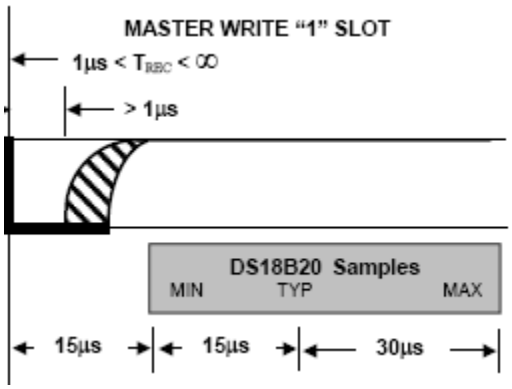


Slika 3. Shema spoja temperaturnog senzora na sustav

Dodatni otpornik služi za pritezanje linije na napajanje (visoko) kada sklopovi svoje izvode, na koje je spojen DQ, postave u stanje visoke impedancije.

Oba uređaja mogu postaviti liniju nisko slanjem logičke nule, dok se linija postavlja visoko pomoću priteznog otpornika kada uređaj koji šalje bit postavi izvod u stanje visoke impedancije (tzv. otpuštanje linije).

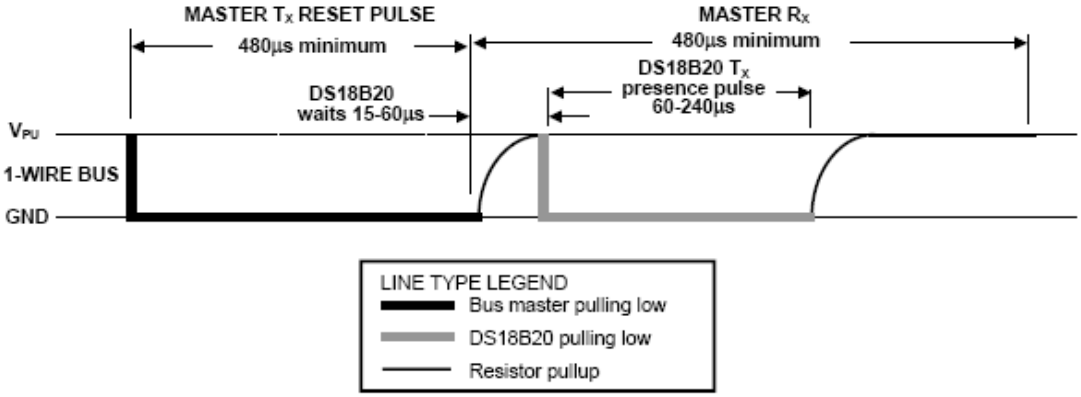
Za čitanje i pisanje podataka preko jedne linije definirani su vremenski prozori koji definiraju način slanja bitova 0 odnosno 1. Tijekom trajanja jednog prozora, moguće je poslati samo jedan bit. Dva su tipa vremenskih prozora za pisanje podataka, logičke 0 i 1, trajanja minimalno 60 µs s minimalnom pauzom od 1 µs. Oba se tipa inicijaliziraju tako da *master* pritegne liniju nisko. Pisanje logičke jedinice prikazuje slika 4.



Slika 4. Prikaz pisanja logičke jedinice

Izvod DQ je definiran kao izlazni i *master* šalje logičku nulu (linija pritegnuta na masu) koja se na sabirnici mora zadržati minimalno 1 µs. *Master* otpušta liniju u 15 µs (DQ se postavlja u stanje visoke impedancije) nakon čega se ona samostalno priteznim otpornikom diže visoko. 15 do 30 µs nakon što senzor detektira padajući brid signala na sabirnici (*master* poslao nulu), senzor očitava visoko stanje sabirnice u kojem ostaje još minimalno 60 µs.

Svaka nova komunikacija sa senzorom započinje na način da *master* inicijalizira sve uređaje slanjem *reset* impulsa trajanja minimalno 500 µs, nakon čega *slave* odgovara *presence* impulsom čime pokazuje svoju spremnost (Slika 5).



