Programsko rješenje uređaja za rehabilitaciju poremećaja slušanja i govora

Tafra, Ivica

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2501

PROGRAMSKO RJEŠENJE UREĐAJA ZA REHABILITACIJU POREMEĆAJA SLUŠANJA I GOVORA

Ivica Tafra

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2501

PROGRAMSKO RJEŠENJE UREĐAJA ZA REHABILITACIJU POREMEĆAJA SLUŠANJA I GOVORA

Ivica Tafra

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 2501

Pristupnik:	**Ivica Tafra (0036504374)**
Studij:	Elektrotehnika i informacijska tehnologija
Profil:	Elektronika
Mentor:	izv. prof. dr. sc. Marko Horvat

Zadatak: **Programsko rješenje uređaja za rehabilitaciju poremećaja slušanja i govora**

Opis zadatka:

Analiza i obrada audiosignala nalazi mnogobrojne primjene u raznovrsnim tehnologijama koje su u uporabi u suvremenom svijetu i društvu. Jedno od područja primjene jest i rehabilitacija poremećaja slušanja i govora. U začecima elektrotehnike i elektronike, a prije pojave široko dostupnih računalnih sustava, analiza i obrada audiosignala oslanjala se na namjenski razvijene i proizvedene elektroničke uređaje. U današnjem se svijetu rad tih uređaja može u potpunosti oponašati odgovarajućim ekvivalentnim programskim rješenjima. U radu je potrebno opisati metode rehabilitacije poremećaja slušanja i govora. Nadalje, potrebno je analizirati rad elektroničkog uređaja pod nazivom Suvag Lingua koji se uspješno koristi upravo u ovu svrhu te opisati njegove glavne značajke i tehničke specifikacije, kao i korisničko sučelje za upravljanje uređajem. Na temelju ovih saznanja potrebno je izraditi programsko rješenje uporabom programskog jezika C++ i razvojnog okvira JUCE koje će u potpunosti i s punom funkcionalnošću oponašati rad uređaja Suvag Lingua. Potrebno je izraditi i odgovarajuće grafičko korisničko sučelje koje će biti preslika onoga na polaznom elektroničkom uređaju te će korisniku pri radu omogućiti jednostavan prijelaz s elektroničkog uređaja na programsko rješenje. Prednost ovakvog rješenja ogleda se u proširenju i pojednostavljenju primjene jer je za provedbu rehabilitacije slušanja i govora dovoljno posjedovati osobno računalo s instaliranim programskim rješenjem umjesto fizički izvedenog elektroničkog uređaja.

Rok za predaju rada: 28. lipnja 2024.

Hvala mentoru profesoru Marku Horvatu na podršci i svoj pomoći koju je pružio za ovaj rad.

Hvala svim profesorima, vanjskim suradnicima, asistentima i tajnici Zavoda za elektroakustiku na pruženoj pomoći, znanju i usmjeravanju.

Hvala svim zaposlenicima Fakulteta elektrotehnike i računarstva.

Hvala mojoj obitelji, a pogotovo roditeljima na svakoj žrtvi koju su učinili za mene.

Hvala mojoj prekrasnoj supruzi što je bila moja inspiracija, moj kamen oslonac i nezamjenjiv dio mog života.

Ovaj rad posvećujem njoj i svakom njezinom prošlom i budućem rehabilitantu.

Sadržaj

1. Uvod

1954. godine akademik Petar Guberina objavio je novu metodu rehabilitacije slušanja i govora *System Universal Verbotonal d'Audition Guberina*, ili skraćeno SUVAG. Verbotonalna metoda bila je doista revolucionarna s obzirom na vrijeme u kojem se razvijala te je, kao takva, zahtijevala popratnu elektrotehničku opremu koja je išla uz korak tehnologiji tog vremena. Za potrebe verbotonalne metode u rehabilitaciji, prvi elektroakustički uređaji napravljeni su već 1955. godine. Kao nasljednik njima, stvoren je uređaj SUVAG LINGUA VFA L11 kojim će se ovaj diplomski rad baviti. SUVAG LINGUA VF1 L11 je audioelektronički uređaj kojim se upravlja i modificira zvučnim signalom kako bi zvučne primjere korištene za vrijeme rehabilitacije maksimalno prilagodili slušnom spektru rehabilitanta.

SUVAG LINGUA sastoji se od šest modula: niskopropusni aktivan filtar, visokopropusni aktivan filtar, mrežni ispravljač, ulazno pretpojačalo, dva aktivna modularna filtra te izlazno pojačalo. Budući da je za potrebe ovog rada bitna samo direktna modularna modifikacija, od svih navedenih modula potrebno je emulirati samo niskopropusni i visokopropusni filtar te jedan modularni pojasnopropusni filtar. Uporabom računala osigurano je pretpojačalo te izlazno pojačalo na samoj zvučnoj kartici računala. Također, iz istog razloga nema potrebe za sklopom za napajanje.

Emulacijom SUVAG LINGUAe na računalu pokušava se osigurati besplatan pristup odgovarajućoj opremi za svakog rehabilitatora koji slijedi verbotonalnu metodu. Time rad Guberine ostaje relevantan, moderan i uz korak današnjoj tehnologiji te više ne ovisi o nezgrapnom uređaju zastarjele tehnologije. Široko dostupan uređaj direktno znači i povećanje opsega rada verbotonalnom metodom, a samim time i potpuno iskorištavanje njezinih blagodati.

Kako bi se uspješno preslikala tehnologija iz elektrotehničkog područja u programsko sučelje, potrebno je izložiti princip verbotonalne metode, analizirati rad i korisničko sučelje uređaja te implementirati saznanja u obliku programske potpore uz JUCE razvojno sučelje. Od završnog proizvoda očekuje se da bude samostalan programski paket namijenjen raznim operacijskim sustavima i stručnjacima (npr. fonetičarima, logopedima).

JUCE razvojno sučelje pomaže u opisu strukture uređaja jer koristi Digital Signal Processing (DSP) ponavljajući vremenski prozor. Samim time, nema potrebe za direktnim upravljanjem

izlaznog signala, dovoljno je upravljati značajkama vremenskog prozora koji će preko DSPa upravljati izlaznim signalom. Želja je olakšati način upravljanja kako ne bi bilo potrebe ovisiti o elektroničkim komponentama (poput operacijskih pojačala u spoju povratne veze) koje imaju svoje varijabilnosti. Navedene varijabilnosti imaju negativan utjecaj na vjerodostojnost rada uređaja te proizlaze iz nesavršenosti proizvodnog procesa, prijenosnih karakteristika i tolerancije upotrjebljenih elemenata i rada uređaja u raznim temperaturnim opsezima.

Korištenjem računalnih procesa obrade podataka eliminira se utjecaj navedenih efekata, samim time, eliminiramo unesene šumove i varijabilnosti analognih tehnologija. Varijabilnost procesa sveli smo na svega par faktora. Linearnost mikrofonske prijenosne karakteristike i prijenosne karakteristike zvučnika ili slušalica neizbježan je faktor u svakoj obradi audiosignala koja zahtijeva snimanje zvuka bez obzira na tehnologiju same obrade. No, za razliku od analogne, izobličenje koje se u audiosignal unosi računalnom, digitalnom tehnologijom ovisi isključivo o kvaliteti zvučne kartice računala te o razini audiosignala. Napredak tehnologije dovodi nas u vrijeme u kojem su zvučne kartice iznimno visoke razine vjerodostojnosti i robusnosti bez obzira na kvalitetu ostalih komponenti računala jer se radi o tehnologiji koja je tijekom godina usavršena, stabilna i. Izobličenja koja mogu nastati zbog previsoke razine signala jednostavno je izbjeći osiguravanjem rada uređaja u predviđenom području amplituda signala.

2. Verbotonalna metoda rehabilitacije slušanja i govora

Slušanje se smatra jednom od osnovnih komponenti u komunikacijskom procesu. Ukoliko je slušanje uredno, utoliko učenje i usvajanje govora ne bi trebali predstavljati problem. S druge strane, ako je slušanje otežano ili narušeno, govor će trpjeti. U slučajevima slušnih i/ili govornih oštećenja, program rehabilitacije veoma je sveobuhvatan, a pri rehabilitaciji pokušava se utjecati, ne samo na rehabilitaciju slušnih poremećaja u slučaju odraslog rehabilitanta, već i na cjelokupni psihofizički razvoj djeteta koje pati od poremećaja slušanja i/ili govora. Jedan od takvih pristupa rehabilitaciji jest akustičko-perceptivna metoda poznatija kao verbotonalna metoda. Iako se ona većinski koristi u rehabilitacijske svrhe djece, njezina se djelotvornost može primijeniti i u svrhu učenja stranih jezika kod odraslih.

Pri proučavanju fenomena slušanja i razumijevanja govora kroz prizmu verbotonalne metode kao središnji kriterij i polazna točka uzima se percepcija, a Guberina (1996) napominje kako su glavni parametri govornih glasova „vrijeme, frekvencija, intenzitet, tijelo u smislu tjelesne vodljivosti ili koštane vodljivosti, napetost i pauza“. Bitno je napomenuti da se navedeni elementi koriste u smislu strukturiranja te time čine glavna polja interesa u verbotonalnom sistemu. Krenemo li od polazišne točke, tj. percepcije, Guberina (1996) ističe „(1) da je vrijeme strukturalni faktor, (2) da su ograničena frekvencijska područja dovoljna da se pod određenim uvjetima govor razumije, (3) da kombinacije frekvencija i intenziteta u diskontinuiranoj formi i sa stanovišta frekvencija i sa stanovišta intenziteta dovode do razumljivosti govora, (4) da čitavo tijelo radi i kao receptor i kao transmiter, (5) da je napetost rezultat agonističkih i antagonističkih mišića i (6) da pauza predstavlja aktivnost.“ Međusobnim kombiniranjem navedenih komponenti otvara se mogućnost stvaranju bogatijih fizičkih struktura govornih glasova.

2.1. Intenzitet

Proučavajući intenzitet i uzimajući u obzir polazišnu točku fenomena slušanja i razumijevanja govora - percepciju, verbotonalna metoda ističe važnost nakupljene zvučne snage (intenziteta) na frekvencijskim područjima koji do tada, nisu bili toliko opaženi od strane rehabilitatora i stručnjaka.

Primjer kojeg ističe Guberina (1996) može se predstaviti korištenjem glasa /i/. Propuštanjem glasa /i/ kroz oktavne filtre na uređaju SUVAG LINGUA, zabilježeno je kako oktava pojasa od 3200 do 6400 Hz glasu /i/ daje najbolju razumljivost. Najbolju razumljivost možemo opisati i kao optimalnu oktavu. „Optimalna je oktava frekvencijski pojas od jedne oktave u kojemu su glas, riječ ili rečenica najbliži izgovoru (glasu, riječi ili rečenici) snimljenom putem direktnog kanala.“ (Guberina, 2010). Krenemo li korak dalje i uzmemo li u obzir druge oktave većih amplituda, u pojasu od 150 do 300 Hz /i/ se čuje kao /u/, kao /o/ u pojasu između 300 i 600 Hz, kao /a/ između 800 i 1600 Hz te kao /e/ u oktavnom području između 1600 i 3200 Hz. Mnogo je ovakvih zakona i kombinacija frekvencijskih područja koje su rehabilitatori definirali.

2.2. Napetost

Napetost je već ranije spomenuta u definiranju glavnih parametara govornih glasova kao rezultat suprotstavljenosti između agonističkih i antagonističkih grupa mišića pri proizvodnji govornih zvukova. Mišićna napetost je, dakle, vrlo bitan faktor koji za zdravu osobu podrazumijeva uvjetovan refleks, dok za osobu oštećena sluha vrlo često zna nedostajati ili biti preizražena.

