Digitalno njihalo

Mihačić, Luka

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni preddiplomski studij računarstva

DIGITALNO NJIHALO

Završni rad

Luka Mihačić

Osijek, 2016.

SADRŽAJ

1. UVOD

Tema ovog završnog rada je izrada desktop aplikacije digitalnog njihala za potrebe laboratorija iz kolegija Fizike 2, koja bi trebala omogućiti studentima lakši način razumijevanja i iščitavanja vrijednosti koje su potrebne za računanje parametara matematičkog njihala, budući da su njihala sastavni dio laboratorijskih vježbi. Obično eksperimenti su ograničeni na korištenje uređaja koji mjere titraje njihala, no ovaj način će biti nešto drugačiji.

Aplikacija se zasniva na temeljnim znanjima stečenim na kolegijima „Fizike 2“ i „Objektno orijentirano programiranje“. Za kreiranje aplikacije korišteno je razvojno okruženje Visual Studio te je bilo dovoljno koristiti objetno orjentirani programski jezik C#. Za potpunu realizaciju ove aplikacije potreban je bio mehanički miš s kuglicom, koji je prerađen u oblik potreban za realizaciju sveukupnog digitalnog njihala.

Prvi dio završnog rada bazira se na teorijskog podlozi, gdje su opisani kratki opisi korištenih tehnologija i uređaja korištenih pri izradi. Nakon toga slijedi kronološki opis nastanka aplikacije, od kreiranja glavnog prozora aplikacije, pa sve do iscrtavanja grafa pomoću miša koji će poslužiti kao njihalo, kao i izrade samo njihala. Sve je to vidljivo na slikama, dijelovima koda opisanih u glavnom dijelu te prilozima uz rad.

1.1. Zadatak završnog rada

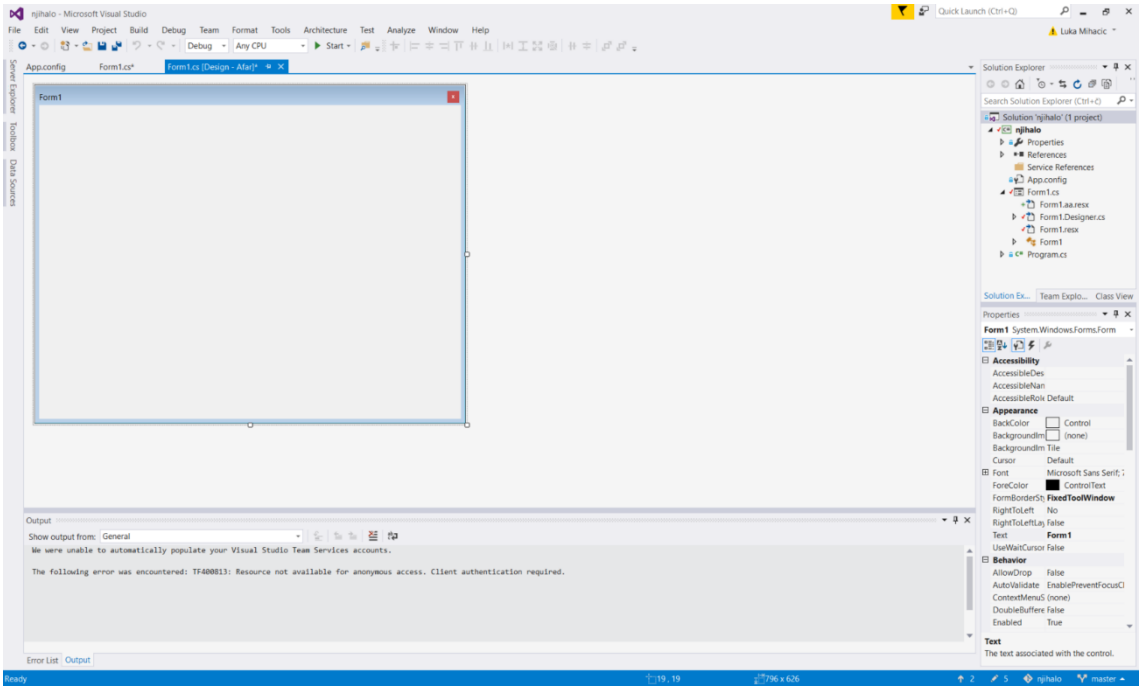
Zadatak završnog rada je izraditi desktop aplikaciju digitalnog njihala koja će iscrtavati graf matematičkog njihala preko enkodera mehaničkog miša s kuglicom koji je povezan sa računalom preko USB (Universal Serial Bus) priključka. Pri izradi je korišteno programsko okruženje Visual Studio, pisana je programskim jezikom C#, koje komunicira sa enkoderom miša preko USB priključka.

2. TEORIJSKE PODLOGE

2.1. Visual Studio

Visual Studio je integrirano razvojno okruženje *(IDE)* tvrtke Microsoft. Serija Microsoftovih alata za razvitak softvera za operacijske sustave Windows, iOS, Android. Za iOS i Android mobilne operacijske sustave je potrebno instalirati dodatne programe *(Xamarin)* u svrhu razvoja aplikacija. Radi se o korisničkim sučeljima povezanim s jezičnim procesorima koji podržavaju rad s raznim programskim jezicima. Ugrađeni programski jezici u Visual Studio su: C, C++, C# (preko Visual C#), VB.NET (preko Visual Basic .NET). Integrirani program za pronalaženje pogrešaka u kodu (eng. debugger) radi na razini izvornog i strojnog koda. Program sadrži alate poput dizajnera oblika koji se koristi za izradu aplikacija s grafičkim sučeljem, dizajna klasa, dizajna web stranica, dizajnera baze podataka itd.

Microsoft pruža (eng. Express) izdanja programa Visual Studio besplatno. Komercijalna izdanja programa Visual Studio kao i određena prethodna izdanja studentima su dostupni preko Microsoftovog DreamSpark programa. Pri izradi ove desktop aplikacije pri izradi projekta korišten je oblik „Windows Form Application“ koja ima već neke svoje određene gotove forme koje treba isprogramirati ne bi li im se pridodala željena funkcija koja je korisniku potrebna.



Sl. 2.1. *Izgled razvojnog okruženja Visual Studio [1]*

2.1.2 C#

C# (eng. C Sharp) je objektno orijentirani jezik kojeg su razvili Anders Hejlsberg i njegov tim pod vodstvom tvrtke Microsoft unutar .NET inicijative, a od 2000. je poznat pod imenom kakvim ga znamo danas, njegova najnovija verzija je 5.0 i izdana je 2012. Teži biti jednostavan, moderan, objektno orijentirani program opće namjene koji pruža mogućnost pisanja „host“ aplikacija i proširene sustave u velikom opsegu, od onih koji koriste sofisticirane operativne sustave, pa do onih manje zahtjevih zadataka. Jezik kao i njegove implementacije omogućava provjeru pisanja u skladu sa sintaksom na koju smo navikli i kod drugih programskih jezika (C, C++, Java), definiranje granica polja, automatsko upravljanje memorijom i brisanjem nepotrebnih stvari kada se ne koriste, provjera uporabe nepozvanih varijabli.

