TUGAS AKHIR – EE 184801

SISTEM PENGENALAN SUARA UNTUK PERINTAH PADA ROBOT SEPAK BOLA BERODA

Muhammad Azhar Ismail
NRP 07111540007002

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng.
Fajar Budiman, ST., M.Sc.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – EE 184801

SISTEM PENGENALAN SUARA UNTUK PERINTAH PADA ROBOT SEPAK BOLA BERODA

Muhammad Azhar Ismail
NRP 07111540007002

Dosen Pembimbing
Dr. Ir. Djoko Purwanto, M.Eng.
Fajar Budiman, ST., M.Sc.

DEPATERMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT – EE 184801

VOICE RECOGNITION SYSTEM FOR COMMAND ON WHEELED SOCCER ROBOT

Muhammad Azhar Ismail
NRP 07111540007002

Supervisor
Dr. Ir. Djoko Purwanto, M.Eng.
Fajar Budiman, ST., M.Sc.

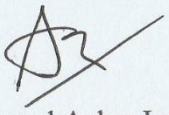
ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology
Sepuluh Nopember Institute of Technology
Surabaya 2019

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Sistem Penganalan Suara untuk Perintah pada Robot Sepak Bola Beroda" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Mei 2019



Muhammad Azhar Ismail
NRP. 07111540007002

SISTEM PENGENALAN SUARA UNTUK PERINTAH PADA ROBOT SEPAK BOLA BERODA

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. Djoko Purwanto, M.Eng.	**Fajar Budiman, ST., M.Sc.**
NIP. 196904261994031003	**NIP. 196904261994031003**

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO

SURABAYA
MEI, 2019

SISTEM PENGENALAN SUARA UNTUK PERINTAH PADA ROBOT SEPAK BOLA BERODA

Nama : Muhammad Azhar Ismail
Pembimbing I : Dr. Ir. Djoko Purwanto, M.Eng.
Pembimbing II : Fajar Budiman, ST., M.Sc.

ABSTRAK

Pemberian perintah kepada robot sepak bola beroda merupakan salah satu hal penting yang dapat digunakan untuk menentukan strategi permainan pada robot. Penyampaian perintah dari manusia menuju robot disebut dengan *high level human coaching*. Salah satu metode yang dapat digunakan untuk melakukan hal tersebut adalah dengan memberikan perintah suara.

Pada penelitian tugas akhir dirancang sebuah sistem yang dapat digunakan untuk mengenali perintah suara yang diberikan oleh manusia kepada robot. Mikrofon digunakan sebagai sensor untuk menangkap perintah suara yang disampaikan. Selain itu digunakan metode VAD (*Voice Activity Detector*) untuk merekam perintah suara yang disampaikan. Fitur pada setiap rekaman suara akan diambil menggunakan metode MFCC (*Mel Frequency Cepstral Coefficients*). Fitur tersebut akan digunakan sebagai input dari CNN (*Convolutional Neural Network*). Metode CNN digunakan agar sistem dapat mengenali setiap perintah suara yang disampaiakn berdasarkan fitur yang didapatkan dari proses MFCC. Sistem pengenalan suara ini akan menjadikan robot dapat melakukan beberapa perintah, berdasarkan suara yang disampaikan.

Berdasarkan pengujian yang telah dilakukan pada 9 orang dengan 7 suara perintah berbeda, didapatkan hasil bahwa sistem pengenalan suara ini mampu mengenali setiap perintah tersebut dengan akurasi keberhasilan rata-rata 83% untuk semua pengujian. Sedangkan untuk mengenali orang yang memberikan perintah didapat akurasi keberhasilan rata-rata sebesar 80% pada 9 orang yang dilakukan pengujian.

Kata kunci: pengenalan perintah suara, *voice activity detector* (VAD), *mel frequency cepstral coefficients* (MFCC), *convolutional neural network* (CNN)

.....Halaman ini sengaja dikosongkan.....

VOICE RECOGNITION SYSTEM FOR COMMANDS ON WHEELED SOCCER ROBOT

Name	: Muhammad Azhar Ismail
Supervisor	: Dr. Ir. Djoko Purwanto, M.Eng.
Co-supervisor	: Fajar Budiman, ST., M.Sc.

ABSTRAC

Giving orders to wheeled soccer robots is one of the important things that can be used to decide the strategy of the game on robot. Deliver orders from humans to robots is called high level human coaching. Method that can be used to do this is by giving voice commands.

In this final assignment research the system is designed that can be used to recognize voice commands given by humans to robots. Microphones are used as sensors to capture voice commands. In addition, the VAD (Voice Activity Detector) method is used to record voice commands. The features of each recorded sound will be taken using the MFCC (Mel Frequency Cepstral Coefficients) method. This feature will be used as input from CNN (Convolutional Neural Network). The CNN method is used so that the system can recognize each voice command based on the features obtained from the MFCC process. This voice recognition system will make the robot can do several commands, based on the delivered sound.

Based on the tests carried out on 9 people with 7 different command sounds, the results were obtained that the voice recognition system was able to recognize each of these commands with an average success of 83% for all tests. Whereas to recognize people who give orders, the accuracy of success is 80% on 9 people who were tested.

***Keywords**: voice command recognition, voice activity detector (VAD), mel frequency cepstral coefficients (MFCC), convolutional neural network (CNN)*

.....Halaman ini sengaja dikosongkan.....

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya penulis dapat menyelesaikan Penelitian Tugas Akhir ini dengan judul:

SISTEM PENGENALAN SUARA UNTUK PERINTAH PADA ROBOT SEPAK BOLA BERODA

Penelitian tugas akhir ini dibuat untuk memenuhi kurikulum di Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember sebagai salah satu syarat untuk menyelesaikan program studi Strata-I.

Pada kesempatan ini, penulis ingin berterimakasih kepada pihak-pihak yang telah membantu dan mendukung dalam proses pembuatan penelitian tugas akhir ini kepada:

1. Dr. Ir. Djoko Purwanto, M.Eng. sebagai dosen pembimbing 1, atas bimbingan dan arahanya selama mengerjakan penelitian tugas akhir ini.
2. Fajar Budiman, ST., M.Sc. sebagai dosen pembimbing 2, atas bimbingan dan arahanya selama mengerjakan penelitian tugas akhir ini.
3. Ir. Tasripan, M.T. selaku kepala Laboratorium Elektonika Cerdas
4. Dr.Eng. Ardyono Priyadi, S.T., M.Eng. selaku Ketua Departemen Teknik Elektro ITS Surabaya.
5. Seluruh dosen Departemen Teknik Elektro ITS Surabaya.
6. Rekan-rekan Laboratorium Elektronika yang telah membantu peneleitian tugas akhir ini.

Penulis sadar bahwa penelitian tugas akhir ini masih memiliki kekurangan. Saran, kritik dan masukan dari semua pihak sangat membantu untuk pengembangan lebih lanjut. Semoga penelitian tugas akhir ini dapat memberikan manfaat bagi banyak pihak.

Surabaya, 20 Mei 2019

Penulis

.....Halaman ini sengaja dikosongkan.....

DAFTAR ISI

.....Halaman ini sengaja dikosongkan.....

DAFTAR GAMBAR

.....Halaman ini sengaja dikosongkan.....

DAFTAR LABEL

.....Halaman ini sengaja dikosongkan.....

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Kontes Robot Speak Bola Indonesia Beroda ini merupakan salah satu kegiatan yang merupakan bagian dari Kontes Robot Indonesia (KRI) sebagai ajang kompetisi rancang bangun dan rekayasa dalam bidang robotika [1].Dalam Kontes Robot Sepak Bola Indonesia Beroda robot difungsikan untuk bermain speak bola seperti permainan speak bola pada manusia. Kontes Robot Speak Bola Indonesia Beroda diselenggarakan berdasarkan aturan yang dilakukan di RoboCup Middle Size League (MSL), dengan menyesuaikan kondisi di Indonesia, misalnya pada ukuran lapangan dan lainnya [1]. Jadi robot akan bermain speak bola dengan aturan permainan yang telah ditetapkan. Didalam aturan yang telah dibuat, untuk menentukan tim pemenang adalah dengan menghitung jumlah gol terbanyak yang dicetak oleh setiap tim [1], untuk itu dibutuhkan strategi permainan yang efektif untuk mencetak gol sebanyak-banyaknya. Setiap tim yang mengikuti Kontes Robot Speak Bola Indonesia Beroda memiliki strategi permainan tertentu yang berbeda antara tim satu dengan lainya. Namun apabila hanya menggunakan satu strategi atau pola permainan yang sama, maka strategi permainan tersebut akan mudah dibaca oleh tim lawan. Sehingga tim lawan akan lebih mudah untuk mencari cara untuk melewati pertahanan dan celah untuk mencetak gol.

Untuk mengatasi hal tersebut maka dibutuhkan perintah dari pelatih untuk mengubah startegi atau pola permainan ketika pertandingan sedang berlangsung. Perintah ini dapat disampaikan menuju robot melalui *High Level human Coaching*. *High Level human Coaching* adalah metode yang dapat digunakan untuk menyampaikan perintah kepada robot [2]. Hal ini sudah sesuai dengan peraturan yang ada pada RoboCup Middle Size League (MSL). Jadi untuk memberikan perintah pada robot dapat melalui dua cara, yaitu dapat menggunakan suara atau gerakan tangan serta hanya diperbolehkan satu orang dari masing-masing tim untuk memberikan perintah pada robot [2]. Selain dapat digunakan untuk

memberi perintah strategi permainan. *High Level human Coaching* juga dapat digunakan untuk memposisikan ulang robot yang *error* [2], sehingga ketika permainan memasuki bola hidup, maka robot dapat bergerak dengan normal kembali dan mencari posisi yang tepat untuk mencetak gol, namun *High Level human Coaching* hanya diperbolehkan ketika terjadi bola mati [2].

Dalam penelitian tugas akhir ini, dirancang sebuah sistem *High Level human Coaching* dengan menggunakan perintah suara. Perintah yang disampaikan dapat berupa strategi permainan atau memposisikan ulang robot. Untuk mengenali perintah suara yang diberikan, maka digunakan sebuah metode *Voice recognition* agar robot dapat membedakan setiap perintah yang disampaikan oleh pelatihnya. Dalam penerimaan perintah suara tersebut nantinya akan dikenali jenis perintah dan suara pelatih, sehingga tidak akan terjadi kesalahan atas perintah yang diterima. Untuk itu dibutuhkan sebuah sistem yang bersifat *privacy* agar tidak semua orang dapat mengaksesnya. Perintah yang akan dijalankan adalah perintah yang hanya disampaikan oleh pelatih dan akan mengabaikan perintah yang disampaikan orang lain.

1.2 Rumusan Masalah

Permasalahan yang dibahas pada penelitian tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara robot sepak bola beroda dapat mengenali suara pelatihnya
2. Bagaimana proses pengiriman perintah dari pelatih menuju robot

1.3 Tujuan

Tujuan yang ingin dicapai dari penelitian tugas akhir ini adalah sebagai berikut:

1. Robot sepak bola beroda dapat mengenali suara pelatihnya
2. Robot sepak bola beroda dapat mengenali perintah apa yang diucapkan pelatih

1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Perintah yang diucapkan berupa sebuah kata
2. Volume suara yang digunakan pada pengambilan dan pengujian adalah volume suara normal.

1.5 Metodologi Penilitian

Metodologi yang digunakan pada penelitian tugas akhir ini adalah senagai berikut:

- Studi literatur

  Pada tahap studi literature ini dilakukan pengumpulan observasi, dan pengkajian teori pada data dan penelitian yang terjamin serta relevan untuk mendukung tugas akhir ini. Sumber yang diambil dan dipakai dapat berasal dari jurnal, buku, dan artikel yang berasal dari pemerintah, institusi, dan sumber terpecaya lainya. Sumber yang digunakan berisi tentang pemrograman python dan metode pengenalan suara dengan menggunakan *Mel Frequency Cepstral Coefficients* dan *Convolutional Neural Network*.

- Perancangan sistem

  Peracangan sistem bertujuan untuk mengenali suara dari orang tertentu. Pertama sistem akan melakukan proses belajar dari dataset suara untuk training dari beberapa orang yang telah disiapkan. Namun sebelum dataset suara tersebut digunakan sebagai inputan dari *Convolutional Neural Network*, data suara tersebut diolah dahulu menggunakan metode *Mel Frequency Cepstral Coefficients*, hasil dari metode tersebut kemudian akan digunakan sebagai inputan dari CNN. Setelah melalu proses belajar tersebut, sistem akan di ujicoba menggunakan dataset suara yang belum pernah diketahui untuk mengetahui seberapa besar akurasi ketepatan dari sistem yang dibuat.

- Pembuatan program

  Program yang digunakan adalah bahasa pemrograman python. Dalam pemrograman ini dibagai menjadi dua bagian yaitu program untuk training data dan program untuk *test* data. Untuk mengolah data suara sendiri digunakan *library speech recognition* dan untuk *training* data digunakan library Keras.

- Pengumpulan data

  Mikrofon diguakan untuk merekam data suara yang akan dipakai pada pelatihan jaringan syaraf tiruan. Data yang akan dikumpulkan adalah data suara dari 5 orang berbeda, masing-masing orang memberi 7 perintah yang sama, setiap perintah terdiri dari 18 data suara. Jadi total data suara yang akan digunakan terdapat 525 data suara dan setiap satu perintah akan digunakan sebagai data test, berarti 420 data suara akan digunakan sebagai data training dan 105 data suara akan digunakan untuk test.
- Pelatihan jaringan syaraf tiruan

  Untuk melakukan pelatihan, data yang telah dikumpulkan akan diambil fitur utamanya dengan metode MFCC menggunakan API Librosa. Selanjutnya fitur tersebut digunakan sebagai input dari jaringan syaraf tiruan. Seluruh proses pelatihan jaringan syaraf tiruan memanfaatkan API Keras. Sedangkan untuk menjalankan proses pelatihan data, digunakan *tools* dari Google yaitu Google Colaboratory.
- Pembuatan Basestation

  Semua aliran data yang akan dikirim atau diterima oleh robot harus melalui sebuah *basestation*. *Basestation* berfungsi sebagai wadah komunikasi antar robot dan meneruskan perintah yang diterima menuju robot. API *openFrameworks* digunakan untuk membuat *graphical user interface* (GUI) pada *basestation* yang akan digunakan.
- Pengujian dan evaluasi

  Pengujian akan dilakukan dengan data test yang telah disiapkan. Apabila terdapat kesalahan dari prediksi atau akurasi yang terlalu rendah, maka perlu dilakukan perbaikan pada sistem yang telah dibuat atau menambah jumlah data yang digunakan untuk training agar mendapatkan hasil yang akurat dan relevan sehingga dapat digunakan dengan baik.
- Penyusunan laporan

  Penyusunan laporan dilakukan seiring dengan tahapan lainnya. Laporan akan dinyatakan selesai jika sistem yang dibuat ini memiliki akurasi yang cukup baik atau mendapatkan analisa kesimpulan dari kemampuan sistem yang telah dibuat.

1.6 Sistematika Penulisan

Dalam buku penelitian tugas akhir ini, pembahasan mengenai sistem yang dibuat dibagi menjadi lima bab dengan sistematika penulisan sebagai berikut:

- Bab I: Pendahuluan
  Bab ini berisi penjelasan latar belakang, perumusan masalah, batasan masalah, tujuan, metodologi penelitian, sistematika penulisan, serta relevansi pada penelitian tugas akhir ini.
- Bab II: Tinjauan Pustaka
  Bab ini menjelaskan teori penunjang dan literatur yang dibutuhkan dalam pengerjaan tugas akhir. Teori yang menunjang penelitian tugas akhir ini antara lain *Mel Frequency Cepstral Coefficients*, Mikrofon, dan jaringan syaraf tiruan.
- Bab III: Perancangan Sistem
  Bab ini menjelaskan tentang proses penyampaian perintah suara dari pelatih menuju robot dan perancangan sistem pengenalan suara yang digunakan untuk mengenali suara dan perintah dari pelatihnya.
- Bab IV: Pengujian
  Bab ini berisi hasil pengujian pengenalan perintah dan pengenalan suara pelatih serta analisa data atas hasil pengujian yang dilakukan.
- Bab V: Penutup
  Bab ini merupakan bagian akhir dari buku laporan tugas akhir yang berisi kesimpulan dari tugas akhir yang dikerjakan. Bagian ini juga berisi saran pengembangan penelitian yang lebih lanjut.

