UNIVERZITET U ISTOČNOM SARAJEVU
ELEKTROTEHNIČKI FAKULTET

MIKROPROCESORSKI SISTEMI

SEMINARSKI

LED MATRIX

Mentor:
Prof. dr Slobodan Lubura

Student:
Marko Pravdić 813

Istočno Sarajevo, maj 2014. god.

SADRŽAJ

1. UVOD

U ovom seminarskom radu objasniće se osnovni pojmovi o komponentama koje su korištene za izradu projektnog zadatka. Dati su opisi pinova komponenti, njihove karakteristike, princip rada, uslovi potrebni za rad, itd.

U nastavku rada biti će objašnjeni mikrokontroleri, detaljnije mikrokontroler PIC 16F628A koji je korišten u ovom projektnom zadatku, USART komunikacija, SPI komunikacija, I watchdog timer.

U trećem poglavlju biti će riječ LED Matrix displayima, a u četvrtom o integrisanom kolu MAX7219 koje se koristi kao LED driver i služi da se poruka prenese od mikrokontrolera ka LED matrici. U petom poglavlju biti će riječ o integrisanom kolu MAX232 koje se koristi kao interfejs između mikrokontrolera i računara

Šesto poglavlje je opis hardverske i softverske realizacije ovog rada pomoću "Proteus Proffesional" programskog paketa i "MicroC PRO for PIC", i biće dat prikaz električne šeme i izgled štampane pločice, program mikrokontrolera sa opisom naredbi koje se koriste, kao i konačan izgled projekta.

2. MIKROKONTROLERI

Mikrokontroler je digitalna elektronska naprava u obliku integrisanog kola. Namjena mikrokontrolera je upravljanje uređajima i procesima, pa u sebi ima integrisan mikroprocesor, memoriju, digitalne i analogne ulaze i izlaze, digitalne satove (timere), brojače (countere), oscilatore, komunikacione sklopove i druge dodatke za koje je nekada bio potreban niz posebnih integralnih kola. Mikrokontroler normalno radi u kontrolnoj petlji, dakle očitava ulaze i zatim podešava izlaze u skladu sa svojim programom. Petlja se stalno ponavlja dok traje kontrola procesa.

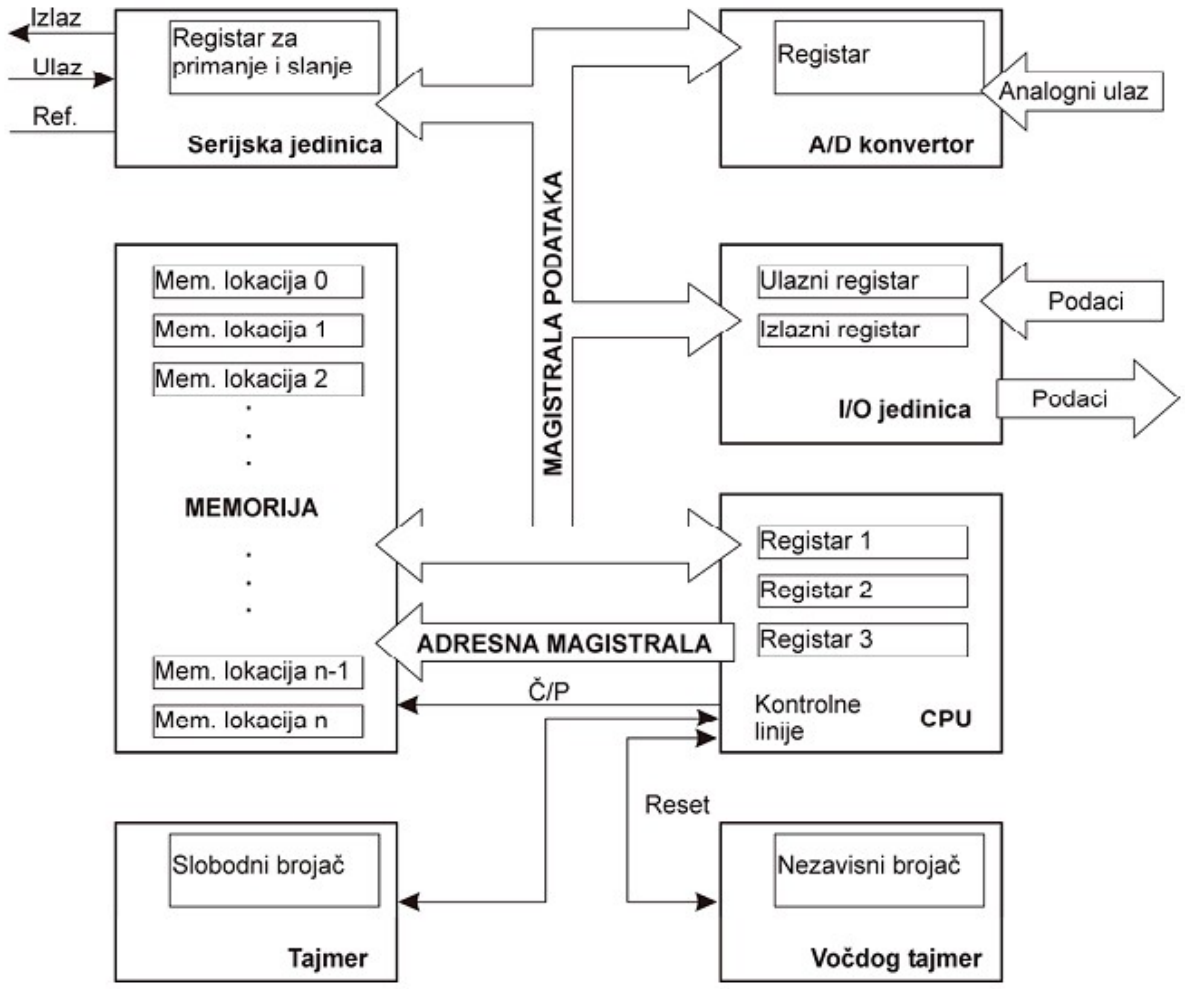
Glavna razlika između modernih mikroprocesora i mikrokontrolera je da su prvi optimizirani za brzinu i performanse kod računarskih programa, dok su mikrokontroleri optimizirani u pravcu integracije većeg broja kola, upravljanja procesima u stvarnom vremenu (real-time control), masovnu proizvodnju, nisku cijenu, i malu potrošnju struje. Mikrokontroleri su otporniji i na varijacije napona, temperature, vlažnosti, vibracije i tako dalje.

Razne vrste mikrokontrolera će imati razne module integrisane u mikrokontroler. Ipak većina uključuje kao minimum slijedeće dijelove:

- Mikroprocesor
- ROM (EPROM ili flash memoriju)
- RAM memoriju
- Oscilator
- Sat (timer)
- Brojač (counter)
- Watchdog timer
- Digitalne ulaze i izlaze

Većina uz te posjeduje i:

- Analogne ulaze i izlaze
- Komunikacioni sklop (interface), USART, SPI, I2C i druge
- Analogne poređivače napona (naponske komparatore)
- Modulator širine pulsa (PWM modulator) za kontrolu motora

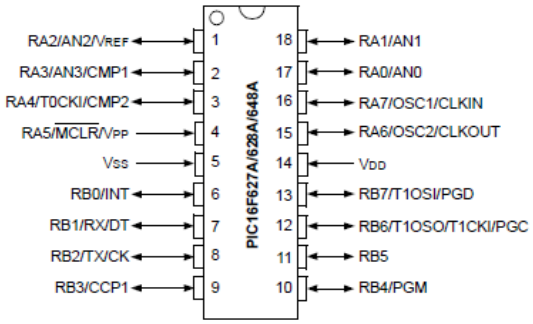


Slika Šema mikrokontrolera sa njegovim osnovnim elementima

2.1 Mikrokontroler 16F628A

Mikrokontroler PIC16F628A firme Microchip predstavlja integraciju mikroprocesora (CPU), memorije i periferija, pa zato ne zahtjeva složeni eksterni hardver da bi se realizovao mikroračunarski sistem. Ovaj mikrokontroler je izrađen u CMOS tehnologiji sa ugrađenom FLASH i EEPROM memorijom za memorisanje programa i podataka. PIC16F628A ima tipičnu RISC (Reduce Instruction Set Computer), Ova arhitektura kao što je poznato, karakteriše se manjom skupinom instrukcijama koje se brže izvršavaju od instrukcija kod CISC arhitekture. RISC arhitektura ima svega 35 intstrukcija. Mikrokontroler PIC16F628A koristi Harvard arhitekturu, kod koje su magistrale za prenos instrukcija i prenos podataka odvojene, omogućava instrukcije dužine 14 bitova. Sve instrukcije se izavršavaju u jednom ciklusu, osim instrukcija grananja i uslovnog skoka.

