

UNIVERZITET U SARAJEVU,
ELEKTROTEHNIČKI FAKULTET

PRAKTIKUM AUTOMATIKE

Prikaz slike na osciloskopu

Studenti:

Samra Salihić, 17980
Fatih Zukorlić, 17861

Asistent:

Mr.Sci. Nedim Osmić

Januar, 2019

SAŽETAK RADA

Tema ovog rada je prikaz slike na osciloskopu realiziran na mikrokontroleru Microchip PIC16F1939. Pored prikaza, aplikacija nudi mogucnost obrade i prijenosa slike sa racunara na mikrokontroler putem serijske komunikacije. U svrhu izrade dijela aplikacije koji se izvrsava na mikrokontroleru koristen je XC8 kompajler unutar okruzenja MPLAB X, dok je za izradu dijela aplikacije koji se pokrece na racunaru koristen Matlab softverski paket. Nacin rada samog crtanja slike na osciloskop se oslanja na nacin rada CRT ekrana. Naime, na ekranu je u trenutku prikazana samo jedna tacka, te se brzinom mijenjanja pozicije te tacke stvara iluzija slike. Crtanje svih, po redu neparnih, linija se odvija slijeva na desno (prvo se crtaju tacke koje su na lijevoj strani), dok se crtanje linija parnih po redu vrsti zdesna na lijevo. Naponski nivoi koji predstavljaju pozicije tacke koja se prikazuje su dovedeni na osiloskop preko digitalno analognih konvertora. Slika formata 30x30 se cuva u RAM memoriji mikrokontrolera, te se moze izmijeniti tako sto korisnik serijskom komunikacijom na modul posalje matricu karaktera(’0’ ili ’1’) dimenzija 30x30.

The purpose of this project is displaying picture on oscilloscope, using Microchip PIC16F1939. Along with displaying the picture, application lets user process and send images from computer to microcontroller via serial communication. For the purpose of developing a part of the application that runs on microcontroller, we used XC8 compiler inside MPLAB X environment, whilst for developing a part that runs on computer, we used Matlab environment. Basic idea of displaying picture on oscilloscope was borrowed from the working principle of CRT displays. To be precise, there is only one dot displayed at the time, and by changing the dot’s position rapidly, we create an illusion of displaying a picture. All pixels located in odd rows are drawn from left to right, while pixels located in even rows are drawn from right to left. Positions of the dot are defined by voltage levels, regulated by digital to analog converters. Image sized 30x30 is stored in the RAM memory of microcontroller and can be overwritten if the user sends same sized matrix of characters(’0’ or ’1’) over serial communication.

Contents

1 UVOD	3
2 OPIS POJEDINIХ DIJELOVA SISTEMA	4
2.1 Digitalno-analogni konvertor	4
2.2 Serijska komuniakcija	6
2.3 Matlab	7
2.4 Mikrokontroler Microchip PIC16F1939	10
3 EKSPERIMENTALNI REZULTATI	18
4 ZAKLJUČAK	22
5 LITERATURA	23

1 UVOD

Za izradu ovog projekta bilo je potrebno realizovati 2 digitalno-analogna konvertora. Dva digitalno-analogna konvertora su bila potrebna da bi se uspostavilo brže slanje signala na osciloskop.

Sama ideja kako bi se ovaj projektni zadatak realizovao leži u mogućnosti da osciloskpi imaju tzv. XY mode rada. Model osciloskopa koji je korišten za izradu ovog projektong zadatka je prikazan na slici 1.



Slika 1: Osciloskop V 252 20MHz

Da bi bilo moguće prikazati sliku na osciloskopu potrebno je prije svega tu sliku obraditi i dobiti je u formatu koji možemo iskoristiti za dalju obradu. Tu obrađenu sliku putem Matlaba, serijskom komunikacijom, šaljemo na razvojni sistem PIC16F1939. PIC16F1939 je isprogramiran tako da primi sliku, sačuva je kao matricu bita, i onda bit po bit šalje na osciloskop. Kako sve to ustavri funkcioniše bit će objašnjeno u nastavku rada.

2 OPIS POJEDINIХ DIJELOVA SISTEMA

Prvo će biti analizirana izrada digitalno-analognih konvertora, pa zatim serijska komunikacija između modula i računara preko programskog okruženja Matlab.

2.1 Digitalno-analogni konvertor

Kako je već rečeno, slika koja se želi prikazati na osciloskopu predstavlja se matricom bita gdje se svaki bit zasebno gleda i prikazuje/ne prikazuje. Sam prikaz slike je sveden na tačke tj. piksele te slike. Kako osciloskop ima XY mode, tj. za prikaz jedne tačke potrebne su dvije vrijednosti signalata - X vrijednosti i Y vrijednost. Na osnovu toga možežemo zaključiti da su nama potrebna dva digitalno-analogna konvertora. Za izradu jednog digitalno-analognog konvertora potrebni su sljedeci elemtni:

- Operaciono pojačalo UA741 NC
- Otpornici (vrijednost otpora će biti izračunata u nastavku)

Korištena je realizacija *"Binary Weighted Resistor DAC"*. Kako nam je bilo potrebno samo 7 bita izlaza, potrebno je odrediti odgovarajuće vrijednosti otpora da bi dobili željeni izlaz. Sljedeći sistem jednačina (1) nam omogućava da odredimo vrijednosti tih otpora.