Softverska robusnost, trajnost i programerska produtivnost, portabilnost programera te portabilnost koda veoma su važni faktori pri uporabi ovog programskog jezika posebice za one koji su već od ranije upoznati sa sa C-om i C++-om programskim jezikom. [2]

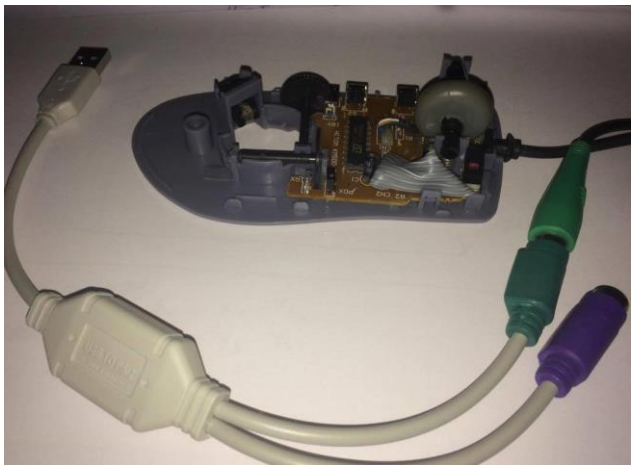
2.2. Prerada mehaničkog miša u njihalo

Za kompletnu izradu digitalnog njihala nužno je napraviti neke preinake na mehaničkom mišu s kuglicom. Prije programskog dijela na računalu i realizacije cijelog projekta potrebno je uspostaviti komunikaciju miša i računala preko USB priključka. Kako su mehanički miševi pretežno proizvedeni sa PS2 priključkom i rijetko koje računalo danas ima još tu vrstu priključka, pogotovo ako se govori o prijenosnim računalima. Nabavljen je PS2 ženski na USB muški priključak s kojim je moguće uspješno spojiti miš sa računalom. Sada kada je to napravljeno, sljedeći korak je doći do digitalnog kutnog enkodera koji je najvažniji dio miša i od njega ovisi cijeli princip njihala.

Mehanički miš s kuglicom ima dvije kordinate: x i y os te se ovisno o pomicanju cijelog miša pomiče kuglica koja vrti mehaničke dijelove koje odražavaju koordinatne osi i preko enkodera koji zatim to pretvara u digitalne impulse, vidljiv je pomak kursora miša na ekranu računala. Za ovaj projekt iskorišten će biti samo jedan valjak koji predstavlja os miša i to os apscisa.

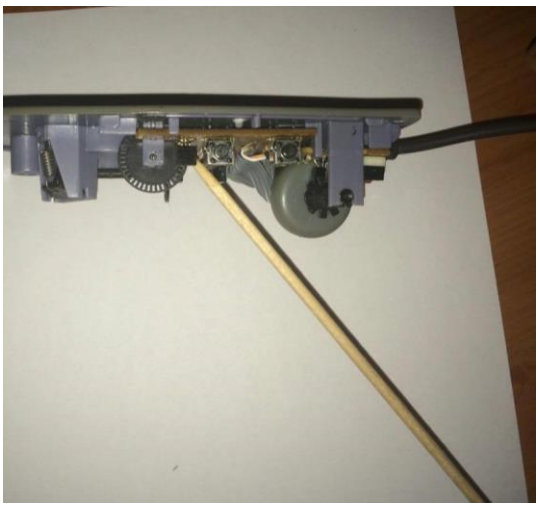
Sljedeća stvar koju je potrebno napraviti je skinuti jedan vijak sa miša kako bi se skinuo plastični okvir miša (Slika.2.2.) kojim se dolazi do glavnog dijela i vidljivi su dijelovi odgovorni

za osi i digitalni enkoder. Prethodno je izvađena kuglica tako da se skine sa donje strane jedan plastični dio laganim zavrtanjem te kuglica ispadne sama. Plastični oklop se izreže na način da se dio sa digitalnim enkoderom lijepo vidi te da postoji oslonac kada se kasnije bude postavio na stalak.



Sl. 2.2. *Prikaz miša bez plastičnog okvira i PS2 ženski na muški USB priključak*

Ideja je takva da se na valjak, koji kada se kuglica vrti, također vrti i određuje položaj na ekranu. Na valjak se pričvrsti drveni štapić koji bi kao takav tvorio njihalo. Drveni štapić se ljepilom pričvrsti za dio koji koji je najslobodniji, tako da nema prepreka i da se može otkloniti za kut od 180^{o}. Kada je to napravljeno cijeli takav prerađeni miš postavljen je na stalak na dovoljnu visinu da štapić ne zapinje od nikakvu podlogu ili predmete sa strane, to jest, da se neometano može gibati (Slika 2.3.).



Sl. 2.3. *Prikaz miša sa pričvršćenim drvenim štapićem i izgled njihala*

3. DIGITALNO NJIHALO

3.1. Ideja i osnovni koncept

Ideja za izradu aplikacije dolazi od činjenice da se eksperimentalna provjera znanja svodi na puko zapisivanje prije izmjerenih veličina ili mjerenje veličina od strane demonstratora ili laboranta što često zna biti jako nezanimljivo i studenti gube želju za stjecanjem praktičnih znanja na osnovi teorijskih, umjesto da im se pomoću praktičnih znanja pomnije približe fizikalne pojave u prirodi. Ovom aplikacijom će studenti moći izabrati parametre te promjenom istih uočavati razlike i povezanosti između veličina vezanih uz njihalo.

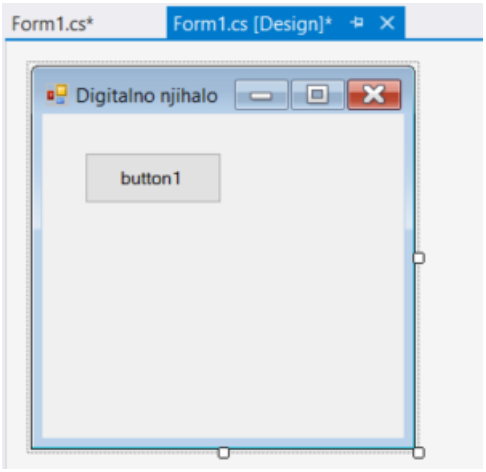
Kako je ranije navedeno, aplikacija je razvijena u razvojnom okruženju Visual Studio. Sučelje aplikacije je poprilično jednostavno kako bi njezina uporaba bila što praktičnija i lakša. Pri pokretanju aplikacije otvara se prozor u kojem je sve jednostavno i pregledno sa osnovnim stvarima za korisnika koji omogućuju odabiranje karakteristika na temelju kojih će digitalno njihalo iscrtavati graf. Odabiranje se vrši vrlo lagano, klikom miša na već predefinirane vrijednosti tako da je na korisniku samo da odabere parametre i pritiskom na gumb iscrtavanje može početi kada štapić otklonemo iz ravnotežnog položaja kako bi se njihalo počelo gibati, a na prozoru aplikacije počinje iscrtavanje grafa za vremenski interval koji smo prethodno odabrali.

3.2. Izrada aplikacije

3.2.1. Općenito o izradi

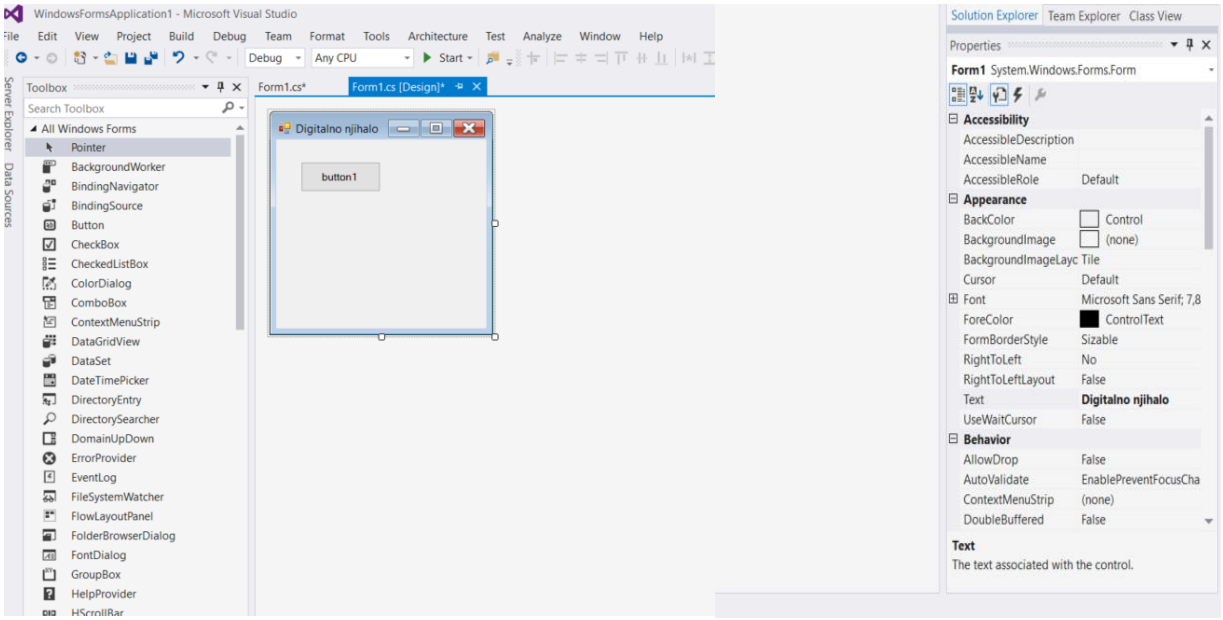
Za kreiranje aplikaciju u okruženju Visual Studio u glavnom prozoru koji je prikazan u prethodnim poglavljima otvara se kartica Novi Projekt te se odabire „Widnows Form Application“ u programskom jeziku C#. Nakon kreiranja projekta otvara se prozor sa prozorom pod imenom „Form 1“ i koja se preimenuje u ime završnog rada „Digitalno njihalo“.

