TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kaido Siimer 164010IACB

KÕNELEJA VERIFITSEERIMINE ESP32 ARENDUSPLAADIL

Bakalaureusetöö

Juhendaja: Tanel Alumäe
PhD

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Kaido Siimer

15.05.2020

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks oli arendada kõneleja verifitseerimise süsteem, mis baseerub ESP32 mikrokontrollerit kasutaval arendusplaadil nimega ESP-EYE. Süsteem peab olema iseseisev ning kõik vajalikud arvutuslikud operatsioonid toimuvad mikrokontrolleris. Lähtuvalt nendest piirangutest ei ole süsteemi loomisel eesmärgiks niivõrd verifitseerimise täpsus vaid süsteemi loomine piiratud ressurssidega keskkonnas.

Töös antakse ülevaade kõneleja verifitseerimise meetoditest ning kirjeldatakse fraasipõhise kõneleja verifitseerimise meetodeid, mida kasutati süsteemi arendamiseks.

Tulemusena valmis tarkvara arendusplaadile, mis on suuteline verifitseerima vähemalt kahte erinevat kõnelejat nende enda vabalt valitud fraaside alusel. Verifitseerimise eelduseks on eelnevalt kõnelejate fraaside mudelite salvestamine seadmesse. Tarkvara on kirjutatud C/C++ keeles, kasutatud on Espressif asjade interneti arendusraamistikku.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 7 peatükki, 17 joonist, 1 tabelit.

Abstract

Speaker Verification on ESP32 Development Board

The aim of this thesis was to develop a speaker verification system that is based on ESP-EYE development board which is based on ESP32 microcontroller. The system must be independent, meaning all computational operations must be made on microcontroller. In accordance to those constraints the main aim is not the accuracy of the verification, but realization of the speaker verification system in constrained computational environment.

Thesis consists of overview about speaker verification methods and descriptions of text-dependent speaker verification methods used to develop this system.

As a result, software was developed for ESP-EYE development board, what can verify at least two different speakers on their freely chosen phrase. Enrollment of speaker's phrase model is needed before speaker verification can be performed. Software is written in C/C++, Espressif IoT Development Framework is used.

The thesis is in Estonian and contains 24 pages of text, 7 chapters, 17 figures, 1 table.

Lühendite ja mõistete sõnastik

AGC	*Adaptive gain control,* adaptiivne võimenduse reguleerimine
API	*Application programming interface,* Rakenduse/raamistiku poolt pakutav liidestus seadmele
Audio augmentation	Helinäidiste modifitseerimine vastavate algoritmidega, näiteks tempo muutmine, helikõrguse mõjutamine
CID	*Complexity-invariant distance,* aegrea keerukusest sõltumatu kaugus
DMA	*Direct memory access,* otsemällupöörduskanal
DNN	*Deep neural networks*, süva närvivõrgud
DTW	*Dynamic time warping*, dünaamiline ajadeformatsioon
ECM	*Electret Condenser Microphone,* analoogmikrofoni tüüp
ESP-IDF	*Espressif IoT development framework,* Asjade interneti arendusraamistik Espressifi seadmetele
ESP-WHO	*Espressif Systemsi* näo märkamise ning tuvastamise raamistik
FAR	*False acceptance rate*, valede õnnestumiste protsent
FatFs	Failisüsteemi liik, mis on kasutusel sardsüsteemides
FRR	*False rejection rate,* valede ebaõnnestumiste protsent
HMM	*Hidden markov model*, kõneleja verifitseerimises kasutatav meetod
I2S	*Inter-IC Sound,* integraalühendustes kasutatav digitaalheli edastus protokoll
MB	Megabait
MEMS	*Micro-Electro-Mechanical-System,* digitaalmikrofoni tüüp
MFCC	*Mel-frequency cepstral coefficients*, Mel-skaala kepstrikordajad, heli sageduslikud tunnusvektorid
ms	millisekund
PCM	*Pulse-code modulation,* impulss-koodmodulatsioon
QDTW	*Quantizied dynamic time warping*, kvanditud dünaamiline ajadeformatsioon
Tomp	Helitöötluses kasutatav võendite hulk
VAD	*Voice activity detection,* helis hääle tuvastuse algoritm

Wear leveling	Meetod mida kasutatakse mälude eluea pikendamiseks
VQ	*Vector quantization,* algoritm, mida kasutatakse signaalitöötluses, mustrisobitus ülesannetes etalon malli loomiseks

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Arvutiteaduse ajaloos üheks suureks ülesandeks on olnud ning on panna arvuti inimese kõnet mõistma. Kõne on inimeste põhiliseks suhtlusviisiks, nimelt sellel põhjusel on antud probleem olnud üheks läbivaks uurimisteemaks. Kõne mõistmise kõrvalt kerkib esile ka pisut teistlaadi ülesanne, milleks on kõneleja tuvastus ja verifitseerimine.

Seoses tehnoloogia ja arvutiteaduse arengutega ning asjade interneti ajastuga tuleb kasutusele järjest enam sardsüsteeme. Teema muudab aktuaalseks aina järjest süvenev sardsüsteemide kasutamine igapäevaelus. Olgu selleks kohvimasinad, aktiivsusmonitorid või nutitelefonid. Tehnika olulisuse kasv kaasab endaga ka tehnika isikustamise. Võib mõelda, et tore oleks kui näiteks kohvimasin saab kõnest aru, kellega on tegu, ning valmistab just vastavalt kõnelejale tema lemmikkohvi. Teiseks võimaluseks on näiteks kaheastmelise autentimise kasutamine, mille teiseks astmeks on kõneleja fraasipõhine verifitseerimine.

Seade, millele lõputöös tarkvara luuakse on laialt levinud ESP32 mikrokontrollerit kasutav arendusplaat. Varasemalt kõneleja verifitseerimise ülesannet täitvat tarkvara antud mikrokontrollerile avalikult leida pole.

Ülesande täitmiseks on kasutatud laialdaselt uuritud meetodeid, mis on hästi dokumenteeritud. Ülesannet on võimalik lahendada ka uudsemate vahenditega, kuid selles töös on põhinetud lihtsatele ning hästi mõistetavatele meetoditele.

Töö ülesandeks on luua tarkvara ESP32 mikrokontrollerile, mis suudaks teha kõneleja verifitseerimist lokaalselt ainult arendusplaadil asuvate ressurssidega.

Töös alustatakse ülesande kirjeldamiselt üldiselt liikudes järjest täpsemaks ning põhjendades tehtud samme, mis on vajalikud eesmärgi saavutamiseks. Olulisemateks infoallikateks on töös teadusartiklid ja veebiallikad.

Tänan kõiki lähedasi ja kursusekaaslaseid, kes olid nõus salvestama oma hääle lõputöö jaoks.

2 Kõneleja verifitseerimine

Kõneleja verifitseerimisele eelnevalt peab olema välja valitud kandidaatkõneleja, kelle kõnet verifitseerima hakatakse, seda protsessi nimetatakse kõneleja identifitseerimiseks. Kõneleja identifitseerimine on protsess, kus valitakse välja registreeritud kõnelejate seast kõige suurema tõenäosusega kõneleja kandidaat.

Kõneleja verifitseerimine on protsess, kus tuleb vastu võtta otsus, kas tegu on kõnelejaga, kelleks ta väidab end olevat. Verifitseerimise protsessis võrreldakse kandidaadi ning kõneleja kõnet vastavalt algoritmile, mida on kasutatud lahenduses. Algoritmiks võib olla näiteks kas tõenäosuspõhine otsus või kauguslävepõhine otsus. Kõneleja verifitseerimise võib jagada kaheks: fraasipõhiseks ning fraasist sõltumatuks.

2.1 Fraasipõhine kõneleja verifitseerimine

Fraasipõhiseks kõneleja verifitseerimiseks kasutatakse kindlaid fraase, mille helisalvestuste põhjal modelleeritakse kõneleja fraasi mudel. Meetodi eeliseks on kõneleja häälduses tulenevad erisused, mis on iseloomulikud just ainult kõnelejale. Puuduseks on muidugi kõneleja fraasi avalikult välja lausumine verifitseerimisel, mis kõneleja seisukohast on ebameeldiv (salvestamine, pealt kuulamine). Samuti vale tagasilükkamise korral, tuleb kõnelejal fraasi korrata, mis süsteemi halva täpsuse juures võib kõnelejale ebamugavust tekitada [1].

2.2 Fraasist sõltumatu kõneleja verifitseerimine

Fraasist sõltumatul verifitseerimisel kasutatakse kõneleja kõnet, ilma kindlatele fraasidele keskendumata. Antud meetod on pakkunud laiemat huvi, sest see on leidnud rakendust jälgimise ning kohtuekspertiisi valdkondades, kus on oluline verifitseerida kõnelejat kõne põhjal, mitte teatud fraasi põhjal. Fraasist sõltumatu verifitseerimise puhul põhineb kõneleja mudel statistilistel mudelitel. Fraasist sõltumatu verifitseerimine on mugavam kõnelejale, kuna kõneleja ei pea esitama kindlat fraasi vaid verifitseerimine toimub suvalise kõne põhjal [1].

