UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA AUTOMATIKU I ELEKTRONIKU

Prepoznavanje pjesme iz kratkog audio uzorka

SEMINARSKI RAD

PREDMET: RAZVOJ SOFTVERA ZA UGRADBENE SISTEME

Autori:
Isam Vrce (1997/18447)
Armin Žunić (1936/18301)

Odgovorni profesor:
Vanr.prof.dr Emir Sokić

Sarajevo,
septembar 2022.

Sadržaj

Uvod

Obrada zvuka je disciplina kojoj su se ljudi još od davnina zanimali, ali koja je i dan danas atraktivno polje. Atraktivno je iz više razloga, među kojima se najviše izdvaja sami razvoj tehnologije koji je uslov za dalji napredak i obrade zvuka kao zasebne cjeline.

Računar opšte namjene omogućava razne manipulacije nad zvučnim signalima, međutim i dosta pojednostavljena verzija računara, u obliku nekog mikrokontrolera, umogućava gotovo istu ili sličnu primjenu. U ovom radu se upravo željelo pokazati da je na mikrokontroleru ograničenih mogućnosti moguće vršiti obradu zvučnog zapisa i to u realnom vremenu.

Ono čemu se posvetila posebna pažnja jeste upravo "rad u realnom vremenu". Kroz nekoliko narednih sekcija se u detalje opisao postupak obrade zvučnog signala, mane i prednosti određenih metoda, brzine izvršavanja i slično, sve u cilju postizanja konačnog cilja, a to je prepozvanje isječka pjesme.

Rad je podijeljen u 5 cjelina, prvi uvodni dio, zatim teorijski osvrt na značaj MFCC koeficijenata u obradi zvučnih signala. U ovom dijelu je detaljno pojašnjen način dobijanja MFCC koeficijenta, te ono za šta su ovi koeficijenti korisni. Nakon teorijske analize, slijedi praktična implementacija alogritama potrebnih da bi se MFCC koeficijenti mogli računati u realnom vrmenu. Ovdje je pokazana i komparativna anliza između brzine izračunavanja DTF u odnosu na FFT. Nakon toga, opisana je jedna primjena MFCC koeficijenata, a to je prepoznavanje pjesme iz skupa pjesama. Te konačno slijedi zaključak u kojem je izložen pregled svih cjelina obrađenih u radu, te mogućnosti za poboljšanja u budućnosti.

MFCCs

Kao jedan od osnovnih problema sa kojim su se autori susreli, već na početku izrade ovog rada, jeste način na koji će se "nešto" ekstraktovati iz zvuka. Za pretpostaviti je da ovo "nešto" čine određeni brojevi koji će na neki način sadržavati informacije o zvučnom signalu. Sada se postavlja pitanje na koji način to izvesti. Iz nekoliko potencijalnih identifikatora, jedni se posebno izdvajaju, a to su Mel-frequency cepstral coefficients (MFCCs). Mel-frequency cepstral koeficijenti su se pokazali kao veoma moćno oružje u izdvajanju karakteristika od značaja u audio zapisu. Isto tako, pokazao je veoma dobre rezultate i pri samom prepoznavanju zvuka/govora u nekom audio zapisu.

Izdvajanje karakteristika na pomenuti način zahtjeva na prvu ruku kompleksnu matematiku, ali iza koje stoji jednostavna logika. Izdvajanje koeficijenata se može podijeliti u nekoliko faza:

- podjela ulaznog signala na dijelove, pri čemu će svaki dio činiti određeni broj uzoraka (frejmova) signala
- za svaki dio (isječak signala) je potrebno izračunati Furierovu transformaciju, što će predstavljati svojevrsni spektar snage svakog isječka signala
- na takve spektre se primijeni Mel filtriranje
- filtrirani uzorci uz određenu matematiku postaju konačni koeficijenti

2.1 Izdvajanje dijelova signala

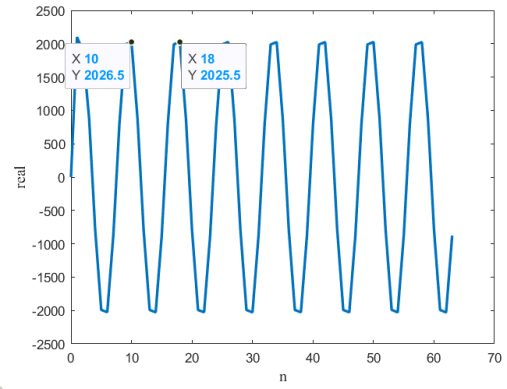
Znamo da je audio signal kontinualni signal, ali isto tako možemo sa velikom sigurnošću reći da u relativno malom koraku takav signal ima konstantnu vrijednost. Ukoliko je taj korak dovoljno mali, a opet ne previše mali zbog mogućeg nepotrebnog utroška resursa, obrada audio signala sa ovim korakom može početi.