$$\frac{V_{ref}}{R}D_6 + \frac{V_{ref}}{2R}D_5 + \frac{V_{ref}}{4R}D_4 + \frac{V_{ref}}{8R}D_3 + \frac{V_{ref}}{16R}D_2 + \frac{V_{ref}}{32R}D_1 + \frac{V_{ref}}{64R}D_0 = -\frac{U_{izl}}{R_f} \quad (1)$$

Kako želimo na izlazu -5V/5V kada je na ulazu vrijednost 79, binarno zapisano kao 01001111, i kako je $V_{ref} = 5V$ dolazimo do sljedećih jednačina, (2) i (3).

$$\frac{5}{R} + \frac{5}{8R} + \frac{5}{16R} + \frac{5}{32R} + \frac{5}{64R} = -\frac{5}{R_f} \quad (2)$$

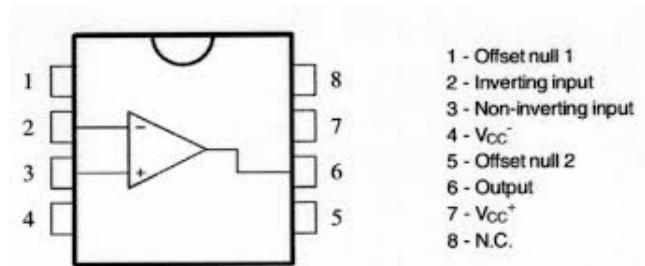
$$\frac{5}{R} + \frac{5}{8R} + \frac{5}{16R} + \frac{5}{32R} + \frac{5}{64R} = -\frac{-5}{R_f} \quad (3)$$

Nakonsređivanja dobijamo potreban odnos otprinika R i R_f , (4)

$$\frac{R}{R_f} = \frac{79}{64} \quad (4)$$

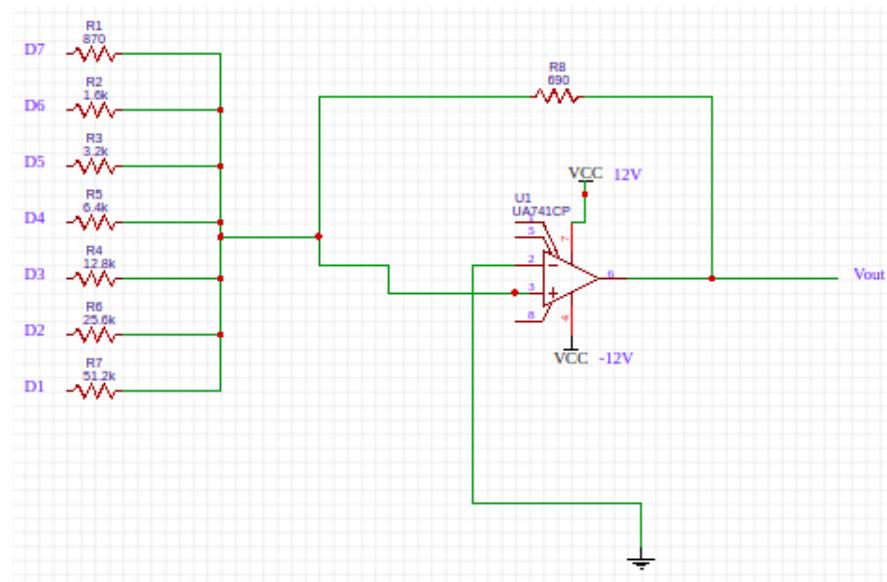
Za otpornik R smo uzeli vrijednost 870 ohma, da bi vrednost struje bila isporučena 15mA, samim time za R_f dobijamo vrijednost otpora 680 ohma.

Još je potrebno napomenuti da je za realizaciju potrebno operaciono pojačalo UA741 NC. Pin dijagram UA741NC prikazan je na slici 2.

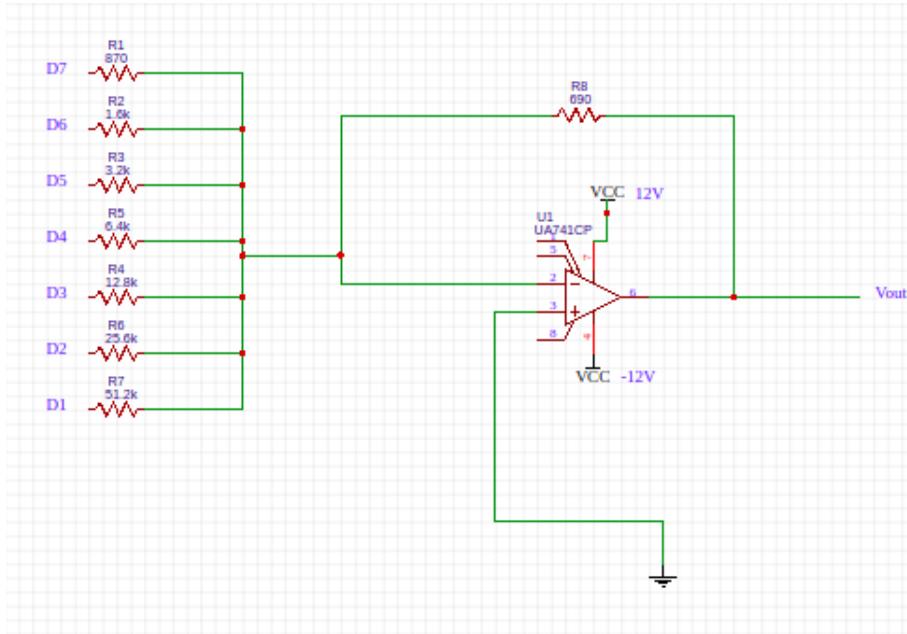


Slika 2: Pin dijagram UA741 NC

Na slikama 3 i 4 prikazane su sheme spajanja za realizaciju digitalno-analognog konvertora.