3 Arendusplaadi valik

Tähtis on valida probleemi lahendamiseks piisavalt hea riistvara, millele tarkvara arendada. Arendusplaadi valikul olid oluliseks järgmised parameetrid, järjestatud vastavalt prioriteedile:

1) Mikrofoni olemasolu
2) Kasutab ESP32 mikrokontrollerit
3) Mikrofoni kvaliteet
4) Arendusplaadi mõõtmed
5) Hind

Töö olemusest tulenevalt on kõige kõrgema prioriteediga kohal mikrofoni olemasolu, sest tegemist on heli sisendit vajava protsessiga. ESP32 mikrokontrolleri kasutamine on autori eelistus tulenevalt varasemast kogemusest seda tüüpi mikrokontrolleritega. Oluliseks näitajaks mikrofoni kvaliteet, sest sellest sõltub sisendheli kvaliteet. Mida väiksemad on mõõtmed seda kergem on hiljem süsteemi integreerida mõne teise süsteemi osaks. Olulisuselt viimaseks on hind, mida madalam on hind, seda meeldivam on toode tarbijale. Arendusplaatide võrdlus on tabelis (Tabel 1). Arendusplaatide hinnad pärinevad veebikataloogist Mouser[1].

	ESP-EYE [2]	ESP32-LyraT v4.3 [3]	AI-Thinker A1S Audio Kit V2.2 [4]
Mikrofoni olemasolu, arv	Jah, 1 tk	Jah, 2 tk	Jah, 2 tk
ESP32 mikrokontroller	Jah	Jah	Jah
Mikrofoni kvaliteet / tüüp	MEMS I2S mikrofon (digitaalne)	ECM mikrofon (analoog)	ECM mikrofon (analoog)
Arendusplaadi mõõtmed	41 x 21 mm	95.5 x 80.6 mm	65 x 58 mm
Hind	17,91€	18,00€	16,26€

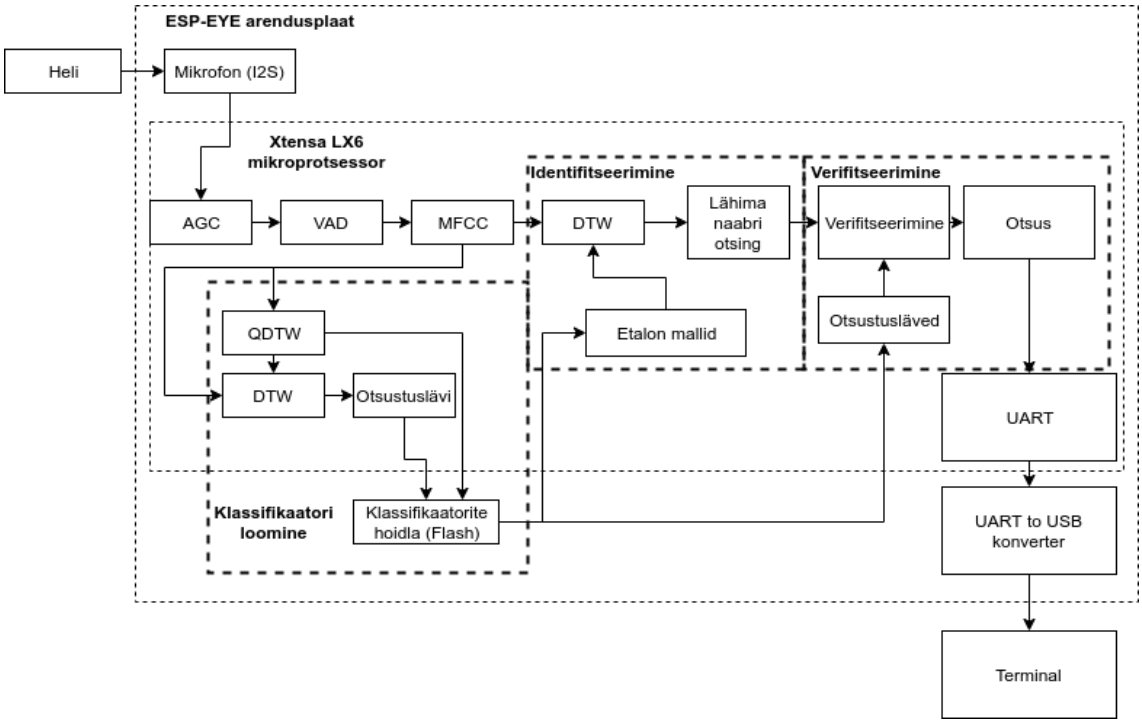
Tabel 1. Arendusplaatide võrdlus.

[1] https://www.mouser.ee

Valituks osutus **ESP-EYE**, kuna arendusplaadil on olemas digitaalne mikrofon ja arendusplaadi mõõtmed on kõige väiksemad. Lisaks on veel plaadil olemas kaamera, mida on võimalik tulevastes projektides kasutada. Arendusplaadi visuaal asub Lisas 2.

4 Fraasipõhine kõneleja verifitseerimise protsess

Selles peatükis kirjeldatakse süsteemis realiseeritud protsesside teooriat ning põhjendatakse valikuid, mida tehti tarkvara arendamisel. Protsessi plokkskeem on kujutatud joonisel (Joonis 1).



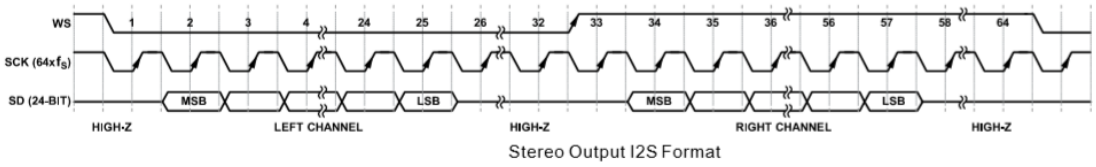
Joonis 1. Fraasipõhise kõneleja verifitseerimise protsessi plokkskeem.

4.1 Helisignaali töötlus

Antud töös on sisendsignaaliks heli. Heli on teatavasti analoogsignaal, töötlemiseks on vaja muundada analoogsignaal digitaalsignaaliks. Analoogsignaali muundamist digitaalseks nimetatakse diskreetimiseks. Diskreetimisel on oluline, et analoogsignaali oleks võimalik digitaalsignaalist taastada. Selle nõude täitmiseks peab olema täidetud Nyquist-Shannoni teoreem, diskreetimissagedus peab olema vähemalt kahekordne analoogsignaali kõige kõrgemast sagedusest [5]. Töös on kasutatud I2S (*Inter-IC Sound*) mikrofoni, mis diskreedib helisignaali mikrokontrolleris tehtavate operatsioonide jaoks.