Na slici 2 prikazan je raspored pinova na mikrokontroleru, a u tabeli 1 vidimo detaljan opis pinova.



Slika 2 Raspored pinova na PIC 16F628a

Tabela 1 Opis pinova

Broj pina	Uloga	OPIS
1	RA2	Digitalni ulaz/izlaz porta A
	AN2	Kanal 2 A/D konvertora
	V_{REF}	Referentni napon A/D konvertora
2	RA3	Digitalni ulaz/izlaz porta A
	AN3	Kanal 3 A/D konvertora
	CMP1	Izlaz komparatora 1
3	RA4	Digitalni ulaz/izlaz porta A
	T0CKI	Spoljašnji izvor klok signala za TMR0 brojač
	CMP2	Izlaz komparatora 2
4	RA5	Digitalni ulaz/izlaz porta A
	MCLR	Pin za reset. Nizak logički nivo na ovom pinu resetuje mikrokontroler
	V_{PP}	Napon za programiranje
5	V_{SS}	Niski nivo napajanja za logiku i ulazno/izlazne pinove
6	RB0	Digitalni ulaz/izlaz porta B
	INT	Spoljni izvor interapta
7	RB1	Digitalni ulaz/izlaz porta B
	RX	Asinhroni ulaz USART modula
	DT	Podaci sinhronog USART modula
8	RB2	Digitalni ulaz/izlaz porta B
	TX	Asinhroni izlaz USART modula
	CK	Taktni ulaz/izlaz asinhronog USART modula
9	RB3	Digitalni ulaz/izlaz porta B
	CCP1	Ulaz/izlaz modula CCP1 i PWM1
10	RB4	Digitalni ulaz/izlaz porta B
	PGM	Dozvola za programiranje ugrađenog čipa
11	RB5	Digitalni ulaz/izlaz porta B
12	RB6	Digitalni ulaz/izlaz porta B
	T1OSO	Izlazni pin oscilatora unutar tajmera 1
	T1CKI	Spoljni taktni ulaz tajmera 1
	PGC	Klok kod ICSP™ programiranja
13	RB7	Digitalni ulaz/izlaz porta B

	T1OSI	Ulazni pin oscilatora unutar tajmera 1
	PGD	Ulaz/izlaz podataka za ICSP™
14	V_{DD}	Pozitivno napajanje za logiku i ulazno/izlazne pinove
15	RA6	Digitalni ulaz/izlaz porta A
	OSC2	Izlaz kristalnog oscilatora
	CLKOUT	Izlaz na kome se pojavljuje signal $F_{OSC}/4$
16	RA7	Digitalni ulaz/izlaz porta A
	OSC1	Ulaz kristalnog oscilatora
	CLKIN	Ulaz za spoljni izvor taktnog signala
17	RA0	Digitalni ulaz/izlaz porta A
	AN0	Kanal 0 A/D konvertora
18	RA1	Digitalni ulaz/izlaz porta A
	AN1	Kanal 1 A/D konvertora

2.2 Osnovne karakteristike mikrokontrolera PIC16F628A

Visoko performansni RISC CPU:

- Radna frekvencija 0-20 MHz
- Postojanje interapta
- 8 nivoa dubokog magacina
- Direktno, indirektno i relativno adresiranje

Posebne osobine mikrokontrolera:

- Interne i eksterne opcije oscilatora:
  - Precizni interni oscilator (4 MHz)
  - Interni oscilator male potrošnje (48 KHz)
  - Eksterni oscilator koji podržava kristalne oscilatore
- Sleep režim rada za uštedu potrošnje
- Programabilni pull-up na PORTB
- Multipleksirani Master Clear ulazni pin
- Watchdog tajmer sa nezavisnim oscilatorom za pouzdano funkcionisanje
- Niskonaponsko programiranje
- In-Circuit Serial Programing (preko 2 pina)
- Programabilna zaštita koda
- Brown-out reset
- Power-on reset
- Power-up tajmer i Start-up tajmer oscilatora
- Široki naponski opseg rada (2.0-2.5)
- Fleš I EEPROM memorijske ćelije velike izdržljivosti:
  - Mogućnost 100 000 upisa u fleš memoriju
  - Mogućnost 1 000 000 upisa u EEPROM memoriju
  - Postojanost podataka 40 godina

Osobine niske potrošnje:

- Vrijednost struje u Standby režimu

- 100 nA (2.0V)

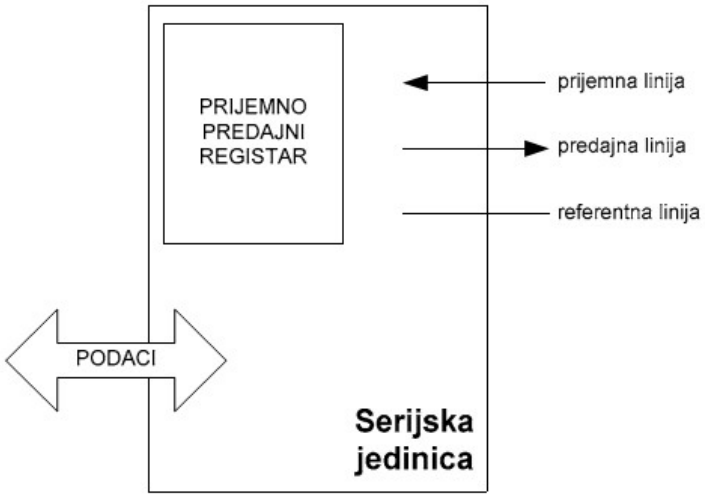
- Vrijednost radne struje:
  - 12 μA (32kHz, 2.0 V)
  - 120 μA (1 MHz, 2.0 V)
- Struja Watchdog tajmera:
  - 1 μA (2.0 V)
- Struja Timer1 oscilatora:
  - 1.2 μA (32 kHz, 2.0 V)
- Dvostruka brzina internog oscilatora:
  - Mogućnost odabira brzine frekvencije 4 MHz i 48 kHz
  - 4 μs potrebno za povratak iz Sleep režima

Osobine periferija:

- 16 I/O pinova sa mogućnošću pojedinačnog određivanja da li će pin biti ulazni ili izlazni
- Analogni komparatorski modul sa:
  - Dva analogna komparatora
  - Programabilnom naponskom referencom na čipu
  - Mogućnošću odabiranja interne ili eksterne reference
  - Izlazi komparatora su i eksterno dostupni
- Timer0: 8-bitni tajmer/brojač sa 8-bitnim preskalerom
- Timer1: 16-bitni tajmer/brojač sa mogućnošću eksternog oscilatora
- Timer2: 8-bitni tajmer/brojač sa 8-bitnim registrom, preskalerom i postskalerom
- Capture, Compare, PWM modul:
  - 16-bitni Capture/Compare
  - 10-binti PWM
  - Aderesabilni USART/SCI

2.3 USART komunikacija

USART komunikacija omogućava povezivanje mikrokontrolera na daljinu. Paralelni prenos nije pogodan za velike udaljenosti zbog velikog broja linija koje su potrebne za prenos podataka. Da bi prenos funkcionisao treba odrediti skup pravila po kojima će se odvijati. Skup pravila po kojima se vrši prenos podataka naziva se protokol. Serijski prenos podataka dijelimo na sinhroni i asinhroni. Sinhroni prenos između dva uređaja radi pod zajedničkim taktom. Asinhroni prenos se koristi kada ne znamo frekvencijski takt drugog uređaja. Ova vrsta prenosa koristi se za slanje manjih podataka.



