

UNIVERZA V LJUBLJANI
FAKULTETA ZA ELEKTROTEHNIKO

Lojze Kunčič

Merjenje temperature s KTY 10

Seminarska naloga

pri predmetu
Elektronska vezja

V Ljubljani, april 2004

KAZALO:

1	Uvod	3
2	Glavni del	3
2.1	Elektronsko vezje (hardware)	3
2.1.1	Napajanje	3
2.1.2	Vhodni del	4
2.1.3	Mikrokontroler	7
2.1.4	Izhodni del	8
2.2	Programski del (firmware)	8
2.2.1	Izračun temperature	9
2.2.2	Izpis temperature	11
3	Zaključek	12
4	Priloge	13
4.1	Shema vezja	13
4.2	Izvorna koda v programskem jeziku C	13
5	Reference	17

KAZALO tabel:

Tabela 1	Temperaturni faktor k_t in upornost senzorja KTY 10	5
Tabela 2	Vrednost A/D pretvornika pri določeni temperaturi oz napetosti	10

KAZALO slik:

Slika 1	Odvisnost upornosti senzorja KTY, $R_{25} = 1970\Omega$	5
Slika 2	Potek merilne napetosti v odvisnosti od temperature	6
Slika 3	A/D pretvornik ADS7809	7
Slika 4	Diagram poteka programa	9
Slika 5	Vrednost A/D pretvornika glede na temperaturo	10

1 Uvod

Temperaturo tipa polprevodniškimi senzor KTY 10, ki pošilja analogni signal A/D pretvorniku, ki ga pretvori v digitalnega, katerega mikrokontroler obdela. Uporabljen je Atmelov mikrokontroler AVR AT90S8515. Vrednost temperature se izpisuje na treh 7-segmentnih LED prikazovalnikih, ki jih krmili mikrokontroler, ter na osebni računalniku, na Terminalskem oknu, preko serijske komunikacije RS-232. Območje delovanja senzorja je od -30°C do 130°C . Cilj ni narediti čimcenejši merilnik temperature, ampak se ob izdelavi le tega čimveč naučiti, tako da izdelava kakšnih podobnih aplikacij ne bi predstavljala večjih težav.

2 Glavni del

Vhodni del vezja je preprost. Iz senzorja KTY 10 in zaporedno vezanega linearnega upora peljemo analogni signal na 16-bitni A/D pretvornik, kjer ga v serijski, digitalni obliki peljemo na mikrokontroler. Na mikrokontrolerju digitalni signal ustrezno računsko obdelamo. Z mikrokontrolerjem multipleksno krmilimo izhodni del, tri 7-segmentne LED prikazovalnike, ter izpis temperature na PC-ju preko Terminalskega okna.

Napajanje:

- integrirano vezje 7805

Vhodni del:

- temperaturni senzor KTY 10
- 16-bitni A/D pretvornik ADS7809

Mikrokontroler:

- AVR AT90S8515

Izhodni del:

- 7-segmentni LED prikazovalnik
- serijska komunikacija RS-232

2.1 Elektronsko vezje (hardware)

2.1.1 Napajanje

Celotno vezje je napajano s +5V, za kar skrbi integrirano vezje 7805. Zaradi manjšega sesedanja napetosti ob obremenitvi izhoda 7805 sta vsebovana dva integrirana vezja. Z enim je napajen mikrokontroler in LED prikazovalnik, z drugim pa napajamo A/D pretvornik in skrbimo za referenčno napetost A/D pretvornika.

2.1.2 Vhodni del

- **temperaturni senzor KTY 10**

Za senzor je izbran pasivni polvodniški senzor KTY, ki se pogosto uporablja za manj zahtevna merjenja temperature do 150 °C.

S posebno metodo metalizacije N-prevodnega Si kristala dobimo časovno stabilen temperaturni senzor. Tipičen podatek senzorja je njegova upornost pri temperaturi 25 °C. Imenujemo jo karakteristična upornost R_{25} . Proizvajalec prebira senzorje glede na njihovo karakteristično upornost. Tako lahko kupimo senzorje s toleranco $R_{25} \pm (1\% - 4\%)$. Ta podatek je izredno pomemben za absolutno točnost meritve.

podatki o senzorju:

- senzor KTY 10-5
- upornost pri 25°C in toku $I_{op} = 1 \text{ mA}$
 $R_{25 \text{ min}} = 1950\Omega$
 $R_{25} = 1970\Omega$
 $R_{25 \text{ max}} = 1990\Omega$
- tip ohišja TO-92

Upornost senzorja KTY pri temperaturi T_A za območje od -30 °C do $+130 \text{ °C}$ podaja naslednja enačba:

$$R_T = R_{25} \cdot (1 + \alpha \cdot \Delta T_A + \beta \cdot \Delta T_A^2) = f(T_A)$$

$$\alpha = 7,88 \cdot 10^{-3} \text{ K}^{-1}; \beta = 1,937 \cdot 10^{-5} \text{ K}^{-2}$$

Izračuni po zgornji enačbi so podani v tabeli **Tabela 1**. V tabeli je podan še temperaturni faktor k_T :

$$k_T = \frac{R_T}{R_{25}} = 1 + \alpha \cdot \Delta T_A + \beta \cdot \Delta T_A^2 = f(T_A)$$

