Jedna realizacija plotera

Ognjen Bjelica
Laboratorija za embedded sisteme
Elektrotehnički fakultet u Istočnom Sarajevu
Bosna i Hercegovina
ognjen.bjelica@etf.unssa.rs.ba

Sadržaj—**U ovom radu predstavljena je jedna realizacija plotera. Mehanička konstrukcija je u potpunosti zadržana sa starog uređaja i iskorištena kao polazna osnova za ovu realizaciju. Originalna elektronika je u potpunosti zamijenjena novom sa USB HID komunikacijom. Takođe je razvijena i PC aplikacija za upravljanje radom plotera.**

Ključne riječi-ploter; rasterizacija; programiranje; mikrokontroler;

I. UVOD

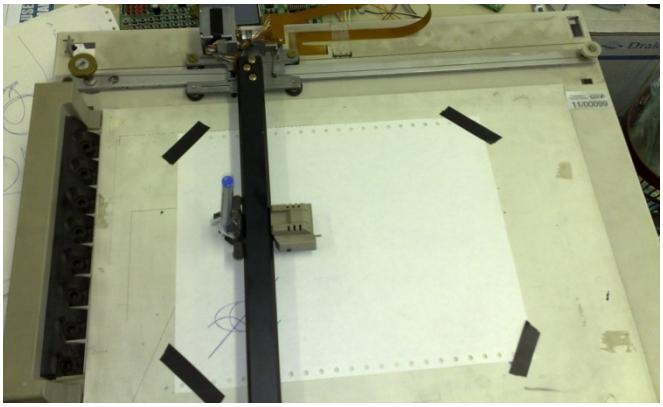
U ovom radu je prikazana jedna praktična realizacija plotera. Posebna pažnja će biti posvećena problemima koji se mogu javiti ili koji su se javljali pri realizaciji ovog projekta, kao i načinu na koji se oni mogu riješiti. Svaka realizacija kompletnog uređaja, pa tako i ovog je relativno kompleksan zadatak i zahtjeva poznavanje više oblasti. Zbog lakšeg razumijevanja i praćenja, ovaj problem se može podijeliti na sljedeće zadatke-dijelove:

1. mehanika
2. rad i upravljanje step (koračnih) motora
3. mikrokontroleri i programiranje mikrokontrolera
4. programiranje interfejsa računar – mikrokontroler
5. rasterizacija
6. programiranje Windows aplikacija

Mehanika, tj. sam mehanizam pokretanja olovke, naslijeđen je iz starog plotera tako da ovom dijelu problema neće biti posvećena dodatna pažnja u nastavku ovog rada. Ovaj mehanizam sastoji se od ploče za crtanje i dva step motora za pomjeranje po x i y osi na koje je pomoću sajli spojena osovina koja nosi olovku. Mehanizam koji je korišten može se vidjeti na Sl. 1.

Rad i upravljanje step motora, kao i način povezivanja računar (PC – Personal Computer) - mikrokontrolerom zbog ograničenog broja stranica će biti izostavljen iz ovog rada. Ovdje samo treba napomenuti da za upravljanje step motorom nisu korištena posebna kola. Upravljačke signale generiše mikrokontroler koji je preko Darlingtonovih spojeva povezana na motor. U ovom projektu korišten je USB - HID (eng. Universal Serial Bus Human Interface Device) interfejs – protokol za povezivanje izabranog mikrokontolera sa PC-jem. Više informacija o načinu na koji su riješeni ovi problemi moguće je pogledati u [1].

Osnovne informacije o mikrokontrolerima, kao i specifičnosti izabranog mikrokontrolera će detaljnije biti opisane u poglavlju Mikrokontroleri, dok će samo programiranje mikrokontolera, kao jedan od važnijih dijelova projekta, da se provlači kroz više poglavlja. Osnovni algoritmi i dijelovi koda će biti izloženi u poglavljima Rasterizacija i Praktična realizacija plotera.



Slika 1. Mehanika plotera koja je iskorištena kao polazna osnova.