4.1.1 I2S Mikrofon

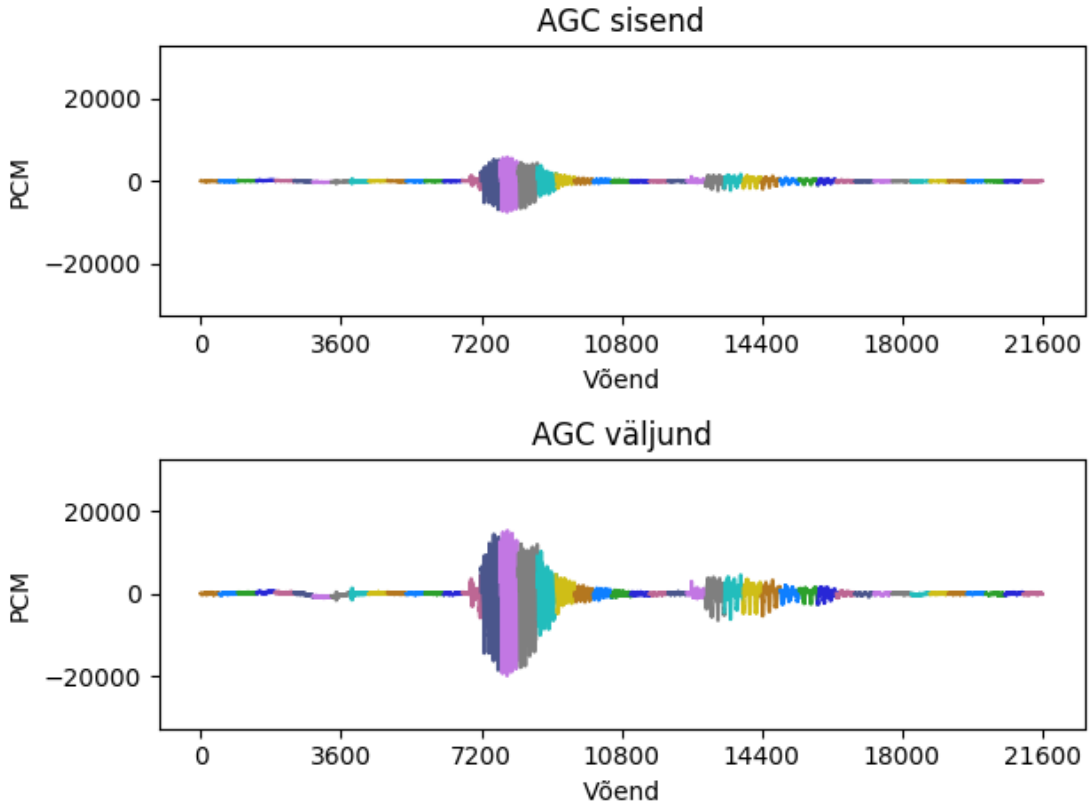
ESP-EYE Arendusplaadil on kasutuses MEMS mikrofon MSM261S4030H0R, mis edastab digitaal helisignaali I2S protokolliga [6]. Arendusplaadi valikul oli oluline, et mikrofon oleks võimalikult hea helikvaliteediga. I2S protokolli kasutava mikrofoni eeliseks on see, et mikrofon teeb heli diskreetimise ära, mis teeb kergemaks süsteemi programmeerimise, kuna heli diskreetimise eest on juba riistvara tasemel hoolt kantud. Heli signaali kätte saamiseks on vaja initsialiseerida I2S siin, kasutatud on ESP-IDF (*Espressif IoT development framework*) poolt pakutud I2S API (*Application programming interface*) kirjelduses pakutud funktsioone [7]. Vajaminevate parameetritega initsialiseerimis kood on võetud ESP-WHO näidisest, kus kasutatakse sama riistvara [8]. Vastavad parameetrid olenevad seadmest, mis kasutab heli saatmiseks I2S protokolli (Joonis 2), sellel juhul on tegemist MSM261S4030H0R mikrofoniga ning vastavad parameetrid tulevad seadme andmelehest. Andmelehes on kirjas ka I2S protokolli kirjeldus [9]. Diskreetimissageduseks on valitud 16 KHz, kuna inimkõne jääb sagedustele alla 8 KHz. Tuginedes Nyquist-Shannoni teoreemile peab olema diskreetimissagedus vähemalt kahekordne, on vastuseks 16 KHz. ESP32 mikrokontrolleris on eraldiseisev DMA (*Direct memory access*) üksus, mis võimaldab mikrofoni suhtluse mäluga ilma mikroprotsessorit kasutamata. Edasiseks töötluseks teisendatakse helisignaal 16-bittiseks PCM (*pulse-code modulation*) formaati signaaliks. Signaal jaotatakse 480 võendi kaupa tompudeks (*chunks*), mida edaspidi töödeldakse.



Joonis 2. I2S stereo väljund formaadi näidis

4.1.2 Kohanduv võimenduse reguleerimine (AGC)

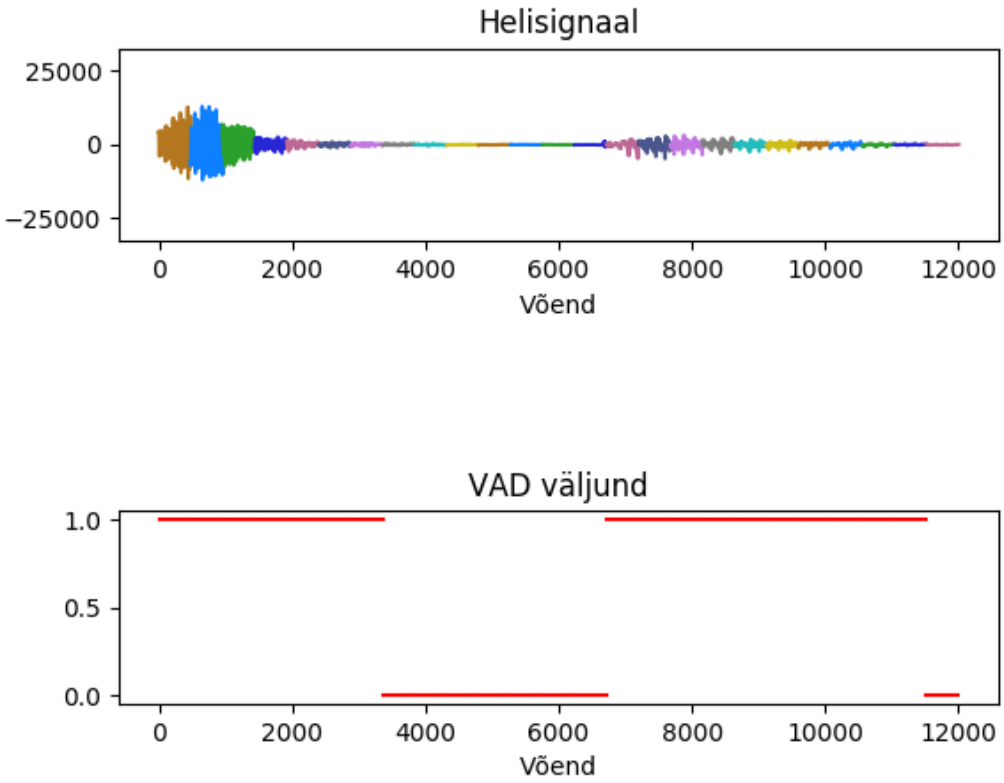
AGC (*adaptive gain control*) on meetod, mis vastavalt sisendsignaali võimsusele üritab hoida väljundsignaali pidevalt ühtlasel võimsusel. Meetod on kasutusele võetud, et vähendada sisendsignaali võimsuse varieerumisest tingitud väljundsignaali võimsuse muutuseid. Lihtsalt seletades võimendatakse nõrku sisendsignaale ning vähendatakse tugevaid sisendsignaale, selle tulemusena üritatakse hoida väljundsignaal ühtlasel tasemel. Antud töös on kasutatud mikrokontrolleri arendajate poolt loodud suletud lähtekoodiga teeki, mis on mõeldud helitöötluseks. Kuna tegemist on kinnise lähtekoodiga tarkvaraga, siis kasutaja jaoks on algoritmi täitev osa must kast ning sisemisest talitusest täpsem ülevaade puudub. Sisendiks funktsioonile on andmemassiiv võenditega ning väljundiks samuti modifitseeritud massiiv võenditega. Algoritmi rakendamist on näha joonisel (Joonis 3). Joonisel on erinevate värvidega kujutatud tompusid. Koodinäited ning dokumentatsioon on leitav mikrokontrolleri arendajate repositooriumis [10].



Joonis 3. AGC näidis sõnaga "Dattel".

4.1.3 Hääle aktiivsuse tuvastus (VAD)

VAD (*voice activity detection*) eesmärgiks on otsustada, kas helisignaal sisaldab kõnet või mitte [11]. Kuna eesmärk on püstitatud realisatsiooniks mikrokontrollerile, siis kõik kokkuhoidmised on teretulnud. Kuna fraas on kõne, siis huvi all on ainult helisignaal, milles sisaldub kõne. Kasutades VAD, saab eraldada signaalist ainult kõne sisaldava osa ning selle tõttu saavutatakse suurt kokkuhoidu. Ilma meetodit rakendamata, tuleks analüüsida läbi kogu diskreeditud helisignaal, sealhulgas ka vaikus. Lahenduses on kasutatud valmis kinnise lähtekoodiga tarkvara, mille on loonud mikrokontrolleri arendajad. Funktsiooni sisendiks on andmemassiiv võenditega ning on väljundiks on binaarne tõeväärtus väär juhul kui kõnet võendite tombus ei tuvastatud, tõene juhul kui kõne tuvastati (Joonis 4). Dokumentatsioon teegi kohta on leitav mikrokontrolleri arendajate repositooriumis [10].



Joonis 4. VAD näidis sõnaga "Dattel".