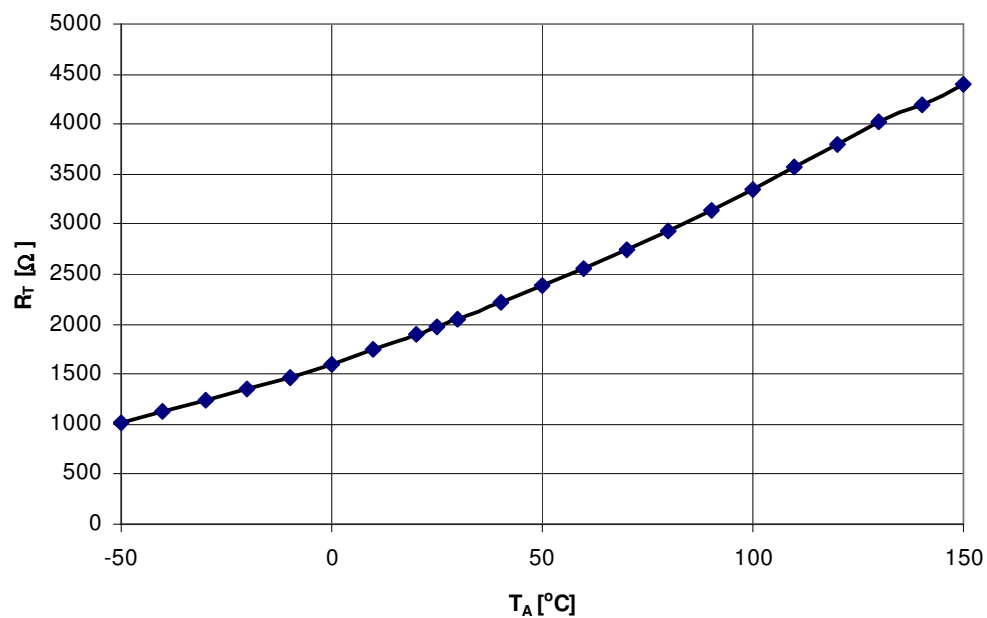
Temperatura senzorja se izračuna iz spremembe upornost senzorja po spodnji enačbi:

$$T = \left(25 + \frac{\sqrt{\alpha^2 - 4 \cdot \beta + 4 \cdot \beta \cdot k_T} - \alpha}{2 \cdot \beta} \right) \text{ °C}$$

$T [^{\circ}\text{C}]$	k_T	$R_T [\Omega]$
-50	0,518	1020,46
-40	0,570	1122,90
-30	0,625	1231,25
-20	0,685	1349,45
-10	0,748	1473,56
0	0,815	1605,55
10	0,886	1745,42
20	0,961	1893,17
25	1,000	1970,00
30	1,040	2048,80
40	1,123	2212,31
50	1,209	2381,73
60	1,300	2561,00
70	1,394	2746,18
80	1,492	2939,24
90	1,594	3140,18
100	1,700	3349,00
110	1,810	3565,70
120	1,923	3788,31
130	2,041	4020,77
140	2,128	4192,16
150	2,235	4402,95

Tabela 1 Temperaturni faktor k_T in upornost senzorja KTY 10

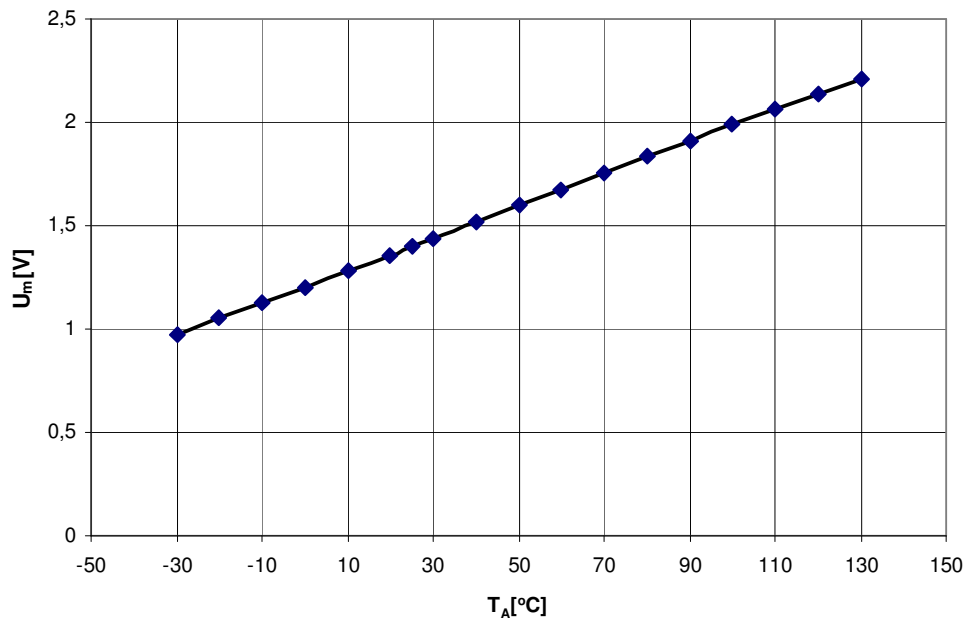
Odvisnost upornosti senzorja KTY, ki ima $R_{25} = 1970\Omega$, kaže **Slika 1**.

Slika 1 Odvisnost upornosti senzorja KTY, $R_{25} = 1970\Omega$

Iz diagrama se vidi, da upornost senzorja KTY ne narašča linearno s temperaturo. Zato jo je potrebno s posebnimi postopki linearizirati.

Temperaturno nelinearnost KTY senzorja lahko lineariziramo z dodatnim uporom, ki ga vežemo v serijo s senzorjem, če napajamo senzor s konstantno napetostjo, ali paralelno, če je senzor napajan s konstantnim tokom.

Uporabljena je metoda merjenja s konstantno napetostjo. Če senzorju KTY zaporedno vežemo upor (5000Ω) priključenega na napetost 5V dobimo naslednji potek merilne napetosti.



Slika 2 Potek merilne napetosti v odvisnosti od temperature

Poglavitni problem ni linearizacija, saj linearizacijo brez problemov opravi mikrokontroler. Z uporom predvsem omejimo tok skozi senzor KTY in s tem omejimo segrevanje le tega. Saj se karakteristična upornost precej poveča že pri merilnem toku 2mA, zato so priporočljivi merilni tokovi skozi senzor KTY največ do 1mA.