1.7 Relevansi

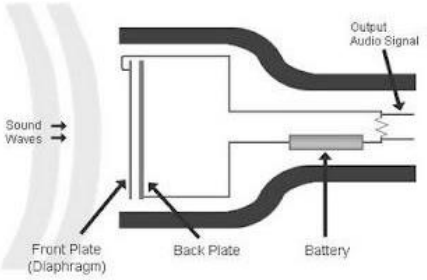
Hasil dari sistem pengenalan suara pada Robot KRSBI Beroda ini diharapkan dapat membantu meningkatkan kemampuan robot ketika bertanding. Metode penegenalan suara ini dapat digunakan untuk mengurangi *error* dan kesalahan penerimaan perintah yang diterima oleh robot, sehingga nantinya robot KRSBI Beroda dapat bermain dengan lebih baik.

.....Halaman ini sengaja dikosongkan.....

BAB 2
TINJAUAN PUSTAKA

2.1 Mikrofon

Mikrofon adalah sebuah tranduser pada alat elektronik yang berfungsi untuk menangkap getaran suara. Mikrofon merupakan tranduser yang dapat menangkap sinyal suara kemudian mengubahnya menjadi sinyal tegangan atau arus yang proporsional terhadap sinyal suara [3]. Mikrofon memberikan output sinyal analog yang sebanding dengan perubahan tekanan akustik bergantung pada fleksibitas diafragma [3]. Sinyal listrik kemudian digunakan untuk pengiriman, perekaman atau pengukuran pada karakteristik sinyal akustik. Penggunaan yang paling umum adalah pada audio broadcasting, perekaman, dan reproduksi, dimana frekuensinya berada pada *range* pendengaran manusia yaitu 20Hz-20KHz [3].



Gambar 2.1 Konstruksi Mikrofon [3]

Setiap jenis mikrofon memiliki cara yang berbeda dalam mengubah bentuk energinya, tetapi mereka semua memiliki persamaan yaitu semua jenis mikrofon memiliki suatu bagian utama yang disebut dengan diafragma. Berikut adalah beberapa jenis mikrofon:

- Mikrofon karbon adalah mikrofon yang terbuat dari sebuah diagram logam yang terletak pada salah satu ujung kotak logam yang berbentuk silinder [4]. Mikrofon ini bekerja berdasarkan resistansi, dimana terdapat sebuah penghubung antara diafragma dengan butir-butir karbon di dalam mikrofon. Perubahan getaran suara akan

menyebabkan nilai resistansi berubah sehingga membuat sinyal keluaran mikrofon juga ikut berubah [4].

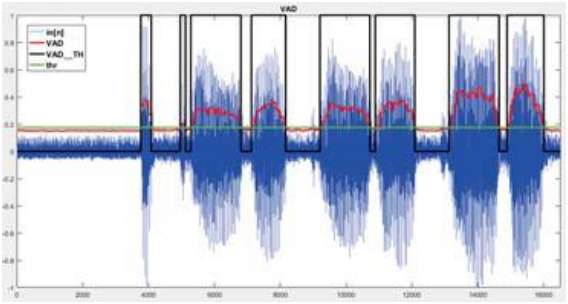
- Mikrofon reluktansi variabel adalah sebuah mikrofon yang terbuat dari diafragma berbahan magnetik. Jika tekanan udara dalam diafragma meningkat karena adanya getaran suara, maka celah udara dalam rangkaian magnetik tersebut akan berkurang, akibatnya reluktansi semakin berkurang dan menimbulkan perubahan-perubahan magnetik yang terpusat di dalam struktur magnetik [4]. Perubahan magnetic tersebut menyebabkan sinyal keluaran mikrofon juga ikut berubah.
- Mikrofon kumparan adalah mikrofon yang terbuat dari kumparan induksi yang digulungkan pada silinder non-magnetik dan dilekatkan pada diafragma. Jika diafragma bergerak karena adanya gelombang suara yang ditangkap, maka kumparan akan bergerak maju mundur di dalam medan magnet, sehingga muncullah perubahan magnetik yang melewati kumparan dan menghasilkan sinyal listrik [4].
- Mikrofon kapasitor adalah mikrofon yang terbuat dari diafragma berbahan logam, digantungkan pada sebuah pelat logam statis dengan jarak sangat dekat, sehingga keduanya terisolasi dan menyerupai bentuk sebuah kapasitor. Adanya getaran suara mengakibatkan diafragma bergerak-gerak. Diafragma yang bergerak menimbulkan adanya perubahan jarak pemisah antara diafragma dengan pelat statis sehingga mengakibatkan berubahnya nilai kapasitansi [4].
- Mikrofon Elektret adalah jenis mikrofon kapasitor yang telah memiliki sumber muatan tersendiri sehingga tidak membutuhkan pencatu daya dari luar. Sumber muatan berasal dari suatu alat penyimpan muatan yang terbuat dari bahan teflon. Bahan teflon tersebut diproses sedemikian rupa sehingga mampu menangkap muatan-muatan tetap dalam jumlah besar, kemudian mempertahankannya untuk waktu yang tak terbatas. Lapisan tipis teflon dilekatkan pada pelat logam statis dan mengandung muatan-muatan negatif dalam jumlah besar. Muatan-muatan tersebut terperangkap pada satu sisi yang kemudian menimbulkan medan listrik pada celah yang berbentuk kapasitor [4]. Getaran suara yang ada mengubah tekanan udara di dalamnya sehingga membuat jarak antara diafragma dan pelat logam statis juga berubah-ubah.

Akibatnya, nilai kapasitansi berubah dan tegangan terminal mikrofon pun juga berubah [4].

- Mikrofon Piezoelektris adalah mikrofon yang terbuat dari bahan kristal aktif . Bahan ini dapat menimbulkan tegangan sendiri saat menangkap adanya getaran dari luar jadi tidak membutuhkan pencatu daya [4]. Kristal akan berubah bentuk bila mendapatkan suatu tekanan sehingga akan terjadi perpindahan muatan sesaat di dalam susunan Kristal tersebut. Perpindahan muatan mengakibatkan adanya perbedaan potensial diantara kedua pelat-pelat lempengan. Kristal tersebut dapat langsung menerima getaran suara, sehingga respon frekuensi yang diterima akan lebih baik dari mikrofon lainnya [4].

2.2 *Voice Activity Detector (VAD)*

Voice activity detector merupakan sebuah metode yang dapat digunakan untuk membedakan sinyal suara dan sinyal *noise*. VAD yang baik adalah VAD yang mampu mendeteksi keberadaan suara manusia pada sebuah sinyal yang memiliki latar belakang *noise* yang cukup kuat [5]. Suatu VAD dapat dikategorikan menjadi tiga jenis yaitu *supervised learning*, *semi-supervised learning*, dan *unsupervide learning* [5]. Sistem VAD dengan sistem learning merupakan sistem VAD yang lebih efektif untuk mendeteksi suara manusia dengan latar belakang *noise* yang kuta. Selain itu juga terdapat metode VAD tanpa menggunakan *learning*, yaitu menggunakan nilai ambang batas rata-rata dari amplitudo sinyal yang tertangkap. Jadi suara yang akan direkam merupakan suara yang dapat melewati nilai ambang batas yang ditentukan seperti pada Gambar 2.2.



Gambar 2.2 VAD dengan Nilai Ambang Batas [6]

2.3 *Mel Frequency Cepstral Coefficients* (MFCC)

Mel Frequency Cepstral Coefficients telah banyak digunakan dalam *voice recognition*. MFCC dapat digunakan sebagai ekstraksi fitur yang baik untuk mewakili ucapan manusia atau sinyal music dan yang paling penting adalah MFCC telah terbukti bermanfaat untuk pengenalan suara. Perhitungan yang dilakukan dalam MFCC menggunakan dasar dari perhitungan *short-term analysis* [7]. MFCC sebenarnrnya merupakan adaptasi dari sistem pendengaran manusia dimana untuk mendapatkan fitur dari sebuah sinyal suara, sinyal suara akan difilter secara linier untuk frekuensi rendah dan secara logaritmik untuk frekuensi tinggi. Berikut adalah gambaran secara umum proses ekstraksi fitur dari MFCC:

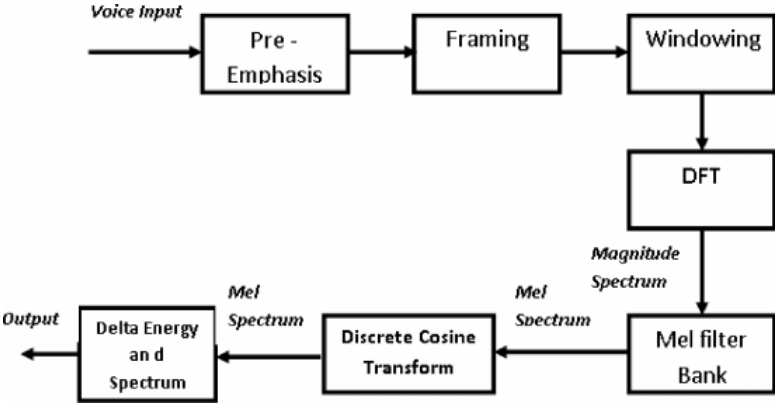
- Melakukan *framing* sinyal suara menjadi beberapa *frame* sinyal yang lebih pendek.
- Setiap *frame* tersebut, dimasukan dalam fungsi FFT.
- Aplikasikan *mel-filterbank* pada spektrum daya setiap frame yang didapat, kemudian jumlahkan energi dari setiap filter tersebut.
- Ambil logaritma dari semua energi *filterbank*.
- Ambil koefisien DCT dari energy *log-filterbank*
- Simpan koefisien DCT 2-13, abaikan koefisien yang lain.

Mel scale dapat menghubungkan antara frekuensi suara yang dirasakan dengan frekuensi asli dari suara yang diukur. Seorang manusia jauh lebih baik dalam hal membedakan suatu suara atau nada pada frekuensi rendah daripada frekuensi tinggi [8]. Dengan menngunakan skala ini membuat fitur MFCC mendekati dengan apa yang manusia dengar. Untuk mengubah frekuensi menjadi *Mel scale* dapat dirumuskan sebagai berikut:

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

Sedangkan untuk mengembalikan *Mel scale* menjadi frekuensi dapat dirumuskan sebagai berikut:

$$M^{-1}(m) = 700(\exp\left(\frac{m}{1125}\right) - 1) \tag{2.2}$$



Gambar 2.3 Proses perhitungan MFCC [8]

2.3.1 *Pre-emphasis*

Proses *Pre-emphasis* adalah proses memindahkan sinyal yang masuk menuju sebuah filter untuk menekan atau menguatkan frekuensi tinggi pada sinyal. Proses ini akan meningkatkan energi dari sinyal pada frekuensi tinggi [8]. *Pre-emphasis* bermanfaat dalam beberapa hal:

- Menyeimbangkan spektrum frekuensi karena pada frekuensi tinggi biasanya memiliki magnitude yang lebih kecil.
- Menghindari masalah numerik ketika melakukan operasi transformasi *fourier*.
- Meningkatkan rasio *noise* pada sinyal [7]

Filter *pre-emphasis* dapat diterapkan pada sebuah sinyal dengan filter orde satu dengan persamaan seabagai berikut:

$$y(t) = x(t) - \alpha x(t-1) \tag{2.3}$$

(α) adalah koefisien dari filter, nilai yang sering digunakan untuk koefisien tersebut adalah 0.95 atau 0.97.

2.3.2 *Framing*

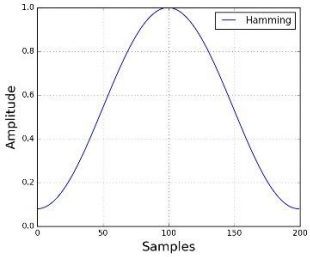
Proses segmentasi sinyal diperoleh dari konversi data analog menjadi digital (ADC) [8]. Kemudian sinyal digital tersebut dipecah menjadi beberapa *frame* kecil dengan Panjang 20 hingga 40 ms [9]. Alasan dilakukanya *framing* adalah karena frekuensi sinyal berubah setiap waktu, sehingga dalam banyak kasus kurang efektif bila melakukan transformasi fourier pada seluruh sinyal karena akan menghilangkan kontur frekuensi sinyal pada setiap waktunya. Untuk menghindarinya, dapat diasumsikan bahwa frekuensi suatu sinyal akan tetap pada periode waktu yang sangat singkat. Oleh karena itu, dengan melakukan transformasi fourier pada *frame* waktu yang singkat, maka akan diperoleh perkiraan kontur frekuensi sinyal yang lebih baik dengan menggabungkan *frame* yang berdekatan.

2.3.3 *Hamming Window*

Hamming window digunakan untuk memetakan sinyal pada setiap *frame* yang diperoleh dengan mempertimbangkan blok berikutnya pada proses ekstraksi fitur dan mengintegrasikan semua frekuensi terdekat [8]. Persamaan *Hamming window* adalah sebagai berikut:

$$w[n] = 0.54 - 0.46cos(\frac{2\pi n}{N-1}) \quad (2.4)$$

$0 \leq n \leq N-1$, diaman N adalah Panjang *window*. Dengan melakukan plotting pada persamaan diatas, maka akan diperoleh sebuah grafik yang dapat dilihat pada Gambar 2.4.



Gambar 2.4 *Hamming window*

Terdapat beberapa alasan dilakukanya Hamming window pada setiap *frame*, alasan utamanya yaitu utuk menangkal asusmsi yang dibuat oleh *fast fourier transform* (FFT) bahwa data tidak terbatas dan untuk mengurangi nilai kebocoran spektral [8].

2.3.4 *Fast Fourier Transform* (FFT)

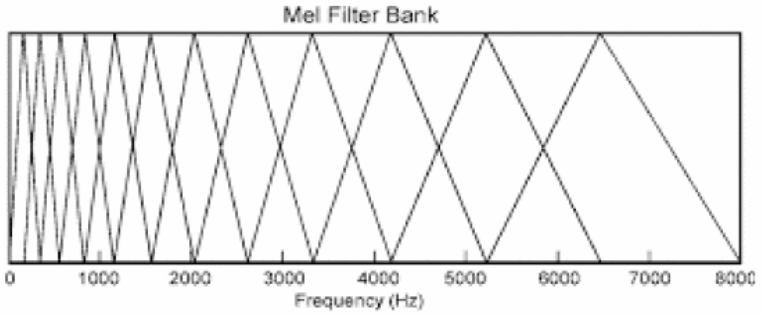
Fast fourier transform berfungsi untuk mengkonversi setiap *frame* dari domain waktu ke domain frekuensi pada seluruh sampel yang didapat. Persamaan dari FFT adalah sebagai berikut:

$$Y(w) = FFT[h(t) * x(t)] = H(w).X(w) \quad (2.5)$$

Y (w), H (w) dan X (w) adalah transformasi fourier dari y(t), h(t) dan x(t).

2.3.5 *Mel Filter Bank*

Rentang frekuensi pada spektrum FFT sangat luas dan sinyal suara tidak dapat mengikuti skala liniernya, sehingga dibutuhkan suatu filter seperti pada Gambar 2.5.



Gambar 2.5 *Mel filter bank* [8]

Pada Gambar 2.5 menunjukkan filter segitiga yang digunakan untuk menghitung jumlah komponen spektral filter, sehingga output proses dapat mendekati skala Mel. Setiap respon frekuensi magnitudo filter berbentuk segitiga dan sama pada frekuensi tengah dan berkurang secara linear menjadi nol pada dua frekuensi tengah filter yang berdekatan [8]. Kemudian, setiap output filter adalah jumlah dari komponen spektral

yang telah difilter. Sedangkan untuk menghitung skala *Mel* dari frekuensi (Hz) yang diberikan adalah dengan mennggunakan persamaan berikut:

$$F(Mel) = [2595.log10(1 + \frac{f}{700})] \quad (2.6)$$

Mel dilter bank pada Gambar 2.4 dapat dimodelkan dengan persamaan sebagai berikut:

$$Hm(k) = \begin{cases} 0 & , k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & , f(m-1) \leq k < f(m) \\ 1 & , k = f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & , f(m) < k \leq f(m+1) \\ x & , x \geq 0 \end{cases} \quad (2.7)$$

2.3.6 *Discrete Cosine Transform* (DCT)

DCT berfungsi untuk mengkonversi spektrum log *mel* kembali dalam domain waktu. Representasi *cepstral* dari spektrum suara memberikan representasi yang bagus pada sifat spektral lokal sinyal yang dihasilkan dari analisa *frame* [8]. Karena koefisien spektrum mel adalah bilangan real, maka kita dapat mengkonversinya ke dalam domain waktu dengan *discrete cosine transform* (DCT). DCT dapat mengkonverikan log mel kembali ke waktu. Hasil dari perhitungan DCT akan menghasilkan koefisien frekuensi spektrum *mel* (MFCC). Set dari koefisien yang didapatkan dapat disebut vektor akustik [8].Berikut adalah persamaan dari *discrete cosine transform*:

$$Cn = \sum_{k=1}^{k} (\log S_k) \cos(\mathrm{n}.(\mathrm{k} - \tfrac{1}{2}).\frac{\pi}{k} \quad (2.8)$$

n = 1,2,…..k, dimana S_k,k = 1,2….k adalah output dari langkah terakhir.