Rasterizacija, odnosno proces prevođenja grafičkih primitiva (linije, kružnice, elipse, ...) iz kontinualnog oblika u diskretni, značajan je problem sam po sebi. Kroz istorijski razvoj računarske grafike razvijen je veći broj algoritama koji obavljaju ovaj posao. Više informacija o ovoj problematici biće iznijeto u poglavlju Rasterizacija.

Na kraju, bilo je potrebno razviti aplikaciju koja će upravljati radom plotera. Ovaj dio se odnosi na programiranje Windows aplikacije (u opštem slučaju aplikacija ne mora biti ograničena platformom). Više informacija o ovom dijelu projekta će biti iznijeto u poglavlju Praktična realizacija plotera.

Detaljiniji opis svih dijelova, kao i načina na koji su oni implementirani u ovoj realizaciji plotera, moguće je pronaći u [1].

II. RASTERIZACIJA

Rasterizacija je, kao što je to i u uvodnom izlaganju navedeno, proces prevođenja grafičkih primitiva iz kontinualnog u diskretni oblik. Postoji veliki broj algoritama koji rješavaju probleme rasterizacije pojedinih primitiva (linije, kružnice, elipse i sl.). Neki od njih su efikasniji od drugih, neki imaju kao rezultat "ljepšu sliku", ali u principu svi rade isti posao. [2], [3] Algoritmi koji su korišteni pri ovoj praktičnoj realizaciji plotera ne daju najbolje rezultate, niti su najefikasniji, ali su najlakši za razumjevanje i idealni su za shvatanje osnovnih principa.

A. *Rasterizacija prave*

1) *Osnovni algoritam*

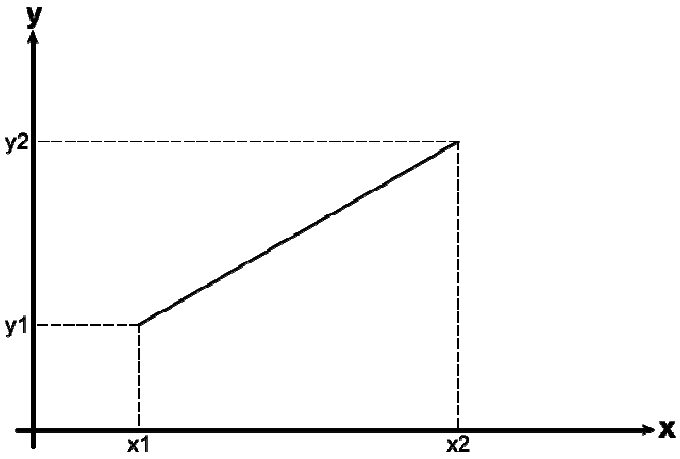
Parametarski oblik jednačine prave kroz dve tačke M1(x1,y1) i M2(x2,y2):

$$x = x_1 + k(x_2 - x_1) \quad (1)$$

$$y = y_1 + k(y_2 - y_1) \quad (2)$$

pri čemu je

$$0 \leq k \leq 1 \quad (3)$$



Slika 2. Prava kroz dvije tačke – pojašnjenje osnovnog algoritma.

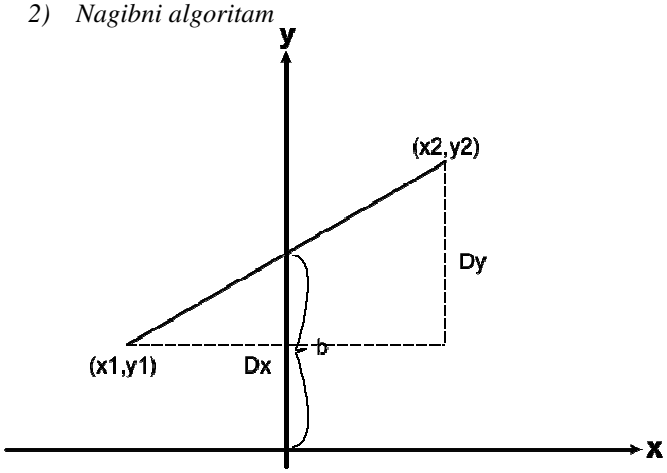
Prethodni izraz je jedan od osnovnih izraza u matematičkoj geometriji i u principu nije konplikovan. Osnovna ideja je da se mijenjanjem koeficijenta k od 0 do 1 sa nekim malim korakom (u principu što manji to je bolja rezolucija, s tim da se posle određene vrijednosti ne mogu postići poboljšanja) i uvrštavanjem u prethodne dvije jednačine mogu proračunati sve diskretne tačke x i y koje leže na toj liniji.

