Upravljanje televizorom pomoću nota

Aleksa Stojković,
EE7/2017

1. Uvod

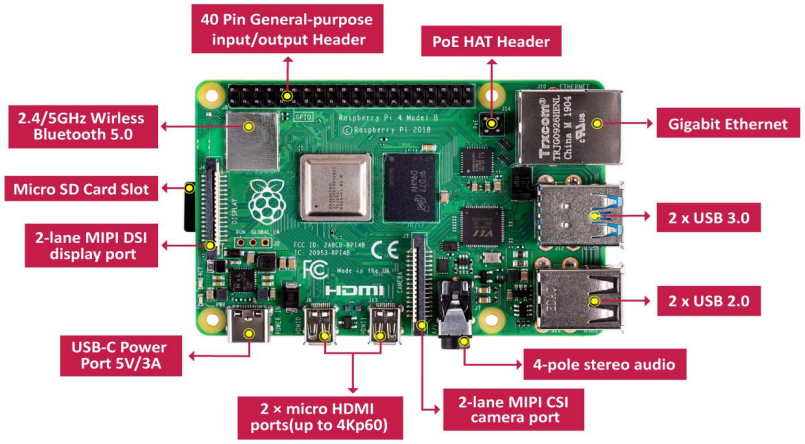
Tema ovog projekta je realizacija upravljanjem televizora korišćenjem muzičkih tonova. Za ostvarivanje ovakvog sistema kao glavna platforma korišćen je Raspberry Pi 4B sa 2 GB memorije. Projekat je urađen kao praktični deo predmeta *Računarska elektronika* na *Fakultetu tehničkih nauka*, smer za *Ugrađene sisteme i algoritme*.

Za realizaciju uređaja bilo je potrebno kombinovati znanja iz različitih oblasti, pri čemu se naročiti akcenat stavlja na razvoj softvera i hardvera, ali i poznavanja ljudskog ponašanja i psihologije (izuzetno važno za dobru implementaciju grafičkog korisničkog interfejsa). U nastavku poglavlja biće dat pregled ostalih sekcija ove dokumentacije.

U drugom poglavlju iznose se informacije vezane za Raspberry Pi, kao i za potreban softver koji je neophodno instalirati na uređaju kako bi se projekat uspešno implementirao. Treće poglavlje govori o procesu generisanja konfiguracionog fajla za izabrani televizor dok se u četvrtom poglavlju iznose informacije vezane za korišćeni algoritam za prepoznavanje tonova i njegovu implementaciju. Peta sekcija se odnosi na realizaciju grafičkog korisničkog interfejsa (u nastavku GUI). Na samom kraju dat je zaključak i navedena je korišćena literatura.

1. Korišćene komponente i potrebni softver

Kao što je već pomenuto u uvodnom poglavlju centralna komponenta ovog sistema je Raspberry Pi 4B od 2GB memorije. Sam Pi predstavlja SBC (*SIngle Board Computer*), jer su sve komponente potrebne za funkcionisanje integrisane na jednom PCB-u. Za izradu *embedded* sistema Pi predstavlja izuzetno moćnu opciju, pogotovo u poređenju sa drugom popularnom platformom, Arduinom. Konkretna verzija korišćena za realizaciju ovog projekta poseduje BCM2711 SoC sa četiri ARM Cortex-A72 jezgra čija maksimalna frekvencija iznosi 1.5 GHz, a širina magistrale je 64 bita. Raspberry ima i veliki broj mogućnosti da se poveže sa različitim perifierijama, kao što je, na primer, kamera, pa ga to čini izuzetno interesantnim za neke vizualne zadatke sa nešto skromnijom potrebom za proračunima. Međutim, čak i za nešto zahtevnije primene postoje dodaci koji poboljšavaju performanse ovog računara. Jedan takav dodatak je *Coral* USB acelerator koji omogućava primene dubokog učenja i *TensorFlow Lite*-a. Opisani dodatak je prikazan na slici 2. Pored ulaza za kameru postoje i USB i mirko USB ulazi, audio konektor, 40-pinski GPIO … Prikaz strukture i samih perifierija Raspberry-ja se može videti na slici 1.

