ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

DEMONSTRACIJA DTMF SIGNALIZACIJE U MATLAB-U

– Diplomski rad –

Kandidat:

Đorđe Zmijanjac 2010/156

Mentor:

doc. dr Zoran Čiča

Beograd, Septembar 2016.

SADRŽAJ

SADRŽAJ **2**

1. UVOD **3**

2. DTMF **4**

2.1. IDEJA, NASTANAK I ISTORIJAT 4
2.2. PRINCIP RADA 5
2.3. SPECIJALNI ZNACI 6

3. APLIKACIJA **8**

3.1. SOFTVER ZA SIMULACIJU 8
3.2. DIZAJN APLIKACIJE 8
3.3. UPUTSTVO ZA KORISNIKA 9
3.4. KREIRANJE GRAFIČKOG INTERFEJSA 11
3.5. POKRETANJE APLIKACIJE 14

4. ZAKLJUČAK **16**

LITERATURA **17**

A. KOD APLIKACIJE **18**

A.1. TASTERI NA BROJČANIKU 18
A.2. RESET 20
A.3. IZLAZ IZ PROGRAMA 21

1. UVOD

Od samih početaka razvoja sistema za komunikaciju javljali su se različiti problemi u njihovom radu. Prvi modeli telegrafa (od 1839. godine) i telefona (od 1876. godine) bili su vrlo skupi, retki i tehnološki ograničeni. Bili su dostupni samo državnim institucijama, vojsci i bogatim pojedincima, dominantno su se koristili za tačka-tačka komunikaciju, te pitanje signalizacije nije odmah došlo do izražaja. Međutim, kako je tehnologija napredovala, postala je dostupnija, a i dometi linija su se povećali, pa su mreže počele da se šire. U bilo kojoj mreži sa više od jednog ulaza i izlaza potrebno je rešiti probleme identifikacije krajnjih korisničkih uređaja, a u samoj mreži problem komutacije. Tada je uspostavljen centralistički princip u radu telekomunikacionih sistema, koji će se održati veoma dugo. Tako, krajnji korisnici imaju pristupnu liniju koja ih povezuje sa lokalnom centralom, a zatim se komutacija vrši među centralama različitih nivoa hijerarhije. Sama komutacija je u prvo vreme izvođena ručno, prespajanjem kola na zahtev korisnika.

Za samu identifikaciju korisnika kao rešenje su se nametnuli korisnički brojevi (1879. godine), koji su u prvo vreme bili vrlo kratki i usmeno su se saopštavali operateru u centrali. Uskoro potom nastaje i prva automatska centrala (1892. godine). Kao prvi sistem za prenos brojeva nastaje pulsno biranje, da bi zatim, sa razvojem tehnologije primat preuzeli višefrekvencijski sistemi, odnosno, prešlo se na tonsko biranje brojeva.

Upravo jedan takav sistem, koji je standardizovan na globalnom nivou i danas predstavlja dominantni signalizacioni sistem za ove namene, biće tema simulacije u okviru ovog rada. U pitanju je DTMF (*Dual Tone Multi Frequency*) - dvotonski višefrekvencijski sistem. Cilj rada je da se izradi GUI (*Graphical User Interface*) - grafički korisnički interfejs, na kome će biti predstavljen standardni telefonski brojčanik. Na ovom brojčaniku se mogu birati brojevi, koji će generisati DTMF signale, sa prikazom u vremenskom i spektralnom domenu. Takođe, dodatna opcija će biti mogućnost dodavanja šuma na signal od strane korisnika, u vidu vrednosti SNR (*Signal to Noise Ratio*) - odnos signal-šum.

Rad je podeljen u četiri poglavlja. U prvom poglavlju dat je kratak uvod u obrađenu temu. U drugom ćemo se osvrnuti na istorijat razvoja telekomunikacionih sistema, i problematiku koja je dovela do razvoja DTMF sistema, koji je simuliran u okviru rada. Takođe, biće razjašnjen princip rada ovog sistema. U trećem poglavlju dati su podaci o softverskom alatu MATLAB, koji je korišćen za simulaciju, i o samoj aplikaciji: kratak opis aplikacije, uputstvo za korisnika, i uputstvo za kreiranje grafičkog interfejsa. Četvrto poglavlje je zaključak rada, a celokupan kod sa potrebnim komentarima se nalazi u prilogu.

2.DTMF

U ovom poglavlju ćemo ukratko predstaviti problematiku koja je morala biti prevaziđena pri razvoju telekomunikacionih sistema, i posledični nastanak jednog ovakvog sistema. Takođe, biće razjašnjen princip rada DTMF signalizacionog sistema.

2.1. Ideja, nastanak i istorijat

Kao što je već navedeno, pri radu prvih telegrafskih i telefonskih sistema, problemi identifikacije korisnika nisu bili izraženi, ali kasnije se moralo naći rešenje za to. Posle dodeljivanja brojeva korisnicima i automatizacije centrala, ti brojevi su se morali prenositi kroz mrežu. Prvi sistem za ovu namenu je bio pulsni, čija je ideja kranje jednostavna. Princip je da se brojevi prenose unarnim kodom - povorkom unipolarnih impulsa koji nastaju fizičkim prekidanjem lokalnog kola. Svakom broju se dodeljuje broj impulsa jednak samom tom broju, uz to da broj ''0'' ima 10 impulsa (moguće su i drugačije kombinacije). Ovakvi pravougaoni impulsi bili su spori za generisanje, i skloni deformaciji pri prolasku kroz telefonski kanal, te su greške bile česte. Uprkos svojim manama, pulsni sistem se dugo održao jer nije bilo tehnoloških preduslova za razvoj boljeg.

Slika 2.1.1. - Telefon sa rotacionim brojčanikom[10]

Na Slici 2.1.1. možemo videti telefon sa rotacionim brojčanikom namenjen za pulsno biranje brojeva. Pojedini telefoni ovog tipa su se mogli ''prevariti'' – Ako je korisnik bio dovoljno vešt, mogao je birati broj bez brojčanika, brzim pritiskanjem telefonske viljuške odgovarajući broj puta. Ovakvi telefoni se i danas mogu ponegde naći u upotrebi.

Preduslov za iskorak ka boljem sistemu je bio razvoj elektronike, do kog dolazi u toku i posle Drugog svetskog rata. Tada nastaju ideje i prve realizacije sistema kod kojih se kodiranje

brojeva vrši generisanjem signala različitih frekvencija. To su *Multi Frequency*, odnosno višefrekvencijski sistemi. Na istraživanju višefrekvencijskih sistema prednjačile su Belove laboratorije, gde je DTMF i prezentovan (1963. godine). Njemu je prethodilo nekoliko sličnih sistema. Održao se u upotrebi do današnjih dana, na globalnom nivou.

2.2. Princip rada

Standardni skup simbola DTMF sistema sastoji se od 16 simbola, usvojenih kao brojevi *0-9* i specijalni znaci: *, #, *A*, *B*, *C* i *D*. Postoje dve grupe frekvencija, viša i niža. Obe grupe sadrže po četiri različite frekvencije. Svakom simbolu su dodeljene po dve frekvencije, jedna iz grupe viših i druga iz grupe nižih, i simbol se prenosi emitovanjem signala koji čini zbir dve sinusoide na datim frekvencijama.