Kod koji ovo implementira dat je u nastavku:

```
void DrawLine(int x1, int y1, int x2, int y2) {
	float i = 0, korak = 0.001;
	postavi_olovku_na(x1, y1);
	while (i <= 1) {
		pom_x = x1 + i*(x2 - x1);
		pom_y = y1 + i*(y2 - y1);
		postavi_olovku_na(pom_x, pom_y);
		i = i + korak;
	}
}
```

Formule su preuzete direktno iz matematičih tablica i stoga ovaj algoritam nema najbolje performanse. Ovaj algoritam je korišten u praktičnoj realizaciji projekta zbog toga što je, kao što je to i u uvodnom dijelu ovog poglavlja rečeno, krajnje jednostavan za razumjeti. U nastavku ovog teksta će biti prikazani još neki algoritmi koji rješavaju isti problem samo poređenja radi i da se pokaže kako se može postići optimizacija korištenih algoritama.

2) *Nagibni algoritam*



Slika 3. Prava kroz dvije tačke – pojašnjenje nagibnog algoritma.

Linija se matematički može definisati i kao:

$$y = m \cdot x + b \quad (4)$$

pri čemu je:

$$m = \frac{D_y}{D_x} \quad (5)$$

$$b = y - m \cdot x \quad (6)$$

$$D_y = y_2 - y_1 \quad (7)$$

$$D_x = x_2 - x_1 \quad (8)$$

[2]

Kod koji ovo implementira dat je u nastavku:

```
void DrawLine(int x1, int y1, int x2, int y2) {
	int Dy = y2-y1, Dx = x2-x1, x, y, xend;
	float m = (float)Dy / Dx, b = y1 - m*x1;
	if (x1>x2) {
		x = x2;  y = y2;
		xend = x1;
	} else {
		x = x1;  y = y1;
		xend = x2;
	}
	while (x <= xend) {
		postavi_olovku_na(x, y);
		x++;
		y = m*x + b;
	}
}
```

Vidimo da je ovaj algoritam veoma sličan korištenom, s tim da se poboljšanje ogleda u tome da je eliminisana konstanta k koja je uticala na kvalitet linije. Mana ovog algoritma je ta što i dalje ostaju operacije sa realnim brojevima.

3) Inkrementalni algoritam

Prethodni algoritam se može poboljšati na sledeći način:

$$y = m \cdot x + b \qquad (9)$$

$$y_{i+1} = m \cdot x_{i+1} + b = m \cdot (x_i + \Delta x) + b =$$

$$= m \cdot x_i + b + \Delta x \cdot m = y_i + \Delta x \cdot m \qquad (10)$$

za

$$\Delta x = 1 \qquad (11)$$

slijedi da je

$$y_{i+1} = y_i + m \qquad (12)$$

[2]

Kod koji ovo implementira dat je u nastavku:

```
void DrawLine(int x1, int y1, int x2, int y2) {
	int Dy = y2-y1, Dx = x2-x1, x, y, xend;
	float m = (float)Dy / Dx;
	if (x1>x2) {
		x = x2;
		y = y2;
		xend = x1;
	} else {
		x = x1;
		y = y1;
		xend = x2;
	}
	while (x <= xend) {
		postavi_olovku_na(x, y);
		x++;
		y += m;
	}
}
```

Kao što je već rečeno, ovaj algoritam predstavlja poboljšanje prethodnog. Vidimo da je smanjen broj operacija sa realnim brojevima, ali one nisu u potpunosti eliminisane.