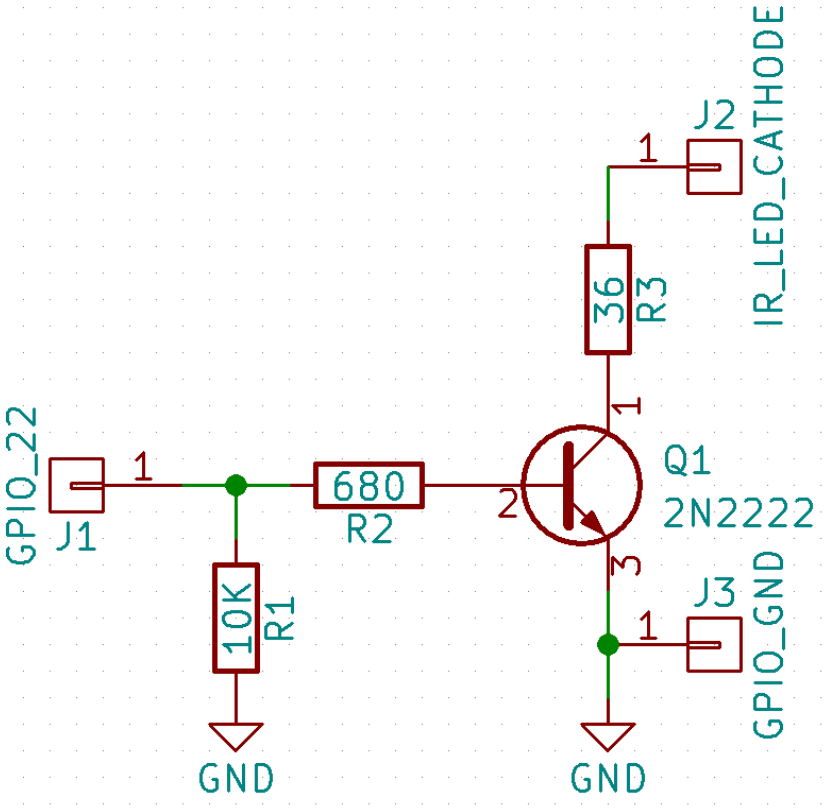


Slika 1. Prikaz strukture Raspberry Pi 4B modela i njegovih periferija

Slika 2. *Coral* USB akcelerator

Zarad prepoznavanja tonova nehodno je imati mogućnost zapisivanja zvuka. Kako bi se to ostavrilo korišćen je USB mini mikrofon koji se vrlo jednostavno koristi uz pomoć ALSA softvera. Na slici 3 se može videti izgled pomenutog mikrofona. Pored prethodno navedenih komponetni neophodni su i delovi vezani za IR signale, a to je TSOP34838 IR prijemnik, kao i odgovarajuće IR LED diode. Opciono, ukoliko se koristi neka jača IR dioda, sa većim strujnim potrebama, onda je nužno koristiti i jednostavan drajver za tu diodu koji sadrži proračunate otpornike i tranzistor. Vrednosti otpornika se proračunavaju na osnovu dokumentacije izabranog tranzistora, vrednosti struje koja želi da se postigne, kao i ograničnja koja unosi Pi-ov GPIO.

Slika 3. USB mini mikrofon



Slika 4. Primer drajvera za LED

Da bi se sam Raspberry osposobio neophodno je prvo instalirati operativni sistem (OS u nastavku). Za potrebe ovog projekta instaliran je Raspbian Buster operativni sistem u punoj verziji. Kako bi se to učinilo treba nabaviti i mikro SD karticu od bar 16 GB. Detaljnija uputstva za instalaciju OS-a se mogu pronaći u priloženoj literaturi. Da bi se osposobio rad sa USB mikrofonom instaliran je ALSA softver, što je urađeno sa narednom komandom:

sudo apt-get install alsa-utils

Nakon toga se sa komandama arecord i aplay mogu zapisivati i puštati .wav fajlovi. U nastavku je dat primer te dve komande gde se kod prve specificira vreme trajanja snimanja zvuka od dve sekunde koji će biti skladišten u fajl pod nazivom test.wav. Takođe, specificiran je i uređaj koji se koristi za snimanje zvuka.

arecord -D plughw:1,0 -d 2 test.wav
aplay test.wav

Da bi se snimio konfiguracioni fajl sa komandama za upravljanje televizorom i kako bi se one kasnije slale, korišćena je LIRC bibloteka. Za instalaciju je potrebno u terminalu pokrenuti naredne dve komande.

sudo apt-get install lirc
sudo apt-get install liblircclient-dev

Osim instalacije nephodno je u određenim fajlovima podesiti ulazne i izlazne pinove za prijem i slanje IR signala. Detaljnija uputsva data su u korišćenoj literaturi. Treba dodati da je za konkretan slučaj Raspberry Pi 4B modela bilo potrebno umesto dev/lirc0 koristiti dev/lirc1.