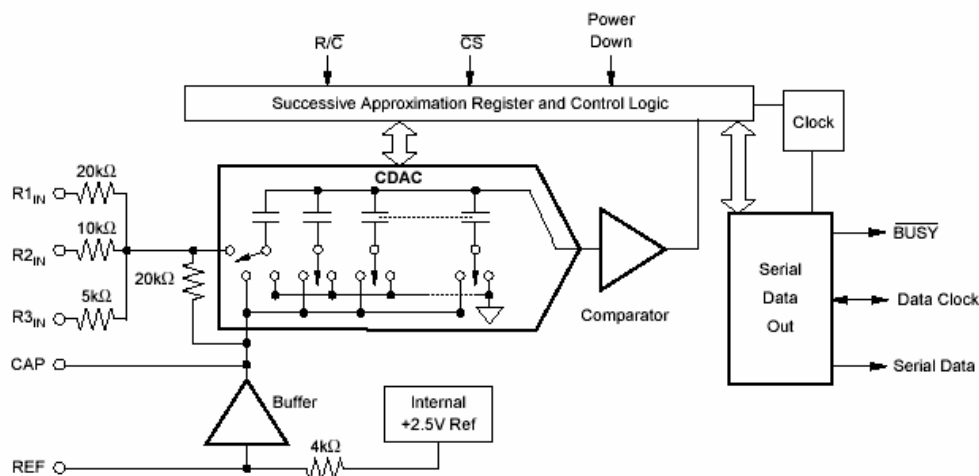
- **16-bitni A/D pretvornik ADS7809**

Izbran je bil Burr-Brownov A/D pretvornik ADS7809. Čeprav bi za potrebe samega vezja temperaturnega senzorja zadostoval cenejši, pa mogoče tudi preprostejši za realizacijo pretvorbe 8-bitni A/D pretvornik s serijskim ali paralelnim izhodom.

A/D pretvornik ADS7809:

- 100kHz frekvenca vzorčenja
- serijski izhod
- zunanja ali notranja referenčna napetost
- zunanja ali notranja sinhronizacijska ura
- analogni vhod $\pm 10V$ in od 0V do +5V

Uporabljena je notranja referenčna napetost ter zunanja sinhronizacijska ura. Analogni vhod je od 0V do 5V. Mikrokontroler dobi 16-bitni podatek po serijskem izhodu iz A/D pretvornika po končani pretvorbi A/D pretvornika.



Slika 3 A/D pretvornik ADS7809

2.1.3 Mikrokontroler

Izbran je bil Atmelov mikrokontroler AVR AT90S8515, saj je bilo zanj že narejeno tiskano vezje za programiranje preko serijske komunikacije, katerega je bilo treba še sestaviti. Možnost pisanja izvorne kode je v programskem jeziku C, uporaba simulatorja AVRStudio, možnosti emulatorja pa žal ni bilo.

AVR AT90S8515 je iz družine CMOS 8-bitnih mikrokontrolerjev na osnovi AVR RISC arhitekture.

Mikrokontroler AT90S8515-8:

- 8K programabilnega Flash pomnilnika
- 512 besed SRAM pomnilnika
- 512 besed EEPROM pomnilnika
- 32 programabilnih vhodno/izhodnih linij
- en 8 bitni števec/časovnik
- en 16 bitni števec/časovnik, Compare, Capture Modes in Dual 8-, 9- ali 10-bit PWM
- analogni primerjalnik
- programabilni Watchdog
- programabilni UART
- Master/Slave SPI serijska komunikacija
- operacijska napetost
4,0V – 6,0V
- hitrost
0 – 8 MHz

Uporabljenih je 11 izhodnih linij za krmiljenje LED prikazovalnikov, dve liniji za serijsko komunikacijo ter 5 linije za komunikacijo in prenos podatkov iz A/D pretvornika. Skupaj je tako zasedenih 18 vhodno/izhodnih linij.

Pri mikrokontrolerju koristimo obe prekinitvi. Prva prekinitvev, ki se izvrši ob prelivnem bitu 8 bitnega števca rabim za krmiljenje LED prikazovalnikov in izpis temperature na njem. Drugo prekinitvev, ki se izvrši ob prelivnem bitu 16 bitnega števca pa za odčitavanje temperaturo in branje podatka iz A/D pretvornika ter za izpis vrednosti temperature preko serijske komunikacije v Terminalsko okno na PC-ju.

2.1.4 Izhodni del

Iz 11-ih izhodnih linij mikrokontrolerja multipleksno krmilimo tri 7-segmentne LED prikazovalnike. LED prikazovalnike krmilimo tako, da najprej prižgemo LED1, na katerem je izpisana željena cifra, nakar LED1 ugasnemo in prižgemo LED2, na katerem izpišemo željeno cifro. Ugasnemo LED2 in prižgemo LED3, kjer izpišemo novo željeno cifro. Ugasnemo LED3. Ta postopek se ponavlja s tako hitrostjo, da imamo občutek, kot da so prižgani vsi trije prikazovalniki.

8 linij, po katerih teče podatek katero cifro bomo izpisali je vezanih na vse prikazovalnike. S tremi linijami pa prižigamo in ugašamo posamezni LED prikazovalnik.

1 linija je uporabljena za serijsko komunikacijo, ki gre iz mikrokontrolerja na integrirano vezje MAX232, s katerim spremenimo napetostne nivoje, kateri ustrezajo RS-232 komunikaciji, da se željena informacija izpisuje na Terminalskem oknu PC-ja.