Slika 3: DA konvertor sa pozitivnim izlaznim naponom



Slika 4: DA konvertor sa pozitivnim izlaznim naponom

2.2 Serijska komuniakcija

Za uspostavljanje serijske komunikacije korišten je TTL-232-3V3-PCB komunikacijski modul. Modul se napaja sa računara, preko USB porta, tako da je Vcc izlaz na kojem daje +5V koje možete koristiti za neku namjenu, ali nije preporučljivo. GND je potrebno spojiti sa GND mikrokontrolera. Pinovi CTS i RTS služe za kontrolu toka slanja i prijema podataka, tj. koriste se kao kontrolni biti. Izgled modula je prikazan na slici 5, a raspored pinova modula na slici 6.



Slika 5: Izgled modula



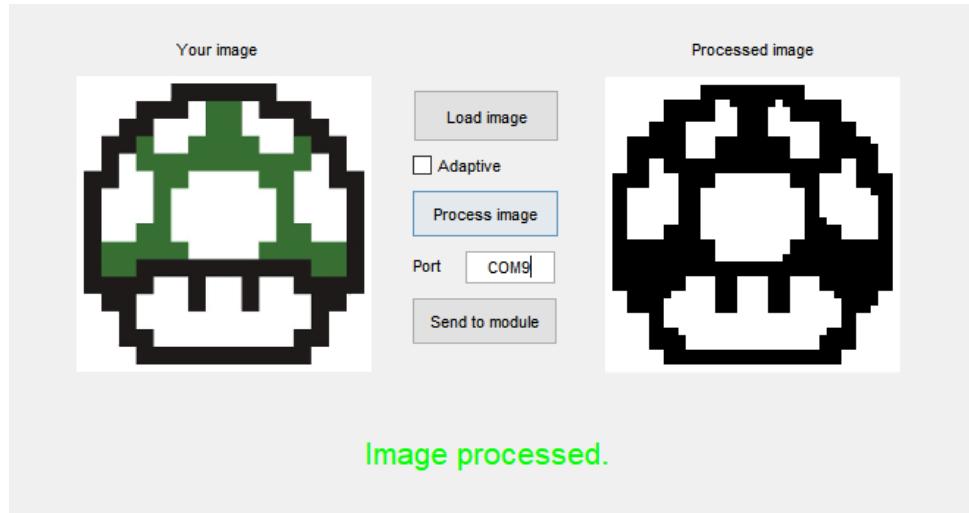
TTL232R PCB PADS

Slika 6: Izgled modula

Što se tiče povezivanja mikrokontrolera sa ovim modulom, potrebno je povezati Tx pin (RC6) mikrokontrolera sa Rx pinom komunikacijskog modula i Rx pin (RC7) mikrokontrolera sa Tx pinom komunikacijskog modula.
Sam prenos podataka na mikrokontroler će biti opisan u narednom poglavlju *Matlab*.

2.3 Matlab

Za slanje podatka korišten je Matlab-a jer on nudi gotove funkcije za komunikaciju. Da bi rukovanje prenesom slika bilo jednostavnije napravljen je GUI koji izgleda kao na slici 10. Vidimo da se GUI sastoji od 3 dugmeta, jednog input polja i checkbox-a.



Slika 7: Izgled GUI-a Matlab-a

Dugme "Load image" omogućava korisniku da učita sliku koju želi prikazati. Listing koda koji omogućava učitavanje slike je dat u nastavku.

```

1
2
3 function loadImage_Callback(hObject, eventdata, handles)
4 % hObject    handle to loadImage (see GCBO)
5 % eventdata   reserved - to be defined in a future version of MATLAB
6 % handles    structure with handles and user data (see GUIDATA)
7 try
8     [filename, user_cancelled] = imgetfile;
9     orig = imread(filename);
10    imshow(orig, 'Parent', handles.axes1);
11    handles.orig=orig;
12    set(handles.poruka, 'ForegroundColor', 'Green');
13    set(handles.poruka, 'String', 'Image loaded. Process image.');
14 catch e
15     set(handles.poruka, 'ForegroundColor', 'Red');
16     set(handles.poruka, 'String', 'No image loaded!');
17 end
18 guidata(hObject, handles);

```

Dugme "Process image" ustvari procesira sliku na nacin da je prikaze kao niz byte-a. Za ovaj proces vezan je checkbox *Adaptive* i kada je on označen ustvar se vrši adaptivna obrada slike. Listing koda je prikazan u nastavku.

```

1
2 function processImage_Callback(hObject, eventdata, handles)
3 % hObject    handle to processImage (see GCBO)
4 % eventdata   reserved - to be defined in a future version of MATLAB
5 % handles    structure with handles and user data (see GUIDATA)
6
7
8 try
9     orig=handles.orig;
10    type=handles.type;
11    if ~strcmp(type, 'adaptive') && ~strcmp(type, 'global')
12        type='global';
13    end
14    processed = imresize(imbinarize(rgb2gray(orig), type), [40 40]);
15    imshow(processed, 'Parent', handles.axes2);
16    handles.processed=processed;
17    handles.type=type;
18    set(handles.poruka, 'ForegroundColor', 'Green');
19    set(handles.poruka, 'String', 'Image processed.');
20 catch e
21     set(handles.poruka, 'ForegroundColor', 'Red');
22     set(handles.poruka, 'String', 'Load image first!');
23 end