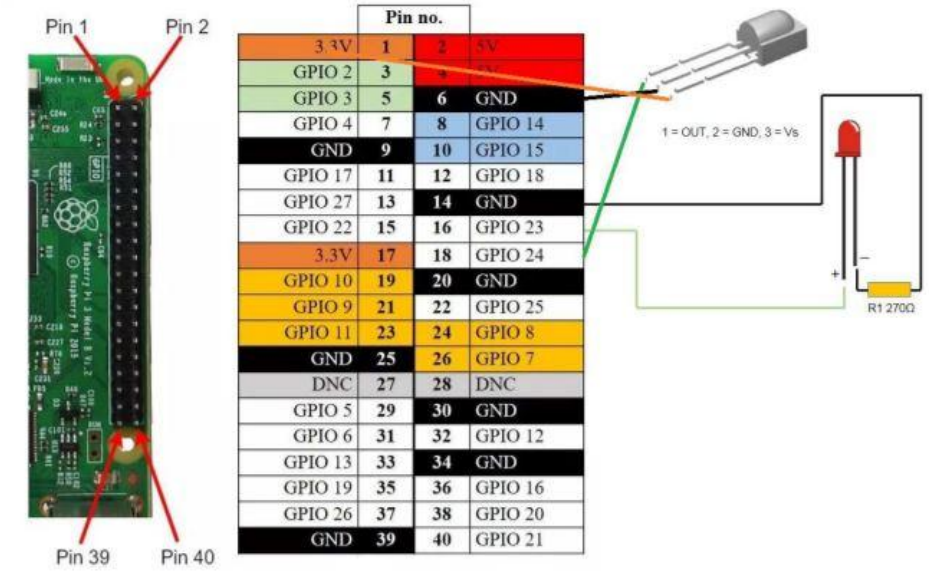
Za izradu GUI-a korišćen je QT koji pruža veliki broj mogućnosti za brz i efikasan razvoj interfejsa. Pored pomenutog softvera iskorišćena je i dodatna biblioteka *QCustomPlots* koja nije standardni deo QT-a. Nju je potrebno preuzeti sa interneta i dobijeni .h i .cpp fajl treba uključiti u QT projekat koji se koristi za razvoj GUI-a. Komande nužne za instaliranje QT-a su prikazane ispod.

sudo apt-get update
sudo apt-get install qt5-default
sudo apt-get install qtcreator

Da bi se otvarali .wav audio fajlovi bila je potrebna određena biblioteka koja pruža takve usluge. Iz tog razloga preuzeta je biblioteka *AudioFile*. Kako bi se ona mogla koristiti potrebno je uključiti biblioteku navođenjem putanje do .h fajla. Više informacija o pomenutoj biblioteci se može naći u sekciji *Literatura.*

2. Generisanje konfiguracionog fajla za TV

Nakon što su sve potrebne hardverske i softverske komponente pribavljene i instalirane, može se pristupiti generisanju konfiguracionog fajla za željeni televizor. Prvo je potrebno povezati IR prijemnik na napajanje, masu i ulazni GPIO. Na slici 5. se može videti kako je to potrebno uraditi. Takođe, na istoj slici vidi se i povezivanje IR LED-ovke u slučaju da nam nije potreban drajver za nju. U ovom projektu korišćen je taj pristup, jer nije bila dostupna snažnija dioda. Strujni limit korićene LED je 20 mA. Ovo može predstavljati problem, ako je neophodno da se uređaj nalazi dalje od samo TV-a, tada je nužno nabaviti jaču diodu. U priloženoj literaturi korišćena je dioda od 100 mA i drajver koji je prikazan na ranijoj slici.