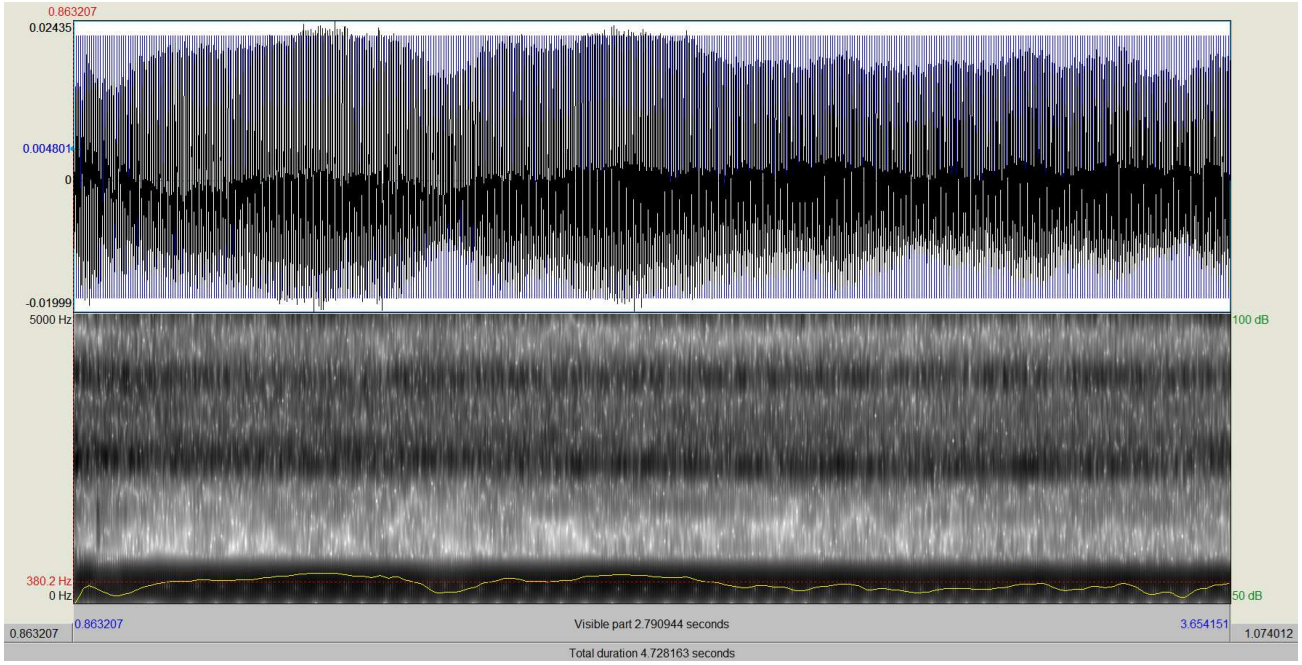
2.3. Pauza

Pauza je uključena u svakodnevno izražavanje naših misli. Posebice je važna u izrazima efektivnog tipa ili između iskaza koji međusobno stoje u određenoj logičkoj vezi (npr. da ne pada kiša, otišao bih u šetnju). Pauza ima i svoju fizičku bazu, a ona je sveprisutna i u drugim fizičkim parametrima – frekvenciji i vremenu. Uz mogućnost fizičkog definiranja, pauza svoju fiziološku važnost očitava i u svojoj ritmičkoj ulozi.

2.4. Frekvencija

Proučavamo li govorne glasove sa stajališta verbotonalnog sistema kroz percepciju kao glavnu kariku, frekvencije pojedinih glasova „uglavnom ne odgovaraju zvučnoj snazi ili linearnoj podjeli frekvencija kao što to pokazuju uobičajeni analizatori frekvencija.“ (Guberina, 1996). To se može objasniti koristeći ponovno glas /i/. Naime, najveća amplituda zvučne snage glasa /i/ nalazi se u frekvencijskom području od 200 do 400 Hz što je prikazano Slikom 2.1. Ako bi glas /i/ propuštali kroz navedeno područje na uređaju SUVAG LINGUA, on ne bi bio čujan.

Dakle, potrebno ga je propuštati kroz ranije navedeno područje od 3200 do 6400 Hz iako je njegova zvučna snaga u tom području skoro pa neznatna. To se događa zbog ranije već spomenute optimalne oktave glasa.



Slika 2.1. Spektrogramska analiza glasa /i/ Praat

2.5. Tijelo kao prijemnik i prijenosnik

Verbotonalna metoda svojim je istraživanjima percepcije glasova govora pokazala kako je „čitavo tijelo angažirano u percepciji glasova govora, a naročito u slučajevima teških oštećenja unutrašnjeg uha." (Guberina, 1996). U slučajevima teške nagluhosti ili gluhoće, tjelesna vodljivost svojim dobrim funkcioniranjem prenosi valove zvučnih izvora te se oni, ako rehabilitator ispravno strukturira emisiju i transmisiju, mogu uspješno transmitirati. Dakle, što je veće oštećenje sluha i lošija transmisija putem uha, dotični pojedinac sve je osjetljiviji na vibracije putem tjelesne vodljivosti. Guberina (1996) nadalje ističe kako „Udaljenost pojedinih dijelova tijela od centra percepcije i fizičke ili mehaničke karakteristike dotičnih dijelova tijela" utječu na „krajnju percepciju govornih glasova." Autor nadalje tvrdi: „… zato je pri konstruiranju aparata za teške nagluhe, i uopće za proučavanje percepcije govornih glasova, potrebno pridati veliku važnost gore navedenim faktorima koji se odnose na tijelo. Osnovna istraživanja verbotonalnog sistema i njegova primjena u rehabilitaciji sluha zasnovana su na stanovištu da čitavo tijelo služi kao receptor i prijenosnik. Zato aparati verbotonalne metode

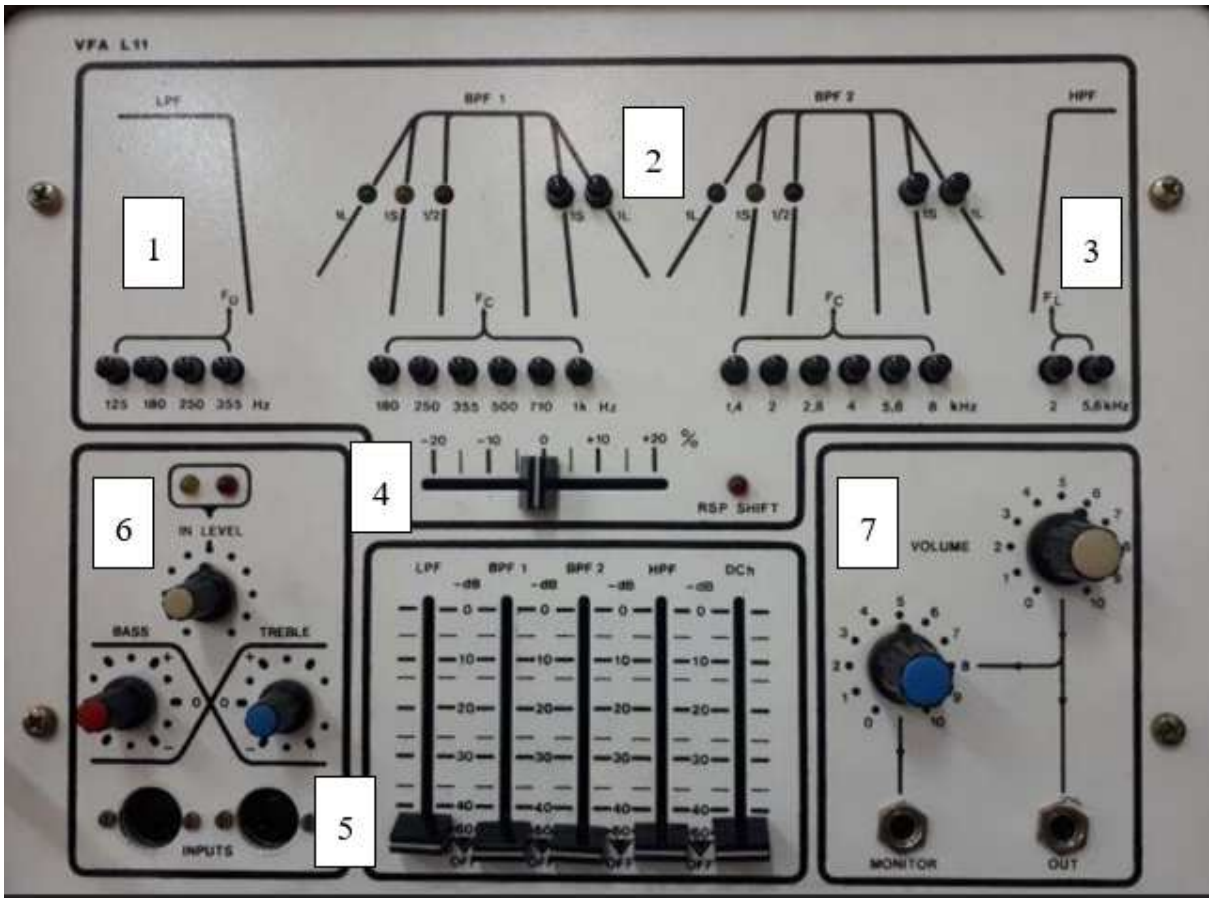
omogućuju linearnu transmisiju s različitim mogućnostima prigušenja pojedinih frekvencijskih područja - od 1 Hz na više, da bi se uključio frekvencijski spektar vibracija tijela."

2.6. Vrijeme kao strukturalni faktor

Vrijeme kao strukturalni faktor u rehabilitaciji govora i slušanja očituje se kao vremensko kašnjenje frekvencijskih komponenata određene riječi. Guberina (1996) daje primjer riječi /si/ ili logatoma /sisi/ koje je prenio diskontinuirano kroz pojasne filtre te će oni biti „percipirani kao da su emitirani linearno (bez faktora) ako su niskofrekvencijske komponente (u području od 300-600 Hz) emitirane od 50-200 milisekundi prije visokofrekvencijskih (u području od 3200-6400 Hz).". Znači, iako smo niske frekvencije prenijeli prije visokih, visoke frekvencije čut ćemo prije niskih. S druge strane, autor navodi logatom /mumu/ bogatog niskofrekvencijskog sadržaja za koji, ako bismo ga slušali pod istim uvjetima kao i logatom /sisi/, ne bismo dobili isti rezultat. U ovom slučaju, prije ćemo čuti niske frekvencije. Iz navedenog, dolazi se do važnog zaključka: "visokofrekvencijski sadržaj u glasovima govora percipiramo najmanje 25-50 milisekundi prije nego niskofrekvencijski. Što je viši glas, to ranije percipiramo visoke frekvencije glasa. To može ići i do 300 milisekundi kad se radi o prijenosu s kašnjenjem kroz dva pojasna područja. Što je glas niži, manja je mogućnost da se ranije čuje viši frekvencijski isječak glasa." (Guberina, 1996).

Zaključno, verbotonalni sistem uvelike pridonosi poznavanju procesa slušanja i govora te se još i danas primjenjuje u području patologije sluha te na područjima rehabilitacije slušanja i govora.

3. SUVAG LINGUA VFA L11



Slika 3.1. Aparat SUVAG LINGUA VFA L11

Na Slici 3.1. prikazana je upravljačka ploča uređaja SUVAG LINGUA. Na vrhu uređaja nalazi se upravljačko sučelje četiriju akustičkih filtara – niskopropusnog (označeno 1), dva pojasnopropusna (označeno 2) te visokopropusnog (označeno 3) filtra. Niskopropusni filtar propušta samo niske frekvencije, a visokopropusni samo visoke. To bi značilo da i visokopropusni i niskopropusni filtar odstranjuju govorni dio spektra. Označeno slovima „Fu“ ispod LPF (niskopropusnog) filtra nalaze se tipke kojima se namješta gornja granična frekvencija, a označeno slovima „FL“ ispod HPF (visokopropusnog) filtra nalaze se tipke za određivanje donje granične frekvencije. Niskopropusni filtar se u rehabilitaciji koristi za rad na globalnoj strukturi govora (intonacija, ritam…), a visokopropusni filtar koristi se, između ostalog, za postizanje diskontinuiteta (kod npr. nedovoljno napetog izgovora).

Pojasnopropusni filtar služi rehabilitatoru kako bi mogao namjestiti vrijednosti optimalnog frekvencijskog područja određenog glasa. Tako se s „pomoću akustičkih filtara aparata eliminiraju frekvencije koje ometaju slušanje pa osoba sluša samo one karakteristične za određenu emisiju čime mu se omogućava da čuje elemente koje prije nije mogao, da ih prepozna i u nefiltriranu govoru te da ih konkretno reproducira.“ (Vuletić, 1980).

Svaki od pojasnopropusnih filtara ima mogućnost odabira željene središnje frekvencije te strmog (60 dB/oktavi) ili blagog (30 dB/oktavi) nagiba frekvencijske karakteristike u području gušenja. Gušenjem se dodatno naglašavaju odabrane frekvencije, tj. frekvencijska područja jer se smanjuje intenzitet ostalih oktavnih područja govornog zvuka.

Označen brojem četiri, prikazan je klizni potenciometar kojim rehabilitator namješta vrijednost pomaka frekvencijske karakteristike željenog oktavnog područja (*engl*. response shift – RSP SHIFT). On je na ovom uređaju od velike važnosti jer se njime postiže optimalno filtriranje željenog glasa i optimalno slušno polje.