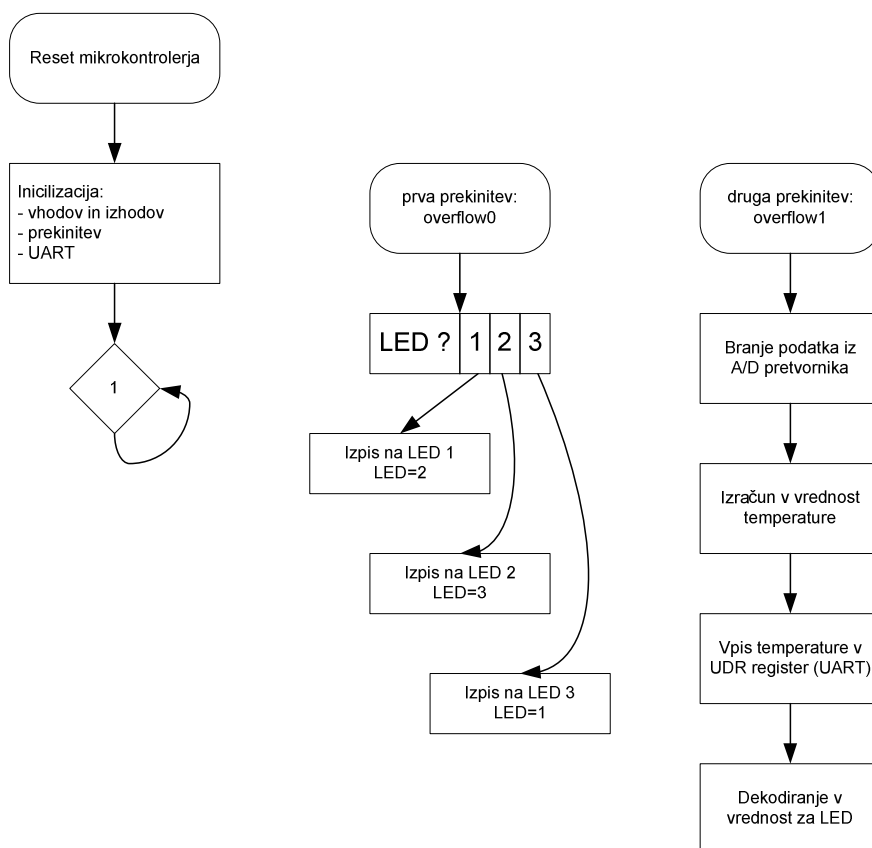
2.2 Programski del (firmware)

Program je pisan v programskem jeziku C. Program mora odčitati vrednost temperature, jo preračunati ter izpisati na LED prikazovalniku in na Terminalsko okno.

Program je preprost in se celotni program izvrši v dveh prekinitvah, ki se izvršita ko pride do preliva pri štetju števca.

S prvim, 8-bitnim števcem izpisujemo, vrednost temperature na LED prikazovalniku. Prekinitvev se izvaja s hitrostjo 488Hz.

Z drugim, 16-bitnim števcem preberemo analogni podatek (napetost) iz senzorja in ga z A/D pretvornikom pretvorimo. Nakar ta podatek (napetost) preračunamo in vrednost temperature izpišemo preko serijske komunikacije na Terminalsko okno. Postopek se ponovi na vsaki 2 sekundi.



Slika 4 Diagram poteka programa

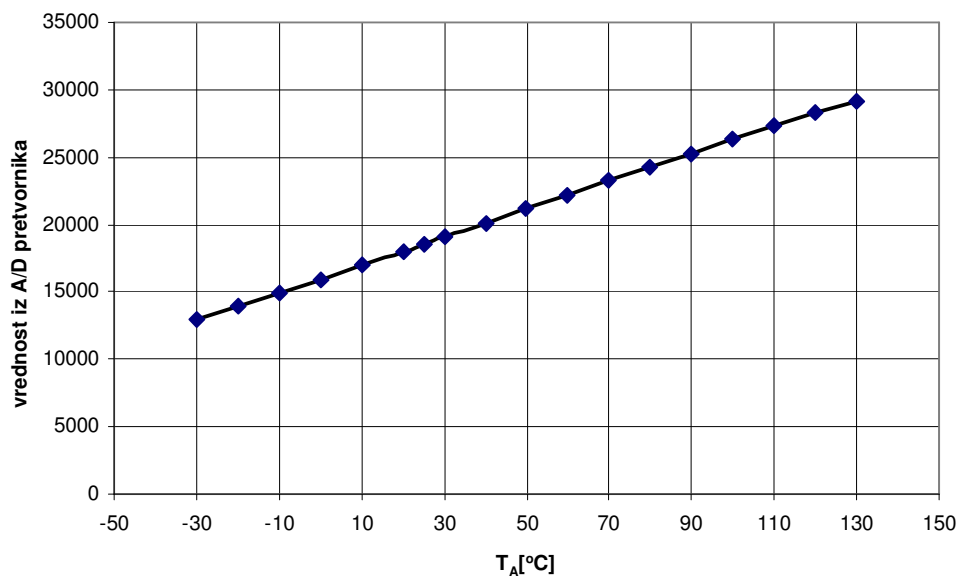
2.2.1 Izračun temperature

Temperaturo izračunamo iz podatka, ki ga dobimo iz A/D pretvornika. Podatek se osveži vsaki 2 sekundi.

Referenčna napetost A/D pretvornika je +5V, Least Significant Bit (LSB) je 76 μ V. Pri napetosti 0V dobimo vrednost A/D pretvornika 0000_{HEX}, pri 4.999924V pa FFFF_{HEX}. Na senzorju padec napetosti znaša od 0,987964V (pri -30°C) do 2,228618 V (pri 130°C), kar je ekvivalentno vrednosti na izhodu A/D pretvornika od 3295_{HEX} (pri -30°C) do 721B_{HEX} (pri 130°C). Podatke nam podaja spodnja tabela, na spodnjem grafu pa vidimo, da se temperatura spreminja linearno z napetostjo.