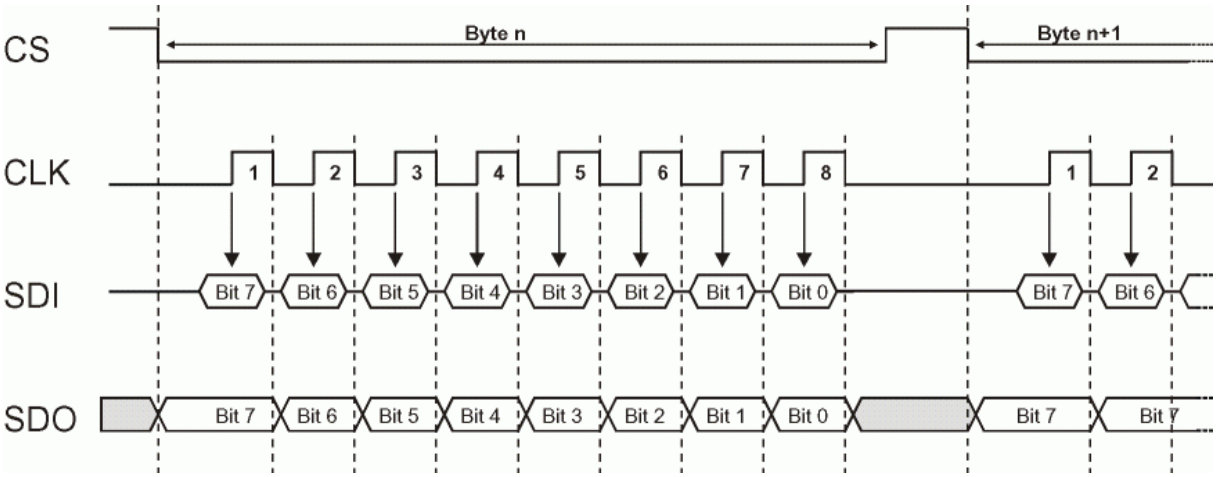
U ovom projektnom zadatku USART komunikacija se koristi za komunikaciju mikrokontrolera sa računarom. Da bi komunikacija ispravno radila, potrebno je koristiti integrisano kolo MAX232 o kojem će biti riječi u nastavku.

2.4 SPI komunikacija

SPI komunikacija omogućava razmjenu (slanje i primanje) simultano, koristeći 3 ulazno – izlazne linije:

- SDO – Serial Data Out – linija za slanje
- SDI – Serial Data In – linija za primanje
- SCK – Serial Clock – linija za sinhronizaciju

Također se može koristiti i četvrta linija SS (Slave Select) ukoliko postoji potreba da mikrokontroler komunicira sa više periferijskih uređaja. Pomoću selekcionog bita CS (Chip Select) biramo s kojim uređajem komuniciramo.



Slika 6 SPI komunikacija

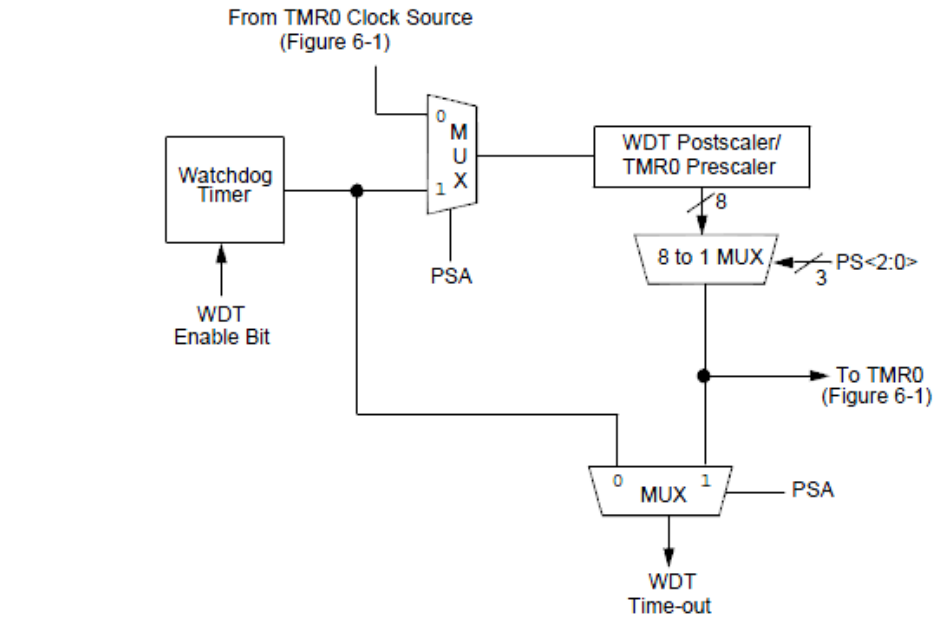
S obzirom da mikrokontroler PIC16F628A hardverski ne podržava SPI komunikaciju, u ovom projektu je korištena softverska SPI komunikacija, koja radi na istom principu kao i hardverska. Implementacija softverske SPI komunikacije se vrši pomoću integrisane biblioteke u MikroC-u.

2.5 Watchdog timer

Jako je bitno obezbijediti da mikrokontroler besprijekorno funkcioniše tokom cijelog svog rada. To se postiže watchdog timerom koji u stvari predstavlja slobodni brojač gdje program treba da upiše nulu svaki put kada se određena operacija izvrši korektno. U slučaju da dođe do problema u izvršavanju neke operacije, nula neće biti upisana u brojač, i brojač će se resetovati kada dostigne maksimalnu vrijednost brojanja. To će dovesti do ponovnog izvršavanja programa, ali ovaj put korektno. Veoma je važno da se svaki dio programa izvršava pouzdano bez čovjekovog nadgledanja.

Watchdog timer je povezan na potpuno odvojeni RC oscillator od mikrokontrolera. Ukoliko je watchdog timer uključen svaki put kada izbroji do kraja, desi se reset mikrokontrolera, i izvršenje programa počinje ispočetka. Zbog toga je programski kod potrebno podijeliti u segmente, i nakon izvršenja datog segmenta resetovati watchdog timer.

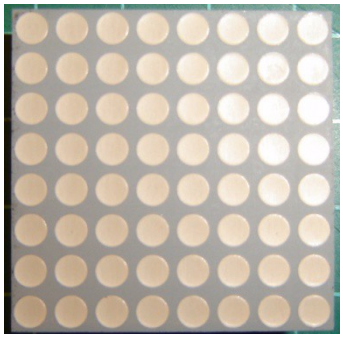
Da bi watchdog timer izbrojao do kraja potrebno je 17ms, nakon čega se mikrokontroler resetuje, međutim, podešavanjem bitova podscalera watchdog timera, možemo podešavati vrijeme brojanja timera. Maksimalno vrijeme iznosi 2.3s.



