SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1536

OBRADBA I MANIPULACIJA GLAZBE POMOĆU UGRADBENIH SUSTAVA

Petra Kastrapeli

Zagreb, lipanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 3. ožujka 2017.

DIPLOMSKI ZADATAK br. 1536

Pristupnik: **Petra Kastrapeli (0036464829)**
Studij: Elektrotehnika i informacijska tehnologija
Profil: Elektroničko i računalno inženjerstvo

Zadatak: **Obradba i manipulacija glazbe pomoću ugradbenih sustava**

Opis zadatka:

U sklopu ovog diplomskog rada potrebno je istražiti mogućnosti digitalne obradbe, sinteze i reprodukcije glazbe pomoću ugradbenih sustava s naglaskom na platformu Raspberry Pi. To uključuje istraživanje parametara potrebnih za obradu, odgovarajuće jezgre Linux operativnog sustava i dodatnih programa i zvučnih kartica kako bi se obradba provela s čim većom izlaznom kvalitetom zvuka. Težište rada je na obradbi glazbe ugradbenim sustavom kojem takva zadaća nije primarna namjena te na mogućnostima prilagodbe sustava za takvu namjenu. Istražuju se mogućnosti višeglasne obradbe glazbe i diskutira primjena takve obradbe. Posebnu pažnju posvetiti korištenju programskog sustava Pure Data za upravljanje postupcima digitalne obradbe i sinteze glazbe.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 29. lipnja 2017.

Mentor:
Prof. dr. sc. Davor Petrinović

Djelovođa:
Prof. dr. sc. Dražen Jurišić

Predsjednik odbora za diplomski rad profila:
Prof. dr. sc. Mladen Vučić

Sadržaj

Uvod

Predmet istraživanja ovog diplomskog rada su mogućnosti digitalne obrade, sinteze i reprodukcije audio signala pomoću ugradbenih sustava pri čemu je naglasak na platformi Raspberry Pi.

Težište rada je na Raspebrry Pi-ju i njegovim mogućnostima za takvu obradu s obzirom da mu to nije primarna namjena te su potrebne prilagodbe u vidu zvučnih kartica i programa za obradu poput Pure Data koji će se korisitit u ovom radu.

U prvom poglavlju govorit će se općenito o digitalnoj obradi signala i njezinoj primjeni u obradi zvučnih signala. Pažnja će se posvetiti i karakteristikama ljudskog sluha jer je obrada audio signala usko vezana uz područje ljudskog sluha. Na kraju poglavlja bit će rečeno nešto o korištenju ugradbenih računalnih sustava za digitalnu obradu audio signala.

U drugom poglavlju naglasak će se staviti na Raspberry Pi 2 kao ugradbeni računalni sustav koji se koristi u ovom projektu. Proučavat će se mogućnosti njegovog ulazno-izlaznog sklopovlja i I2S protokola za obradu audio signala na Raspberry Pi 2. U sklopu audio karakteristika posebna pažnja posvetit će se ALSA upravljačkom programu.

U trećem poglavlju proučavat će se karakteristike grafičkog programskog jezika za audio i video signale Pure Data te će korištenje navedenog jezika biti stavljeno u kontekst korištenja na Raspberry Pi 2.

Posljednje poglavlje će demonstrirati realizaciju projekta na Raspberry Pi 2 u nekoliko primjera u kojima će biti prikazano generiranje zvuka pomoću Pure Data, obrada i manipulacija prethodno snimljene datoteke te manipulacija ulaznih zvukova s mikrofona.

1. Digitalna obrada zvučnih signala

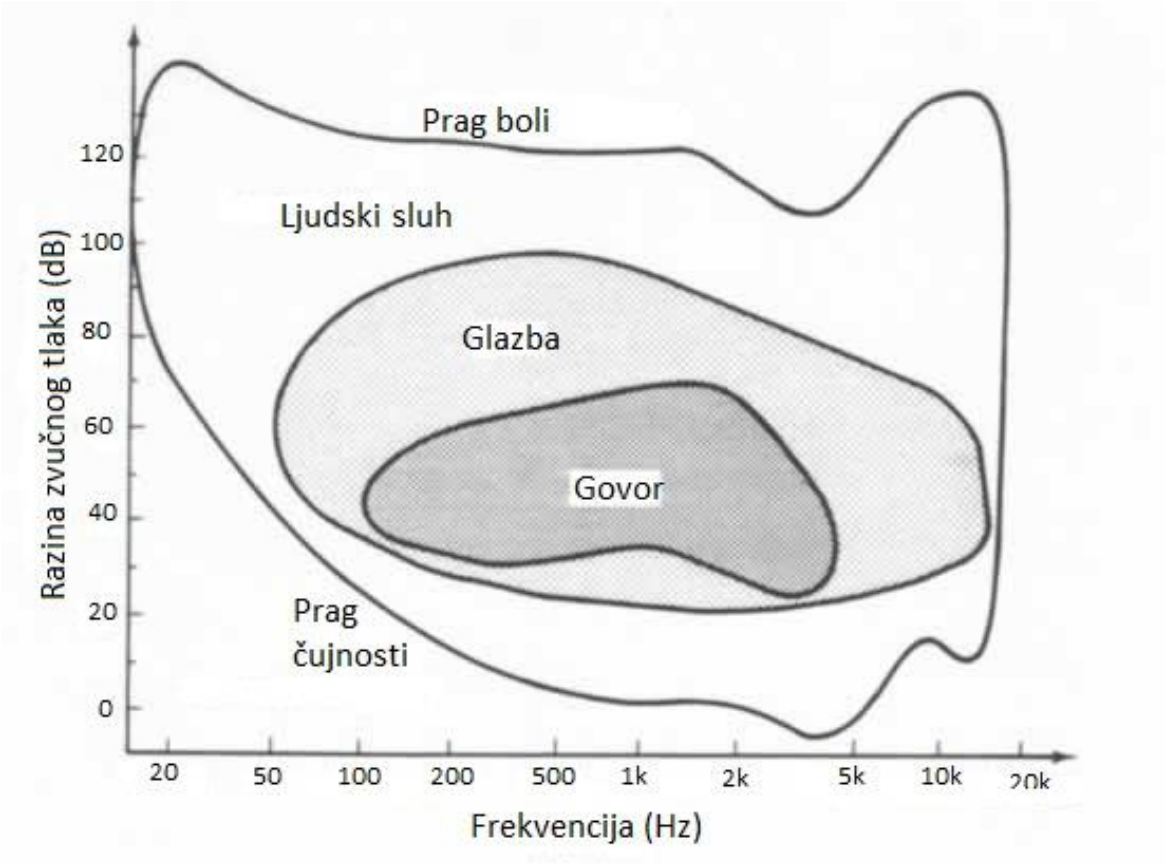
Digitalna obrada signala, u općenitom smislu, podrazumijeva digitalizaciju i matematičku obradu signala prikupljenih iz vanjskog svijeta poput zvuka, videa, tlaka, temperature i svih matematički mjerljivih pojava. Signali se obrađuju kako bi se mogli analizirati, prikazivati ili pretvarati u drugačiji oblik signala za daljnju obradu. S obzirom da su svi realni signali analognog oblika, a računala žive u digitalnom svijetu, prije bilo kakve obrade je potrebna konverzija analognog signala u digitalni. Pojam „digitalni“ podrazumijeva upravo diskretizirani analogni, realni signal. Prednosti digitalne obrade signala nad analognom su višestruke:

1. Programibilnost - Jednostavna promjena koda za optimiziranje postojeće ili realizaciju nove aplikacije
2. Ponovljivost
    - Karakteristike nisu promjenjive niti ovise o vremenu ili temperaturi kao kod analognih komponenata
    - Više DSP-a može raditi isti program i nema potrebe za ugađanjem svakog posebno kao kod analogne obrade
3. Veličina i snaga
    - Rješenje je uglavnom programsko i koristi manje snage od rješenja s isključivo hardverskim komponentama
4. Pouzdanost
    - Analogni sustavi su pouzdani dokle hardver radi kako treba, ako dio ne radi, cijeli sustav prestaje raditi
    - Digitalno rješenje radi dobro dokle god je kod ispravno implementiran
5. Proširivost - Jednostavno dodavanje novih fukcionalnosti bez potrebe za dodatnim hardverskim komponentama

1.1. Karakteristike ljudskog sluha i zvučnih signala

Ljudsko uho bez napora može procesirati i kategorizirati veliki broj različitih zvukova u isto vijeme. Proces odvajanja i identificiranja izvora zvuka iz složene kompozicije zvukova naziva se analiza zvučne slike [1]. Izvedba takvog sustava u elektroničkom obliku (razdvajanje i klasifikacija zvuka) može biti vrlo korisna u prepoznavanju govora iz buke, automatskoj transkripciji glazbe ili pretraživanju multimedijalnih podataka. Zvučni signali podrazumijevaju vrlo široki opseg signala, no općenito bi se mogli podijeliti u četiri skupine na temelju zajedničkih karakteristika. U ovom radu bit će proučavane dvije skupine. Prva bi bila skupina koja ima čiste frekvencijske karakteristike kao što je instrumentalna glazba, dok bi u drugu spadali govor i njegova glazbena varijacija poput pjesme te drugi zvukovi koji u sebi sadrže komponente šuma. Ljudski govor, pa tako i pjesma, nikad nisu čisti zvukovi točne frekvencije zbog efekata ljudskog vokalnog trakta. S druge pak strane, instrumenti su namijenjeni dobivanju čistog tona točne frekvencije što je glavna karakteristika na temelju koje se određuje ton. Osim dvije proučavane skupine, posebna skupina su zvukovi okoline te umjetno stvoreni zvukovi. Svaki zvuk određen je svojom frekvencijom/visinom, glasnoćom, duljinom trajanja i bojom.

Zvučni signali sastoje se od skupine koju ljudsko može procesirati (20Hz do 20kHz) i one koju ne može, dok su za računalnu obradu te skupine određene filtrima koji izoliraju dio signala koji nam je zanimljiv ili propuštaju sve bez filtra. S obzirom da je zvučni signal jednodimenzionalni signal kojem je neovisna varijabla vremenska razlika tlaka na mjestu mjerenja, na primjer mikrofon, za daljnu obradu potrebno je uzorkovati dobiveni signal. Iz povijesnih razloga najčešće se koristi 44,1 kHz u 16-bitne riječi što se naziva CD kvalitetom uzorkovanja, iako su često u upotrebi i frekvencije od 48 kHz i 92 kHz. Govor i glazba su vremenski promjenjivi signali s kratkim segmentima određenog trajanja (glas i ton), od kojih svaki može biti opisan svojim vremenskim, frekvencijskim i spektralnim karakteristikama.



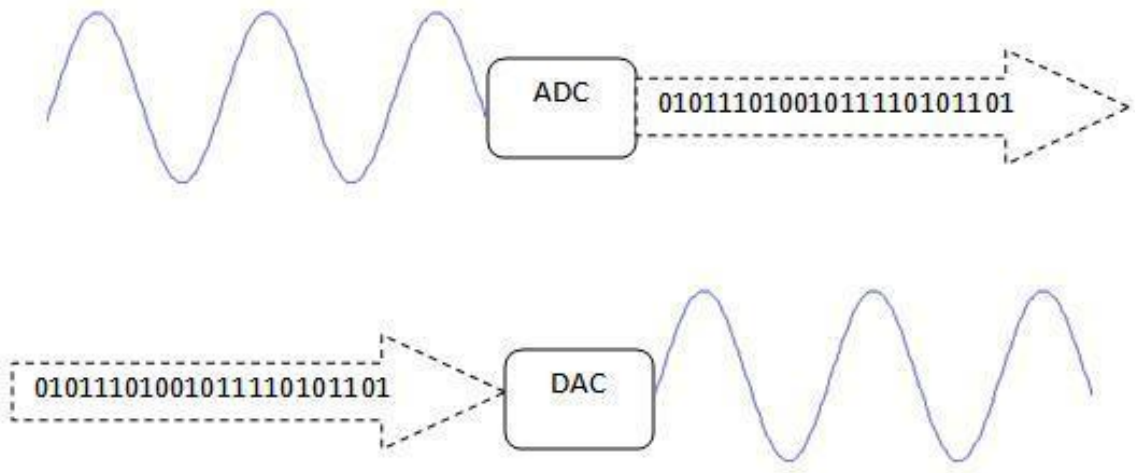
Slika 1.1 Područje ljudskog sluha

Osim frekvencija koje možemo procesirati, postoji i određena snaga koju možemo procesirati koja se najčešće prikazuje u decibelima te iznosi 120dB što je prag boli za ljudsko uho te nam nije u interesu proizvoditi i procesirati glasnije zvukove. Slika 1.1 prikazuje područje ljudskog sluha s nama interesantnim područjima glazbe i govora. Iz gornje slike možemo vidjeti da je općenito područje ljudskog sluha 20 Hz do 20kHz pri čemu nam je najosjetljivije područje sluha od 2 do 4 kHz, no s godinama se gornja granica smanjuje. Također je vidljivo da nam je na nekim frekvencijama potrebna veća razlika u tlaku kako bismo registrirali zvuk. Područje u kojem se nalazi intrumentalna glazba nalazi se od 100 Hz do 2kHz. Frekvencija određuje visinu tona, informaciju o kojem se tonu radi (npr. 440 Hz označava A1). Osim frekvencijom, instrumentalni tonovi i govor određeni su i bojom zvuka. To znači da je da je svaki ton određen svojom frekvencijom te može imati niz popratnih, tzv. alikvotnih tonova. Njih nije moguće svjesno prepoznati, ali utječu na našu percepciju punog tona. Iako ih ljudsko uho ne prepoznaje, u signalnom obliku su oni prisutni. Alikvotni tonovi imaju frekvencije koje su jednake višekratnicima broja 2 pomnoženima s osnovnim frekvencijom, npr. ako je osnovna frekvencija

220 Hz, alikvotne frekvencije su 440, 660, 880 i tako dalje. Maksimalan broj alikvotnih tonova određen je nekom graničnom frekvencijom koja se razlikuje za svaki instrument, no smatra se da je 8 dovoljno za vrlo bogat zvuk jer svaki novi višekratnik ima manju amplitudu te je manje čujan. Instrumenti kod kojih je granična frekvencija niža, poput flaute, imaju manje alikvotnih tonova što daje „nježniji“ zvuk od instumenata kod kojih je ta frekvencija viša, na primjer truba.

1.2. Primjena digitalne obrade na zvučne signale

Budući da su zvučni signali, kao i svi prirodni signali, kontinuirani te nisu stalni kao običan sinus, njihovo prikazivanje u računalu u takvom obliku je nemoguće jer kontinuirani signali predstavljaju beskonačan niz brojeva. Zbog toga se radi digitalna reprezentacija takvih signala koja se potom obrađuje. Digitalnu reprezentaciju možemo dobiti na više načina, no ona je u svojoj funkcionalnosti uvijek analogno-digitalna pretvorba. Bitna stavka kod analogno digitalne pretvorbe je svakako frekvencija očitavanja, koja za zvučne signale uobičajeno iznosi 44,1kHz što se naziva CD kvalitetom. Osim frekvencije otipkavanja na kvalitetu utječe i širina riječi, pri čemu veća širina riječi znači manju razliku između uzoraka te možemo razlikovati zvuk s većom preciznošću. Logično, želimo što veću preciznost, no veća preciznost znači i više potrošene memorije pa se najčešće koriste 32 i 64-bitne riječi koje daju finu preciznost, a količina memorije koju zauzimaju je prihvatljiva za redovitu upotrebu.