4) Bresenham-ov algoritam

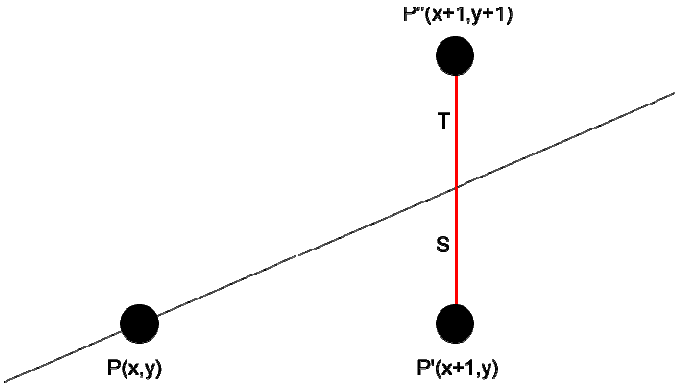
Ovaj algoritam je jedan od najboljih (ako ne i najbolji) i najefikasnijih algoritama rasterizacije linije.

$$d = S - T \qquad (13)$$

$$d < 0 \rightarrow tačka\ P' je\ bliža\ matematičkoj\ pravoj$$

$$d \geq 0 \rightarrow tačka\ P'' je\ bliža\ matematičkoj\ pravoj$$

[2]



Slika 4. Odlučivanje da li će linija proći kroz taču P' ili P''.

Kod koji ovo implementira dat je u nastavku:

```
void DrawLine(int x1, int y1, int x2, int y2) {
	int Dy = y2-y1, Dx = x2-x1, x, y, xend,
	d = 2*Dy-Dx, inc1 = 2*Dy,
	inc2 = 2*(Dy*Dx);
	if (x1>x2) {
		x = x2;
		y = y2;
		xend = x1;
	} else {
		x = x1;
		y = y1;
		xend = x2;
	}
	while (x <= xend) {
		postavi_olovku_na(x, y);
		if (d<0)
		{
			d += inc1;
		}
		else
		{
			d += inc2;
			y++;
		}
	}
}
```

Kao što se to iz samog algoritma može vidjeti, sve operacije se obavljaju nad cjelobrojnim brojevima, a sva množenja se vrše sa brojem dva, što se radi efikasnijeg izvršenja može zamjeniti operacijom šiftovanja za jedno mjesto ulijevo. Na primjer:

inc1=2*Dy

se može napisati kao:

inc1=Dy<<1

Prikaz implementacije ostalih algoritama zbog ograničenog broja stranica, ne može stati u ovaj rad.

III. PRAKTIČNA REALIZACIJA PLOTERA

Nakon sagledavanja svih važnijih problema i upoznavanja sa svim dijelovima plotera, može se pristupiti praktičnoj realizaciji plotera. Najvažniji dio elektronike koja je zadužena za upravljanje ploterom predstavlja mikrokontroler. Za ovu svrhu izabran je Microchip-ov PIC18F4550 mikrokontroler, prvenstveno zbog činjenice da posjeduje USB interfejs [4]. Ovim je izbjegnuta upotreba dodatnih kola za komunikaciju. Osnovni problem pri spajanju mikrokontrolera sa step motorima predstavlja razlika u zahtjevanim naponima napajanja. Naime, mikrokontroler zahtjeva napon napajanja od 5 V, dok korišteni step motori rade sa 12 V. Da bi se prevazišao ovaj problem iskorišteni su Darlington-ovi spojevi, tj. ULN2803 kola koja u sebi sadrže niz od 8 Darlington-ovih spojeva [5]. U principu, ova kola nisu idealan izbor za kontrolu step motora, jer postoji veliki broj drajvera za upravljanje step motorima. Razlog zašto su izabrana baš ova kola je taj što zahtjevaju „ručno“ kontrolisanje motora, što ih čini idealnim za shvatanje principa upravljanja step motora.

Napajanje samog mikrokontolera je uzeto sa USB porta, dok napajanje step motora zahtjeva transformator i 7812 kolo (stabilizator napona) [6].