Iznad prozora nalaze se dvije kartice koje predstavljaju dizajn aplikacije koji korisnik vidi i programski dio koda koji korisnik ne vidi i potrebno je pisati razne naredbe i funkcije kako bi na samom kraju sučelje aplikacije odgovaralo potrebama korisnika (Sl. 3.1).



Sl. 3.1. *Prikaz dizajna i programskog dijela forme*

Za bolji izgled i za pridodavanje većoj funkcionalnosti novokreiranoj formi se mogu mijenjati postavke, kao i svakom objektu koji će se u njoj nalaziti od mijenjanja fonta, boje, oblika, dimenzija te do pridruživanja određenih funkcija koje će ti objekti izvršavati sukladno sa zadaćama i zahtjevima korisnika. Mogućnosti je zaista mnogo, a korisit će se one koje budu primjerene ovom radu. Forma sama bez sebe je prazan prozor koji kada se otvori ništa ne radi, ali pri kreiranju ove vrste projekta na raspolaganje je dana široka paleta gotovih oblika iz operacijskog sustava Windows koji su mnogima jako dobro poznati i koje možemo iskoristi i međusobno povezivati u jednu kompaktnu cijelinu. Takvi oblici mogu se pronaći u lijevom dijelu prozora Visual Studija pod nazivom „Toolbox“ koji je prikazan na Slici 3.2., zajedno sa prikazom prozora svojstva forme, koja unutar svojstava ima još nekoliko podkartica koje se odnose na dodatne funkcionalnosti i ostalih budućih objekata, a nalazi se u donjem desnom kutu prozora programa.



Sl. 3.2. *Prikaz kartica Toolbox i Svojstava*

Pokretanje aplikacije i programa za pronalaske pogrešaka u kodu pokretat će pomoću gumba „Start“ u alatnoj traci ili pomoću kombinacije tipaka Ctrl+F5. Ovisno o uspješnosti i

točnosti napisanog koda prikazane će biti pogreške koje će nas uputiti na pogreške kao i crveno podcrtane naredbe tijekom pisanja koda se nešto ne podudara sa pravilima programskog jezika. Navedene su neke osnove koje su potrebne pri izradi aplikaciji u daljnjim poglavljima prijeći će se na samu ideju aplikacije i prikazan će biti tijek izrade kao i konačni rezultat.

3.2.2. Izgled sučelja aplikacije

Sučelje aplikacije predstavlja oblik kojeg korisnik dobije pri pokretanju, na njemu se trebaju nalaziti osnovne funkcije koje su korisniku dane na raspolaganje i ovisno o tome odabiru djelomično se mijenjaju i mogućnosti i karakteristike same aplikacije.

U prozoru ove aplikacije nalazi se pet vidljivih dijelova kojima su pridodane neke karakteristike i mogu se odabirati i mijenjati prilikom pokretanja i prije negoli se graf počne automatski iscrtavati pritskom na gumb. Postoje ukupno četiri gumba koja kontroliraju akcije u aplikaciji. Prvi gumb je ujedno i gumb za pokretanje i zaustavljanje aplikacije kada je to potrebno ili je želja korisnika. Gumb koji resetira, odnosno vraća uvijek aplikaciju u prvotno stanje, što znači da briše do sada iscrtane grafove i zaustavlja daljnje iscrtavanje. Snimanje zaslona nacrtanog grafa i automatski otvaranje snimljene slike ima gumb naziva „Spremi graf“ obliku .jpg. Spremanje se odvija u mapi projekta „digitalno njihalo“. Posljednji je za zatvaranje aplikacije, koje je još moguće pomoću tipke Esc (eng. „Escape“), što znači izlaz.Izgled gumbova je dan na Slici 3.3.

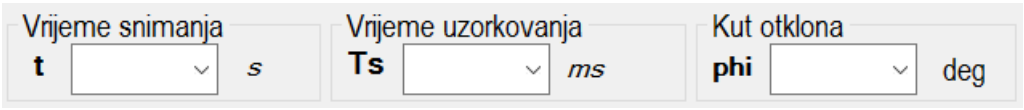
Sl. 3.3. *Izgled gumbova dostupnih u aplikaciji*

Dva prikazana gumba nađeni su u Toolboxu pod imenom gumb (eng. „button“) te su se zatim prevukli samo na sučelje aplikacije, a u svojsvima su promijenjena grafička svojstva. Dvoklikom miša na gumb ulazi se u programski dio aplikacije u kojem se pridjeljuju funkcije određenom objektu o kojem će biti riječi kasnije te isti postupak ponovili i za drugi gumb koji se odnosi na spremanje slike.

Kako se već govorilo u uvodnim poglavljima korisniku će se omogućiti odabiranje nekih parametara o kojima ovisi graf koji će se iscrtavati na temelju digitalne pretvorbe mehaničkog valjka miša u digitalne impulse koje će se odražavati na prozoru aplikacije. Radi se o tri parametra.

Vrijeme snimanja odnosi se na vremenski interval u kojem će računalo uzimati podatke od digitalnog enkodera, zatim vrijeme uzorkovanja koje predstavlja vrijeme između svakog pojedinog uzorka *n,* dok se ukupno vrijeme snimanja upravo računa kao umnožak vremena uzorkovanja i količine ukupnih uzoraka N, $t = NT_s$. Treći parametar je kut otklona njihala u stupnjevima.

Pri izradi ova tri parametra u sučelju prozora koristila su se tri različita gotova oblika iz „Toolboxa“. Dodavanje na sučelje izvodi se isto kao i kod dodavanja gubmova tzv. „Drag and drop“ metodom, kod koje se forma jednostavno premjesti na željeno mjesto. Gotova forma „Combobox“ služi kako bi se dodale već predefinirane vrijednosti tako da je na korisniku samo da odabere neku od vrijednosti. Forma „Label“ koristila se kako bi se dodala tekstualna oznaka parametra kao i njegova mjerna jedinica, a te forme posebno za svaki parametar je objedinio oblik „GroupBox“ iznad kojeg piše puni naziv parametara koje je moguće odabrati. Krajnji rezultat je vidljiv na Slici 3.5.