Slika 5. Povezivanje LED i IR prijemnika na GIPO

Da bi se započeo proces snimanja konfiguracionog fajla neophodno je prvo zaustaviti LIRC, pa zatim sa irrecord pokrenuti snimanje. Nakon toga biće dato uputsvo šta treba raditi kako bi se dekodovao protokol koji koristi odabrani TV. Pre samog pokretanja irrecord može se proveriti ispravnost rada TSOP34838 tako što se pokrene komanda mode2. Nakon što je završeno snimanje fajla, potrebno je dati fajl prebaciti u /etc/lirc/lircd.conf.d. Pored toga, treba dodati u /etc/lirc/lircd.conf liniju *include "lircd.conf.d/Naziv_Uređaja.lircd.conf".* Posle svih navedenih koraka može se proslediti komada sa irsend. Primeri komandi dati su u nastavku.

sudo /etc/init.d/lircd stop
irrecord -nd /dev/lirc1 ~/lircd.conf
irsend SEND_ONCE VoxTV KEY_POWER

3. Prepoznavanje tonova

Problem prepoznavanja tonova spada u oblast digitalne obrade signala. Za dati problem postoji veći broj tehnika koje ga rešavaju, manje ili više uspešno. Danas su pogotovo popularne metode zasnovane na mašinskom učenju. U ovom projektu korišćen je algoritam koji spada u klasu koja pokušava da pristupi rešavanju problema u vremenskom domenu, nasuprot onim koji to rade u frekvencijskom domenu. Glavna operacija koja se koristi je autokorelacija koja je data jednačinom (1).

$$r(\tau) = \sum_{n=0}^{N-1} x(n) \cdot x(n - \tau) \qquad (1)$$

Autokorelacija može da se tumači kao mera sličnosti dva signala za dati pomeraj. Ako se pažljivo pogleda jednačina (1), može se zaključiti da u slučaju kada je x periodični signal, tada ako je τ jednako periodi ili celobrojnom umnošku periode r će da daje velike vrednosti. Suprotno, očekuju se male vrednosti. Budući da su tonovi superponirani sinusni signali, što znači da su periodični, onda to implicira da treba naći τgde imamo skokove u vrednostima autokorelacije. Nakon toga, treba locirati prvu pozitivnu promenu autokorelacionog signala. Kada se ona nađe traži se indeks na kom dolazi do maksimuma autokorelacionog signala, ali za vrednosti indeksa većeg od kada je registrovana prva pozitivna vrednost. Kada je nađen taj indeks on predstavlja procenu periode onovnog harmonika, ali zarad veće preciznosti koristi se kvadratna interpolacija sa vrenostima u okolini maksimuma koji je nađen. Kako bi se postigla veća pouzdanost algoritma, originalni audio snimak se obrađuje u prozorima, pri čemu se veličina prozora i razmak između dva susedna prozora može podešavati. U slučaju ovog sistema veličina prozora i korak iznose 1024. U tom slučaju dobija se procena note za svaki prozor. Nakon što su svi prozori odrađeni, kao krajnji rezultat vraća se nota koja je najviše puta bila data kao procena.