Iznad označenog broja pet, nalaze se klizni potenciometri za uključivanje željenih filtara, odnosno isključivanje direktnog kanala. Bitno je napomenuti kako u praktičnom radu na analognom uređaju nije moguće imati više filtara istovremeno uključeno u proces obrade, tj. u datom je trenutku koristan samo klizni potenciometar onog filtra koji je uključen u proces obrade. Ne postoji mogućnost da se ostvare dva pojasa propuštanja kombinacijama filtara. U analognom uređaju jednostavnije rješenje bilo bi to da svaki filtar ima svoj potenciometar u povratnoj vezi, umjesto da jedan potenciometar koristimo za pojačanje svakog od filtara. Potenciometri kanala služe za kontrolu dinamike izvornog zvuka u slučaju da signal ode u *clipping* (zato se i uvodi isključivo prigušenje, a ne i pojačanje). Zadaća pretpojačala jest poboljšanje prijenosne karakteristike mikrofonskog sustava na niskim frekvencijama te naglašavanje pojasno govornog frekvencijskog područja. Sve funkcije pojačanja u biti su samo kontrola glasnoće i dinamike ili kompenzacija nesavršenosti analognih sustava te nisu vezane uz direktnu kontrolu pojačanja i prigušenja filtara. Jedini faktori kojima se upravlja tijekom rehabilitacije su namještanje centralne i graničnih frekvencija te kontrola strmine područja gušenja. S druge strane, fina kontrola glasnoće unutar digitalne obrade podataka nije toliko bitna jer imamo cijeli *headroom* na raspolaganju (povratna veza operacijskog pojačala neće uvesti efekt uskog grla). Nadalje, sveukupnu glasnoću reproduciranog zvuka upravljamo preko sučelja operacijskog sustava računala. Iz istog razloga nema potrebe ni za izlaznim pojačalom. Što se glasnoće tiče, ponovno se oslanjamo na kontrolu preko računala, a što se izlazne snage

tiče, zvučna kartica računala pruža dovoljno snage za bilo koje slušalice koje će se koristiti tijekom rehabilitacije.

Brojem šest označen je dio koji se sastoji od dva ulazna konektora za mikrofone, regulatora ulaznog intenziteta te lepezastih regulatora boje tona - *bass* i *treble.*

Brojem sedam označen je glavni regulator izlazne razine glasnoće, tj. dio uređaja gdje se nalazi stereo izlaz za slušalice te mono izlaz za primjerice zvučnik ili vibracijsku dasku.

3.1. Elektrotehničke komponente uređaja

Elektronički gledano, SUVAG LINGUA VFA L11 sklop sastavljen je od pretpojačala, sumacijskog pojačala s pet kanala čiji kanali služe za modifikaciju spektra govornog signala te izlaznog pojačala. Svaku od tih komponenti moguće je opisati prijenosnom karakteristikom budući da imamo uvid u vrijednosti komponenata.

3.1.1. Pretpojačalo

Frekvencijski raspon pretpojačala seže od 20 Hz do 18 kHz, a ono je odgovorno za ulaznu liniju signala koji dolazi s mikrofona s fantomskim napajanjem (48V). Sveukupno postoje dvije mikrofonske linije, jedna za rehabilitatora, druga za rehabilitanta. Glasnoću ulaznog signala kontroliramo kliznim potenciometrom, dok boju kontroliramo dvopojasnim aktivnim filtrom s mogućnošću biranja nagiba prijenosne karakteristike. Pretpojačalom moramo osigurati ispravne uvjete ulaznog signala za rad uređaja.

3.1.2. Sumacijsko pojačalo s pet kanala

Sumacijsko pojačalo sastoji se od četiri aktivna filtra i jednog direktnog kanala, odnosno pet potenciometara kojima se određuje intenzitet izlaznog zvuka.

Niz aktivnih filtara započinjemo niskopropusnim filtrom modularne granične frekvencije s četiri mogućnosti odabira frekvencije - od 125 Hz do 355 Hz.

Niskopropusni filtar odgovoran je za prilagodbu niskofrekvencijskog pojasa signala ovisno o tome je li uslijed redundantnosti frekvencijskih komponenata potrebna redukcija ili je potrebna kompenzacija radi oštećenja slušnog područja osobe koja se rehabilitira.

Prvi i drugi pojasnopropusni filtar od velike su važnosti za proces verbotonalne metode jer se upravo u njihovom radnom području događa najviše željenih promjena signala kako bi postigli željeni rezultat. Samim time, moraju biti najfleksibilniji i najefikasniji, no, potrebno je uzeti u obzir i omjer fleksibilnosti filtra i kompleksnosti njegovog sučelja. S obzirom na to da se radne točke filtara namještaju ovisno o predefiniranim postavkama prema specifikacijama verbotonalne metode, i same vrijednosti parametara filtra imaju određene predefinirane točke rada.

Prvi pojasnopropusni filtar aktivan je filtar realiziran operacijskim pojačalom u spoju povratne veze s preklopnim poljem. Upravljajući preklopnim poljem, središnja frekvencija može se podešavati na jednu od šest ponuđenih predefiniranih vrijednosti u rasponu od 180 Hz do 1 kHz.

Koristeći šest vrijednosti osigurali smo optimalni kompromis između fleksibilnosti pojasnog propusta i jednostavnosti njegove realizacije. Postavlja se pitanje hoće li biti potrebno i u programskoj inačici postaviti predefinirane razine graničnih frekvencija budući da je računalni program apsolutno fleksibilan, no radi potrebe rehabilitatora koji su već upoznati s načinom rada i parametrima verbotonalne metode, bilo bi poželjno ostaviti jednake funkcionalnosti. Time se za rehabilitatora osigurava bezbolan prijelaz s analognog uređaja na novo programsko rješenje, uz minimalnu prilagodbu.

Preko preklopnog polja povratne veze operacijskog pojačala imamo mogućnost birati između strmine gušenja frekvencijske karakteristike u iznosu od 30 dB/oktavi (100 dB/dekadi) i 60 dB/oktavi (200 dB/dekadi). Podešavanje razine pojačanja, tj. gušenja filtra moguće je preko kliznog potenciometra koji upravlja pojačanjem filtra u području propuštanja. Raspon pojačanja kreće se od maksimalnog gušenja do jediničnog pojačanja.

Drugi pojasnopropusni filtar također je aktivan filtar realiziran operacijskim pojačalom u spoju povratne veze s preklopnim poljem. Jednako kao i prvi filtar, i drugi pojasnopropusni filtar ima mogućnosti izbora šest središnjih frekvencija. Raspon frekvencije iznosi od 1,4 do 8 kHz. Također, moguće je namjestiti nagib karakteristike na 30 dB/oktavi ili 60 dB/oktavi, jednako kao i kod prvog pojasnopropusnog filtra.

Za rehabilitaciju određenih glasova, uz korištenje pojasnopropusnog filtra, rehabilitator mora namjestiti i klizač RSP SHIFT kako bi postigao optimalnu oktavu glasa i slušnog polja. RSP SHIFT seže od -20 do 20 %.

Svaki od navedenih filtra u sumacijskom pojačalu ima svoj potenciometar za regulaciju pojačanja te se uz njih nalazi i još jedan, potenciometar za direktni kanal. Raspon klizača iznosi od -60 do 0 dB. Ukoliko je potenciometar namješten na nulu, razina izlaznog signala odgovarat će razini ulaznog signala. Namjestimo li potenciometar na -30 dB, izlazna razina bit će 30 dB manja od ulazne, a namjestimo li ga na -60 dB, dolazimo do maksimalnog gušenja čime se utjecaj filtra uklanja iz radnog lanca.

3.1.3. Izlazno pojačalo

Izlazno pojačalo sastoji se od dva preklopna potenciometra s deset fiksnih položaja/razina signala te položajem za isključivanje, koji služe kao regulatori izlazne razine glasnoće, te dva izlaza za slušalice i/ili zvučnik, vibracijsku dasku itd.

4. SUVAG LINGUA emulator

Kako bismo uspješno prenijeli analogni uređaj u digitalni oblik, potrebna je doza modernizacije istog. Sam proces transformacije analognog uređaja u digitalni, a potom i programski oblik, zahtijeva određene kompenzacije te funkcionalne promjene. To je posljedica ograničenja tehnologija mješovite obrade signala.

U ovom programskom rješenju, težnja je stavljena ka tome da implementacijska tehnologija bude što jednostavnija i stabilnija te vjerodostojnog rezultata obrade. Naglasak je stavljen na prijenos bitnih značajki čiju važnost slažemo prioritetno prema samoj funkciji uređaja i načelima verbotonalne metode. Postavlja se pitanje što određeni sklop unutar uređaja želi postići, ali ne nužno i na koji način to isto želi postići.

Temeljni dio sklopovlja SUVAG LINGUAe čine četiri modularna filtra: visokopropusni, niskopropusni te dva pojasnopropusna filtra. Budući da isključivo jedan filtar vrši funkciju u vremenskom periodu (tj. za vrijeme rada uređaja uključen je samo jedan filtar), dva analogna pojasnopropusna filtra daju se svesti na jedan digitalni pojasnopropusni filtar. Poteškoća zbog rada samo jednog filtra u vremenskom periodu otklonjena je iskorištavanjem parametra *quality* pojasnopropusnog filtra. Kao dio programa mogu se spremati i učitati predefinirana stanja svih parametara prema optimalama glasova. U slučaju predefiniranog stanja nazala (koji su jedini konsonanti koji ponekad zahtijevaju kombinaciju dva filtra), niskopropusni i visokopropusni filtar čine primarni pojas, dok je pojasnopropusni filtar predefinirano namješten na oktavno područje sekundarnog pojasa. Time se iskorištava mogućnost uključivanja više filtera u isto vrijeme. U većini slučajeva, pojasnopropusni filtar bit će isključen i *quality* mu se neće namještati, a za preostalu manjinu rehabilitator će imati na raspolaganju predefinirana stanja, čime mu se olakšava korištenje uređaja.

Nadalje, prijenosna karakteristika pojasnopropusnog filtra može se postići i kaskadom visokopropusnog i niskopropusnog filtra. Može se zaključiti kako su za proizvoljno oblikovanje digitalnog pojasnopropusnog filtra dovoljna samo dva filtra (niskopropusni i visokopropusni), naspram analognog slučaja u kojem su nam potrebna četiri filtra kako bi obavili isti rad.

Upotrebom samo niskopropusnog i visokopropusnog filtra uklonjena je prva redundancija uređaja u digitalnom obliku. Druga redundancija funkcija je RSP SHIFT-a. Naime, RSP SHIFT

odgovoran je za pomak spektra po frekvencijskoj osi bez utjecaja na širinu pojasa. Tu istu transformaciju DSP rješava proizvoljnim i preciznim namještanjem parametara. To znači da rehabilitator više ne mora odrediti jednu od nominalnih središnjih frekvencija te namještati RSP SHIFT klizač, već je dovoljno direktno namjestiti željenu središnju frekvenciju. Nominalne središnje frekvencije posljedica su preklopne kapacitivne mreže povratne veze sklopovlja koji upravljaju prijenosnim karakteristikama filtara. Postoji konačan broj nominalnih središnjih frekvencija, čiji se iznos transverzalno pomiče uz pomoć RSP SHIFTa, no digitalna izvedba sadrži skup središnjih frekvencija dovoljno širok za direktno proizvoljno biranje.

Kao što je ranije napomenuto, nema potrebe za realizacijom ulaznog i izlaznog pojačala, budući da je isto već osigurano zvučnom karticom računala. Za funkcioniranje navedenih uređaja dovoljno je samo komunicirati preko API sučelja.

Ako se sagleda korisničko sučelje analognog uređaja te shematski prikaz frekvencijskih karakteristika filtara koje uključuje, vidljivo je kako grafički prikaz (pri korištenju uređaja) služi kao dodatna ispomoć u vizualizaciji prijenosne karakteristike filtara. Ograničenje analogne tehnologije (bez uporabe spektrograma i osciloskopa) dovelo je do ovako jednostavnog rješenja. Međutim, implementacija spektrograma i osciloskopa digitalnim programom ne predstavlja dodatno neisplativo ulaganje budući da već postoji modul unutar JUCE razvojnog sučelja kojeg se relativno lagano inicijalizira u sučelje. Korisničko sučelje programa bit će obogaćeno prikazom prijenosne karakteristike filtra te spektralnim uzorkom promatranog audio signala u realnom vremenu. Time se rehabilitatoru, ali i rehabilitantu omogućuje kvalitetniji i lakši prikaz fizike govora i slušanja.

U duhu verbotonalne metode i stvaranja novih pristupa i načina rehabilitacije slušanja i govora, kao i korištenja senzoričkih komponenti ljudskog tijela, smatram kako je dodavanje dodatne vizualne karakteristike od velike važnosti.

4.1. Audio DSP

Pirkle (2019) tvrdi kako je vrijedno promotriti način funkcioniranja hardvera računalnog sustava koji vrši zadaću programa. Bitno je znati na koji način pribavlja podatke, rekonstruira audiosignal te kako izgleda numerička reprezentacija u pozadini audiosignala. Tek tada mogu se definirati ispitni signali i jednostavni procesni blokovi.

4.1.1. Analiza audio signala

Audiosignal po svojoj je prirodi kontinuiran u vremenu, i njegov spektar poprima kontinuiran oblik prema teoremu Fourierove transformacije. Fourierova transformacija pretpostavlja antikauzalan, beskonačno velik vremenski prozor. Realan slučaj ne može postići te uvjete jer je zvuk konačan u trajanju. Tehnika kojom će se premostiti takav problem jest ponavljajući vremenski prozor koji je konačnog trajanja, ali beskonačnog broja ponavljanja. Nadalje, Spanias, Painter et. Atti (2007.) pišu: "Kako bi se izgladili prijelazi između vremenskih prozora i umanjio efekt curenja spektra, signal se često modificira postupnim smanjenjem amplitude prema krajevima vremenskog prozora. Pritom se koriste definirane funkcije vremenskih prozora poput Hammingovog, Bartlettovog i trapezoidnog".