Tabela 2.2.1. - Raspored frekvencija po simbolima.

Niže f.\Više f.	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

U Tabeli 2.2.1. možemo videti raspored frekvencija po simbolima u okviru sistema DTMF, pri čemu je viša grupa frekvencija u gornjem redu tabele, a niža grupa frekvencija u koloni sa leve strane. Raspored simbola u prve tri kolone odgovara rasporedu na standardnom telefonskom brojčaniku. Krajnja desna kolona je na većini komercijalnih uređaja izostavljena jer sadrži znake koji nisu od interesa za krajnjeg korisnika, kao što se vidi na Slici 2.2.2.

Slika 2.2.2. - Telefon sa standardnim brojčanikom od 12 simbola[11]

Frekvencije koje su korišćene za realizaciju sistema su pažljivo odabrane, po nekoliko kriterijuma. Prvi je bio da se moraju nalaziti u opsegu govornog signala, tj. 300 – 3400 Hz, da bi mogle da se prenose kroz govorne kanale. Određena je grupa viših i nižih frekvencija, dovoljno međusobno udaljenih da bi se mogle jednostavno razdvojiti tadašnjim filtrima. Same frekvencije su odabrane tako da svaka sledeća bude sa prethodnom u odnosu od 21/19, i da nijedna nije jednaka zbiru ili razlici neke druge dve, niti njihovih umnožaka. Ovim je postignuto maksimalno potiskivanje eventualnih harmonika koji bi naškodili signalu. Čist sinusoidalni signal, kakav je potreban za ovakav sistem, nije moguće imitirati ljudskim glasom, tako da nema problema pri prenosu kroz telefonski kanal.

Pri slanju signala obično se praktikuje slanje više frekvencije većom snagom, jer je ona osetljivija na eventualna slabljenja. Po standardima, signal više frekvencije se šalje snagom većom za 1 - 4 dB (najčešće 3). Maksimalno validno frekvencijsko odstupanje na predaji iznosi ±(1.5% + 2Hz) u odnosu na definisanu vrednost (opciono do ±1.8%). Da bi bio validan, prenošeni simbol mora da sadrži znak, tokom kog se emituje signal i pauzu, tokom koje nema signala. Kao minimalni intervali definisani su: bar 40 ms za znak i pauzu, kao i bar 85 ms za celokupan simbol. Pojedini proizvođači koriste i veće intervale (čak do 70 ms za znak), da bi se smanjila mogućnost greške. Maksimalna brzina signalizacije DTMF sistema je 12.5 simbola u sekundi, što je značajno više od pulsnog sistema, uz manju mogućnost greške.

Na prijemu, odstupanje frekvencije ne sme biti veće od 3.5%, inače se signal odbacuje. Za detekciju su se u početku koristile banke filtara na prijemu a kasnije se prešlo na digitalnu obradu signala. Danas se detekcija vrši primenom Gercelovog algoritma. To je modifikacija Diskretne Furijeove transformacije, koja daje dobre rezultate pri računanju malog broja frekvencijskih odbiraka od značaja, kao što je i slučaj kod ovog sistema.

2.3. Specijalni znaci

Pored 10 simbola koje čine brojevi *0-9*, sistem omogućava i prenos dodatnih znakova: *, #, *A*, *B*, *C* i *D*. Prva dva su našla primenu za krajnje korisnike, te se zato nalaze na brojčanicima aparata, dok su slovni znaci uklonjeni. Upotrebom znakova * i #, mogu se zahtevati neki korisnički servisi.

Znakovi *A*, *B*, *C* i *D* su ostali u upotrebi samo za signalizaciju među centralama, kao i za radio amatere i daljinsku kontrolu uređaja, npr. ripitera. Prva upotreba ovih znakova je bila za vojne potrebe, pri čemu su služili za prioritetizaciju saobraćaja.

Slika 2.3.1. - Telefonski brojčanik sistema Autovon[12]

Na Slici 2.3.1. prikazan je brojčanik telekomunikacionog sistema Autovon (*Automatic Voice Network*), koji je projektovan za vojne potrebe početkom šezdesetih godina XX veka. Ovde je prisutno svih 16 simbola, a 4 slovna simbola imaju oznake: FO (*Flash Override*), F (*Flash*), I (*Immediate*), P (*Priority*). Na taj način, omogućava se dobijanje prioriteta pri pozivu, koji su hijerarhijski poređani od najnižeg (P), do najvišeg (FO). Na ovom brojčaniku su vidljiva i slova pridružena brojnim simbolima, koja se takođe mogu prenositi DTMF simbolima, za potrebe dobijanja određenih servisa, pristupanja uređajima, i slično. Ako želimo prenos slova ispisanih na brojčaniku, treba samo pritisnuti taster sa željenim slovom onoliki broj puta koji odgovara mestu slova na samom tasteru. Znak * u tom slučaju služi za brisanje, a # za potvrdu unosa.

3.APLIKACIJA

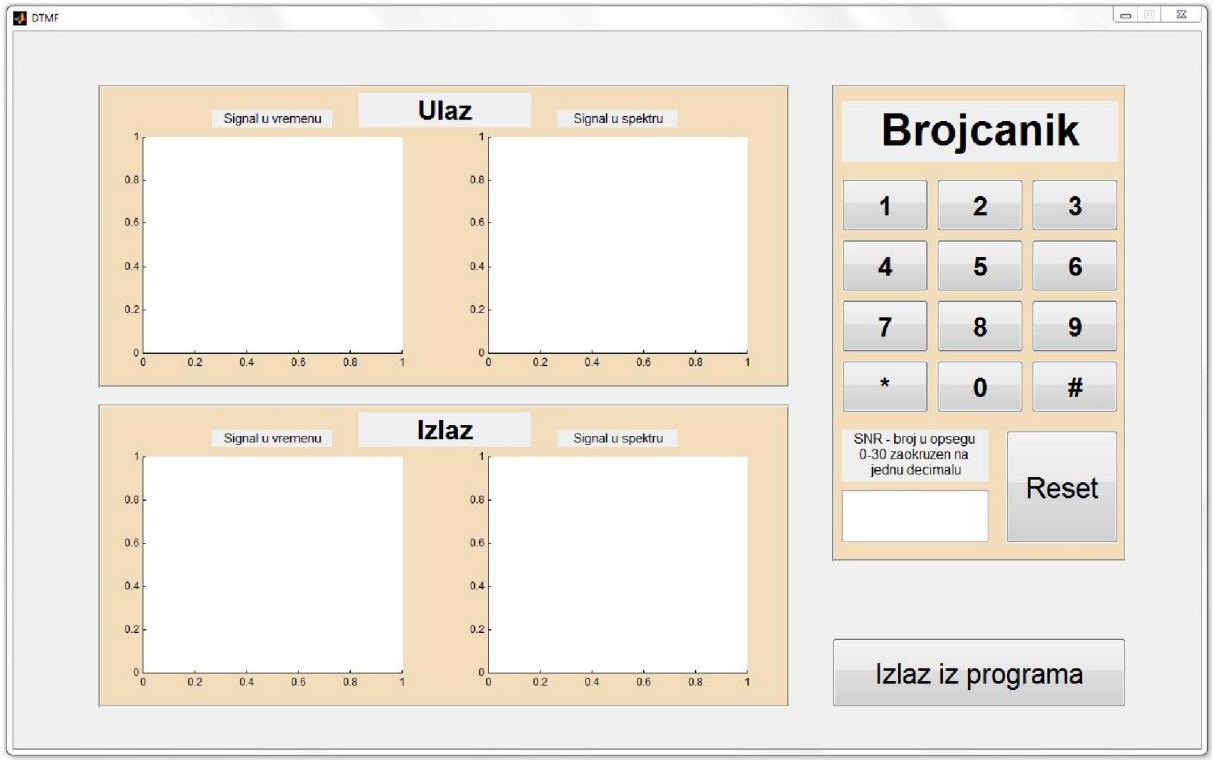
U okviru ovog poglavlja biće predočene osnovne informacije o samom softveru u kome je izvršena simulacija, a zatim će biti detaljno objašnjen princip rada i sam proces kreiranja grafičkog interfejsa aplikacije.