Slika 5. Inicijalizacija senzora

Nakon što *master* detektira *presence* impuls, spreman je odaslati ROM naredbu. Svaki slave uređaj ima jedinstveni 64-bitni ROM serijski broj pomoću kojega ga *master* može prepoznati. Postoji pet ROM naredbi duljine 8 bita i *master* mora koristiti odgovarajuću prije nego što pošalje funkcijsku naredbu poput pretvorbe temperature, čitanja temperature i sl.

Dodatak B: Programski kôd

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "Opcenite.h"
#include "Dallas.h"
#include "LCD.h"

//Funkcije
void AddressDevice(const unsigned char *ROM);
void LCDIspisBCD(char data);
void Error(void);
void LCDCool(unsigned char index);
void LCDHeat(unsigned char index);
void LCDOk(unsigned char index);
void USART_Init(unsigned int ubrr);
char USARTReadChar(void);
void USARTWriteChar(char data);

//Globalne varijable
char reftemp1=29, status1=0, reftemp2=29, status2=0;
unsigned char klima;

#define FOSC 16000000 // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1

int main(void)
{
	unsigned char temp,dec,pom,i,flag=0;
	char ptemp1=0,ptemp2=0,pdec1=0,pdec2=0;
	const char specchar[]={0x04,0x24,0x44,0x64,0x95,0xAE,0xC4,0xE0,
	//Strelica dolje

0x04,0x2E,0x55,0x64,0x84,0xA4,0xC4,0xE0,	//Strelica gore

0x08,0x34,0x48,0x60,0x80,0xA0,0xC0,0xE0};	//Stupnjevi
	const char
dec_table[]={'0','0','1','2','2','3','4','4','5','6','6','7','8','8','9','9'}
;

	//const unsigned char DS1[]={0xF9,0x00,0x00,0x00,0xF5,0x81,0x1D,0x28};
	const unsigned char DS1[]={0x56,0x00,0x00,0x00,0xCC,0x96,0x1E,0x28};
	//const unsigned char DS2[]={0x2D,0x00,0x00,0x00,0xF5,0xBC,0x62,0x28};
	const unsigned char DS2[]={0x98,0x00,0x00,0x00,0xF5,0x94,0x17,0x28};
	//Inicijalizacije
	LCDInit();
	USART_Init(MYUBRR);

	//PD2 ulazni + pull-up
	DDRD &=~(1<<PD2);
	PORTD|=(1<<PD2);

	//Ispis pocetne poruke @ 5sec *
	//*****************************
	LCDSendText("     Eva Fancev 2009",1);
	LCDSendText("           SPVP",2);
	delay(0x9896, 0x05);
	LCDClear();
```

```
//Specijalni znakovi *
//**********************
LCDSendData(0x40, 0);            //Set CGRAM address
for(i=0; i<24; i++)
       LCDSendData(specchar[i], 1);

LCDSendText("Tin1:",1);
LCDSendText("Tin2:",2);

sei();

for(;;)
{
       //Zapocni novu konverziju
       if(flag==0)
       {
              WireInit();
              command(0xCC);
              command(0x44);
              flag=1;
       }
       //Ocitaj i ispisi temperaturu
       DDRD&=~(1<<PD2);
       if(PIND & (1<<PD2))
       {
              flag=0;
              //Ocitaj s prvog senzora
              AddressDevice(DS1);
              command(0xBE);
              temp=read_owi();
              dec=temp & 0x0F;
              temp=temp >> 4;
              pom=read_owi();
              pom=pom << 4;
              temp=temp | pom;
              //Ispis strelice
              LCDSendData(0x85,0);
              if((temp>ptemp1 && dec>pdec1) || ((temp-ptemp1)>=2))
              {
                     LCDSendData(0x01,1);
                     ptemp1=temp;
                     pdec1=dec;
              }

              else if(temp<ptemp1 && dec<pdec1 || ((ptemp1-temp)>=2))
              {
                     LCDSendData(0x00,1);
                     ptemp1=temp;
                     pdec1=dec;
              }
              else
                     LCDSendData(0x86,0);
              //Ispis minusa
              if(temp & 0x80)
              {
                     temp=~temp+1;
                     LCDSendData('-',1);
              }
              else
                     LCDSendData(' ',1);
              //Ispis temperature
              LCDSendData((temp/10)+48,1);
```