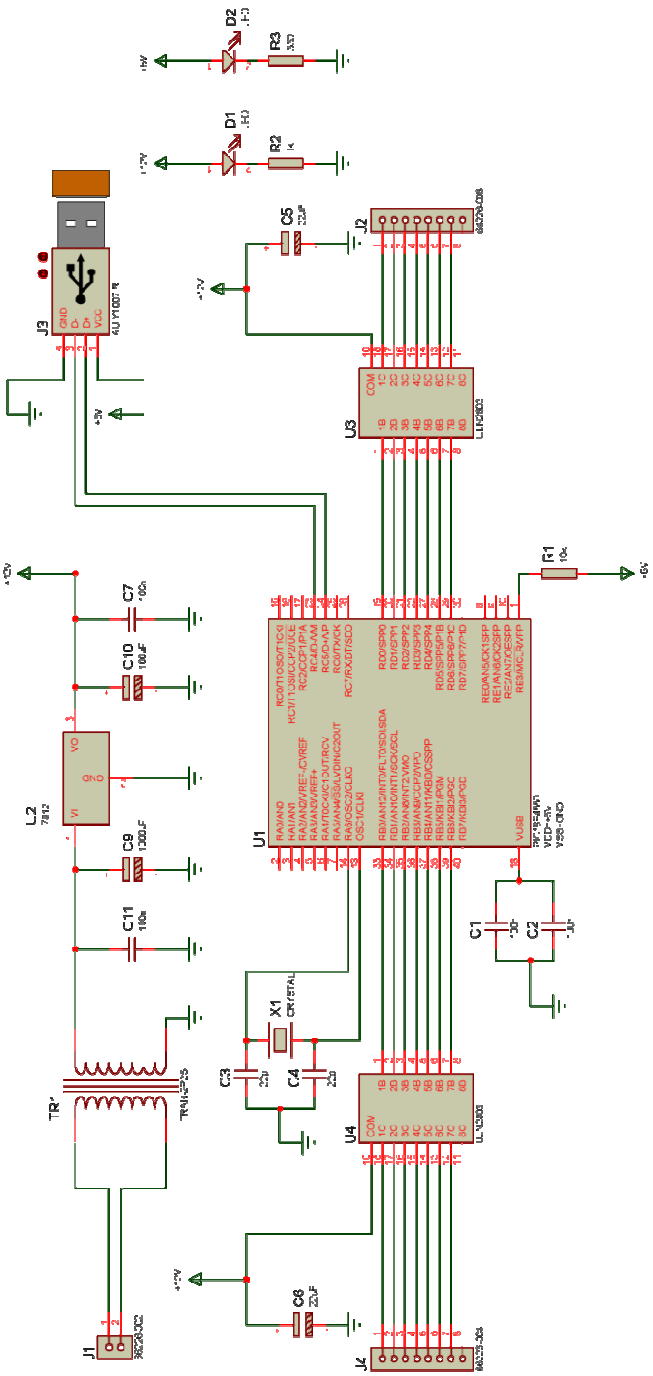
Na portove B i D mikrokontrolera spojena su dva ULN2803 kola (po jedan na port), a njihovi izlazi su izvedeni na konektore na koje će se povezati step motori. Glavni razlog zašto su iskorišteni baš ovi portovi je raspored njihovih pinova na mikrokontroleru, tj.rutiranje štampane pločice je jednostavnije. U opštem slučaju mogu se koristiti proizvoljni pinovi, ne moraju čak ni biti na istom portu.

Kompletna šema prikazana je na Sl. 5. Kao što se to može vidjeti, šema je poprilično jednostavna. Kao što je to u više navrata rečeno, osnovna ideja prilikom realizacije bila je napraviti sistem koji je što jednostavniji za razumjevanje i koji je potpuno funkcionalan. Kasnije je lako pojedine dijelove koda optimizovati ili koristiti druga kola kad je hardver u pitanju.