2.3.7 Delta Energi dan Delta Spektrum

Energi merupakan sesuatu yang berhubungan dengan identitas suara dan ini bermanfaat sebagai isyarat deteksi suara [8]. Energi pada

sebuah *frame* untuk sebuah sinyal x didalam *window* untuk rentang waktu t1 dan t2 dapat direpresentasikan dengan persamaan berikut:

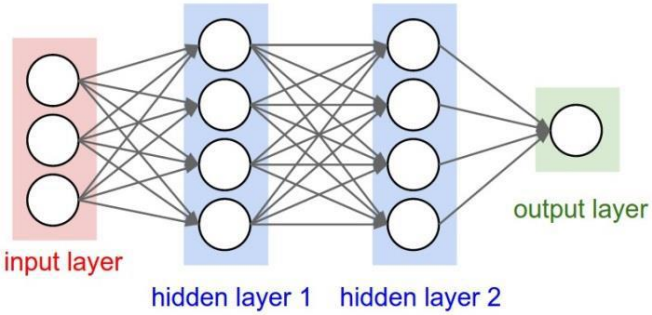
$$Energy = \sum_{t=t1}^{t2} \mathrm{x}^2(t) \quad (2.9)$$

Selain itu, sinyal suara juga tidak konstan pada setiap *frame*. Ini merupakan fakta penting tentang sinyal suara pada perubahan *frame*. Untuk alasan ini dibutuhkan penambahan fitur yang berhubungan dengan pada perubahan fitur *cepstral* pada setiap waktu. Fitur yang dapat ditambahkan yaitu delta dan *double* delta [8]. Setiap fitur delta merepresentasikan perubahan diantara *frame* didalam fitur energi atau *cepstral* yang sesuai. Cara termudah untuk menghitung delta adalah dengan menghitung perbedaan diantara *frame*. Dengan demikian nilai delta dapat dihitung menggunakan persaaman berikut:

$$d(t) = \frac{c(t+1)-c(t-1)}{2} \quad (2.10)$$

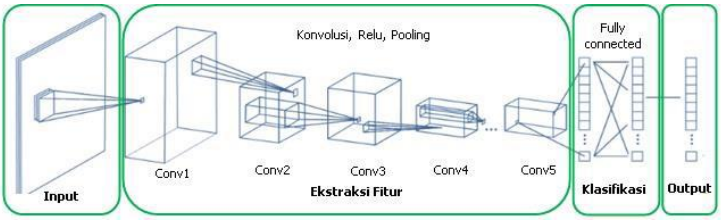
2.4 *Convolutional Neural Network* (CNN)

Convolutional neural network dapat disebut sebagai kelanjutan atau pengembangan dari model jaringan saraf tiruan tradisional. *Convolutional Neural Network* merupakan salah satu metode *machine learning* dari pengembangan *Multi Layer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam *deep learning* karena kedalaman jaringan atau *neuron* yang tinggi [10]. *Deep learning* adalah cabang dari *machine learning* yang dapat mengajarkan komputer untuk melakukan pekerjaan selayaknya manusia, seperti komputer dapat belajar dari proses *training* [11]. CNN merupakan operasi konvolusi yang menggabungkan beberapa lapisan pemrosesan, menggunakan beberapa elemen yang beroperasi secara paralel dan terinspirasi oleh sistem saraf manusia. CNN memiliki dua metode, klasifikasi menggunakan *feedforward* dan tahap pembelajaran menggunakan *backpropagation*. Cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap jaringan dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap jaringan hanya berukuran satu dimensi [12].



Gambar 2.6 Arsitektur MLP [10]

MLP menerima input data satu dimensi dan mempropagasikan data tersebut pada jaringan hingga menghasilkan output. Setiap hubungan antar jaringan pada dua *layer* yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode. Sedangkan pada CNN, data yang dipropagasikan pada jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda [10]. Pada CNN operasi linear menggunakan operasi konvolusi dan nilai bobot tidak lagi satu dimensi saja.

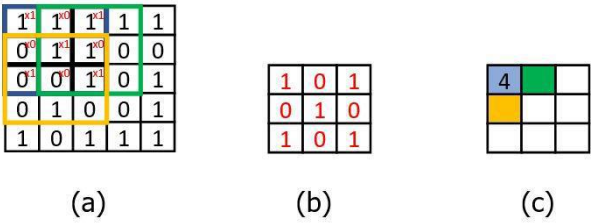


Gambar 2.7 Arsitektur CNN [11]

CNN memanfaatkan proses konvolusi. Dengan menggerakan sebuah kernel konvolusi berukuran tertentu ke sebuah gambar, komputer mendapatkan informasi representatif baru dari hasil perkalian bagian gambar tersebut dengan filter yang digunakan. Karena sifat proses konvolusi, maka CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara [10].

2.4.1 *Convolutional Layer*

Convolution Layer melakukan operasi konvolusi pada data yang dimasukan. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada output fungsi lain secara berulang. Dalam pengolahan citra, konvolusi berati mengaplikasikan sebuah filter pada data. Filter akan bergerak pada seluruh bagian data. Tujuan dilakukannya konvolusi pada data adalah untuk mengekstraksi fitur dari data input [10]. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada *layer* tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN [10].



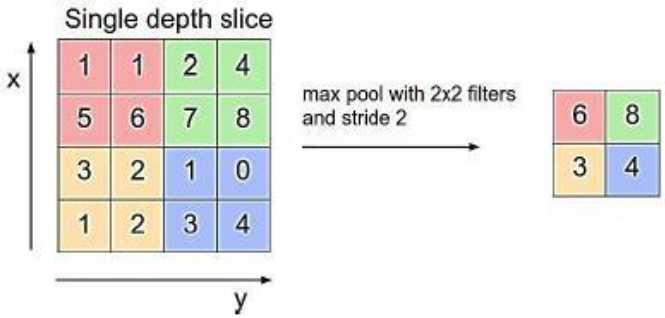
Gambar 2.8 Proses konvolusi [11]

Terdapat parameter yang dapat diubah untuk memodifikasi sifat tiap lapisan, yaitu ukuran filter, *stride* dan *padding*. *Stride* mengontrol bagaimana filter diterapkan pada data input dengan bergerak sepanjang ukuran data yang telah ditentukan [11]. *Padding* adalah penambahan ukuran data dengan nilai tertentu disekitar data input agar hasil dari bidang *receptive* tidak terlalu kecil sehingga tidak banyak informasi yang hilang [11]. Nilai *padding* ini biasanya nol sehingga disebut dengan *zero padding*. Output dari proses konvolusi ini dijadikan sebagai input untuk lapisan konvolusi selanjutnya.

2.4.2 *Pooling Layer*

Pooling atau *subsampling* merupakan metode melakukan pengurangan ukuran matriks. *Pooling* juga bertujuan untuk meningkatkan invariansi posisi dari fitur [10]. Terdapat dua macam *pooling* yang sering

digunakan yaitu *average pooling* dan *max pooling*. Nilai yang diambil pada *average pooling* adalah nila rata-rata sedangkan pada *max pooling* adalah nilai maksimal [11]. Dalam CNN, metode subsampling yang paling sering digunakan adalah *max pooling*.



Gambar 2.9 Operasi *Max pooling* [10]

Max pooling membagi output dari layer konvolusi menjadi beberapa kotak kecil lalu mengambil nilai maksimal dari setiap kotak untuk menyusun matriks data baru yang telah direduksi [10]. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek data mengalami pergeseran.

2.4.3 *Fully Connected Layer*

Fully connected layer merupakan kumpulan dari proses konvolusi yaitu merubah data hasil konvolusi menjadi satu dimensi. Lapisan ini mendapatkan input dari proses sebelumnya untuk menentukan fitur mana yang paling berkorelasi dengan kelas tertentu. Fungsi dari lapisan ini adalah untuk menyatukan semua *node* menjadi satu dimensi [11]. Setiap *neuron* pada layer konvolusi perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah *fully connected layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversible [10]. *Fully connected layer* hanya dapat diimplementasikan di akhir proses konvolusi.

2.4.4 Fungsi Aktivasi

Didalam proses CNN tidak akan lepas dari fungsi aktivasi. Fungsi aktivasi adalah fungsi non-linear yang memungkinkan sebuah jaringan saraf tiruan untuk dapat mentransformasi data input menjadi data dengan ciri tertentu sehingga memungkinkan dilakukan klasifikasi pada data tersebut [10]. Fungsi yang paling sering digunakan pada jaringan saraf tiruan yaitu relu, softmax, dan sigmoid.

2.4.4.1 *ReLU*

ReLU (*Rectification Linear Unit*) merupakan operasi untuk mengenalkan nonlinearitas dan meningkatkan representasi dari model. Nilai output dari *neuron* bisa dinyatakan sebagai 0 jika inputnya adalah negatif. Jika nilai input adalah positif, maka output dari neuron adalah nilai input aktivasi itu sendiri [11].Berikut adalah persamaan dari fungsi aktivasi ReLU:

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{2.11}$$

2.4.4.2 *Softmax*

Fungsi aktivasi softmax digunakan untuk mendapatkan hasil klasifikasi. Fungsi aktivasi menghasilkan nilai yang diinterpretasi sebagai probabilitas yang belum dinormalisasi untuk tiap kelas. Nilai kelas dihitung dengan menggunakan fungsi *softmax* [11]. Berikut adalah persamaan dari fungsi aktivasi *softmax*:

$$S(q_i) = \frac{exp(q_i)}{\sum_{j=1}^{n} exp(q_j)} \tag{2.12}$$

2.4.4.3 *Sigmoid*

Sigmoid biasanya digunakan untuk melakukan klasifikasi data yang memiliki lebih dari satu label atau yang bisa disebut dengan multilabel. Fungsi *sigmoid* mentransformasi range nilai dari input x menjadi antara 0 dan 1 dengan bentuk distribusi yang sesuai dengan persamaan berikut:

$$f(x) = \frac{1}{(1+e^{-x})} \tag{2.13}$$

Fungsi *sigmoid* sekarang sudah tidak banyak digunakan dalam praktek karena memiliki kelemahan utama yaitu range nilai output dari fungsi sigmoid tidak terpusat pada angka nol [10]. Hal tersebut menyebabkan terjadinya proses *backpropagation* yang tidak ideal, selain itu bobot tidak terdistribusi rata antara nilai positif dan negatif serta nilai bobot akan banyak mendekati ekstrim 0 atau 1 dan dapat menyebabkan efek saturasi dimana jika nilai bobot cukup kecil, maka lama kelamaan nilai bobot akan mendekati nol [10]. Jika hal tersebut terjadi, maka jaringan tersebut tidak akan dapat mengalami update yang signifikan dan akan nonaktif.

2.5 Keras

Keras merupakan *open-source library* yang banyak digunakan dalam jaringan saraf tiruan tingkat tinggi [13], Keras menyediakan API untuk pemrograman jaringan saraf tiruan berbasis bahasa python. Model dari *machine lerarning*, jaringan saraf tiruan, dan *deep learning* dapat dibuat menggunakan Keras [13]. Keras mudah untuk dipahami dan dibangun karena kode programnya yang dipecah menjadi beberapa bagian. Bagian untuk menghasilkan model normalnya terdiri dari *neural layers*, *cost functions*, *optimizer*, dan fungsi aktivasi. Untuk memebuat fungsi atau kelas yang baru dapat dengan mudah dikembangkan menggunakan python.

Gambar 2.10 Logo Keras

Keras mendukung berbagai model deep learning seperti Convolutional Neural Network (CNN, ConvNet), Recurrent Neural Network, maupun model Multilayer Perceptron yang lebih sederhana [13]. Untuk masalah pelatihannya juga sudah menggunakan algoritma-algoritma

pengembangan SGD seperti RMSPROP, Adadelta, dan juga Adam. Keras juga mendukung penggunaan GPU untuk komputasi, sehingga pelatihan dapat dilakukan dengan lebih cepat.

2.6 Google Colaboratory

Google Colaboratory atau biasa disebut dengan Colab adalah sebuah proyek yang memiliki tujuan untuk menyebarluaskan edukasi dan penelitian dari *machine learning* [14]. *Tools* ini secara penggunaan mirip seperti Jupyter Notebook dan dibuat diatas *envirounment* Jupyter yang tidak memerlukan pengaturan terlebih dahulu sebelum digunakan dan berjalan sepenuhnya pada Cloud dengan memanfaatkan media penyimpanan Google Drive.

Gambar 2.11 Logo Google Colaboratory

Colbaoratory menyediakan interpreter untuk python 2 dan 3 yang telah di konfigurasi dengan *library machine learning* dan *artificial intelligence*, seperti Tensorflow, Matplotlib, dan Keras [14]. Mesin virtual yang disediakan oleh Colab akan dinonaktifkan dan semua data pengguna dan konfigurasinya akan hilang, apabila tidak ada aktifitas setelah beberpa waktu. Namun, *notebook* tetap dipertahankan dan juga memungkinkan untuk memindah *file* dari *hard disk* mesin virtual menuju akun Google Drive pengguna [14]. Colab juga menyediakan layanan GPU untuk mempercepat proses menjalankan program.

2.7 OpenFrameworks

OpenFrameworks adalah sebuah *open source library* untuk pemrograman kreatif yang diciptakan oleh Zachary Lieberman, Theo Watson dan Arturo Castro [15]. OpenFrameworks menyediakan fungsionalitas grafik dan suara tanpa perlu pemrograman yang eksplisit [15]. OpenFrameworks ditulis dalam bahasa pemrograman c++ dan dibangun diatas *platform* OpenGL, dapat dijalankan pada Microsoft Windows, macOS, Linux, iOS, Android dan Emscripten.

Gambar 2.12 Logo OpenFrameworks

2.8 Protokol UDP (*User Datagram Protocol*)

UDP merupakan salah satu jenis protokol jaringan komputer. Melalui UDP, sebuah komputer dimungkinkan untuk mengirim pesan kepada komputer lain pada sebuah jaringan tanpa perlu melalui proses komunikasi awal. Protokol UDP adalah protokol yang bersifat *connectionless* dalam mentransmisi data dan tidak mengenal dalam pengecekan terhadap *error* pengiriman data [16]. Karakteristik UDP lainnya adalah *unreliable*, artinya semua pesan yang dikirimkan tidak memiliki nomor urut [16]. Jika selama transmisi ada pesan-pesan yang hilang, maka protokol aplikasi yang letaknya di atas UDP harus memulihkan pesan tersebut. Protokol UDP pada dasarnya hanya mengandung IP (*internet protocol*) dengan tambahan *header* singkat dan

tidak melakukan sebuah proses kontrol alur data, baik itu kontrol kesalahan ataupun pengiriman ulang terhadap kesalahan [16]. Berikut adalah karakteristik dari protokol UDP:

- Efektif untuk mengirimkan data yang mana lebih mengutamakan kecepatan.
- Protokol jaringan yang ringan karena tidak memakan *resources* yang terlalu besar.
- *End-to-end*, yaitu UDP dapat mengidentifikasi proses yang berjalan dalam komputer [16].
- Dapat digunakan untuk mengirim transmisi data secara *broadcast*.
- *Message-oriented*, yaitu mengirimkan dan menerima data secara segmen [16].
- *Arbitrary interaction*, yaitu UDP dapat menerima dan mengirim dari banyak proses [16].