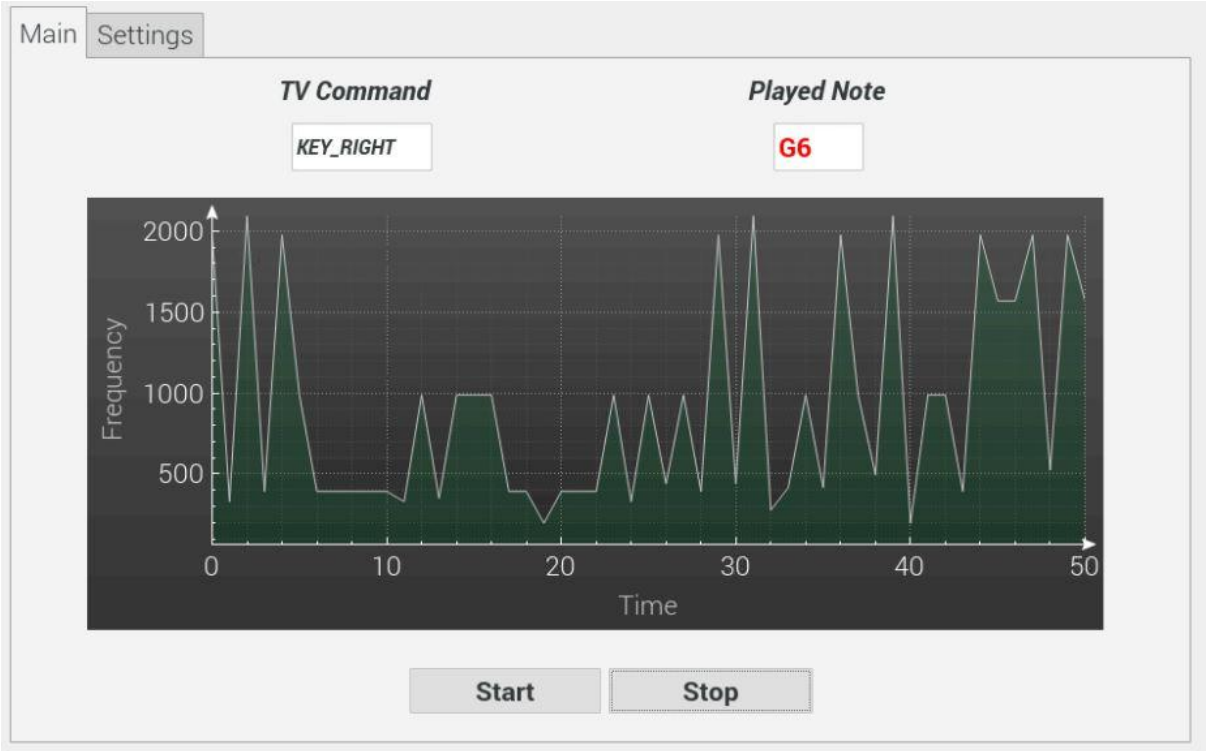
Prvobitno je algoritam bio implementiran u *python*-u dok nisu bili podešeni svi parametri i dok sama struktura algoritma nije dobila konačan oblik. Nakon toga je urađena implementacija u *C++* kako bi kasnija integracija sa QT projektom bila jednostavnija. Napravljena je klasa Pitch_Detector i njeni .hpp i .cpp fajlovi su uključeni u QT projekat. Kako su performanse bile nezadovoljavajuće da bi se obezbedio rad u realnom vremenu korišćeno je višenitno izvršavanje gde se po četiri prozora obrađuju u četiri zasebne niti. Druga mogućnost za optimizaciju je korišćenje efikasnijeg algoritma za implementaciju autokorelacije koji se bazira na računanju konvolucije pomoću FFT. U slučaju da se opredeli za ovaj put savetuje se korišćenje neke od biblioteka koje implementiraju FFT zbog izuzetno efikasnih realizacija. Jedna od takvi biblioteka je FFTW o kojoj se može više saznati u priloženim materijalima pod sekcijom *Literatura.*

Kako bi se izvršilo zapisivanje audio fajla i kako bi se pokretale pojedine IR komande korišćena je funkcija *system()* koja kao parametar prima string. Prosleđeni string predstavlja komandu koja će se izvršiti u terminalu. Za samo čitanje .wav fajla kotistile su se funkcije *AudioFile* biblioteke.