```

Dugme "Send to module" priprema obradjene podatke za slanje, uspostavlja komu-

nikaciju i šalje podatke. Za ovaj proces vezano je i input polje Port jer on ustavlja sadrži ime porta na koji će se podaci slati. Bitno je napomenuti da pri svakom slanju sliku je potrebno zatvoriti i obrisati otvorenu konekciju. Listing koda koji ovo omogućava je prikazan u nastavku.

```
1
2
3
4 % —— Executes on button press in sendImage.
5 function sendImage_Callback(hObject, eventdata, handles)
6 % hObject    handle to sendImage (see GCBO)
7 % eventdata   reserved – to be defined in a future version of MATLAB
8 % handles    structure with handles and user data (see GUIDATA)
9 try
10     slika = handles.processed;
11     bajti = '';
12     % generisanje bajti 1–bijelo 0–crno
13     for i=1:40
14         for j=1:40
15             bajti = strcat(bajti, num2str(slika(i,j)));
16         end
17     end
18     % kraj generisanja
19     'prije otvaranje'
20     try
21         fclose(s);
22     catch e
23     end
24     try
25         delete(s);
26     catch e
27     end
28     s = serial(handles.comPort, 'BaudRate', 9600, 'Terminator', '')
29     fopen(s);
30     'poslije opena'
31     s
32     for i=1:size(bajti,2)
33         fprintf(s, bajti(i));
34     %         t = tic();
35     %         while toc(t) < 0.00001
36     %         end
37     end
38     fclose(s);
39     'poslije closea'
40     s
41     delete(s);
42     s
43     clear s;
44     set(handles.poruka, 'ForegroundColor', 'Green');
```

```

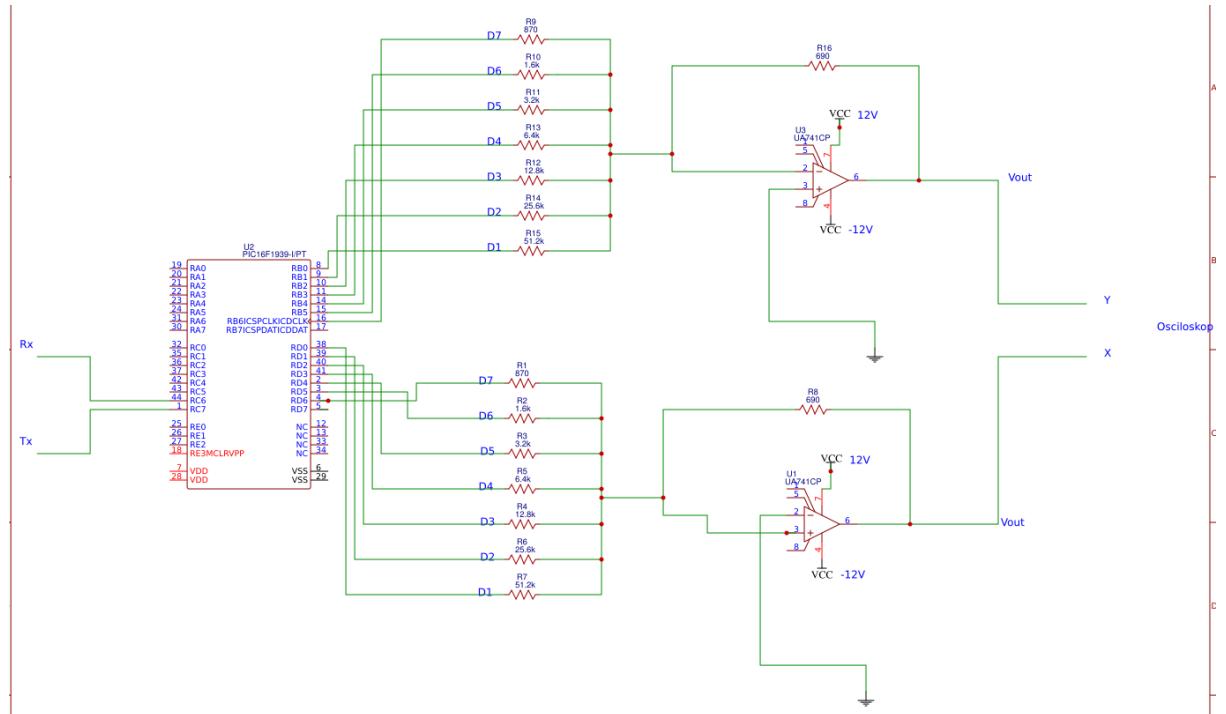
45     set(handles.poruka,'String','Image sent!');
46 catch e
47     e
48     set(handles.poruka,'ForegroundColor','Red');
49     set(handles.poruka,'String','Sending not successful!');
50     delete(s);
51 end

```

Potrebno je napomenuti da slike koje se procesiraju budu dimenzija 40x40.