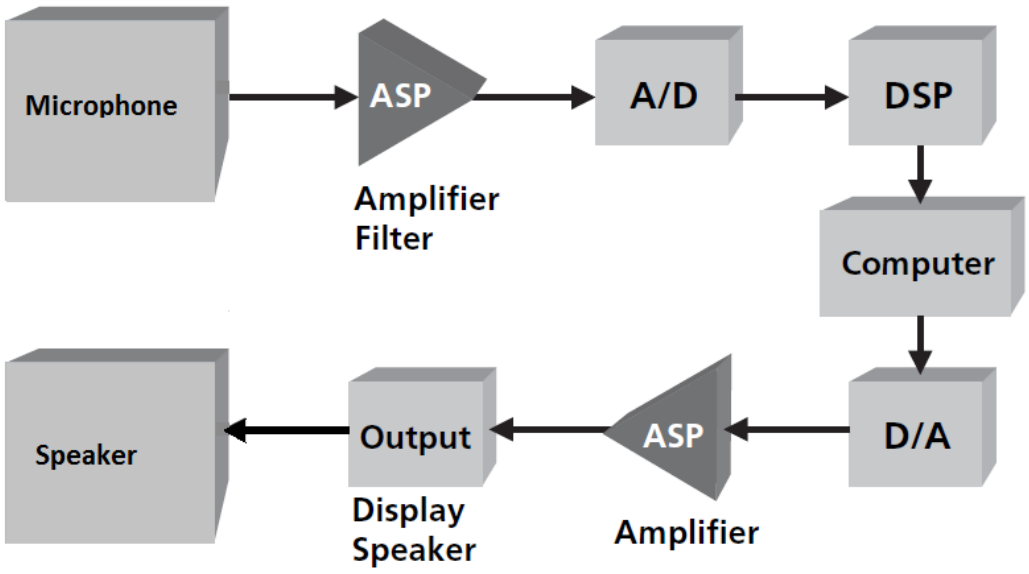


Slika 1.2 ADC - DAC

Glavna karakteristika obrade zvučnih signala je obrada u kratkim segmentima jednakog trajanja koji su relativno stacionarni. Takav način obrade naziva se *vremenski kratkotrajna analiza* (engl.*short-time analysis*). Govorni signali ili njihova glazbena reprezentacija često se aproksimiraju poznatim modelom ljudskog govora. Osim na govor ili pjesmu, slični modeli su primjenjivi i u instrumentalnom obliku kod instrumenata poput žičanih koji zvuk stvaraju svojom vibracijom, iako se češće promatraju karakteristike signala s instrumenta s obzirom da je takav zvuk najčešće kombinacija sinusnih funkcija. S obzirom na takve karakteristike instrumentalnog zvuka, relativno je lako izolirati šum koji utječe na analizu i reprodukciju. Budući da su realni zvučni signali često zašumljeni, na primjer snimanje koncerta u dvorani s publikom, digitalna obrada se može primjeniti i na izolaciju signala od šuma i dobivanja čistog zvučnog zapisa.

1.3. Digitalna obrada zvučnih signala na ugradbenim sustavima

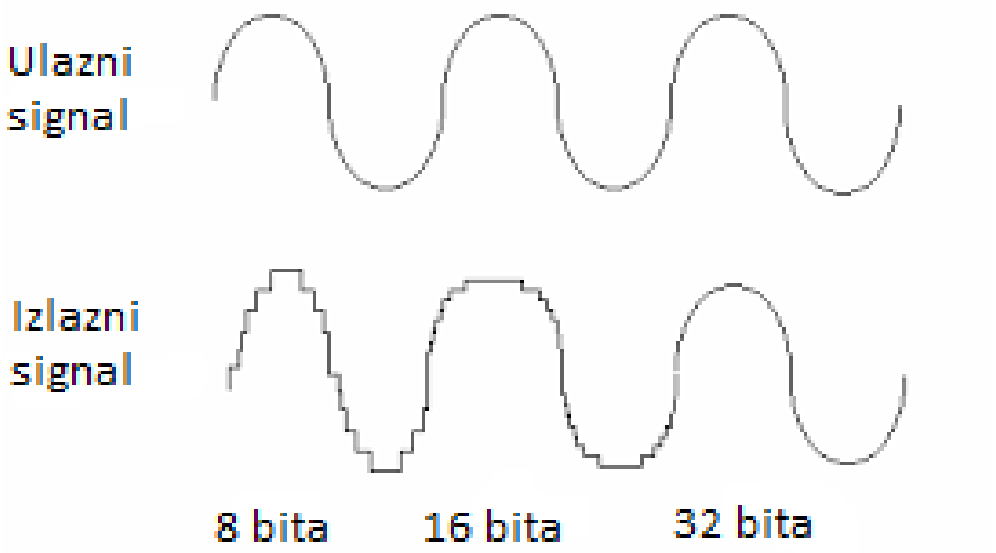
Ugradbeni sustavi su, u svojoj funkcionalnosti, slični ljudima. Kada smo obučeni za neku zadaću i izvještili smo se u njoj, manje vremena i energije trošimo na njezino obavljanje [2]. Slično je s mikroprocesorima na ugradbenim računalima, imaju svoju specifičnu namjenu. Tako postoje mikroprocesori namijenjeni za digitalnu obradu signala.



Slika 1.3 Općeniti izgled *audio embedded* sustava

Ugradbeni sustavi za obradu signala (DSP) često imaju ugrađeni mikrofon što je, kao što se može vidjeti na slici 1.3, ulazna točka sustava. U slučaju kada mikrofon ne postoji ili ga ne želimo koristiti, ulazna točka je prethodno snimljeni signal pohranjen u memoriji. Ulazni signal može biti glazba ili glas, no u ovom radu će nagasak biti na glazbi. Obrada govora može biti vrlo kompleksna i zahtjevnija od obrade glazbe te se u ugradbenim sustavima koristi u mobilnim uređajima za uklanjanje jeke i sažimanje signala za prijenos. S obzirom da je ulazni signal s mikrofona analogni signal, sljedeća točka je analogno - digitalna pretvorba kojoj može i ne mora prethoditi analogna predobrada signala. Na sličan način ulaznoj točki koncipirana je i izlazna linija u kojoj je postupak obrnut od ulaza, točnije, radi se digitalno - analogna pretvorba kako bi signal na zvučnicima bio analogni koji ljudsko uho prepoznaje.

Tijekom analogno - digitalne pretvorbe signal je moduliran i za zvučne signale najčešće se koristi pulsno - kodna modulacija (engl. *Pulse-code modulation*, PCM). Preciznost kojom možemo prenijeti točne analogne vrijednosti ovisi o broju kvantizacijskih razina. Većina današnjih pretvornika za audio signale je sigma-delta s 24-bitnom preciznošću, što znači da ima 2^{24} kvantizacijskih razina. Na primjeru signala kojem se ampliture kreću između -5 i 5 V, to znači da može prikazati svaki pomak signala od $10V/2^{24}= 596nV$.

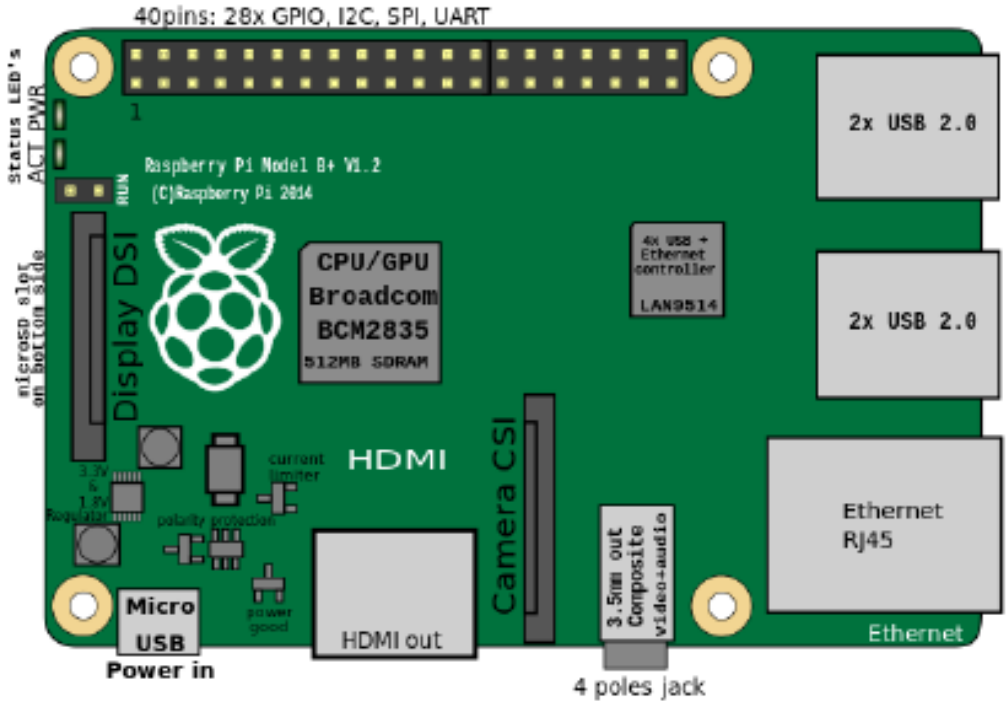


Slika 1.4 Razlika među pretvornicima

Najbitniji dio obrade na DSP je onaj između dvije pretvorbe signala s obzirom da tu manipuliramo i obrađujemo signal po želji ili potrebi. Upravo na tom dijelu će biti naglasak u ovom radu.

2. Raspberry Pi

Raspberry Pi je minijaturno računalo (engl. *Single-board computer*) veličine kreditne kartice čiji je razvoj započeo idejom da se mladim ljudima predstavi jednostavno i financijski prihvatljivo okruženje za učenje programiranja [3]. Verzija korištena u ovom radu, Raspberry Pi 2, na tržištu se pojavio početkom 2015. godine te se pokazao kao znatno poboljšana verzija ranijeg modela. Najveća promjena se dogodila kod procesora koji je promjenjen s ARM11 (jedna jezgra i maksimalna brzina 700MHz) na ARM Cortex-A7 (četiri jezgre i makimalna brzina 900MHz) te u broju GPIO pinova čiji je broj povećan s 26 na 40. Veći broj pinova bio je ključan za razvoj većeg broja pločica koje su bile kompatibilne s RPi te mu pružale širi opseg funkcionalnosti, ponajviše u raznim obradama signala. Osnovni izgled Raspberry Pi 2 Model B prikazan je na slici 2.1. Za svoje napajanje RPi koristi MicroUSB utor uz 5V@2A za stabilno napajanje. Napajanje može biti izvedeno i preko GPIO pinova 2 (5V) i (GND).



Slika 2.1 Raspberry Pi 2 Model B

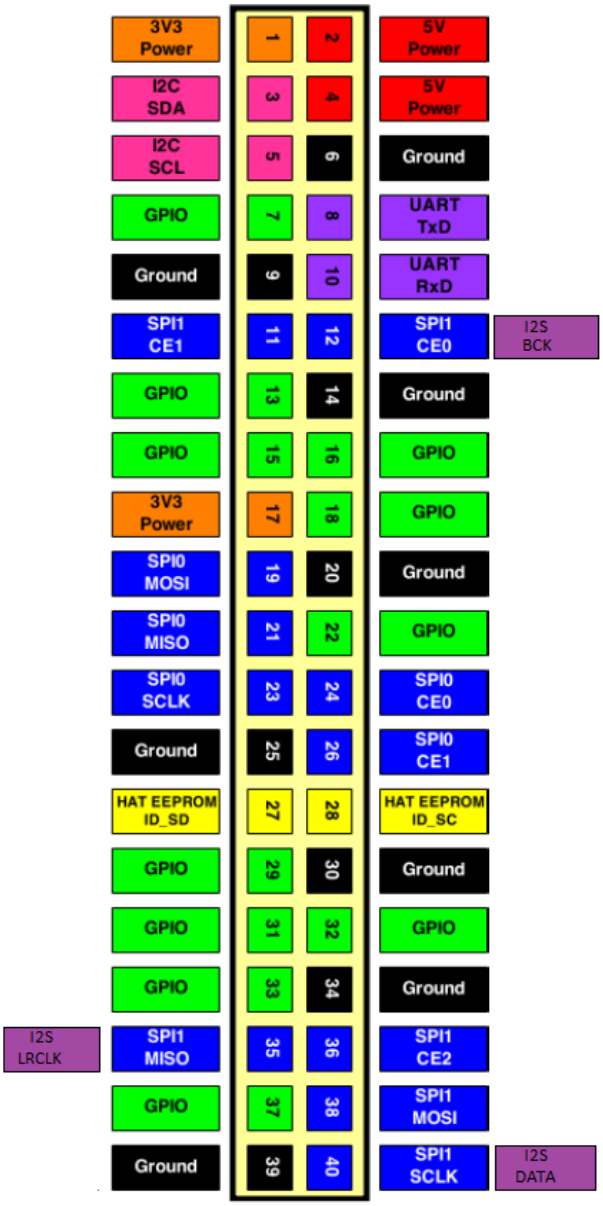
Budući da se u ovom radu RPi koristi u takozvanom *headless* načinu rada, koji podrazumijeva rad bez bilo kakvog grafičkog sučelja, te u *remote access* načinu, u nastavku će biti objašnjeni samo dijelovi pločice i postupci nužni za takvo funkcioniranje. *Headless* način rada isključuje upotrebu bilo kakvog grafičkog sučelja poput monitora te korištenje tipkovnice, točnije pristupa mu se preko mreže pomoću *remote access-a.* S obzirom da RPi nema Wireless modul, najjednostaviji način spajanja na mrežu je Ethernet. Sami pristup se radi preko IP adrese koja se RPi-ju može postaviti kao statička, no u ovom slučaju je, uz opciju statičkog IP, stvorena skripta koja svaki dan u isto vrijeme šalje vanjski IP, kako bi se pločici moglo pristupati i izvan lokalne mreže. Kasnije će biti objašnjeno zbog čega je pristup bez grafičkog sučelja bio potreban u sklopu ovog rada.

2.1. GPIO

RPi 2 ima 40 GPIO pinova koji mogu biti konfigurirani kao općeniti ulazi za signale, općeniti izlazi ili kao jedna od 6 posebnih funkcionalnosti:

1. Klasični GPIO koji se može koristiti za bilo koju potrebu takvih pinova poput paljenja i gašenja LEDica
2. I2C sučelje koje omogućuje spajanje dodatnog hardvera preko samo dva pina
3. SPI sučelje
4. UART
5. HAT EEPROM
6. I2S

Na slici 2.2 prikazan je raspored GPIO pinova po protokolima za koje se koriste. Pinovi koje odabrani protokol ne koristi mogu i dalje biti općeniti ulazni ili izlazni, no svi preostali pinovi mogu biti korišteni kao klasični ulazni ili izlazni pinovi.



Slika 2.2 Raspored GPIO pinova

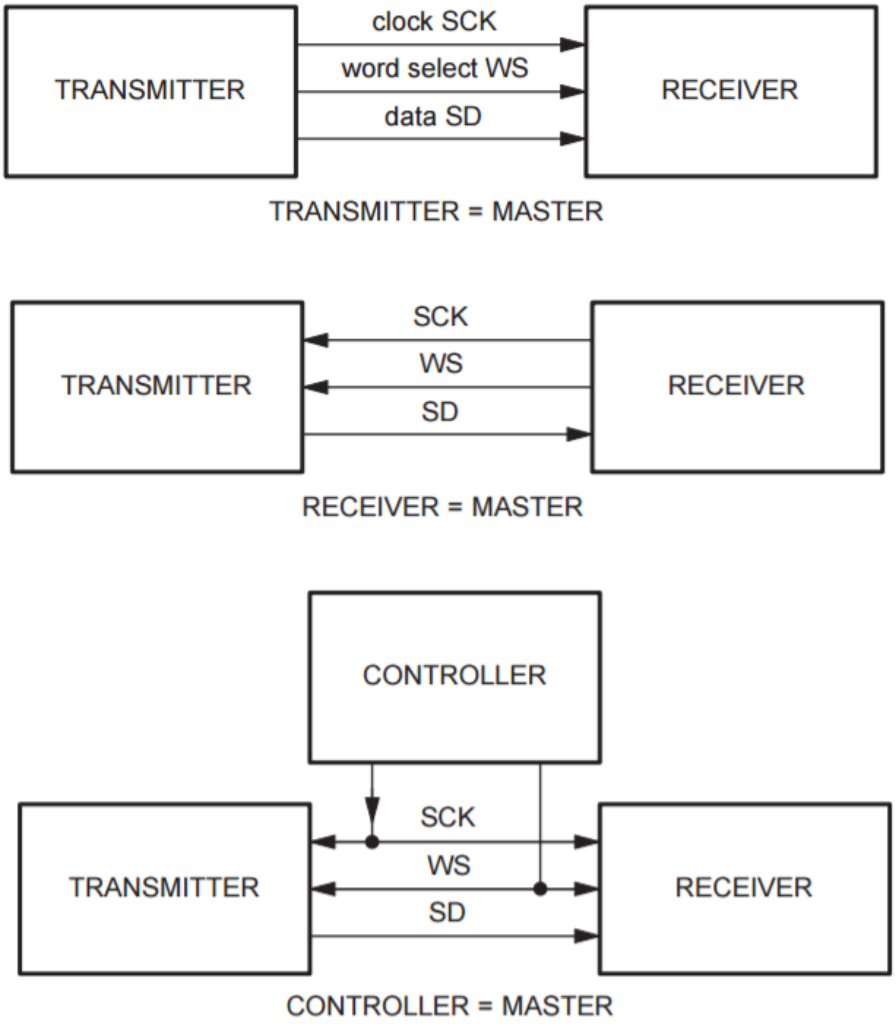
Protokol I2S nije uobičajen na dodatnim karticama koje se spajaju na GPIO pinove, no u ovom slučaju najbolje je koristiti upravo taj protokol što će detaljnije biti objašnjeno u sljedećem potpoglavlju. Dodatne kartice, u ovom slučaju zvučna, spajaju se na svih 40 pinova, no ovisno o protokolu prema gornjem prikazu, aktivni su samo neki pinovi.