3.1. Softver za simulaciju

Softverski alat u kome je izvršena simulacija je MATLAB (*matrix laboratory*). To je programski paket, inicijalno pokrenut od strane kompanije MathWorks 1984. godine. Njegovo ime označava karakteristiku da se svi podaci predstavljaju matricama. Podesan je za razne vrste inženjerskih zadataka, omogućava brzo i jednostavno izračunavanje i prikaz rezultata, kao i kreiranje grafičkih interfejsa. Od njegovog inicijalnog pokretanja do danas izašao je veliki broj verzija, a matlab kod simulacije koja je obrađena u sklopu ovog rada je napisan u verziji R2013a.

3.2. Dizajn aplikacije

Po otvaranju programa ''DTMF.m'' iz softverskog alata MATLAB i pritiskom na taster *Run*, dobijamo osnovni prozor aplikacije, tj. grafički interfejs, u kome možemo započeti simulaciju.



Slika 3.2.1. - Osnovni prozor aplikacije.

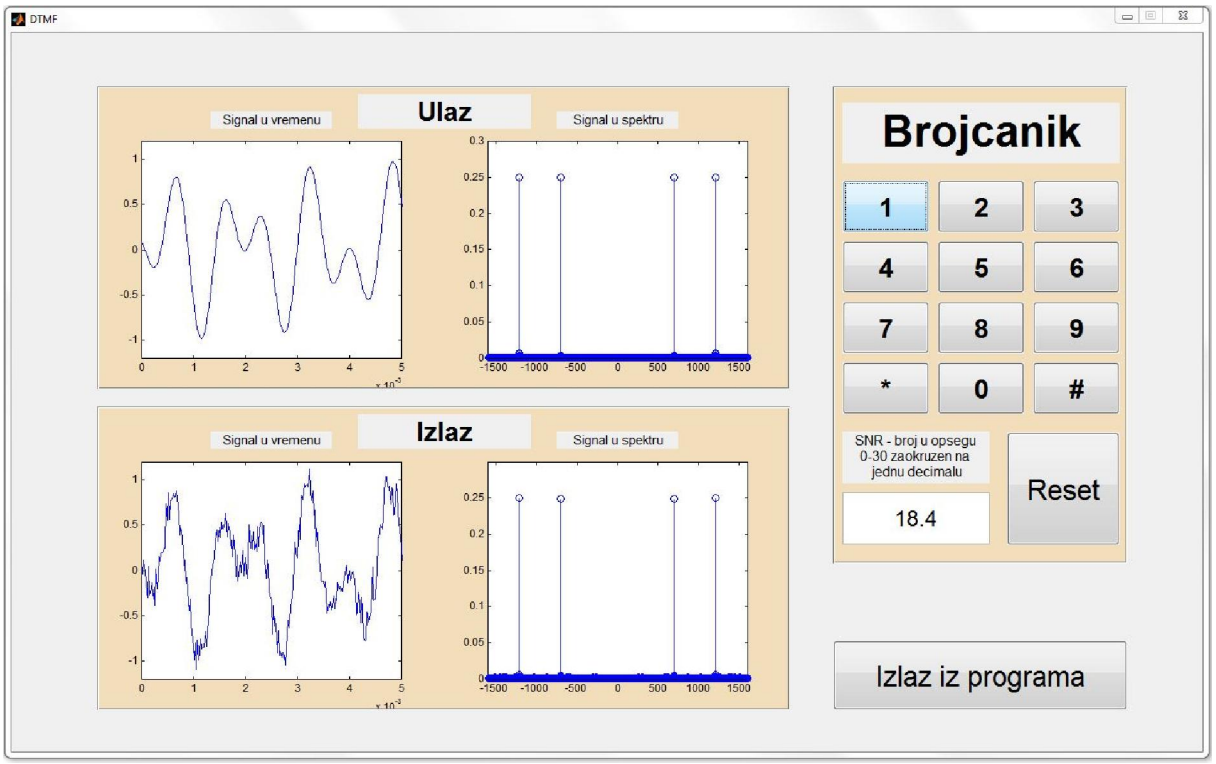
Na Slici 3.2.1. možemo videti prikaz osnovnog prozora aplikacije. Na njemu se mogu videti tri celine:

- Deo za prikaz signala, sa leve strane. Na njemu imamo četiri grafika. Dva grafika na gornjem panelu za prikaz signala na ulazu (originalni generisani signal), i dva na donjem panelu za prikaz signala na izlazu (signal sa dodatim šumom). Za oba signala imamo prikaz u vremenu, sa leve, i prikaz u spektru, sa desne strane. Svi grafici su dinamički, tj. prikazuju signal u vremenu, i reaguju na promenu vrednosti SNR.
- Brojčanik, sa desne strane. Ovde je predstavljen standardni telefonski brojčanik, sa tasterima *0-9*, * i #. Takođe, ispod tastera se nalazi polje za unos šuma, iskazanog kroz vrednost SNR. Tu je i dodatni taster *Reset*, koji poništava unos i prikazane funkcije na graficima.
- Taster za izlaz iz programa, koji se nalazi u donjem desnom uglu. On zatvara aplikaciju.