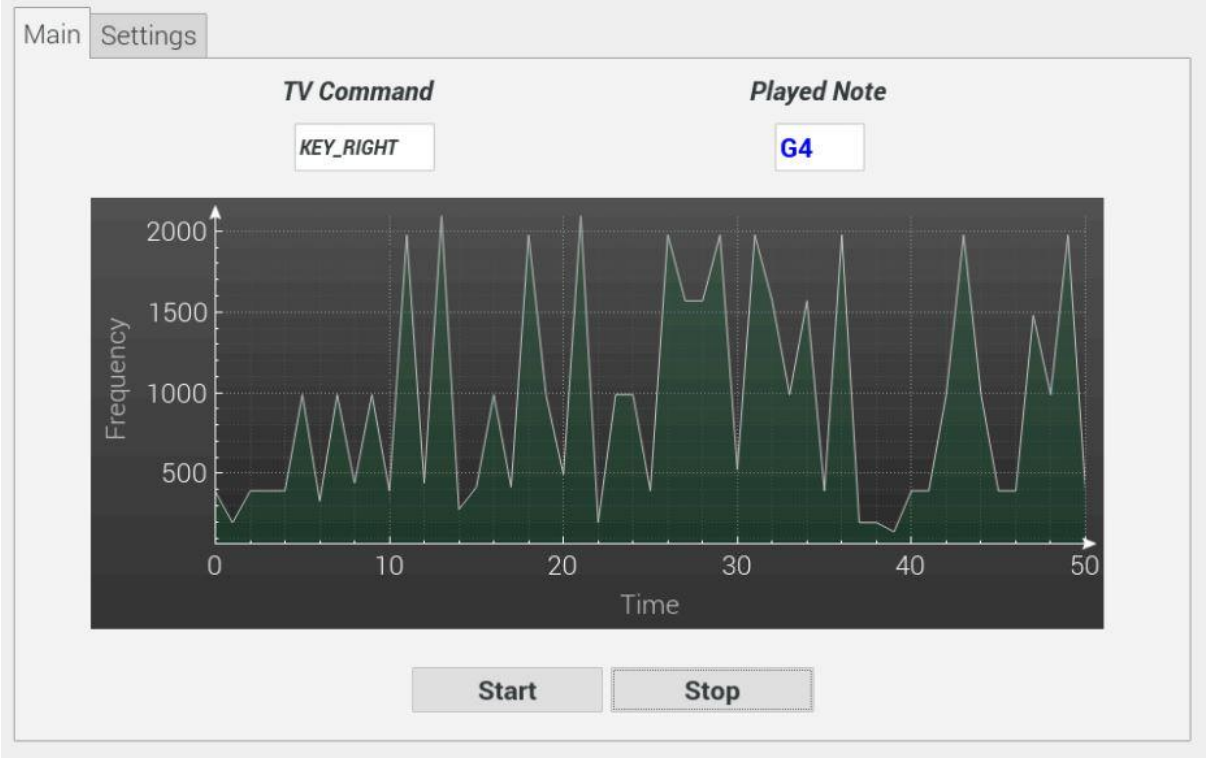
4. Izrada grafičkog korisničkog interfejsa

GUI je napravljen uzimajući u obzir jednostavnost i intuitivan način korišćenja. Interfejs se sastoji iz dva taba koja se nazivaju *main* i *settings.* U *main* tabu postoje dva dugmeta *start* i *stop*. Njihova uloga je da zaustavljaju i pokreću proces snimanja audio signala i njegove obrade. U slučaju kada se stopira rad, stanje koje je poslednje bilo u glavnom prozoru ostaje *zamrznuto.* Pored pomenutih elemenata glavni prozor sadrži isipis trenutne note, kao i poslednje komande prosleđene televizoru. U slučaju ispisa trenutne note, ako je procenjen ton među skupom validnih tonova koji su povezani sa određenom komandom za TV, onda je boja ipisa plava, a u suprotnom je crvena. Najveći de površine *main* taba zauzima grafik koji prikazuje frekvenciju poslednjih 50 otsviranih nota. Za njegovu izradu korišćena je *QCustomPlots* biblioteka. Slike 6 i 7 prikazuju izgled glavnog taba.