Obično se okvir za izdvajanje signala postavljanja na 10-40ms, pri čemu se unutar ovog vremenskog perioda treba nalaziti određeni broj uzoraka. Taj broj direktno zavisi od frekvencije uzorkovanja signala. Tako u periodu od 16ms, sa frekvencijom 8kHz možemo prikupiti 128 uzoraka signala. Druga stvar je koliko će sistemu (mikrokontroleru) biti potrebno vremena za uzimanje 128 uzoraka i njihovu potencijalno kompleksnu obradu, koja će u ovom radu biti neminovna, uz napomenu da se to sve treba završiti za 16ms, jer nakon toga dolaze novi uzorci na obradu.

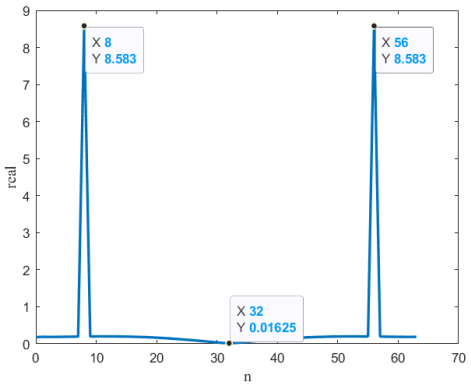
2.2 Spektar snage

U narednom koraku je potrebno naći spektar snage svakog okvira, tj. potrebno je pronaći Furierovu transformaciju niza uzoraka koji se nalaze u jednom okviru, i tako za svaki naredni okvir posebno. Implementirani su i DFT i FFT, te su se dobili veoma zanimljivi rezultati o kojima će biti govora u narednim sekcijama. Ovdje vrijedi pokazati da se Fourierova transformacija uspješno

realizirala, što potvrđuju slike (slika 2.1 i slika 2.2).



Slika 2.1: Ulazni sinusni signal frekvencije 1kHz



Slika 2.2: Izlazni spektar sinusnog signala frekvencije 1kHz

Na lijevoj slici je prikazan ulazni signal frekvencije 1kHz, uzorkovan frekvencijom 8kHz. Upravo zbog toga imamo 8 uzoraka u jednom periodu sinusa. Na desnoj slici je prikazan spektar i ako znamo da bi sinus frekvencije 4kHz (maksimalna frekvencija signala kojeg možemo uzorkovati frekvencijom 8kHz) imao peak na 32, onda je sasvim logično zašto sinus frekvencije 1kHz ima peak u 8 i 56 po x osi. Ovim se željelo pokazati da je sistem u mogućnosti prikupiti 64 uzorka za 8ms, te nad istim uraditi Furierovu transformaciju, bez da izađe iz okvira od 8ms. Više o samoj implementaciji ove transformacije, kao i o problemi sa kojima su se autori susreli u toku rada, će biti govora u narednim sekcijama.

2.3 Mel filtriranje

Potencijalni problem neraspoznavanja frekvencija bliskih jedna drugoj (pogotovu izraženih kod audio zapisa nastalih muzičkim instrumentima) se rješava uvođenjem Mel filtera. Svaki od Mel filtera će izdvojiti određene frekvencije, dok će ostale zanemariti. Nad izdvojenim spektrom se računa svojevrsno množenje takvih frekvencija sa filterom, te se dobijaju veoma bitnu podaci za konačne koeficijente.

Međutim, prije svega vrlo važno za odrediti jeste opseg frekvencija koje se žele i mogu obrađivati na sistemu. U našem slučaju je to opseg [100Hz, 3900Hz], i ovo su faktički ulazni podaci za formiranje trokutastih Mel filtera. Za formiranje Mel skale potrebno je prvenstveno odrediti minimalnu i maksimalnu Mel frekvenciju uz pomoć formule:

$$M(f) = 1125 \cdot ln(1 + \frac{f}{700}) \tag{2.1}$$

Nakon ovoga je moguće formirati linearnu Mel skalu od minimalne do maksimalne Mel frekvencije uz korak koji zavisi od broja filtera koji se koristi. Za primjenu muzičkih instrumenata je poželjno imati što veći broj filtera, kako bi se mogao izdvojiti svaki ton zasebno. Primjera radi, ton B2 daje frekvenciju 123.47Hz na klaviru, dok njegov susjedni ton C3 daje frekvenciju 130.81Hz. Očigledno je u ovom slučaju potrebno imati filtere sa početnim korakom od 7Hz, što je prilično nezgodan zahtjev. Naravno, za veće frekvencije koraci su dosta veći, što nas upravo navodi na činjenicu da će naša frekventna skala filtera biti nelinearna.