Tabel 2.1 *Format Header* Protokol UDP

Bit	8	16	24	32
0	*Port* sumber		*Port* tujuan	
32	Panjang data		*Header* dan *Cheksum*	
	Data			

Format Header protokol UDP memiliki ukuran byte tertentu yang menggandung informasi pengirim dan tujuanya, berikut adalah penjelasanya:

1) *Port* sumber memiliki panjang data 16 bit. Jika *host* sumber sebagai *client*, kebanyakan nomor *port* ditentukan oleh *software* UDP yang berjalan di *host* sumber, namun jika *host* sumber sebagai *server*, nomor *port* menggunakan nomor *port* yang umum digunakan [17].
2) *Port* tujuan memiliki panjang 16 bit. Jika *host* tujuan adalah *server*, biasanya nomor *port* adalah yang biasa digunakan, jika *host* tujuan adalah *client*, nomor *port* adalah nomor yang salin dari nomor *port* sementara yang diterima pada paket [17].

3) Panjang data memiliki panjang 16 bit, untuk menyatakan panjang seluruh data yang dikirimkan termasuk *header* UDP.
4) Checksum dengan Panjang 16 bit digunakan untuk mengetahui adanya *error* pada proses transfer data.

2.9 Robot Operating System (ROS)

ROS adalah sebuah *open source framework* yang digunakan untuk mengoperasikan robot dan berbasis pada sistem operasi Linux [18]. ROS menyediakan *library* dan *tools* untuk membuat perangkat lunak pada suatu robot menggunakan bahasa pemrograman C++ dan Python. ROS merupakan sebuah *middleware* robot yang dapat menghubungkan perangkat keras robot dengan sistem operasi komputer secara fleksibel [19]. ROS memiliki 3 konsep yaitu *level filesystem*, *level* grafik komputasi, dan *level* komunitas [19]. *Level filesystem* berisi sumber daya ROS yang ada pada sistem. Tahap awal saat menggunakan ROS adalah dengan membuat konsep seperti sebuah sistem operasi. Pada sebuah *filesystem* ROS terdapat *folder*, diaman masing-masing *folder* mempunyai deskripsi file sesuai fungsinya masing-masing. *Level filesystem* ROS meliputi *packages*, *metapackages*, *packages manifests*, *metapackage manifests*, *message* dan *service* [19]. *Level* grafik komputasi adalah menggunakan jaringan komunikasi *peer-to-peer* pada pengolahan data bersama-sama dan saling terkoneksi ketika semua proses saling terhubung. Setiap node dalam sistem akan saling terhubung, melihat informasi dan berinteraksi dengan node lainnya. Grafik komputasi ROS terdiri dari *level nodes*, *master*, *parameter server*, *message*, topik, *services*, dan *bags* [19]. *Level* komunitas ROS berupa suatu komunitas yang dapat membagi sumber daya ROS secara terpisah agar dapat saling bertukar pengetahuan dan perangkat lunak yang telah dibuat. Sumber daya tersebut meliputi distribusi ROS, *repository*, ROS wiki, *bug ticket system*, *mailing list*, ROS *answer* dan *blog* [19]. ROS menggunakan sistem komunikasi *peer to peer*, yaitu setiap *node* program yang dapat dieksekusi akan saling berkomunikasi satu sama lain ketika program dijalankan, setiap *node* tersebut harus terdaftar pada master ROS agar *node* program dapat mengetahui keberadaan satu sama lain [18].

Gambar 2.13 Logo ROS

Penggunan ROS dapat memudahkan pengembang suatu robot dalam membuat perangkat lunaknya tanpa harus membuat program atau *source code* dari awal. Selain itu, ROS juga bertujuan untuk dapat dikembangkan bersama-sama karena sifatnya yang *open source*. Setiap *node* tidak berkomunikasi secara langsung. Komunikasi yang terjadi pada *node* yaitu berupa *publishing* dan *subscribing* pada topik [18]. Jadi, apabila sebuah *node* membutuhkan suatu data, maka *node* harus melakukan *subscribing* pada topik yang bersangkutan dan *node* yang menghasilkan data juga harus melakukan *publishing* data pada topik tersebut [18]. Dengan adanya sistem yang saling terpisah tersebut, maka sistem tersebut dapat digunakan untuk menjalankan beberapa beberapa proses yang berbeda pada sebuah robot tanpa menghambat proses satu dengan lainya. Bahkan, apabila salah satu sistem dari sebuah robot mengalami kegagalan sistem, maka hal tersebut tidak akan menghentikan seluruh sistem pada robot [19].

2.10 Mini PC Intel NUC NUC6i7KYK

Komputer produk Intel NUC biasanya memiliki prosesor yang hemat daya, sehingga kurang mampu untuk melakukan proses-proses komputasi yang berat. Namun untuk produk NUC6i7KYK, Intel NUC merancang produknya untuk melakukan proses-proses komputasi yang berat. NUC6i7KYK ini dilengkapi oleh prosesor Core i7 6770HQ. Core i7 6770HQ merupakan prosesor *quad core* dengan teknologi *Hyper-Threading*, sehingga mampu menjalankan proses komputasi sekaligus dalam satu kali proses jalan.

Gambar 2.14 Komputer mini NUC6i7KYK

Proses yang digunakan NUC6i7KYK memiliki nilai *clock speed* dasar sebesar 2,7 GHz dan dapat dipercepat menjadi 3,5 GHz. Core i7 6770HQ dilengkapi dengan IGP Iris Pro Graphics. Kelemahan dari NUC6i7KYK ini adalah panas yang dihasilkan cukup tinggi karena kurangnya runag untuk sistem pendinginya dan kecepatan proses komputasinya yang tinggi.

2.11 Mikrokontroler STM32F4 Discovery

STM32F4 Discovery merupakan salah satu mikrokontroler keluaran STMicroelectronics yang menggunakan arsitektur ARM Cortex M4 32 bit yang bekerja pada rentang tegangan 1.8V hingga 3.6V [20]. Beberapa operasi matematika dengan data *float* 32bit dapat dilakukan oleh mikrokontroler ini hanya dengan 1 siklus mesin. Komputasi *float* biasanya dipakai untuk pemrosesan sinyal digital yang memerlukan keakuratan yang tinggi atau bisa juga dipakai untuk pemrosesan data suara atau gambar digital secara lebih mudah tanpa khawatir terjadi *overflow*. Mikrokontroler STM32F4 ini memiliki kecepatan *clock* maksimal hingga 168 MHz. Selain itu STM32F4 juga dilengkapi dengan memori internal sebesar 1024 Kb dan RAM internal 192 Kb.

Gambar 2.15 STM32F4 Discovery *board* [20]

Dalam mikrokontorler ini terdapat tiga ADC (*Analog to Digital Converter*) sebanyak 24 *channels* dan dua DAC (*Digital to Analog Converter*) yang masing-masing memiliki resolusi maksimal 12bit. STM32F4 memiliki 14 *timers*, dimana 12 timer memiliki resolusi 16bit dan 2 timer memiliki resolusi 32bit. Terdapat 4 UART, 3 I^2C, 3 SPI, dan 2 CAN *interfaces* yang dapat digunakan untuk melakukan komunikasi serial. STM32F4 ini juga dilengkapi dengan DMA yang bermanfaat untuk mempercepat proses komunikasi serial UART dan pembacaan data ADC. Pin input dan output pada STM32F4 ini memiliki *clock speed* hingga mencapai 84 Mhz dan memiliki tegangan toleransi hingga 5V.

2.12 Kamera *Omnidirectional*

Kamer *omnidirectional* berguna untuk mengambil gambar pada sekeliling wilayah kamera. Kamera ini dapat menampilkan gambar dari berbagai arah, baik itu arah depan, arah belakang, samping kanan maupun samping kiri [21]. Cara yang sering digunakan dalam pengambilan gambar dengan kamera *omnidirectional* ini adalah dengan memanfaatkan sebuah cermin cembung yang dihadapkan ke bawah dan kamera yang dihadap ke atas, sehingga antara cermin dan kamera saling berhadapan. Jadi gambar yang ditangkap oleh kamera merupakan gambar pada pantulan cermin cembung yang mampu memperlihatkan keadaan atau kondisi disekitarnya.

Gambar 2.16 Kamera *Omnidirectional* [22]

Gambar yang dihasilkan oleh kamera *omnidirectional* memiliki *distorsi* pada segi perseptif sudut pandang, di mana citra yang dihasilkan memiliki pandangan yang, hal ini dapat mempengaruhi proses deteksi suatu objek berdasarkan bentuknya, dengan menggunakan kamera *omnidirectional* objek yang akan terlihat pada pantulan cermin akan memiliki bentuk yang berbeda dengan bentuk aslinya [21]. Namun kamera *omnidirectional* memiliki kelebihan, yaitu dapat menampilkan gambar dari segala arah hanya dengan satu kali pengambilan citra, berbeda dengan kamera mono, untuk dapat menghasilkan citra dari segala arah seperti kamera *omnidirectional* membutuhkan 6 kali pengambilan gambar [21].

2.13 Roda *Omnidirectional*

Roda *omnidirectional* adalah sebuah roda yang didesain agar dapat bergerak bebas pada segala arah. Roda *omnidirectional* ini cukup unik karena memiliki kemampuan bergerak bebas dua arah. Roda ini berputar seperti roda pada umumnya dan mampu bergeser kesamping menggunakan roda di sepanjang lingkraran luar roda [23].

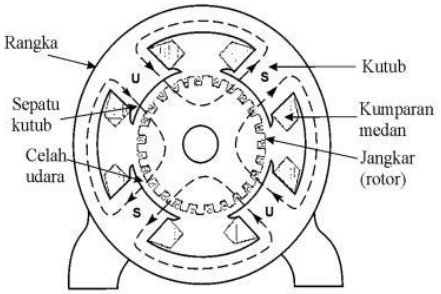
Gambar 2.17 Roda *Omnidirectional* [23]

Dengan menggunakan roda omnidirectional ini memungkinkan sebuah robot untuk memgubah robot *non-holonomic* menjadi robot *holonomic* [23]. Sebuah robot *non-holonomic* yang menggunakan roda normal hanya memiliki 2 DOF (*Degree of Freedom*) yang terkendali, sehingga hanya dapat bergerak maju atau mundur dan rotasi. Robot *non-holonomic* juga tidak memiliki kemampuan untuk bergerak kesamping kiri atau kanan sehingga membuat robot kurang efisien dalam mencapai tempat tujuanya [23]. Dengan menggunakan roda *omnidirectional*, maslah tersebut dapat teratasi karena roda memiliki 3 DOF. Berbeda dengan robot *non-holonomic*, robot dengan roda *omnidirectional* mampu bergerak ke segala arah tanpa mengubah arah roda. Roda omnidirectional dapat bergerak maju mundur, geser ke samping, dan berputar pada posisi tetap. Kemampuan ini memungkinkan robot yang menggunakan roda omnidirectional mampu bermanuver lebih lincah dan efisien [23].

2.14 Motor DC

Motor DC atau motor arus searah adalah mesin penggerak yang merubah energi listrik arus searah menjadi energi mekanis berupa putaran. Motor DC terdiri atas bagian yang diam (*stator*) dan bagian yang berputar (*rotor*). Pada bagian yang diam terdapat kumparan medan yang berfungsi untuk menghasilkan medan magnet sedangkan pada bagian yang berputar terdapat kumparan jangkar, komutator dan sikat. Motor arus searah bekerja berdasarkan prinsip interaksi antara dua medan magnet, dimana kumparan medan akan menghasilkan medan magnet

yang arahnya dari kutub utara menuju kutub selatan dan kumparan jangkar akan menghasilkan medan magnet yang melingkar, sehingga akan menimbulkan suatu gaya [24].



Gambar 2.18 Konstruksi Motor DC [24]

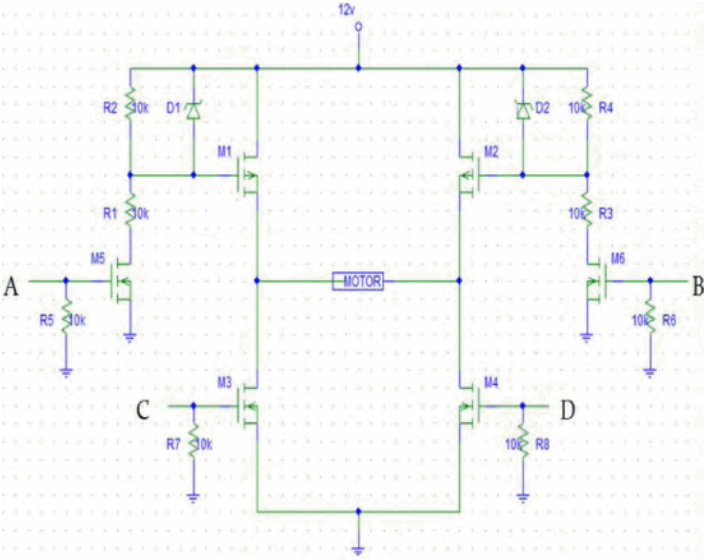
Setiap bagian dari motor DC memiliki peran dan fungsinya masing-masing, berikut adalah penjelasan dari masing-masing bagianya:

- Kutub medan berfungsi untuk menghasilkan medan magnet dari kutub utara menuju kutub selatan
- *Stator* terbuat dari lapis-lapis pelat beralur yang terbuat dari besi atau pelat baja yang di pabrikasi, biasanya tempat melekatnya kutub-kutub magnet.
- *Rotor* merupakan bagian bergerak dari motor DC yang berupa sebuah kumparan. Pada bagian *rotor* ini merupakan tempat melekatnya lilitan kawat yang dapat menghasilkan arus listrik bila dilewati oleh arus.
- Komutator berfungsi untuk membalikan arah arus listrik dalam *stator*. Komutator juga membantu dalam transmisi arus antara stator dan sumber daya [24].

2.15 *Driver* Motor DC

Driver motor DC berfungsi untuk mengatur keceptan putaran dan arah putaran dari motor. Rangkaian yang umum digunakan untuk *driver* motor DC ini adalah rangkaian *H-bridge*. Rangkaian *H-bridge* sendiri

terdiri dari empat transistor yang berguna untuk menentukan arah putaran dari motor.



Gambar 2.19 Rangkaian H-bridge [25]

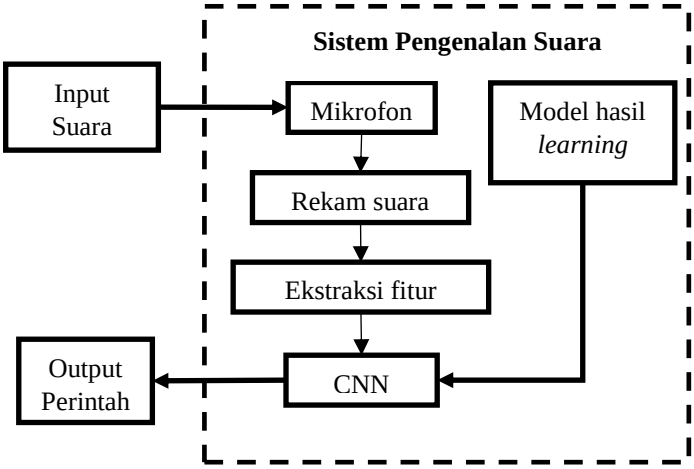
.....Halaman ini sengaja dikosongkan.....

BAB 3
PERANCANGAN SISTEM

Pada bab ini akan membahas tentang setiap komponen yang dirancang dalam sistem pengenalan suara yang dibuat. Bab ini akan membahas sistem *software* yang dirancang meliputi metode yang digunakan untuk menangkap data suara, proses *learning* data suara yang didapat, prediksi menggunakan hasil *learning*, dan komunikasi antara sistem penegenalan suara yang dibuat dengan *basestation* robot. Selain itu pada bab ini juga akan dijelaskan tentang komponen *hardware* dari robot sepak bola beroda meliputi desain mekanik robot sepak bola beroda, sistem elektronik dari robot, sensor dan aktuator pada robot, mikrokontroler dan komputer yang digunakan. Pada penelitian tugas akhir ini lebih difokuskan dalam pembahasan sistem pengenalan perintah suara yang akan digunakan untuk memberikan perintah pada robot sepak bola beroda. Komponen sistem pengenalan suara yang dibuat antara lain yaitu metode *voice activity detector* (VAD), model CNN yang dibangun, dan metode ekstraksi fitur suara yang digunakan. Setiap komponen tersebut akan saling terhubung sehingga dapat menghasilkam sebuah sistem penegenalan suara yang baik.