Slika 7 Watchdog timer blok dijagram

3. LED MATRIX DISPLAY

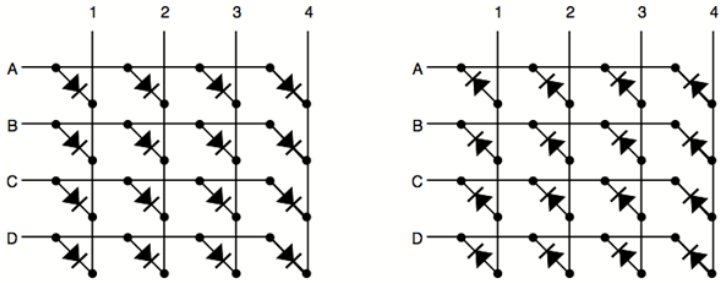
LED matrix display pruža fleksibilnost pri prikazu teksta, grafike i animacija, pa su postali popularno sredstvo za prikazivanje informacija danas. Možemo ih sresti na raznim javnim mjestima gdje prikazuju razne informacije, npr. na benzinskim pumpa gdje prikazuju cjene



Slika 8 Izgled LED matrix displaya

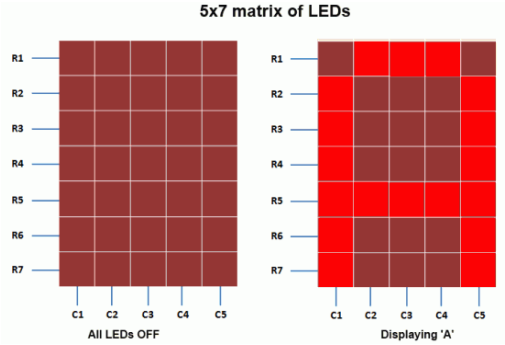
LED matrix je ustvari dvodimenzionalna struktura koju čine LED diode međusobno povezane u redove i kolone, čime je minimiziran broj ulaza displaya. Kontrolisanjem protoka struje kroz svaku od vrsta i kolona moguće je upravljati pojedinačnom LED diodom. Brzim skeniranjem redova i kolona postiže se brzo blinkanje dioda, i na taj način je moguće prikazivanje karaktera ili slika, čime se prenosi informacija korisniku. Brzina skeniranja je dovoljno brza da ljudsko oko ne primjeti treperenja

Na slici 8 prikazana je strukuta 4x4 LED matrice, i to sa zajedničkom anodom po vrsti (lijevo) i zajedničkom katodom po vrsti (desno).



Slika 8 Struktura 4x4 LED matrice

Razlika između LED matrice sa zajedničkom katodom i zajedničkom anodom je u načinu paljenja (polarizacije) LED dioda. Na slici 8 vidimo da su kolone označene sa 1...4, a vrste sa A...D. Ukoliko želimo upaliti pojedinačnu diodu na matrici sa zajedničkom anodom po vrsti, tada moramo dovesti pozitivan napon na vrstu matrice, a masu na kolonu. Npr. ukoliko želimo upaliti prvu diodu po vrsti i koloni, tada ćemo na pin A dovesti pozitivan napon, a na kolonu 1 ćemo dovesti masu. Za display za zajedničkom katodom vrijedi analogno –inverzna logika.



Slika 9 Prikaz karaktera na displayu

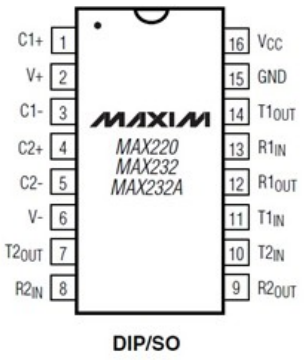
Da bi prikazali karakter na displayu sve ulaze moramo dovesti u odgovarajuće stanje, pa se tako npr. za slovo A dobija tabela stanja prikazana na slici 10.

Row\Col	C1	C2	C3	C4	C5
R1	0	1	1	1	0
R2	1	0	0	0	1
R3	1	0	0	0	1
R4	1	0	0	0	1
R5	1	1	1	1	1
R6	1	0	0	0	1
R7	1	0	0	0	1

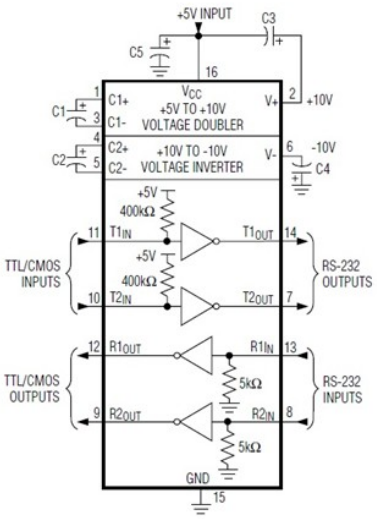
Slika 10 Prikaz stanja ulaza displaya

4. INTEGRISANO KOLO MAX232

Integrisano kolo MAX232 se koristi kao interface između mikrokontrolera i računara. Njegov zadatak je da konvertuje signale koji dolaze sa RS-232 serijskog porta i pretvori ih u signale pogodne za rad u kolima koja su kompatibilna sa TTL logičkim kolima. Izgled komponente i raspored pinova prikazan je na slici 8 , dok je struktura kola max232 prikazana na slici 9.



Slika11 Integrisano kolo *MAX232*



Slika 12 Struktura integrisanog kola MAX232

Integrisano kolo MAX232 sadrži sljedeće tri cjeline:

- dva DC-DC konvertora,
- dva RS232 drajvera,
- dva RS232 prijemnika.

Prvi DC-DC konvertor koristi eksterne kondenzatore C1 i C3 za udvostručavanje napona (sa +5V na +10V), dok drugi konvertor koristi eksterne kondenzatore C2 i C4 za invertovanje napona (sa +10V na -10V). Vrijednosti kondenzatora C1-C5 ne treba da budu manje od 100nF (najčešće se stavljaju vrijednosti od 100 ili 220nF, a u izuzetnim slučajevima, kada treba obezbijediti veće izlazne struje, vrijednosti su do 4.7uF).

RS232 drajveri (sastoje od logičkih invertora i otpornika) vrše transformaciju

ulaznih RS232 signala na sljedeći način:

- logičku '0' transformišu u signal vrednosti +8V (pri opterećenju 5KΩ),
- logičku '1' transformišu u signal vrednosti -8V (pri opterećenju 5KΩ).

RS232 prijemnici koji se sastoje od logičkih invertora i otpornika vrše

transformaciju ulaznih RS232 signala na sljedeći način:

- signal u opsegu od +3V do +25V transformišu u logičku '0',
- signal napona u opsegu od -3V do -25V transformišu u logičku '1'.

Osobine:

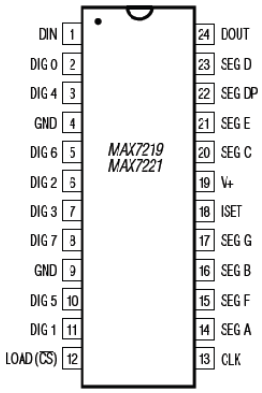
- Ulazni naponski nivoi su kompatibilni sa standardnim CMOS nivoima
- Izlzazni naponski nivoi su kompatibilni sa EIA/TIA-232-E nivoima
- Napon napajanja: 5V
- Niska ulazna struja: 0.1 μA na TA = 25°C

Primjena:

- Baterijski napajani RS232 sistemi
- Teriminali
- Modemi
- Računari

5. INTEGRISANO KOLO MAX7219

Integrisano kolo MAX7219 je display driver pomoću kojeg možemo upravljati sa 7-segmentim displayima do 8 cifara, bar-graph displayima, ili sa 64 diode pojedinačno (led matrix display).