Iako neke literature koje se bave ovom tematikom kroz prizmu muzičkih instrumenata koriste 200 Mel filtera, autori su se ipak odlučili na nešto rasterećeniju i standardizovanu verziju od 48

Mel filtera. Nakon što se formirala linearna Mel skala, moguće je inverznom formulom dobiti nelinearnu frekventnu skalu na već pomenutom opsegu. Inverzna formula ima oblik:

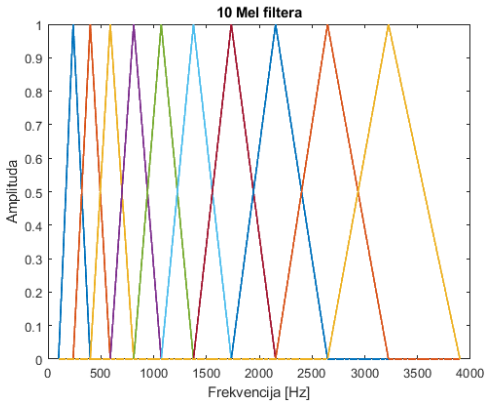
$$f = M^{-1}(m) = 700 \cdot (e^{\frac{m}{1125}} - 1) \tag{2.2}$$

Formiranje konačnih filtera se izvodi prema sljedećoj relaciji [1]:

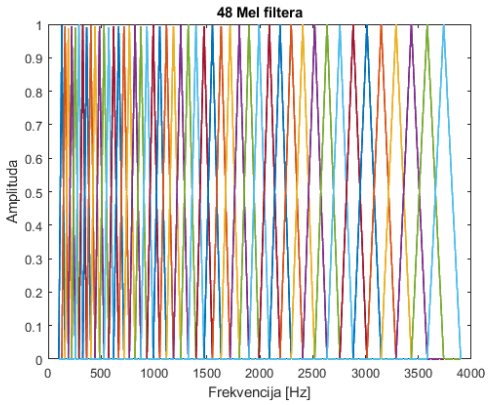
$$H_m(k) = \begin{cases} 0, & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) \leq k \leq f(m+1) \\ 0, & k > f(m+1) \end{cases} \tag{2.3}$$

Prvi filter počinje u prvoj tački, dostiže maksimum u drugoj tački, te se vraća na nulu u trećoj tački. Drugi filter počinje u drugoj tački, dostiže maksimum u trećoj, dok se vraća na nulu u četvrtoj itd. Na slici (slika 2.3) je prikazano 10 Mel filtera, samo kako bi se dobio vizuelni efekat nelinearnosti. Jasno je da su filteri pri manjim frekvencijama uži, obuhvataju manji opseg, dok su na većim frekvecnijama i opsezi mnogo veći.

Na slici (slika 2.4) je prikazano 48 filtera. Može se primijetiti velika gustina filtera pri manjim frekvencijama, gdje je korak $\approx 30Hz$. Korak na najvišim frekvencijama iznosi $\approx 170Hz$.



Slika 2.3: 10 Mel filtera [100Hz - 3900Hz]



Slika 2.4: 48 Mel filtera [100Hz - 3900Hz]

2.4 Formiranje konačnih koefcijenata

Na kraju svega su nađeni fft propusti kroz ovakve filtere. Kroz svaki od filtera se propusti cijeli FFT, množeći tačke filtera sa uzorcima FFT-a. Svi takvi produkti se sabiraju i po završetku posljednjeg uzorka FFT-a se formira konačna suma jednog takvog filtera. Još jedan korak je ostao do formiranja konačnog rezultata jednog filtera, a to je množenje prethodne sume sa diskretnom kosinusnom transformacijom indeksa određenog Mel koeficijenta. Takva transformacija je data formulom [1]:

$$T_{DCT}(m) = cos(\frac{\pi \cdot m}{N} \cdot (l - 0.5)), \tag{2.4}$$

gdje je m Mel koeficijent, N broj Mel filtera, dok je l trenutni redni broj filtera koji se obrađuje. Moguće je računati mnogo Mel koeficijenata, dok je pokazano da je optimalan broj njih 13, dok su autori u radu koristili 5 takvih koeficijenata.