Korištenjem točno određenih oblika vremenskih prozora, uklanjaju se diskontinuiteti na rubovima ponavljanja. Time se zaustavlja curenje spektra signala van optimalnih granica.

Ako se audiosignal promatra u diskretnom obliku vremena, potrebno je držati se načela teorema uzorkovanja koji direktno slijedi Nyquistov teorem uzorkovanja: "Ovaj teorem tvrdi da signal ograničene širine pojasa B rad/s može biti jedinstveno prikazan kao niz uzorkovanih vrijednosti raspoređenih na jednako udaljenim periodičnim intervalima duljine trajanja π/B sekundi. Drugim riječima, ako označimo period uzorkovanja kao T_s, tada teorem uzorkovanja navodi da je $T_s = \pi/B$. U frekvencijskoj domeni, i uz definiranu frekvenciju uzorkovanja, on iznosi $\omega_s = 2\pi f_s = 2\pi/T_s$. (Spanias, Painter et. Atti, 2007)."

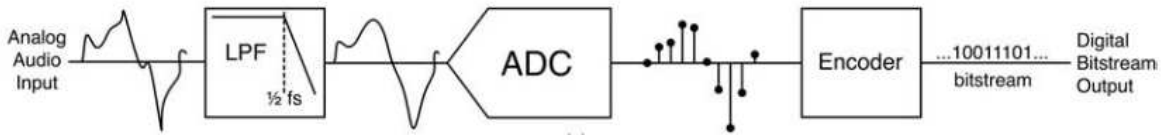
Potrebno je koristiti frekvenciju uzorkovanja koja je minimalno dvostruko veća od maksimalne frekvencije promatranog signala. Budući da je gornja granična frekvencija ljudskog sluha (kod mladih, zdravih ljudi) 20 kHz, standardizirana frekvencija uzorkovanja audiosignala iznosi 44,1 kHz kako bi zadovoljila Nyquistov teorem uz određenu rezervu. U suprotnom, javlja se mogućnost aliasinga.

"Osnovna teorija digitalne obrade zvuka otkriva da je broj kvantizacijskih razina dostupnih za kodiranje $q = 2^N$, gdje je N broj bitova kodiranja signala. Dakle, 8-bitni sustav može kodirati 2^8 vrijednosti ili 256 kvantizacijskih razina. 16-bitni sustav može kodirati 65536 različitih razina." (Pirkle, 2019.)

Kvantizacija audio signala izvršit će se uz 16 bita po uzorku. Analogni signal na ulazu pretvara se u sekvencijski tok binarnih podataka na izlazu, a Pohlmann (2010.) tvrdi kako: "Kvantizacija je tehnika mjerenja analognog audio događaja kako bi se prema trenutnoj vrijednosti signala oblikovala numerička vrijednost koja ju predstavlja. Digitalni sustav koristi binarni brojevni

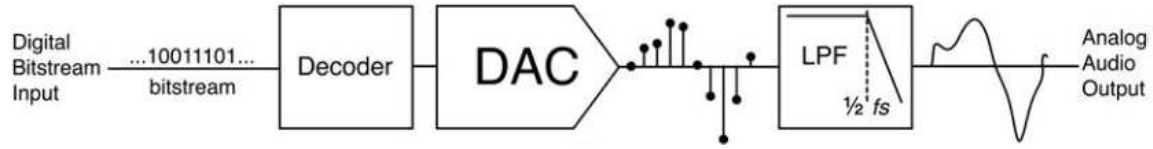
sustav. Broj mogućih vrijednosti određen je duljinom binarnog podatkovnog niza, tj. brojem bitova dostupnih za formiranje prikaza."

Kaskada sastavljena od niskopropusnog filtra, sklopa za uzorkovanje i zadržavanje te analogno-digitalnog pretvarača odgovorna je za vjerodostojno prikazivanje digitalnog toka kvantiziranih podataka prema analognom ulaznom kontinuiranom signalu. U posljednjem koraku, koder pretvara tok kvantiziranih podataka u tok binarnih podataka na izlazu iz sklopovlja. Binarni tok podataka obrađuje se procesnim slojem i/ili zapisuje se u memorijske registre računala. Cijeli proces opisan je Slikom 4.1.



Slika 4.1. Proces pretvorbe analognog signala u binarni tok podataka (Pirkle, 2019.)

Pretvorba diskretnog digitalnog signala u analogni, kontinuirani oblik odvija se dekoderom, digitalno-analognim pretvaračem i niskopropusnim filtrom. Slikom 4.2. opisan je proces obrnut od prethodno navedenog čija je zadaća binarni tok podataka prevesti u kontinuirani analogni signal.



Slika 4.2. Proces pretvorbe binarnog toka podataka u kontinuirani analogni signal koji prenosi zvučni podražaj (Pirkle, 2019.)

Podaci o audiosignalima mogu se pohraniti na više načina: "Proces kodiranja konstruiran je na sljedeći način: skup osnovnih poruka a(i) i pripadajućih vjerojatnosti pojavljivanja za svaku od osnovnih poruka p(a(i)) navedene su prema padajućem redoslijedu vjerojatnosti. Ova lista zatim biva podijeljena tako da se formiraju dvije grupe s gotovo jednakim ukupnim vjerojatnostima (koliko je to moguće). Svaka poruka u prvoj grupi dobiva 0 kao prvu znamenku

svoje kodne riječi; poruke u drugoj polovici imaju kodne riječi koje započinju s 1. Svaka od ovih grupa zatim je podijeljena isto prema prethodno navedenom kriteriju i dodaju se dodatne znamenke koda. Postupak se nastavlja dok svaki podskup ne sadrži samo jednu poruku." (ics.uci.edu, 2023).

Na raspolaganju za spomenuto stoje tipovi podataka cjelobrojnih brojeva (paritetni i neparitetni) te tipovi podataka realnih brojeva (s pomičnom i fiksnom točkom). Ukoliko na umu imamo činjenicu kako je veoma korisno odrediti jednu kvantizacijsku razinu koja će predstavljati digitalnu nulu, utoliko dolazimo do problema neparnog broja kvantnih razina za preostale vrijednosti. Iz tog razloga često na raspolaganju imamo jednu razinu više za negativne vrijednosti u odnosu na one pozitivne.

4.1.2. Algoritmi obrade podataka

Algoritam obrade podataka predstavlja skup naredbi modulacije ulaznih podataka u izlazni tok podataka. Kako bi se postigla obrada u realnom vremenu, algoritam mora biti u stanju prihvatiti podatke, obraditi ih i pretvoriti u izlazni tok podataka prije nego što dođe sljedeći uzorak podataka. U suprotnom, dolazi do pojave *flicker* šuma i pucanja vremenske trake. Iz navedenih razloga, ograničenje svakog digitalnog uređaja proizlazi iz restrikcije vremena potrebnog za obradu jednog uzorka ulaznih podataka.

Algoritam računa konvoluciju ulaznog signala i odziva prijenosne karakteristike na jedinični skok: "Algoritmi koriste položaj trenutnog ulaznog uzorka kao svoj referentni vremenski okvir i postavljaju periodičan razmak prema vrijednosti tog uzorka. U sljedećem uzorku, sve se ponovno reorganizira u odnosu na (novi) trenutni ulazni uzorak." (Pirkle, 2019).

Iako ovaj način pohrane pozicije zvuči kompleksno, zapravo je dobra metoda očuvanja vremenskog slijeda uzoraka i jasne definicije prijenosne karakteristike.

Jedan od osnovnih procesnih blokova algoritma jest vremensko kašnjenje signala. U analognim tehnologijama vremensko kašnjenje može se realizirati kapacitivnim i induktivnim elementima, čija je zadaća utjecati na fazno kašnjenje. U digitalnim tehnologijama, dovoljno je matematički obraditi signal prema definiciji algoritma. Uvođenje operatora kašnjenja u diskretnoj frekvencijskoj domeni, čiji eksponent određuje iznos kašnjenja, na jednostavan način dopušta implementaciju kašnjenja proizvoljnog iznosa. Obradom signala u realnom vremenu nije moguće postići negativno kašnjenje, no to nije slučaj za već pohranjene audiosignale.

Algoritam kašnjenja slijedi nekoliko ključnih pravila:

- Svaki put kad se u registar kašnjenja zapiše novi uzorak, potrebno je osloboditi registar od prethodno zapisanog uzorka
- Prethodni se uzorak može koristiti za daljnju obradu ili može biti izbrisan
- Sklop za kašnjenje može se nizati serijski u proizvoljnu količinu sklopova kako bi se dobio željeni iznos kašnjenja

Sljedeći procesni blok jest množenje signala vektorom koji sadrži skalarne koeficijente za svaki od parametara: "Radi se o jednostavnom matematičkom operatoru koji množi ulazne parametre sa skalarnim koeficijentima. Operator množenja koristi se u gotovo svakom DSP algoritmu. Izlazi su, jednostavno, skalirane verzije ulaza." (Pirkle, 2019).

Analogna obrada signala zahtjeva kompleksne sklopove za realizaciju pojačanja ili prigušenja. S druge strane, u digitalnoj obradi signala, skaliranje je izvedivo jednostavnim slijednim množenjem. Naposljetku, zbrajanje i oduzimanje, također je moguće realizirati jednostavnim slijednim zbrajanjem, tj. množenjem. Zbrajanje u analognoj tehnologiji je jednostavno do razine spajanja dva voda signala, no pri oduzimanju postoji potreba za jednim inverterom signala. S pomoću jednog zbrajala, dva množila i sklopa za kašnjenje, moguće je realizirati procesni blok u povratnoj vezi.

4.2. Razvojno sučelje audio programa

Postoje tri standarda za izradu dodataka (*engl. plugin*) za obradu audiosignala koji se koriste u virtualnim studijskim okruženjima (*engl*. Digital Audio Workstation ili DAW): *Avid Audio eXtension (AAX), Audio Unit (AU)* te *Virtual Studio Technology (VST)*. Sva tri standarda jednaka su u temeljnom načinu funkcioniranja i konstituciji API komunikacijskog sučelja, no postoje određene razlike u strukturi implementacije procesnih blokova unutar koda. Za potrebe ovog programa, realizirani program bit će izvezen u VST3 verziju dodatka, AU verziju te samostalnu verziju s pomoću JUCE razvojnog sučelja koje ima mogućnost izvoza u željeni standard prema strukturi dijeljenog koda programa.

Pirkle (2019.) opisuje ulogu API pokazivača na temeljnu klasu: "Plugin je, naravno, naš mali dragulj za obradu signala koji će učiniti nešto zanimljivo sa zvukom. Pakiran je kao C++ objekt, a domaćin je DAW ili drugi softver, koji učitava *plugin*, dobiva jedan pokazivač na svoju osnovnu baznu klasu i komunicira s njim."

Pluginov API nalazi se unutar pakiranja *Software Development Kita* (SDK) što nije ništa drugo nego skup C++ datoteka i drugih komponenata prethodno deklariranih u svrhu pojednostavnjivanja razvoja novih audio programa. Svako razvojno okruženje ima pripadajući SDK pa tako i razvojno okruženje JUCE koje je korišteno za potrebe ovog programa. Zadaća SDK je sastaviti pravila za kompilaciju API sučelja na temelju strukture opisane C++ datotekama koda. JUCE SDK sadrži razne module koji imaju unaprijed deklariranu strukturu ponašanja koda i način kompiliranja osnovnog dijeljenog koda u različite standarde *plugina.*

Svaki *plugin* pakiran je kao biblioteka dinamičnih poveznica (*engl*. Dynamic-Link Library ili DLL). Razlika DLLa i „običnih" biblioteka jest u tome što se program povezuje s običnim bibliotekama u trenutku kompilacije, dok se s DLL bibliotekama povezuje u trenutku izvođenja. Prednost korištenja dinamičnog ili eksplicitnog povezivanja biblioteka jest neovisnost koda spram vanjskih metoda, što dovodi do veće zalihosti i stabilnosti koda. "Prednost je u tome što, ako se pronađe greška u biblioteci, potrebno je samo redistribuirati novu kompiliranu DLL datoteku umjesto ponovnog rekompajliranja izvršne datoteke s ispravljenom statičkom bibliotekom. Druga prednost je u tome što način na koji je ovaj sustav postavljen - domaćin koji se povezuje s komponentom za vrijeme izvođenja - savršeno funkcionira kao način proširenja funkcionalnosti domaćina bez da on zna bilo što o komponenti prilikom kompilacije. Također, predstavlja idealan način postavljanja sustava za obradu *pluginova*. Domaćin postaje DAW, dok *plugin* postaje DLL." (Pirkle, 2019).

Postoji više etapa implementacije *plugin* programa u DAW: faza validacije, faza učitavanja, faza obrade te faza rasterećivanja. Faza validacije testira zadovoljava li program strukturu API sučelja s obzirom na okruženje i verziju poveznica. Faza učitavanja odgovorna je za privlačenje programa u adresni blok procesnog dijela registra s pomoću pokazivača te se ona događa isključivo ako program uspješno ostvari fazu validacije. Faza obrade strukturno se ponaša kao petlja, u smislu da se beskonačno puno puta ponavlja od početka do kraja. Fazom obrade transformira se protok ulaznih podataka preko DSPa u protok izlaznih obrađenih audio podataka koji se zatim šalje na API zvučne kartice računala. Naposljetku, potrebno je terminirati program i osloboditi alociranu memoriju preko faze rasterećivanja.