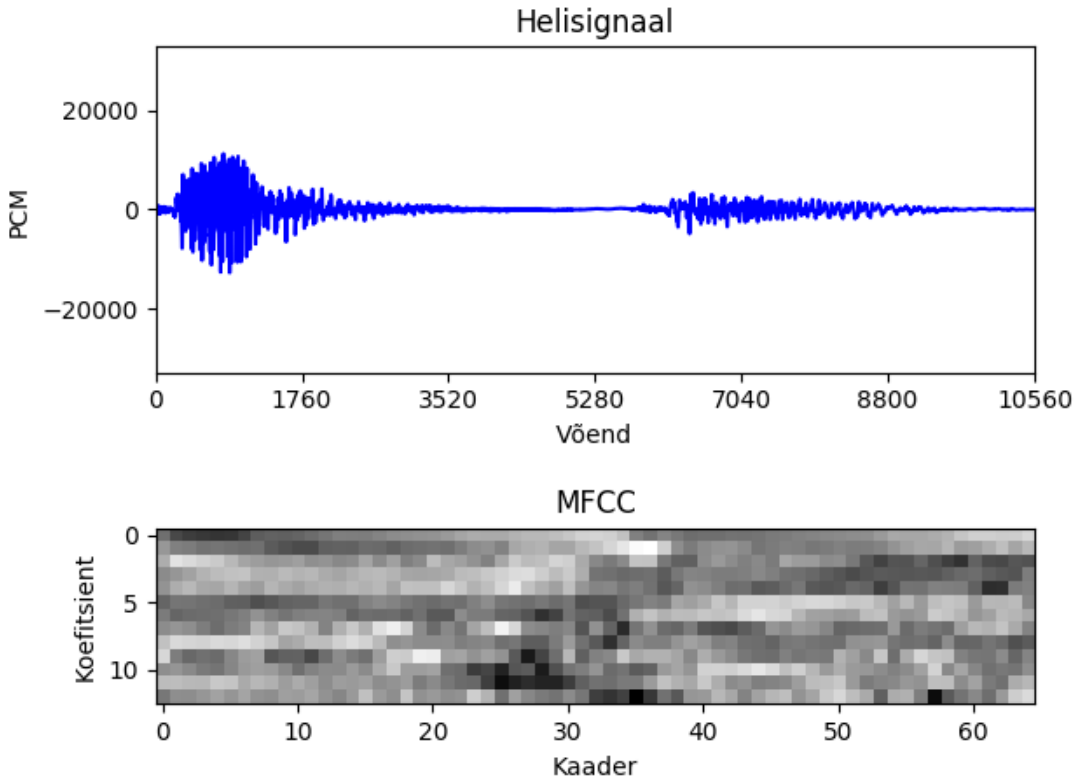
4.1.4 Fraasi tuvastamine

Fraasi tuvastamise eesmärgiks on teha kindlaks, millal kõneleja alustab fraasi ütlemist ning millal lõpetab. Alguse leidmiseks kasutatakse VAD funktsiooni väljundit. Kui väljund on tõene, ehk tombus sisaldus kõne, märgitakse vastavas muutujas, et fraasi algus on tuvastatud. Kõik tombud, kus VAD väljund on tõene, edastatakse puhvrisse, mille suuruseks on 64000 võendit, ajaliselt neli sekundit. Puhvri suurus määrab ära, kui kaua saab olla ajaliselt pikim fraas mida tuvastatakse. Fraasi lõpu tuvastamiseks peab esinema järjest kümme ilma kõneta tompu, ehk ajaliselt 300 ms (millisekundit) vaikust. Fraasi lõpu tuvastamiseks määratud aeg on tarkvaraliselt muudetav. Väärtused on seadistatud vastavalt autori katsetamiste käigus leitud väärtustega. Kui fraasi lõpp on tuvastatud alustatakse puhvrisse kogunenud signaalile akustiliste tunnusvektorite arvutamist.

4.1.5 Akustiline tunnusvektor (MFCC)

Esimeseks oluliseks sammuks mustrituvastuse ülesandes on tunnusjoonte leidmine. Tunnusjooned on tähtsad, kuna terves signaalis on tavaliselt palju rohkem informatsiooni kui seda mingi seaduspärasuse leidmiseks vaja on. Inimkõnes on näiteks oluliseks osaks lingvistiline osa kõnest. Näitena, kõneleja on siiski sama kõneleja, isegi kui tema kõnet mõjutab emotsioon. MFCC (*Mel-Frequency Cepstral Coefficients*) on üheks enim kasutatud tunnusjooneks, mida kasutatakse kõne- ning kõnelejatuvastuses. Kõneleja verifitseerimine on üheks kõnelejatuvastuse osaks (Joonis 5). Kuna helisignaal on ajas pidevalt muutuv, siis probleemi lihtsustamiseks eeldatakse, et lühikesel ajaskaalal, on helisignaal statistiliselt fikseeritud. MFCC kasutatakse tema heade omaduste poolest tuua välja kõnes sisalduvate foneemide tunnusjooned. MFCC arvestab inimese kuulmise iseärasustega, milleks on helitugevuse mittelineaarne tajumine ja helisageduse väheste muutuste halb tajumine [12]. Kanali mürade vähendamiseks on kasutatud *z-score* normaliseerimist. *Z-score* normaliseerimise valemiks on valem (1). Normaliseerimist rakendatakse igale MFCC vektorile eraldi.

$$z(i) = \frac{x(i) - \bar{x}}{S}, kus\ \bar{x}\ \text{on}\ vektori\ arit.keskmine, S\ on\ \text{vektori}\ standardhälve \quad (1)$$



Joonis 5. MFCC näidis sõnaga "Dattel".

Lahenduses on kasutatud viites [12] kirjeldatud eeskirja MFCC arvutamiseks. Kasutatud on *python_speech_features*[1] teegi C keelde porditud varianti nimega *c_speech_features*[2]. Teegis sisaldub kõik vajalik MFCC arvutuseks. MFCC arvutamisel on kasutatud järgnevaid parameetreid:

- Akna pikkus: 25 ms, 400 võendit
- Akna samm: 10 ms
- Koefitsientide arv: 13
- Aknafunktsioon: Hammingu aken [13] valem (2).

$$w(n) = 0.54 - 0.46\cos\left(2\pi\frac{n}{N}\right), 0 \leq n \leq N \quad (2)$$

Lahenduses ei ole kasutatud delta ega delta-delta koefitsiente, ressurssi kokkuhoiu eesmärgil. Võrdlusel kasutatud kaksteist vektorit, esimene vektor mis kirjeldab 0 Hz energiat on jäätud kasutamata.

[1] https://github.com/jameslyons/python_speech_features

[2] https://github.com/Cwiiis/c_speech_features

4.2 DTW (dünaamiline ajadeformatsioon)

DTW (*dynamic time warping*) on algoritm, mida kasutatakse aegridade võrdlemiseks. Algoritm kasutab dünaamilise programmeerimise põhimõtet, jagades suure probleemi väiksemateks osadeks. Algoritm sobib hästi kõne analüüsimiseks, kuna kõne akustiline tunnusvektor on aegrida. MFCC puhul vektor on aegrida. DTW headeks omadusteks algoritmi realiseerimise lihtsus ja võime toime tulla erineva pikkuse ning kiirusega aegridade võrdlemisel. Algoritmi suurimaks nimetatud puuduseks on aga kõrge arvutuskeerukus O(n * m), kus n ja m on aegridade pikkus. Õnneks on võimalik arvutuskeerukust vähendada teatud kitsenduste rakendamisega [14].

DTW näiteks kasutab autor juhendit veebimaterjalist [15]. Olgu meil kaks aegrida, mis on esitatud vektoritena **vektor1 [1, 2, 3, 5, 5, 5, 6]** ja **vektor2 [1, 1, 2, 2, 3, 5]**. Vektoritest koostatakse maatriks, kus üks vektor asetseb maatriksi esimesel real ning teine vektor maatriksi esimeses veerus. Järgmiseks täidetakse teine rida ning teine veerg väärtustega lõpmatus, maatriksi lahtrisse aadressiga Maatriks(2, 2), täidetakse arvuga null, kuna mõlemasse aegritta on lisatud esimeseks väärtuseks null (Joonis 6).

	0	1	1	2	2	3	5
0	0	inf	inf	inf	inf	inf	inf
1	inf						
2	inf						
3	inf						
5	inf						
5	inf						
5	inf						
6	inf						

Joonis 6. DTW ettevalmistatud kumulatiivne kauguste maatriks.

Lisamine on vajalik eeskirja (3) toimimiseks. Järgmiseks täidetakse maatriksi tühjad lahtrid järgides eeskirja (3). Saades tulemuseks täidetud maatriksi (Joonis 7).

$$Maatriks(i,j) = d(i,j) + \min\left(Maatriks(i-1,j), Maatriks(i-1,j-1), Maatriks(i,j-1)\right), \text{kus } d(i,j) = |vektor1(i) - vektor2(j)| \quad (3)$$

	0	1	1	2	2	3	5
0	0	inf	inf	inf	inf	inf	inf
1	inf	0	0	1	2	4	8
2	inf	1	1	0	0	1	4
3	inf	3	3	1	1	0	2
5	inf	7	7	4	4	2	0
5	inf	11	11	7	7	4	0
5	inf	15	15	10	10	6	0
6	inf	20	20	14	14	9	1

Joonis 7. DTW täidetud kumulatiivne kauguste maatriks.

Maatriksi viimases lahtris aadressiga Maatriks(9)(8) väärtuseks on minimaalne DTW kaugus, mis iseloomustab aegridade sarnasust. Loomulikult tuleb arvestada, et DTW algoritmiga arvutatud kaugus on suhteline, ehk kaugust mõjutavad aegridade pikkus kui ka nende väärtuste vahemik [15]. Mõjutuste ühtlustamiseks on võetud kasutusele CID (*complexity-invariant distance*). DTW kauguse saab teisendada CID kauguseks valemi (4) abil. Valem (4) koosneb omakorda DTW kauguse ning korrektsioonifaktori korrutisest, kus korrektsioonifaktor on kirjeldatud valemis (5). Korrektsioonifaktori leidmiseks omakorda on vaja leida keerukus hinnang aegreale, seda leitakse valemiga (6) [16].

$$CID(Q,C) = DTW(Q,C) \times CF(Q,C), kus\ Q\ ja\ C\ on\ aegread \quad (4)$$

$$CF(Q,C) = \frac{\max\left(CE(Q), CE(C)\right)}{\min\left(CE(Q), CE(C)\right)} \quad (5)$$

$$CE(Q) = \sqrt{\sum_{i=1}^{n-1}(q_i - q_{i+1})^2} \quad (6)$$

Liikudes tagasi maatriksi algusesse mööda maatriksis leiduvaid minimaalseid väärtusi, leitakse optimaalse tee, mis kirjeldab kuidas kaks aegrida kõige paremini kokku sobivad (Joonis 8). DTW puhul võrrelda ei andmepunkte vastavalt nende indeksitele (eukleidiline

kaugus) vaid parimale kokkusobivusele. Sellest omadusest tuleneb algoritmi võime tulla toime aegridadega, mis on erinevate kiirustega [15]. Optimaalne tee mängib olulist rolli QDTW (*quantized dynamid time warping*) algoritmi realiseerimises.