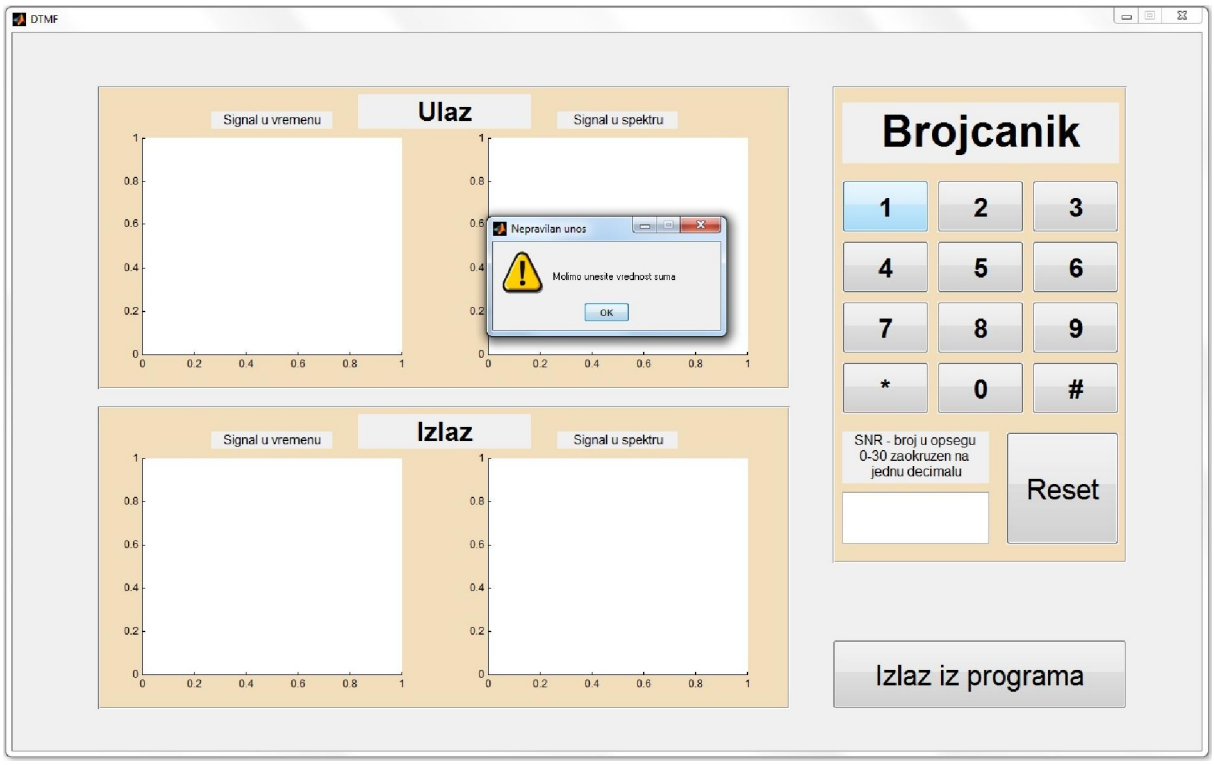
3.3. Uputstvo za korisnika

Rad u ovakvom grafičkom interfejsu je jednostavan i intuitivan. Korisnik prvo unosi vrednost odnosa signal-šum u polje za unos. Ova vrednost je dostupna u opsegu od 0 (situacija kada su srednje snage signala i šuma jednake) do 30 (situacija kada je signal 1000 puta jači od šuma). Vrednost se unosi zaokružena na prvu decimalu.

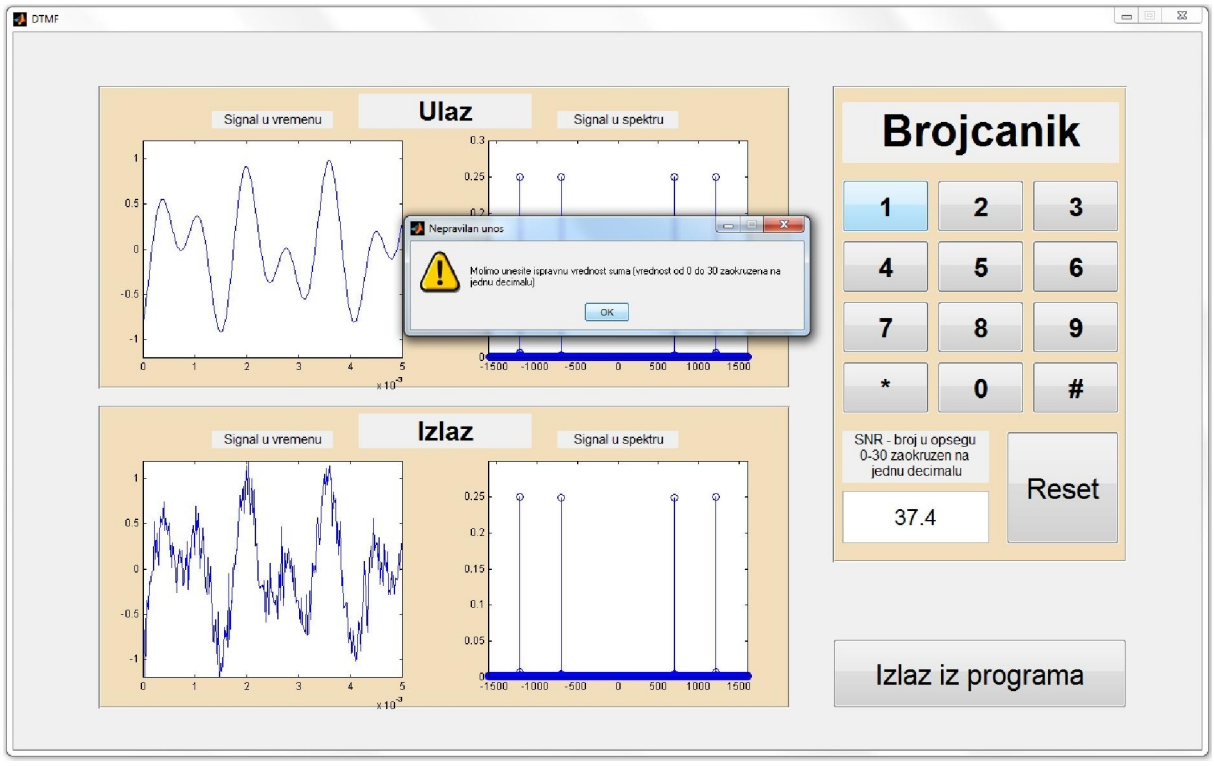
Zatim se bira neki od tastera čiji DTMF signal želimo da vidimo. Dalje, po želji, korisnik bira tastere, ili menja odnos signal-šum u polju za unos. Po unosu novog broja, pritiskom na taster *Enter* na tastaturi nova vrednost se preuzima, te se odmah može videti njen efekat na graficima sa leve strane. Ako korisnik želi prazne grafike i polje za unos, to se postiže klikom na taster *Reset*.



Slika 3.3.1. - Prozor aplikacije tokom rada.



Slika 3.3.2. - Upozorenje da nije uneta vrednost SNR.



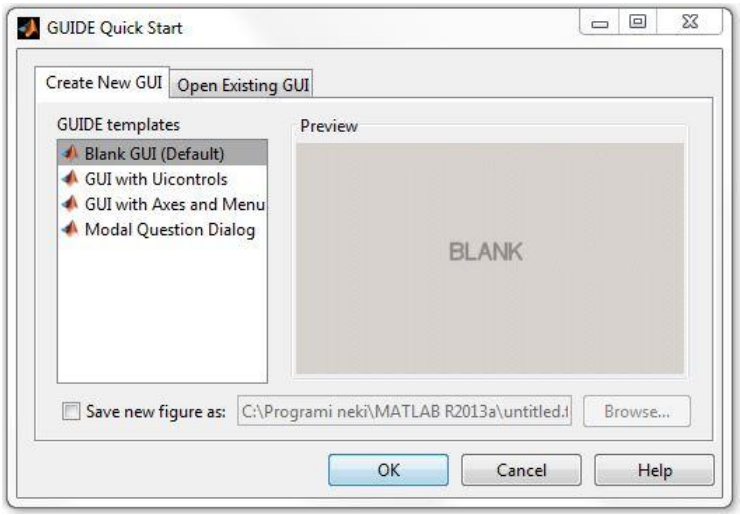
Slika 3.3.3. - Upozorenje da je unos loš.