T [°C]	R _T [Ω]	U [V]	vrednost A/D pretvornika DEC	vrednost A/D pretvornika HEX
-30	1231,25	0,987964	12949	3295
-20	1349,45	1,062651	13928	3668
-10	1473,56	1,138137	14918	3A46
0	1605,55	1,215304	15929	3E39
10	1745,42	1,293782	16958	423E
20	1893,17	1,373222	17999	464F
25	1970,00	1,413199	18523	485B
30	2048,80	1,453297	19049	4A69
40	2212,31	1,533704	20103	4E87
50	2381,73	1,613260	21145	5299
60	2561,00	1,693559	22198	56B6
70	2746,18	1,772603	23234	5AC2
80	2939,24	1,851084	24263	5EC7
90	3140,18	1,928815	25281	62C1
100	3349,00	2,005629	26288	66B0
110	3565,70	2,081383	27281	6A91
120	3788,31	2,155312	28250	6E5A
130	4020,77	2,228618	29211	721B

Tabela 2 Vrednost A/D pretvornika pri določeni temperaturi oz napetosti



Slika 5 Vrednost A/D pretvornika glede na temperaturo

Ker pa napetost ni natanko +5V, serijski upor s senzorjem ni 5000Ω , in tudi samo senzor pri 25°C nima natanko 1970Ω , ampak nastopajo določene tolerance. Zaradi tega ni napetost na senzorju pri določeni temperaturi natanko takšna kot je prikazana v zgornji tabeli **Tabela 2**. Zaradi linearne odvisnosti med napetostjo oz. A/D pretvorbo in temperaturo je umerjanje preprosto. Pomerimo temperaturo v dveh točkah in izračunamo enačbo premice, po kateri se spreminja napetost v odvisnosti od temperature.

2.2.2 Izpis temperature

Temperatura se izpisuje na LED prikazovalnik in preko RS232 na PC.

Meritev je točna do 1°C natančno.

Izračun je pravilen na območju od -30°C do 130°C .

Če je temperatura npr. 21°C na LED prikazovalniku izpiše **21**, v Terminalskem oknu pa:

Temperatura je 21 stopinj Celzija.

Če pa je temperatura izven območja (npr. izračunana -80°C), nam na LED prikazovalniku izpiše **Err**, v Terminalskem oknu pa:

T = -80 oC (TEMPERATURA IZVEN OBMOCJA!!!)

- **prva prekinitev**

V tej prekinitev temperaturo na LED prikazovalnik izpisujemo multipleksno. Prekinitev se izvaja s frekvenco 488Hz. Ker imamo 3 prikazovalnike izpisujemo ob vsaki prekinitvi na en prikazovalnik, tako da je frekvenca osveževanja LED prikazovalnika 163Hz. Kar je dovolj hitro, da ne zaznamo utripanja. Najmanjša frekvenca, kjer ne zaznamo utripanja je približno 45Hz.

Temperatura se na LED prikazovalnik izpisuje multipleksno s frekvenco osveževanja 163Hz. Naše oko ne zazna tako visoke frekvence in imamo občutek, da so vsi LED prikazovalniki vklopljeni istočasno. Ta prekinitev bi se lahko izvajala vsaj dvakrat počasneje, pa še vedno nebi videli utripanja.

- **druga prekinitev**

Druga prekinitev se izvede na vsaki 2 sekundi. V tej prekinitvi najprej signal (napetost) iz senzorja pretvorimo preko A/D pretvornika v 16-bitno besedo. Zaradi velike občutljivosti A/D pretvornika (LSB je $76\mu\text{V}$) zadnji štirje biti skoraj niso uporabni. Nato digitalni signal preračunamo in iz napetosti dobimo temperaturo. Nakar vrednost temperature izpišemo preko RS-232 v Terminalsko okno, in vrednost temperature priredimo, da ustreza izpisu na LED prikazovalnik.

3 Zaključek

Cilj tega projekta ni bil izdelati čim cenejši merilnik temperature, ampak se pri projektiranju in izdelavi le tega naučiti čim več in uporabiti čim več svojega znanja in preprosti primer nekoliko razširiti. Projekt je mišljen bolj kot pedagoški, da bi se naučil samostojno začeti in uspešno zaključiti projekt in mogoče vanj vplesti več področij kot je nujno potrebno.

Že pri izbiri mikrokontrolerja bi se lahko odločil za takega, ki že vsebuje A/D pretvornik, vendar bi bilo treba na novo narediti programsko ploščico (zaradi drugačne porazdelitve pinov). Vendar s tem, ko sem vzel zunanji A/D pretvornik in vzel sem zahtevnejšega, čeprav ne bi bilo potrebno, sem se soočil še z njim. Prav tako je dodana serijska komunikacija.

Ker je šlo za prvi samostojni projekt pri uporabi mikrokontrolerjev so bile temu primerne težave in problemi, ki so se pa z nekaj poglobljanja v mikrokontroler in samo vezje uspešno rešili. Največi problem oziroma problem, ki mi je vzel največ časa je bilo nihanje izhodne napetosti na integriranem vezju 7805. Ob vsakem vklopu LED prikazovalnika se je poraba povečala in je napetost na 7805 rahlo padla, kar je zmedlo A/D pretvornik, ki je imel referenčno napetost nastavljeno glede na izhod 7805. Sprva sem mislil, da je problem v izvorni kodi, vendar, ko sem šel skozi celotni program, napisal sem ga tudi na drugačen način, sem prišel do zgoraj omenjenega zaključa. Ko sem ločil napajanje A/D pretvornika od preostalega vezja je vezje delovalo.

Vezje je bilo potrebno še preizkusiti. Za to sem uporabil navaden alkoholni termometer (uporablja se za merjenje zunanje temperature bivalnih prostorov). Z njim sem meril temperaturo v zamrzovalni skrinji LTH, zunanjo temperaturo in temperaturo v ogrevani sobi pri temperaturah -18°C , -14°C , 9°C , 14°C , 19°C in 29°C . Temperatura mojega vezja se je ujemala s temperaturo termometra. K večji točnosti meritve bi pripomogla boljše merilna oprema (temperaturna komora).