Slika 13 Integrisano kolo MAX7219

Tabela 2 Opis pinova

PIN	ULOGA	OPIS
1	DIN	Ulaz serijske komunikacije. Podaci se učitavaju u interni 16 bitni shift registar na svaki takt.
2,3, 5 – 8, 10, 11	DIG 0-DIG7	Linije koje se povezuju na katodu displaya i koje "vuku" struju iz displaya. Kada nisu aktivne daju +V.
4,9	GND	Masa (oba pina GND moraju biti spojena na masu)
12	CS'	Chip Select ulaz. Podaci se učitaju u shift registar kada je CS' nula.
13	CLK	Ulaz za takt za serijku komunikaciju. Max vrijednost 10MHz. Na uzlaznu ivicu takta podaci se učitaju.
14-17,20-24	SEG A-SEG G,DP	Izlazne linije koje se povezuju na anodu drivera i koje daju struju. Kada nisu aktivne u stanju su visoke impedance.
18	ISET	Povezuje se na napajanje preko otpornika Rset koji postavlja maximalnu struju.
19	V+	Napon napajanja +5V
24	DOUT	Izlaz serijske komunikacije. Podaci primljeni na DIN su validni na DOUT nakon 16.5 taktova. Ovaj pin se koristi za kaskadnu vezu max7219 integrisanih kola.

MAX7219 prima podatke od mikrokontrolera preko serijske komunikacije. Također ima opciju da proslijedi primljene podatke ka sljedećem istom integrisanom kolu, čime je lako ostvariva kaskadna veza.

Tabela 3 Format ulaznih podataka

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

Podatak koji prima kolo čine 2 bajta. Prva 4 bita su bez značaja, dok naredna 4 bita nose adresu registra kojem mikrokontroler pristupa. Drugi bajt su podaci. U tabeli 4 je lista registara integrisanog kola.

Tabela 4 Lista registara integrisanog kola

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

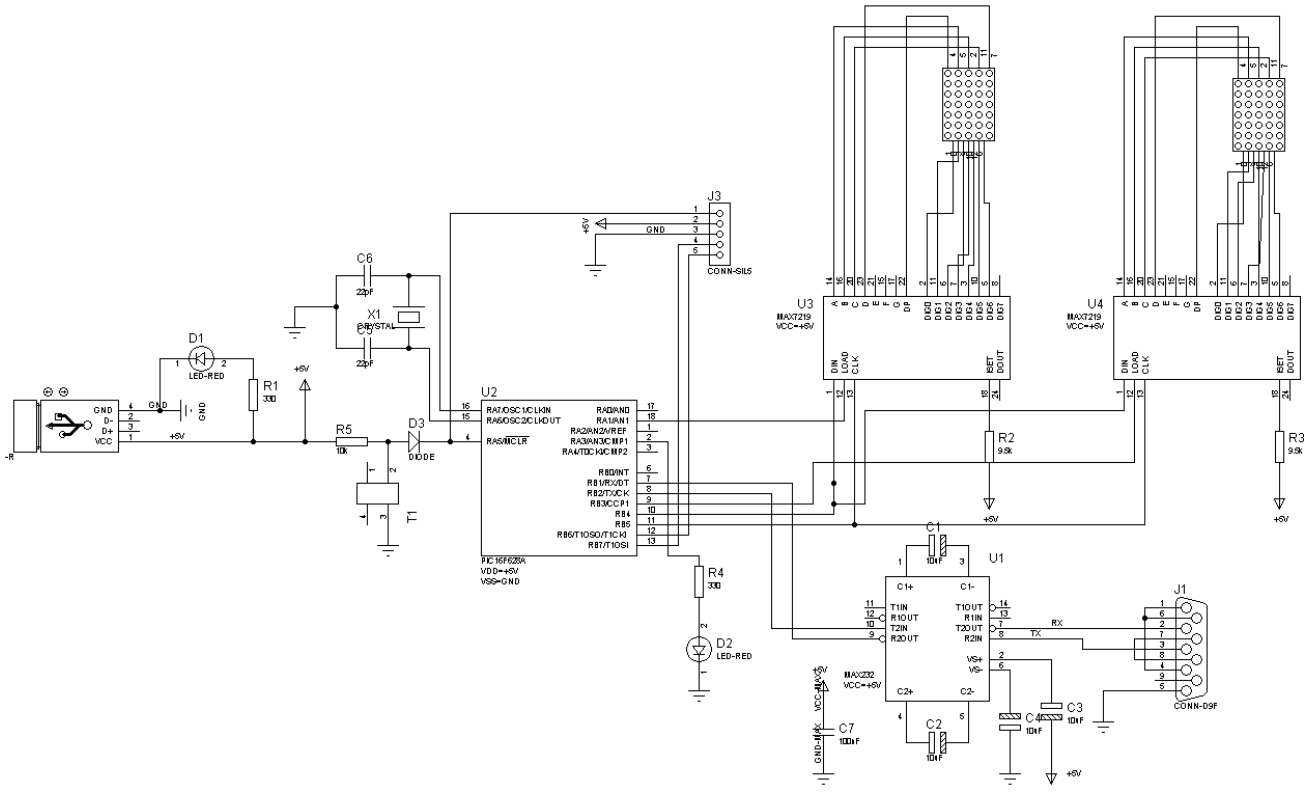
Potrebno je izvršiti inicijalizaciju integrisanog kola prije nego pokrene display. Treba upaliti display, ugasiti testni režim, podesiti intenzitet svjetlosti, a to se sve radi pristupanjem svakom od registara u tabeli 3.

6. PRAKTIČNA REALIZACIJA

Praktična realizacija na ovu temu satoji se iz nekoliko faza:

- Pisanje koda u programskom jeziku MicroC
- Izrada simulacione šeme u programu Proteus Isis
- Izrada šeme u programskom paketu Proteus Isis Ares na osnovu koje je izrađena štampana (PCB- Printed Circuit Board) pločica
- Izrada PCB pločice i postavljanje komponenti na pločicu
- Testiranje serijske komunikacije (RS-232) korištenjem virtualnog terminal Hercules