```
LCDSendData((temp%10)+48,1);
//Decimalna tocka
LCDSendData('.',1);
//Decimala
LCDSendData(dec_table[dec],1);
//Stupanj Celzij
LCDSendData(0x02,1);
LCDSendData('C',1);

//Upravljanje klimom 1
if (temp > reftemp1)
{
       LCDCool(0x91);
       status1=1;
}
if (status1==1 && temp < reftemp1)
{
       LCDOk(0x91);
       status1=0;
}
if (temp < reftemp1 -1)
{
       LCDHeat(0x91);
       status1=2;
}
if (status1==2 && temp == reftemp1)
{
       LCDOk(0x91);
       status1=0;
}

//Ocitaj s drugog senzora
AddressDevice(DS2);
command(0xBE);
temp=read_owi();
dec=temp & 0x0F;
temp=temp >> 4;
pom=read_owi();
pom=pom << 4;
temp=temp | pom;
//Ispis strelice
LCDSendData(0xC5,0);
if((temp>ptemp2 && dec>pdec2) || ((temp-ptemp2)>=2))
{
       LCDSendData(0x01,1);
       ptemp2=temp;
       pdec2=dec;
}
else if(temp<ptemp2 && dec<pdec2 || ((ptemp2-temp)>=2))
{
       LCDSendData(0x00,1);
       ptemp2=temp;
       pdec2=dec;
}
else
       LCDSendData(0xC6,0);
//Ispis minusa
if(temp & 0x80)
{
       temp=~temp+1;
       LCDSendData('-',1);
}
else
```

```
                        LCDSendData(' ',1);
                    //Ispis temperature
                    LCDSendData((temp/10)+48,1);
                    LCDSendData((temp%10)+48,1);
                    //Decimalna tocka
                    LCDSendData('.',1);
                    //Decimala
                    LCDSendData(dec_table[dec],1);
                    //Stupanj Celzij
                    LCDSendData(0x02,1);
                    LCDSendData('C',1);

                    //Upravljanje klimom 2
                    if (temp > reftemp2)
                    {
                        LCDCool(0xD1);
                        status2=1;
                    }
                    if (status2==1 && temp == reftemp2)
                    {
                        LCDOk(0xD1);
                        status2=0;
                    }
                    if (temp < reftemp2 -1)
                    {
                        LCDHeat(0xD1);
                        status2=2;
                    }
                    if (status2==2 && temp == reftemp2)
                    {
                        LCDOk(0xD1);
                        status2=0;
                    }

                }
        }
        return 0;
}

void AddressDevice(const unsigned char *ROM)
{
        signed char i;
        if(WireInit())
        {
                command(0x55);
                for(i=7; i>=0; i--)
                        command(ROM[i]);
        }
        else
        {
                LCDClear();
                LCDSendText("ERROR: Dallas One Wire",1);
                while(1);
        }
}

void LCDIspisBCD(char data)
{
        LCDSendData((data>>4)+48,1);
        LCDSendData((data & 0x0F)+48,1);
}
```