Na Slici 3.3.1. možemo videti izgled grafičkog interfejsa aplikacije tokom simulacije (konkretno u ovom slučaju pritisnut je taster *1*). Grafici sa leve strane se prikazuju u realnom vremenu. Na njima se može videti uticaj promenljivog šuma na signal, kao i efekti promene vrednosti SNR.

Što se tiče mogućih grešaka, moguće je da korisnik ostavi prazno polje za unos, ili da uneta vrednost ne odgovara zahtevanom opsegu vrednosti odnosa signal-šum. U oba slučaja pojavljuje se *Messagebox* prozor, koji podseća korisnika da unese vrednost SNR ako to nije učinio (Slika 3.3.2.), ili zahteva ispravnu vrednost ukoliko je uneta pogrešna (Slika 3.3.3.).

3.4. Kreiranje grafičkog interfejsa

Kreiranje grafičkog interfejsa za jednu ovakvu aplikaciju započinjemo ulaskom u MATLAB, i kucanjem komande ''guide''. U prozoru koji se tada pojavi, biramo opciju *Create New GUI*.



Slika 3.4.1. - Prozor u kome nam se nude opcije za kreiranje grafičkog interfejsa.

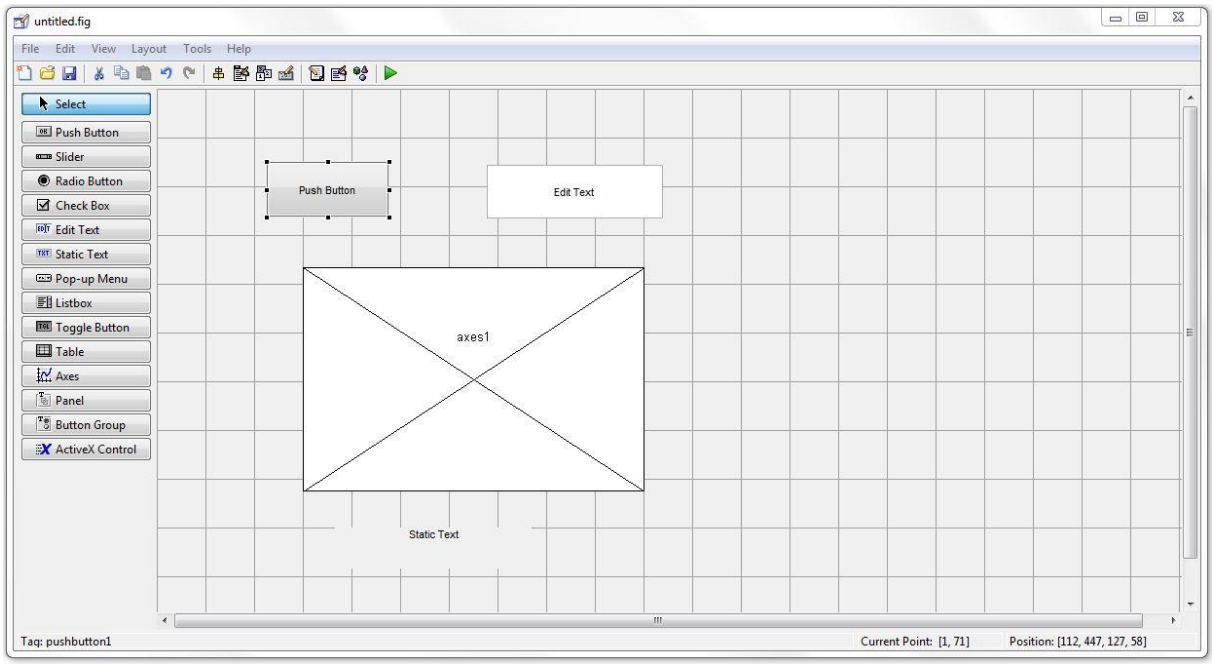
Na Slici 3.4.1. vidimo prozor koji se pojavljuje posle komande ''guide'', i u kome možemo izabrati da kreiramo nov GUI (sa nekim dodatnim elementima) ili da otvorimo neki već postojeći. Biramo *Create New GUI → Blank GUI (Default)*.

Na Slici 3.4.2. vidimo prozor za kreiranje grafičkog interfejsa dobijen izborom opcije *Blank GUI (Default)*. On ima tri celine:

- Najveći deo prozora čini radna površina na kojoj ćemo kreirati sam interfejs.
- Na gornjem delu prozora je meni linija i standardni tulbar, na kome imamo opcije *Save* (sačuvati), *Run* (pokrenuti), *Align* (poravnati elemente). Tu su još neke opcije koje se mogu po potrebi upotrebiti.
- Na levom delu prozora su elementi za kreiranje interfejsa. Od ponuđenih, koristićemo: *Push Button* (taster), *Edit Text* (polje za unos), *Static Text* (statički tekst), *Axes* (grafik) i *Panel* (panel). Tasteri su nam potrebni za predstavljanje telefonskih tastera kojima ćemo pokretati simulaciju odgovarajućih signala, kao i dva dodatna, *Reset* i *Izlaz iz programa*. U polje za unos će korisnik unositi potrebne podatke, u našem slučaju odnos signal-šum. Statički tekstovi su nam potrebni za imenovanje odgovarajućih delova GUI-a, što će pomoći organizaciji i boljem snalaženju korisnika. Na graficima će se prikazivati željeni

signali, a paneli su podesni za lakšu organizaciju i bolji vizuelni utisak samog grafičkog interfejsa.

Slika 3.4.2. - Prozor za kreiranje grafičkog interfejsa.