Implementacija algoritama na mikrokontroleru

Implementacija algoritma na digitalnom računaru može djelovati trivijalna nakon što su poznate formule i postupci koje je potrebno provesti. Nakon što smo shvatili teorijsku pozadinu i postupak dobijanja MFCC koeficijenata krenuli smo u implemetaciju. Obzirom da je za računanje MFCC koeficijenata neophodno poznavanje spektra ulaznog signala u vremenskom domenu, jasno je da je potrebno prije svega izračunati Fourierovu transformaciju ulaznog signala.

Međutim, prije svega je potrebno razumjeti da se implementacija **ne** vrši na računaru opšte namjene, nego na mikrokontroleru. Računari opšte namjene posjeduju procesore koji su neuporedivno moćniji od onih koji koriste mikrokontroleri, te je ovaj podatak potrebno imati na umu tokom cijelog postupka implementacije. Dodatno autori su željeli implementirati algoritam koji će biti u mogućnosti vršiti obradu signala u realnom vremenu. Obrada signala u realnom vremenu je predstavljala osnovno ograničenje u ostatku ovog rada. Upravo zbog ovog ograničenja su izvršeni neki "ustupci".

Da li bilo jasnije na kojem hardveru se vrši implementacija, ali i da se ukaže na to da se radi o zaista moćnom hardveru (jer u protivnom bi bilo nesuvislo pokušavati osigurati rad u realnom vremenu) slijedi kratak pregled karakteristika mikrokontrolera koji se koristi:

Mikrokontroler koji se koristi je: FM4 S6E2CC [2]

- 200 MHz ARM Cortex MCU (bitan podatak da je brzima otkucaja kristala što veća)
- 675 CoreMark® MCU Single precision IEEE 754 compliant floating point unit
- 2 MB Flash memorije, 256 KB SRAM memorije, te 190 GPIO pinova (broj GPIO pinova nije pretjerano bitan, jer ne koristimo niti jedan GPIO pin. Međutim, podatak o količini dostupne memorije za pohranu programskog koda na sistem je jako bitan. Obzirom da se ovdje uračunavaju i "header" fajlovi u kojima se nalaze razne "look-up" tabele, koje se koriste za ubrzanje rada sistem (o kojima će biti govora u nastavku), te podaci koje će program koristiti za poređenje i određivanje korelacija, ova memorija se "brzo troši". Pa je o tome potrebno voditi računa.
- komunikacija: 10/100 Ethernet (IEEE 802.3), USB host, USB device, CAN, LIN, high-speed quad SPI, I²S, I²C, and UART (od svih komunikacijskih protokola, nama je bitan "I²S", jer se ovaj protokol koristi prilikom akvizicije signala sa mikrofona, odnosno line-in ulaza sa računara.)

Okruženje u kojem je vršena implementacija je: "µVision® IDE", uz napomenu da koristimo licenciranu verziju. Ograničenje na veličinu koda u slučaju korištenja edukacijske verzije je 32 KB, koja se u slučaju korištenja velikog broja "look-up" tabela brzo potroši.

3.1 Akvizicija ulaznog signala

Ulazni signal je kontinualni signal koji je potrebno uzorkovati. Korišteni mikrokontroler posjeduje ugrađenu funkciju koja omogućava da se prekidna rutina "I2S HANDLER" poziva tačno određenom frekvencijom. Odabir frekvencije kojom će se vršiti poziv prekidne rutine mora biti iz skupa (8000 Hz, 32000 Hz, 48000 Hz, 96000 Hz). Odabrali smo da se uzorkovanje vrši frekvencijom od 8000 Hz. Ovo znači da između svaka dva uzorka imamo 0.000125 s da se obradi sve ono što se nalazi unutar prekidne rutine, u protivnom ponašanje sistema nije predvidljivo. Ovo za posljedicu ima da je kod koji se izvršava u prekidnoj rutini potrebno što više skratiti. Odnosno nepoželjno je raditi bilo kakve složene operacije koje oduzimaju vrijeme. Obzirom da je nama potrebno da se rade operacije koje su računarski zahtjevne u ovoj prekidnoj rutini samo postavljamo "zastavice", odnosno pohranjujemo diskretizirane ulazne signale koji će se izvršavati dok procesor ništa ne radi.