```
void Error(void)
{
	LCDClear();
	hexprint(TWSR>>4);
	hexprint(TWSR & 0x0F);
	while(1);
}

void LCDCool(unsigned char index)
{
	LCDSendData(index, 0);
	LCDSendData('C',1);
	LCDSendData('o',1);
	LCDSendData('o',1);
	LCDSendData('l',1);
}

void LCDHeat(unsigned char index)
{
	LCDSendData(index, 0);
	LCDSendData('H',1);
	LCDSendData('e',1);
	LCDSendData('a',1);
	LCDSendData('t',1);
}

void LCDOk(unsigned char index)
{
	LCDSendData(index, 0);
	LCDSendData(' ', 1);
	LCDSendData('O', 1);
	LCDSendData('k', 1);
	LCDSendData(' ', 1);
}

void USART_Init(unsigned int ubrr)
{
	//Set baud rate
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;

	//Enable receiver and transmitter
	UCSR0B = (1<<RXEN0)|(1<<TXEN0);

	//Set frame format: 8data, 2stop bit
	UCSR0C = 0x06;

	UCSR0B |= (1 << RXCIE0);
}

void USARTWriteChar(char data)
{
   while(!(UCSR0A & (1<<UDRE0)))
   {
      //Do nothing
   }

   UDR0=data;
}



char USARTReadChar()
```

```
{
   while(!(UCSR0A & (1<<RXC0)))
   {
      //Do nothing
   }
   return UDR0;
}