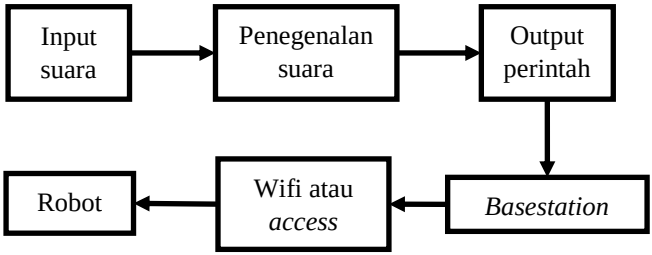
3.1 Perancangan Sistem Pengenalan Suara

Untuk memberikan perintah melalui suara pada robot sepak bola beroda, maka dibutuhkan proses pengenala suara atau *voice recognition* pada robot tersebut. Proses dari penegenalan suara dapat dilihat pada Gambar 3.1. Pertama suara dari manusia akan ditangkap mikrofon, kemudian suara tersebut akan direkam dan disimpan. Selanjutnya rekaman suara suara tersebut diambil fiturnya dengan metode *Mel Frequency Cepstral Coefficients* (MFCC). Data hasil dari ekstraksi fitur tersebut kemudian akan dijadikan input dari *Convolutional Neural Network* (CNN). Hasil atau output dari proses CNN adalah berupa prediksi perintah yang disampaikan melalui suara, nantinya akan disampaikan kepada robot speak bola beroda.



Gambar 3.1 Diagram Blok Sistem Penegenalan Suara

Perintah suara dari manusia atau yang disebut dengan *high level human coaching* hanya dapat dilakukan ketika terjadi bola mati. Jadi ketika permainan dalam kondisi bola hidup, maka tidak boleh ada perintah yang dikirimkan kepada robot. Komunikasi yang dilakukan pada robot harus melalui sebuah *basestation*. Karena setiap komunikasi yang dilakukan oleh robot harus melalui sebuah *basestation*, maka untuk memberikan perintah suara pada robot speak bola beroda juga harus melalui *basestation* terlebih dahulu, setelah diterima oleh *basestation* selanjutnya perintah tersebut baru akan diteruskan kepada robot. Sistem pengiriman perintah suara pada robot sepak bola beroda dapat dilihat pada Gambar 3.2.



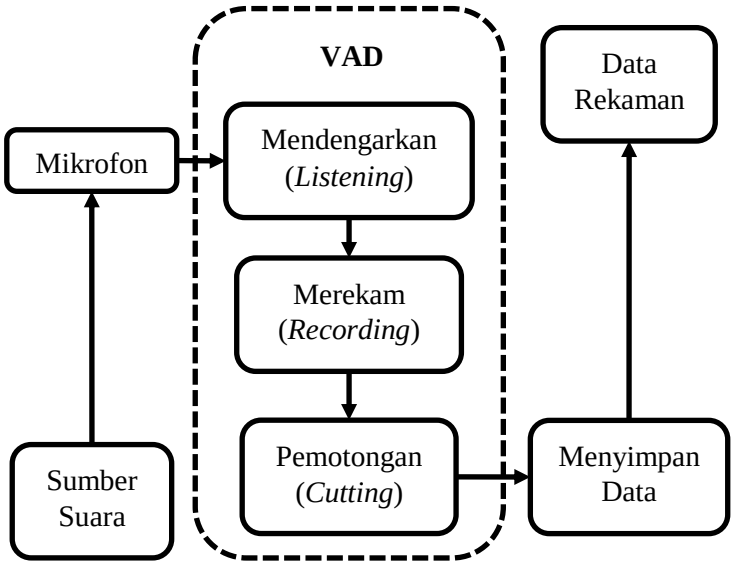
Gambar 3.2 Diagram Blok Perintah Suara Menuju Robot

3.1.1 Pengambilan Data Suara

Dalam proses pengenalan suara pada penelitian tugas akhir ini, langkah awal yang harus dilakukan adalah mengambil atau merekam suara perintah yang diberikan melalui mikrofon. Mikrofon akan berfungsi sebagai sensor yang dapat menangkap gelombang suara disekitarnya. Mirkofon yang digunakan dapat dilihat pada Gambar 3.3. Selain itu hal lain yang dibutuhkan yaitu bagaimana cara mengetahui apakah suara yang masuk adalah suara perintah atau suara *noise*. Untuk itu sistem yang dirancang harus dapat membedakan antara suara perintah dengan suara *noise* disekitarnya. Untuk mengatasi hal tersebut, maka digunakan metode VAD (*Voice Activity Detector*) .

Gambar 3.3 Mikrofon BOYA BY-M1

Dalam *voice activity detector* digunakan metode *threshold* atau ambang batas untuk membedakan suara perintah dan suara *noise*. Jadi suara yang akan direkam adalah suara yang berada diatas nilai ambang batas, sedangkan suara yang berada dibawah nilai ambang batas, maka akan dianggap sebagai suara *noise*. Secara garis besar, terdapat beberapa proses yang dilakukan oleh VAD agar suara yang direkam adalah suara perintah, bukan suara *noise*. Proses pengambilan data atau perekaman suara dengan VAD dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram Blok Proses Merekam Suara

Dalam proses merekam suara, data yang masuk melalui sebuah mikrofon tidak akan langsung direkam dan disimpan, melainkan harus melalui beberapa tahap, yaitu *listening*, *recording*, dan *cutting*. Apabila suara yang masuk telah melalui tigas proses tersebut, maka suara tersebut akan disimpan sebagai data rekaman suara dalam *waveform audio format* (.WAV). Untuk mengakses mikrofon pada sebuah komputer, maka digunakan API yang disediakan oleh pyaudio.

3.1.1.1 Proses Mendengarkan Suara

Tahap awal dari metode *voice activity detector* yaitu proses mendengarkan suara yang masuk atau disebut dengan *listening*. Proses ini adalah proses yang paling penting karena proses ini adalah penentu suara yang masuk akan direkam atau tidak. Jadi menentukan nilai ambang batas atau *threshold* merupakan hal yang sangat penting agar suara yang terekam nantinya adalah suara perintah yang diberikan, bukan suara *noise*. Untuk mengetahui suara yang masuk sudah melebihi nilai ambang

batas atau tidak, maka dibuat sebuah *sliding window* yang berisi nilai akar kuadarat rata-rata setiap paket perekaman diambil atau bisa disebut dengan *chunk*. Penulisan kode programnya adalah sebagai berikut:

```
cur_data = stream.read(CHUNK)
slid_win.append(math.sqrt(abs(audioop.avg(cur_
                data, 4))))
if(sum([x > THRESHOLD for x in slid_win])> 0):
```

Sliding window merupakan sebuah array yang berisi beberapa nilai akar kuadarat rata-rata setiap *chunk*. Apabila terdapat nilai pada *sliding window* yang melebihi nilai ambang batas, maka akan masuk pada proses selanjutnya. Sedangkan data suara asli juga disimpan agar ketika terdapat nilai yang melebihi ambang batas pada *sliding window*, data tersebut tidak segera hilang karena data tersebut merupaka data suara awal dimulainya perekaman suara.. Untuk kode programnya adalah sebagai berikut:

```
prev_audio.append(cur_data)
```

Durasi waktu yang disimpan pada data suara awal adalah 0.25 detik. Ketika durasi 0.25 detik tersebut sudah penuh, maka data tersebut akan selalu diperbarui apabila data suara yang masuk tidak melebihi ambang batas yang ditentukan. Jadi pada proses mendengarkan suara ini, sebenarnya mikrofon sudah aktif untuk menangkap suara yang masuk, namun suara yang masuk tersebut tidak disimpan hingga terdapat nilai yang melebihi ambang batas pada *sliding window*.

3.1.1.2 Proses Merekam Suara

Setelah terdapat suara masuk yang melebihi ambang batas, selanjutnya adalah proses merekam suara. Pada proses merekam suara ini, suara yang masuk akan disimpan sebagai data rekaman. Berikut adalah programnya:

```
if(not started):
   print ("Starting record of phrase")
   started = True
   record.append(cur_data)
```

Setiap suara yang ditangkap oleh mikrofon akan disimpan dalam variabel record. Ketika masuk proses merekam suara ini, *sliding window* tetap aktif untuk mengecek untuk menghitung nilai yang masuk. Apabila semua nilai pada *sliding window* kurang dari niali ambang batas, maka proses merekam suara akan berakhir. Durasi waktu yang dibuat pada *sliding window* adalah 0.8 detik, jadi dapat dikatakan bahwa proses merekam suara akan berhenti apabila selama 0.8 detik tidak ada suara yang melebihi nilai ambang batas.

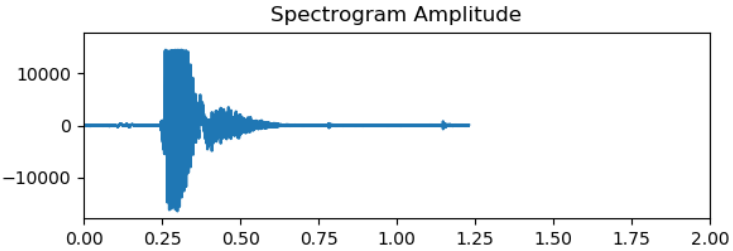
3.1.1.3 Proses Pemotongan Suara

Proses pemotongan suara merupakan proses terkahir dari metode *voice activity detector*. Sebenarnya dua proses diatas saja sudah cukup untuk mengambil suara yang masuk, namun pada bagian akhir dari rekaman akan terdapat data yang sebenarnya tidak mengandung informasi apapun tetapi tetap akan tersimpan. Tujuan dari proses pemotongan suara ini adalah agar data suara yang terekam menjadi lebih efektif dan seluruh durasi rekaman suara yang didapat mengandung informasi. Untuk itu proses pemotongan suara ini diperlukan. Data suara yang dipotong merupakan data bagian terkahir yang masuk pada proses perekaman suara. Jadi pada data terkahir tersebut dialakukan pengecekan lagi dengan menggunakan *sliding window* dengan durasi waktu 0.25 detik. Berikut adalah programnya:

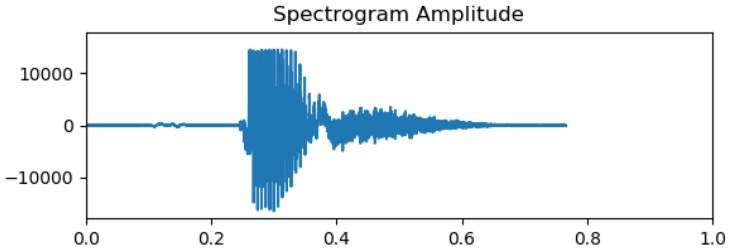
```
last_record = record[(len(record)-1)-
last_rel:]
   for i in range(len(last_record)-1):
     record.pop(len(record)-last_rel+i)
   for i in range(len(last_record)-1):
     last_cur_data = last_record[i]
     last_slid_win.append
(math.sqrt(abs(audioop.avg (
     np.asarray(last_cur_data), 4))))
     if (sum([x>LAST_THRESHOLD for x in
         last_slid_win])>0):
       record.append(last_cur_data)
```

Pada proses ini data suara yang telah direkam akan dipecah kembali untuk memisahkan data rekaman suara bagian akhirnya. Setalah didapat bagian

akhir data tersebut *sliding window* akan bergerak untuk melakukan pengecekan nilai ambang batasnya. Nilai ambang batas pada bagian akhir ini bisa berbeda dengan nilai ambang batas pada proses awal merekam. Apabila nilai yang terdapat pada *sliding window* memiliki nilai kurang dari ambang batasnya, maka data suara tersebut akan dipotong pada bagian *sliding window* saat itu berada, sedangkan untuk sisanya akan dibuang. Perbedaan merekam suara melalui porses pemotongan suara dan tidak dapat dilihat pada Gambar 3.5 dan Gambar 3.6.



Gambar 3.5 Plot Rekaman Suara tanpa *Cutting*



Gambar 3.6 Plot Rekaman Suara dengan *Cutting*

Berdasarkan kedua gambar diatas dapat disimpulkan bahwa, melalui proses pemotongan suara atau *cutting* data rekaman yang didapat akan lebih efektif karena bagian rekaman yang tidak menggandung informasi telah dipotong.

3.1.1.4 Menyimpan Data Rekaman

Setelah melalui proses VAD selanjutnya data rekaman suara yang didapatkan akan disatukan untuk selanjutnya disimpan dalam

waveform audio format (.WAV). Berikut adalah program untuk menyimpan data rekaman yang telah diproses:

```
save_speech(list(prev_audio) + record, p)
```

Varibael prev_audio berisi data rekaman suara pada bagian awal, sedangkan variabel record berisi data rekaman suara dari awal sampai rekaman berhenti dan telah melalui porses pemotongan suara pada bagian akhirnya. Jadi agar menjadi rekaman yang utuh maka kedua variabel tersebut harus ditambahkan. Berikut adalah isi dari fungsi save_speech:

```
def save_speech(data, p):
    filename = 'data_record/output'
    data = b''.join(data)
    wf = wave.open(filename + '.wav', 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(p.get_sample_size(
    pyaudio.paInt16))
    wf.setframerate(44100)
    wf.writeframes(data)
    wf.close()
    return filename + '.wav'
```

Pada fungsi tersebut terdapat beberapa parameter yang harus ditentukan seperti tipe data yang digunakan, *frame rate*, dan jumlah *channel*.

3.1.2 Membangun Model CNN

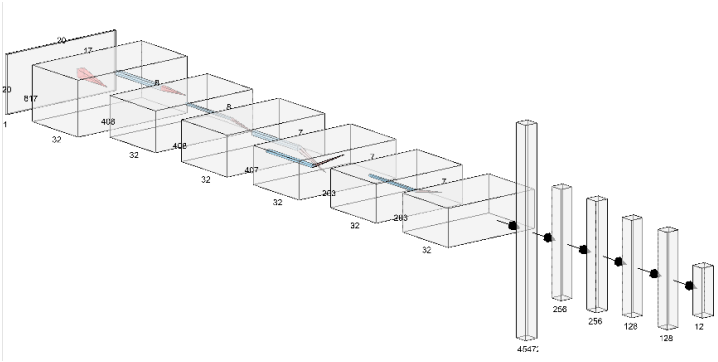
Untuk membuat model CNN (*Convolutional Neural Network*) digunakan API Keras dengan bahasa pemrograman python. Keras telah menyediakan fitur-fitur yang berguna dalam proses yang berkaitan dengan CNN, sehingga untuk membangun sebuah model CNN menjadi lebih mudah. Didalam membangun model CNN terdapat dua bagian yang harus ditentukan yaitu bagian layer konvolusi dan *fully connected layer*. Berikut adalah program dari layer konvolusi:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(4, 4),
          activation='relu',input_shape=
          (20, 820, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(32, kernel_size=(2, 2),
          activation='relu'))
model.add(MaxPooling2D(pool_size=(1, 2)))
model.add(Dropout(0.25))
```

Fungsi dari layer konvolusi adalah untuk mengeluarkan bagian-bagian yang dominan dari setiap data masukan, jadi meskipun letak atau posisi dari bagian-bagian yang dominan pada setiap data berbeda, namun dengan adanya layer konvolusi ini bagian tersebut tetap dapat diketahui. Setelah melalui layer konvolusi selanjutnya adalah menuju *full connected layer*. Berikut adalah program dari *fully connected layer*:

```
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(12, activation='sigmoid'))
```

Pada bagian *fully connected layer* digunakan satu *hidden layer* dengan fungsi relu dan sigmoid. Alasan digunakanya fungsi sigmoid yaitu karena data yang dimasukan memiliki dua buah label atau disebut dengan multilabel, sehingga fungsi aktifasi yang lebih cocok adalah menggunakn sigmoid karena akan digunakan untuk melakukan klasifikasi dua label. Sedangkan fungsi aktifasi softmax lebih efektif untuk klasifikasi dengan satu label.



Gambar 3.7 Ilustrasi Model CNN yang Dibangun

3.1.3 Mengambil Fitur Suara

Untuk mengambil fitur suara yang telah didapatkan yaitu dengan menggunakan metode *Mel Frequency Cepstral Coefficients* (MFCC). Dalam proses pengambilan fitur dengan MFCC digunakan API Librosa yang telah menyediakan fitur-fitur yang berhubungan dengan MFCC. Karena semua perhitungan dari MFCC telah disediakan oleh Librosa, maka yang perlu dilakukan yaitu melakukan *setting* beberapa nilai parameter agar mendapat hasil keluaran fitur yang diinginkan. Berikut adalah program untuk mengambil fitur MFCC pada rekaman suara yang telah didapatkan:

```
mfcc=librosa.feature.mfcc(wave, sr=44100, n_mfcc=20,
hop_length=64)
```

Nilai dari n_mfcc adalah jumlah koefisien yang ingin didapatkan sedangkan nilai hop_length adalah nilai *sampling* yang digunakan untuk rekaman suara yang dimasukan. Sehingga pada program ektraksi fitur tersebut nantinya akan didapat keluaran sebuah matriks dua dimensi berukuran 20 x N karena jumlah dari koefisien yang didapat akan selalu tetap, sedangkan hasil dari *sampling* bergantung pada durasi waktu rekaman. Jadi apabila hasil dari *sampling* yang didapat juga akan berbeda apabila durasi waktu setiap rekaman berbeda. Setiap input yang akan dimasukan kedalam CNN harus memiliki nilai ukuran yang sama,

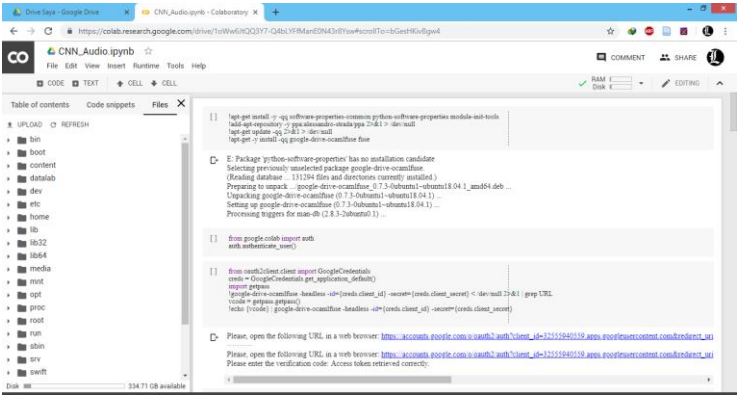
sehingga untuk mengatasi hasil ektrkasi fitur yang memiliki ukuran yang tidak sama, maka perlu dilakukan pemotongan ukuran atau penambahan ukuran pada ektraksi fitur yang didapatkan agar setiap ekstraksi fitur tersebut memliki ukuran yang sama. Berikut adalah programnya:

```
if (max_len > mfcc.shape[1]):
  pad_width = max_len - mfcc.shape[1]
  mfcc = np.pad(mfcc, pad_width=((0, 0),
         (0, pad_width)),mode='constant')
else:
  mfcc = mfcc[:, :max_len]
```

Nilai dari variabel max_len akan menentukan ukuran maksimal dari ekstraksi fitur yang didapat, pada program yang tekah dibuat, ukuran maksimalnya diatur pada nilai 820. Sehingga setiap fitur yang didapatkan dari sebuah rekaman akan menghasilkan keluaran matriks berukuran 20 x 820.

3.1.4 *Learning* Data Suara

Proses *learning* data suara berfungsi untuk menghasilkan model jadi yang nantinya dapat digunakan untuk melakukan pengenalan suara.



Gambar 3.8 Tampilan Google Colaboratory

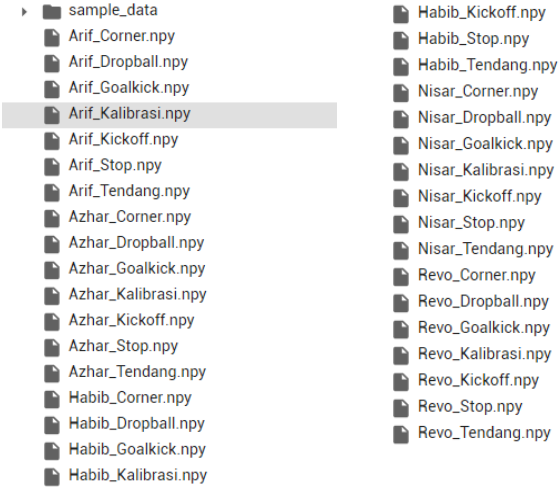
Proses *learning* data suara dilakukan pada *tools* yang telah disediak oleh Google yaitu Google Colaboratory. Tampilan dari Google Colaboratory dapat dilihat pada Gambar 3.8. Sebelum melakukan proses *learning* data, pertama kali yang harus dialakukan yaitu menghubungkan Google Colaboratory dengan Google Drive karena data rekaman suara yang telah didapatkan disimpan pada Google Drive. Berikut adalah program untuk menghubungkan Google Colaboratory dan Google Drive:

```
!apt-get install -y -qq software-properties-
common python-software-properties module-
init-tools
!add-apt-repository -y ppa:alessandro-
strada/ppa 2>&1 > /dev/null
!apt-get update -qq 2>&1 > /dev/null
!apt-get -y install -qq google-drive-
ocamlfuse fuse

from google.colab import auth
auth.authenticate_user()

from oauth2client.client import
GoogleCredentials
creds =
GoogleCredentials.get_application_default()
import getpass
!google-drive-ocamlfuse -headless -
id={creds.client_id} -
secret={creds.client_secret} < /dev/null
2>&1 | grep URL
vcode = getpass.getpass()
!echo {vcode} | google-drive-ocamlfuse -
headless -id={creds.client_id} -
secret={creds.client_secret}
```

Setelah saling terhubung dengan Google Drive, kemudian menuliskan program *learning* data pada *notebook* Google Colaboratory. Selanjutnya, sebelum data dimasukan didalam proses *learning*, setiap data akan disimpan dalam array yang berisi nilai dari ektrkasi fitur dan labelnya masing-masing. Berikut adalah programnya:

```
def save_data_to_array(path=DATA_PATH,
                       max_len=11):
    labels = get_labels(path)
    for label in labels:
        mfcc_vectors = []
        wavfiles = [path + label + '/' +
wavfile for wavfile in
        os.listdir(path + '/' + label)]
        for wavfile in wavfiles:
            mfcc = wav2mfcc(wavfile,max_len =
                            max_len)
            mfcc_vectors.append(mfcc)
        np.save(label + '.npy', mfcc_vectors)
```

Setiap data rekaman suara akan disimpan pada array dengan labelnya masing-masing, sehingga nantinya akan membuat proses *learning* data menjadi lebih cepat dan efisien. Hasil dari penyimpanan data ekstraksi fitur dan labelnya dapat dilihat pada Gambar 3.9.



Gambar 3.9 Hasil Ekstraksi Fitur dan Label

Selanjutnya isi dari setiap *file* tersebut siap untuk dimasukna sebagai input CNN yang telah dibuat. Sebelum menjalankan proses *learning* data, sebaiknya melakukan beberapa pengaturan parameter seperti pada program berikut:

```
model.compile(loss=keras.losses.
              binary_crossentropy,
              optimizer= keras.optimizers.
              Adam(lr=0.00005),
              metrics=['accuracy'])
```

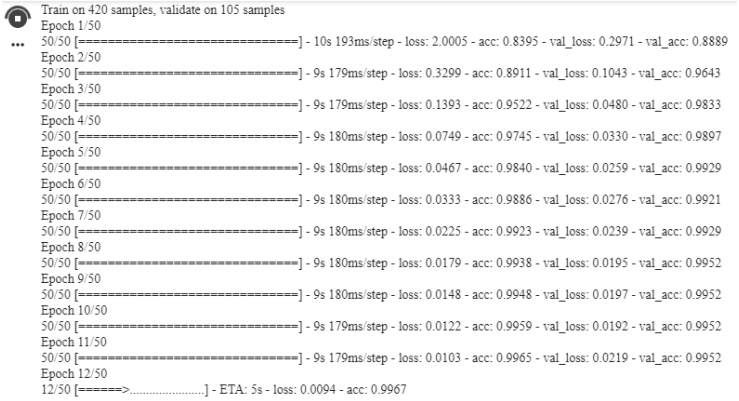
Untuk menghitung nilai *loss* pada proses *learning* digunakan metode Binary Cross-Entropy karena akan digunakan untuk pengklasifikasian dua label. Berikut adalah persamaan dari Binary Cross-Entropy:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \quad (3.1)$$

Sedangkan untuk *optimizer* menggunakan Adam Optimizer karena cara pengaturanya yang mudah dan bekerja dengan baik pada banyak masalah yang ingin diselesaikan. Selanjutnya yaitu menentukan parameter yang akan dilakukan saat proses *learning* berlangsung. Berikut adalah program untuk menjalankan proses *learning* CNN:

```
history = model.fit(X_train,y_train,
          batch_size=None,steps_per_epoch =
          50,epochs=50,verbose=1,
          validation_data=(X_test, y_test),
          validation_steps=100)
```

Pada proses *learning* CNN, data yang telah disiapkan akan dibagi menjadi dua bagian yaitu data *train* dan data *validation*. Data *train* adalah data yang akan dimasukan dalam proses *learning*, sedangkan data *validation* hanya digunakan untuk melakukan pengujian pada model selama proses *learning* berlangsung. Jadi data *validation* tidak masuk didalam proses *learning*. Proses *learning* dari CNN dapat dilihat pada Gambar 3.10.



Gambar 3.10 Proses *Learning* CNN

Apabila proses learning data dari CNN sudah selesai, maka selanjutnya model yang telah dilatih tersebut akan disimpan didalam format .h5, sehingga model tersebut nantinya dapat digunakan kembali untuk melakukan pengenalan suara. Berikut adalah program untuk menyimpan model yang telah dilatih:

```
model.save('/content/godrive/data_train/
            Save_Model/model.h5')
```

Jadi pada berkas tersebut berisi model yang telah dibuat dan nilai dari bobot yang diperoleh setalah melalui proses *learning*.

3.1.5 Prediksi Menggunakan Hasil *Learning*

Setelah proses *learning* data telah selesai, kemudian hasil tersebut disimpan didalam *file* model.h5. model tersebut digunakan dalam sistem pengenalan suara yang dibut. Untuk itu, model yang telah disimpan tersebut harus di muat kembali agar dapat digunakan. Berikut adalah program untuk memuat *file* tersebut:

```
model = Sequential()
model = load_model ('Model_Hasil_Train/
                     model.h5')
```

Setelah *file* tersebut dimuat, artinya model hasil *learning* tersebut telah siap digunakan kembali untuk melakukan pengenalan data suara. Dalam melakukan prediksi sistem pengenalan suara, terdapat dua label yang ditambahkan, yaitu "Ucapan tidak dikenal" dan "Orang tidak dikenal". Kedua label ini tidak dimasukan dalam proses pelatihan data. Kedua label tersebut diberikan apabila data suara yang diprediksi memiliki nilai kurang dari 0.9 untuk masing-masing kalasifikasi. Berikut adalah program prediksi pada sistem pengenalan suara yang dibuat:

```
Prediksi = model.predict(sample_reshaped)
prediksi_speech = prediksi[:,np.r_[2:5,
                            6:8, 10:12]]
prediksi_speaker = prediksi[:,np.r_[0:2, 5,
                             8:10]]
data_speech=speech[np.argmax
                   (prediksi_speech)]
data_speech_raw=prediksi_speech[:,np.argmax
                          (prediksi_speech)]
if data_speech_raw < 0.9:
   data_speech = 'Ucapan tidak dikenal'
data_speaker = speaker[np.argmax
                       (prediksi_speaker)]
data_speaker_raw = prediksi_speaker
            [:,np.argmax(prediksi_speaker)]
if data_speaker_raw < 0.9:
   data_speaker = 'Orang tidak dikenal'
```

Jadi didalam sistem penegenalan yang dibuat, setelah data suara dimasukan kedalam model CNN yang telah dimuat, selanjutnya label dibagi menjadi dua, yaitu label ucapan dan nama orang. Setelah itu, proses prediksi dilakukan pada masing-masing label tersebut. Hasil akhir dari proses prediksini ini adalah klasifikasi ucapan dan nama orang yang cocok.

3.1.6 Pengiriman Hasil Pengenalan suara

Data suara yang telah selesai memasuki proses pengenalan, selanjutnya data hasil penegenalan tersebut disampaikan atau dikirimkan menuju *basestation*. Untuk mengirimkan data hasil pengenalan tersebut digunakan metode UDP (*User Datagram Protocol*) secara lokal, sehingga

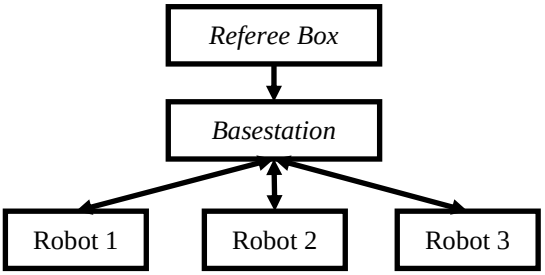
diperlukan untuk melakukan pengaturan dari IP dan port lokalnya. Berikut adalah program untuk mengirimkan data hasil pengenalan menuju *basestation*:

```
if data_speaker == 'Azhar':
  sock.sendto(data_speech.encode(),(UDP_IP,
              UDP_PORT))
```

Hasil dari pengenalan sebuah suara yang masuk adalah menghasilkan prediksi klasifikasi data suara tersebut kedalam dua label, yaitu siapa yang berbicara dan apa yang diucapkan. Apabila hasil prediksi dari suara yang masuk bukan suara dari pelatihnya, maka hasil prediksi ucapanya tidak diteruskan menuju *basestation*. Sistem yang dibuat ini dirancang untuk meneruskan perintah yang diucapkan oleh pelatihnya, sedangkan ucapan dari orang lain akan diabaikan.

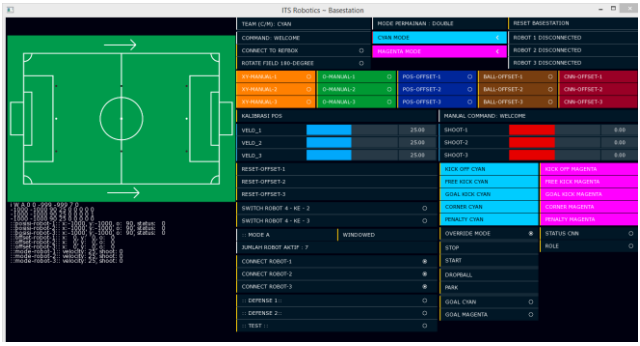
3.2 Perancangan *Basestation*

Basestation merupakan bagian yang berperan penting dalam permainan robot sepak bola beroda. Semua komunikasi yang dilakukan oleh robot harus melalui sebuah *basestation*. Jadi dengan menggunakan *basestation* setiap robot dapat mengetahui semua informasi tentang robot lain seperti nilai pembacaan sensor setiap robot, posisi setiap robot, dan kondisi setiap robot. Selain itu *basestation* juga dapat meneruskan perintah yang diberikan oleh *referee box* atau *game controller* wasit menuju setiap robot yang sedang bermain, untuk koneksi jaringan pada *basestation* dapat dilihat pada Gambar 3.11. Untuk itu *basestation* yang digunakan pada robot sepak bola ini dirancang dengan menggunakan GUI (*Graphical User Interface*). Dengan menggunakan GUI interaksi antara pengguna dan komputer dalam memantau setiap robot yang sedang beroperasi menjadi lebih mudah. Tampilan GUI yang dibuat dapat dilihat pada Gambar 3.12.



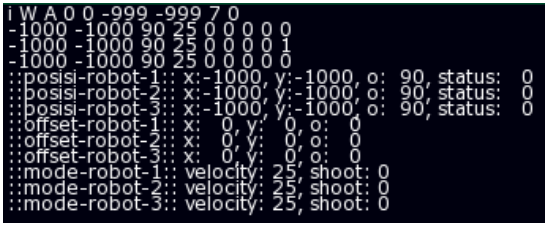
Gambar 3.11 Diagram Blok koneksi *Basestation*

Agar *basestation* dan robot dapat saling berkomunikasi sesuai dengan digram blok pada Gambar 3.11, maka *basestation* dan setiap robot yang aktif harus terhubung dengan sebuah *access point* atau Wi-fi. Komunikasi pengiriman dan penerimaan data dilakukan menggunakan protokol UDP (*User Datagram Protocol*). Dalam berkomunikasi *basestation* dan robot memiliki cara yang berbeda, *basestation* menggunakan metode *multicast*, sedangkan setiap robot menggunakan metode *unicast*. Jadi untuk mengirimkan data, *basestation* hanya perlu menggirimkan data pada IP (*Internet Protocol Address*) dan *port multicast* yang telah dibuat. Agar robot dapat menerima data dari *basestation*, maka setiap robot harus mengakses IP dan *port* tersebut untuk mendapatkan data yang dikirimkan. Sedangkan pada robot, pengiriman data terjadi secara *unicast*, artinya setiap robot akan mengirimkan data lanngsung menuju pada IP dan *port* yang dimiliki oleh *basestation*.



Gambar 3.12 Tampilan GUI *basestation*

Sebenarnya pada GUI *basestation* yang telah dibuat telah disediakan beberapa tombol untuk memberikan perintah pada robot, seperti mode permainan, *offset* posisi dan sudut, status permainan, dan lain-lain. Perintah-perintah yang diberikan oleh *basaestation* tersebut dikirimkan pada setiap robot dalam bentuk array byte. Data yang dikirimkan dan diterima dari robot pada basestation dapat dilihat pada Gambar 3.13.



Gambar 3.13 Data Penerimaan dan Pengiriman Basestation

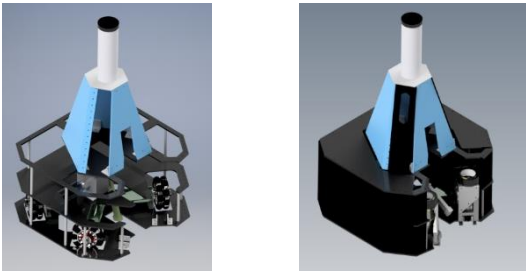
Data yang dikirimkan pada robot dapat dilihat pada Gambar 3.13 pada baris pertama, sedangkan pada baris kedua sampai keempat merupakan data yang diberikan oleh setiap robot, kemudian data tersebut dikirimkan kembali menuju semua robot agar setiap robot dapat mengetahui kondisi dan status robot yang lain. Inti dari penelitian tugas akhir ini adalah memberikan perintah pada robot menggunakan suara, artinya data yang dikirimkan *basestation* akan berubah secara otomatis sesuai dengan perintah yang diberikan melalui suara. Jadi sistem pengenalan yang dibuat akan dihubungkan dengan *basestation* ini. Pengiriman data hasil pengenalan suara tersebut akan dikirimkan melalui UDP lokal dan akan diterima oleh *basestation* untuk memberikan perintah pada robot. Berikut adalah program pada *basestation* untuk menerima data dari sistem pengenalan suara yang telah dibuat.