Menim da je projekt uspešno narejen, saj sem z njim dosegel željeni cilj. Cilj ni bil samo narediti merilnik temperature ampak se ob tem tudi naučiti tako serijsko komunikacijo, uporabo A/D pretvornika,...

Ta projekt bi se dalo še nadgraditi. Ker je omogočena serijska komunikacija je mogoče vrednost temperature na PC-ju obdelovati na poljuben način (izrisavanje grafov, ...).

4 Priloge

4.1 Shema vezja

Glej zadnjo stran dokumenta!

4.2 Izvorna koda v programskem jeziku C

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <stdio.h>

/*
#define d0 outp(0x3F, PORTA);           // 0
#define d1 outp(0x06, PORTA);           // 1
#define d2 outp(0x5B, PORTA);           // 2
#define d3 outp(0x4F, PORTA);           // 3
#define d4 outp(0x66, PORTA);           // 4
#define d5 outp(0x6D, PORTA);           // 5
#define d6 outp(0x7D, PORTA);           // 6
#define d7 outp(0x07, PORTA);           // 7
#define d8 outp(0x7F, PORTA);           // 8
#define d9 outp(0x6F, PORTA);           // 9
#define d outp(0x40, PORTA);            // -
#define c outp(0x80, PORTA);            // .
#define dn outp(0x00, PORTA);           //
#define D0 outp(0xBF, PORTA);           // 0.
#define D1 outp(0x86, PORTA);           // 1.
#define D2 outp(0xDB, PORTA);           // 2.
#define D3 outp(0xCF, PORTA);           // 3.
#define D4 outp(0xE6, PORTA);           // 4.
#define D5 outp(0xED, PORTA);           // 5.
#define D6 outp(0xFD, PORTA);           // 6.
#define D7 outp(0x87, PORTA);           // 7.
#define D8 outp(0xFF, PORTA);           // 8.
#define D9 outp(0xEF, PORTA);           // 9.
#define E outp(0x79, PORTA);            // E
#define r outp(0x50, PORTA);            // r

#define dis1 outp(0xF3, PORTB);          // display 1
#define dis2 outp(0xF5, PORTB);          // display 2
#define dis3 outp(0xF6, PORTB);          // display 3
#define dis outp(0xFF, PORTB);          // disable all display
*/

void segment(int *zaslon);
void convert(int *vrednost);

char led = 1, prikaz[4];
char temper[50];
int pos = 0;
int display, result;

// UART transmit
SIGNAL(SIG_UART_TRANS)
{
    pos++;
    if (temper[pos] != 0) {
        outp(temper[pos], UDR);
    }
    else
        pos = 0;
}

//Interrupt for LED display
SIGNAL(SIG_OVERFLOW0)                                // signal handler for tcnt0 overflow interrupt
{
    segment(&display);
    switch (led) {
        case 1:
            led = 2;
            outp((1 << PC1) | (1 << PC2), PORTC);      // display 1
            outp(display, PORTA);
            break;
        case 2:
            led = 3;
    }
}
```

```

        outp((1<<PC0)|(1<<PC2), PORTC);           // display 2
        outp(display, PORTA);
        break;
    case 3:
        led = 1;
        outp((1<<PC0)|(1<<PC1), PORTC);           // display 3
        outp(display, PORTA);
        break;
    }
    outp(0x00, TCNT0);                             // reset counter to get this interrupt again (TCNT0)
}

//Interrupt for reading value of the temper sensor
SIGNAL(SIG_OVERFLOW1)                             // signal handler for tcnt1 overflow interrupt
{
    int k;

    convert(&result);
    result=(result-14775)/87.773;

    if ((result < -30)||(result > 130)) {
        sprintf(temper, 50, "T = %d oC (TEMPERATURA IZVEN OBMOCJA!!!)\n", result);
    }

    else {
        sprintf(temper, 50, "Temperatura je %d stopinj Celzija.\n", result);
    }

    pos = 0;
    outp(temper[pos], UDR);                         // write the first byte to vrednost buffer

    if ((result < -30)||(result > 130)) {
        prikaz[0]=0x79;
        prikaz[1]=0x50;
        prikaz[2]=0x50;
    }

    else {
        sprintf(prikaz, "%d", result);

        if ((prikaz[2] == 0x00)&(prikaz[1] != 0x00)) {
            prikaz[2]=prikaz[1];
            prikaz[1]=prikaz[0];
            prikaz[0]=0x00;
        }

        if (prikaz[1] == 0x00) {
            prikaz[2]=prikaz[0];
            prikaz[1]=0x00;
            prikaz[0]=0x00;
        }

        for (k=0; k<3; k++)
        {
            if (prikaz[k]==0x2D)                    // -
                prikaz[k]=0x40;
            else if (prikaz[k]==0x30)                // 0
                prikaz[k]=0x3F;
            else if (prikaz[k]==0x31)                // 1
                prikaz[k]=0x06;
            else if (prikaz[k]==0x32)                // 2
                prikaz[k]=0x5B;
            else if (prikaz[k]==0x33)                // 3
                prikaz[k]=0x4F;
            else if (prikaz[k]==0x34)                // 4
                prikaz[k]=0x66;
            else if (prikaz[k]==0x35)                // 5
                prikaz[k]=0x6D;
            else if (prikaz[k]==0x36)                // 6
                prikaz[k]=0x7D;
            else if (prikaz[k]==0x37)                // 7
                prikaz[k]=0x07;
            else if (prikaz[k]==0x38)                // 8
                prikaz[k]=0x7F;
            else if (prikaz[k]==0x39)                // 9
                prikaz[k]=0x6F;
            else if (prikaz[k]==0x00)                //
                prikaz[k]=0x00;
        }
    }

    outp(0x0B, TCNT1H);                             // reset counter to get this interrupt again (TCNT1)
    outp(0xDB, TCNT1L);
}

```

```
//Display
void segment(int *zaslon)
{
    switch (led) {
        case 1:
            *zaslon = prikaz[0];
            break;
        case 2:
            *zaslon = prikaz[1];
            break;
        case 3:
            *zaslon = prikaz[2];
            break;
    }
}