4.3. Definicija parametarskog sučelja

Neizostavan dio svakog audio programskog formata jest korisničko sučelje. Međutim, jedan od najbitnijih dijelova audio programa jest parametarsko sučelje, a ono je odgovorno za

povezivanje procesnih radnji i pripadajućih elemenata prikaza parametara na korisničkom sučelju. Pirkle (2019.) dalje tvrdi: "Postoje dva različita tipa parametara: kontinuirani parametri i parametri u obliku liste stringova. Kontinuirani parametri obično su povezani s rotacijskim ili linearnim potenciometrima koji mogu prenositi kontinuirane vrijednosti. To uključuje podatkovne tipove float, double i int. Parametri u obliku liste stringova korisniku predstavljaju listu nizova teksta među kojima može birati, a upravljanje korisničkim sučeljem (GUI) može biti jednostavno (padajući izbornik) ili složeno (prikaz prekidača s grafičkim simbolima)."

Svaki od parametara posjeduje skup atributa koji ga definiraju. Atributi poput imena, mjerne jedinice, tipa parametra itd. prenose informacije API sučelju za ispravan rad i za deklaraciju spojne mreže procesa unutar DAW programa.

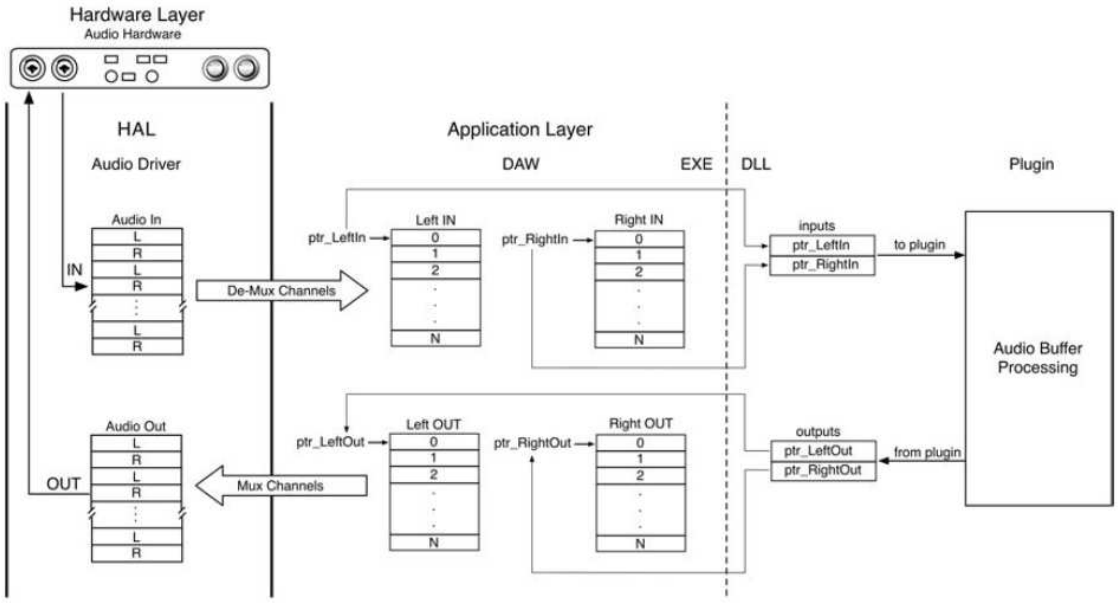
Parametarsko sučelje sadrži i informacije o ulaznim i izlaznim kanalima te može prepoznati mono, stereo, 5.1 stereo i 7.1 stereo ulaznu ili izlaznu kombinaciju kanala. Svaku od kombinacija broja kanala ulaza i izlaza moguće je realizirati kroz parametarsko sučelje, potrebno ih je samo ispravno deklarirati kako ne bi došlo do pogrešne prespojne mreže ulaza i izlaza. Još jedna važna informacija koju parametarsko sučelje ima zadaću prenijeti jest brzina uzorkovanja. *Plugin* mora biti veće ili jednake brzine uzorkovanja od samog DAW procesa, kako ne bi došlo do diskontinuiranih smetnji tijekom procesa. "U svim APIevima postoji barem jedan mehanizam koji omogućuje *pluginu* da učita trenutnu brzinu uzorkovanja za korištenje u svojim izračunima. U svim slučajevima, *plugin* dohvaća brzinu uzorkovanja putem poziva funkcije ili članske varijable APIa." (Pirkle, 2019). Iz navedenog razloga, iznimno je važno predvidjeti memorijsku alokaciju i pokazivač za informaciju o brzini uzorkovanja.

4.4. Obrada podataka i priprema za *stream* audio podataka

Svako API sučelje u sebi mora imati funkciju *prior to audio streaming*. Navedena funkcija ima zadaću pripremiti procesni lanac za nadolazeći tok audio podataka (*engl.* audio stream). Ova zadaća općenito obuhvaća sinkroni *reset* algoritma, čišćenje podataka prethodnog lanca iz memorije te resetiranje *buffera* (*hrv.* Procesni blok) i pokazivača. Unutar JUCE razvojnog sučelja ta funkcija dobiva ime *prepareToPlay().* Trebalo bi napomenuti kako treba pripaziti na korištenje povratnih informacija DAW programa unutar procesnog algoritma, a o tome piše i Pirkle (2019.): „Važno je napomenuti: pokušati koristiti prijenosne informacije domaćina (tj. Informacije o prijenosnim stanjima domaćina: reprodukcija, zaustavljanje, ponavljanje itd.) obično bit će problematično, osim ako dodatak izričito zahtijeva informacije o procesu – na

primjer, algoritam uzorka petljanja koji treba znati koliko mjera se petlja i što trenutna mjera uključuje.“

Kakav god analogni audio podatak bio, on se u digitaliziranom obliku predstavlja numeričkim tipom podataka *float* (*floating point*) u rangu od -1.0 do +1.0. Iz tog razloga, bilo kakav oblik podataka unutar C++ koda bit će deklariran kao varijabla tipa *float*. Na taj način, slijedimo standard međusobne komunikacije audio API sučelja. Također, slijedeći standard komunikacije, osigurava se nesmetana komunikacija između digitalnih audio sklopova različitih priroda obrade podataka. Na primjer, sklop za snimanje podataka u realnom vremenu i sklop za obradu podataka spremljenih u memoriju mogu nesmetano međusobno dijeliti podatke jer su njihova API komunikacijska sučelja jednako standardizirana.



Slika 4.3. Struktura procesnog događaja i modula koju su u komunikaciji tijekom procesnog rada (Pirkle, 2019.)

Hardverski dio procesa prikazan je na lijevoj strani Slike 4.3. U sredini slikovnog prilaza nalazi se program, tj. DAW i EXE (*executable file*), a na desnoj strani nalazi se *plugin.* Ovaj program kompiliran je kao VST3, AU te kao samostalni program. To znači da korištenjem VST ili AU inačice programa, referenciramo isključivo *plugin* stranu slikovnog prikaza. U slučaju referenciranja na samostalnu inačicu, ona preuzima i programski i *plugin* dio slikovnog prikaza. Također, radi se o stereo ulazima i izlazima.

Zvučna kartica računala predstavlja hardverski dio sklopa koji je sabirnicom povezan na audio upravljački program (*engl*. *Driver*). Upravljački program je unutar HALa, tj. apstraktnog sloja hardvera. Bez obzira na način na koji zvučna kartica oblikuje podatak o zvuku, upravljački program formatira ga prema definiciji operacijskog sustava računala. Na sličan način formatirani su i podatci podatkovnog zapisa tipa .wav (*waveform audio file format*), što bi značilo da programski dio pristupa zapisanim podatcima i podatcima u stvarnom vremenu na jednak način. Upravljački program šalje podatke programskom sloju preko skupova podataka koje nazivamo *buffer*s. Veličina buffer skupa podataka određena je preko DAW, preference upravljačkog programa ili preference operacijskog sustava. „Zatim DAW preuzima te podatke i razdvaja ih, stavljajući svaki kanal u vlastiti *buffer* pod nazivom ulazni kanalni *buffer*. DAW također priprema (ili rezervira) *buffere* u koje će *plugin* zapisati obrađene audio podatke, nazvane izlazni kanalni *bufferi*" (Pirkle, 2019). DAW je odgovoran za prosljeđivanje podataka (preko pokazivača i informacija o veličini i poziciji) *buffera* prema DLLu. U trenutku kad *Plugin* vrati podatke preko DLLa na način da proslijedi pokazivače na izlazne *buffere,* DAW ponovno obrađuje podatke te ih prosljeđuje upravljačkom programu zvučne kartice, ili ih zapisuje u .wav datoteku.

Postoje dvije metode obrade podataka iz *buffera.* Ukoliko koristimo *buffer processing,* utoliko je potrebno obraditi sve uzorke jednog po jednog *buffera*. Ako obrađujemo po jedan uzorak od svakog *buffera,* potrebno je koristiti *frame processing*. Budući da je izrađeni program implementiran prema načelima ekvalizatora (višekanalni plugin čije kanale možemo obraditi neovisno), odabrana je *buffer processing* metoda. U slučaju korištenja algoritama kojima je potrebna informacija o svakom kanalu u isto vrijeme, koristili bismo *frame processing* (npr. ping-pong kašnjenje, reverb, procesori stereo dinamike…).

4.5. Obrada događaja promjene podataka

Najzahtjevniji dio izrađenog programa odvija se u trenutku kad program paralelno obrađuje audio podatke i istovremeno implementira rezultate promjene parametara: „tijekom rada našeg *plugina* događaju se dvije različite aktivnosti. S jedne strane, DAW šalje audio podatke *pluginu* radi obrade, a s druge strane, korisnik djeluje nad korisničkim sučeljem – ili izvodi automatizaciju ili obavlja oboje – što zahtijeva od *plugina* da periodično rekonfigurira svoj unutarnji rad i parametre algoritma." (Pirkle, 2019.) Baš kao i reakcija na promjenu, svaki podražaj i/ili rad programa odvija se kroz ciklički niz *buffera.* Procesi poput mjerenja razina,

grafičkih prikaza i automatizacijskih procesa moraju se odviti unutar istog perioda trajanja jednog *buffera*. U svakom slučaju, cijeli proces odrađuje se jednim pozivom na glavnu metodu koju možemo podijeliti u tri faze radnje.

1) *Preprocessing* faza radnje: odgovorna je za prijenos promjene parametara, automatizirano ili s korisničkog sučelja te po potrebi mijenja interne varijable
2) *Processing* faza radnje: odgovorna je za izvršavanje DSP algoritama na audio podatke prema uputama novo dobivenih vrijednosti parametara
3) *Postprocessing* faza radnje: odgovorna je za slanje podataka o parametrima natrag na korisničko sučelje te po potrebi generiranje MIDI zapisa

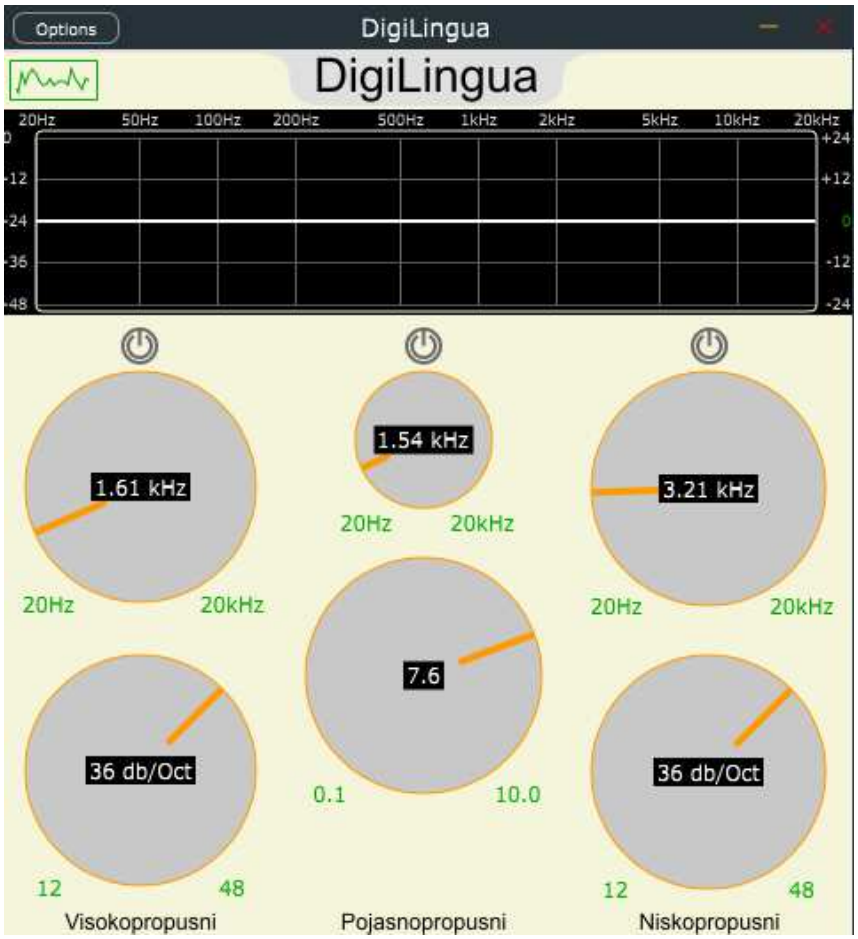
Važno je razlikovati parametre i varijable. Parametri se koriste isključivo u svrhu komunikacije korisničkog sučelja i DSP algoritama, dok su varijable rezervirane samo za grafičke prikaze i interakciju s korisnikom na grafičkom sučelju. Varijable su svojevrsna maska pravih parametara programskog procesa. „Tijekom ciklusa obrade *buffera* (*buffer processing cycle*), čitaju se ulazni parametri, obavlja se obrada zvuka, a zatim se ispisuju izlazni parametri. Izlazni zvuk se šalje na zvučnike ili audio datoteku, dok se izlazni parametri šalju na grafičko sučelje.“ (Pirkle, 2019).