	0	1	1	2	2	3	5
0	0	inf	inf	inf	inf	inf	inf
1	inf	0	0	1	2	4	8
2	inf	1	1	0	0	1	4
3	inf	3	3	1	1	0	2
5	inf	7	7	4	4	2	0
5	inf	11	11	7	7	4	0
5	inf	15	15	10	10	6	0
6	inf	20	20	14	14	9	1

Joonis 8. DTW optimaalne tee kumulatiivses kauguste maatriksis.

Üheks enim kasutatud kitsenduseks on koolutusakna (*warping window*) kasutamine. Koolutusaken määrab ära, kui palju on lubatud aegridadel ajaliselt üksteise suhtes kõrvale kalduda. DTW maatriksis väljendab koolutusaken, kui palju peadiagonaalist lubatakse optimaalsel teel kõrvale kalduda. Koolutusaken sõltub kõrvale kaldest maatriksis tähistatud r ning aegrea mõõtmetest tähistatud n valem (7).

$$w = \frac{r}{n} \tag{7}$$

Antud valem kehtib, kui mõlemad aegread on ühe pikkused. Vastasel korral on võimalusi kaks, kas muuta aegread ühepikkusteks või võrrelda erinevate pikkustega aegridu. Erinevate pikkustega aegridade võrdlemisel tuleb teha lisasamme. Süsteemis kasutatud DTW arvutustel võetakse aluseks võrdsete aegridade meetod [14].

4.3 QDTW (kvanditud dünaamiline ajadeformatsioon)

QDTW eesmärgiks on koostada tsentroid, DTW arvutuse optimaalse tee põhjal (Joonis 8). Näidises kasutab autor samu vektoreid **vektor1 [1, 2, 3, 5, 5, 5, 6]** ja **vektor2 [1, 1, 2, 2, 3, 5]**. QDTW eeskiri põhineb kolmel juhtumil. Juhul kui optimaalsel teel on liikumine diagonaalis valem (8) või juhul kui optimaalsel teel on liikumine reas valem (9) või veerus valem (10). Valemeid (9, 10) on võimalik üldistada, kui optimaalse tee liikumine jätkub mööda veergu või rida rohkem kui kahel sammul [17]. Vektorite puhul on tsentroidiks samuti vektor. Rakendades eeskirja leitud optimaalsele teele ning vektoritele vektor1 ja vektor2 saame tulemuseks tsentroidi [1, 2, 3, 5.2] (Joonis 9).

$$tsentroid(i,j) = \frac{vektor1(i)+vektor2(j)}{2} \tag{8}$$

$$tsentroid(i,j) = \frac{vektor1(i)+vektor2(j)+ vektor2(j+1)}{3} \tag{9}$$

$$tsentroid(i,j) = \frac{vektor1(i)+vektor1(i+1)+ vektor2(j)}{3} \tag{10}$$

		0	1	1	2	2	3	5
	0	0	inf	inf	inf	inf	inf	inf
	1	inf	0	0	1	1	2	4
	2	inf	1	1	0	0	1	3
	3	inf	2	2	1	1	0	2
	5	inf	4	4	3	3	2	0
	5	inf	4	4	3	3	2	0
	5	inf	4	4	3	3	2	0
	6	inf	5	5	4	4	3	1

```
tsentroid = [
(1 + 1 + 1) / 3,
(2 + 2 + 2) / 3,
(3 + 3) / 2),
(5 + 5 + 5 + 5 + 6) / 5
]
```

Joonis 9. DTW optimaalne tee kauguste maatriksis koos arvutuskäikudega.

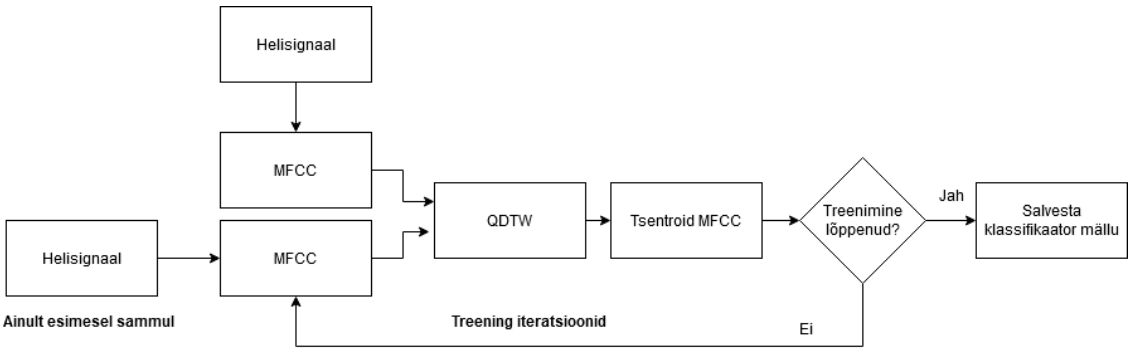
Uus nelja arvuga vektor kirjeldab mõlemat algselt esitatud vektorit. Tulemusena on näha, et QDTW aitab vähendada kasutatavat mäluruumi, kuna algselt oleks vaja olnud mällu salvestada kolmteist arvu.

4.4 Klassifikaator

Klassifikaatoriks on tavaliselt mingi üldistatud näidis, mis omab kindlaid tunnusjooni. Klassifikaatorit kasutatakse klassifitseerimise ülesandes, mis kujutab endast mingisuguste nähtuste rühmitamist. Rühmitamise aluseks on sarnaste tunnusjoonte esinemine klassifikaatoris ning rühmitatavas nähtuses. Kõneleja verifitseerimise puhul on vaja klassifitseerida kõneleja eelnevalt loodud klassifikaatori alusel. Antud töös on klassifikaatoriks korduvalt öeldud fraasi MFCC põhjal vastavalt teatud eeskirjale kohaldatud MFCC. Eeskiri on kirjeldatud järgnevas alapeatükis nimega Klassifikaatori treenimine. Klassifikaatori loomine koosneb kolmest sammust treenimisest, otsustusläve leidmisest ning klassifikaatori ja otsustusläve hoiustamisest arendusplaadil asuvasse välkmällu.

4.4.1 Klassifikaatori treenimine

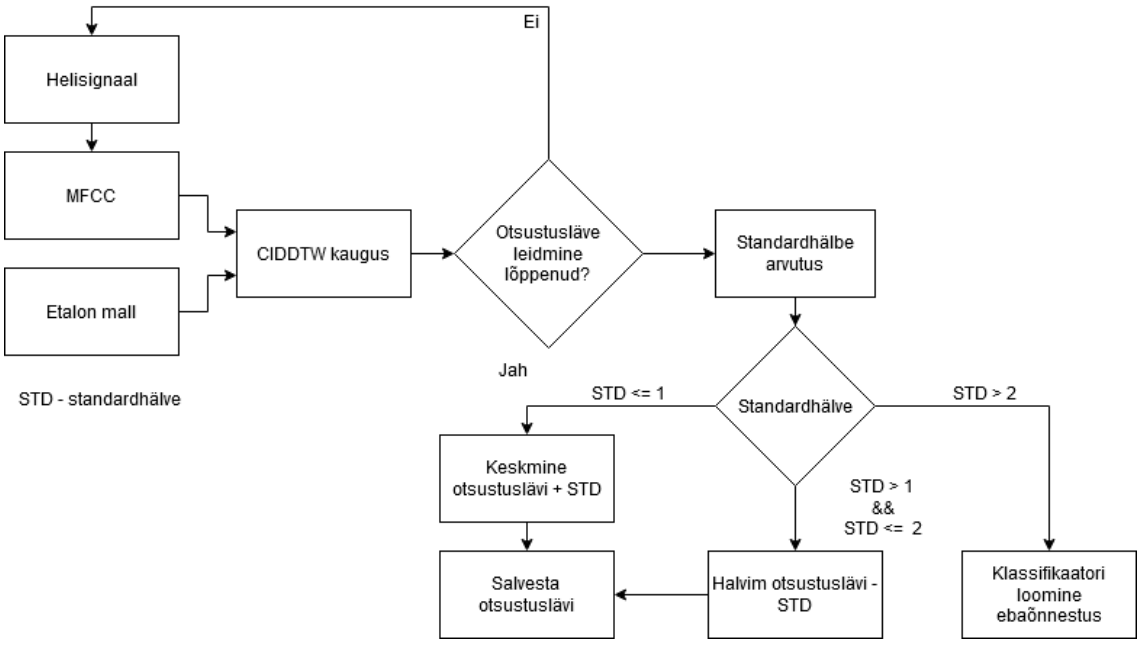
Kõneleja verifitseerimise ülesandes on keeruline valida kõneleja fraasile head etalon malli, mis annaks hea identifitseerimise täpsuse. Üheks täpsuse tõstmise võimaluseks on fraasi salvestatud näidiste arvu tõstmine. Sellega kaasneb koheselt mälu ning arvutusressursi kasv, mis oleks vastuolus sardsüsteemides kasutatavate arendamise tavadega (võimalikult väike ressursi kasutus). Treenimise eesmärgiks on kõneleja esitatud fraasi kordustest moodustada ainult üks etalon mall (tsentroid MFCC), mis ongi kõneleja fraasi klassifikaatoriks. Etalon malli arvutamiseks on kasutatud QDTW [17], mille eesmärgiks on hoida kokku mälu ning vähendada identifitseerimise etapis võrdluste arvu. QDTW eeskiri on kirjeldatud peatükis 4.3, vt lk 24. Treeningiteratsioonide arv on konfiguratsioonifailist muudetav. Lõputöös on treenimiseks kasutatud fraasi korduste arvuks neli. Joonisel (Joonis 10) illustreeritakse klassifikaatori treenimise protsessi.