//Read data from AD converter
void convert(int *vrednost)
{
    char d[16];
    int podatek, i, f;

    cbi(PORTC, PINC4); // CSn 0
    cbi(PORTC, PINC5); // R/Cn 0
    sbi(PORTC, PINC4); // CSn 1
    sbi(PORTC, PINC5); // R/Cn 1

    loop_until_bit_is_set(PINC, 3);

    cbi(PORTC, PINC4); // CSn 0

//SYNC
    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);

//read data from ADC
    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[15]=1;
    else d[15]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[14]=1;
    else d[14]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[13]=1;
    else d[13]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[12]=1;
    else d[12]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[11]=1;
    else d[11]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[10]=1;
    else d[10]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[9]=1;
    else d[9]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[8]=1;
    else d[8]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[7]=1;
    else d[7]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[6]=1;
    else d[6]=0;

    sbi(PORTC, PINC6);
```

```

    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[5]=1;
    else d[5]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[4]=1;
    else d[4]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[3]=1;
    else d[3]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[2]=1;
    else d[2]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[1]=1;
    else d[1]=0;

    sbi(PORTC, PINC6);
    cbi(PORTC, PINC6);
    if(bit_is_set(PINC, 7)) d[0]=1;
    else d[0]=0;

    sbi(PORTC, PINC4); // CSn 1

    f=1;
    podatek=0;
    for(i=0; i<16; i++) // Put all received bits in vrednost
    {
        podatek+=d[i]*f;
        f*=2;
    }
    *vrednost=podatek; // 16 bits off vrednost from AD converter
}

//main program
int main(void)
{
    outp(0xFF, DDRA); // use all pins on PortA for output (LED)
    outp(0x77, DDRC); /* use pins 0 - 2 on PortC for output (LED)
                       and others pins for output (CSn, R/Cn, CLK) and input (BUSYn, vrednost) */

    outp((1<<TOIE0)|(1<<TOIE1), TIMSK); // enable TCNT0 overflow and TCNT1 overflow

    outp(0x00, TCNT0); // reset TCNT0

    outp(0xFF, TCNT1H); // reset TCNT1
    outp(0x00, TCNT1L); // reset TCNT1

    outp(0, TCCR1A);

    outp(3, TCCR0); // count with cpu clock/1024
    outp(4, TCCR1B); // count with cpu clock/1024

    sbi(PORTC, PINC5); // R/Cn 1
    sbi(PORTC, PINC4); // CSn 1

    outp((1<<TXCIE)|(1<<TXEN), UCR); // enable TX interrupt
    outp(51, UBRR); // transfer speed