5. Program DigiLINGUA

Za stvaranje navedenog programa, korišteno je JUCE razvojno sučelje iz više razloga, a neki od njih su: sadržavanje modula i biblioteka koji su potrebni za uredan rad programa, inicijalizacija projekta uvelike je olakšana koristeći program *Projucer*, a koristeći program *AudioPluginHost* može se na jednostavan i intuitivan način podesiti API sučelje upravljačkog programa zvučne kartice.

Izvorni kod programa sadrži četiri datoteke, dvije datoteke za sloj obrade podataka i dvije datoteke za sloj oblikovanja korisničkog sučelja. Jedna datoteka sadrži glavni kod sloja programa u kojem su definirani parametri, varijable i metode, dok druga datoteka sadrži *header* kôd sloja programa u kojem su ti isti parametri, varijable i metode deklarirani uz poveznice na potrebne programske biblioteke.

5.1. Korisničko sučelje



Slika 5.1. Korisničko sučelje programa DigiLINGUA

Ako se program promatra sa stajališta korisnika, korisničko sučelje (prikazano Slikom 5.1.) najbitniji je dio programa. Velik naglasak stavljen je na jednostavnost programa za potrebe rehabilitatora. Na prvi pogled program se može podijeliti na četiri sektora:

1) Gornji sektor: sadrži grafički prikaz prijenosne karakteristike uključenog filtra u stvarnom vremenu te frekvencijski spektar snimanog zvuka (također u stvarnom vremenu) čiji prikaz korisnik može uključiti ili isključiti
2) Lijevi sektor: sadrži upravljačko sučelje visokopropusnog filtra. Varijable na koje korisnik može utjecati su iznos granične frekvencije filtra i strmina gušenja. Korisnik može uključiti ili isključiti filtar iz procesa
3) Centralni sektor: sadrži upravljačko sučelje pojasnopropusnog filtra, a varijable kojima korisnik može upravljati su centralna frekvencija te faktor kvalitete/pojačanje filtra. Također, korisnik može uključiti ili isključiti filtar iz procesa.
4) Desni sektor: sadrži upravljačko sučelje niskopropusnog filtra, a varijable na koje korisnik može utjecati su granična frekvencija i strmina gušenja filtra. Jednako kao i ranije, filtar se može uključiti ili isključiti iz procesa.

Na prvi je pogled vidljivo kako GUI ne emulira fizički uređaj. U digitalnoj verziji željeno je bilo dobiti da se od kombinacije visokopropusnog i niskopropusnog definiraju proizvoljne karakteristike jednog pojasnopropusnog filtra po cijelom spektru.

Pozicija NP i VP filtra na analognom uređaju određena je činjenicom da se koriste izolirano, ali u digitalnom obliku takav redoslijed otežava korištenje. Korisniku izgleda neintuitivno dovlačiti VP sa lijeve na desnu stranu spektra i NP sa desne na lijevu stranu spektra kako bi stvorio pojasni propust. Za korisnike je jednostavniji ovakav razmještaj jer pomicanjem lijevog rotacijskog klizača pomiču lijevu strminu filtra, a pomicanjem desnog rotacijskog klizača pomiču desnu strminu filtra. U trenutku kad im zatreba samo VP ili samo NP onda im nije vizualno otežavajuće na grafici (jer im se prikazuje samo ta komponenta i mogu intuitivno dovlačiti pojas na ispravnu graničnu frekvenciju). U nekim slučajevima, nema potrebe čak niti za gledanjem grafike, već je moguće manualno namjestiti brojčanu vrijednost granične frekvencije koristeći rotacijski klizač.

Što se PP filtra tiče, on je većinu vremena ugašen i koristi se samo za specifične slučajeve (za koje nema potrebe uvoditi dodatne parametre graničnih frekvencija), dovoljno je kontrolirati *quality* za širinu pojasa (potrebna je mala devijacija koja neće previše utjecati na glasnoću

sekundarnog pojasa). Kontrola *quality* je tu iz razloga što ju je potrebno definirati i inicijalizirati kako bi mogli upravljati njome za te specifične situacije.

Sva tri filtra ugašenog stanja automatski aktiviraju svepropusni filtar, tj. direktni kanal, a svakom promjenom parametara grafički prikaz osvježava prikazano stanje. Pritiskom kursora na tipku *Options* javljaju se četiri ponuđene opcije: ulazak u postavke programa, spremanje seta trenutnih vrijednosti parametara u memoriju računala, učitavanje prethodno spremljenog seta vrijednosti parametara s računala te ponovno postavljanje svih parametara u početno stanje.

Postavke programa nude mogućnosti odabira željenog ulaza i izlaza audio signala. Moguće je utjecati i na brzinu uzorkovanja te veličinu *buffera*, no za potrebe rada ovog programa, nije potrebno mijenjati te postavke.

5.2. Procesni sloj kôda

Kao što je već napomenuto, audio program upravlja DSP procesom parametrima. Potrebno je podatke o parametrima povezati u jedan skup kako bi pristupili podatcima preko pokazivača. Za takve slučajeve služi *Audio Processor Value Tree State* (APVTS). Srećom, biblioteke JUCE razvojnog sučelja koriste interno definirane tipove podataka i metode pomoću kojih implementiramo do sada promatrane teorijske pojmove i procese.

U ovom slučaju, unutar *header* datoteke, deklariramo skup parametara *apvts* pozivom na metodu *createParameterLayout()* koja je dio paketa *AudioProcessorValueTreeState::ParameterLayout* na način koji je prikazan Kôdom 5.1.

```
static juce::AudioProcessorValueTreeState::ParameterLayout
createParameterLayout();

juce::AudioProcessorValueTreeState apvts{
      *this,
      nullptr,
      "Parameters",
      createParameterLayout()
```

```
};
```

Kôd 5.1. (PluginProcessor.h) Deklaracija povezivanja parametara u skup kojem pristupamo pokazivačem

Nakon deklaracije unutar *header* datoteke, potrebno je unutar glavne datoteke inicijalizirati navedeni skup parametara definirajući raspored parametara unutar njega – *layout* kao odgovor na poziv metode *createParameterLayout()*. Kôd 5.2 prikazuje spomenutu inicijalizaciju.

```
juce::AudioProcessorValueTreeState::ParameterLayout
DigiLinguaAudioProcessor::createParameterLayout()
{
      juce::AudioProcessorValueTreeState::ParameterLayout
layout;


      layout.add(
            std::make_unique<juce::AudioParameterFloat>(
                  "LowCut Freq",
                  "LowCut Freq",
                  Juce::NormalisableRange<float>(
                        20.f,
                        20000.f,
                        1.f,
                        0.25f
                  ),
                  20.f
            )
      );
// …
// … dodatni parametri za svaki od filtara …
```

```
// …
        return layout;
}
```

Kôd 5.2. (*PluginProcessor.cpp*) Definicija rasporeda parametara *layout* unutar skupa *apvts*

Svaki parametar ima svoj identifikacijski zapis (*ParameterID &parameterID*), zapis imena (*String &parameterName*), deklariran skup vrijednosti koje parametar može poprimiti (*NormalizableRange<data_type> normalizableRange*) te početnu vrijednost (*data_type defaultValue*). Prikazani dio Kôda 2 opisuje podatke parametra kojim se upravlja graničnom frekvencijom visokopropusnog filtra. Na jednak način definirani su parametri: granična frekvencija niskopropusnog filtra, centralna frekvencija pojasnopropusnog filtra, strmine gušenja filtara, Q faktor pojasnog propusta te sklopke paljenja i gašenja pojedinih filtara i grafičkog prikaza.

JUCE razvojno sučelje unutar svojih modula ima i predefinirani oblik digitalnog IIR filtra. Pozivajući se na njegov nominalan oblik, mogu se stvoriti procesni lanci koji unutar sebe sadrže više kaskadno nanizanih elemenata. Kôd 5.3 opisuje na koji se način stvaraju procesni lanci za proizvoljan oblik filtara te kako prenijeti kontekst u nove procesne lance za lijevi i desni procesni kanal. Nizanjem više nominalnih IIR filtara postižemo mogućnost biranja strmine gušenja uključujući i isključujući slijedne nominalne filtre unutar lanca. Mogućnost biranja strmine dade se iskoristiti za niskopropusni te visokopropusni filtar.

```
using Filter = juce::dsp::IIR::Filter<float>;
using CutFilter = juce::dsp::ProcessorChain<
        Filter,
        Filter,
        Filter,
        Filter,
        Filter
>;
using MonoChain = juce::dsp::ProcessorChain<
```

```
        CutFilter,
        Filter,
        CutFilter
    >;
    MonoChain leftChain, rightChain;
```

Kôd 5.3. (*PluginProcessor.h*) Deklaracija procesnih lanaca za svaki od kanala

Gniježđenjem procesnih lanaca stvoreni su procesni lanci lijevog i desnog kanala. Time je definiran procesni kontekst potreban *bufferu* za ispravan rad. Prije uključivanja, filtre je potrebno pripremiti za rad na način da im se proslijede podatci o DSP algoritmu: broj uzoraka po *bufferu*, broj kanala koje obuhvaća procesni lanac te brzina uzorkovanja.

Unutar metode *prepareToPlay()* koja zastupa funkciju „*prior to audio stream*“ potrebno je definirati skup obilježja procesa *spec* unutar kojeg koristimo pokazivače na potrebne podatke prikazano Kôdom 5.4.

```
juce::dsp::ProcessSpec spec;
spec.maximumBlockSize = samplesPerBlock;
spec.numChannels = 1;
spec.sampleRate = sampleRate;
leftChain.prepare(spec);
rightChain.prepare(spec);
```

Kôd 5.4. (*PluginProcessor.cpp*) Inicijalizacija prosljeđivanja globalnih procesnih specifikacija procesnim lancima

Pozivom na metodu *processBlock()* započinje jedan ciklus obrade podataka *buffera* kroz pre-processing, processing i post-processing etape. Potrebno je dohvatiti podatke o lijevom i desnom kanalu *buffera* i uz pomoć tih podataka stvoriti procesni kontekst koji se prosljeđuje procesnom bloku prema Kôdu 5.5 unutar metode *processBlock()*.

```
juce::dsp::AudioBlock<float> block(buffer);


auto leftBlock = block.getSingleChannelBlock(0);

auto rightBlock = block.getSingleChannelBlock(1);


juce::dsp::ProcessContextReplacing<float>
leftContext(leftBlock);

juce::dsp::ProcessContextReplacing<float>
rightContext(rightBlock);


leftChain.process(leftContext);

rightChain.process(rightContext);
```

Kôd 5.5. (*PluginProcessor.cpp*) Stvaranje procesnog konteksta i prosljeđivanje procesnim lancima

Nakon stvaranja procesnih lanaca i pripadajućeg procesnog konteksta, mogu se i konfigurirati parametri lanca filtra. Parametri koji se obrađuju, a nalaze se u skupu APVTS podataka, inicijaliziraju se kroz strukturu *ChainSettings*. Također, deklarira se pomoćna metoda dohvaćanja APVTS podataka *getChainSettings()*. Objašnjeni postupak prikazan je Kôdom 5.6.

```
struct ChainSettings{

    float bandPassFreq{ 0 },  bandPassQuality{ 1.f };

    // float bandPassGainInDecibels{ 0 };

    float lowCutFreq{ 0 }, highCutFreq{ 0 };


    Slope lowCutSlope{ Slope::Slope_12 }, highCutSlope{
Slope::Slope_12 };


    bool lowCutBypassed{ false }, bandPassBypassed{ false },
highCutBypassed{ false };
```

```
};

ChainSettings
getChainSettings(juce::AudioProcessorValueTreeState& apvts);
```

Kod 5.6. (*PluginProcessor.h*) Inicijalizacija parametara filtarskog lanca *ChainSettings* te deklaracija metode *getChainSettings()*