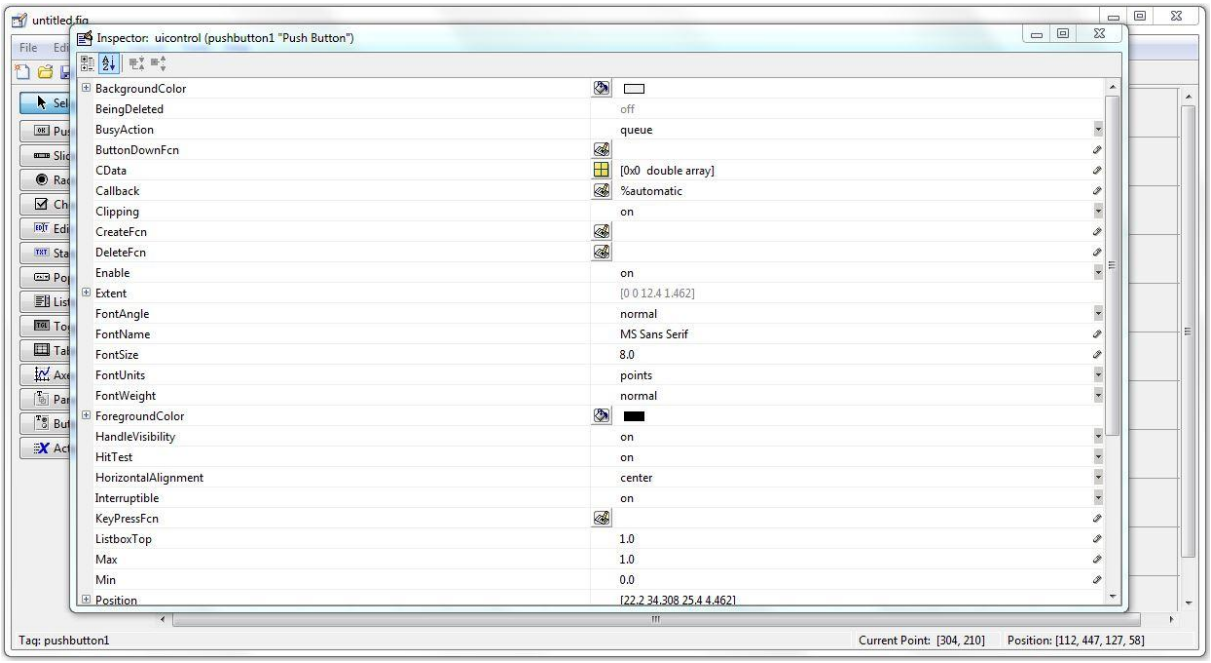


Slika 3.4.3. - Dodavanje elemenata.

Na Slici 3.4.3. vidimo neophodne elemente postavljene na radnu površinu. Elementi se preuzimaju i postavljaju klikom miša. Sada dizajniramo GUI, tako što raspoređujemo elemente na radnu površinu, a zatim podešavamo njihov izgled i osobine.

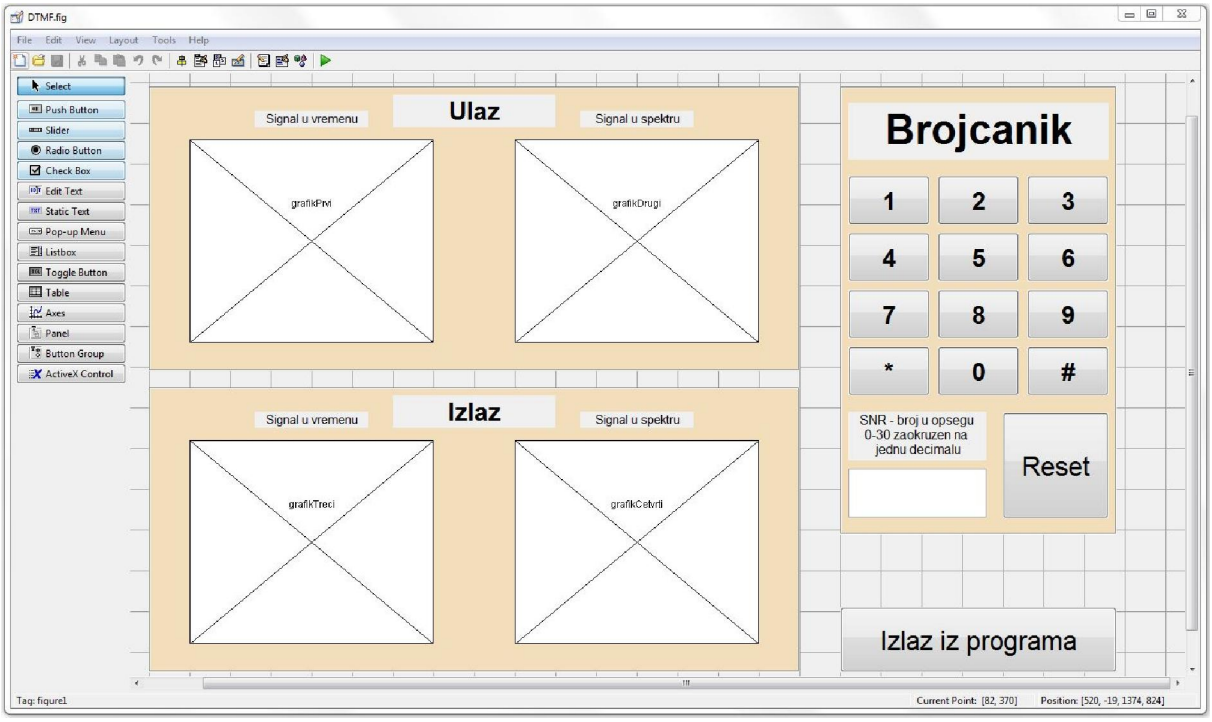
Osobine elemenata podešavamo u prozoru *Property inspector*, koji se može videti na Slici 3.4.4. *Property inspector* možemo pokrenuti ili dvostrukim klikom na željeni element čije karakteristike želimo da podesimo ili desnim klikom pa izborom opcije *Property inspector* iz padajućeg menija. Ovde podešavamo osobine elementa (dimenzije, položaj, boju, naziv, font, kao i ostale ukoliko su nam potrebne). Pošto nam je potrebno više elemenata istih karakteristika, konkretno 12 tastera, 4 grafika i 2 panela, možemo kreirati samo jedan element, i onda ga kopirati potreban broj puta. Time štedimo vreme, a istovremeno redukujemo mogućnost greške, tj. kreiranja dva različita elementa u situacijama kada su nam potrebni identični.

Najznačajnija osobina elementa je njegov tag, tj. ime njemu pridružene promenljive sa kojom ćemo kasnije pozivati ovaj element i dodeljivati mu podatke, ili ih očitavati sa njega. Svaki element ima svoj jedinstveni tag. Važno je da tagovi budu konzistentni, i da ne dođe do grešaka pri dodeli, jer su oni od ključnog značaja za ispravno funkcionisanje programa. Tasterima u ovom programu su dodeljeni tagovi: *dugmeJedan*, *dugmeDva*,..., *dugmeDevet*, *dugmeNula*, *dugmeZvezdica*, *dugmeTaraba*, *dugmeReset* i *dugmeIzlaz*. Grafici su nazvani: *grafikPrvi*, *grafikDrugi*, *grafikTreci* i *grafikCetvrti*, a polje za unos je nazvano *poljeZaUnos*. Preostali elementi, paneli i polja sa statičkim tekstom, neće biti pozivani u programu, te njihovi tagovi nisu od značaja, i mogu se dodeliti proizvoljno.



Slika 3.4.4. - Property inspector prozor.

Na Slici 3.4.4. vidimo prozor *Property inspector*, u kome se podešavaju osobine elemenata.



Slika 3.4.5. - Konačan izgled grafičkog interfejsa.

3.5. Pokretanje aplikacije

Pošto je dizajniran celokupan GUI, kao što se vidi na Slici 3.4.5. možemo pristupiti izradi koda. Prvo, sačuvamo GUI, pod imenom ''DTMF.fig'', a zatim se automatski generiše ''.m'' fajl, u kome ćemo kucati kod za pojedinačne elemente. Sačuvamo ga, takođe kao ''DTMF.m''.