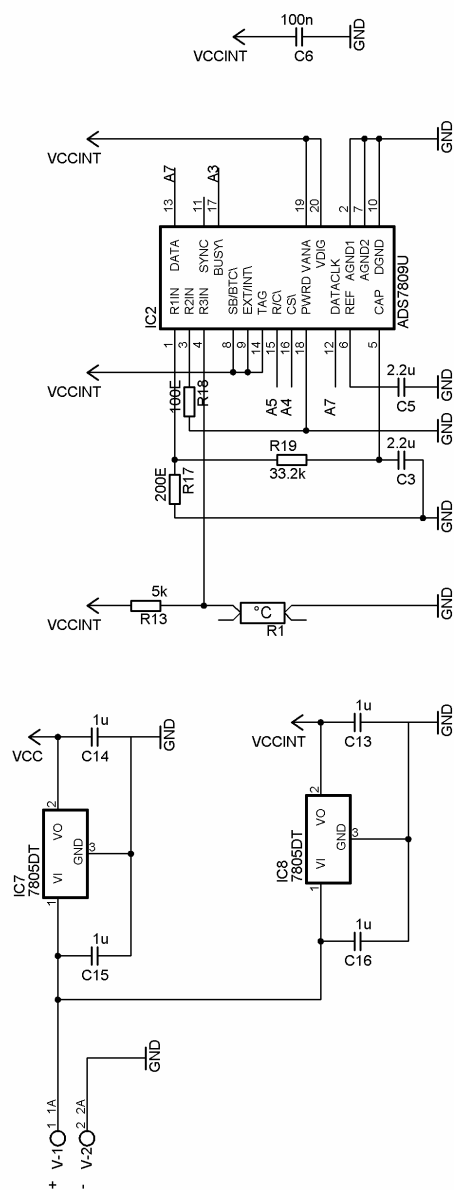
    sei(); // enable interrupts

    while(1) {
    }
}

```

5 Reference

1. Svet Elektronike, številka 13, CD1, letnika 1994 in 1995
2. Tadej Tuma; Mikrokrmilniški sistemi: programiranje za družino HC11, 2000
3. Mikrokontroler AVR AT90S8515
http://www.atmel.com/dyn/products/product_card.asp?part_id=2002
4. Senzor temperature KTY 10-5
http://www.infineon.com/cmc_upload/0/000/012/047/kt_10_.pdf
5. 16-bitni A/D pretvornik ADS7809
<http://focus.ti.com/docs/prod/folders/print/ads7809.html>
6. 7 segmentni LED prikazovalnik
<http://www.kingbright-led.com/data/spec/SA52-11GWA.pdf>



TITLE: Merjenje temperature s KTY 10

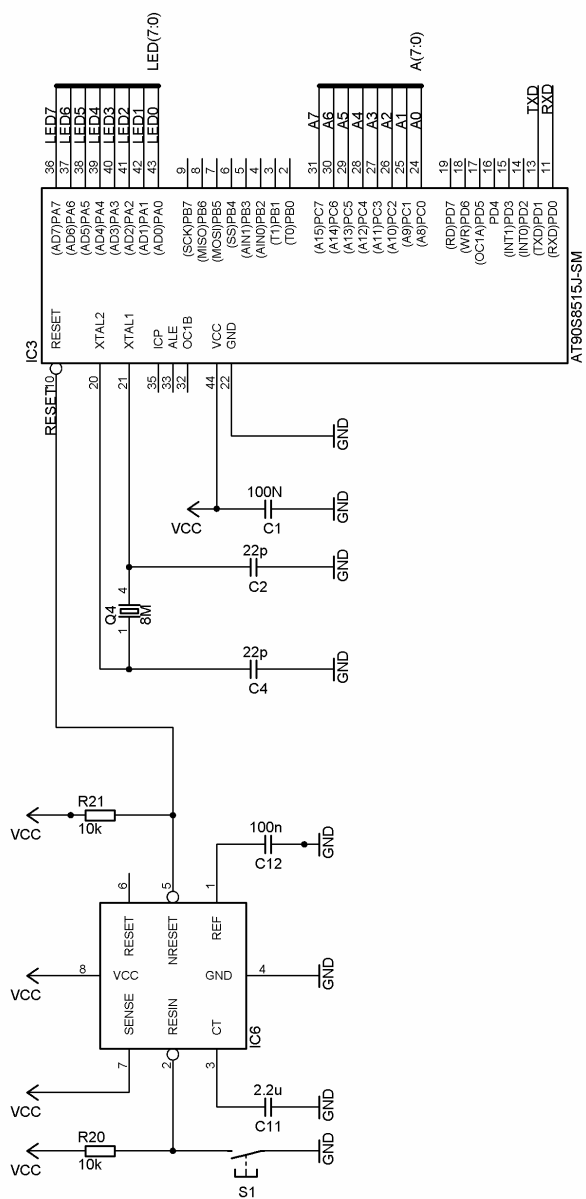
Document Number:

REV:

Date: 15.04.2004 22:03:24

Sheet: 1/1

Napajanje in vhodni del



TITLE: Merjenje temperature s KTY 10

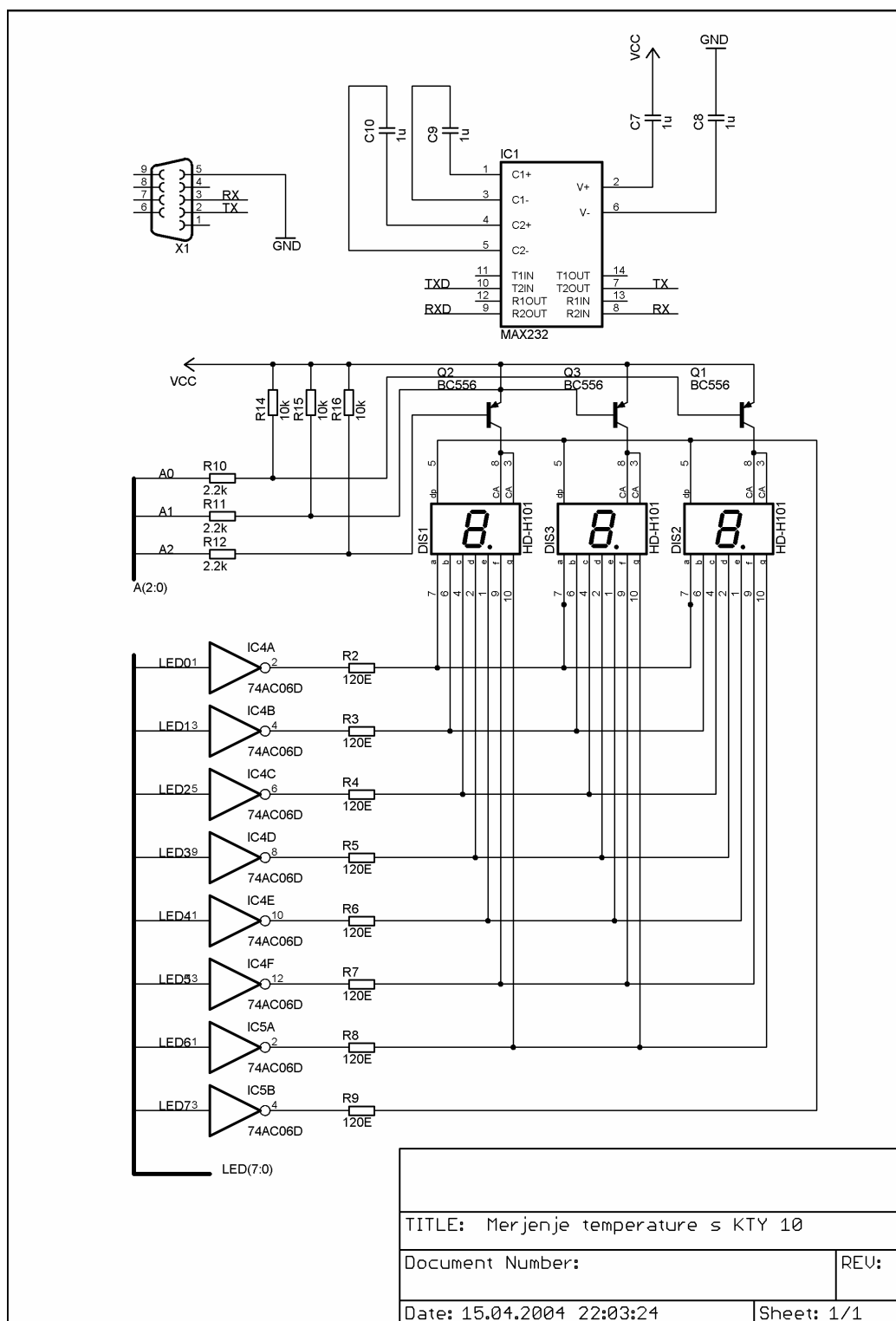
Document Number:

REV:	
------	--

Date: 15.04.2004 22:03:24

Sheet: 1/1

Mikrokontroler



Izhodni del