Joonis 10. Klassifikaatori treenimine.

4.4.2 Otsustuslävede leidmine

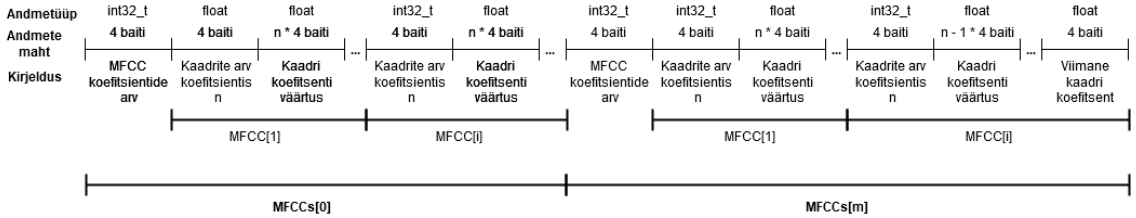
Otsustuslävede leidmine on oluline hilisemalt tehtava verifikatsiooni protsessi jaoks. Otsustuslävi on piir, millest suurema kaugusega kõneleja fraas lükatakse tagasi ehk tegemist ei ole sama kõnelejaga, kelleks väidab ta end olevat. Väiksema kauguse korral loetakse kõneleja samaks, kelleks ta väidab end olevat. Otsustuslävi leitakse pärast uue kõneleja klassifikaatori treenimist. Otsustusläved on kõneleja-spetsiifilised, igale kõnelejale arvutatakse isiklik lävi. Läve leidmiseks peab kõneleja kordama enda valitud fraasi, mille põhjal on loodud eelnevalt klassifikaator. Korduste arv, mida kõneleja kordama peab on tarkvaraliselt muudetav konfiguratsiooni failist. Korduste arvu tõstmisel on oodata paremat tulemust, kuid see muudab kõneleja jaoks oma fraasi lisamise ebamugavamaks. Iga korduse korral arvutatakse DTW algoritmiga kõneleja etalon malli ning fraasi korduse vaheline kaugus. Vastavalt otsustusläve kauguste standardhälbele tehakse otsus, kuidas otsustuslävi lõplikult paika panna. Kui hälve on madal (alla ühe kaugusühiku), siis arvutatakse kauguste keskmine ning sellele liidetakse standardhälve. Selline teguviis annab autori hinnangul parema tulemuse. Kui hälve on keskmine (ühe ja kahe kaugusühiku vahel) siis võetakse kõige kaugem otsustuslävi ning lahutatakse sellest standardhälve. Juhul kui hälve on liiga kõrge, siis klassifikaatori loomine ebaõnnestub, kuna tegemist oleks ebakvaliteetse klassifikaatoriga (Joonis 11).



Joonis 11. Otsustusläve leidmine.

4.4.3 Hoiustamine

Süsteemi toimimiseks on oluline, et süsteemil registreeritud kõnelejate klassifikaatorid ja otsustusläved säiliksid ka seadme taaskäivitamisel. Selle saavutamiseks on vaja klassifikaatorid ja otsustusläved kuhugi salvestada. Salvestamiseks sobib hästi arendusplaadil olev välkmälu, mille suuruseks on 4 MB. Mõlemad salvestamist vajavad andmed salvestatakse välkmällu kasutades ESP-IDF pakutavat *wear leveling* API funktsionaalsust. Programmeerimise lihtsustamiseks on kasutusel FatFs, mis võimaldab salvestada välkmällu faile. Iga uue kõneleja fraasi etalon malli lisamisel salvestatakse see koos otsustuslävega välkmällu. Välkmällu salvestamiseks on vaja C struktuur jadastada (Joonis 12), ning muutmällu laadimisel uuesti objektida kujule (Joonis 13). Seda joonisel kujutatud eeskirja alusel. Süsteemi taaskäivitamisel laetakse esimese asjana kõik salvestatud etalon mallid ja otsustusläved mikrokontrolleri muutmällu, tagades sellega süsteemi töövalmiduse [18].



Joonis 12. MFCC jadastus eeskiri.

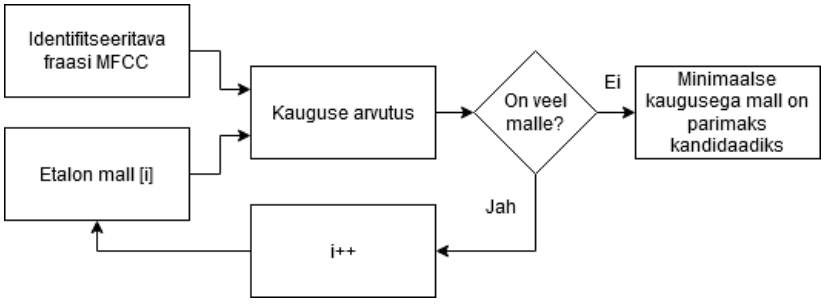
```
// Feature vector
typedef struct mfcc_vector {
  int32_t frames;
  float * values;
} mfcc_vector_t;

// Fingerprint
typedef struct mfcc_fingerprint {
  int32_t num_of_coeffs;
  mfcc_vector_t * vectors;
} mfcc_fingerprint_t;
```

Joonis 13. MFCC struktuuri kirjeldus C keeles.

4.5 Identifitseerimine

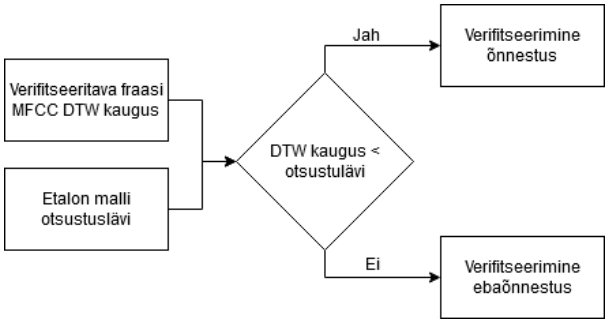
Kõneleja identifitseerimine on protsess, kus valitakse välja kõige sarnasem kõneleja etalon mall identifitseeritavale fraasile. Sarnasuse mõõdikuks on kõneleja esitatud fraasi ning salvestatud fraasi vaheline kaugus, tegemist on lähima naabri otsinguga (Joonis 14). Kauguse saamiseks kasutatakse DTW algoritmi optimeeritud realisatsiooni [19]. Kauguse arvutusel on DTW algoritmile rakendatud koolutusaken w = 0.15.



Joonis 14. Identifitseerimise protsess.

4.6 Verifitseerimine

Verifitseerimise protsessis võrreldakse parima kandidaadi otsustusläve väärtust identifitseerimise etapis leitud kaugusega. Kui kaugus on väiksem kui lävi, arvestatakse, et verifitseerimine õnnestus ning kõneleja on sama kõneleja (Joonis 15). Vastasel juhul ebaõnnestub verifitseerimine ning arvatavasti ei ole tegemist kõnelejaga, kelleks identifitseerimise etapis süsteem kõnelejat pidas.



Joonis 15. Verifitseerimise protsess.

5 Tulemused

Loodud süsteemi katsetamiseks kogus autor vabatahtlikelt lähedastelt ja kursusekaaslastelt fraaside salvestusi. Fraasideks olid eestikeelsed puuviljade nimetused. Iga kõneleja salvestas kaks fraasi. Üks, mis sildistati kõneleja tõeseks fraasiks ning teine fraas, mis oli mõne teise kõneleja tõeseks fraasiks. Igat kõneleja salvestas mõlema fraasi kohta kolm kordust. Autorit abistas viisteist kõnelejat. Katsetamiseks valis autor välja kümme kõige selgemate helisalvestistega fraasi. Helisalvestiste tegemisel kasutati arendusplaadil asuvat mikrofoni. Kuna igast fraasist oli kuus salvestust (kolm tõese kõneleja poolt, kolm petiskõneleja poolt), siis nägi autor vajalikuks helisalvestuste juurde loomist tehniliste võtete abil (*audio augmentation*). Klassifikaatori loomiseks läheb vaja kaheksat helisalvestist sama kõneleja poolt. Fraaside salvestisi kiirendati, aeglustati ja vähesel määral muudeti helikõrgust. Klassifikaatorid treeniti nelja fraasi korduse põhjal, otsustusläved leiti samuti nelja fraasi korduse põhjal. Identifitseerimise ja verifitseerimise protsesside hindamiseks ei kasutatud treenimiseks ega otsustuslävede leidmiseks kasutatud salvestusi.