Ovo znači da možemo osigurati da prekidna rutina prikuplja 128 ulaznih signala, a da za vrijeme dok se u prekidnoj rutini prikupljaju ovi signali mi obrađujemo prethodne. Za slučaj kada se prikuplja 128 ulaznih signala, imamo 128 * 1 / 8000 Hz = 16 ms da obradimo prethodne ulazne signale i generiramo izlaz da bismo mogli govoriti o tome da procesor radi u realnom vremenu. Za slučaj da prikupljamo 64 ulazna signala imamo 8 ms da ih obradimo. Analogno za bilo koji broj ulaznih signala. U prethodnoj sekciji rada je bilo govora o tome koliki vremenski interval treba biti (10ms do 40ms) a mi smo odabrali 16ms i 128 uzoraka.

3.2 Računanje diskretne Furierove transformacije

Za računanje MFCC koeficijenata je potrebno poznavanje Furierove transformacije (FT) u nekom prethodnom intervalu vremena. Pošto se implementacija vrši na digitalnom računaru, gdje imamo diskretne vrijednosti nije moguće računati kontinualnu FT. Moguće je samo numerički izračunati Fourierovu transformaciju. Diskretizacijom Fourierove transformacije dobija se diskretna Fourierova transformacija – DFT (engl.Discrete Fourier Transform) [3].

Jedna od mogućih implementacija DFT-a jeste direktna implementacija, odnosno implementacija po formuli [4]:

$$X_n(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \tag{3.1}$$

pri čemu su tzv twiddle faktori jednaki:

$$W_N = e^{-j\frac{2\pi}{N}} \tag{3.2}$$

Iz jednačina 3.1 i 3.2 se može zaključiti da je za implementaciju potrebno koristi dvije ugnježdene petlje, odnosno da je kompleksnost ovog algoritma O(n^2) [5]. Iako kompleksnost jeste kvadratna, nemamo nikakvu predstavu o vremenu izvršenja koda. Zbog toga je u nastavku dat kratki pregled vremena izvršenja programa.

Listing 3.1: DFT

```
1
2  void dft(COMPLEX *ulaz)
3  {
4    COMPLEX rezultat[N];
5    int k,n;
6    for (k = 0; k < N; k++) {
7      rezultat[k].real = 0.0;
8      rezultat[k].imag = 0.0;
9
10     for (n = 0; n < N; n++) {
11       rezultat[k].real += ulaz[n].real*cos(2*PI*k*n/N)
12                         + ulaz[n].imag*sin(2*PI*k*n/N);
13       rezultat[k].imag += ulaz[n].imag*cos(2*PI*k*n/N)
14                         - ulaz[n].real*sin(2*PI*k*n/N);
15     }
16   }
17   for (k = 0; k < N; k++) {
18     x[k] = rezultat[k];
19   }
20 }
```

Obzirom da mi efektivno računamo short-time DFT, ova obrada se mora završiti mnogo prije nego li pristigne naredni skup podataka, jer je potrebno vršiti razne druge proračune. Kompleksnost DFT algoritma je kvadratna, pa je za očekivati da će obrada trajati 4 puta kraće za duplo manje uzoraka. Zbog toga smo sa analizom krenuli od 16 uzoraka. Da bi program radio u realnom vremenu potrebno je da izračunavanje DFT-a završi unutar 2 ms. Međutim, programu je potrebno 12 ms. Što je jasan pokazatelj da se DFT u izvornoj formi ne može koristiti.

Međutim, prilikom računanja DFT-a se unutar dvostruke petlje se računaju twiddle faktori. Ali twiddle faktori se mogu izračunati unaprijed i isti se mogu spremiti u obliku look-up tabele. Alternativno mogu se koristi ugrađene "brze" funkcije za računanje sinusa i kosinusa unutar bibiloteke CMSIS-DSP, koje koriste look-up tabele i interpolaciju za računanje[6]. Kada se iskoriste ove promjene, specifično korištenje ugrađenih funkcija "arm_cos_f32" i "arm_sin_f32" dobija se veliko poboljšanje u odnosu na prethodnu implementaciju, jer se DFT za 32 uzorka izračuna za 2 ms (na raspolaganju imamo 4ms).

Malo prije ovoga je naglašeno da je složenost algoritma kvadratna, pa ukoliko se pokuša odrediti DFT za 64 uzorka, potrebno je 8ms, a na raspolaganju imamo 64 * 1 / 8000 = 8 ms. Odnosno potrošili bismo sve vrijeme samo na računanje DFT-a. Ne bismo imali vremena računati MFCC niti bilo koje druge proračune, a da pri tome očekujemo rad sistema u realnom vrmenu. Još jedna napomena, koja stoji u vezi sa brojem uzoraka za koji je potrebno računati MFCC, a samim tim i DFT. Preporučeno je da se obrada podataka vrši za posljednjih 10 do 40 ms. Samim tim ukoliko želimo da obrađujemo broj uzoraka koji je jednak potenciji broja 2, onda prvi takav broj koji zadovoljava prethodno ograničenje je 128 (16 ms).