Sl. 3.4. *Izgled Groupboxeva za parametre njihala*

3.2.3. Pridjeljivanje funkcija pojedinom objektu

Sada je potrebno pojedinom objektu u programskom dijelu pridjeliti pojedine funkcije. Iz kartice „Toolbox“ gumbu za početak pridružujemo funkciju „*START_STOP()*“. Funkcija za pokretanje koja postavlja kursor miša u ishodište koordinatnog sustava, početnu vrijednost odakle počinje iscrtavanje, ako nije zadan parametar kuta (Slika 3.5). Navedena funkcija se sastoji od dvije posebne funkcije za pokretanje i zaustavljanje crtanja.„Timer“ koji je potreban kako bi se točno mjerilo vrijeme snimanja uzoraka preko enkodera miša. U oprecijskom sustavu Windows vrijeme nije realno, to jest Visual Studio ne uzima realno vrijeme, potrebno ga je kao takvog podesiti *(Real Time)*. Zaustavljanje će se dogoditi kada zadani uvjet u „*if“* petlji više nije ispunjen odnosno kada je broj trenutnih uzoraka *n* veći ili jednak ukupnom broju uzoraka *N (*Slika 3.6.) i poziva se funckcija za zaustavljanje crtanja koje se odvija u „Timeru“. U suprotnom ako taj uvjet ispunjen nije, snimljeni uzorci primljeni od uzorka miša i dalje će iscrtavati graf, a to znači da vrijeme odabrano u „Comboxu“ još nije isteklo. [3]

```
private void START()
{
    start_stop = false;
    button1.Text = "Stop";
    InitParameters();
    Rectangle screenRectangle = RectangleToScreen(this.ClientRectangle);
    this.Cursor = new Cursor(Cursor.Current.Handle);
    Cursor.Position = new Point(this.Location.X + x1,
                                this.Location.Y + (screenRectangle.Top-this.Top) + yOffset);
    timer1.Enabled = true;
    timer1.Start();
}
3 references
private void STOP()
{
    start_stop = true;
    button1.Text = "Start";
    timer1.Stop();
    timer1.Enabled = false;
```

Sl. 3.5. *Prikaz funkcija START() i STOP()*

```
private void timer1_Tick(object sender, EventArgs e)
{
    if (n < (N - 1)){
        n++;
        SnimljenaTocka[n].x = Convert.ToInt32(Convert.ToDouble(x1) + Convert.ToDouble(n) * Convert.ToDouble(ax));
        SnimljenaTocka[n].y = Convert.ToInt32(Convert.ToDouble(TrenutnaTocka.y - yOffset) * POJACANJE) + yOffset;
        CRTAJ_UZORAK();
    }
    else{
        STOP();
    }
```

Sl. 3.6. *Prikaz naredbi za iscrtavanje grafa u „Timeru“*

Nakon što se vrijeme zaustavilo u bilo kojem trenutku klikom miša ili pomoću tipkovnice te je potrebno daljnje vrijeme za proučavanje grafa, moguća je opcija spremanja grafa kao slike na način da ćemo uzeti dimenzije zaslona računala te prikazanu sliku u formatu *.jpg* spremiti u mapu gdje je spremljen cjelokupni projekt (Slika 3.7.). Dodatna funcija je „Reset“ koja vraća sve u početno stanje koja se poziva kako se sve održalo pregledno i uredno vidljivo na Slici 3.8. [4]

```
private void button3_Click(object sender, EventArgs e)
{
    Rectangle screenRectangle = RectangleToScreen(this.ClientRectangle);
    int bmpX = this.Location.X + this.Left;
    int bmpY = this.Location.Y + screenRectangle.Top - this.Top + y1;

    Bitmap bitmap = new Bitmap(Convert.ToInt32(Convert.ToDouble(W)  * 1.25d),
                               Convert.ToInt32(Convert.ToDouble(dy) * 1.35d));

    Graphics graphics = Graphics.FromImage(bitmap as Image);
    graphics.CopyFromScreen(bmpX, bmpY, 0, 0, bitmap.Size);
    bitmap.Save("img.jpg");
    this.ActiveControl = null;

    System.Diagnostics.Process.Start(@"img.jpg");
}
```

Sl. 3.7. *Prikaz funkcije gumba Save graph*

```
private void CLEAR()
{
    STOP();
    InitParameters();
    System.Drawing.Graphics g = this.CreateGraphics();
    g.Clear(SystemColors.Control);
    Refresh();
}
```

Sl. 3.8. *Prikaz funcije CLEAR()*

U „Comboxeve“ dodaju se vrijednosti parametara ručno na samom sučelju. Jedna stvar koja se mora napraviti je pretvorba tipova podatka što je česta pojava u programskom jeziku C#. Po zadanoj vrijednosti svi „Comboxevi“ su tipa (eng. „string“), odnosno znakovnog tipa, a kako se u njih unose brojevi, potrebno je prebaciti tip podatka koji prima iz znakovnog u brojevni tip podatka tipa (eng. „double“), tip višestruke preciznosti, koji omogućava veću preciznost jer omogućava unos brojeva iza decimalne točke. U ovom slučaju koristili smo također „if“ petlju u kojoj je uvijet ako je vrijednost u „Comboboxu“ različita od nule tada se provjerava (eng. „parse“) odabranu vrijednost tako da će napisana funkcija koja prebacuje vrijednost znakovnog zapisa u „double“ vrijednost, zatim tu vrijednost pohranjuje u jednu od tri varijable pomoću funkcije „*To.String()“*. Postupak se ponavlja za svaku varijablu koja predstavlja pojedini parametar (Slika 3.9.). Postavljanje vrijednosti za pojedini parametar se odvija pri pokretanju aplikacije kada se odabere određeni indeks parametra za početnu vrijednost. [5]

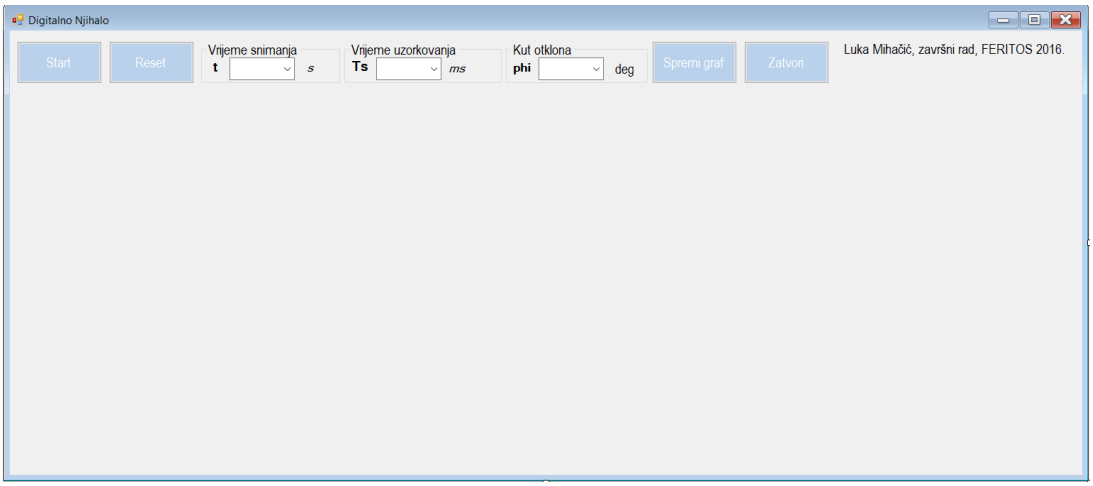
```
tMax = 10 * 1000; // Vrijeme snimanja u s
if (comboBox1.SelectedItem != null){
    tMax = Convert.ToInt32(comboBox1.SelectedItem) * 1000;
}

Ts = 10; // Vrijeme uzorkovanja u ms
if (comboBox2.SelectedItem != null){
    Ts = Convert.ToInt32(comboBox2.SelectedItem);
}

phiMax = 0; // Kut otklona u deg
yOffset = y0;
if (comboBox3.SelectedItem != null){ // unos parametara iz comboboxa
    phiMax = Convert.ToInt32(comboBox3.SelectedItem);
}
```

Sl. 3.9. *.Prikaz pretvorbe tipova podataka u Comboboxu*

Sada kada su svim formama na sučelju pridružene njihove fukncije, grafički izgled u dizajnu forme je prikazan na Slici 3.10. U preostali prostor prozora aplikacije smjestit će se graf čije funkcije će biti opisane kao i ideja cijelog grafa u sljedećem poglavlju.



Sl. 3.10. *Prikaz konačnog sučelja aplikacije digitalnog njihala*

3.2.4. Ideja i realizacija iscrtavanja grafa

Kako je prikazano Slikom 3.10. gumbi i forme za odabiranje parametara njihala smješteni su u gornjem dijelu sučelja aplikacije, zbog oslobađanja donjeg dijela za graf digitalnog njihala. U tome praznome dijelu treba biti smješten graf koji se iscrtava otklon štapića iz ravnotežnog položaja za određenu vrijednost kuta. Postavlja se pitanje kako to napraviti. Prije odgovora na ovo pitanje prvo treba osigurati parametre grafa da svi budu u pravom tipu i da budu što precizniji.