Esmalt on esitatud süsteemi tehnilised tulemused, mis kirjeldavad tarkvara poolt kasutatava riistvara ressursside kasutust. Seejärel on esitatud arendatud tarkvara poolt täidetava ülesande hinnangud. Verifitseerimise ülesanne nõuab esmalt parimat kandidaati treenitud klassifikaatorite seast, seega esimene katsetuste etapp on määrata treenitud klassifikaatori täpsus. Teise etapina kontrolliti kõneleja verifitseerimise süsteemi täpsust.

5.1 Tehnilised

Süsteem suudab toimida ühel mikrokontrolleri tuumal, mis tähendab, et mikrokontrolleril on kasutamata üks tuum. See võimaldab mikrokontrolleri teisele tuumale rakendada veel lisafunktsionaalsust, näiteks oleks võimalik teisel tuumal realiseerida näotuvastust sooritav tarkvara. Olulisemad süsteemiga seotud tehnilised tulemused on järgmised:

- Protsessori taktsageduseks on 240MHz, keskmiselt võtab ühe etalon malli võrdluse tegemiseks 1,5 miljonit takti
- Muutmälu on mikrokontrolleril kasutada 4,4 MB, millest tarkvara poolt kasutuses on keskmiselt 210 kB
- Ühe kõneleja mudel keskmiselt 2 kB välkmälus

Muidugi oleneb võrdluse aeg võrreldavate andmete pikkusest (ajaliselt kauem kestev fraas nõuab rohkem võrdlusi). Ilma QDTW algoritmiga loodava klassifikaatoriga oleks vaja mälus hoida ühe kõneleja kohta nelja malli, keskmiselt MFCC on 4 kB suur, näitena sõna „Apelsin“ etalon malli loomisel mõõdetud andmed. MFCC suurused: 4532, 4064, 3752, 4532 baiti. Treenitud etalon mall vajab mälus hoidmiseks 2058 baiti.

Tehnilised tulemused näitavad, et antud ülesanne on mikrokontrolleri ressursikasutuse poolest hästi lahendatud.

5.2 Identifitseerimine

Identifitseerimise katsetusteks mängiti süsteemile helisalvestusi ning kontrolliti, kas süsteem arvab (*predict*) õigesti ära millise fraasiga tegu on. Helisalvestusi kasutati, sest kõnelejate kasutamine arendusprotsessis ja hilisematel katsetustel oli võimatu. Joonisel (Joonis 16) esitatakse katse tulemused segadusmaatriksis. Segadusmaatriksi peadiagonaalil asetsevad tõeste ennustuste arvud. Peadiagonaalist ülespoole jääksid valepositiivsed tulemused ning peadiagonaalist alla jääksid valenegatiivsed otsused. Kuna katsetustes treenitud klassifikaatori täpsus oli 100% arvutuskäik (11), siis peadiagonaalilt kõrvalekaldeid ei esinenud. Ideaalne täpsuse protsent tuleneb hästi valitud helisalvestustest mille põhjal klassifikaator treeniti. Mürarikkamates keskkondades oleks kindlasti klassifikaatori täpsus madalam.

$$Klassifikaatori\ täpsus = \frac{Õigesti\ arvatud\ fraasid}{Kõikide\ fraaside\ arv} = \frac{100}{100} * 100 = 100\% \qquad (11)$$

		Identifitseerimine									
		Tegelik fraas									
	Fraasid(kõnelejad)	Apelsin	Aprikoos	Dattel	Kiivi	Klementiin	Laim	Mandariin	Papaia	Viinamari	Õun
Arvatav fraas	Apelsin	10									
	Aprikoos		10								
	Dattel			10							
	Kiivi				10						
	Klementiin					10					
	Laim						10				
	Mandariin							10			
	Papaia								10		
	Viinamari									10	
	Õun										10

Joonis 16. Identifitseerimise segadusmaatriks.

5.3 Verifitseerimine

Verifitseerimise etapi katsetustel mängiti süsteemile samuti helisalvestusi. Kontrolliti, kui hästi suudab süsteem kõnelejaid verifitseerida. Katsetamiseks mängiti süsteemile esmalt kümme tõest salvestust ning märgiti tabelisse tõese kõneleja lahtrisse verifitseerimise tulemus. Pärast kümmet tõest salvestust mängiti süsteemile sama fraasiga, kuid teise kõnelejaga salvestus. Verifitseerimise protsessi tulemused märgiti petise (*imposter*) lahtrisse. Rohelise värvitooniga on märgitud õigeks määratud otsused ja punase tooniga on märgitud valeks määratud otsused (Joonis 17).

Verifitseerimine					
		Kõnelejad			
		Tõene		Imposter	
	Verifitseerimise tulemus	Õnnestus	Ebaõnnestus	Õnnestus	Ebaõnnestus
Fraasid	Apelsin	9	1	0	10
	Aprikoos	8	2	1	9
	Dattel	8	2	0	10
	Kiivi	9	1	0	10
	Klementiin	10	0	0	10
	Laim	9	1	0	10
	Mandariin	9	1	0	10
	Papaia	10	0	1	9
	Viinamari	8	2	0	10
	Õun	9	1	0	10
	Kokku	**89**	**11**	**2**	**98**

Joonis 17. Verifitseerimise tulemused.

Tulemustest saab välja arvutada valede ebaõnnestumiste protsendi (FRR) valem (12), valede õnnestumiste protsendi (FAR) valem (13) ning verifitseerimise süsteemi täpsuse (*accuracy*) valem (14).

$$FRR = \frac{Tõese\ kõneleja\ ebaõnnestumised}{Tõese\ kõneleja\ kõik\ verifitseerimis\ katsed} = \frac{11}{100} * 100 = 11\% \quad (12)$$

$$FAR = \frac{Imposter\ kõneleja\ õnnestumised}{Imposter\ kõneleja\ kõik\ katsed} = \frac{2}{100} * 100 = 2\% \quad (13)$$

$$Täpsus = \frac{Õiged\ otsused}{Kõik\ otsused} = \frac{(89+98)}{(89+11+2+98)} * 100 = 93.5\% \quad (14)$$

6 Järeldused ja edasiarendus

Lõputöö tulemustest võib järeldada, et püstitatud ülesanded said edukalt täidetud. Tehnilistest tulemustest on näha, et mikrokontrolleri ressurssidest ei jäänud ülesande lahendamisel puudu. Autor on nõus, et rohkemate kõnelejate võrdlusel on mõistlik võtta kasutusele meetodeid, mis kiirendaksid võrdlusi varakult arvutuse lõpetamisega, juhul kui saadakse aru, et kaugus on lubatud piiridest väljas [14]. Kuna tegemist oli arendusplaadiga piirduva süsteemiga, siis tulemustest selgub ka, et kiirendavaid lisameetmeid polnud vaja kasutusele võtta.

QDTW on hea meetod, mis aitab eelkõige säästa ruumi aegridade jaoks loodavate klassifikaatorite hoiustamisel mistahes mäludesse. Mälu kokkuhoid umbes 8 korda nelja fraasi korduse korral. Samuti tekib ka aja kokkuhoid, näiteks kõneleja identifitseerimise etapist tuleb teha üks võrdlus nelja asemel, kui ei kasutataks QDTW meetodit.

Lihtsamates ning ressursi kriitilisemates süsteemides on mõistlik kasutada lihtsamaid algoritme. DTW on idee poolest vägagi lihtne algoritm. Autor tõdeb fakti, et on olemas mõningaid tänapäevasemaid meetodeid kõneleja verifitseerimise probleemi lahendamiseks. Näiteks HMM (*Hidden Markov Model*) või DNN (Deep Neural networks) [1]. Mõlemad meetodid on teoorialt tunduvalt keerulisemad.

Klassifikaatorite täpsus 100% on liiga hea tulemus. Autor on kindel, et ebasoodsamates tingimustes klassifikaatorite täpsus ei püsi 100% peal. Katsed olid tehtud minimaalse müraga keskkonnas ning helisalvestised, mille põhjal klassifikaatorid treeniti, olid hoolikalt valitud.

Verifitseerimise täpsus 93.5% on mõjutatud suuremas osas tõese kõneleja verifitseerimise ebaõnnestumisest. Autor on meeldivalt üllatunud töö tulemustest. Töö põhiliseks eesmärgiks oli süsteemi loomine piiratud arvutusressurssidega, mitte kõrge verifitseerimise täpsus.