2.1.1. I2S sučelje

I2S sučelje (Integrated Interchip Sound) je standard za serijsku sabirnicu korištenu za spajanje digitalnih audio uređaja[4]. Uobičajeno se koristi za prijenos PCM (engl. *Pluse-code modulation*) audio podataka između integriranih krugova na

pločici. Uz TDM i PDM, I2S je najčešće korišten za prijenos na jednoj pločici, bez dodatnih kablova, pri čemu su TDM i PDM nastali kao izvedenice I2S za veći broj slave-a. I2S sabrinica se koristi isključivo za audio podatke, dok se ostali signali poput kontrolnih, prenose posebno. I2S koristi tri signalne linije kako bi se smanjio broj potrebnih pinova, od kojih su dvije *clock*:

- bit clock (BCLK, SCK) čija frekvencija ovisi o frekvenciji otipkavanja (512kHz@8kHz otipkavanja do 12.288MHz@192kHz otipkavanja)
- Word select koji se naziva i *Left-right clock* (LRCLK, WD) koji odlučuje koji kanal šalje (0 lijevi, 1 desni)
- Podaci (SDATA, SDIN, SDOUT) koji mogu biti 16, 24 i 32-bitni



Slika 2.3 Način povezivanja I2S[3]

Na slici 2.3 prikazan je odnos preko I2S veze u ovom radu, pri čemu je Raspberry Pi master, a zvučna kartica slave. Kako odašiljač i prijemnik koriste isti *clock*, strana koja je *master* mora generirati oba *clock*-a, dok podaci uvijek idu u istom smjeru, od odašiljača do prijemnika. Ako u sustavu postoji više odašiljača i prijemnika, postoji i kontrolni sustav koji tada daje oba *clock*-a te je tada on *master*. Neki IC podržavaju 32 ili čak 48 bitni stereo clock po jednom uzorku. Moguće je slanje više od jednog ulaznog ili izlaznog podatkovnog signala na nekim IC, ali to povećava broj pinova potrebnih za slanje podataka te se tada često koriste TDM i PDM za prijenos.

Podaci se preko linije prenose u dvojnom komplementu s MSB (*Most Significant Bit*) na prvom mjestu. Struktura podatka je takva zbog različite duljine riječi na različitim stranama. Na taj način odašiljač ne mora znati koliko bitova prijemnik može primiti i prijemnik ne mora znati koliko je bitova poslano. Podatkovna linija može biti sinkronizirana na rastući i na padajući brid *clock*-a.

2.2. Audio karakteristike

S obzirom da nije inicijalno namijenjen aplikacijama koje uključuju zvuk, Raspberry Pi ne posjeduje priključak za ulazni zvučni signal te je njegova mogućnost izvođenja izlaznog zvuka niske kvalitete (pulsno-širinska modulacija). Razlika između uobičajeno korištene pulsno - kodne modulacije i pulsno - širinske korištene na Raspberry Pi-ja je činjenica da pulsno-širinska nema potrebu za digitalno - analognim pretvornikom već signal šalje direktno na izlaz. Takvo rješenje je jeftino i prikladno u ovom slučaju jer to nije inicijalna namjena.

Iako ne posjeduje ulazni priključak, s vremenom su se razvili dodaci, poput USB i I2S zvučnih kartica te *open-source* audio distribucija, koji omogućavaju takve aplikacije. Iako oba načina komunikacije ispunjavaju svoju svrhu, I2S ima nekoliko prednosti na USB zvučnim karticama. Raspberry Pi posjeduje nekoliko načina reprodukcije izlaznog zvuka (sa ili bez dodataka):

1. Klasična 3.5 mm izlazna linija za slušalice koja je primarna izlazna linija i ima nisku kvalitetu
2. HDMI port koji prenosi zvuk kada je RPi spojen na monitor ili sličan izlaz

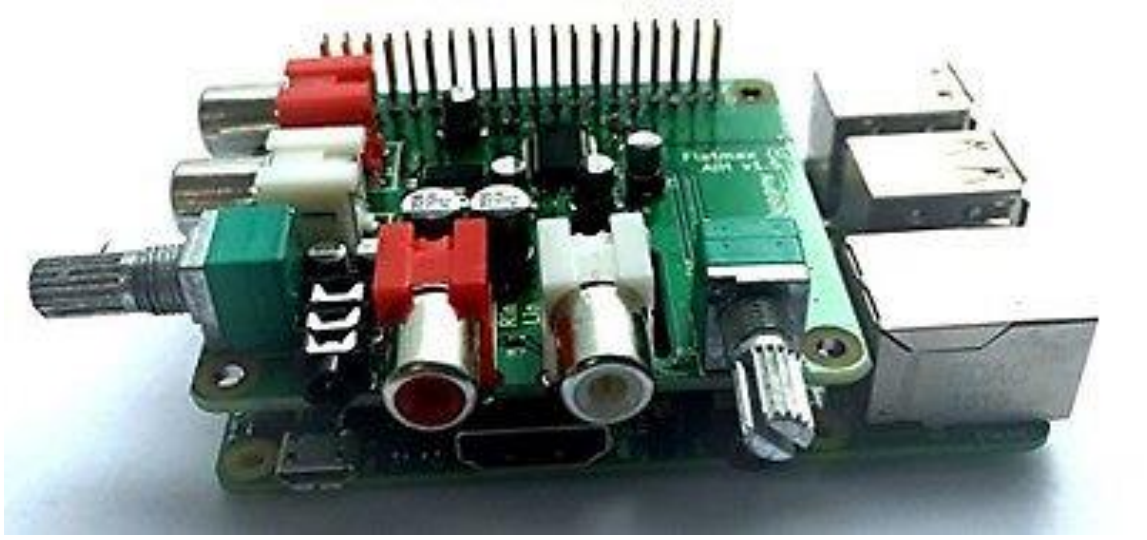
3. USB zvučna kartica
4. USB na I2S adapter što ima veću kvalitetu zvuka, no ima gubitke zbog dodatne „karike“ u lancu te kvaliteta uvelike ovisi o tehnologiji korištenih komponenata i kvaliteti *clock*-a
5. Direktni I2S spoj preko GPIO pinova na RPi koji daje najbolju kvalitetu i koji će se koristiti u ovom radu

U nastavku će biti objašnjen način spajanja preko I2S sučelja i njegove prednosti nad ostalim načinima. Također će se posvetiti pažnja karakteristikama USB zvučnih kartica te njihovim nedostacima u odnosu na I2S zvučne kartice.

2.2.1. Zvučne kartice

Na početku poglavlja navedeno je da postoje dvije kategorije zvučnih kartica za RPi, USB i I2S [5]. USB zvučne kartice su jednostavnije i povoljnije rješenje, no imaju nekoliko nedostataka u odnosu na I2S zvučne kartice. Sama fizička veza je lošija je nema nikakvog osiguranja od iskapčanja, kvaliteta zvuka je manja nego kod I2S kartica zbog nemogućnosti reprodukcije u kvaliteti 24bita@192kHz te je podložnija podrhtavanju signala na što uvelike može utjecati vanjska USB memorija ukoliko je spojena.

I2S zvučne kartice su nešto skuplje rješenje, no u ovom projektu i velikom broju ostalih bolje rješenje. Fizička veza preko 40-pinskog GPIO je kvalitetnija te je mogućnost gubitka kontakta za vrijeme rada manja. Većina I2S kartica je kompatibilna s ALSA driverima bez potrebe za dodatnim instalacijama te jednostavnim prepravcima *config* datoteka za prepoznavanje kartice. Također, razlika u odnosu na USB kartice je u namjeni, točnije I2S kartice su dizajnirane upravo za Raspberry Pi u odnosu na USB kartice koje su dizajnirane za općenitu upotrebu.



Slika 2.4. AudioInjector zvučna kartica + RPi2

U ovom projektu se koristi I2S zvučna kartica Stereo AudioInjector [6] koja ima sljedeće karakteristike:

- Stereo RCA ulaznu i izlaznu liniju s mogućnošću kontrole glasnoće na obje linije
- Izlazna 3.5mm linija za slušalice s ugrađenim pretpojačalom snage 30mW@32Ω i 50mW@50Ω
- Ugrađeni mikrofon kao mogućnost ulazne linije
- Kratko vrijeme kašnjenja na obje linije (min. 540ns)
- Bypass funkcionalnost za direktno provođenje analognog ulaza na izlaz
- Ugrađeni standardni 40-pinski GPIO za mogućnost dodavanja većeg broja dodatnih pločica
- Kristalni oscilator s niskim podrhtavanjem

2.2.2. ALSA upravljački program

ALSA (engl. *Advanced Linux Sound Architecture*) je istovremeno projekt i skupina programskih rješenja [7]. Projekt je započet zbog nedostataka OSS (Open Sound System) arhitekture te zbog mogućnosti pristupa nekim driverima samo u komercijalnoj verziji, a ne i u besplatnoj. Prvih nekoliko godina ALSA software je bio razvijan neovisno o Linuxu. Projekt razvoja ALSA upravljačkog programa vodio je Jaroslav Kysela i bazirao se na Linux upravljačkom programu za Gravis

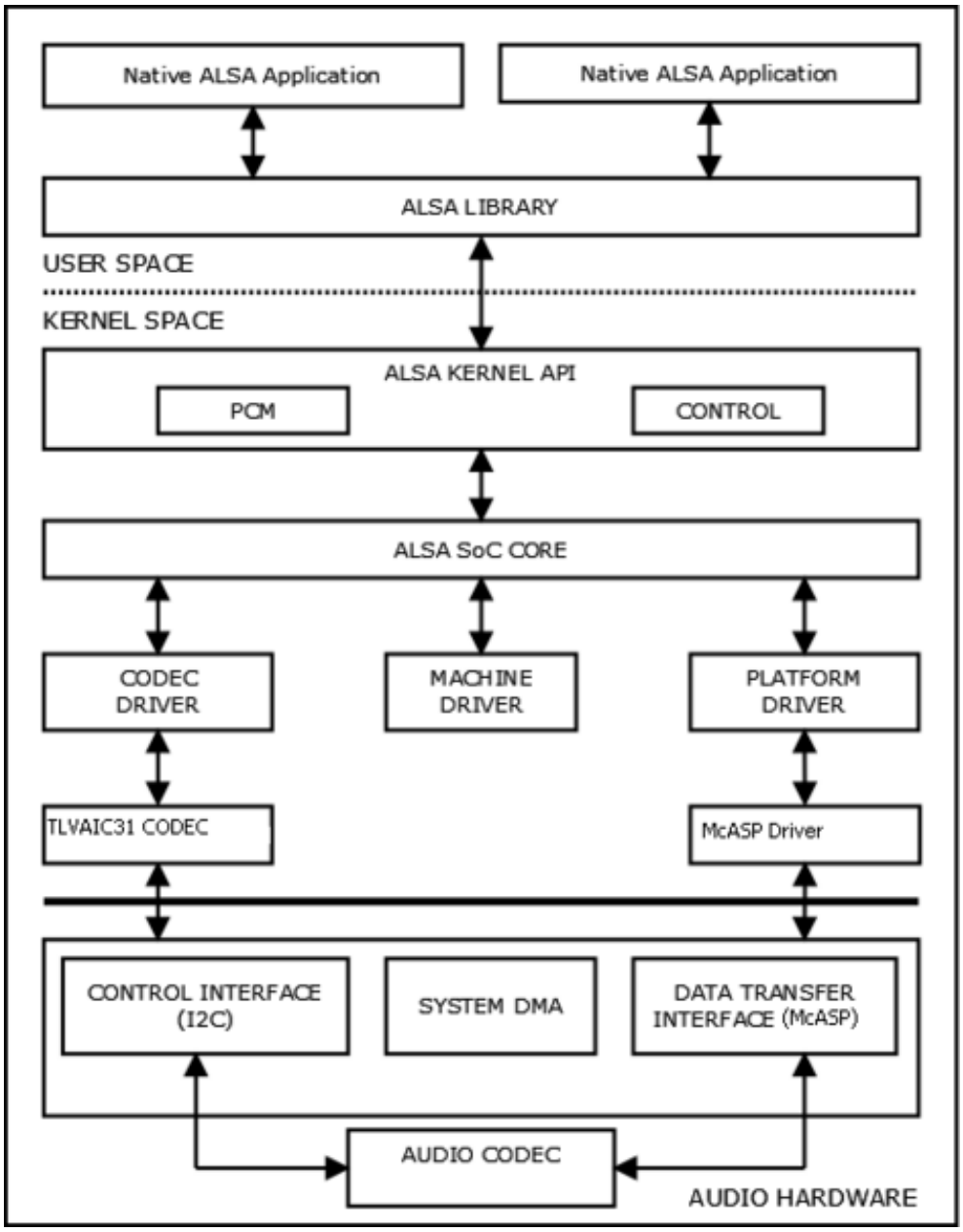
Ultrasound zvučnu karticu. Započet je 1998 i razvijan neovisno od Linux jezgre sve dok nije dodan u Linuxovu bazu programskih rješenja u razvoju serije 2.5 (dodano u verziji 2.5.5) i postali strandardni upravljački program (engl. *driver*) za zvuk u verziji 2.6 te time potpuno zamijenili stariji OSS.

ALSA je, uz funkciju zvučnog upravljačkog programa, biblioteka s proširivim aplikacijskim programskim sučeljem (engl. *Application Programming Interface, API*) koji omogućuje aplikacijama pristup mogućnostima zvučne kartice. API je kod ALSA upravljačkog programa veći i složeniji nego kod OSS, što može otežati razvoj aplikacije koja koristi ALSA-u. ALSA upravljački program omogućava audio i MIDI funkcionalnosti Linux OS-u. Važnije karakteristike, koje čine ALSA upravljački program korisnim, su:

1. Učinkovita podrška za sve vrste audio sučelja, od korisničkih, *off-the-shelf* zvučnih kartica do profesionalnih višekanalnih audio sučelja
2. Mogućnost potpune modularizacije zvučnog upravljačkog programa
3. Simetrično multiprocesiranje i sigurnost dretvi
4. Biblioteka dostupna u user - space-u (alsa - lib) za jednostavnije programiranje aplikacija i omogućavanje više razine funkcionalnosti
5. Podrška za stariji OSS API, kompatibilnost s većinom OSS programa

Učinkovita podrška znači da je korisniku omogućeno upravljanje osnovnim i naprednim karakteristikama podržanih zvučnih kartica na jednostavan način korištenjem ALSA alata poput programa za konfiguraciju ili programa za miksanje. Takvi alati su integrirani u cjelovitu ALSA instalaciju.
Mogućnost modularizacije podrazumijeva jednostavnu instalaciju i nadogradnju upravljačkog programa te pruža sredstva kojima korisnik može detaljnije proučavati mogućnosti svoje zvučne kartice.
Simetrično multiprocesiranje (engl. *Symmetric multiprocessing, SMP*) omogućava izvođenje zadatka na više jednakih procesora istovremeno ako je dostupno više od jednog procesora. Sigurnost dretvi omogućava izvršavanje koda ili zadatka paralelno u više dretvi, a da pri tome ne ometaju jedna drugu. U suvremenoj audio i MIDI obradi sigurnost dretvi je vrlo korisna.
User-space biblioteka omogućava programerima i njihovim kodovima jednostavan pristup ALSA uslugama i njihova važnost se može činiti običnom korisniku blaga. Ipak ALSA biblioteka omogućava sučelje preko kojega aplikacija može pristupiti

funkcijama, stvarajući homogeniju okolinu na korisničkoj razini. Korisnički programi se mogu izvršavati međusobno usklađeno s povećanim mogućnostima za povezivanje i komunikaciju između aplikacija.