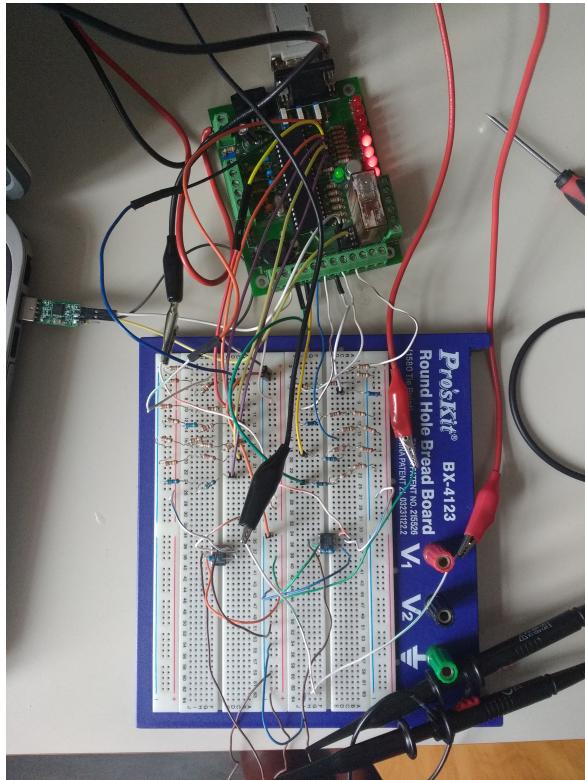
2.4 Mikrokontroler Microchip PIC16F1939

Prethodno je rečeno kako se povezuje mikrokontroler sa komunikacijskim modulom. Na slici 8 i 9 je prikazana shema spajanja mikrokontrolera sa prethodnom realizovanim digitalno-analognim konvertorima i komunikacijskim modulom.



Slika 8: Shema spajanja

Serijska komunikacija je realizovana putem prekida. Kada se završi komunikacija na našem mikrokontroleru bude sačuvana matrica koja ima sljedeći oblik: t_byte



Slika 9: Shema spajanja

```
slika[40][5] = {{0b11111111, 0b00000000, 0b00000000, 0b00000000, 0b00000000},  
{0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000},  
{0b11111111, 0b00000000, 0b00000000, 0b00000000, 0b00000000},  
{0b00000000, 0b00000000, 0b00000000, 0b00000000, 0b00000000}, ...};
```

Nakon prijema slike, vrši se prikazivanje iste do sljedećeg prekida odnosno dok korisnik ne zatraži prikaz nove slike. Bitno je napomenuti da je prikaz slike programiran tako da se prikaže jedan red slike sa lijeva ka desnu, a naredni red se prikazuje sa desna ka lijevu. Razlog tome je da se ne prekida snop svjetlosti koji prikazuje slku prelaskom na novi red, već da ide "cik-cak". Posmatrajući matricu koja se formira treba istaći da bit "1" na osciloskopu predstavlja postojanje piksela, a bit "0" obr-

nuto.

Kakvu ulogu u svemu igraju digitalno-analogni konvertori? Naime, kako se obradjuje matrica bit po bit, i za svaki bit koji postavljen tj. ima vrijednost "1" odredjene vrejdnsoti se pojavljuju na PORTB i PORTD. Npr. ako je prvi bit u matrici postavljen tada će se na portovima PORTB i PORTD prikazati vrijednost 0. Ako je drugi bit postavljen na 1, onda će na PORTB biti vrijednost 0 (y koordinata) a na PORTD vrijednost 1+const (x koordinata). Kada se predje na naredni red matrice tada se mijenja i vrijednost na PORTB.

Listing koda programiran na PIC16F1939 je prikazan u nastavku.

```
1 #include <xc.h>
2 #include <stdbool.h>
3 #include <stdint.h>
4 #include <stdio.h>
5
6
7 #pragma config FOSC=HS,WDTE=OFF,PWRTE=OFF,MCLRE=ON,CP=OFF,CPD=OFF,BOREN
8     =OFF,CLKOUTEN=OFF
9 #pragma config IESO=OFF,FCMEN=OFF,WRT=OFF,VCAPEN=OFF,PLLEN=OFF,STVREN=
10    OFF,LVP=OFF
11
12 void port_init() {
13     TRISD=0x00;
14     TRISB=0x00;
15     TRISC=0xBF;
16 }
17
18 void EUSART_Initialize()
19 {
20     // Set the EUSART module to the options selected in the user
21     // interface .
22
23     // ABDOVF no_overflow ; SCKP Non-Inverted ; BRG16 16 bit_generator ;
24     WUE disabled ; ABDEN disabled ;
25     BAUDCON = 0x08 ;
26
27     // SPEN enabled ; RX9 8-bit ; CREN enabled ; ADDEN disabled ; SREN
28     // disabled ;
29     RCSTA = 0x90 ;
30
31     // TX9 8-bit ; TX9D 0 ; SENDB sync_break_complete ; TXEN enabled ; SYNC
32     // asynchronous ; BRGH hi-speed ; CSRC slave ;
33     TXSTA = 0x24 ;
34
35     // SPBRGL 207;
36     SPBRGL = 0xCF ;
```

```

33     // SPBRGH 0;
34     SPBRGH = 0x00;
35
36
37 }
38 }
39
40 bool EUSART_is_tx_ready()
41 {
42     return (bool)(PIR1bits.TXIF && TXSTAbits.TXEN);
43 }
44
45 bool EUSART_is_rx_ready()
46 {
47     return PIR1bits.RCIF;
48 }
49
50 bool EUSART_is_tx_done()
51 {
52     return TXSTAbits.TRMT;
53 }
54
55 uint8_t EUSART_Read()
56 {
57     while (!PIR1bits.RCIF)
58     {
59     }
60
61
62     if (1 == RCSTAbits.OERR)
63     {
64         // EUSART error - restart
65
66         RCSTAbits.CREN = 0;
67         RCSTAbits.CREN = 1;
68     }
69
70     return RCREG;
71 }
72
73 void EUSART_Write(uint8_t txData)
74 {
75     while (0 == PIR1bits.TXIF)
76     {
77     }
78
79     TXREG = txData;      // Write the data byte to the USART.
80 }
81