A. *Program mikrokontrolera (Firmware)*

Imajući na umu princip rada step motora, kao i način na koji je step motor povezan sa PIC18F4550 mikrokontolerom, program koji upravlja ploterom je relativno jednostavan. Najvažnije funkcije koje su zadužene za interakciju step motora i mikrokontolera su:

- void pomjeri_se_lijevo()
- void pomjeri_se_desno()
- void pomjeri_se_dole()
- void pomjeri_se_gore()
- void spusti_olovku()
- void podigni_olovku()
- void postavi_olovku_na(int xx, int yy)

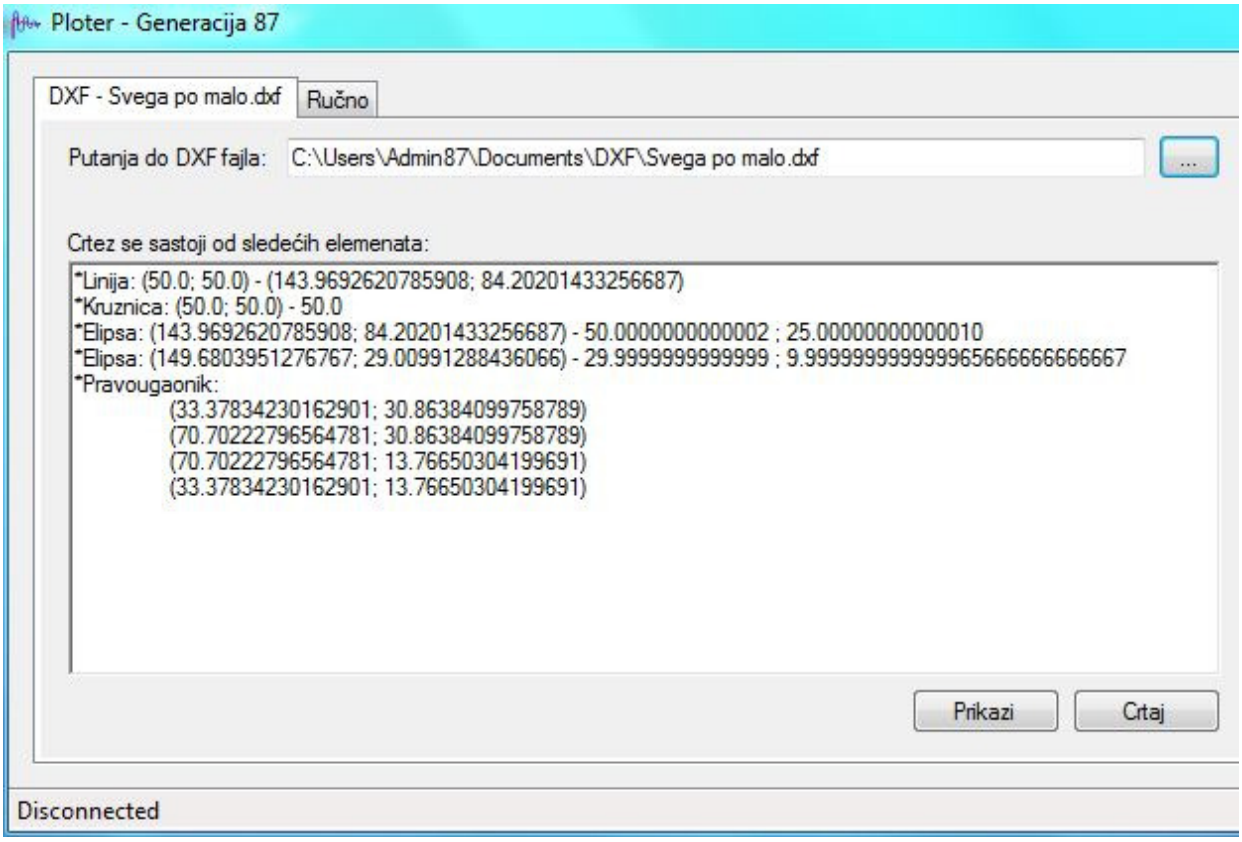


Slika 5. Šema plotera.