Slika 2.5. Općenita struktura ALSA drivera

S obzirom da se ALSA počela razvijati tijekom rane faze zvučne podrške za Linux kada je većina aplikacija koristila OSS/Free API, kompatibilnost s OSS/Free je bila od velike važnosti za obične korisnike. Veliki broj audio aplikacija i dalje traži OSS kompatibilnosti pa ALSA pruža podršku za starije API-je iako programeri trebaju imati na umu da je stariji API sada je službeno obustavljen.

Uobičajeno ALSA podržava do osam zvučnih kartica u isto vrijeme koje su označene brojevima od 0 do 7. Svaka kartica može biti fizička ili logička jezgra uređaja koja može biti definirana kao input ili output. Također, svaka kartica može biti adresirana po svojem ID koji je opisni *string* poput „*Headset*“ ili „*ICH9(I/O Controller Hub 9)“.*

Zvučna kartica ima „uređaje“ (engl. *device*) i „podуređaje“ (engl. *subdevice*) koji se razlikuju po nekoliko karakteristika. „Uređaj“, pri čemu je prvi određen oznakom 0, može biti *playback* tipa što znači da je u tom slučaju izlaz za zvuk iz računala ili može biti nekog drugog tipa poput *capture, control, timer* ili *sequencer.* Uređaj s brojem 0 se koristi ako nije specificiran neki drugi uređaj. „Podуređaj“, redoslijeda određenog jednako kao kod „uređaja“, označava neki značajan krajnji izlaz zvuka za uređaj, na primjer slušalice. Ako nije specificiran podуređaj ili je specificiran oznakom -1, koristi se bilo koji raspoloživi podуređaj.

Sučelje kartice je opis ALSA protokola za pristup kartici. Moguća sučelja uključuju *hw, plughw, default* i *plug:dmix*. Sučelje *hw* omogućava izravan pristup jezgri uređaja, ali bez podrške za softversko miješanje ili prilagođavanje prijenosa. *Plugin* i *default* omogućava izlaz zvuka u slučajevima kada bi *hw* sučelje dovelo do pogreške. Aplikacija uobičajeno opisuje izlaz zvuka kombinacijom svih navednih specifikacija zajedno u „*device stringu*“ koji može biti u jednoj od dvije dalje navedene forme, pri čemu je forma osjetljiva na razliku između malih i velikih slova:

1. interface:card,device,subdevice
2. interface:CARD=1,DEV=3,SUBDEV=2

ALSA prijenos („stream“) je niz podataka koji predstavljaju zvuk te je uobičajeno format prijenosa kodiran pulsko - kodnom modulacijom koja mora biti izvedena na način koji se podudara s karakteristikama ili parametrima samog hadrvera, što uključuje:

- frekvenciju uzorkovanja (engl. *„sampling rate“*) koja se razlikuje ovisno o funkciji – 44.1kHz za kućni stereo, 48kHz za kućno kino, 88.2kHz/96kHz/192kHz za hi-fi audio produkciju
- širinu uzorka mjerenu u broju bitova po uzorku (*bit/sample*)
- spremanje uzoraka većih od jednog bajta u memoriju (endianness)

- broj kanala koji se kreće od 1 za mono, preko 2 za stereo do 6 za AC-3/IEC958

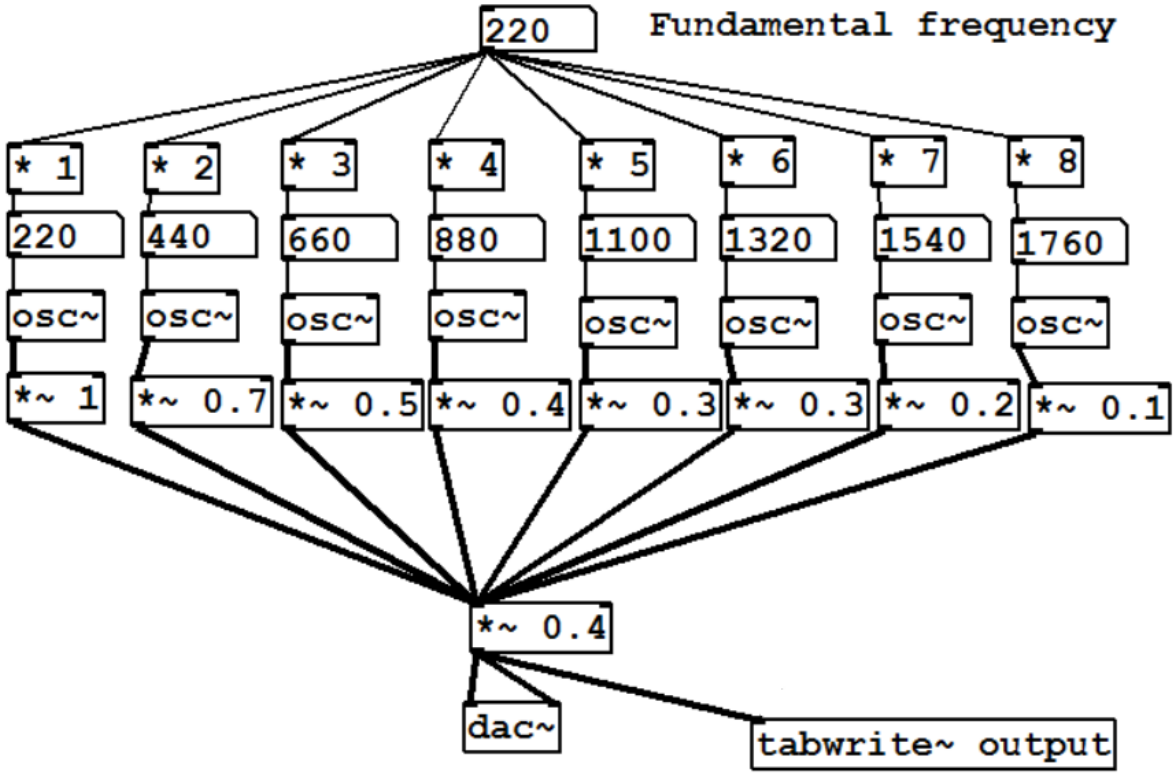
S obzirom da je u većini današnjih embedded sustava memorija organizirana u bajtovima, a riječi koje se spremaju u memoriji su veće od bajta, mora postojati način spremanja riječi u memoriju. Način spremanja riječi u memoriju naziva se endianness ili ponekad endianity te posjeduje dvije jednostavne mogućnosti i veliki broj njihovih varijacija. Dvije glavne mogućnosti su „little endian“ kod kojeg je najniži bajt spremljen na najnižoj adresi i „big endian“ kod kojeg je najviši bajt spremljen na najnižoj adresi. U većini slučajeva programer ne mora znati kako je riječ spremljena u memoriji, no primjerice kod prijenosa podataka preko mreže ili kad se podaci obrađuju na više reprezentacija softvera, programer mora biti upoznat s načinom pohrane riječi.

3. Grafički programski jezik PureData

PureData je grafički programski jezik za upravljanje tokom podataka namijenjen stvaranju glazbe i općenito multimedije pomoću računala u stvarnom vremenu. Uobičajeno se koristi za live performanse, kompoziciju, audio analizu i razmjenu podataka između senzora i programa. [8]

Budući da su sve mogućnosti prikazane kao digitalni podaci, postoji veliki broj mogućnosti za suradnju među njima. Zvuk se može koristiti za manipulirnje videa, koji može biti poslan preko interneta na drugo računalo koje može analizirati taj video za sljedeću upotrebu. Dizajniran je kako bi radio na svim većim operacijskim sustavima poput Linuxa, Windowsa, Andrioda i Apple operacijskih sustava te je program s otvorenim kodom (engl. *open-source*). Pripada klasi programskih jezika i okruženja koji koriste *proceduralni zvuk* (engl. *Procedural audio*), što znači da stvara glazbu iz rutina koje generiraju ili modificiraju nizove brojeva pomoću kojih hardver reproducira zvuk.

Zbog toga je programiranje u Pd slično manipuliranju objektima u stvarnom svijetu. Većina osnovnih funkcionalnosti je ostvarena *out of the box* te se program stvara jednostavnim spajanjem struktura u dijagrame koji predstavljaju tok podataka od ulaza do izlaza. Sami program se uvijek vrti kada je DSP aktiviran te nema razlike između pisanja i pokretanja programa što znači da svaki atom ima efekt u trenutku kada je stvoren i povezan. Navedeno se može vidjeti na slici 3.1 koja daje slikoviti prikaz stvaranja alikvotnih tonova za ton određene frekvencije. Osnovna frekvencija je jednostavno promjenjiva i zahvaljujući kućicama u drugom redu, automatski se mijenjaju i sve ostale frekvencije. Uz generiranje zvuka, Pd može obrađivati postojeći zvuk u digitalnom obliku što podrazumijeva da je stvarni zvuk iz vanjskog svijeta pretvoren u digitalni oblik.



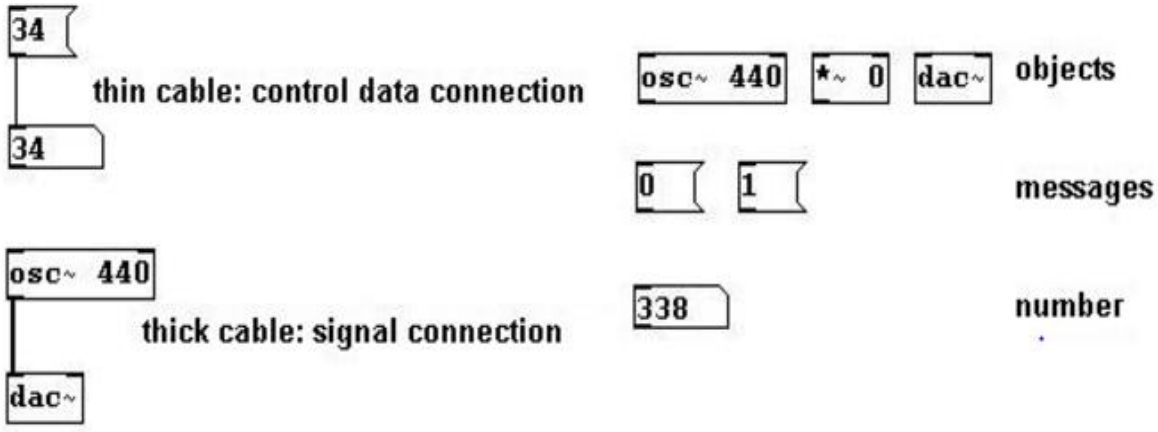
Slika 3.1 Primjer Pd programa

Budući da se radi o grafičkom programskom jeziku, za korisnika to znači da ne piše kod, već koristi vizualne blokove koje raspoređuje i međusobno povezuje na ekranu kako bi dobio željenu funkcionalnost. Blokovi se nazivaju atomima i svaki atom ima svoju funkcionalnost poput slanja poruke drugim atomima, brojčane vrijednost ili upravljanje objektima trećih strana. Svaki „kod“ naziva se *zakrpa* (engl. *patch)* koja je u memoriji tekstualna datoteka s informacijama o atomima i njihovoj međusobnoj povezanosti [9]. Prozor u kojem se stvaraju zakrpe i manipuliraju atomi naziva se *platno* (engl. *canvas*) po uzoru na slikarsko platno. Ukoliko ne postoji atom s funkcionalnosti koja nam je potrebna, moguće ju je napisati u više klasičnih programskih jezika poput C i Pythona. Postoji nekoliko različitih tipova objekata i veza među njima u programu, pri čemu su objekti:

1. Objekt (engl. *Object*) koji je najvažnija i najčešće korištena funkcionalna jedinica i u njemu se stvaraju funkcionalnosti poput oscilatora, DA i AD pretvornika, delay i sličnih. Izgledom je pravokutnik s dva ulaza, od kojih je lijevi signalni, a desni podatkovni te jedan ili nijedan izlaz
2. Poruka (engl. *Message*) interpretira zapis kao poruku koju predaju sljedećoj jedinici kada je kućica aktivirana dolazećom porukum ili klikom miša. Uvijek

ima jednu ulaznu i jednu izlaznu poziciju. Moguće je zapisivati različite vrijednosti poput float brojeva ili tekstualnih poruka bez potrebe za definicijom tipa podataka kao u C.

3. Broj (engl. *Number*) je jedina kućica čiji se sadržaj može mijenjati kada je *patch* pokrenut, bilo ručno ili da prikazuje trenutno stanje izvođenja. Primjerice na slici 3.1 osnovna frekvencija 220Hz može biti promjenjena pritiskom kućice i povlačenjem miša lijevo - desno
4. Simboli (engl *Symbol*) se koriste za unošenje ili prikazivanje tekstualnih zapisa (može uključivati i brojčani zapis). Šalje unesenu poruku na izlaz ili ispisuje poruku dobivenu na ulaz
5. Bang, Toggle i Slider kućice služe za kontrolu zvuka. Bang šalje kratak impuls kroz ostatak *patcha*, Toggle aktivira zvuk dokle god je označen, dok Slider kontrolira glasnoću



Slika 3.2 Objekti i veze među njima u Pd

Veze između objekata mogu biti signalne (engl. *Signal connection*) ili podatkovne (engl. *Control data connection*). Tanke veze predstavljaju podatkovne veze koje prenose isključivo kontrolne podatke te su to obično brojčane i tekstualne vrijednosti. Podatkovne veze mogu povezivati dvije kućice s kontrolnim podacima ili biti veza između kontrolne na jednoj strani i signalne na drugoj. Signalne kućice razlikujemo od kontrolnih po znaku tilda („~“) uz ime funkcije. Isključivo dvije signalne kućice mogu biti međusobno povezane debljom, signalnom vezom. Signalne kućice su one koje produciraju zvuk, kontrolne daju parametre za obradu. Uz gore navedene elemente i veze među njima, Pd poput svakog drugog