```

```

82 char getch()
83 {
84     return EUSART_Read() ;
85 }
86
87 void putch(char txData)
88 {
89     EUSART_Write(txData) ;
90 }
91
92
93 #define on 0
94 #define off 1
95 typedef union {
96     unsigned char byte;
97     struct {
98         unsigned b0:1, b1:1, b2:1, b3:1, b4:1, b5:1, b6:1, b7:1;
99     };
100 } t_byte;
101
102 t_byte slika [40][5];
103
104 void ispisi(char *s){
105     while(*s){
106         EUSART_Write(*s) ;
107         s++;
108     }
109 }
110
111 void int_init(){
112     RCIE=1;
113     PEIE=1;
114     GIE=1;
115 }
116
117
118 void primiSliku(){
119     char temp;
120     for(char i=0;i<40;i++){
121         LATB=i;
122         for(char j=0;j<5;j++){
123             while (!EUSART_is_rx_ready()) ;
124             temp = EUSART_Read() ;
125             slika [i] [j].b7 = ((temp=='1') ? 1 : 0);
126             while (!EUSART_is_rx_ready()) ;
127             temp = EUSART_Read() ;
128             slika [i] [j].b6 = ((temp=='1') ? 1 : 0);
129             while (!EUSART_is_rx_ready()) ;
130             temp = EUSART_Read() ;

```

```

131         slika[i][j].b5 = ((temp=='1') ? 1 : 0);
132     while (!EUSART_is_rx_ready());
133         temp = EUSART_Read();
134         slika[i][j].b4 = ((temp=='1') ? 1 : 0);
135     while (!EUSART_is_rx_ready());
136         temp = EUSART_Read();
137         slika[i][j].b3 = ((temp=='1') ? 1 : 0);
138     while (!EUSART_is_rx_ready());
139         temp = EUSART_Read();
140         slika[i][j].b2 = ((temp=='1') ? 1 : 0);
141     while (!EUSART_is_rx_ready());
142         temp = EUSART_Read();
143         slika[i][j].b1 = ((temp=='1') ? 1 : 0);
144     while (!EUSART_is_rx_ready());
145         temp = EUSART_Read();
146         slika[i][j].b0 = ((temp=='1') ? 1 : 0);
147     }
148 }
149 }
150
151 void interrupt fija(){
152     if(RCIE==1 && RCIF==1){
153         LATD=0xff;
154         primiSliku();
155         RCIF=0;
156     }
157 }
158
159
160 void main(void)
161 {
162     // initialize the device
163     port_init();
164     EUSART_Initialize();
165     int_init();
166
167     char rxData;
168     ispisi("test");
169     TRISB=0x00; // y
170     TRISD=0x00; // x
171     /* LATB=0x00;
172     primiSliku();
173     LATB=0xff;
174     __delay_ms(500);*/
175     while(1){
176         for(char i=0;i<40;i++){
177             LATB = i;
178             for(char j=0;j<5;j++){
179                 char temp = j*8;

```

```

180         if( slika [ i ][ j ].b7 == on){
181             LATD = temp ;
182         }
183         // if(EUSART_is_rx_ready()) primiSliku();
184         // __delay_us(2);
185         if( slika [ i ][ j ].b6 == on){
186             LATD = temp + 1;
187         }
188         // if(EUSART_is_rx_ready()) primiSliku();
189         // __delay_us(2);
190         if( slika [ i ][ j ].b5 == on){
191             LATD = temp + 2;
192         }
193         // if(EUSART_is_rx_ready()) primiSliku();
194         // __delay_us(2);
195         if( slika [ i ][ j ].b4 == on){
196             LATD = temp + 3;
197         }
198         // if(EUSART_is_rx_ready()) primiSliku();
199         // __delay_us(2);
200         if( slika [ i ][ j ].b3 == on){
201             LATD = temp + 4;
202         }
203         // if(EUSART_is_rx_ready()) primiSliku();
204         // __delay_us(2);
205         if( slika [ i ][ j ].b2 == on){
206             LATD = temp + 5;
207         }
208         // if(EUSART_is_rx_ready()) primiSliku();
209         // __delay_us(2);
210         if( slika [ i ][ j ].b1 == on){
211             LATD = temp + 6;
212         }
213         // if(EUSART_is_rx_ready()) primiSliku();
214         // __delay_us(2);
215         if( slika [ i ][ j ].b0 == on){
216             LATD = temp + 7;
217         }
218         // if(EUSART_is_rx_ready()) primiSliku();
219         // __delay_us(2);
220     }
221     i++;
222     LATB = i ;
223     for( char j=4;;j--){
224         char temp = j*8;
225         if( slika [ i ][ j ].b0 == on){
226             LATD = temp + 7;
227         }
228         // if(EUSART_is_rx_ready()) primiSliku();