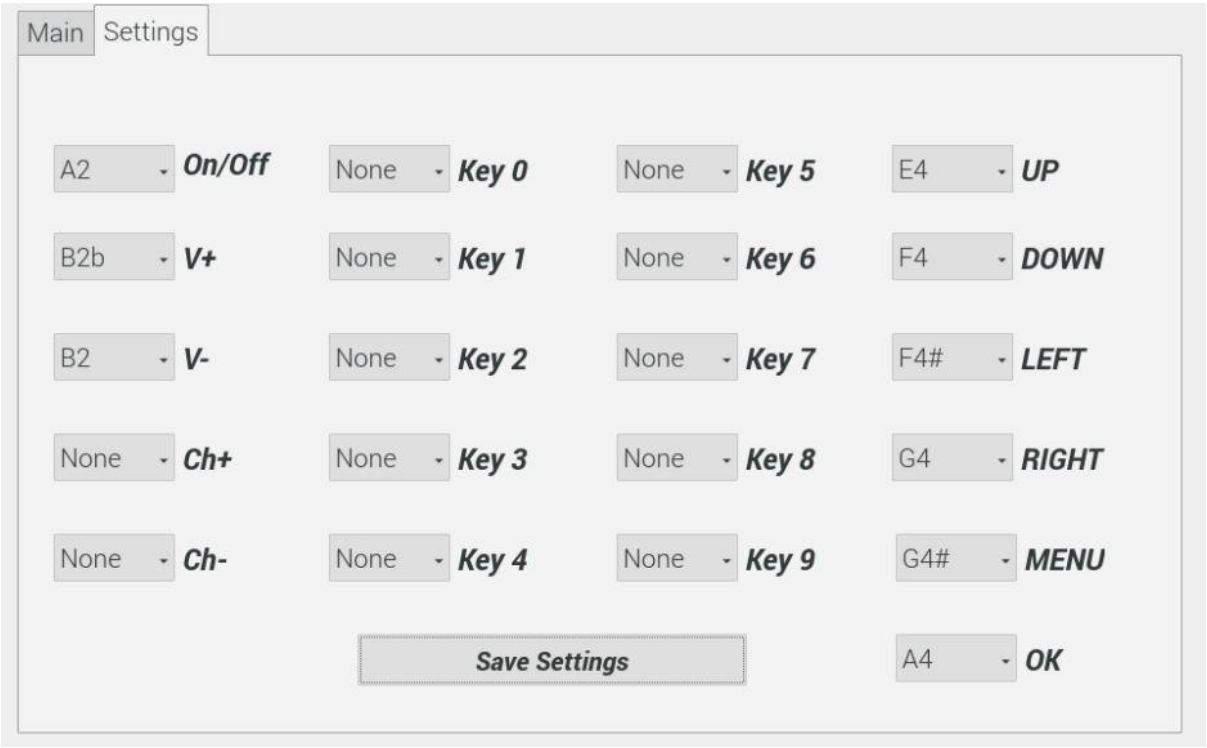
Drugi tab služi za podešavanja. Za svaki od dugmića postoji opcija za izbor tona koji će predstavljati naznačenu akciju. Za prikaz tih opcija korišćen je *combo box.* U listi izbora nalaze se tonovi između C2 i C6. Na slikama 8 i 9 može se videti izgled taba koji služi za podešavanja. Budući da ima dosta opcija za podešavanje, GUI je realizovan tako da se pritiskom na dugme *save* sačuva trenutna konfiguracija koja se upisuje u jedan .txt fajl. Pomenuti fajl se učitava u konstruktoru. Takođe, moguće je da se neke opcije ne koriste, pa se onda može izabrati za te komande *None.*



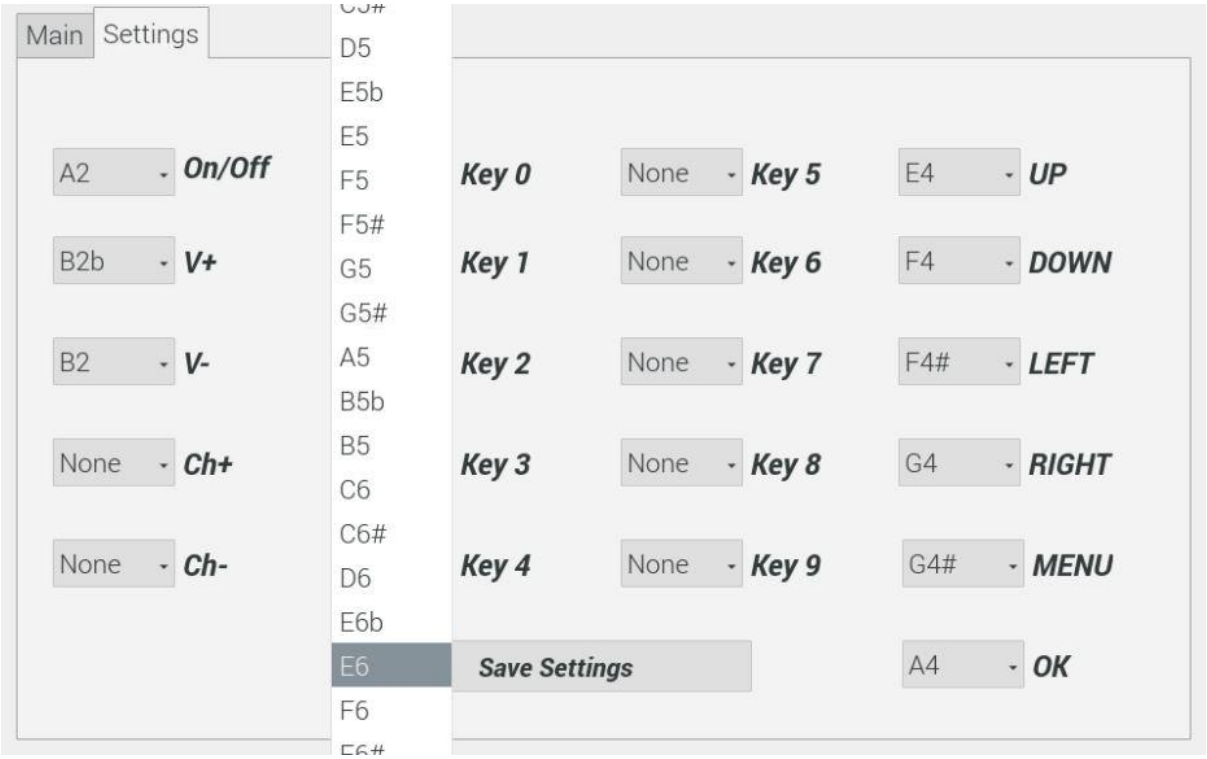
Slika 6. Prikaz main prozora GUI-a u slučaju kad je registorvana nota koja nije validna