Kindlasti küsimus, mis tekib kõneleja verifitseerimise süsteemidega, on turvalisus. Kui kerge on antud süsteemi petta, verifitseerimaks kõnelejat kellekski, kes ta tegelikult pole.

Üheks ründeks kindlasti on kõneleja helisalvestamine ning süsteemile tagasimängimine. Selle probleemiga töös tegeletud ei ole. Oluline on veel mainida, et hääl on biomeetrilise tunnusena liialt varieeruv. Hääl on mõjutatud vanusest, õhutemperatuurist ja tujust. Sellel põhjusel ei ole seni väga laialt kasutuses ainult häälel põhinevad turvasüsteemid [20]. Autor samastub arvamusega ning ei soovitaks kasutada enda loodud süsteemi turvalahendusena.

Süsteemi edasiarenduse suundadeks võiks kindlasti olla võrdlustele kuluva aja vähendamine. Autor on uurinud ning viidanud töös meetoditele, mida oleks selle probleemi leevendamiseks võimalik kasutada.

Edasiarenduse võimalus on ka otsustuslävede leidmisel kasutatava loogika parendamises.

Kindlasti oleks edasiarenduseks samal arendusplaadil ka näotuvastuse funktsionaalsuse lisamine. Näotuvastuse lisamisega saaks näiteks kasutada arendusplaati mõnes teises süsteemis kaheastmelise autentimise lahendusena. Kindlasti sellise lahenduse puhul on vaja rõhku panna turvalisusele.

Üldisemalt edasiarendus oleks kasutada loodud süsteemi ning integreerida see mingi teise süsteemi osaks.

7 Kokkuvõte

Lõputöö tulemusena valmis ESP-EYE arendusplaadile kõneleja verifitseerimise süsteem, mis suudab eristada vähemalt kahte kõnelejalt neile vabalt valitud fraasi alusel. Süsteem ei vaja väliseid arvutus- ega mäluressursse. Kõneleja verifitseerimiseks kasutatakse DTW meetodit. Kõnelejate klassifikaatorite treenimiseks kasutatakse QDTW algoritmi ning iga kõneleja klassifikaatori treenimiseks kasutatakse nelja fraasi salvestust. Otsustuslävede leidmiseks kasutatakse samuti nelja fraasi salvestust. Tehtud klassifikaatorite ning otsustuslävede katsetamisel ei kasutatud treenimiseks

Töös loodud klassifikaatori täpsuseks on 100%, mis on kindlasti heade tingimuste tulemus. Verifitseerimise täpsuseks on 93.5%, mis on autori hinnangul hea tulemus. Mõningaid ebamugavusi kasutajale võib tekitada 11% suurune tõese kõneleja verifitseerimise ebaõnnestumine. Mis tähendab, et kõneleja peab verifitseerimiseks enda fraasi kordama. 2% suurune viga verifitseerida petiskõneleja tõese kõnelejana antud lahenduses on autori hinnangul piisavalt madal.

Täidetud sai eesmärk luua ESP32 mikrokontrollerite perekonnale kõneleja verifitseerimise süsteem. Avalikult saadaval olevat tarkvara selleks seni pole eksisteerinud. Autori loodud programmikoodi leiab Lisast 1. Süsteemi videodemonstratsioon on leitav Lisast 3.

Autor leiab, et töös õpitu ning läbi töötatud materjal tuleb edaspidiselt kasuks. Töö eesmärgid said edukalt täidetud.

Kasutatud kirjandus

[1] S. Furui, „Speaker recognition,“ *Scholarpedia,* kd. 3, nr 4, p. 3715, 2008.

[2] Espressif Systems Co., Ltd., „ESP-EYE Getting Started Guide,“ 4 Juuni 2019. [Võrgumaterjal]. Saadaval: https://github.com/espressif/esp-who/blob/master/docs/en/get-started/ESP-EYE_Getting_Started_Guide.md. [Kasutatud 28 Aprill 2020].

[3] Espresiff Systems Co., Ltd., „ESP32-LyraT V4.3 Getting Started Guide,“ [Võrgumaterjal]. Saadaval: https://docs.espressif.com/projects/esp-adf/en/latest/get-started/get-started-esp32-lyrat.html. [Kasutatud 28 Aprill 2020].

[4] AI-Thinker, „ESP32-Audio-kit,“ 03 09 2019. [Võrgumaterjal]. Saadaval: http://wiki.ai-thinker.com/esp32-audio-kit. [Kasutatud 28 04 2020].

[5] B. Olshausen, „Aliasing,“ 10 10 2000. [Võrgumaterjal]. Saadaval: https://www.rctn.org/bruno/npb261/aliasing.pdf. [Kasutatud 28 04 2020].

[6] Espressif Systems Co., Ltd., „ESP-EYE_V2.1 Reference Design,“ 24 03 2020. [Võrgumaterjal]. Saadaval: https://www.espressif.com/sites/default/files/documentation/ESP-EYE_V2.1_Reference_Design_0.zip. [Kasutatud 28 04 2020].

[7] Espressif Systems Co., Ltd., „I2S,“ 11 12 2019. [Võrgumaterjal]. Saadaval: https://docs.espressif.com/projects/esp-idf/en/v3.3.1/api-reference/peripherals/i2s.html. [Kasutatud 28 04 2020].

[8] Espressif Systems Co., Ltd., „ESP-WHO recognition solution example,“ 17 02 2020. [Võrgumaterjal]. Saadaval: https://github.com/espressif/esp-who/blob/master/examples/single_chip/recognition_solution/main/app_speech_recsrc.c. [Kasutatud 28 04 2020].

[9] MEMSensing Microsystems Co., Ltd., „MSM261S4030H0R Data sheet,“ 01 04 2017. [Võrgumaterjal]. Saadaval: https://www.memsensing.com/webupfile/15006185591557240879.0-ENG.pdf. [Kasutatud 28 04 2020].

[10] Espressif Systems Co., Ltd., „ESP Acoustic Algorithm,“ 31 03 2020. [Võrgumaterjal]. Saadaval: https://github.com/espressif/esp-sr/tree/master/acoustic_algorithm. [Kasutatud 28 04 2020].

[11] J. Lyons, „Voice Activity Detection (VAD) Tutorial,“ 2012. [Võrgumaterjal]. Saadaval: http://practicalcryptography.com/miscellaneous/machine-learning/voice-activity-detection-vad-tutorial/. [Kasutatud 28 04 2020].

[12] J. Lyons, „Mel Frequency Cepstral Coefficient (MFCC) tutorial,“ 13 03 2013. [Võrgumaterjal]. Saadaval: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. [Kasutatud 28 04 2020].

[13] MathWorks®, „Hamming window,“ 2006. [Võrgumaterjal]. Saadaval: https://www.mathworks.com/help/signal/ref/hamming.html. [Kasutatud 06 05 2020].

[14] A. Mueen ja E. J. Keogh, „Extracting Optimal Performance from Dynamic Time Warping,“ %1 *22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, San Francisco, California, 2016.

[15] Rip Tutorial, „Introduction To Dynamic Time Warping,“ [Võrgumaterjal]. Saadaval: https://riptutorial.com/algorithm/example/24981/introduction-to-dynamic-time-warping. [Kasutatud 03 05 2020].

[16] X. W. E. J. K. Gustavo E.A.P.A. Batist, „CID: An efficient complexity-invariant distance for time series,“ *Data Mining and Knowledge Discovery*, kd. 28, 2013.

[17] T. Zaharia, S. Segarceanu, M. Cotescu ja A. Spataru, „Quantized Dynamic Time Warping (DTW) algorithm,“ *2010 8th International Conference on Communications*, pp. 91-94, 2010.

[18] Espressif Systems Co., Ltd., „Wear levelling example,“ 31 10 2018. [Võrgumaterjal]. Saadaval: https://github.com/espressif/esp-idf/tree/v3.3.1/examples/storage/wear_levelling. [Kasutatud 03 05 2020].

[19] T. Rakthanmanon, B. Campana, A. Mueen, Q. Zhu, J. Zakaria, E. Keogh, G. Batista ja B. Westover, „The UCR Suite,“ 2012. [Võrgumaterjal]. Saadaval: http://www.cs.ucr.edu/~eamonn/UCRsuite.html. [Kasutatud 29 04 2020].

[20] A. Nordrum, „Automatic Speaker Verification Systems Can Be Fooled by Disguising Your Voice,“ IEEE Spectrum, 20 11 2017. [Võrgumaterjal]. Saadaval: https://spectrum.ieee.org/tech-talk/telecom/security/automatic-speaker-verification-systems-can-be-fooled-by-disguising-your-voice. [Kasutatud 05 05 2020].

Lisa 1 – Lähtekood

Lõputöö käigus arendatud tarkvara on leitav autori Github repositooriumist[1].

[1] https://github.com/kasiim/ESP-EYE-speaker-verification

Lisa 2 – Arendusplaadi visuaal

Lisa 3 – Videodemonstratsioon

Süsteemi toimimist kirjeldav videodemonstratsioon asub järgneval veebiaadressil: https://www.youtube.com/watch?v=--DIeC4df1U