```
while (isThreadRunning()){
   while (true) {
   int bytesIn = recvfrom(in, buf, 1024, 0,
   (sockaddr*)&client,&clientLength);
    status = 1;
    if (bytesIn == SOCKET_ERROR){
    cout << "Error receiving from client
    "<< endl;
    continue;
      }
   }
}
```

Data yang telah diterima tersebut kemudian akan diolah oleh *basestation* untuk mengganti pesan yang dikirim pada robot.

3.3 Desain Mekanik Robot

Robot sepak bola beroda yang dibuat pada penelitian tugas akhir ini menggunakan plat alumunium untuk membuat kerangka badan robot. Robot sepak bola ini juga dilapisi oleh akrilik pada bagian tempat peletakan perangkat elektroniknya. Robot ini menggunakan sistem penggerak empat roda dengan menggunakan roda *omni wheel*. Selain itu robot ini dilengkapi dengan *encoder* untuk menentukan posisi robot. Pada bagian atas robot terdapat kamera omnidireksional yang dilindungi oleh tabung akrilik. Desain mekanik robot dapat dilihat pada Gambar 3.14.

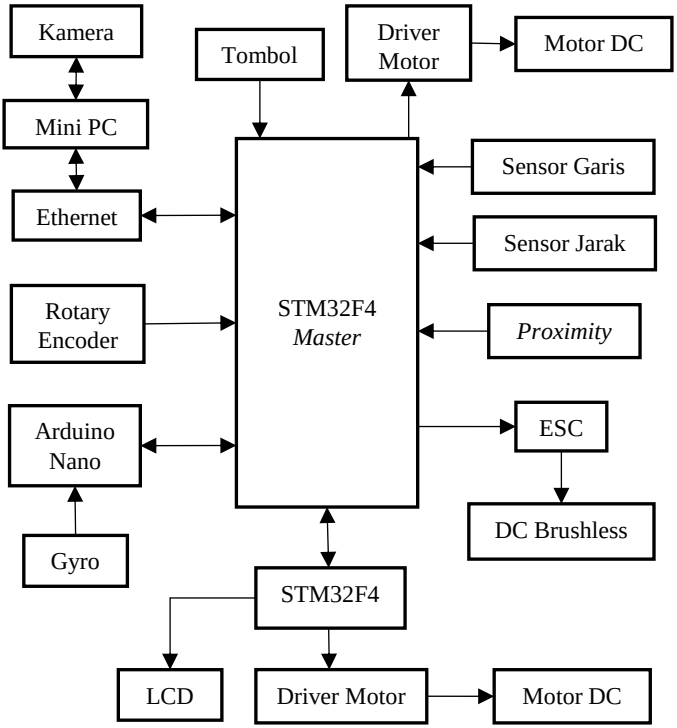


(a) (b)

Gambar 3.14 Desain (a) Kerangka Badan (b) Seluruh Robot

3.4 Perancangan Elektronik

Perancangan elektronik pada robot sepak bola beroda yang dibuat terdiri dari beberapa bagian yang saling terhubung. Semua sensor dan aktuator akan diproses menjadi satu sehingga menjadi satu kesatuan sistem elektronik robot yang utuh. Perancangan elektronik pada robot dapat dilihat pada Gambar 3.15.



Gambar 3.15 Diagram Blok Sistem Elektronik Robot

3.4.1 *Power Supply*

Power supply merupakan komponen penting pada robot karena berfungsi untuk mengubah tegangan sumber menjadi tegangan yang dibutuhkan oleh robot. Sumber tegangan yang digunakan pada robot adalah dua buah baterai Li-ion dengan tegangan 24V, satu untuk sumber

tegangan motor dan satunya lagi untuk sumber tegangan mikrokontroler, mini PC, dan sensor-sensor yang dipakai. Sehingga pemilihan spesifikasi *power supply* merupakan hal yang harus diperhatikan karena *power supply* yang digunakan harus dapat mencukupi kebutuhan tegangan dan arus yang dibutuhkan oleh robot. Dalam robot sepaka bola pada penelitian tugas akhir terdapat tiga jenis *power supply* yang digunakan untuk memberikan *supply* kepada komponen-komponen elektronik yang digunakan pada robot.

3.4.1.1 *Buck Converter* untuk Mikrokontroler

Kebanyakan mikrokontroler membutuhkan tegangan sumber dengan rentang tegangan 3.3V sampai 5V. Terdapat tiga mikrokontroler yang digunakan pada robot sepak bola beroda ini sehingga *power supply* yang digunakan harus dapat mencukupi kebutuhan arus pada ketiga mikrokontorler tersebut. Selain itu, terdapat beberapa sensor yang juga membutuhkan sumber tegangan 5V.

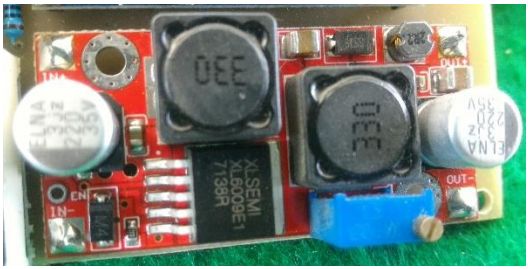
Gambar 3.16 LM2596 *Buck Converter*

Modul LM2596 *Buck Converter* ini berfungsi untuk menurunkan tegangan sumber baterai 24V menjadi tegangan 5V utuk digunakan pada mikrokontroler dan beberapa sensor. Selain itu, modul ini juga memiliki arus keluaran maksimal sebesar 3A, sehingga modul ini lebih dari cukup untuk mensupali tegangan dan arus pada mikrokontroler yang digunakan.

3.4.1.2 *Buck Boost Converter* untuk Sensor

Buck boost converter memiliki kemampuan menaikan atau menurunkan tegangan masukanya dengan keluaran arus maksimal sebesar 4A. Pada robot sepak bola beroda ini digunakan beberapa sensor

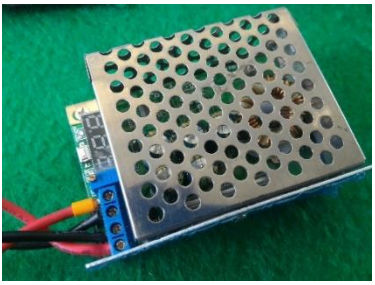
yang membutuhkan tegangan masukan sebesar 12V. Untuk itu *buck boost converter* digunakan untuk mensuplai tegangan dan arus pada sensor-sensor tersebut. Sensor yang menggunakan tegangan 12V yaitu sensor garis, sensor *proximity*, dan *rotary encoder*.



Gambar 3.17 XL6009 Buck Boost Converter

3.4.1.3 *Buck Converter* untuk Mini PC

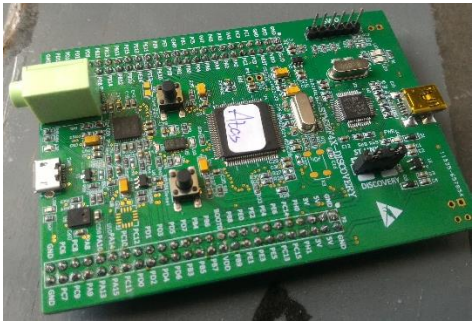
Mini PC yang digunakan pada robot sepak bola beroda ini yaitu NUC6i7KYK. Mini PC tersebut membutuhkan tegangan masukan sebesar 19V. selain itu, mini PC yang digunakan membutuhkan suplai arus sebesar 3A ketika program robot dijalankan, untuk itu *buck converter* yang digunakan juga harus memiliki kemampuan untuk mensuplai arus melebihi arus yang dibutuhkan oleh mini PC. *Buck converter* yang digunakan pada robot ini memiliki keluaran arus sebesar 10A, sehingga sudah lebih dari cukup untuk memenuhi kebuthan sumber arus yang dibuthkan oleh mini PC.



Gambar 3.18 Buck Converter 10A

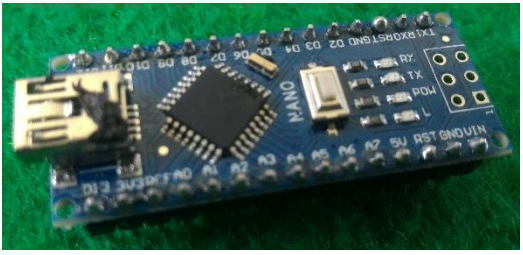
3.4.2 Mikrokontroler

Pada robot sepak bola beroda ini digunakan tiga buah mikrokontroler, yaitu dua STM32F4 *discovery* dan satu arduino nano. Dua mikrokontroler STM32F4 difungsikan sebagai *master* dan *slave*, sedangkan arduino nano juga difungsikan sebagai mikrokontroler *slave*.



Gambar 3.19 STM32F4 *discovery*

STM32F4 *master* memiliki beberapa tugas yang lebih banyak daripad STM32F4 yang digunakan sebagai *slave*. Pada mikrokontorler *master* bertugas untuk membaca seluruh nilai hasil pembacaan sensor yang digunakan, mengatur tendangan dan *dribbling* bola pada robot, menerima data *heading* gyro dari arduino nano, dan untuk berkomunikasi dengan mini PC. Sedangkan STM32F4 *slave* bertugas untuk mengatur fungsi pergerakan robot dan menampilkan data hasil pembacaan sensor dan data kecepatan yang diberika oleh mini PC pada LCD.



Gambar 3.20 Arduino Nano

Slave arduino nano betugas untuk mengirimkan data *heading* gyro menuju mikrokontroler *master*. Data *heading* gyro tersebut didapatkan dari sensor IMU MPU6050. Untuk mendapatkan data tersebut arduino nano bertugas untuk meminta data pada IMU MPU6050 melalui protokol I^2C.

3.4.3 Sensor

Pada robot sepak bola ini, digunakan beberapa sensor yang berfungsi untuk membantu robot dalam bermain sepak boal. Setiap sensor yang digunakan memiliki peran dan fungsinya masing-masing.

3.4.3.1 Sensor Jarak

Sensor jarak pada robot sepak bola, berfungsi untuk mendeteksi apakah bola sudah memasuki sistem *dribble* robot. Sensor jarak yang digunakan yaitu Sharp GP2D12. Sensor jarak tersebut menggunakan sinsar infra merah untuk mengukur jaraknya.

Gambar 3.21 Sensor Jarak Sharp GP2D12

Rentang jarak pembacaan sensor ini efektif pada jarak 10cm sampai 90cm. Sedangkan untuk keluaran dari sensor ini adalah sinyal analog, yang nantinya dapat dibaca oleh mikrokontroler melalui ADC. Sensor jarak GP2D12 dapat dilihat pada Gambar 3.21.

3.4.3.2 Sensor Garis

Sensor garis pada robot sepak bola berodai ini berfungsi untuk mendeteksi garis putih yang ada pada lapangan, sehingg sensor garis tersebut dapat digunakan untuk mengkalibrasi posisi dan sudut robot pada

lapangan. Sesnsor garis yang digunakan yaitu autonic tipe BF5R yang membutuhkan tegangan masukan sebesar 12V. Sensor garis tersebut dapat dilihat pada Gambar 3.22.

Gambar 3.22 Sensor Garis Autonic BF5R

Sebenarnya sensor garis ini menggunakan data analog sebagai pembacaan sensornya, namun pada sistem sensor garis tersebut sudah tersedia pengaturan nilai ambang batas pembacaan sensor, sehingga keluran dari sensor garis tersebut adalah logika 1 dan 0. Logika 1 akan dikeluarkan sensor ketika tidak mendeteksi garis dan logika 0 untuk mendeteksi garis. Hasil pembacaa sensor tersebut dihubungkan pada pin *logic* STM32F4.

3.4.3.3 Sensor *Proximity*

Robot sepak bola beroda yang dibuat ini memiliki sistem penendang menggunakan motor, sehingga sensor *proximity* pada robot ini dibutuhkan untuk menghitung jumlah putaran kaki penendang robot. Sumber tegangan yang dibutuhkan oleh sensor ini yaitu 12V dan keluaran dari sensor ini berupa logika 1 dan 0. Keluaran dari sensor ini dihubungkan pada pin eksternal *interrupt* dari STM32F4.

Gambar 3.23 Sensor Proximity

3.4.3.4 Sensor IMU

Sensor IMU yang digunakan pada robot sepak bola beroda ini adalah MPU6050. Sensor MPU6050 ini merupakan sensor MEMS (*Micro Electro Mechanical System*). Didalam MPU6050 terdapat dua jenis sensor yaitu 3 axis gyroskop dan 3 axis akselerometer. Data kedua sensor tersebut diolah dan dikombinasikan melalui proses internal sensor IMU tersebut. Proses pengolahan data tersebut dikenal dengan nama DMP (*Digital Motion Processor*).

Gambar 3.24 Sensor IMU MPU6050

Selain itu, MPU6050 sebenarnya dapat ditambahkan sensor magnetometer eksternal agar perhitungan *heading* menjadi lebih baik. Untuk mengakses data dari sensor IMU ini digunakan *interface* I^2C. Untuk itu agar arduino nano harus menggunakan pin SCL dan SDA untuk mendapatkan nilai pembacaan sensor tersebut. Tegangan sumber yang dibutuhkan oleh sensor IMU ini adalah sebsesar 5V.

3.4.3.5 *Rotary Encoder*

Fungsi dari *rotary encoder* pada robot sepak bolad pada penelitian tugas akhir ini adalah untuk menghitung jarak perpindahan pada robot. *Rortary encoder* yang digunakan pada robot ini adalah sensro autonic seri E30, dapat dilihat pada Gambar 3.24. Sensor ini bekerja pada rentang tegangan 5 – 24V. pada *rotary encoder* tersebut akan dihasilkan sebanyak 200 pulsa setiap satu kali putaran. Untuk memberikan informasi posisi robot pada sumbu x dan y, maka dibutuhkan dua buah *rotary encoder* pada robot sepak bola beroda ini

Gambar 3.25 Rotary Encoder Autonic E30