Iz prethodnog se može zaključiti da je DFT previše spor algoritam. Zbog toga je potrebno pronaći algoritam koji je brži, jedan takav algoritam je "Fast Fourier transform" (FFT). Ovaj algoritam ima mnogo implementacija, a dva su osnovna: brza Fourierova transformacija po bazi 2 sa decimiranjem po vremenu, te brza Fourierova transformacija po bazi 2 sa decimiranjem po učestanosti. Iako u biblioteci CMSIS-DSP postoje ugrađene funkcije koje računaju FFT, autorima je od ranije poznat algoritam napisan u C programskom jeziku koji provjereno radi. Vjerovatno implemen-

Slika 3.1: Trajanje izračunavanja 128 uzoraka korištenjem FFT-a

tacija algoritma nije najbrža, ali je mnogo brža od naivne DFT implementacije. Dakle, kod za računanje FFT je preuzet sa interneta. Kod je izvorno napisan u C-u i koristi neke karakteristike programskog jezika C, koje nisu dostupne u programskom okruženju Keil, pa je kod bilo potrebno "prepraviti".

Nakon što je implementiran kod za računanje FFT, pokušali smo vidjeti da li smo dobili određena poboljšanja. Obzirom da je kompleksnost algoritma za računanje FFT-a O(N * log(N)), za očekivati je da će izvršavanje biti mnogo brže, ali koliko brže i da li je to ubrzanje dovoljno. Pa tako za 64 uzorka ulaznog signal računanje FFT-a traje 400 μs. Dok je naivna DFT implementacija trajala 8ms, što je 16 puta duže. Naravno što je broj uzoraka za obradu veći, razlika je veća. Pa tako za 128 uzoraka, FFT se izvrši za 1ms, što je oko 32 puta brže od DFT implementacije.

Vremena koja su ovdje prikazan su određena na način da se izlaz P15 pina mikrokontrolera postavi na HIGH prije početka izračunavanja diskretne Fourierove transformacije, a zatim se izlaz tog istog pina postavi na LOW kada se izračunavanje završi. Na osciloskopu je moguće izmjeriti vremena. Ovo je prikazano na slici (slika 3.1). Frekvencija između dva izračunavanja FFT-a je 62.5Hz, odnosno imamo 16 ms da izvršimo obradu svih signala, a mi trenutno na FFT potrošimo 1 ms. Zaključak je da nam je ostalo 15-tak ms za ostatak obrade. Nažalost, u trenutku snimanja slike 3.1 nismo posjedovali USB koji je mogao biti očitan od strane osciloskopa, pa je slika uslikana mobilnim telefonom...

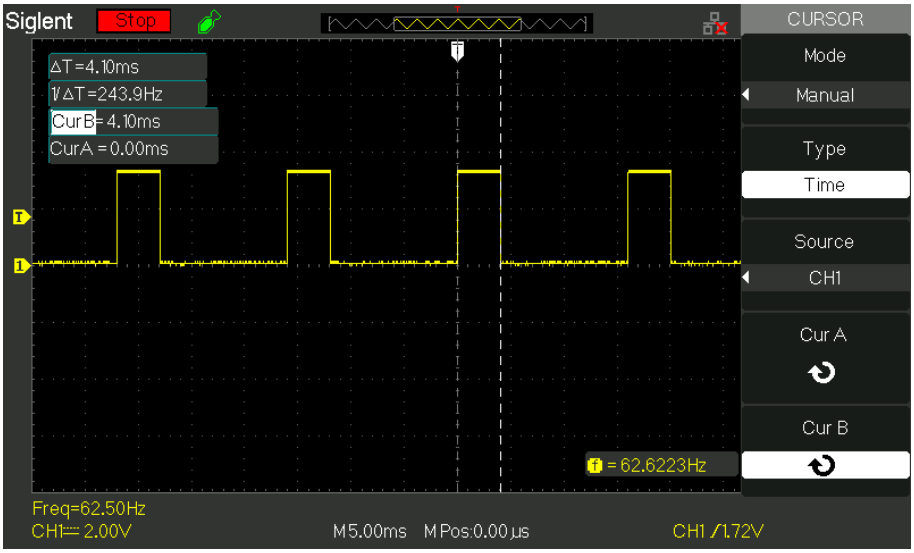
3.3 Implementacija MFCC

Sve stavke pobrojane u prethodnoj sekciji ovog rada vezane za MFCC je bilo potrebno samo zakodirati. Ipak "samo" nekad nije tako kratko i jednostavno kao što zvuči i izgleda ta riječ. Za preostalih 15ms, koliko smo imali na raspolaganju za ostatak obrade se na početku činilo kao relativno mnogo u odnosu na jednu ms potrošenu na računanje FFT-a.