Maksimalni broj uzoraka koji se bio spomenut ranije, izračunava se kvocijentom vremena trajanja i vremena uzorkovanja, koji su također opisani ranije. Javlja se potreba za pretvaranjem tipova podataka, budući da je maksimalni broj uzoraka *N* deklariran kao tip podatka (eng. „integer“), odnosno cjelobrojni tip podataka. Koristit će se matematička funkcija (eng. „ceil“, a ono što ona radi je zaokruživanje vrijednosti nakon dijeljenja na prvi veći cijeli broj, neovisno bio taj broj veći od 0.5 ili pak manji, uvijek zaokružuje na veći (Slika 3.11.).

```
N = (Convert.ToInt32(Math.Ceiling(tMax / Ts * 1.0))); // max broj uzoraka
```

Sl. 3.11. *Matematička funcija ceil*

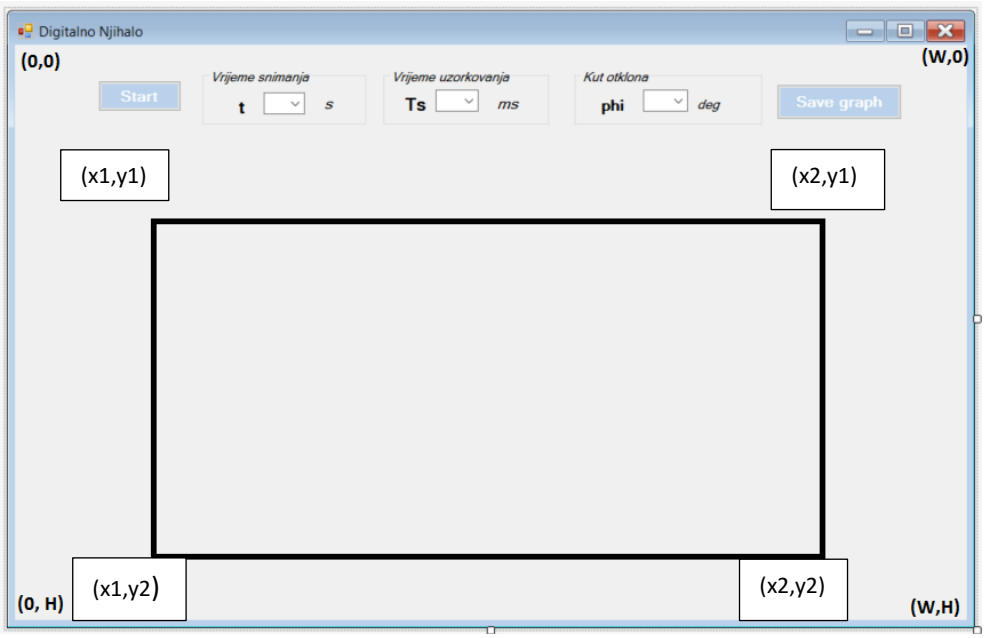
Kada je poznat broj ukupnih uzoraka moguće je alocirati, unaprijed odrediti određeni broj podataka u memoriji koji će se koristiti na temelju ukupnog broja uzoraka N. Definira se polje „SnimljenaTocka“ koja opisuje snimljene točke u trenutcima *n*, a veličina će biti jednaka ranije izračunatom *N* (Slika 3.12.). Takvo polje već je ranije spomenuto kod „Timera“ i gumba za početak crtanja grafa što je rezultiralo iscrtavanjem trenutnih položaja enkodera miša.

```
Tocka[] SnimljenaTocka; // snimljene tocke u trenucima n
SnimljenaTocka = new Tocka[N]; // uzorci
```

Sl. 3.12. *Definiranje polja*

Nazad na postavljeno pitanje, kako odrediti prostor u kojem će se iscrtavati graf će sada biti razmotreno. Budući da kreirane forme zauzimaju dio zaslona aplikacije, crtanje grafa preko cijelog ekrana nije moguće jer bi se podudarao sa formama. Unutar prozora aplikacije definiran će biti manji prozor unutar kojeg će se vršiti iscrtavanje grafa njihala. Prvi korak pri kreiranju manjeg prozora koji se neće ni po čemu razlikovati od ostatka sučelja, neće biti nekih vidljivih granica osim onih da se izvan granica tog prozora točke grafa neće pojavljivat.

Za bolje razumijevanje vrijednosti i koordinata aplikacije prozora dana je Slika 3.13. koja je zapravo temelj stvaranja prozora za iscrtavanje grafa digitalnog njihala.



Sl. 3.13. *Prikaz položaja koordinata i prozora za iscrtavanje grafa*

Slično kao kod snimanja zaslona, u dvije varijable širinu i visinu spremit će se vrijednosti dimenzija prozora aplikacije te će se dalje računati s tim vrijednostima. Takve dimenzije potrebno je smanjiti, a način obavljanja tog zadatka je najprije deklariranje konstantne varijable *p (postotak)* koja označava odstupanje i mjerilo koliko će virutalni prozor aplikacije odstupati od pravih dimenzija, uzet je postotak od 10%.

Određivanjem vrijednosti četiri koordinate koje su vidljive na Slici 3.13. stvoren će biti uvijet za izračunavanje konačnih dimenzija virtualnog prozora. Cijeli postupak prikazan je Slikom 3.14. Zbog boljeg prikaza i jednostavnosti sa cjelobrojnim vrijednostima izvršit će se pretvorba iz „double“ tipa podatka u „int“.

```
W = this.Width;
H = this.Height;
y0 = Convert.ToInt32(this.Height / 2);
x1 = Convert.ToInt32(p * Convert.ToDouble(W));
y1 = Convert.ToInt32(p * Convert.ToDouble(H));
x2 = Convert.ToInt32((1.0d - p) * Convert.ToDouble(W));
y2 = Convert.ToInt32((1.0d - p) * Convert.ToDouble(H));
dx = x2 - x1;
dy = y2 - y1;

  const double p = 0.10, p5 = 0.05; // 10% 5% border
```

Sl. 3.14. *Određivanje dimenzija prozora za iscrtavanje grafa digitalnog njihala*

Poznate su dimenzije prozora točka dx i točka dy. Jednadžba za položaj y osi drugačije je zapisana u programskom dijelu kao polje „TrenutnaTocka“ jer je kut otklona promjeljiva veličina i ona se mijenja za određeno odstupanje od trenutnog položaja y osi (eng. „Offset“).

Sada kada je uspostavljena potpuna komunikacija i aplikacija potpuno razumije gdje se nalazi kursor miša preko enkodera i valjka koji se mehanički vrti, moguće je te položaje tako da oni za sobom ostavljaju određeni trag, to jest linije i na takav način tvoriti graf. Iscrtavanje grafa provest će se pomoću grafičkih funkcija programskog jezika, konkretno funcije *„CRTAJ_UZORAK()“*. Potrebno je stvoriti jedan grafički objekt koji će omogućiti crtanje po njemu. Definiranje olovke za crtanje gdje se definira boja, širina olovke, oblik će odrediti izgled i linije grafa. Iscrtavanje grafa se odvija u odnosu na prethodnu točku što znači da se koristi „*if*“ u kojoj se stavlja uvjet za trenutni uzorak *n* da je veći od nule. Funkcija *DrawLine()* će spajati prethodnu točku sa trenutnom sve dok se ne dode do konačnog broja uzoraka *N* ili zaustavljanja iscrtavanja. (Slika 3.15.). [6]

```
private void CRTAJ_UZORAK()
{
    System.Drawing.Graphics g = this.CreateGraphics();
    System.Drawing.Pen pen = new System.Drawing.Pen(PenColor, PenWidth);

    if (n > 0)
    {
        g.DrawLine(pen, SnimljenaTocka[n - 1].x, SnimljenaTocka[n - 1].y , SnimljenaTocka[n].x, SnimljenaTocka[n].y);
    }
}
```