Nadalje, u glavnom kôdu potrebno je definirati metodu *getChainSettings()* i način na koji povezuje parametre *buffera* i filtarskog lanca prikazanog Kôdom 5.7. Također, potrebno je metodu *prepareToPlay()* modificirati na određeni način, kako bi utjecala na parametre upravljanih filtara. Umjesto direktnog upisa u metodu *prepareToPlay()*, stvara se nova metoda *updateFilters()*. Tu metodu potrebno je deklarirati i u header datoteci procesnog sloja. Navedena metoda sadrži pokazivač na APVTS podatke te pozive na nove metode *updateLowCutFilters()*, *updateHighCutFilters()* i *updateBandPassFilter()* koje modificiraju parametre pojedinih filtara na način prikazan Kôdom 5.8.

```
ChainSettings
getChainSettings(juce::AudioProcessorValueTreeState& apvts){

    ChainSettings settings;

    settings.lowCutFreq = apvts.getRawParameterValue("LowCut
Freq")->load();

    settings.highCutFreq = apvts.getRawParameterValue("HighCut
Freq")->load();

    settings.bandPassFreq =
apvts.getRawParameterValue("BandPass Freq")->load();

    // settings.bandPassGainInDecibels =
apvts.getRawParameterValue("BandPass Gain")->load();

    settings.bandPassQuality =
apvts.getRawParameterValue("BandPass Quality")->load();

    settings.lowCutSlope =
static_cast<Slope>(apvts.getRawParameterValue("LowCut Slope")-
>load());
```

```
    settings.highCutSlope =
static_cast<Slope>(apvts.getRawParameterValue("HighCut Slope")-
>load());

    settings.lowCutBypassed =
apvts.getRawParameterValue("LowCut Bypassed")->load() > 0.5f;

    settings.bandPassBypassed =
apvts.getRawParameterValue("BandPass Bypassed")->load() > 0.5f;

    settings.highCutBypassed =
apvts.getRawParameterValue("HighCut Bypassed")->load() > 0.5f;

    return settings;

}
```

Kôd 5.7. (*PluginProcessor.cpp*) Definicija metode očitavanja vrijednosti parametara *getChainSettings()*

```
void DigiLinguaAudioProcessor::updateFilters(){

    auto chainSettings = getChainSettings(apvts);

    updateLowCutFilters(chainSettings);

    updateBandPassFilter(chainSettings);

    updateHighCutFilters(chainSettings);

}

void updateCoefficients(Coefficients& old, const Coefficients&
replacements){

    *old = *replacements;

}

void DigiLinguaAudioProcessor::updateBandPassFilter(const
ChainSettings& chainSettings){

    auto bandPassCoefficients =
    makeBandPassFilter(chainSettings, getSampleRate());
    leftChain.setBypassed<ChainPositions::BandPass>(chainSettin
    gs.bandPassBypassed);
    rightChain.setBypassed<ChainPositions::BandPass>(chainSetti
    ngs.bandPassBypassed);
```

```
    updateCoefficients(leftChain.get<ChainPositions::BandPass>(
    ).coefficients, bandPassCoefficients);
    updateCoefficients(rightChain.get<ChainPositions::BandPass>
    ().coefficients, bandPassCoefficients);

}

void DigiLinguaAudioProcessor::updateLowCutFilters(const
ChainSettings& chainSettings){

    auto cutCoefficients = makeLowCutFilter(chainSettings,
getSampleRate());

    // ...

    updateCutFilter(leftLowCut, cutCoefficients,
chainSettings.lowCutSlope);

}

void DigiLinguaAudioProcessor::updateHighCutFilters(const
ChainSettings& chainSettings){

    auto highCutCoefficients = makeHighCutFilter(chainSettings,
getSampleRate());

    // ...

    updateCutFilter(rightHighCut, highCutCoefficients,
chainSettings.highCutSlope);

}
```

Kôd 5.8. (*PluginProcessor.cpp*) Metode modifikacija parametara filtara ovisno o strukturi podataka dobivenoj od strane APVTSa

Parametre svakog od filtara mono procesnog lanca potrebno je deklarirati, a onda i inicijalizirati rang njihovih vrijednosti u slučaju numeričkih podataka (npr.

```
settings.lowCutSlope =
static_cast<Slope>(apvts.getRawParameterValue("LowCut Slope")-
>load());

settings.highCutSlope =
static_cast<Slope>(apvts.getRawParameterValue("HighCut Slope")-
>load());
```

```
case Slope_12:
            *leftLowCut.get<0>().coefficients =
*cutCoefficients[0];
            break;
```

) ili u slučaju mogućnosti odabira liste s izbornikom (npr.

```
rightLowCut.setBypassed<0>(true);
rightLowCut.setBypassed<1>(true);
rightLowCut.setBypassed<2>(true);
rightLowCut.setBypassed<3>(true);
```

) kako bi promjene varijabli grafičkog sučelja pobudile promjene točno željenih procesnih lanaca u procesnom bloku.

5.3 Grafički sloj kôda

Obrada grafičkog sloja kôda odvija se u dvije etape:

1. Implementacija učitavanja i upisivanja parametara procesnog sloja programa
2. Stvaranje i modifikacija grafičkih elemenata i njihovog rasporeda na stranici

Implementacija učitavanja i upisivanja parametara odvija se pomoću dvije metode: *getStateInformation()* i *setStateInformation()*. Njihova definicija opisana je Kôdom 5.9. Stanje parametara procesnog bloka pohranjeno je u APVTS-u u obliku predefiniranog podatkovnog tipa *ValueTree* iz kojeg se vrlo lako stanja očitavaju serijskim nizom. Serijski niz omogućava korištenje *Memory Output Streama* koji zapisuje podatke unutar memorije na koju pokazivač *destData* tipa *MemoryBlock* upućuje. Koristeći iste alate, postupak se može ponoviti, ali ovog puta u redoslijedu potrebnom za upisivanje novih vrijednosti parametara, uz provjeru validnosti APVTS-a.

```
void
DigiLinguaAudioProcessor::getStateInformation(juce::MemoryBlock
& destData){
    juce::MemoryOutputStream mos(destData, true);
    apvts.state.writeToStream(mos);
```

```
}

void DigiLinguaAudioProcessor::setStateInformation(const void*
data, int sizeInBytes){

    auto tree = juce::ValueTree::readFromData(data,
sizeInBytes);

    if (tree.isValid()){

        apvts.replaceState(tree);

        updateFilters();

    }

}
```

Kôd 5.9. (*PluginProcessor.cpp*) Postavljanje metoda učitavanja i spremanja vrijednosti parametara procesnog sloja *getStateInformation()* i *setStateInformation()*

Stvaranje i modifikacija grafičkih elemenata i njihovog rasporeda odvija se u datotekama *PluginEditor.h* koja predstavlja header datoteku i *PluginEditor.cpp* koja predstavlja glavni kôd. Header datoteka pohranjuje deklaracije svih grafičkih elemenata poput prekidača, kliznika, teksta, skočnih prozora, itd. Datoteka glavnog kôda sadrži inicijalizaciju parametara za svaki od tih elemenata, pretvarajući ih u komponente. Komponente se mogu dalje povezivati u cjeline, a uz navedeno, može im se i upravljati rasporedom unutar glavnog prozora grafičkog sučelja te svim drugim relevantnim faktorima. Kôd 5.10 prikazuje deklaraciju klase kliznika unutar header datoteke, dok Kôd 5.11 prikazuje inicijalizaciju i upravljanje varijablama klase kliznika unutar glavne datoteke grafičkog sloja.

```
struct RotarySliderWithLabels : juce::Slider{

    RotarySliderWithLabels(juce::RangedAudioParameter& rap,
const juce::String& unitSuffix) :
juce::Slider(juce::Slider::SliderStyle::RotaryHorizontalVertica
lDrag,
juce::Slider::TextEntryBoxPosition::NoTextBox)

{}

);
```

```
RotarySliderWithLabels  bandPassFreqSlider,
                        // bandPassGainSlider,
                        bandPassQualitySlider,
                        lowCutFreqSlider,
                        highCutFreqSlider,
                        lowCutSlopeSlider,
                        highCutSlopeSlider;
```

Kôd 5.10. (*PluginEditor.h*) Deklaracija klase kliznika pomoću kontruktora

```
std::vector<juce::Component*>
DigiLinguaAudioProcessorEditor::getComps(){
    return {
        &bandPassFreqSlider,
        // &bandPassGainSlider,
        &bandPassQualitySlider,
        &lowcutFreqSlider,
        &highcutFreqSlider,
    };
}
```

Kôd 5.11. (*PluginEditor.cpp*) Inicijalizacija određenih kliznika

Pozicioniranje elemenata računa funkcija *resized()* izvršava se uz pomoć metoda pozicioniranja poput *removeFromTop()*, *removeFromBottom()*, *setBounds()*, itd. Povezivanje varijabli kliznika i parametara procesnog bloka izvršava se klasom *Attachment* na način da se za svaki od parametara stvori pripadajući *Attachment*. Kôd 5.12 prikazuje deklaraciju varijabli klase *Attachment* unutar header datoteke, dok Kôd 5.13 prikazuje inicijalizaciju istih varijabli unutar glavnog kôda.

```
Attachment bandPassFreqSliderAttachment,
```

```
        // bandPassGainSliderAttachment,
        bandPassQualitySliderAttachment,
        lowCutFreqSliderAttachment,
        highCutFreqSliderAttachment,
        lowCutSlopeSliderAttachment,
        highCutSlopeSliderAttachment;
```

Kôd 5.12. (*PluginEditor.h*) Deklaracija varijabli klase *Attachment*

```
DigiLinguaAudioProcessorEditor::DigiLinguaAudioProcessorEditor(
DigiLinguaAudioProcessor& p)
    : AudioProcessorEditor(&p), audioProcessor(p),
    bandPassFreqSliderAttachment(audioProcessor.apvts,
"BandPass Freq", bandPassFreqSlider),
    // bandPassGainSliderAttachment(audioProcessor.apvts,
"BandPass Gain", bandPassGainSlider),
    bandPassQualitySliderAttachment(audioProcessor.apvts,
"BandPass Quality", bandPassQualitySlider),
    lowCutFreqSliderAttachment(audioProcessor.apvts, "LowCut
Freq", lowCutFreqSlider),
    highCutFreqSliderAttachment(audioProcessor.apvts, "HighCut
Freq", highCutFreqSlider),
    lowCutSlopeSliderAttachment(audioProcessor.apvts, "LowCut
Slope", lowCutSlopeSlider),
    highCutSlopeSliderAttachment(audioProcessor.apvts, "HighCut
Slope", highCutSlopeSlider),
    lowcutBypassButtonAttachment(audioProcessor.apvts, "LowCut
Bypassed", lowcutBypassButton),
    bandPassBypassButtonAttachment(audioProcessor.apvts,
"BandPass Bypassed", bandPassBypassButton),
    highcutBypassButtonAttachment(audioProcessor.apvts,
"HighCut Bypassed", highcutBypassButton),
```

```
        analyzerEnabledButtonAttachment(audioProcessor.apvts,
    "Analyzer Enabled", analyzerEnabledButton)
```

Kôd 5.13. (*PluginEditor.cpp*) Inicijalizacija varijabli klase *Attachment*

Ostatak kôda, koji opisuje grafičko sučelje, ispisan je u skladu s proizvoljnim izgledom grafičkog sučelja te obuhvaća sve dosad definirane *Attachment*e, kao i *Attachment*e za preostale implementirane komponente (poput prekidača i grafičkog prikaza prijenosne karakteristike filtra).

5.4 Program u praksi

Program je razvijen kao samostalna izvršna datoteka i ne zahtijeva dodatne instalacije. Takav pristup uvelike olakšava proces korištenja programa i proces navikavanja na istog. Pri korištenju programa, postoji mogućnost i za učitavanjem predložaka kontinuiranih optimalnih područja za određeni glas prema podatcima iz tablica znanstvenog rada „Slušanje glasova govora na uskim kontinuiranim i diskontinuiranim frekvencijskim područjima“ (Desnica-Žeravić, 1987.). Mogućnost učitavanja predložaka uklanja potrebu za ručnim namještanjem varijabli grafičkog sučelja, što rehabilitatoru daje na vremenu posvećenom rehabilitaciji, a ne namještanju opreme. Rehabilitator uvijek ima opciju ugoditi prijenosnu karakteristiku filtra prema specifičnim potrebama rehabilitanta te spremiti predložak namještenih vrijednosti. Time se dolazi do još jednog načina preko kojeg rehabilitator štedi na vremenu i, samim time, povećava kvalitetu rehabilitacije.