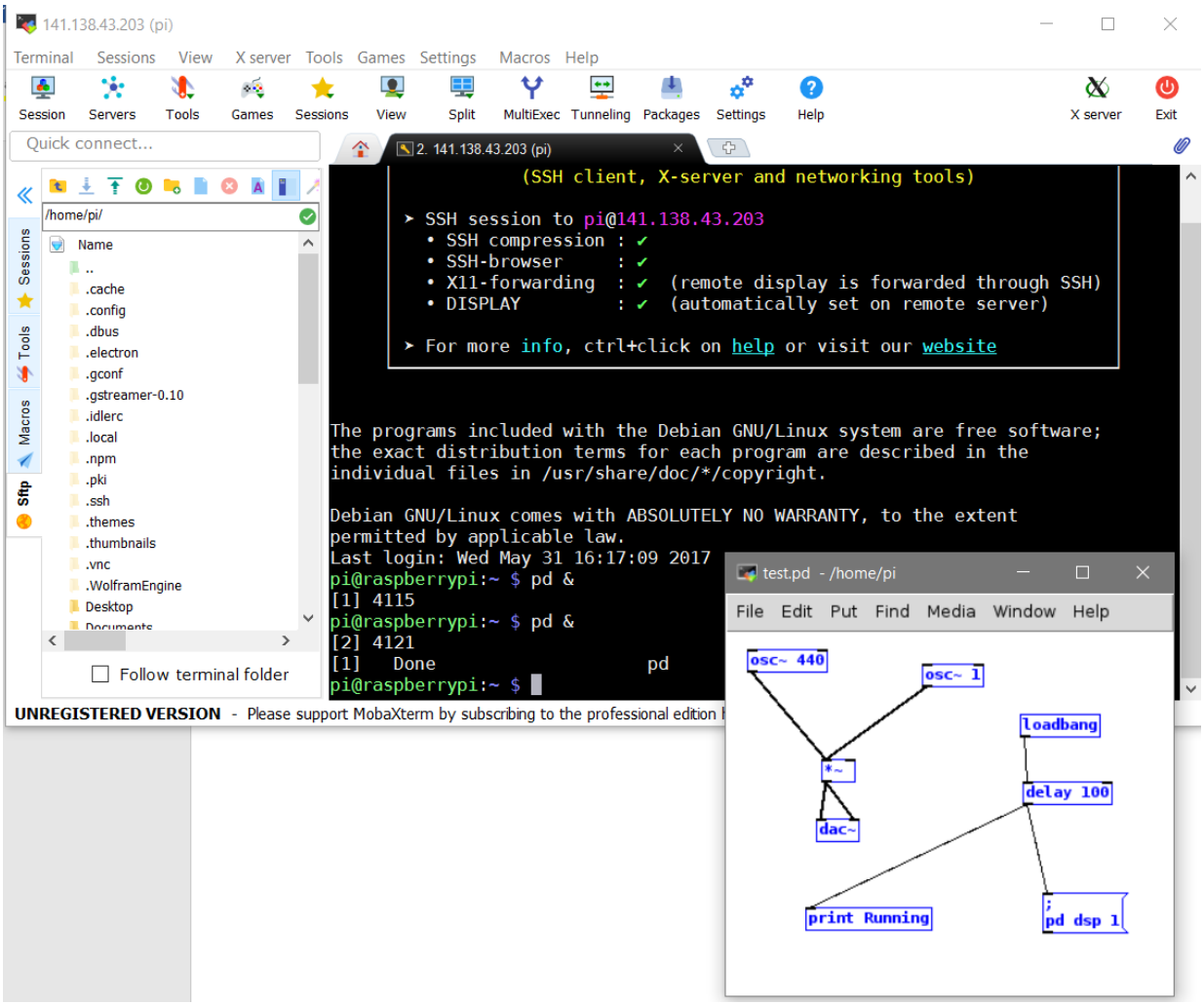
programskog jezika sadrži komentare. S obzirom da *patch* vrlo lako može postati kompleksan, komentari su vrlo važna sastavnica kako bi bilo lako pratiti tok podataka samom autoru, kao i drugim zainteresiranim stranama te se iz tog razloga u svakom programskom jeziku, pa tako i PureData, smatraju pristojnom programerskom praksom.

3.1. Pure Data i Raspberry Pi

Pure Data dolazi u dvije verzije, *Vanilla* i Extended. Budući da raspolaže većim brojem objekata, apstrakcija ili GUI-plugina, Pd-extended je za prve projekte jednostavniji. Iako je Pd-extended lakši za korištenje, korisno je znati se koristiti i s Pd-vanilla. Vanilla je obično malo aktualnija verzija jer je manje vremena potrebno za razvoj manjeg broja objekata. Kada se Pure Data koristi za Raspberry Pi, limitiran je isključivo na Vanilla verziju. Ranije je navedeno kako RPi nije namijenjen obradi audio signala te da su iz tog razloga njegove karakteristike u tom području loše. S obzirom da obrada audia na Rpi zahtjeva većinu njegove procesorske moći i potreban nam je u načinu rada bez grafičkog sučelja i periferalnih komponenti, a PureData je grafički programski jezik, potreban je drugačiji pristup radu od uobičajenog. Nekoliko je koraka u postizanju takvog pristupa radu RPi-ja:

1. Realizacija udaljenog pristupa Raspberry Pi-ju preko internetske mreže SSH protokola
2. Gašenje grafičkog sučelja te odspajanje kompletne periferije (ekran, tipkovnica i miš) pri čemu jedino spojeno ostaje Ethernet kabel i napajanje
3. Instalacija ALSA drivera za zvuk
4. Instalacija PureData
5. Testiranje

Budući da se radi o udaljenom pristupu, Raspberry Pi-ju pristupamo pozivanjem njegove IP adrese u terminalu računala koje nam služi za pristup ili posebnim programom poput MobaXterm koji nam omogućava korištenje PureData s grafičkim karakteristikama bez potrebe za ekranom.



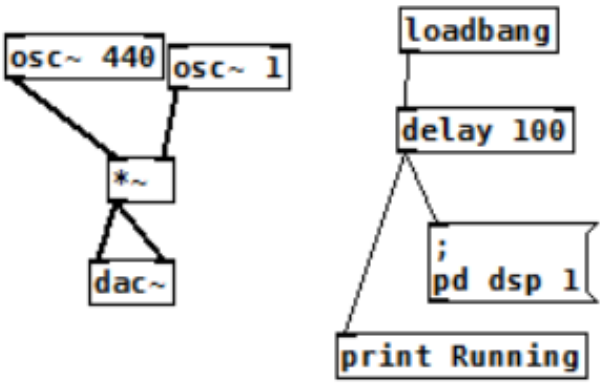
Slika 3.3 Izgled MobaXterm s primjerom Pd *patch*-a

Također, postoji specifičnost vezana uz Pd na RPi kada se Pd pokreće iz terminala bez grafičkog sučelja. Kada se Pd pokreće naredbom

```
pd -alsa -nogui test.pd
```

javlja se greška:

```
snd_pcm_open (input): Device or resource busy

snd_pcm_open (output): Device or resource busy
```



Slika 3.4 *patch* za -nogui pokretanje na RPi

Navedena pogreška je posljedica nepodizanja grafičkog sučelja koje se događa inače prije pokretanja DSP. Kako bi se ta pogreška uklonila, dodaje se atom koji simulira kašnjenje od 100ms, što je vrijeme koje bi bilo potrebno grafičkom sučelju za podizanje. Na taj način RPi smatra da je grafičko sučelje odradilo svoje konfiguracije i spreman je započeti digitalnu obradu signala.

4. Realizacija na Raspberry Pi 2

U ovom poglavlju će biti objašnjeno kako je realizirana obrada audio signala na Raspberry Pi 2 razvojnoj platformi s dodatnom zvučnom karticom AudioInjector uz upotrebu Pure Data programa. U prethodnom poglavlju su objašnjeni koraci za pravilan rad Pure Data na Raspberry Pi-ju koji će biti detaljnije pojašnjeni, kao i njihova realizacija i testiranje. Dalje će biti prikazani rezultati rada u Pure Data te će biti dana usporedba kvalitete zvuka na Raspberry Pi bez dodatne zvučne kartice i s dodatnom karticom.

4.1. Konfiguracija sučelja

Početna konfiguracija sučelja za ovaj projekt uključuje podešavanje udaljenog pristupa preko SSH protokola te uklanjanje grafičkog sučelja i periferije (ekran, miš i tipkovnica). Ova dva koraka su povezana jer postavke za SSH protokol ne možemo riješiti bez periferije, a uklanjanje periferije omogućuje korištenje ukupne procesorske moći za projekt. Pokrivena su dva slučaja za pristupanje Raspberry Pi-ju preko mreže pri čemu prvi slučaj podrazumijeva da su računalo preko kojeg pristupamo i Raspberry Pi u istoj mreži, dok drugi podrazumijeva različite mreže. Za prvi slučaj je rješenje vrlo jednostavno i uključuje postavljanje statičke IP adrese za RPi. Postavljanje statičke IP adrese uključuje nekoliko koraka koji će biti objašnjeni zajedno sa svojim *terminal* naredbama:

1. Omogućavanje povezivanja preko SSH na Raspberry Pi

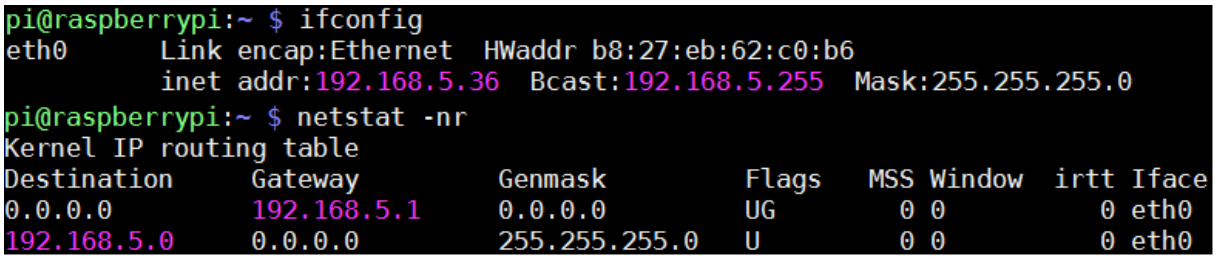
   ```
   Sudo raspi-config -> ssh enable
   ```

2. Provjera trenutne konfiguracije naredbom

   ```
   nano /etc/network/interfaces
   ```

   koja pokazuje da se adresa dinamički alocira preko *router*-a DHCP protokolom što je potrebno promijeniti u statičku

3. Prikupljanje informacija o ethernet vezi naredbama `ifconfig` i `netstat -nr`



Slika 4.1 Podaci o ethernet vezi

4. Dodavanje statičke IP adrese u `/etc/network/interfaces`

   ```
   iface eth0 inet static

   address 192.168.5.36

   netmask 255.255.255.0

   network 192.168.5.0

   broadcast 192.168.5.255

   gateway 192.168.5.1
   ```

   pri čemu se koriste podaci iz prva dva koraka, iako je za `address` zadnji broj moguće staviti bilo koji do 255 ukoliko nema drugog uređaja na toj adresi

5. Gašenje i ponovno podizanje sustava te provjera postavki u `/etc/network/interfaces` koje bi trebale biti jednake postavkama iz prethodnog koraka

U drugom slučaju rješenje je nešto kompleksnije zbog nemogućnosti postavljanja statičke IP adrese te je to rješeno pomoću skripte koja jednom dnevno, ako je Raspberry Pi aktivan, šalje trenutnu adresu preko koje se može pristupiti. Kao što je prikazano na slici 4.1, skripta je vrlo jednostavna, no ispunjava svoju funkciju i omogućava pristup iz vanjske mreže. Nakon konfiguracije SSH protokola i testiranja pristupa na oba načina, možemo odspojiti periferiju i nadalje sve raditi preko udaljenog pristupa.

```
#!/bin/bash

MOJ_IP=$(curl ifconfig.me)

echo "${MOJ_IP}" | mail -s "Moj trenutni ip" petra.kastrapeli@gmail.com

exit 0
```

Slika 4.2 Skripta za IP

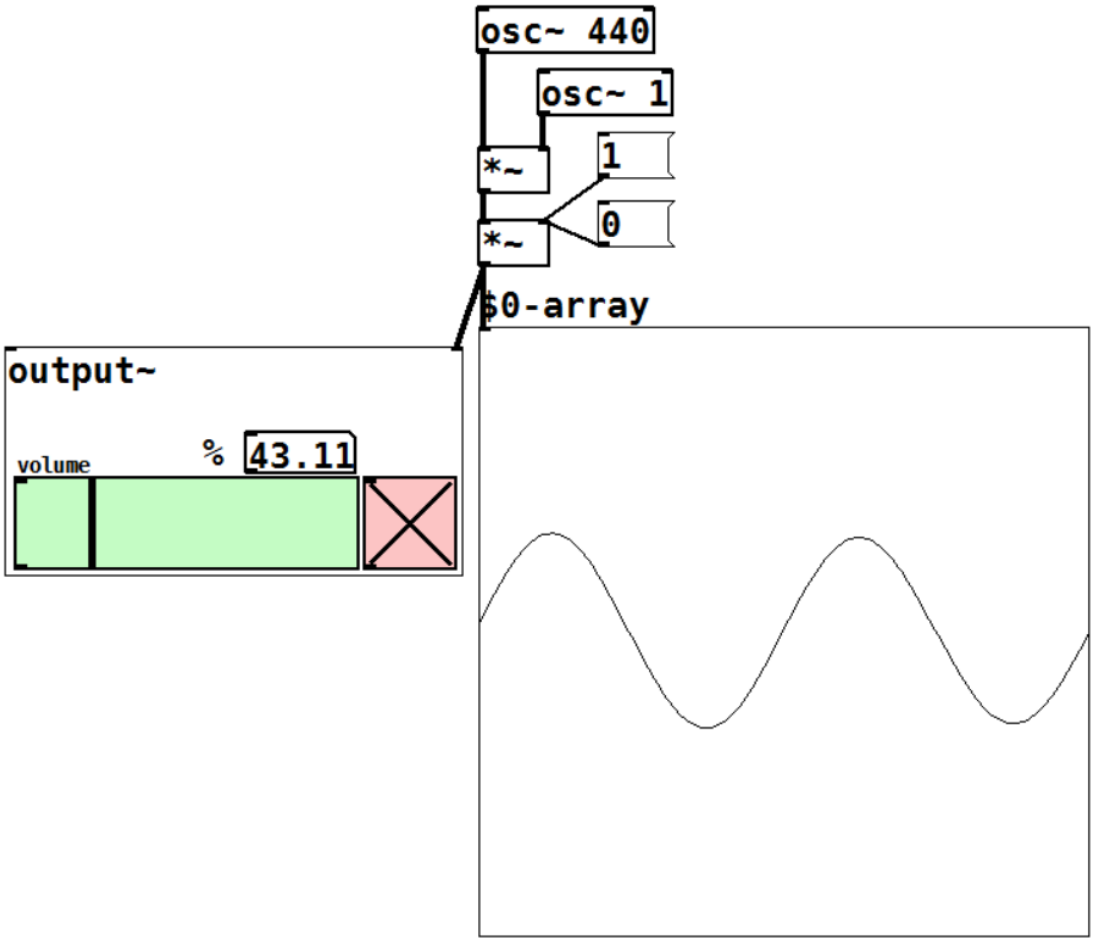
Budući da je Pure Data grafički programski jezik, a SSH pristup se uobičajeno koristi preko terminala, potrebno je instalirati neki program koji će nam omogućiti rad u Pure Data na grafičkoj razini. Kao što je opisanu u prethodnom poglavlju, odabrani program MobaXTerm omogućava grafičko sučelje bez potrebe za monitorom. Na taj način može se programirati direktno na Raspberry Pi, umjesto na računalu pa premještati datoteke, te je svaku promjenu moguće testirati.

Kada je uspješno testiran udaljeni pristup i periferija odspojena, može se započeti konfiguracija samog Raspberry Pi-ja za audio performanse. Prvi korak u tome je instalacija i testiranje ALSA drivera za zvuk:

1. Instalacija ALSA drivera preko terminala `Sudo apt-get install alsa-utils`
2. Aktivacija drivera `sudo modprobe snd_bcm2835`
3. Testiranje drivera zvučnom datotekom koja je dio instalacijskog paketa

   `sudo aplay /usr/share/sounds/alsa/Front_Center.wav`

Nakon što je to testiranje uspješno provedeno, sljedeći korak je instalacija PureData i testiranje *patch*-a. Pure Data dolazi u dvije verzije, *Vanilla* i *Extended,* pri čemu se u ovom projektu koristi *Vanilla* verzija koja zauzima manje memorije te je zbog toga prikladnija za Raspberry Pi. Za testiranje je kreiran jednostavan *patch* koji testira isključivo pojavljuje li se zvuk na izlazu. Testni *patch,* na slici 4.3, radi amplitudnu modulaciju, s frekvencijom prijenosa inicijalno postavljenom na 440 Hz s mogućnošću oscilacije frekvencije i frekvencijom modulacije od 1 Hz. Ovakav test omogućava testiranje zvučne kartice bez kompleksnog *patch*-a.