U prvoj verziji smo se nadali sporoj egzekuciji, međutim da će biti toliko spora i autorima je bilo iznenađenje. Kod je skinut sa interneta i kao takav ima MIT licencu, što nam se na trenutak učilo kao dobra opcija. Međutim, traženje jednog Mel koeficijenta uz samo 20 filtera nad 32 uzorka je trajalo 15ms. Izračunavanje FFT-a za 32 uzorka je bilo neuporedivo brže, pa to ne treba ni spominjati. Ali zaključak je da sa ovakvom implementacijom ne bismo uopšte mogli raditi u realnom vremenu.

Nakon što smo uvidjeli problem (dvostruka for petlja sa mnogo kompleksnih množenja), posegnuli smo za najjednostavnijim rješenjem. To je bila lookup tabela. Jednostavno smo sva računanja vezana za trokutaste filtere izračunali offline, te iste ubacili u header fajl u vidu lookup tabele. Takvi koeficijenti su se samo množili sa FFT uzorcima.

Uz uvođenje ove lookup tabele koja je u našem radu veličine 48*128, ubrzanje programa je postalo enormno. Sada za računanje 5 Mel koeficijenata, uz korištenje 48 Mel filtera, nad 128 uzoraka FFT-a, je potrebno samo 4ms. Još nam je ostalo "slobodnih" 12ms, te su autori bili dovoljno slobodni da ulazni prije svega "pripreme" za bilo kakvu obradu. Pa je tako prije smještanja ulaza u niz za FFT obradu, svaki od njih podvrgnut Blackmanovom filteru, čime je konačni FFT dobio mnogo ljepši izgled, bez razlijevanja spektara. Nakon dodanog i ovog dijela, sveukupno računanje traje 4.1ms, što je prikazano na slici (slika 3.2).



Slika 3.2: Trajanje izračunavanja FFT + 5 MFCC koeficijenata

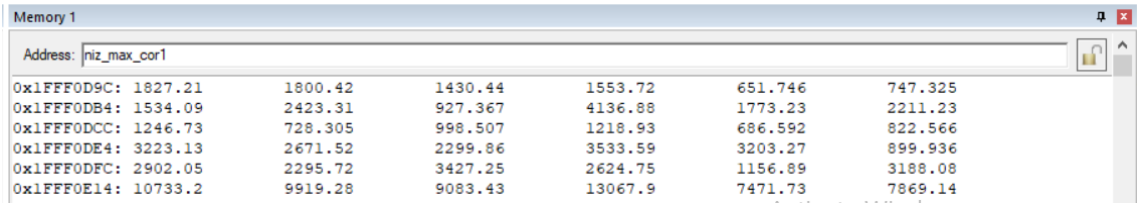
Eksperimentalni rezultati

Jednom kada smo u mogućnosti računati MFCC koeficijente u realnom vremenu, potrebno je uraditi nešto korisno sa njima. Kao što je u uvodu rečeno primjena MFCC koeficijenata na obradu zvuka je velika. Ipak, mi smo se odlučili da pokušamo prepoznati pjesmu nakon što pustimo npr 5 sekundi neke pjesme. Skup pjesama koje prepoznajemo smo proizvoljno odabrali na youtube-u.

Način na koji je vršeno prepoznavanje je dat u nastavku.

- Prvo smo za odabranu pjesmu (npr Korobeiniki - svima poznata Tetris tema) snimili isječak od deset sekundi.
- Zatim smo za taj isječak pronašli 5 MFCC koeficijenata. Odnosno, pronašli smo 10 * 8000 / 128 = 625 puta po 5 koeficijenata.
- MFCC koeficijenti za deset sekundi pjesme su zatim spašeni u obliku niza u header fajl.
- Sada se za odabrani interval vremena pjesme koja se želi prepoznati snimaju MFCC koeficijenti.
- Snimljeni koeficijenti se mogu uporediti sa koeficijentima koji su već ranije snimljeni. Za pjesmu za koju je poklapanje snimljenih koeficijenata najveće, proglašavamo da je ista "prepoznata"
- Metrika koja određuje najveće poklapanje je odabrana tako da bude što je moguće jednostavnija za implementaciju. Pa smo zbog toga pozvali ugrađenu funkciju u CMSIS-DSP biblioteci "arm_correlate_f32", koja određuje kroskorelaciju između dva jednodimenzionalna signala. Tako da smo za sve pjesme i svih 5 koeficijenata pronašli kroskorelacije, i odabrali one koji imaju najveću korelaciju.