Sl. 3.15. *Iscrtavanje grafa u odnosu na položaj miša*

Za lakše snalaželje dodana je u sučelje aplikacije jedna labela koja prikazuje koordinate miša u stvarnom vremenu preko polja „TrenutnaTocka“ koja olakšava snalaželje u koordinatnom sustavu, odnosno od pomoći je bila pri kreiranju aplikacije. (Slika 3.14.).

```
1 reference
private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    TrenutnaTocka.x = e.X;
    TrenutnaTocka.y = e.Y;
    label6.Text = "x= " + TrenutnaTocka.x.ToString() + "\n" + "y= " + TrenutnaTocka.y.ToString();
}
```

Sl. 3.14. *Prikaz koordinata x i y u stvarnom vremenu*

3.2.5. Relativne vrijednosti i konačni izgled grafa

Kako bi graf dobio svoj konačni izgled potrebno je povući linije koje predstavljaju koordinatne osi, os apscisa i os ordinata te na njima točke koje dijele određeni interval kuta i vremena trajanja snimanja položaja miša preko enkodera. Labela koja snima i prikazuje na sučelju koordinate pomaže u pozicioniranju linija koje treba iscrtati. Zbog bolje orijentacije dodaju se linije, stvorit će se podioci pomoću kojih će korisnik lakše snaći u očitavanju vrijednosti, što kuta, što vremena.

Vrlo je važan element da se aplikacija može prilagoditi bilokakvom obliku glavnog prozora aplikacije, prema volji korisnika. To će biti moguće korištenjem relativnim vrijednosti prozora aplikacija, tako da će pri svakoj promjeni veličine prozora funkcija povući dimenzije trenutnne veličine prozora i prema njima će prilagoditi iscrtavanje grafa, kao i koordinatnih osi i osi podioka pri kojem ćemo kao relativne veličine koristiti već ranije definirane veličine dimenzija virtualnog prozora, takav postupak prikazan je na Slici 3.15.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    os = this.CreateGraphics();
    Pen myPen = new Pen(System.Drawing.Color.Black, 1);// pen za koordinatne osi
    Pen myPen1 = new Pen(System.Drawing.Color.LightSkyBlue, 1); // pen za orijentacijske linije

    // bounding box
    os.DrawLine(myPen1, x1, y1, x2, y1);
    os.DrawLine(myPen1, x1, y1, x1, y2);
    os.DrawLine(myPen1, x1, y2, x2, y2);
    os.DrawLine(myPen1, x2, y1, x2, y2);

    // glavne osi
    os.DrawLine(myPen, x1, y0, x2, y0); // x
    os.DrawLine(myPen, x1, y1, x1, y2); // y

    //linije vrijednosti kuta
     os.DrawLine(myPen1, x1, y45, x2, y45);
     os.DrawLine(myPen1, x1, y_45, x2, y_45);

    //linije vremenskog intervala
    os.DrawLine(myPen1, Convert.ToInt32(this.Width * 0.2), y1, Convert.ToInt32(this.Width * 0.2), y2);
    os.DrawLine(myPen1, Convert.ToInt32(this.Width * 0.3), y1, Convert.ToInt32(this.Width * 0.3), y2);
    os.DrawLine(myPen1, Convert.ToInt32(this.Width * 0.4), y1, Convert.ToInt32(this.Width * 0.4), y2);
    os.DrawLine(myPen1, Convert.ToInt32(this.Width * 0.5), y1, Convert.ToInt32(this.Width * 0.5), y2);
    os.DrawLine(myPen1, Convert.ToInt32(this.Width * 0.6), y1, Convert.ToInt32(this.Width * 0.6), y2);
    os.DrawLine(myPen1, Convert.ToInt32(this.Width * 0.7), y1, Convert.ToInt32(this.Width * 0.7), y2);
```

Sl. 3.15. *Crtanje x i y koordinata i mreže za bolju orijentaciju*

Prilikom mijenjanja veličine prozora događa se par stvari. Ako je iscrtani trenutni graf i korisnik ga odluči promijeniti taj iscrtani graf ne smije ostati jer bi se tada preklapali i izgledalo bi nezgrapno. Rješenje je dodavanje događaja (eng. „Event“) koji će na svaku promijenu dimenzije (eng. „Resize“), brisati postojeći i crtati novi, što se odvija gotovo trenutno Slika 3.17.

```
private void Form1_Resize(object sender, EventArgs e)
{
    CLEAR();
}
```

Sl. 3.17. *Resize funkcija*

Uz linije koordinatnih osi, smještene su vrijednosti u pojedinim labela koje također doživljavaju promjenu u veličini istovremeno sa cijelim virutalnim prozor, na Slici 3.18. prikazane su labele sa vrijednostima osi ordinata, koje označavaju vrijednosti kuta.

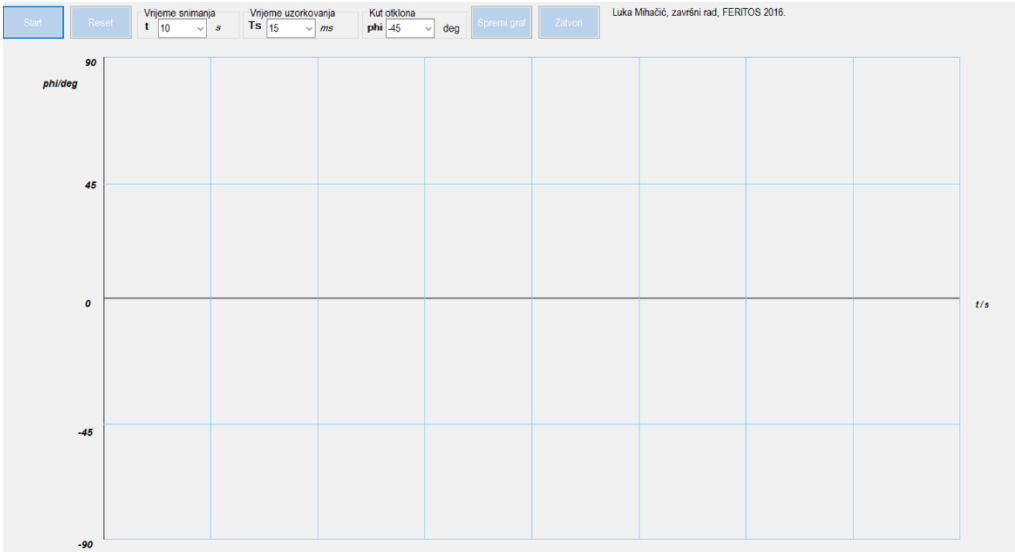
```
label7.Location  = new Point(x1 - 90, y1+30);
label8.Location  = new Point(x2 + 20, (this.Height/2));
label9.Location  = new Point(x1 - 30, y1);
label11.Location = new Point(x1 - 30, Convert.ToInt32(this.Height / 3.3));
label10.Location = new Point(x1 - 30, this.Height/2);
label12.Location = new Point(x1 - 40, Convert.ToInt32(this.Height / 1.4));
label13.Location = new Point(x1 - 40, y2);
```

Sl. 3.18. *Mijenjanje veličina labela*

Jedan vrlo bitan aspekt odnosi se na količinu piksela za koju se može okrenuti valjak mehaničkog miša između 0 i 180 stupnjeva. Kod korištenog miša taj broj iznosi oko 50 piksela, što je vrlo malo i teško je takav odziv pregledno prikazati na velikom ekranu kada je dužina osi ordinata oko 670 piksela. Potrebno je provesti skaliranje veličina tako da ćemo eksperimentalno odrediti pojačanje koje je potrebno ne bi li tih 50 piksela pokrilo cijeli ekran, odnosno raširiti ih na neki način na 670 piksela, time bi se osigurala preglednost i lakše očitavanje vrijednosti. To je učinjeno tako da se veći broj, u ovom slučaju podijeli sa 50 pri čemu se dobije zaokružena vrijednost od 14. Tim pojačanjem množili smo trenutni položaj y osi „TrenutnaTocka“, koja je već ranije uvećana za „Offset“ ili vrijednost y0 što označava da je u početnom položaju, odnosno ishodištu koordinatnog sustava (Slika 3.19.). Sada je moguće prikazati konačni izgled grafa pri pokretanju (Slika 3.20.)