Slika 4.3 *patch* test.pd

S obzirom da je cjelokupan projekt izveden s dodatnom zvučnom karticom AudioInjector, potrebno je konfigurirati istu te ju testirati na jednaki način kad Raspberry Pi:

1. Fizičko spajanje Raspberry Pi i AudioInjector kartice preko 40-pinskog GPIO
2. Konfiguriranje /boot/config.txt datoteke kako ne bi koristila klasični Raspberry Pi izlaz, nego zvučnu karticu

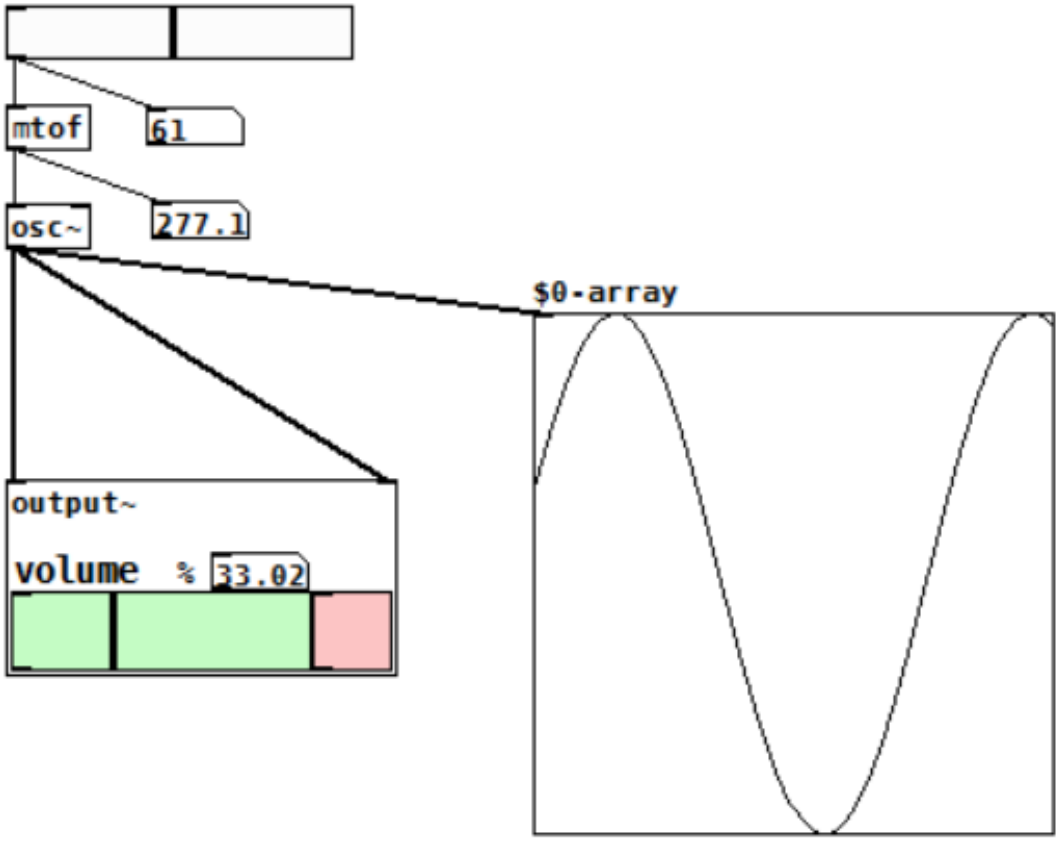
```
#Enable audio (loads snd_bcm2835)

#dtparam=audio=on

dtoverlay=audioinjector-wm8731-audio
```

3. Omogućavanje korištenja mikrofona i izlazne linije u Alsamixeru pomoću MobaXTerm

Nakon što su provedena sva testiranja za oba slučaja, može se prijeći na programiranje i testiranje kodova u oba okruženja. Kodovi će biti podijeljeni u tri dijela ovisno o tome što testiraju. Prvi dio testira funkcionalnosti izlazne linije i mogućnost generiranja zvukova u Pure Data.

4.2. Generiranje zvuka

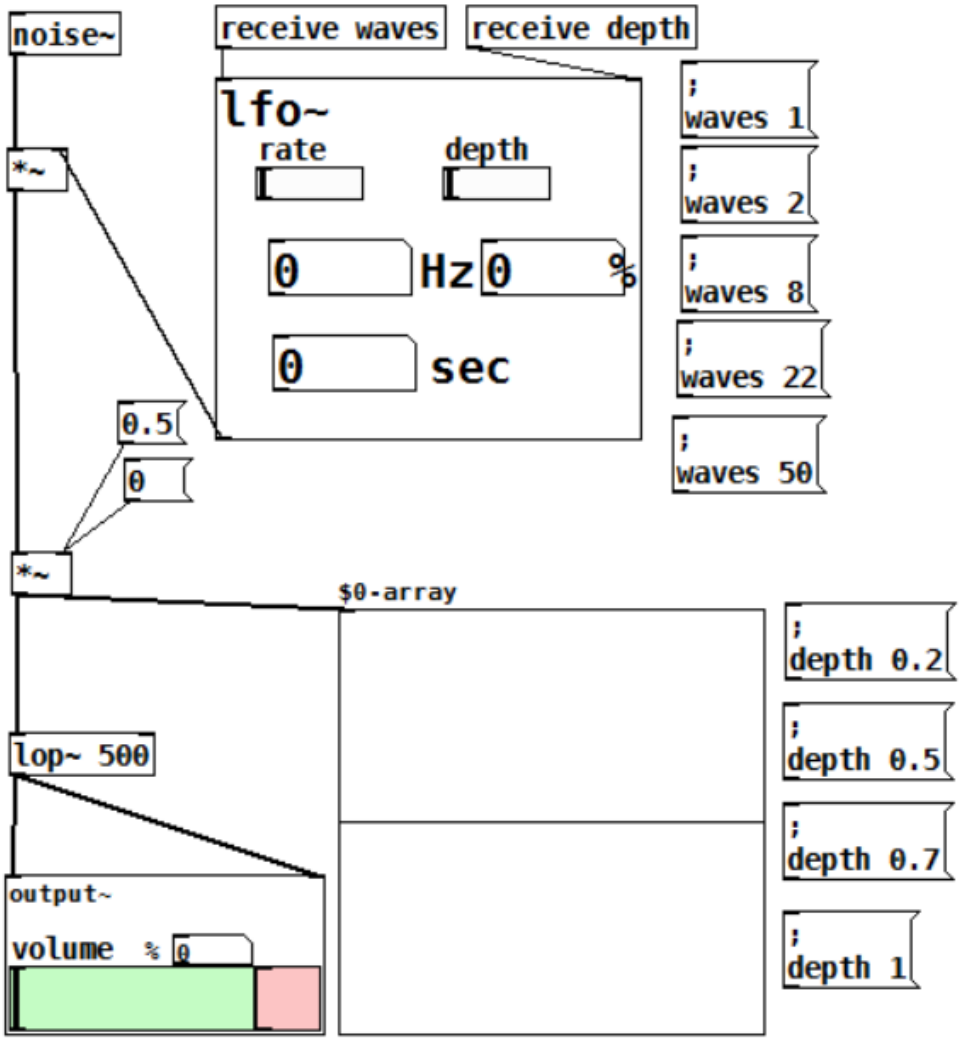
Inicijalna namjena Pure Data je upravo generiranje zvukova i manipulacija istima. Budući da računalo „razmišlja“ matematički, osnovni zadatak za generiranje zvuka je nalaženje poveznice između glazbe koju želimo i njezinog matematičkog modela.



Slika 4.4 *patch* MIDI to freq

Prvi primjer toga je *patch* na slici 4.4 koji prikazuje vrlo jednostavno generiranje zvuka pomoću vodoravnog *slider*a koji generira MIDI tonove. Vrijednost MIDI tonova se kreće između 0 i 127 pri čemu 0 predstavlja najnižu notu koju je moguće odsvirati, a 127 najvišu. Budući da se najniže vrijednosti nalaze na frekvencijama koje ljudsko uho ne može registrirati, njih možemo čuti ako dopunimo ton njegovim

alikvotnim frekvencijama. No takvi tonovi nisu zanimljivi jer ih ljudsko uho ne može čuti te postoji minimalan broj instrumenata koji ih mogu izvesti pri čemu je najniži ton koji neki instrument stvara na frekvenciji 8 Hz za orgulje.

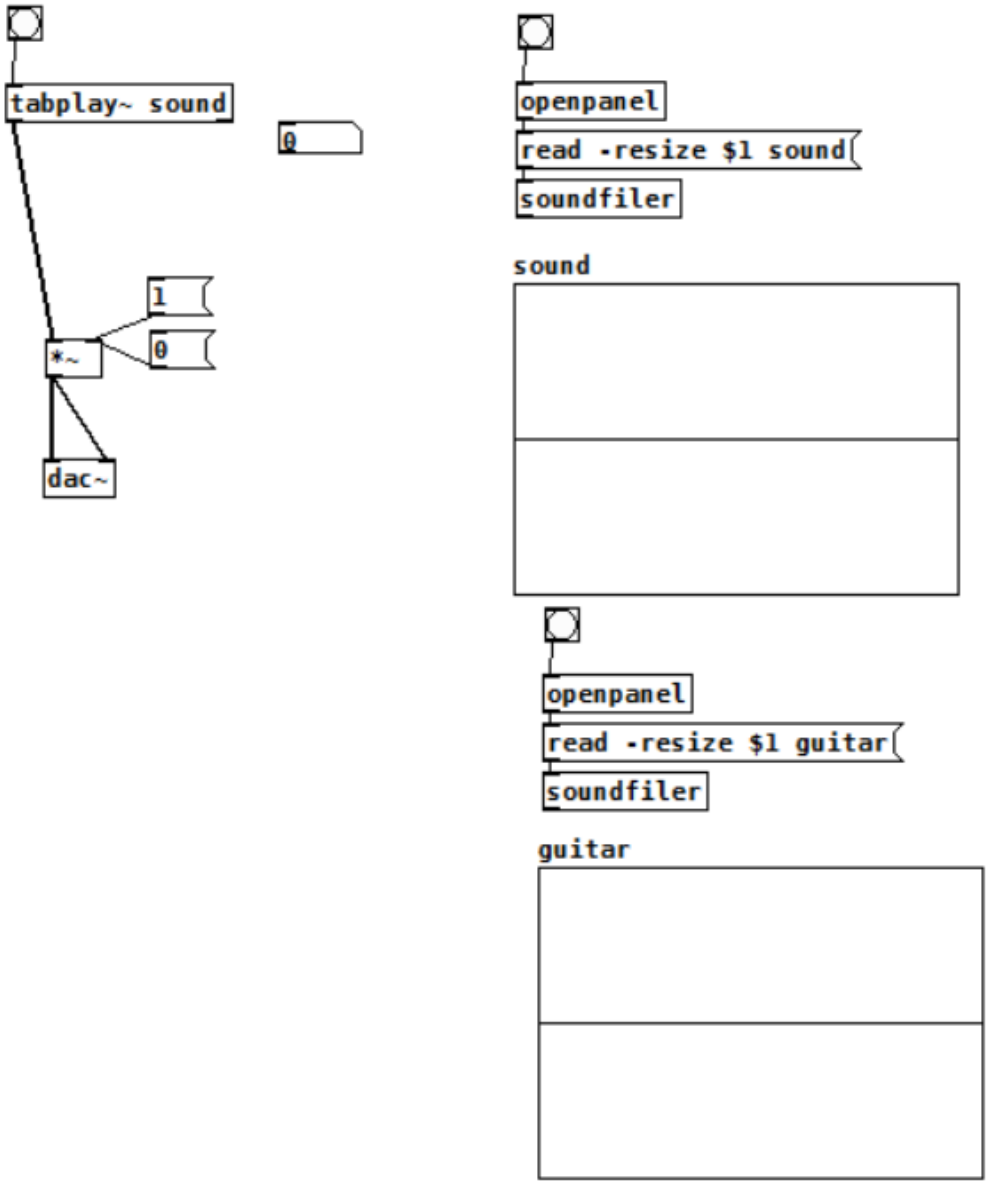


Slika 4.5 *patch* waves.pd

Ovaj *patch* testira mogućnosti za generiranja zvukova koje pruža šum, pa je tako pomoću ovog moguće generirati zvuk šuma valova, stare parne lokomotive ili helikoptera u letu. Atom lfo~ je *subpatch*, što znači da je stvoren od osnovnih atoma, spremljen u memoriju kao lfo~.pd te se može u drugim *patchevima* koristiti na isti način kao osnovni atomi. Prednost toga što je tako isti *subpatch* može koristiti u više *patcheva* bez potrebe za kopiranjem svih atoma. Za crtanje grafa koji prikazuje šum amplitudno moduliran niskofrekvencijskim oscilatorom je također napisan *subpatch*. Svi *subpatchevi* korišteni u ovom projektu prikazani su u Dodatku ovog rada, no njihove funkcionalnosti će biti dodatno pojašnjeni na

mjestima gdje se pojavljuju u *patch*-evima. *Subpatch* lfo~ je niskofrekvencijski oscilator koji može primati dvije ulazne vrijednosti ili ulazni parametri mogu biti određeni pomoću vodoravnih *slider*-a. Ulazni parametar *rate* je frekvencija ovojnice modulacije kojoj se u ovom slučaju vrijednost može postaviti između 0.05 i 100 Hz. Kada se ne koristi *slider*, može se na ulaz staviti parametar koji može biti točno određeni broj u posebnom atomu, koji nam omogućuje da je jedan klik dovoljan za postavljanje na tu frekvenciju. U ovom slučaju te frekvencije su 1, 2 8 i 22 Hz, od kojih svaka ima svoj efekt. Efekti se dobivaju u kombinaciji s drugim parametrom, *depth*, koji određuje amplitudu ovojnice između 0 i 1. Moguće je podesiti i druge frekvencije i amplitude oscilacije, no točno zapisane oscilacije daju točno određene efekte.

4.3. Manipulacija snimljene datoteke i zvuka s mikrofona



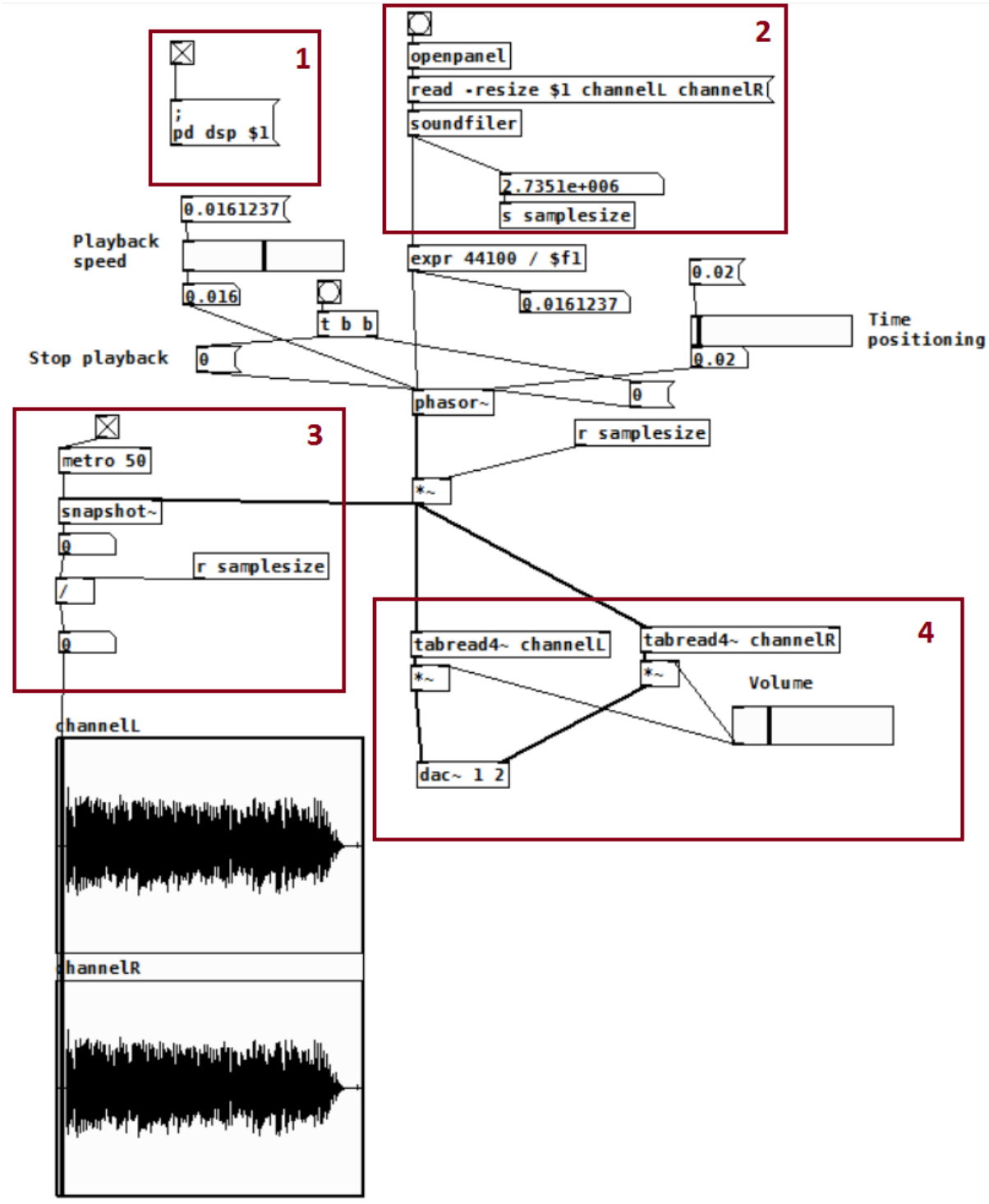
Slika 4.6 *patch* za otvaranje datoteka

Slika 4.6 prikazuje jednostavno otvaranje snimljene datoteke te isključivo njezino preslušavanje. Ova dateteka je temelj za nastavak rada, s obzirom da je potrebno svaku datoteku otvoriti ako želimo s njom raditi. Na ovaj način moguće je učitati dvije glasovne datoteke i jednostavnom promjenom parametra u atomu `tabread~`. `Tabread~` atom čita iz grafa koji ima isto ime kao atom te je na taj način lako mjenjati s kojim grafom radimo.