```

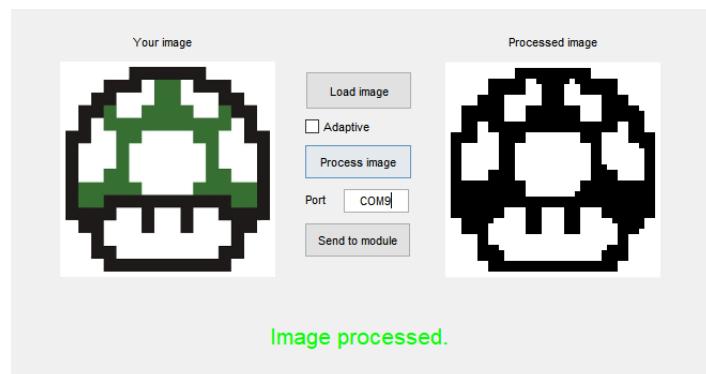
```

229         // --delay_us(2);
230         if(slika[i][j].b1 == on){
231             LATD = temp + 6;
232         }
233         // if(EUSART_is_rx_ready()) primiSliku();
234         // --delay_us(2);
235         if(slika[i][j].b2 == on){
236             LATD = temp + 5;
237         }
238         // if(EUSART_is_rx_ready()) primiSliku();
239         // --delay_us(2);
240         if(slika[i][j].b3 == on){
241             LATD = temp + 4;
242         }
243         // if(EUSART_is_rx_ready()) primiSliku();
244         // --delay_us(2);
245         if(slika[i][j].b4 == on){
246             LATD = temp + 3;
247         }
248         // if(EUSART_is_rx_ready()) primiSliku();
249         // --delay_us(2);
250         if(slika[i][j].b5 == on){
251             LATD = temp + 2;
252         }
253         // if(EUSART_is_rx_ready()) primiSliku();
254         // --delay_us(2);
255         if(slika[i][j].b6 == on){
256             LATD = temp + 1;
257         }
258         // if(EUSART_is_rx_ready()) primiSliku();
259         // --delay_us(2);
260         if(slika[i][j].b7 == on){
261             LATD = temp;
262         }
263         // if(EUSART_is_rx_ready()) primiSliku();
264         // --delay_us(2);
265         if(j==0)break;
266     }
267 }
268 }
269 }
```

3 EKSPERIMENTALNI REZULTATI

U ovom poglavlju će biti prikazani rezultati dobiveni sa prikazom određenih slika.
NAPOMENA: Sljedeće slike su uređene jer nije bilo moguće slikati čitavu sliku na osciloskopu iz tehničkih razloga.

Na slikama 10, 11, 12, 13, 14, 15, 16 i 17 su prikazani izgled prozora u Matlab-u i slike na osciloskopu.

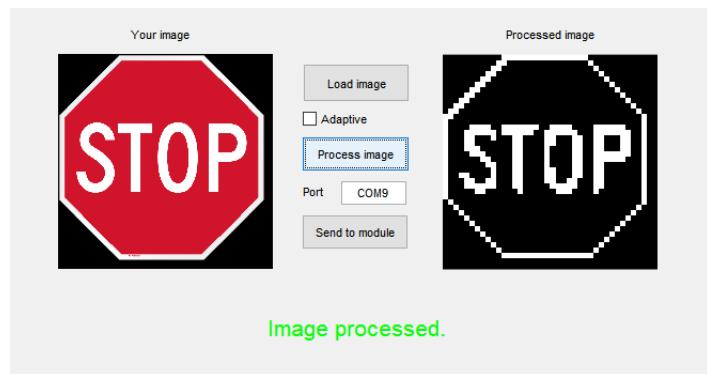


Slika 10: Izgled GUI-a Matlab



Slika 11: Prikaz slike na osciloskopu

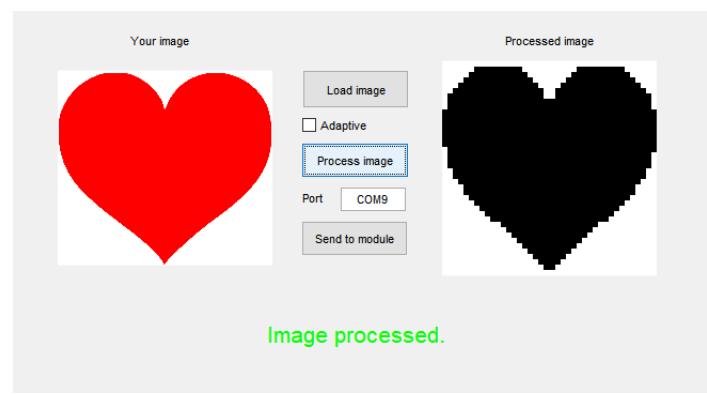
Na slikama 18 i 19 je prikazana razlika izmedju neadaptivnog i adaptivnog procesiranja slike.