Kada smo implementirali sve gore pobrojano, trajanje izvršavanja ne pralazi 8ms, te je zaključak da sistem radi u realnom vremenu. Na slici (slika 4.1) su vidljevi razultati nakon puštanja 6 sekundi pjesme pod rednim brojem 4. Kolone predstavlju koeficijente pojedinih pjesama (1 do 6), a redovi predstavljaju koeficijente za te pjesme (1 do 5). Uz iznimnku posljednjeg reda koji predstavlja sumarnu metriku za sve pjesme. Pjesma sa najvećim koeficijentom predstavlja pjesmu za koju program smatra da ima najveće pokalapanje. Vidimo da najveće poklapanje ima pjesma pod rednim brojem 4 (13068). Odnosno program je pravilo odredio o kojoj pjesmi se radi.



Slika 4.1: Trajanje izračunavanja FFT + 5 MFCC koeficijenata

Ovdje je bitno naglasiti, ukoliko se pusti 5 sekundi pjesme unutar intervala od snimljenih 10 sekundi, onda je pogađanje (na ovom malom testnom skupu 100 posto). Međutim ukoliko se pusti neki dio pjesme koji ne odgovara dijelu od snimljenih 10 sekundi (npr negdje na sredini pjesme i li na samom kraju) program pogriješi jednu od pet pjesama.

Zaključak

Kroz eksperimentalne rezultate se pokazala uspješnost metoda koje su detaljno obrađene u teorijekom i implementacijskom dijelu. Iako je skup nad kojim je vršeno testiranje oskudan, ipak je bio dovoljan da se ukaže na određene stavke od značaja. Rad smo usmjerili na prepoznavanje pjesama, ali nam se u toku izrade rada "rodila" želja da na neki način prepoznajemo govor. Željeli bismo u nekom narednom istraživanju napraviti aplikaciju koja će biti u mogućnosti govorni tekst na bosanskom jeziku konvertirati i ispisivati na neki lcd ekran. Ovo bi našlo primejnu kod ljudi koji slabije čuju, ili nikako ne čuju. Upravo iz ogih razloga samo se fokusirali na rad sistema u realnom vrmenu, jer je u ovim aplikacijama to ključno. Mnoga su poboljšanja, koja se mogu iskoristiti da bi se rad naprijedio, od kojih su neka:

- obogaćenje testnog skupa podataka, te potencijalno korištenje neuronskih mreža, i metoda mašinskog učenja.
- izdvajanje specifičnih značajki svake pjesme, a ne samo nekoliko sekundi proizvoljnog isječka iste
- korištenje većeg broja Mel koeficijenata koji će sigurno dati i neke dodatne informacije pored trenutnih
- korištenje prvog i drugog izvoda ovih koeficijenata, jer se pokazalo da takvi dodatni koeficijenti daju još neke vrlo zanimljive informacije o zvučnom signalu
- implementirati alogirtam za brže računanje FFT-a.
- prepoznavanje govornika, i u konačnici riječi na bosanskom jeziku koje govorink izgovara.

Literatura

[1] Laurent Daudet Bob L. Sturm Marcela Morvidone. “Musical Instrument Identification using Multiscale Mel-frequency Cepstral Coefficients”. Disertacija. 2010.

[2] URL: https://www.digikey.com/en/product-highlight/c/cypress/fm4-176l-s6e2cc-eth-arm-cortex-m4-mcu-starter-kit.

[3] Melita Ahić-Đokić. “Signali i sistemi”. Disertacija. Elektrotehnički fakultet Sarajevo, 2010.

[4] Emir Sokić. “Predavanja na predmetu Razvoj softvera za ugradbene sisteme”. Disertacija. Elektrotehnički fakultet Sarajevo, 2022. URL: https://c2.etf.unsa.ba/mod/resource/view.php?id=89363.

[5] URL: https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Introductory_Electrical_Engineering/Electrical_Engineering_(Johnson)/05%3A_Digital_Signal_Processing/5.08%3A_DFT_-_Computational_Complexity.

[6] URL: https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html.