Kompleksnija verzija prikazana je na slici 5. te će biti objašnjena po segmenitma označenima brojevima na slici i neoznačeni segment kao jezgra *patcha*:

1. Ovdje je prikazana aktivacija i deaktivacija digitalnog procesora unutar programa. Osnovni prozor ima kućicu koja označena aktivira DSP, no pisanje i testiranje je jednostavnije ako se ta funkcionalnost može kontrolirati iz *patch*a
2. Na pritisak okrugle tipke *bang* otvara se prozor u kojem se odabire zvučna datoteka, idealno u .wav formatu. Atom `-resize $1` mijenja veličinu polja u koje se upisuje prema broju uzoraka u datoteci te je taj broj zapisan u kasnijem atomu (trenutna vrijednost $2{,}74*10^6$, maksimalna vrijednost $4*10^6$) i ovisi o trajanju zapisa
3. Aktivacijom metronoma pokreće se praćenje broja uzorka na kojem se izvođeni zapis nalazi. Dijeljenjem s brojem uzoraka dobiva se vrijednost između 0 i 1 kojom se upravlja vodoravnim *slider*om koji je preklopljen s grafovima i prikazuje slikovito tijek izvođenja.
4. Tabread4~ je modificirana verzija naredbe tabread~ na način da interpolira susjedne vrijednosti te je na taj način prikladnija za čitanje iz polja

Središnji dio *patch*a je glavna sastavnica u kojem se događa sva manipulacija sa zvukom. Naredba `expr 44100 / $f1` određuje brzinu čitanja iz polja idućom naredbom `phasor~` koja daje signal s vrijednostima između 0 i 1 što omogućava manipulaciju brzine izvođenja i trenutka od kojeg se započinje izvođenje. Lijeva strana prikazuje dio za određivanje brzine izvođenja koji se može postaviti između -1 i 1 pri čemu negativne vrijednosti izvode snimku unazad. Tu se, također, nalazi *bang* koji zaustavlja izvođenje i vraća snimku na početak. Na desnoj strani nalazi se dio za pozicioniranje u vremenu s fiksnom vrijednosti postavljenom na 0.02 s obzirom da snimka ne počinje točno u trenutku početka snimanja.



Slika 4.7 *patch* za manipulaciju zvuka

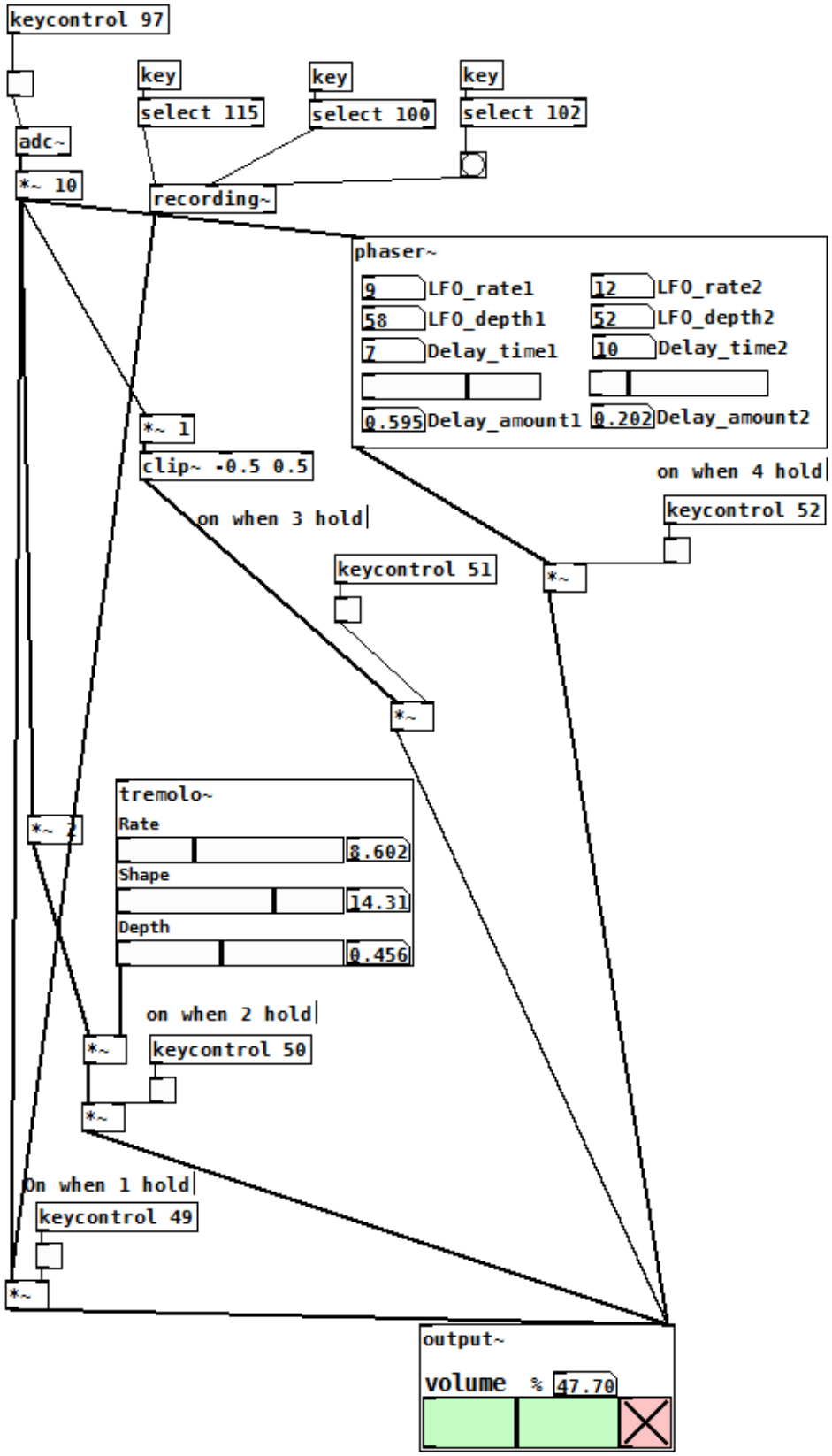
Iako je ranije bilo navedeno da se u ovom projektu RPi koristi bez bilo kakvog vanjskog sučelja, sljedeći *patch*, prikazan na slici 4.7 će pokazati da je korištenje tipkovnice ipak moguće kako bi kontrolirali koji dio *patcha* se koristi i kako bi upravljali izvorom zvuka. Za izvor zvuka postavljene su dvije mogućnosti pri čemu je jedna zvuk s mikrofona, dok je druga zvučna datoteka iz memorije RPI. Sami

patch se može podjeliti u dva dijela gdje prvi uključuje izbor izvora, a drugi izbor efekta koji na zvuk primijenjujemo. Za izbor u oba dijela koristi se tipkovnica, brojčane vrijednosti od 1 do 4 za izbor efekata i slova, a za ulaznu liniju s mikrofona, f za otvaranje snimljene datoteke, s za pokretanje snimljene datoteke i d za zaustavljanje snimke. Rješenje nije idealno jer je sve tipke koje želimo potrebno držati cijelo vrijeme dok želimo da se aktivnost događa, dok bi bolje rješenje bilo da kratak pritisak tipke mijenja stanje on/off.

Neovisno o tome odabere li se kao ulaz mikrofon ili spremnjena snimka, efekti koje se može raditi nad zvukom su isti te će biti objašnjeni redosljedom ovisno o tome koji ih broj aktivira:

1. Ne stvara nikakav efekt, već služi isključivo za preslušavanje snimke. Realizirano je ovdje kako bi se svi ostali efekti mogli uspoređivati na temelju originala i kako korisnik ne bi moralo pamtiti zvuk originala, već ga preslušati.
2. Tremolo efekt koristi LFO za kreiranje signala koji oscilira amplitudu zvučnog signala te ima najmanje dvije, u ovom slučaju tri karakteristične vrijednosti, *Rate* koja omogućuje sinkroniziranje ili odmak od istog sa izvođenom snimkom, *Depth* koji određuje maksimalnu glasnoću tonova i, u ovom slučaju, *Shape* koji određuje oblik ovojnice, točnije, koliko se ista približava 0 u trenucima najmanje vrijednosti
3. Ovaj efekt reže sve vrijednosti amplitude veće od 0.5 i manje od -0.5 što dovodi do velike distorzije zvuka
4. Posljednji efekt je „efekt skupnog muziciranja“, odnosno manipulacija zvukom kako bi se od snimke koju izvodi jedan istrument dobio osjećaj izvođenja više jednakih instrumenata, u ovom slučaju gitare, ili promjenom parametara efekt prirodne jeke u prostoriji

Efekti se mogu izvoditi i istovremeno te se na taj način može promatrati kako efekti utječu jedni na druge ili kako efekti izobličuju zvuk u odnosu na originalni



Slika 4.8 *patch* za efekte

.

4.4. Kvaliteta signala i daljnje mogućnosti razvoja

Ranije je spomenuto kako RPi nije namijenjen obradi audio signala te je stoga kvaliteta izlaznog zvuka niska. Ovdje će biti rečeno što točno niska kvaliteta zvuka znači te poboljšanja koja su dobivena zvučnom karticom. Sami RPi ima bolju kvalitetu zvuka na frekvencijama do 1kHz, na kojima se THD+N kreće oko 1%, no nakon toga brojka naglo raste i kvaliteta zvuka je lošija. RPi kao standardnu modulaciju koristi pulsno-širinsku modulaciju dok zvučna kartica koristi pulsno-kodnu modulaciju koja ima manji unos šuma i manji gubitak podataka te je time kvalitetnija. Neke od ostalih prednosti korištenja zvučne kartice za bolju kvalitetu zvuka su viša maksimalna frekvencija uzorkovanja (92 kHz, za RPi 48 kHz) te veća širina riječi (32 bita). Najbitnija stavka, iza koje stoje svi brojevi, je ipak percepcija za ljudsko uho te je kvaliteta određena na taj način, sa zvučnom karticom, vrlo slična onoj koja se dobiva iz osobnog računala.

Ovaj rad se bavio mogućnošću korištenja Raspberry Pi-ja u kombinaciji s Pure Data za jednostano rješenje za manipulaciju zvuka. U budućnost bi se sva rješenja navedena u ovom radu mogla proširiti te, za posljednji *patch* prikazan na slici 4.8, priključiti ulaznu jedinicu sa sklopkama kojom se može upravljati nogom kako bi ruke ostale slobodne za sviranje. Također mogućnost poboljšanja za isti *patch* je uklanjanje potrebe konstantnog pritiska tipke za aktivnost efekta, već da se navedeno rješi kratkim pritiskom i promjenom stanja na taj način.

Zaključak

Cilj ovog rada bio je istražiti mogućnosti korištenja Raspberry Pi 2 u kombinaciji s dodatnom zvučnom karticom i programom Pure Data. Proučene su karakteristike općenite digitalne obrade signala, zatim digitalne obrade audio signala te, u konačnici, digitalne obrage signala na ugradbenim računalim sustavima. Argumentirani su razlozi za korištenje Raspberry Pi 2 uz dodatnu zvučnu katicu za ovakav projekt umjesto korištenja ugradbenog sustava namijenjenog isključivo za obradu signala.

Prikazane su sastavnice grafičkog programskog jezika Pure Data i njegovo korištenje u obradi audio signala. Stvoren je veći broj *patcheva* u navedenom programu koji pokrivaju tri različite mogućnosti manipulacije signala: generiranje zvuka isključivo pomoću sastavnica programa bez vanjskih datoteka, obrada i manipulacija prethodno snimljene zvučne datoteke u .wav formatu te obrada zvuka koji dolazi s ugrađenog mikrofona na zvučnoj kartici.

Diskutirana su potencijalna buduća poboljšanje sustava, kao i kvaliteta izlaznog signala. Predloženo poboljšanje uključuje aktivaciju i deaktivaciju zvučnog efekta kratkim pritiskom tipke bez potrebe za konstantnim pritiskom i uključivanje tipkovnice upravljane nogom umjesto klasične tipkovnice.

Literatura

[1] Prao, P. 2007. *Audio Signal Processing*. Department of Electrical Engineering, Indian Institute of Technology Bombay, India.

[2] Oshana, R. 2006. *DSP Software Development Techniques for Embedded and Real-Time Systems*. Newnes, London

[3] Raspbery Pi Official page

[4] Phillips Semiconductors,1986, *I2S bus*

[5] eLinux, URL: http://elinux.org/RPi_Hub, pristupljeno: 20.3.2017.

[6] AudioInjector, URL: http://www.audioinjector.net/rpi-hat, pristupljeno: 14.3.2017.

[7] Alsa, URL: http://alsa-project.org/main/index.php/Main_Page, pristupljeno: 10.3.2017.

[8] Kreidler, J. 2008. *Programming Electronic Music in Pd,* http://www.pd-tutorial.com/

[9] Hillerson, T., 2014, *Programming Sound with Pure Data*, The Pragmatic Bookshelf, Dallas, Texas, SAD

Naslov, sažetak i ključne riječi

NASLOV: Obradba i manipulacija glazbe pomoću ugradbenih sustava

SAŽETAK: Ovaj rad bazira se na korištenju Raspberry Pi 2 za obradu audio signala bez ili s dodatnom zvučnom karticom. Za obradu signala koristi se grafički program Pure Data. Prikazana je mogućnost generiranja zvukova unutrar navedenog programa te mogućnost manipulacije snimljenim zvučnom datotekom i zvukom prikupljenim s mikrofona zvučne kartice. Diskutirana su poboljšanja koja zvučna kartica unosi u kvalitetu izvođenja zvuka te poboljšanja koja bi se u budućnosti mogla uvesti za efikasnije korištenje cijelog sklopa.