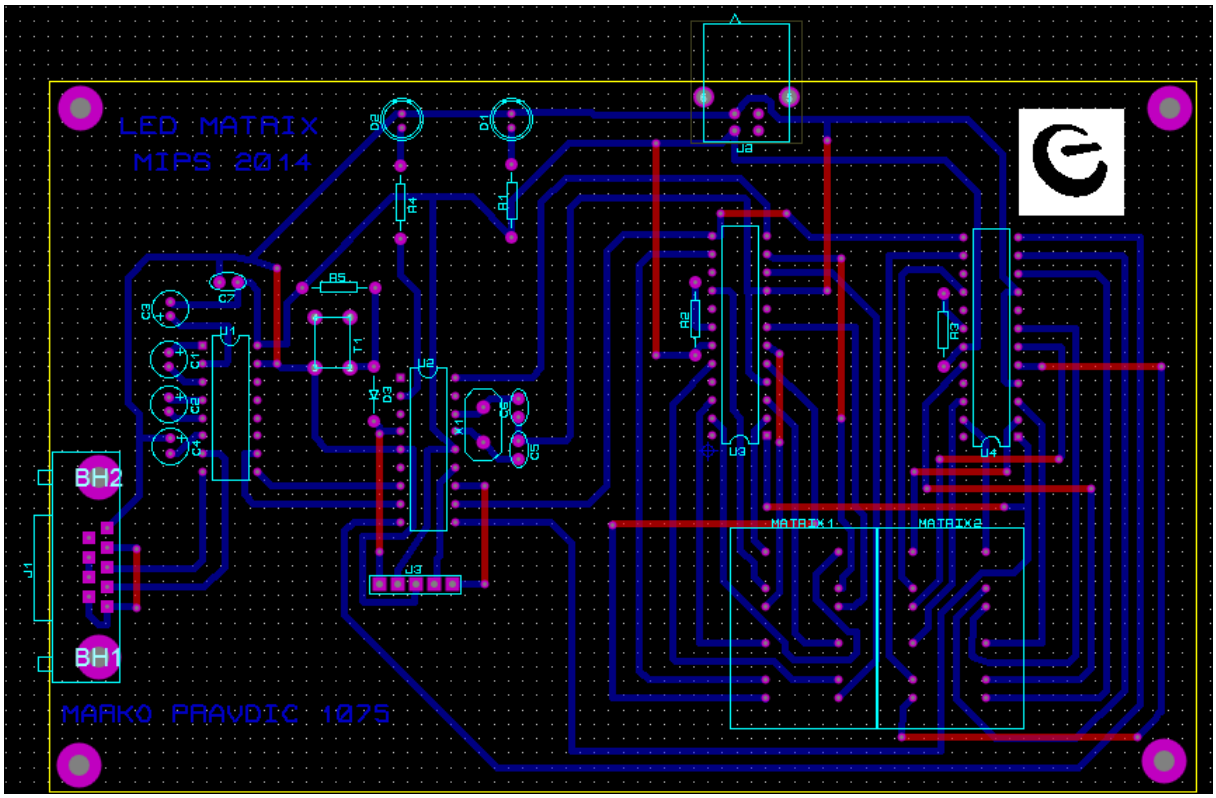
6.1 Hardverska realizacija



Slika 14 Električna šema projektnog zadatka

Tabela 5. Lista komponenti

Kategorija	**Količina**	**Oznaka na šemi**	**Vrijednost**
Kondenzatori	4	C1-C4	10uF
Kondenzatori	2	C5-C6	22pF
Kondenzatori	1	C7	100nF
Otpornici	2	R1,R4	330
Otpornici	2	R2-R3	9.5k
Otpornici	1	R5	10k
Integrisana kola	1	U1	MAX232
Integrisana kola	1	U2	PIC16F628A
Integrisana kola	2	U3-U4	MAX7219
Diode	2	D1-D2	LED-RED
Diode	1	D3	DIODE
MAX 232 Konektor	1	J1	CONN-D9F
ISCP konektor	1	J2	AU-Y1007-R
USB konektor za napajanje	1	J3	CONN-SIL5
Led matrix display	2	MATRIX1-MATRIX2	
Taster	1	T1	



Slika15 Izgled štampane pločice u Ares-u

6.2 Softverska realizacija

```
// Software SPI configuration
sbit SoftSpi_SDI at RA2_bit;
sbit SoftSpi_SDO at RB4_bit;
sbit SoftSpi_CLK at RB5_bit;
sbit SoftSpi_SDI_Direction at TRISA2_bit;
sbit SoftSpi_SDO_Direction at TRISB4_bit;
sbit SoftSpi_CLK_Direction at TRISB5_bit;

// max7219 LOAD/CS(-) pin configuration
sbit Chip_Select at RB3_bit;
sbit Chip_Select_Direction at TRISB3_bit;
sbit Chip_Select1 at RA0_bit;
sbit Chip_Select1_Direction at TRISA0_bit;

char message[50],msg[35];
unsigned char i,StringLength,k;
```

```
unsigned char temp;

short  l, scroll, shift_step=1,row,speed;
char index;

const char UART_txt1[] = "-8X16 LED Matrix Display -\r\n";
const char UART_txt2[] = "Zelite li unos poruke? D/N\r\n";
const char UART_txt3[] = "Zavrsite poruku sa slovom f\r\n";
const char UART_txt4[] = "Cekanje na poruku(MAX. 50 char.)\r\n";
const char UART_txt5[] = "Poruka primljena\r\n";

//Buffer registar for displaying message
unsigned short Buffer[7][2] = {
                        {0,0},
                        {0,0},
                        {0,0},
                        {0,0},
                        {0,0},
                        {0,0},
                        {0,0},
                       };
//Structure of characters data including č, ž, ć, đ, dž
const unsigned short CharData[][7] ={
{0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000},
{0b00000100, 0b00000100, 0b00000100, 0b00000100, 0b00000100, 0b00000000,  0b00000100},
{0b00001010, 0b00001010, 0b00001010, 0b00000000, 0b00000000, 0b00000000, 0b00000000},
{0b00000000, 0b00001010, 0b00011111, 0b00001010, 0b00011111, 0b00001010,  0b00011100},
{0b00000111, 0b00001100, 0b00010100, 0b00001100, 0b00000110, 0b00000101, 0b00011100},
{0b00011001, 0b00011010, 0b00000010, 0b00000100, 0b00000100, 0b00001000, 0b00001011},
{0b00000110, 0b00001010, 0b00010010, 0b00010100, 0b00001001, 0b00010110, 0b00001001},
{0b00000100, 0b00000100, 0b00000100, 0b00000000, 0b00000000, 0b00000000, 0b00000000},
{0b00000010, 0b00000100, 0b00001000, 0b00001000, 0b00001000, 0b00001000, 0b00000100},
{0b00001000, 0b00000100, 0b00000010, 0b00000010, 0b00000010, 0b00000010, 0b00001000},
{0b00010101, 0b00001110, 0b00011111, 0b00001110, 0b00010101, 0b00000000, 0b00000000},
{0b00000000, 0b00000000, 0b00000100, 0b00000100, 0b00011111, 0b00000100, 0b00000100},
{0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000110, 0b00001000},
{0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00001110, 0b00000000, 0b00000000},
{0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000100},
{0b00000001, 0b00000010, 0b00000010, 0b00000100, 0b00000100, 0b00001000, 0b00010000},
{0b00001110, 0b00010001, 0b00010011, 0b00010001, 0b00010101, 0b00010001, 0b00001110},
{0b00000100, 0b00001100, 0b00010100, 0b00000100, 0b00000100, 0b00000100, 0b00011111},
{0b00001110, 0b00010001, 0b00010001, 0b00000010, 0b00000100, 0b00001000, 0b00011111},
{0x1C,0x22,0x02,0x0C,0x02,0x22,0x1C},
{0b00010000, 0b00010000, 0b00010100, 0b00010100, 0b00011111, 0b00000100, 0b00000100},
{0b00011111, 0b00010000, 0b00010000, 0b00011110, 0b00000001, 0b00000001, 0b00011110},
```

```
{0b00000111, 0b00001000, 0b00010000, 0b00011110, 0b00010001, 0b00010001, 0b00001110},
{0b00011111, 0b00000001, 0b00000001, 0b00000001, 0b00000010, 0b00000100, 0b00010000},
{0b00001110, 0b00010001, 0b00010001, 0b00001110, 0b00010001, 0b00010001, 0b00001110},
{0x1C,0x22,0x22,0x1E,0x02,0x22,0x1C},
{0b00000000, 0b00000100, 0b00000100, 0b00000000, 0b00000000, 0b00000100, 0b00000000},
{0b00000000, 0b00000100, 0b00000100, 0b00000000, 0b00000000, 0b00000100, 0b00001000},
{0b00000001, 0b00000010, 0b00000100, 0b00001000, 0b00001000, 0b00000100, 0b00000001},
{0b00000000, 0b00000000, 0b00000000, 0b00011110, 0b00000000, 0b00011110, 0b00000000},
{0b00010000, 0b00001000, 0b00000100, 0b00000010, 0b00000010, 0b00000100, 0b00010000},
{0b00001110, 0b00010001, 0b00010001, 0b00000010, 0b00000100, 0b00000100, 0b00000100},
{0b00001110, 0b00010001, 0b00010001, 0b00010101, 0b00010101, 0b00010001, 0b00011110},
{0b00001110, 0b00010001, 0b00010001, 0b00010001, 0b00011111, 0b00010001, 0b00010001},
{0b00011110, 0b00010001, 0b00010001, 0b00011110, 0b00010001, 0b00010001, 0b00011110},
{0b00000111, 0b00001000, 0b00010000, 0b00010000, 0b00010000, 0b00010000, 0b00000111},
{0b00011100, 0b00010010, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00011100},
{0b00011111, 0b00010000, 0b00010000, 0b00011110, 0b00010000, 0b00010000, 0b00011111},
{0b00011111, 0b00010000, 0b00010000, 0b00011110, 0b00010000, 0b00010000, 0b00010000},
{0b00001110, 0b00010001, 0b00010000, 0b00010000, 0b00010111, 0b00010001, 0b00001110},
{0b00010001, 0b00010001, 0b00010001, 0b00011111, 0b00010001, 0b00010001, 0b00010001},
{0b00011111, 0b00000100, 0b00000100, 0b00000100, 0b00000100, 0b00000100, 0b00011111},
{0x1E,0x04,0x04,0x04,0x04,0x24,0x18},
{0b00010001, 0b00010010, 0b00010100, 0b00011000, 0b00010100, 0b00010010, 0b00010001},
{0b00010000, 0b00010000, 0b00010000, 0b00010000, 0b00010000, 0b00010000, 0b00011111},
{0b00010001, 0b00011011, 0b00011111, 0b00010101, 0b00010001, 0b00010001, 0b00010001},
{0b00010001, 0b00011001, 0b00011001, 0b00010101, 0b00010101, 0b00010011, 0b00010001},
{0b00001110, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00001110},
{0b00011110, 0b00010001, 0b00010001, 0b00011110, 0b00010000, 0b00010000, 0b00010000},
{0b00001110, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00010101, 0b00001111},
{0b00011110, 0b00010001, 0b00010001, 0b00011110, 0b00010100, 0b00010010, 0b00010001},
{0x0E,0x11,0x10,0x0E,0x01,0x11,0x0E},
{0b00011111, 0b00000100, 0b00000100, 0b00000100, 0b00000100, 0b00000100, 0b00000100},
{0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00001110},
{0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00000100},
{0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00010001, 0b00010101, 0b00001010},
{0b00010001, 0b00010001, 0b00001010, 0b00000100, 0b00000100, 0b00001010, 0b00010001},
{0b00010001, 0b00010001, 0b00001010, 0b00000100, 0b00000100, 0b00000100, 0b00000100},
{0b00011111, 0b00000001, 0b00000010, 0b00000100, 0b00001000, 0b00010000, 0b00011111},
{0x14,0x08,0x1C,0x22,0x20,0x22,0x1C},
{0x04,0x08,0x1C,0x22,0x20,0x22,0x1C},
{0x78,0x44,0x44,0xe4,0x44,0x44,0x78},
{0x14,0x08,0x1e,0x20,0x3c,0x02,0x3c},
{0x14,0x08,0x3E,0x04,0x08,0x10,0x3e},
};

unsigned short Find_StrLength()
```

```
{
 return strlen(message);

}
//Function to convert data from ROM to RAM (used for UART messages)
char * CopyConst2Ram(char * dest, const char * src)
{
 char * d ;
 d = dest;
 for(;*dest++ = *src++;)
  ;
 return d;
}


void ListenSerial()
{
 unsigned char RxByte;

 asm CLRWDT;


 if (UART1_Data_Ready())
  {
  RxByte = UART1_Read();
  if(RxByte == 'D')
  {
   UART1_Write_Text(CopyConst2Ram(msg,Uart_txt3));
   UART1_Write_Text(CopyConst2Ram(msg,Uart_txt4));
   //CONFIG=0x216A;
   while (!UART1_Data_Ready());

   UART1_Read_Text(message, "f", 250);    // reads text until 'f' is found

    asm CLRWDT;
   StringLength = Find_StrLength();

   UART1_Write_Text(CopyConst2Ram(msg,Uart_txt5));
   UART1_Write_Text(CopyConst2Ram(msg,Uart_txt1));
   UART1_Write_Text(CopyConst2Ram(msg,Uart_txt2));

  }
 }
}
// max7219 configuration
```

```
void max7219_init1()
{

 Chip_Select = 0;       // SELECT MAX
 Soft_SPI_Write(0x09);      //  Disable decoding
 Soft_SPI_Write(0x00);
 Chip_Select = 1;       // DESELECT MAX

 Chip_Select = 0;
Soft_SPI_Write(0x0A);     //  Max segment luminosity intensity
 Soft_SPI_Write(0x0F);
 Chip_Select = 1;

 Chip_Select = 0;
 Soft_SPI_Write(0x0B);
 Soft_SPI_Write(0x07);      // Display refresh
 Chip_Select = 1;



 Chip_Select = 0;
 Soft_SPI_Write(0x0C);
 Soft_SPI_Write(0x01);      // Turn on the display
 Chip_Select = 1;

 Chip_Select = 0;
 Soft_SPI_Write(0x00);
 Soft_SPI_Write(0xFF);      // No test
 Chip_Select = 1;
}

void max7219_init2()
{

 Chip_Select1 = 0;
 Soft_SPI_Write(0x09);
 Soft_SPI_Write(0x00);
 Chip_Select1 = 1;

 Chip_Select1 = 0;
Soft_SPI_Write(0x0A);
 Soft_SPI_Write(0x0F);
 Chip_Select1 = 1;

 Chip_Select1 = 0;
 Soft_SPI_Write(0x0B);
```