```
if (n < (N - 1)){
    n++;
    SnimljenaTocka[n].x = Convert.ToInt32(Convert.ToDouble(x1) + Convert.ToDouble(n) * Convert.ToDouble(ax));
    SnimljenaTocka[n].y = Convert.ToInt32(Convert.ToDouble(TrenutnaTocka.y - yOffset) * POJACANJE) + yOffset;
    CRTAJ_UZORAK();
}
else{
    STOP();
```

Sl. 3.19. *Prikaz funkcije CRTAJ_UZORAK() i pojačanja*

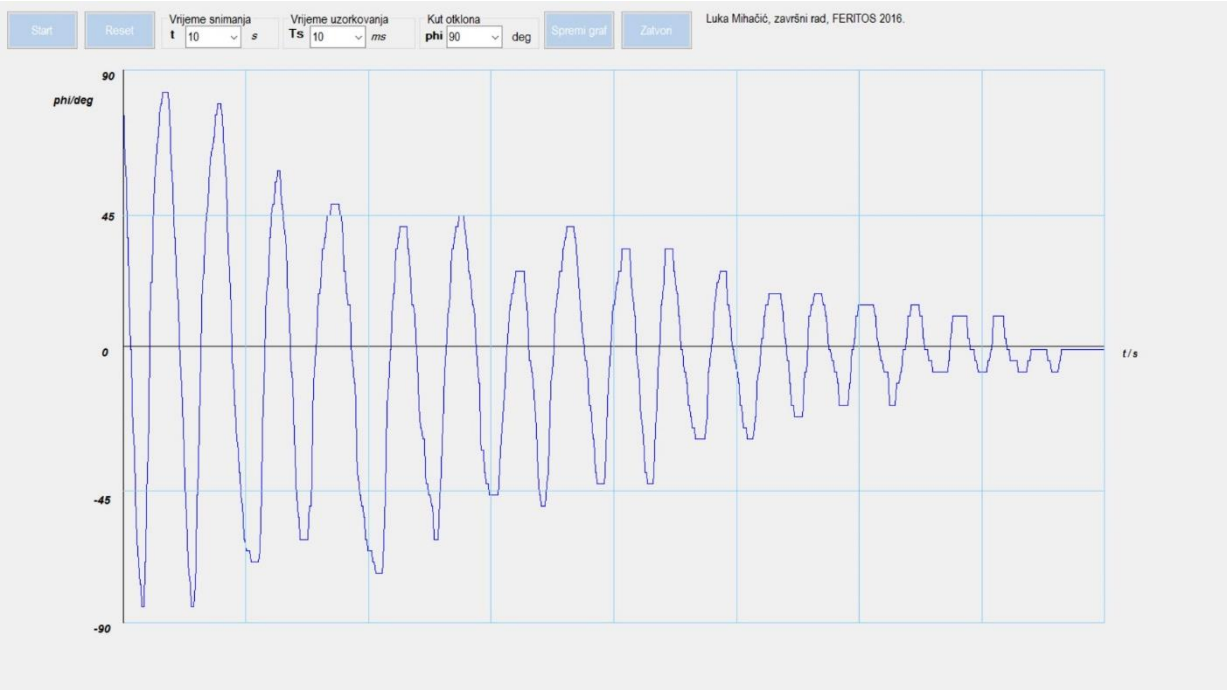


Sl. 3.21. *Konačni izgled glavnog prozora pri pokretanju aplikacije*

3.3. Testiranje i korištenje

Dizajn glavnog sučelja aplikacije je zathjevao pregledan i jednostavan prikaz svih sporednih dijelova, pritom se misli na odabir pojedinih vrijednosti za iscrtavanje grafa, te gumbova koji se koriste za pokretanje crtanja i spremanja slike zaslona, koji nebi bili toliko očiti ili prekrivali bilo koji dio glavnog dijela aplikacije, a to je prostor crtanja grafa. Problem je bio postaviti vrijeme na stvarno vrijeme „real time“ u programu Visual Studio, kao i određivanje početne točke od koje kreće snimanje za pojedinu izabranu vrijednost kuta, ali uspješno je rješeno. Izabrana je bijela pozadina sa blagim nijansama plave kako se ništa ne bi posebno isticalo na glavnom sučelju.

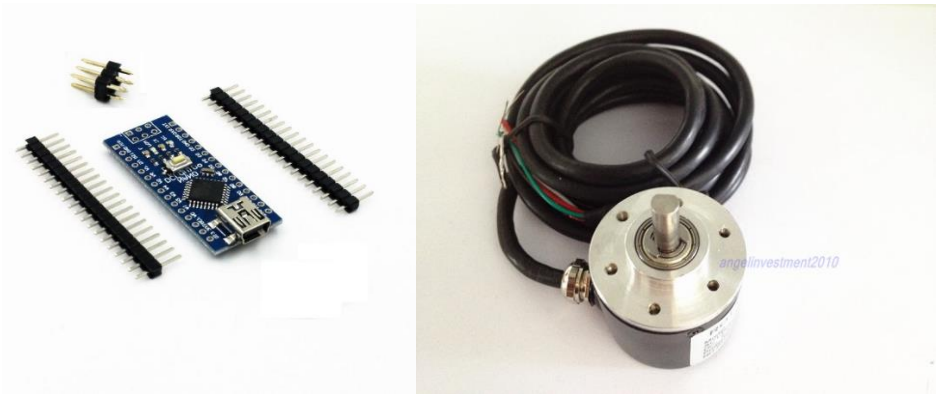
Testiranje se odvijalo na dva ekrana dijagonala 15" i 21". Prva dijagonala pripada prijenosnom računalu, a druga monitoru koji je takoder spojen na prijenosno računalo. Na oba ekrana različitih razlučivosti aplikacija je pokazala dobro ponašanje i stabilnost. Nije uočena nikakva deformiranost budući da aplikacija uzima rezoluciju ekrana i prilagođava virtualni prozor koji se nalazi i nije vidljiv unutar glavnog prozora aplikacije digitalnog njihala, što je u konačnici rezultiralo preglednošću same aplikacije. Snimak grafa sa početkom u 90 stupnjeva dan je Slikom 3.21. Potrebno je objašnjenje grafa. Na početku izgleda kao da ne počinje od 90 stupnjeva, ali kada se pusti štapić ta promjena bude brza. Primjećuje se da štapić ne dolazi do drugih 90 stupnjeva, a to je isključivo zbog fizičkog ograničenja miša, budući da postoji dio koji malo nekada zasmeta. Vidljivo je kako se ne prigušuje posve postupno, a to je zbog debljine štapića koja nekad zna zasmetati te prigušenje nije posve točno iz istog razloga kao pije, a to je fizičko ograničenje. Iz priloženog se vidi da aplikacija radi kako treba i da je uspješno obavljen programerski dio.



Sl. 3.21. *Snimak grafa sa početkom u 90 stupnjeva*

4. BUDUĆI RAD

Ovaj rad moguće je unaprijediti na mnoge načine. Iako sve radi kako treba Windowsi sami i nisu baš najbolji odabir za ovako nešto. Prvi problem se javlja kod vremena koje ne uzimaju kao realno nego sa malim odstupanjima od stvarnog vremena. Puno pogodnije rješenje za tako nešto postavlja se Arduino Nano na kojem se može osim vrlo preciznog vremena prikazati i vrijeme digitalno na digitalnim pokaznicima, fizičkim gumbima koji mogu pokretati i zaustavljati crtanje.



Sl. 4.1.. *Arduino Nano razvojna pločica (a) i digitalni enkoder (b)*

Kako korišteni miš ima mali broj piksela po okretaju, predstavlja dosta problema od nepreglednosti pri crtanju na velikom ekranu potreba skaliranja do nepreciznosti zbog male mogućnosti prikazivanja piksela po položaju što ne ostavlja glatku liniju iza sebe, nego zrnatiju. Ovaj enkoder od 600 piksela po okretaju ima veću mogućnost uzorkovanja, naravno i veću preciznost. Puno je praktičniji od rastavljenog mehaničkog miša na koji se sa trenutnim ljepilom lijepi drveni štapić. Metalna građa i rupe na kučištu enkodera omogućavaju pričvršćivanje na čvrstu podlogu bez micanja, što predstavlja veliko olakšanje, kao i zavarivanje metalnog štapića za metalno kučište. Za početak vrlo obećavajuće jer nema zabrinutosti o lakom puknuću ili odvajanju. Postoji trenje između metalnih dijelova, no ono je zanemarivo. Takvo unaprijeđenje stajalo bi 2.33 američkih dolara (USD) što je 14.74 kuna (HRK) za Arduino Nano i nešto više za enkoder. Cijena enkodera je 15.03 američkih dolara, što je oko 100.25 kuna.

Za ovakvu opremu koja bi se mogla nabaviti putem internet trgovine „Ebay“ za manje od 120 kuna moglo bi se toliko toga napraviti, što bi se zaista moglo koristiti u svrhe učenja i biti od pomoći kako studentima, tako i profesorima, samo je potrebno programerskih vještina i volje za izradu

5. ZAKLJUČAK

Aplikacija digitalnog njihala pomoću jednostavnog enkodera prerađenog miša sa kuglicom dokazuje da se pomoću minimalnog utroška materijala i poznavanja teorijskih osnova fizike i poznnavanja programskih jezika može napraviti vrlo koristan alat za učenje i bolje razumijevanja fizikalnih pojava koje studentima omogućuje bolje shvaćanje i razumijevanje teorijskih znanja, primjenjenim na praktičan način.

Zadani ciljevi su ispunjeni, unatoč problemima koji su se stvorili pri izradi aplikacije i u nekim trenutcima usporili izradu i iziskivali dobro promišljanje i otklanjanje problema. Pri pokretanju početni prozor aplikacije mogućuje korisniku odabiranje parametara njihala klikom miša, kao što su vrijeme trajanja za koje aplikacija snima titraje digitalnog njihala, mogućnost izbora kuta otkolona, kao i vremena uzorkovanja. Pokretanje se izvršava klikom miša na gumb „Start“, a nakon snimljenog odziva digitalnog njihala moguće je usnimiti taj graf isto tako pritiskom na gumb, kako bi se on mogao kasnije, zbog nemogućnosti za vrijeme snimanja, pomnije proučiti i izračunati određene parametre.