Slika 12: Izgled GUI-a Matlab



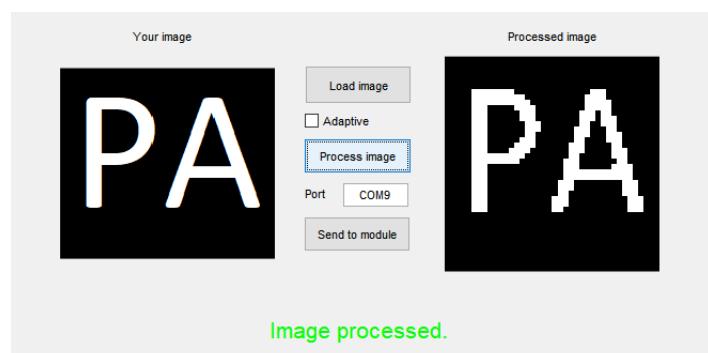
Slika 13: Prikaz slike na osciloskopu



Slika 14: Izgled GUI-a Matlab-a



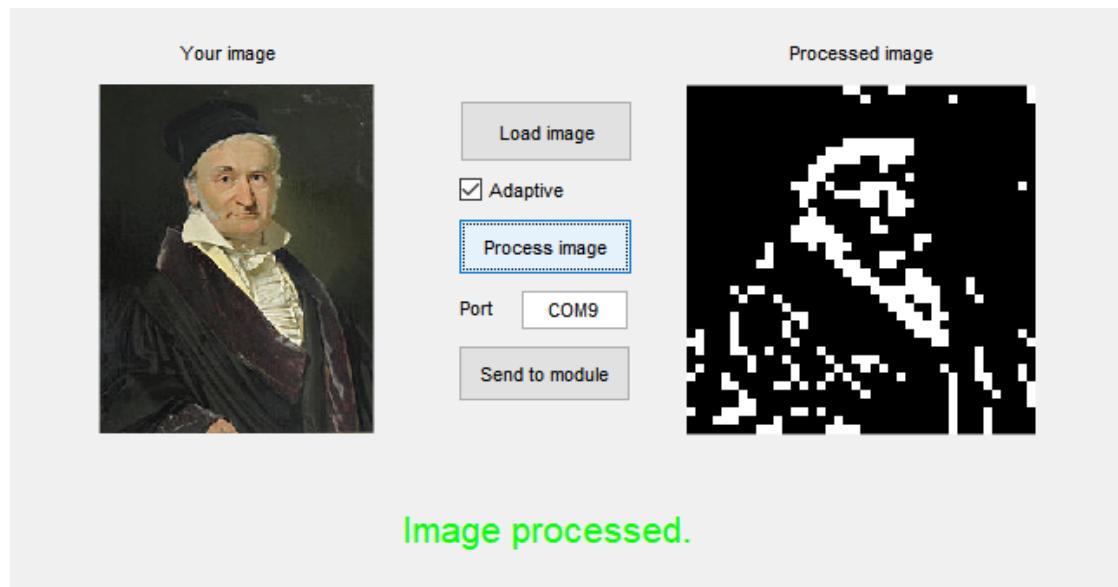
Slika 15: Prikaz slike na osciloskopu



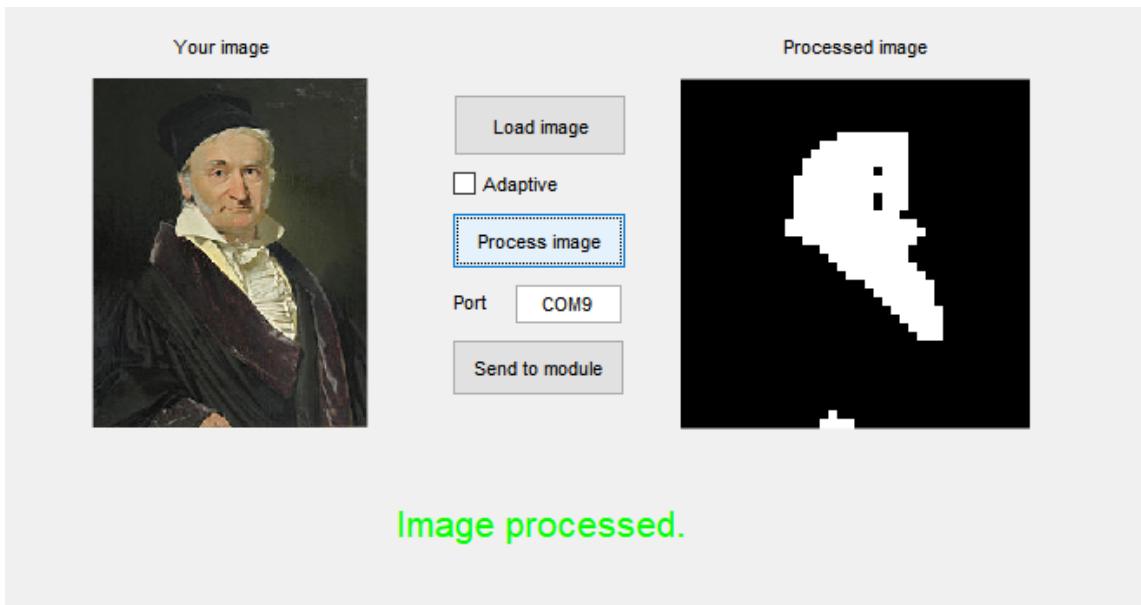
Slika 16: Izgled GUI-a Matlab-a



Slika 17: Prikaz slike na osciloskopu



Slika 18: Adaptivno procesiranje slike



Slika 19: Neadaptivno procesiranje slike

4 ZAKLJUČAK

Projekat smo realizirali na osnovu prethodno stečenog znanja iz predmeta Praktikum Automatike, ali i predmeta Ugradbeni sistemi. Također, bilo je potrebno dodatno upoznavanje sa funkcijama mikrokontrolera jer na laboratorijskim vježbama nije rađena serijska komunikacija. Glavni cilj bio je teoretsko shvatanje pozadine koja se dešava iza samog ekrana osciloskopa tj. kako doći do prikaza slike. Kako smo izabrali da za prikaz slike koristimo dva digitalno-analogna konvertora bilo je potrebno dodatno proučiti samo implementaciju istih. Najveći problem predstavlja slanje signala na osciloskop jer razvojni sistem nije dovoljno brz da bi uspio prikazati sve piksele a da to ljudsko oko posmatra kao jednu sliku.

5 LITERATURA

- Dokumentacija za mikrokontroler Microchip PIC16F1939
- Praktikum automatike i elektronike (Skripta - radna verzija) - M. Kuric, S. Konjicija A. Aksamovic
- Praktikum automatike – predavanje za ak.god. 2018/2019
Vanr. prof. dr Samim Konjicija, dipl. ing. el.
Vanr. prof. dr Abdulah Akšamović, dipl. ing. el.