Slika 7. Prikaz main prozora GUI-a u slučaju kad je registorvana validna nota



Slika 8. Izgled settings prozora



Slika 9. Ilustracija prikaza opcija u settings prozoru

5. Zaključak

Za samu izradu projekta korišćena je metoda *podeli, pa vladaj*. Identifikovani su graditivni blokovi celog sistema i onda se u izolaciji radilo na njima. Kada je sve bilo verifikovano započeta je integracija u QT projekat. Nakon što je napravljena prvobitna verzija izvršeno je par testova i otklonjene su greške. Konačna verzija uspešno radi u praksi gde se tonovi odsvirani na gitari sa velikim uspehom prepoznaju. Treba naglasiti da algoritam uspešno prepoznaje i tonove složenijih zvukova kao što je ljudski glas, tako da sam sistem nije ograničen na određene instrumente. Treba istaći da se frekvencije zaokružuju na najbližu notu, tako da nije moguće detektovati koliko je neki ton precizno odsviran ili otpevan.

6. Literatura

1. https://www.instructables.com/Raspberry-Pi-Zero-Universal-Remote/?fbclid=IwAR0lch3v3Jag09Blw-KEMzD0iDn9Wu_D85nskdlMA03On89prHTkCQ-os2A, Septembar 2021.
2. https://makersportal.com/blog/2018/9/13/audio-processing-in-python-part-i-sampling-and-the-fast-fourier-transform?fbclid=IwAR3AWkznECkwDkJrLyVQAaiLcZxsxmz6q4aVLa9kkX6IGwlySJFrBDMPQrY, Septembar 2021.
3. https://www.instructables.com/Raspberry-Pi-Guitar-Tuner/?fbclid=IwAR2vdtCa7JC0uwpEPZe4gzZGk551nt84M2EkJnWcCCo7qQT3E0LwEx_1vfM, Septembar 2021.
4. https://github.com/Virtualan/musical-note-trainer?fbclid=IwAR3pUL3unK_vLv_sAo_m9s1o4lLg2wVNtqgJCsfQLnzxs3WKpAACRx6WwiM, Septembar 2021.
5. https://github.com/adamstark/AudioFile, Septembar 2021.
6. https://www.fftw.org/, Septembar 2021.
7. https://www.qcustomplot.com/, Septembar 2021.
8. https://www.elektronika.ftn.uns.ac.rs/racunarska-elektronika/specifikacija/specifikacija-predmeta/, Septembar 2021.
9. Alain de Cheveigne, Hideki Kawahara *,YIN, a fundamental frequency estimator for speech and music,* 2002.
10. https://www.makeuseof.com/tag/install-operating-system-raspberry-pi/, Avgust 2021.
11. https://www.qt.io/, Setptembar 2021.
12. https://www.lirc.org/, Avgust 2021.