KLJUČNE RIJEČI: Raspberry Pi 2, GPIO zvučna kartica, Pure Data, digitalna obrada signala

Title, summary and keywords

TITLE: Processing and Manipulation of Music using Embedded Systems

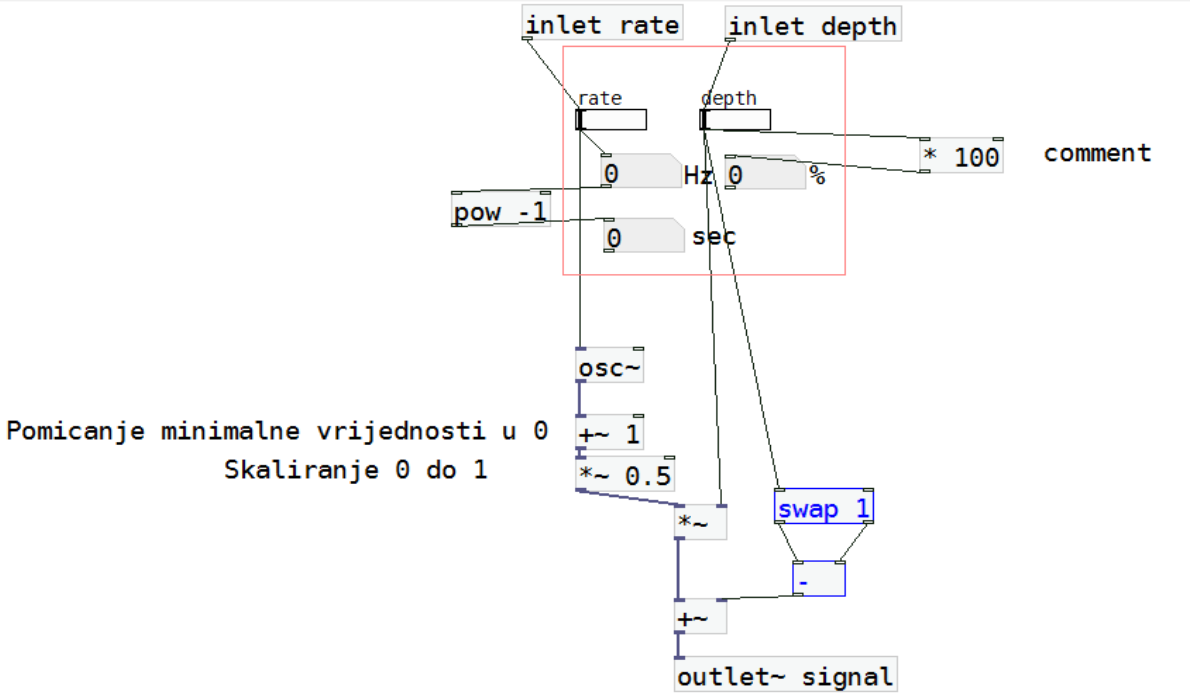
SUMMARY: This paper is focused on using Raspberry Pi as Digital Signal Processor, with or without external sound card. For signal processing a graphical programming language called PureData is used. Possibility for generating sound purely atoms inside program is shown and possibility for manipulating redorded audio file of sound coming directly from microphone on sound card.

KEYWORDS: Raspberry Pi 2, Pure Data, GPIO sound card, digital signal processing

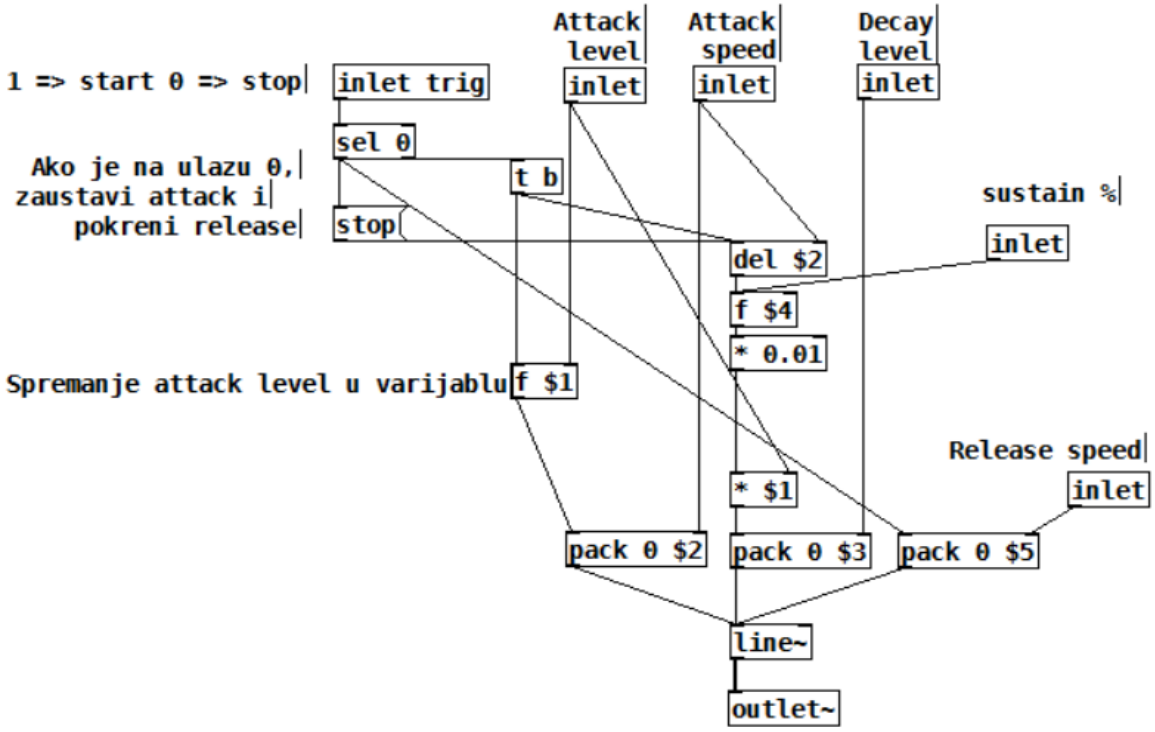
Skraćenice

RPi	*Raspberry Pi 2*
Pd	*Pure Data*
GPIO	*General Pupose Input Output (*hrv. *Ulazno-izlazno sklopovlje)*
THD+N	*Total Harmonic Distortion plus Noise* (hrv. *Ukupno harmonijsko izobličenje plus šum*)

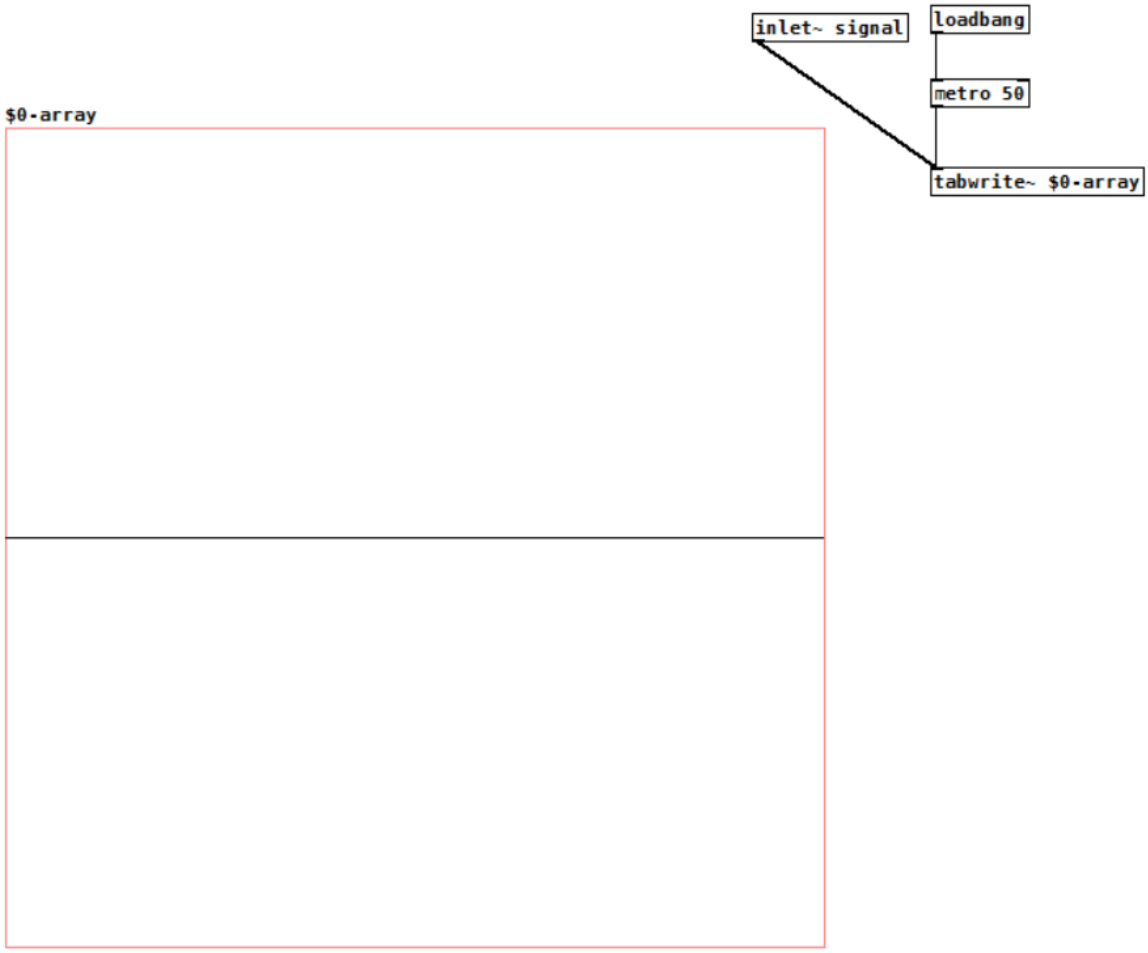
Dodatak



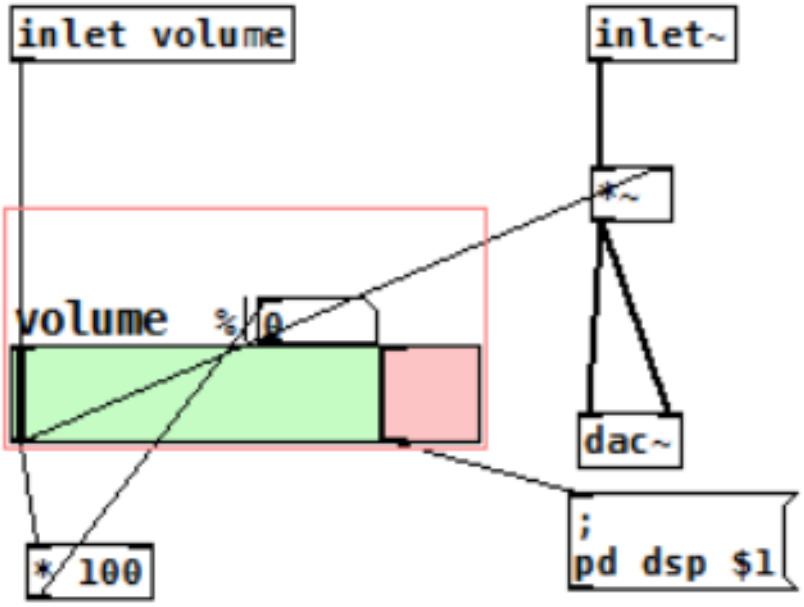
lfo~.pd


1 => start 0 => stop
inlet trig
Attack level
inlet
Attack speed
inlet
Decay level
inlet
sel 0
Ako je na ulazu 0, zaustavi attack i pokreni release
t b
stop
sustain %
inlet
del $2
f $4
* 0.01
Spremanje attack level u varijablu
f $1
Release speed
inlet
* $1
pack 0 $2
pack 0 $3
pack 0 $5
line~
outlet~

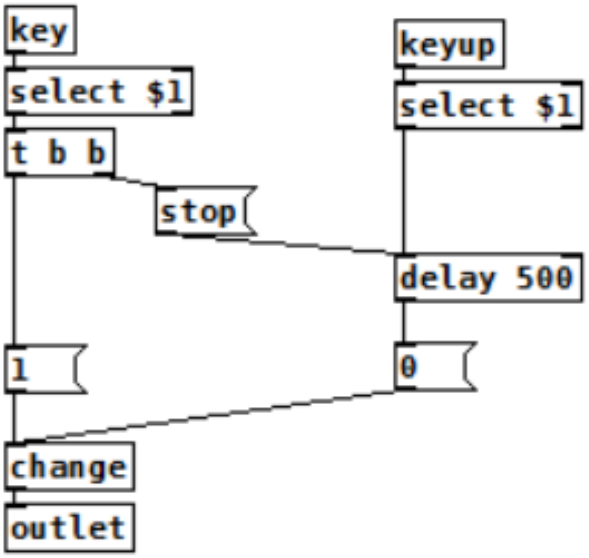
adsr~.pd



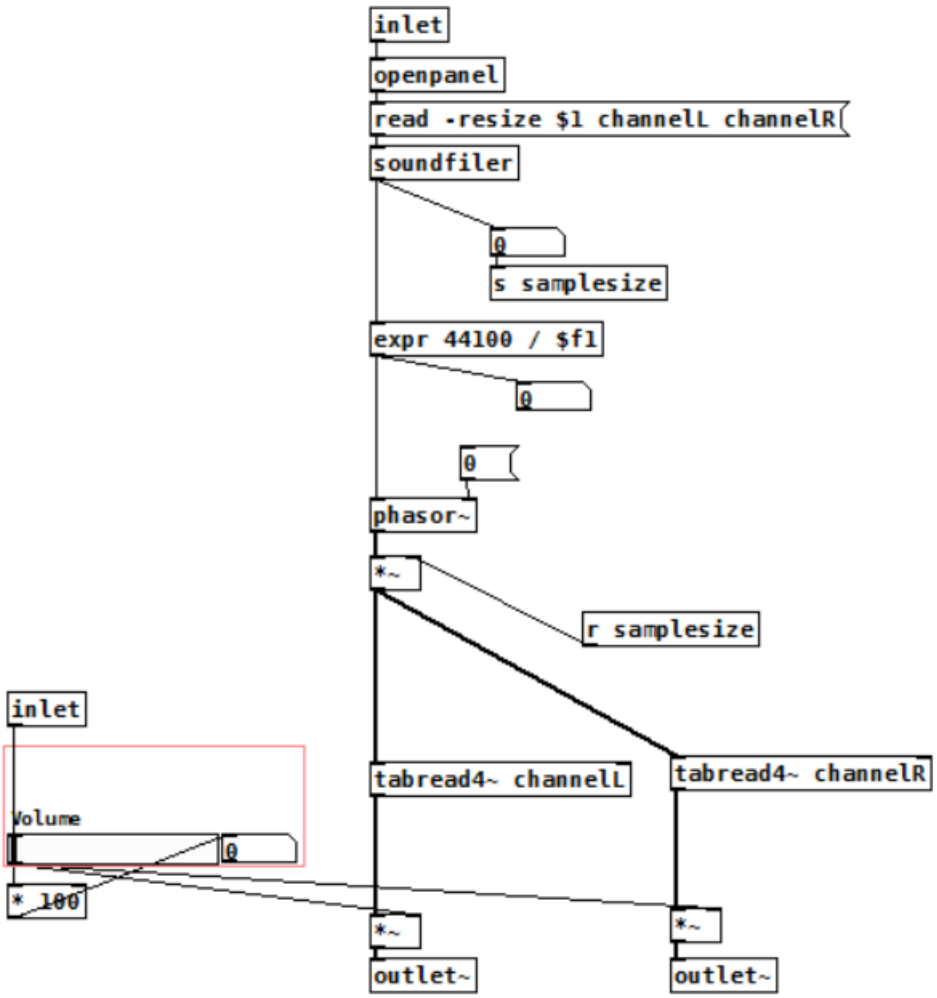
grapher~.pd



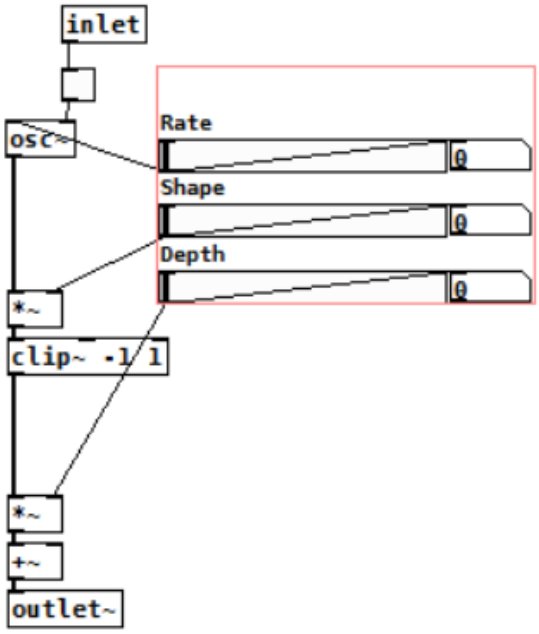
output~.pd



keycontrol.pd



recording~.pd



tremolo~.pd