Primer automatski generisanog koda za taster *1*:

```
% --- Executes on button press in dugmeJedan.
function dugmeJedan_Callback(hObject, eventdata, handles)
% hObject    handle to dugmeJedan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Zatim u okviru pojedinačnih *callback* funkcija, kucamo kod za svih 12 simulacionih tastera, kao i dodatne za *Reset* i *Izlaz iz programa*.

Što se tiče simulacionih tastera, pri svakom pritisku na neki od njih potrebno je:

1) Učitati vrednost sa polja za unos.
2) Proveriti da li je vrednost dobra, i zatim ako nije, obavestiti korisnika, a ako jeste, nastaviti dalje u program. Provera se vrši upotrebom *if-else-elseif* strukture, tj. imamo dve grane za dve moguće greške: Ako korisnik ostavi polje za unos prazno ili ako unese neodgovarajuću vrednost.
3) Definisati zajedničke promenljive na vrhu, i ući u glavni deo programa, u okviru *for* petlje. Unutra se nalaze grafici, koji se u svakoj iteraciji petlje osvežavaju, da bi se dobio prikaz u realnom vremenu, i prikazao uticaj šuma na signal.

4) U okviru same petlje, u svakoj iteraciji se vrši novo učitavanje vrednosti iz polja za unos, i dodatna provera, da korisnik ne bi naknadno uneo neodgovarajuću vrednost SNR. Provera u petlji je identična kao inicijalna provera, tj. sastoji se od *if-else-elseif* strukture.

Svi tasteri na brojčaniku, tj. numerički tasteri (*0-9*) kao i dva specijalna znaka (* i #), imaju identičan kod, sa razlikom korišćenih frekvencija (promenljive *fv* i *fn* se dodeljuju posebno za svaki taster). U slučaju da su nam potrebni i slovni znaci na brojčaniku, dodajemo ih analogno postojećim, uz korišćenje frekvencije iz poslednje kolone (1633 Hz). U tom slučaju, mora se promeniti opseg x-ose na graficima 2 i 4, da bi bila obuhvaćena i ova dodatna frekvencija. Celokupan kod za simulacione tastere dostupan je u prvom podpoglavlju priloga (taster *0*).

Što se tiče tastera *Reset*, njegov kod je idejno isti kao kodovi za tastere na brojčaniku, ali je znatno kraći. Potrebno je obrisati sadržaj polja za unos, i zatim na graficima poništiti sve vrednosti, što postižemo iscrtavanjem niza popunjenog nulama. Takođe je realizovan putem *for* petlje. Ovde nije potrebna provera unosa. Kod se nalazi u drugom delu priloga.

Taster *Izlaz iz programa* ima namenu samo da isključi aplikaciju. Njegov kod je najkraći, i dostupan je u trećem delu priloga.

4.ZAKLJUČAK

U ovom radu je predstavljen i simuliran jedan bitan sistem, čije je funkcionisanje od velikog značaja za moderne TK sisteme. Uprkos tome što je nastao pre više od pola veka, to je jedan robustan i pouzdan sistem. Iako se uveliko razvijaju alternativni sistemi za ove namene, izvesno je da će se DTMF održati u upotrebi još nekoliko decenija, i sa njim treba svakako biti upoznat.

Takođe, uvideli smo prednosti računarske simulacije sistema, koja je danas neophodna u svim sferama života. Moderni sistemi su složeni i skupi, te bi njihovo stvarno zauzimanje u cilju prezentacije, merenja i testiranja bilo skupo, i vremenski i finansijski. Čak i kod ovako jednostavnog sistema, vidimo kako nam softverska simulacija daje alternativu, da možemo predstavljati sistem bez zauzeća ikakvih stvarnih resursa.

Jedna ovakva simulacija je višestruko korisna, može se koristiti u sklopu pojedinih predmeta, kao ispomoć pri učenju softverskog alata MATLAB, za početnike. Takođe, na zanimljiv način prezentuje rad sistema i daje mogućnost da ga korisnik sam isporbava i posmatra rezultate. Ovakve i slične simulacije mogu poslužiti na laboratorijskim vežbama, da bi se slušaocima lakše dočarao rad pojedinih sistema.

Sama aplikacija može poslužiti kao dobra osnova za dalju nadgradnju, u više mogućih varijanti. Prvo, moguće je, sa malom modifikacijom koda i uz pomoć zvučnika, proveriti rad sistema na pravom telefonu. U tom slučaju imali bismo proveru da li aplikacija zaista radi ono što treba. Takođe, aplikacija se može proširiti dodatkom još jednog sistema za prepoznavanje signala, kao i za praćenje uticaja šuma na signal, pri čemu bi mogle biti simulirane karakteristike pravog telefonskog kanala. U okviru takvog dodatka mogao bi biti simuliran Gercelov algoritam.

LITERATURA

[1] http://www.etsi.org/deliver/etsi_es/201200_201299/20123502/01.01.01_60/es_20123502v010101p.pdf

[2] https://www.specialtyansweringservice.net/wp-content/uploads/resources_papers/dtmf-tone/Dual-Tone-Multi-Frequency-Signalling.pdf

[3] http://www.genave.com/dtmf.htm

[4] https://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling

[5] https://en.wikipedia.org/wiki/Autovon

[6] http://www.tech-faq.com/abcd-tones.html

[7] https://groups.google.com/forum/#!topic/comp.dcom.telecom/xhh6Lu3IUaA

[8] http://www.makeuseof.com/tag/autovon-the-phone-system-designed-for-world-war-3/

[9] http://www.mathworks.com/

[10] http://bachelor-mag.com/tbt-landline-phones/

[11] http://www.staritelefoni.si/

[12] http://www.classicrotaryphones.com/forum

A. KOD APLIKACIJE

A.1. Tasteri na brojčaniku

```
% --- Executes on button press in dugmeNula.
function dugmeNula_Callback(hObject, eventdata, handles)    % Automatski
generisani deo koda za taster '0'.

% Promenljive za unos šuma.
    PA = 0:1:300;                           % Inicijalni skup vrednosti.
    A = PA./10;                             % Skup validnih vrednosti šuma za
unos (od 0 do 30 dB sa zaokruživanjem na jednu decimalu).
    S = get(handles.poljeZaUnos,'string');  % Preuzimanje vrednosti za proveru
da li je polje za unos prazno. Vrednost je preuzeta kao string.
    snr = str2num(S);                       % Konvertovanje vrednosti preuzete
sa polja za unos u numeričku vrednost za dalju obradu.

% Deo koji se odnosi na prepoznavanje unosa sa edittext polja.
if isempty(S)                                                              %
Provera da li je polje za unos prazno, pomoću funkcije 'isempty'.
    h = msgbox('Molimo unesite vrednost suma','Nepravilan unos','warn')    %
Ako je polje za unos prazno, korisnik se podseća da unese vrednost.
    uiwait                                                                 %
Pauza da bi korisnik imao vremena da unese vrednost u polje za unos. Funkcija
'uiwait' zaustavlja izvršenje programa dok se ne unese odgovarajuća vrednost.
elseif (ismember(snr,A))                                                   %
Provera da li je uneta odgovarajuća vrednost i, ako jeste, prolazak dalje u
program. Funkcija 'ismember' proverava da li je uneti član jedan od elemenata
inicijalno zadatog skupa 'A'.