Uz vizualnu predodžbu utjecaja promjene parametara karakteristike filtra, upravljanje izgledom prijenosne karakteristike filtra lakše je nego što je to bilo prije. Zahvaljujući vizualnoj predodžbi, i rehabilitator i rehabilitant imaju prilike spoznati kako izgleda spektar željenog glasa u stvarnom vremenu, sa ili bez modulacije. Primjerice, glas /Š/ i glas /S/ mimikom lica i auditivnom spoznajom rehabilitantu možda ne predstavljaju dovoljnu razliku, no na grafičkom prikazu, jasno se vidi pomak energije zvuka iz područja 1,6 – 3,2 kHz (za glas /Š/) u područje 6,4 – 8,2 kHz (za glas /S/). Također, jedina razlika glasova /S/ i /Z/ jest stanje uključenosti glasnica. U slučaju glasa /S/, glasnice su opuštene i ne proizvode zvuk, a s druge strane, u slučaju glasa /Z/ glasnice su napete, vibriraju i proizvode referentni ton govornog sustava. Mimikom lica i auditivnom spoznajom nije uvijek jednostavno primijetiti razliku između tvorbe ta dva glasa, no na grafičkom prikazu jasno se vidi kako, u trenutku kad glas /S/ postaje /Z/, u niskofrekvencijskom spektru zvuka javlja se nagli porast energije. Postojanje ili nedostatak niskofrekvencijskog sadržaja olakšava spoznavanje razlike tih dvaju glasova.

DigiLINGUA obnaša jednaku funkciju kao i SUVAG LINGUA VFA L11, samo što to postiže na drugi, moderniji način. Razvijena je za operacijske sustave „Windows“ i „MacOS“. Valja još napomenuti kako tijekom korištenja programa treba pripaziti na kašnjenje zvuka u stvarnom vremenu. Preporučuje se koristiti slušalice i mikrofon koji su fizičkim ožičenjem spojeni sa zvučnom karticom računala. Ne preporučuje se koristiti Bluetooth protokol jer Bluetooth unosi dodatno kašnjenje u proces.

6. Zaključak

Cilj ovog diplomskog rada bio je ostvariti programsku emulaciju rada analognog elektroničkog uređaja SUVAG LINGUA VFA L11 kao alata koji se koristi u svrhu rehabilitacije poremećaja slušanja i govora, kao dio verbotonalne metode. Motivacija rada jest modernizacija samog rehabilitacijskog postupka te povećanje dostupnosti alata za provedbu terapije potencijalnim korisnicima njegovom implementacijom na računalu Verbotonalna metoda ima mnogobrojne primjene u rehabilitaciji, a njezini najvažniji parametri su: vrijeme, frekvencija, intenzitet, tijelo u smislu tjelesne vodljivosti ili koštane vodljivosti, napetost i pauza. Upotrebom SUVAG LINGUAe priprema se i prilagođava zvučni signal te se njime dalje barata koristeći tehnike verbotonalne metode. SUVAG LINGUA radi na principu električnog filtriranja signala kako bi eliminirala redundantne spektralne frekvencije zvučnog uzorka.

Osnovni zahtjev na programski izvedeni emulator jest vjerodostojna obrada zvučnog signala jednaka onoj koja se ostvaruje uređajem SUVAG LINGUA. Na početku je bilo potrebno analizirati rad analognog uređaja te promotriti ciljeve i motivaciju iza određenih specifikacija uređaja. Nadalje, iz dobivenih informacija, bilo je potrebno sintetizirati digitalni audio dodatak (*plugin*) koji postiže rad jednak analognom uređaju.

Kako bi se sintetizirao digitalni audio *plugin*, kroz ovaj rad promotreni su i način rada pretvorbe analognog signala u kodirani binaran tok podataka, procese obrade digitalnih audiosignala, DSP te funkcionalnosti korisničkog sučelja.

Program odgovoran za emulaciju razvijen je u C++ programskom jeziku uz JUCE razvojno sučelje. JUCE sučelje reflektira pravila DSPa kroz sintaksu i ima bogatu količinu modula koji olakšavaju razvoj audio *plugina.* Korisničko sučelje programskog rješenja *plugina* modernizirano je u odnosu na ono analognog uređaja te je izvedeno na način koji rehabilitatorima kao korisnicima omogućava jednostavan prijelaz s analognog uređaja SUVAG LINGUA na ovdje prikazano programsko rješenje, uz minimalnu potrebnu prilagodbu.

Literatura

[1] Guberina, P. *Metodologija verbotonalnog sistema*, Defektologija, broj 2, Sveučilište u Zagrebu Filozofski fakultet (1996)

[2] Guberina P., Asp C. *Verbotonalna metoda u rehabilitaciji osoba sa smetnjama u komunikaciji*. 1. izdanje. Zagreb: Poliklinika SUVAG, 2002.

[3] Guberina P., Crnković V., Jurjević-Grkinić I. *Govor i čovjek: verbotonalni sistem*. 1. izdanje. Zagreb: ArtTresor naklada, 2010.

[4] Vuletić, B. *Gramatika govora.* Zagreb: Grafički zavod Hrvatske, 1980.

[5] Jakopec, D. *Fonetska korekcija izgovora po verbotonalnoj metodi u ovladavanju inojezičnim hrvatskim na primjeru izvorne govornice španjolskoga jezika.* Diplomski rad. Sveučilište u Zagrebu Filozofski fakultet, 2018.

[6] Donald Bren School of Information & Computer Sciences, *Data Compression.* Poveznica: https://ics.uci.edu/~dan/pubs/DC-Sec3.html; pristupljeno 2. studenog 2023.

[7] Pirkle, C. W. *Designing audio effect plugins in C++*. 2. izdanje. New York: Routledge, 2019.

[8] Spanias, A., Painter, T., Atti, V. *Audio signal processing and coding*. New Jersey: John Wiley & Sons, Inc., Hoboken, 2007.

[9] Pohlmann, K.C. *Principles of Digital Audio.* 6. izdanje. New York: McGraw Hill, 2010.

[10] Desnica-Žerjavić, N. *Slušanje glasova govora na uskim kontinuiranim i diskontinuiranim frekvencijskim produčjima,* Govor, broj 4, Zagreb (1987), str. 19-34.

Sažetak

Analiza i obrada audiosignala nalazi mnogobrojne primjene u raznovrsnim tehnologijama koje su u uporabi u suvremenom svijetu i društvu. Jedno od područja primjene jest i rehabilitacija poremećaja slušanja i govora. U začecima elektrotehnike i elektronike, a prije pojave široko dostupnih računalnih sustava, analiza i obrada audiosignala oslanjala se na namjenski razvijene i proizvedene elektroničke uređaje. U današnjem se svijetu rad tih uređaja može u potpunosti oponašati odgovarajućim ekvivalentnim programskim rješenjima. Kroz ovaj rad potrebno je opisati metode rehabilitacije poremećaja slušanja i govora. Nadalje, potrebno je analizirati rad elektroničkog uređaja pod nazivom SUVAG LINGUA koji se uspješno koristi upravo u ovu svrhu te opisati njegove glavne značajke i tehničke specifikacije, kao i korisničko sučelje za upravljanje uređajem. Na temelju ovih saznanja potrebno je izraditi programsko rješenje uporabom programskog jezika C++ i razvojnog okvira JUCE koje će u potpunosti i s punom funkcionalnošću oponašati rad uređaja SUVAG LINGUA. Potrebno je izraditi i odgovarajuće grafičko korisničko sučelje koje će biti preslika onoga na polaznom elektroničkom uređaju te će korisniku pri radu omogućiti jednostavan prijelaz s elektroničkog uređaja na programsko rješenje. Prednost ovakvog rješenja ogleda se u proširenju i pojednostavljenju primjene jer je za provedbu rehabilitacije slušanja i govora dovoljno posjedovati osobno računalo s instaliranim programskim rješenjem umjesto fizički izvedenog elektroničkog uređaja.

Summary

Analysis and processing of audiosignals is one of the most dispersed technologies in the present-day society. One of many areas of application is speech and language therapy. In the early days of electronics, especially without computers, audio signal processing was based solely on dedicated hardware. With the state of technology today, we can recreate the behaviour of the dedicated hardware with simple software tools. This work will describe the verbotonal methods used in speech and language therapy, analyse the behaviour of an electronic device called "SUVAG LINGUA" which is used for this specific purpose, and describe its inner mechanisms, functions and user interface. Based on this information, we will develop a software solution using C++ programming language and JUCE developing framework. We hope to fully replicate and improve the performance of SUVAG LINGUA using this software solution. We also need to develop a graphical user interface using tools from JUCE framework which will emulate the user interface of the SUVAG LINGUA device. This way, the adaptation period for the therapist is kept as short as possible. The main advantage of the software solution is the fact that speech and language therapists no longer need to rely on an old and obsolete device but have a possibility to use their personal computer to administer the therapy. This kind of approach will hopefully help spread the use of the verbotonal speech and language therapy using a simple and available tool.

Prilog 1

OPTIMALNA PODRUČJA GLASOVA HRVATSKOG JEZIKA

GLAS	KONTINUIRANO PODRUČJE
A	800 - 1600
E	800 - 2560
O	400 - 640
I	3200 - 4096
U	200 - 400
P	320 - 800
T	1600 - 3200
K	800-1600
B	200-400
D	512-1600
G	400-800
S	6400-8192
Š	1600-3200
F	1280-2560
Z	6400-8192
Ž	1280-2560
V	320-640
M	1024-2048
N	1600-3200

Nj	2560-3200
H	800-1024
C	5192-10240
Č	1600-3200
Ć	3200-6400
Dž	1280-2560
D	2560-5120
L	640-1600
Lj	1600-3200
R	800-1600
J	2048-4096

N. Desnica-Žerjavić, 1987.

Prilog 2

UPUTE ZA POKRETANJE I KORIŠTENJE PROGRAMA

Popis opreme potrebne za rad:

Neophodno: Računalo (Windows ili Mac OS), žični mikrofon, žične slušalice

Izborno: Vanjska zvučna kartica sa pojačalima i fantomskim napajanjem, dvije žične slušalice, dva žična mikrofona, audio *splitter*

Priprema programa za rad:

1. **Spojite dodatnu opremu na računalo**

2. **Locirajte mapu:**
   `.\DigiLingua\Builds\VisualStudio2022\x64\Debug\Standalone Plugin\`

3. **Pokrenite "DigiLingua.exe":**
   - Pokrenite s administratorskim pravima (izborno)

4. **Postavke audio/midi:**
   - U gornjem lijevom kutu pritisnite gumb **Options** --> **Audio/MIDI Settings...**

5. **Omogućite ulazni zvuk:**
   - Odznačite polje **"Mute audio input"** kako bi program počeo obrađivati ulazni tok podataka sa mikrofona

6. **Postavite izlaz zvuka:**
    - U prozoru **Output** postavite izlaz zvuka na slušalice i pokrenite **Test** za provjeru ispravnog rada

7. **Postavite ulaz zvuka:**
    - U prozoru **Input** postavite ulaz zvuka na mikrofon i provjerite pomiče li se nivo glasnoće ulaza

8. **Izađite iz postavki:**
    - Pritisnite **X** u gornjem desnom kutu.

OPREZ! Nikada ne postavljajte slušalice rehabilitantu dok se niste uvjerili da zvuk nije preglasan i ne dolazi do mikrofonije. U protivnom, riskirate dodatna oštećenja sluha rehabilitantu.

Priprema postavki filtera za rad:

1. **Visokopropusni filtar:**
    - Uključite filtar pritiskom na simbol iznad rotacijskog klizača za postavke parametara
    - Gornjim rotacijskim klizačem odredite graničnu frekvenciju filtra prema uputama za optimalne oktave (Prilog 1)
    - Donjim rotacijskim klizačem namjestite strminu gušenja filtra

2. **Pojasnopropusni filtar:**
    - Uključite filtar pritiskom na simbol iznad rotacijskog klizača za postavke parametara
    - Gornjim rotacijskim klizačem odredite centralnu frekvenciju filtra
    - Donjim rotacijskim klizačem odredite faktor kvalitete filtra

3. **Niskopropusni filtar:**
    - Uključite filtar pritiskom na simbol iznad rotacijskog klizača za postavke parametara
    - Gornjim rotacijskim klizačem odredite graničnu frekvenciju filtra prema uputama za optimalne oktave (Prilog 1)
    - Donjim rotacijskim klizačem namjestite strminu gušenja filtra

Korištenjem jednog ili više filtara postavite optimalnu oktavu za rehabilitaciju željenog fonema prema uputama u Prilogu 1.

Ostale postavke:

- **Kontrola izlazne glasnoće:**
    - Pomoću postavki glasnoće zvuka na računalu (donji desni kut na programskoj traci)

- **Vizualizacija spektra signala:**
    - Možete uključiti ili isključiti pritiskom na kvadratni simbol ispod gumba **Options**

- **Spremanje trenutnih postavki:**
    - Pritisnite **Options** --> **Save current state...** kako biste spremili vrijednosti parametara programa za naknadno učitavanje istih

- **Učitavanje spremljenih postavki:**
    - Pritisnite **Options** --> **Load saved state...** kako biste učitali prethodno spremljene vrijednosti parametara ili predloške koji postavljaju parametre prema željenom fonemu

- **Vraćanje na početne postavke:**
    - Pritisnite **Options** --> **Reset to a default state** kako biste poništili sve promjene i vratili parametre programa na početne postavke

Završetak rada:

Kad ste gotovi s radom, izađite iz programa pritiskom na **X** u gornjem desnom kutu prozora.