Funkcija *pomjeri_se_lijevo()* generiše sekvencu signala na odgovarajućim pinovima mikrokontrolera koja kao rezultat pokreće motor plotera (koji je zadužen za x-osu), a koji pomjera olovku u lijevu stranu za jedan korak. Prije nego što izvrši pomjeranje olovke, provjerava se da li je ona došla do kraja papira (tj. da li je x koordinata veća od 1). Ako je uslov ispunjen (olovka nije došla do kraja papira), tada se generiše ranije pomenuta sekvenca i ažurira se novo stanje koordinata (vrednost x koordinate se smanji za jedan).

Ostale funkcije implementirane su na sličan način.

B. Windows aplikacija koja upravlja radom plotera



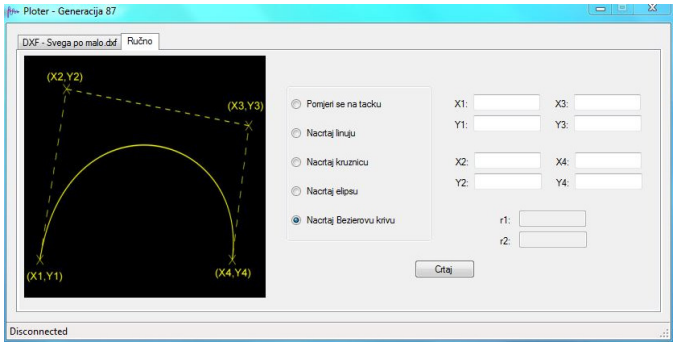
Slika 6. Razvijena aplikacija - rad sa DXF fajlovima.

S obzirom da PC raspolaže znatno većim resursima, on omogućava i realizaciju složenijih zadataka. Sam proces komunikacije sa PIC18F4550 mikrokontrolerom, zahvaljujući HID protokolu, dosta je jednostavan. Međutim, da bi program dobro izgledao, kao i da bi korisnika zaštitio od mogućih grešaka, realizovana aplikacija je kompleksnija nego što je to neophodno.

Jedan od većih problema pri pisanju programa za PC bio je problem čitanja i DXF-a (Drawing Exchange Format). Da bi svoj crtež, zajedno sa svim svojim podešavanjima (korištena metrika, uglovi,...) mogli rekonstruisati, AutoDesk-ovi stručnjaci razvili su pomenuti format [7]. Koliko je format komplikovan govori informacija da prazan crtež sadrži 11490 linija koda, a broj razlika između praznog crteža i crteža koji sadrži samo jednu liniju iznosi 138. Srećom, DXF koristi tekstualne fajlove, pa je bilo moguće analizirati njihov sadržaj.

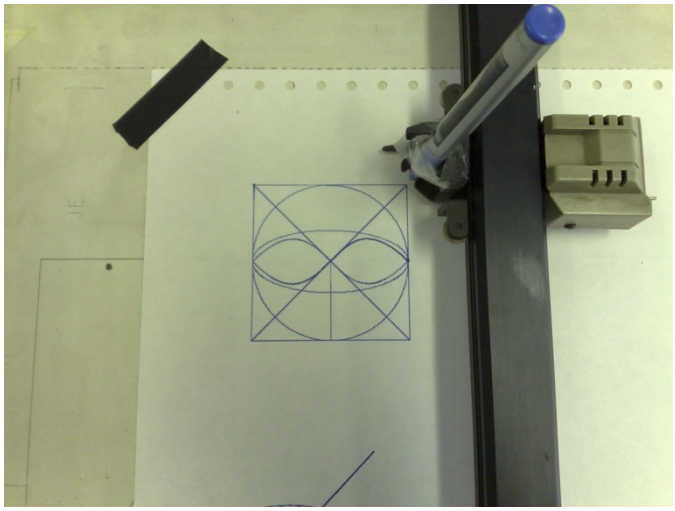
Korisnički interfejs razvijene aplikacija prikazan je na Sl 6., 7. i 8.

Slika 7. Razvijena aplikacija – Vizuelni prikaz sadržaja DXF fajla.



Slika 8. Razvijena aplikacija – „ručno“ zadavanje komandi ploteru.