% Zajedničke promenljive.
    t = linspace(0,1,50001);    % Vremenska osa.
    fn = 941;                   % Niža frekvencija.
    fv = 1336;                  % Viša frekvencija.
    Fs = 50000;                 % Zadavanje broja odbiraka frekvencijske ose.
    dF = 1;                     % Definisanje odbirka frekvencijske ose.
    f = -Fs/2:dF:Fs/2;          % Frekvencijska osa za dvostrani spektar.

% Glavni deo programa.
    for a = linspace(0,5,100000)      % Ulazak u for petlju, kao glavni deo
programa.

    S = get(handles.poljeZaUnos,'string');              % Dodatno preuzimanje
vrednosti u svakoj iteraciji za proveru da li je polje za unos prazno, ako
korisnik tokom rada izmeni snagu šuma.
    snr = str2num(get(handles.poljeZaUnos,'string'));   % Konvertovanje u
numeričku vrednost za proveru.
        if isempty(S)
% Provera da li je polje za unos prazno, pomoću funkcije 'isempty'.
            h = msgbox('Molimo unesite vrednost suma','Nepravilan unos','warn')
% Ako je prazno polje, korisnik se opominje da unese vrednost.
```

```
            pause(5);
% Pauza da bi korisnik imao vremena da unese vrednos.
        elseif (ismember(snr,A))
% Provera da li je uneta odgovarajuća vrednost i, ako jeste, prolazak dalje u
program.

    x = @(t)((sin(2*pi*fn*t) + sin(2*pi*fv*t))/2);              % Signal,
normalizovan na maksimalnu vrednost 1.
    y = @(t)(awgn((sin(2*pi*fn*t) + sin(2*pi*fv*t))/2, snr))    % Signal, sa
dodatim šumom, dodat funkcijom awgn.

axes(handles.grafikPrvi);       % Na grafiku 1 će biti signal na ulazu u
vremenu.
    plot(t,x(t-a))              % Pokretni grafik, koji se osvežava u svakoj
iteraciji petlje sa promenom brojača a.
    xlim([0 0.005])             % Ograničenje x ose.
    ylim([-1.2 1.2])            % Ograničenje y ose.
    drawnow;                    % Funkcijom 'drawnow' se osvežava grafik posle
svake iteracije, da bismo postigli željeni efekat, tj. pokretnu funkciju na
grafiku.

axes(handles.grafikTreci);      % Na grafiku 3 će biti signal na izlazu u
vremenu, sve je identično kao za prethodni grafik, samo je funkcija druga.
    plot(t,y(t-a))              % --- grafik ---
    xlim([0 0.005])             % --- grafik ---
    ylim([-1.2 1.2])            % --- grafik ---
    drawnow;                    % --- grafik ---

axes(handles.grafikDrugi);      % Na grafiku 2 će biti signal na ulazu u
spektru.
    X = fftshift(fft(x(t)));    % Furijeova transformacija signala, da dobijemo
njegov spektar.
    stem(f,(abs(X)/Fs))         % Crtanje spektra, funkcija 'abs' se koristi
pošto nam treba samo amplitudska karakteristika. Funkcija 'stem' je podesnija za
prikaz spektra nego 'plot', te je zato koristimo.
    xlim([-1600 1600])          % Podešavanje x ose na grafiku.
    ylim([0 0.3])               % Podešavanje y ose na grafiku.
    drawnow;                    % Funkcijom 'drawnow' se osvežava grafik posle
svake iteracije.

axes(handles.grafikCetvrti);    % Na grafiku 4 će biti signal na izlazu u
spektru.
    Y = fftshift(fft(y(t)));    % Računamo Furijeovu transformaciju signala u
svakoj iteraciji, jer se šuma menja, i u svakoj iteraciji ćemo imati novu
vrednost odbiraka spektra na izlazu. To se radi funkcijom 'fft'.
    stem(f,(abs(Y)/Fs))         % --- grafik ---
    xlim([-1600 1600])          % --- grafik ---
    ylim([0 0.3])               % --- grafik ---
    drawnow;                    % --- grafik ---

        else
% Else grana if-else strukture. Ukoliko vrednost nije zahtevani broj.
    h = msgbox('Molimo unesite ispravnu vrednost suma (vrednost od 0 do 30
zaokruzena na jednu decimalu)','Nepravilan unos','warn') % Poruka za korisnika
da je unos loš.
    pause(5);
% Pauza dok korisnik unese odgovarajuću vrednost.
        end
% Kraj if-else strukture za proveru u programu.
```

```
    end
% Kraj for petlje, kao glavnog dela programa.

else
% Else grana if-else strukture. Ukoliko vrednost nije validna, tj. ako broj
odstupa iz definisanog opsega, ili je korisnika greškom uneo nešto što nije
broj.
    h = msgbox('Molimo unesite ispravnu vrednost suma (vrednost od 0 do 30
zaokruzena na jednu decimalu)','Nepravilan unos','warn') % Poruka za korisnika
da je unos loš.
    uiwait
% Funkcija 'uiwait' zaustavlja izvršenje programa, dok se ne unese odgovarajuća
vrednost.
end
% Kraj inicijalne provere, tj. glavne if-else strukture.
```

A.2. Reset

```
% --- Executes on button press in dugmeReset.
function dugmeReset_Callback(hObject, eventdata, handles)   % Automatski
generisani deo koda.

t = linspace(0,0.5,2000);   % Vremenska osa.
x = zeros(2000,1);          % Vrednosti na vremenskoj osi.
f = linspace(0,1600,1600);  % Frekvencijska osa.
y = zeros(1600,1);          % Vrednosti na frekvencijskoj osi.

set(handles.poljeZaUnos,'string','');   % Briše sadržaj edittext polja.

for a=linspace(0,1,5000)

axes(handles.grafikPrvi);       % Na grafiku 1 će biti signal na ulazu u
vremenu.
    plot(t,x)                   % Pokretni grafik.
    xlim([0 0.005])             % Ograničenje x ose.
    ylim([-1.2 1.2])            % Ograničenje x ose.
    drawnow;                    % Osvežavanje grafika.

axes(handles.grafikTreci);      % Na grafiku 3 će biti signal na izlazu u
vremenu.
    plot(t,x)                   % Pokretni grafik.
    xlim([0 0.005])             % Ograničenje x ose.
    ylim([-1.2 1.2])            % Ograničenje x ose.
    drawnow;                    % Osvežavanje grafika.

axes(handles.grafikDrugi);      % Na grafiku 2 će biti signal na ulazu u
spektru.
    stem(f,y)                   % Pokretni grafik.
    xlim([0 1600])              % Ograničenje x ose.
    ylim([0 0.3])               % Ograničenje x ose.
    drawnow;                    % Osvežavanje grafika.

axes(handles.grafikCetvrti);    % Na grafiku 4 će biti signal na izlazu u
spektru.
    stem(f,y)                   % Pokretni grafik.
    xlim([0 1600])              % Ograničenje x ose.
    ylim([0 0.3])               % Ograničenje x ose.
```

```
    drawnow;                    % Osvežavanje grafika.

end
```

A.3. Izlaz iz programa

```
% --- Executes on button press in dugmeIzlaz.
function dugmeIzlaz_Callback(hObject, eventdata, handles)

clc;        % Čisti sve.
close all;  % Zatvara sve.
```