ISR(USART_RXC_vect)
{
   if(UDR0==1 || UDR0==2)
       klima=UDR0;
   else
   {
       if(klima==1)
              reftemp1=UDR0;
       else
              reftemp2=UDR0;
   }
}
```

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

//**************************************
//        Dallas 1-wire driver
//**************************************
#define PORT_DS PORTD
#define DDR_DS  DDRD
#define PIN_DS  PIND
#define PAD 0x04
#define NPAD 0xFB

//Inicijalizira 1 wire sabirnicu
char WireInit(void)
{
       DDR_DS=DDR_DS | PAD;       //Pin izlazni i nisko
       PORT_DS=PORT_DS & NPAD;
       delay(0x1F40,0x01);        //500us
       DDR_DS=DDR_DS & NPAD;   //Otpusti liniju
    PORT_DS=PORT_DS | PAD;        //Pull-up
       delay(0x640,0x01);         //100us
       if(!(PIN_DS & PAD))        // Precense pulse
       {
              delay(0x1F40,0x01); //500us
              return 1;
       }
       delay(0x1F40,0x01);     //500us
       return 0;
}

void write_0(void)
{
       DDR_DS=DDR_DS | PAD;    //Pin izlazni i nisko
       PORT_DS=PORT_DS & NPAD;
       delay(0x3C0,0x01);         //60us
       DDR_DS=DDR_DS & NPAD;   //Otpusti liniju
       PORT_DS=PORT_DS | PAD;  //Pull-up
       delay(0x05,0x01);          //Vrlo kratak delay
}

void write_1(void)
{
       DDR_DS=DDR_DS | PAD;       //Pin izlazni i nisko
       PORT_DS=PORT_DS & NPAD;
       delay(0x05,0x01);       //Vrlo kratak delay
       DDR_DS=DDR_DS & NPAD;   //Otpusti liniju
       PORT_DS=PORT_DS | PAD;  //Pull-up
       delay(0x3C0,0x01);         //60us
}

char read_bit(void)
{
       char bit=1;
       DDR_DS=DDR_DS | PAD;       //Pin izlazni i nisko
       PORT_DS=PORT_DS & NPAD;
       delay(0x05,0x01);            //Vrlo kratak delay
       DDR_DS=DDR_DS & NPAD;       //Otpusti liniju
       PORT_DS=PORT_DS | PAD;     //Pull-up
       delay(0x0A,0x01);            // Pricekaj da se pojavi podatak
       if(!(PIN_DS & PAD))
              bit=0;
       delay(0x3C0,0x01);     //60us, pricekaj kraj read time slota
       return bit;
```

```
}

//Salje 8-bitni podatak
void command(unsigned char data)
{
	char i;
	for(i=0; i<8; i++)
	{
		if(data & 1)
			write_1();
		else
			write_0();
		data=data>>1;
	}
}

//Cita 8-bitni podatak
char read_owi(void)
{
	unsigned char i, data=0;
	for(i=0; i<8; i++)
	{
		data=data>>1;
		if(read_bit())
			data=data | 0x80;
	}
	return data;
}
```

```
#include <avr/io.h>
#include "Opcenite.h"
#include "LCD.h"

//***************************************
//      Text LCD Driver
//***************************************
#define DDR    DDRB
#define PORT   PORTB
#define PIN    PINB

//Inicijalizira LCD
void LCDInit(void)
{
       DDR=0x3F;
       DDRD|=0x80;
       //20ms
       delay(1550, 0x04);
       PORT = 0x03;
       LCDtoggle();
       //5ms
       delay(512, 0x04);
       LCDtoggle();
       //160uS
       delay(5560, 0x01);
       LCDtoggle();
       LCDWait();
       PORT = 0x02;
       LCDtoggle();
       LCDWait();
       LCDSendData(0x28,0);
       LCDSendData(0x08,0);
       LCDSendData(0x01,0);
       LCDSendData(0x06,0);
       //Cursor ON/OFF blinking
       LCDSendData(0x0C,0);
}

//Enable pulse, paziti na Enable HighTime na vecim frekv.
void LCDtoggle(void)
{
       PORTD|=0x80;
       asm ("NOP");
       asm ("NOP");
       asm ("nop");
       asm ("nop");
       PORTD&=~0x80;
       asm ("NOP");
       asm ("NOP");
       asm ("nop");
       asm ("nop");
}

//Cita busy flag-->paziti na delay kod veceg takta
void LCDWait(void)
{
       char tmp;
       DDR= 0x30;
       DDRD|=0x80;
       PORT=0x20;
       do
       {
              asm ("nop");
```

```
                asm ("nop");
                PORTD|=0x80;
                asm ("nop");
                asm ("nop");
                asm ("nop");
                asm ("nop");
                tmp=(PIN & 0x0F) << 4;
                PORTD&=~0x80;
                PORTD|=0x80;
                asm ("nop");
                asm ("nop");
                asm ("nop");
                asm ("nop");
                tmp|=(PIN & 0x0F);
                PORTD&=~0x80;
                asm ("nop");
                asm ("nop");
                asm ("nop");
                asm ("nop");
        }while(tmp & 0x80);
        DDR=0x3F;
}

//Cisti ekran
void LCDClear(void)
{
        LCDWait();
        PORT=0x00;
        LCDtoggle();
        PORT=0x01;
        LCDtoggle();
}

//Slanje podataka
void LCDSendData(unsigned char data,char slovo)
{
        LCDWait();
        PORT = 0x00;
        if(slovo)
                PORT|=0x10;
        PORT = PORT | (data >> 4);
        LCDtoggle();
        PORT = PORT & 0xF0;
        PORT = PORT | (data & 0x0F);
        LCDtoggle();
}

//Salje tekst
void LCDSendText(char *data,char redak)
{
        switch(redak)
        {
                case 1: { LCDSendData(0x80,0); break; }
                case 2: { LCDSendData(0xC0,0); break; }
                default:{ LCDSendData(0x80,0); break; }
        }
        while(*data)
        {
                LCDSendData(*data,1);
                data++;
        }
}
```

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

//**************************************
//              OPCENITE
//**************************************

//Funkcija za kašnjenje
void delay(short int count, char presc)
{
       OCR1A=count;
       TCCR1B=presc;
       while(!(TIFR1 & 0x02));
              TIFR1=0x02;
       TCCR1B=0x00;
       TCNT1=0x00;
}
//Ispisuje na LCD jednu HEX znamenku
void hexprint(char broj)
{
       if(broj>9)                                          //A-F
              LCDSendData(('A' + (broj-10)),1);
       else                                                //0-9
              LCDSendData((broj+48),1);
}
```