Kao što se to sa prethodnih slika može vidjeti, korisnički interfejs je jednostavan, ali omogućava potpunu kontrolu nad ploterom. Samo crtanje je ostavljeno kompleksnim i popularnim programima, kao što su to AutoCAD, CorelDRAW i sl. Na Sl. 9 prikazano je nekoliko primitiva nactanih iz ručnog režima. Unutar ovog režima, kao što se to vidi na Sl. 8, moguće je izabrati tip primitive koja želi da se nacrta i zadati njene parametre, dok se sa lijeve strane prikazuje značenje pojedinih polja na formi. Tako npr. za Bezierovu krivu potrebno je popuniti polja X1, Y1, X2, Y2, X3, Y3i X4, Y4. Ostala polja (r1 i r2 u ovom slučaju) su onemogućena.



Slika 9. Primjer iscrtanih primitiva sa realizovanim hardverom i softverom.

IV. ZAKLJUČAK

U ovom radu izložena je jedna praktična realizacija plotera i, kao što je to ranije rečeno, cilj ovog projekta nije bio napraviti najoptimalnije moguće rešenje. Osnovni cilj bio je upoznati se sa osnovama svih korištenih tehnologija.

Mjesta optimizaciji uvjek ima i samo je pitanje koliko pažnje vrijedi posvetiti ovom problemu. Dvije optimizacije koje su dosta jednostavne i ne zahtjevaju nikakvu izmjenu hardvera su:

- Zamijeniti korištene algoritme rasterizacije efikasnijim algoritmima i na to je skrenuta pažnja ranije u radu (poglavlje Rasterizacija). Moglo bi se reći da je ova optimizacija poprilično očigledna.
- Druga stvar koja bi se mogla optimizovati jeste Windows aplikacija. Ovaj problem i nije toliko očigledan, ali sam redoslijed iscrtavanja grafičkih objekata je veoma važan za performanse sistema - plotera. Treba napisati algoritam za sortiranje grafičkih primitiva tako da se završetak jedne primitive nađe što bliže početku sledeće. Drugim riječima, treba izbjeći duga premještanja olovke između crtanja.

Moguće je čak, relativno jednostavno, poboljšati i rezoluciju plotera.:

- Bolja rezolucija može se postići ako se izmjene funkcije *pomjeri_se_lijevo(), pomjeri_se_desno(), pomjeri_se_dole()* i *pomjeri_se_gore().* Trenutno su funkcije realizovane tako da se pomjere za par koraka odjednom, radi jednostavnijeg izvođenja. Način na koji se ovo može izvesti je uvođenjem niza koji će pamtiti trenutni položaj i njegove susjedne položaje (stanja pinova – redoslijed paljenja namotaja step motora).
- Takođe, moguće je iskoristiti neko gotovo kolo koje omogućava upravljanje step motorom mikrokoracima. Ovaj način zahtjeva malu izmjenu hardvera i prethodno pomenutih funkcija. Sve ostalo ostaje isto. Na ovaj način rezolucija se može povećati i par stotina puta.

LITERATURA

[1] Ognjen Bjelica, "PRAKTIČNA REALIZACIJA PLOTERA," Završni rad, Elektrotehnički fakultet u Istočnom Sarajevu, 2010.
[2] Slobodanka Đorđević-Kajan, "Skripte sa predavanja iz predmeta Računarska grafika."
[3] James D. Foley, Andries Van Dam, Steven K. Feiner, John F. Hughes, *Computer Graphics: Principles and Practice in C*, (Second Edition). 1995.
[4] Microchip, "PIC18F2455/2550/4455/4550 Data Sheet."
[5] Texas Instruments, "ULN2803A Darlington Transistor Arrays."
[6] Texas Instruments, "LM78XX Series Voltage Regulators."
[7] AutoCAD, "DXF Reference." 2007.

ABSTRACT

This paper presents one realisation of plotter. All mechanical parts are salvaged from old device which provided starting point for this realization. Original electronics was completely replaced with new one with USB HID communication. For purpose of controlling plotter, PC application was developed.

ONE REALIZATION OF PLOTTER

Ognjen Bjelica