Zahvaljujući programskom okruženju Visual Studio, enkoder miša lagano je povezan preko USB priključka sa računalom i mogli smo koristit sve njegove karakteristike. Miš je fizički prerađen u digitalno njihalo u samo nekoliko koraka.

U budućnosti se može poraditi na poboljšanju aplikacije, dodati neke nove mogućosti, neke nove grafičke elemente ili slično jer prostora za nove ideje uvijek ima.

LITERATURA

[1] Wikipedia, Microsoft Visual Studio

https://bs.wikipedia.org/wiki/Microsoft_Visual_Studio (stranica posjećena: 24. lipnja 2016.)

[2] Wikipedia, C Sharp (programming language), https://en.wikipedia.org/wiki/C_Sharp_(programming_language) (stranica posjećena: 24. lipnja 2016.)

[3] Stack Overflow, Timer

http://stackoverflow.com/questions/2989350/timer-takes-10-ms-more-than-interval (stranica posjećena: 13. lipnja 2016.)

[4] Microsoft, Visual Studio

https://social.msdn.microsoft.com/Forums/vstudio/en-US/79efecc4-fa6d-4078-afe4-bb1379bb968b/print-screen-in-c?forum=csharpgeneral (stranica posjećena: 13. lipnja 2016.)

[5] Stack Overflow, Combox; Etfos, LV 6 „Objektno orijentirano programiranje“

http://stackoverflow.com/questions/2831082/convert-combobox-string-value-to-int

https://app.box.com/files/0/f/3670768617/1/f_30572825209 (stanica posjećena: 13. lipnja 2016.)

[6] Microsoft, Drawing in C#

https://msdn.microsoft.com/en-us/library/w02swk1z(v=vs.90).aspx (stanica posjećena: 17. lipnja 2016.)

[7] Technopia, Drawing Graphics

http://www.techotopia.com/index.php/Drawing_Graphics_in_C_Sharp (stanica posjećena: 17. lipnja 2016.)

[8] Ebay

http://www.ebay.com/itm/NEW-Incremental-Rotary-Encoder-600p-r-6mm-Shaft-5-24vdc-/121568516551?hash=item1c4e0c59c7:g:Y~MAAMXQS6pRxbt1, Encoder

http://www.ebay.com/itm/MINI-USB-Nano-V3-0-ATmega328P-CH340G-5V-16M-Micro-controller-board-Arduino-T1-/181846906547?hash=item2a56eb96b3:g:Ir8AAOSwBahVL6BH, Arduino Nano

(stanica posjećena: 9. rujna 2016.)

SAŽETAK

Cilj ovoga rada bio je izrada aplikacije digitalnog njihala za Windows operacijski sustav koji će studentima omogućiti bolje uočavanje praktičnih znanja i proširivanje kao i bolje utvrđivanje već stečenih teorijskih znanja. Teorijski dio rada obuhvaća obuhvaća osnove o razvojnom okruženju Visual Studio i programskoj jezika C#. Opisano je korištenje i opcije glavnog prozora sa Windows Form Application, kao i kartice i alati koji su dostupni za izradu ove aplikacije digitalnog njihala. Glavno sučelje je jednostavno i sastoji se od gotovih formi kojima su pridodane funkcije, te praznog prostora u kojem se iscrtava graf klikom na gumb. Moguće je odabrati brojevne vrijednosti karakteristika njihala padajućim izbornikom već predefiniranih vrijednosti. Stvaranjem virutalnog prozora unutar glavnog omogućeno je crtanje grafa neometano ne prelazeći ostale forme koje se nalaze u prozoru i time osiguravajući preglednost i jednostavnost.

Ključne riječi: Windows, Visual Studio, C#, fizikalno njihalo, enkoder

ABSTRACT

Windows application of digital pendulum

The goal of this project was to develop an desktop application of digital pendulum for Windows operating system that will help students to obtain practical knowledge and expand and better understanding of their previously acquired theoretical knowlegde. The theoretical part of this assignment includes the basics of integrated development environment Microsoft Visual Studio and programming language C# in which application was written. Furthermore, it contains description and options of main window of Windows Form Application, also tabs and bulit-in tools and form designer which are available to use in order to make this particular application of digital pendulum. The main screen is simple and it consists of drop down menus known as comboboxes that are filled with already predefined values and drawn Cartesian coordinate system. Finally, by creating virtual window for graph of digital pendulum inside main window will enable drawing with no worries about covering other forms that are in the same main window as drawn graph, which results in transparency and simplicity.

Keywords: Windows, Visual Studio, C#, physical pendulum, encoder, graph

ŽIVOTOPIS

Luka Mihačić rođen je 10. travnja 1994. godine u Osijeku, Hrvatska. Stanuje u Belišću, na adresi Matije Gupca 2. Godine 2001. započinje osnovnoškolsko obrazovanje u OŠ Ivana Kukuljevića u Belišću, gdje prolazi svih osam razreda s odličnim uspjehom i izrazito aktivan u Košarkaškom Klubu Belišće. Nakon završene osnovne škole, 2009. godine upisuje opću gimnaziju u Valpovu. Sa odličnim uspjehom tijekom cijelog školovanja i položenom državnom maturom završava dotadašnje obrazovanje. Preddiplomski sveučilišni studij računarstva na Elektrotehničkom fakultetu u Osijeku upisuje 2013. godine izravnim upisom na temelju izrazitog uspjeha postignutim u srednjoškolskom obrazovanju. Od ostalih znanja i vještina posjeduje znanje engleskog jezika, poznaje rad na računalu te ima vozačku dozvolu B kategorije.

PRILOZI

DVD

- Visual Studio projekt
- Rad u .docx i .pdf formatu