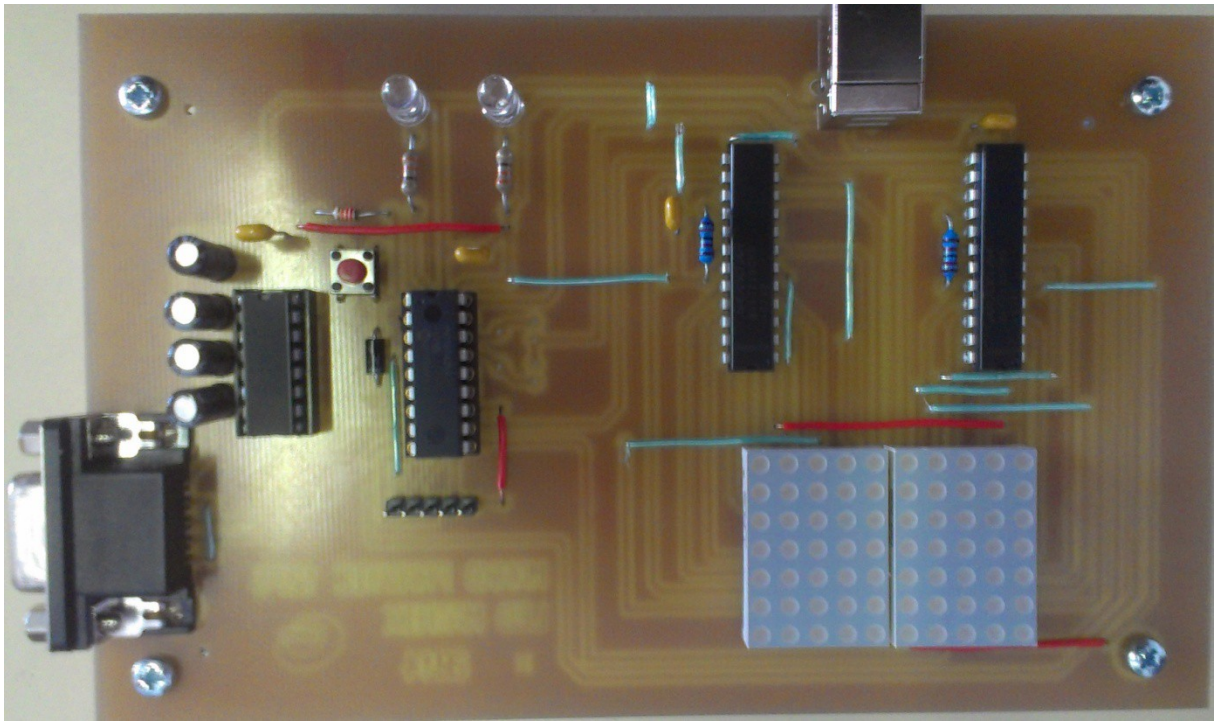
```
 Soft_SPI_Write(0x07);
 Chip_Select1 = 1;

 Chip_Select1 = 0;
 Soft_SPI_Write(0x0C);
 Soft_SPI_Write(0x01);
 Chip_Select1 = 1;

 Chip_Select1 = 0;
 Soft_SPI_Write(0x00);
 Soft_SPI_Write(0xFF);
 Chip_Select1 = 1;
}
//Send data function for fist max
 void Write_Byte_First(unsigned short Row, unsigned short Value)
{
    Chip_Select = 0;
    Soft_SPI_Write(Row);   //send adress
    Soft_SPI_Write(Value);   //send value
    Chip_Select = 1;
 }
////Send data function for second max
 void Write_Byte_Second(unsigned short Row, unsigned short Value)
{
     Chip_Select1 = 0;
     Soft_SPI_Write(Row);
     Soft_SPI_Write(Value);
     Chip_Select1 = 1;

}
//Clear matrix function
void Clear_Matrix(void)
{
 unsigned short x;

 for(x=1;x<9;x++)
 {
  Write_Byte_First(x,0x00);
  Write_Byte_Second(x,0x00);
  }
}
void Find_Char()
{
    if (message[k]>100)    //Find č, ć, ž, š, đ
     {
```

```
        switch (message[k])
        {
          case 200: temp = CharData[59][row]; break;
          case 198: temp = CharData[60][row]; break;
          case 208: temp = CharData[61][row]; break;
          case 138: temp = CharData[62][row]; break;
          case 142: temp = CharData[63][row]; break;
        }
    }
    else
        {
          index = message[k];
          temp = CharData[index-32][row];
        }
        asm CLRWDT;
}


void main()
{     unsigned char RxByte;
      TRISA=0;
      //UART pins
      TRISB.b1=1;
      TRISB.b2=0;

      //Status led
      TRISA.b3=0;
      PORTA.b3=0;

      OPTION_REG=0b00101110;

      //MAXX tris pins
      Chip_Select_Direction = 0;
      Chip_Select1_Direction = 0;

      UART1_Init(9600);
      delay_ms(100);

      Soft_SPI_init();
      delay_ms(100);

      max7219_init1();
      max7219_init2();
      asm CLRWDT;
```

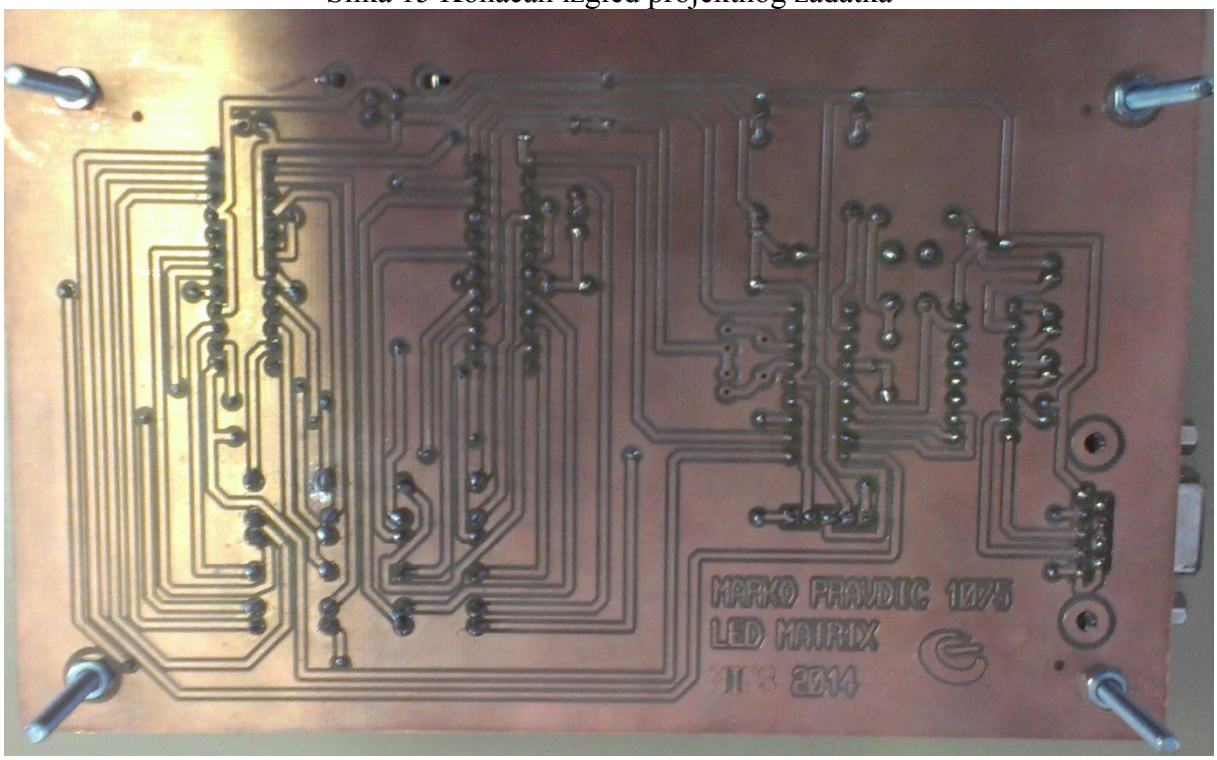
```
Clear_Matrix();

 StringLength = Find_StrLength();

//Test displays
for (i=1;i<8;i++)
{
Write_Byte_First(i,0xff);
Write_Byte_Second(i,0xff);
delay_ms(100);
 }

 Clear_Matrix();

UART1_Write_Text(CopyConst2Ram(msg,Uart_txt1));
UART1_Write_Text(CopyConst2Ram(msg,Uart_txt2));
asm CLRWDT;

while(1)
{
    ListenSerial();

   for (k=0; k<StringLength; k++)
   {

      PORTA.b3=~PORTA.b3;

    for (scroll=0; scroll<(8/shift_step); scroll++)
     {
       for (row=0; row<7; row++)
       {
           Find_Char();
         Buffer[row][0] = (Buffer[row][0] << Shift_Step)| (temp >> ((5-shift_step)-scroll*shift_step));
         Buffer[row][1] = (Buffer[row][1] << Shift_Step) | (Buffer[row][0] >> (5-Shift_Step));


       }
         speed=5;
          asm CLRWDT;
         for(l=0; l<speed;l++)
      {
          for (i=0; i<7; i++)
            {
               Write_Byte_First(i+1,Buffer[i][0]);
               Write_Byte_Second(i+1,Buffer[i][1]);
            }
```

```
        }
        }
       }
    }
}
```

6.3 Konačan izgled projektnog zadatka



Slika 15 Konačan izgled projektnog zadatka



Slike 16 Konačan izgled PCB pločice

7. ZAKLJUČAK

Cilj projektnog zadatka je bio da se preko integrisanih kola MAX7219 (LED driveri) i pomoću mikrokontrolera PIC 16F628a izvrši ispis tekstualne poruke na dvije LED matrice 7x5. Mikrokontroler sa računarom ostvaruje komunikaciju putem RS232 modula. Omogućen je unos nove tekstualne poruke bez potrebe za resetom mikrokontrolera. Integrisano kolo MAX232 je korišten kao prilagođivač naponskih nivoa između mikrokontrolera i računara. Računar šalje podatke mikrokontroleru, a ovaj putem softverske SPI komunikacije podatke šalje ka LED driverima, a isti upravljaju displayima. Tekst poslan ka LED matricama treba da se smiče slijeva na desno. Na LED matricu je moguće ispisati velika slova, brojeve i specijalne karaktere. Također je omogućen ispis naših slova (č, ć, ž, š, đ, dž). Brzina smicanja teksta se može softverski regulisati (promjenjiva speed u .C programu).

8. LITERATURA

[1] PIC16F628A, Datasheet, Microchip
[2] MAX7219, Datasheet, Maxim
[3] MAX232, Datasheet, Maxim
[4] PIC Microcontrolers, Milan Verle
[5] http://www.appelsiini.net/2011/how-does-led-matrix-work
[6] http://embedded-lab.com/blog/?p=2478 (Basics of LED dot matrix display)
[7] http://en.wikipedia.org/wiki/Dot-matrix_display