Sensor *rotary encoder* ini memiliki dua keluaran, yaitu berupa *channel* A dan *channel* B. Kedua keluaran ini adalah berupa sebuah pulsa yang memiliki perbedaan fasa sebesar 90 derajat. Sehingga *rotary encoder* dapat memberikan informasi tentang arah putaranya dan berapa banyak putaran yang telah dilakukan. Kedua keluaran tersebut dihubungkan pada mikrokontroler STM32F4 *master* untuk diproses menjadi sebuah perhitungan perpindahan posisi robot.

3.4.3.6 Kamera

Robot sepak bola beroda pada penelitian tugas akhir menggunakan sensor kamera untuk dapat mendeteksi bola dan halangan yang ada dilapangan. Jadi sensor kamera ini dapat dikatakan sebagai mata untuk robot. Jenis kamera yang digunakan pada robot ini adalah kamera logitech seri C922. Kamera ini mendukung perekaman video pada resolusi 1080p dengan 30fps atau 720p dengan 60fps (*frame per second*).

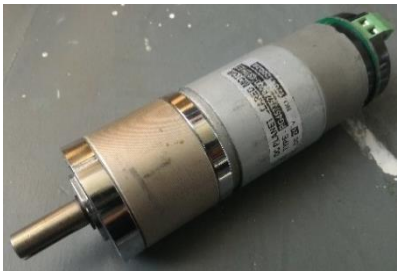
Gambar 3.26 Kamera Logitech C922

3.4.4 Aktuator

Aktuator merupakan suatu peralatan mekanis untuk menggerakan sebuah mekanisme atau melakukan sebuah mekanisme dan sistem tertentu. Robot sepak bola pada penelitian tugas akhir ini menggunakan tigas jenis aktuartor yang masing-masing memiliki peran dan fungsi tersendiri.

3.4.4.1 Motor DC *Brushed*

Terdapat dua jenis motor DC *brushed* yang digunakan pada robot sepak beroda ini, yaitu motor DC PG45 dan PG36. Motor DC PG45 (*Planetary Gear*) berfungsi sebagai motor penggerak bagi robot



Gambar 3.27 Motor DC PG45

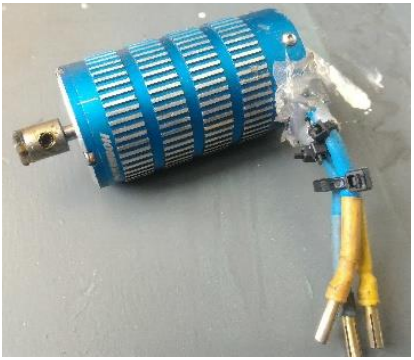
Pada robot sepak bola ini dibutuhkan empat motor PG45 sebagai mekanisme penggerak pada robot sepak bola beroda ini. Moto ini memiliki tegangan suplai efektif sebesar 24V dengan arus maksimal 4A. Motor PG45 ini memiliki kemampuan berputar sebesar 500rpm (*rotation per minute*) dan torsi sebesar 25Kg.cm.

Gambar 3.28 Motor DC PG36

Sedangkan untuk motor DC PG36 berfungsi untuk melakukan mekanisme sistem *dribble* bola pada robot. Untuk menjalankan sistem tersebut digunakan dua buah motor PG36 yang diletakan pada bagian depan robot. Dengan adanya dua buah motor tersebut, maka robot dapat mengatur kecepatan dan arah putaran bola yang sedang dibawa robot. Motor PG36 ini memiliki tegangan kerja efektif pada tegangan suplai 24V dengan arus maksimal sebesar 2.5A. Kecepatan motor PG36 ini adalah 440rpm dengan torsi 20Kg.cm.

3.4.4.2 Motor DC *Brushless*

Motor DC *brushless* pada robot sepak bola ini berfungsi untuk melakukan mekanisme tendangan pada robot. Karakteristik dari motor DC *brushless* adalah kecepatanya yang tinggi, sedangkan untuk pada robot sepak bola beroda ini membutuhkan torsi yang tinggi untuk dapat melakukan tendangan yang kencang. Untuk itu motor ini dihubungkan pada sebuah *gearbox* yang mengubah kecepatan motor ini menjadi torsi yang tinggi atau kuat.



Gambar 3.29 Motor DC brushless

3.4.5 *Driver* Motor

Pada robot sepak bola beroda yang dibuat ini digunakan dua jenis moto yaitu motor DC *brushed* dan motor DC *brushless*. Kedua jenis motor tersebut memiliki jenis *driver* motor yang berbeda, sehingga pada

robot sepak bola beroda ini digunakan dua jenis *driver* motor yang berfungsi untuk mengatur perputaran kedua jenis motor tersebut.

3.4.5.1 BTN7970

IC BTN7970 merupakan sebuah IC dengan rangkaian *half-bridge*, jadi IC ini hanya dapat mengatur arah putaran motor pada satu direksi saja. Sehingga untuk membuat sebuah rangkaian *H-bridge* yang dapat mengatur putaran motor pada dua direksi arah putaran baik searah jarum jam ataupun berlawanan arah jarum jam setidaknya dibutuhkan dua buah IC BTN7970. Rangkaian tersebut berfungsi untuk mengatur kecepatan dan arah putar dari motor penggerak robot sepak beroda ini. *Driver* motor ini mampu mengalirkan arus maksimal sebesar 70A untuk mensuplai arus yang dibutuhkan motor penggerak robot. *Driver* motor BTN7970 dapat dilihat pada Gambar 3.30.

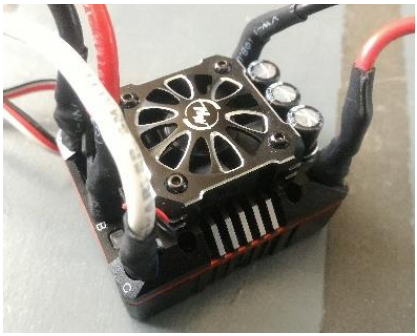
Gambar 3.30 Modul Driver Motor BTN7970

Modul *driver* motor ini memiliki lima buah input, dua input untuk mensuplai motor dan tiga input untuk mengatur putaran dari motor. Untuk mengatur putaran dari motor penggerak robot atau motor *brushed* terdapat tiga input penting yaitu input PWM (*Pulse Width Modulation*) atau kecepatan motor dan input direksi atau arah putaran motor.

3.4.5.2 ESC Xerun XR8 Plus

ESC(*Electronic Speed Controller*) berfungsi untuk mengatur putaran dari motor DC *brushless*. Berbeda dengan cara rangkaian H-*bridge* dalam mengatur putaran motor DC *brushed*. ESC ini memiliki cara

sendiri untuk mengatur arah putaran dari sebuah motor DC *brushless*. ESC ini hanya membutuhkan satu buah input agar dapat mengatur kecepatan motor dan arah putaranya. Input yang dibutuhkan ESC merupakan sebuah pulsa dengan periode 20ms atau frekuensi 50Hz. Pada kondisi netral atau motor tidak beputar, maka sinyal masukan memiliki 1.5ms pulsa *high* dan 18.5 pulsa *low*. Sedangkan untuk arah putaran kedepan sinyal masukan harus memiliki sinyal *high* lebih dari 1.5ms, semakin tinggi nilai sinyal *high* maka putaran kedepan motor akan semakin kencang. Apabila sinyal *high* kurang dari 1.5ms maka motor akan berputar kebelakang. ESC Xerun XR8 Plus dapat dilihat pada Gambar 3.31.



Gambar 3.31 ESC Xerun XR8 Plus

ESC ini mampu mensuplai motor dengan arus maksimal 150A. Selain itu ESC ini dapat diprogram sesuai dengan kemampuan yang dibutuhkan seperti kemampuan *reverse* atau *break* dari motor *brushless*.

3.4.6 Ethernet

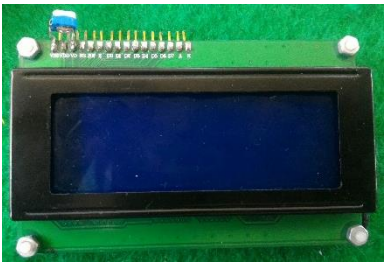
Pada robot sepak bola beroda yang dibuat ini digunakan modul ethernet untuk melakukan komunikasi antara mikrokontroler *master* dengan mini PC. Modul ethernet yang digunakan pada robot sepak bola ini adalah modul LAN8720. Degan menggunakan modul ethernet ini memungkinkan terjadinya komunikasi antara mikrokontroler dengan PC melalui protokol UDP (*User Datagram Protocol*). Komunikasi yang

terjalin antara mikrokontroler dan PC memiliki kecepatan transfer data sebesar 10Mbit/s.

Gambar 3.32 Modul Ethernet Lan 8720

3.4.7 LCD

LCD pada robot sepak bola beroda ini berfungsi untuk menampilkan data-data hasil pembacaan sensor. LCD yang digunakan pada robot ini adalah LCD *alphanumeric* dengan ukuran 20x4. LCD ini terhubung dengan mikrokontroler STM32F4 *slave*. dengan adanya LCD ini dapat mempermudah dalam mengawasi apakah sensor-sensor yang terpasang telah bekerja secara normal atau terjadi suatau permaslahan. LCD yang digunakan dapat dilihat pada Gambar 3.33.



Gambar 3.33 LCD Alpahnumeric 20x4

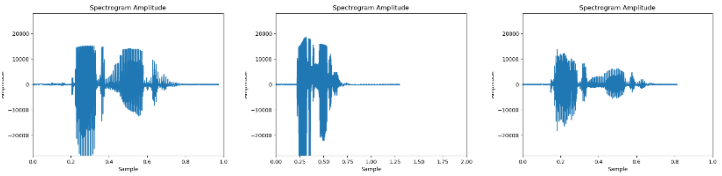
.....Halaman ini sengaja dikosongkan.....

BAB 4
PENGUJIAN DAN ANALISA

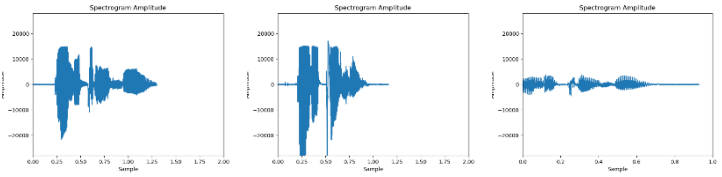
Pada bab ini akan menjelaskan tentang pengujian dari sistem pengenalan suara. Pengujian dan analisa pada bab ini bertujuan untuk mengetahui seberapa besar akurasi sistem pengenalan suara yang dibuat ini dalam memprediksi suara perintah yang diterima dan untuk mengetahui permasalahan dalam perancangan sistem ini sehingga kedepanya dapat ditingkatkan menjadi sistem yang lebih baik. Pengujian pada bab ini terdiri dari pengujian *voice activity detector*, pengujian *learning* model CNN, pengujian pengenalan suara yang terdiri dari pengujian suara *learning* dan pengujian suara *non-learning*.

4.1 Pengujian *Voice Activity Detector*

Voice activity detector (VAD) berfungsi untuk menentukan apakah sinyal yang masuk melalui mikrofon akan direkam atau tidak. Nilai *sampling* yang digunakan utuk VAD ini adalah sebesar 44100, dalam satu detik ada 44100 data yang direkam. Dari hasil pengujian yang dilakukan terhadap sistem VAD yang dibuat dapat dilihat pada Gambar 4.1.



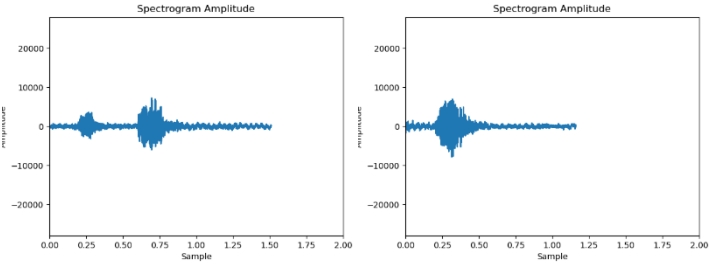
(a)



(b)

Gambar 4.1 Grafik Sinyal Perintah (a)*Corner* (b)Kalibrasi Hasil VAD

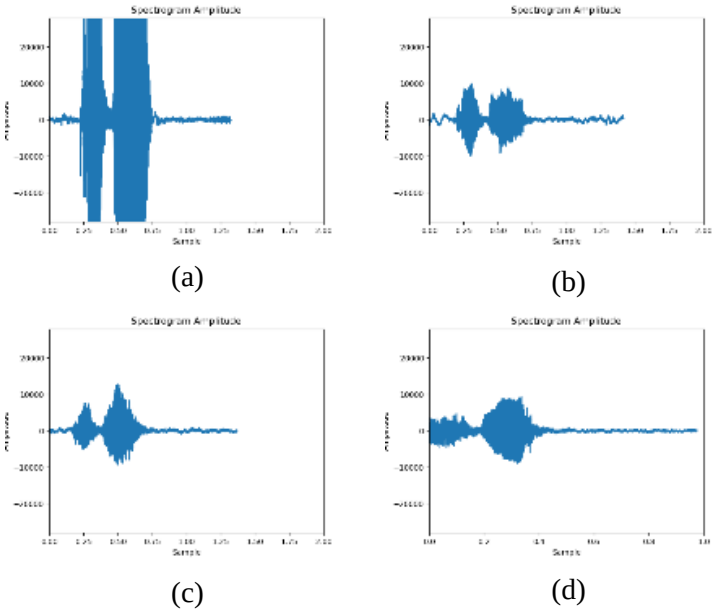
Grafik sinyal yang ditampilkan pada Gambar 4.1 merupakan hasil dari perekaman yang dilakukan oleh VAD. Sinyal suara tersebut berasal dari tiga orang yang berbeda dan dua perintah yang berbeda pula, sehingga sinyal didapatkan memiliki perbedaan. Pada Gambar 4.1 merupakan sinyal yang sukses terekam oleh VAD secara utuh tanpa ada sinyal yang terpotong. Namun pada VAD yang dirancang ini sebenarnya juga terkadang mengalami kegagalan dalam merekam perintah suara yang diucapakan seperti pada Gambar 4.2.



(a) (b)

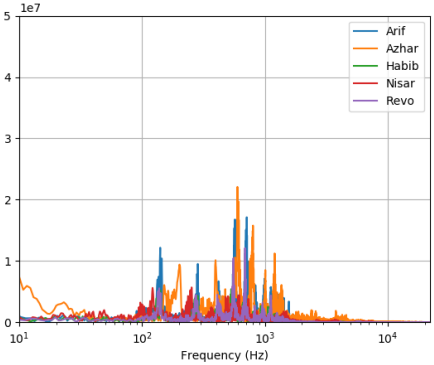
Gambar 4.2 Grafik Sinyal Perintah *Kickoff* (a)Utuh (b)Terpotong

Dari Gambar 4.2 terlihat berbeda, pada gambar (a) suara perintah yang ditangkap dapat terekam secara utuh, sedangkan pada gambar (b) suara perintah yang terlihat nampak mengalami penggurangan. Sehingga informasi yang dikandung suara pada gamabr (b) tidak tersampaikan secara lengkap. Bila dilihat pada gambar (a) sinyal awal yang terekam memiliki nilai amplitudo yang cukup kecil, hal ini dapat menyebabkan sistem VAD yang dibuat ini mengira sinyal tersebut adalah berupa *noise*. Kesalahan tersebut dapat terjadi karena sistem VAD yang digunakan menggunakan metode ambang batas, sehingga ketika suara dari seseorang masuk kedalam mikrofon namun memiliki nilai amplitudo yang rendah, maka suara tersebut dapat dianggap sebagai *noise* oleh VAD.

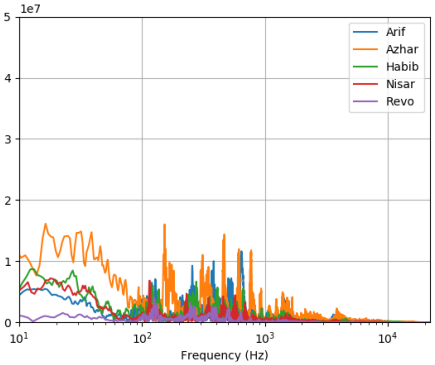


Gambar 4.3 Grafik Sinyal Perintah Dropball pada Jarak (a)5cm (b)25cm (c)50cm (d)100cm

Pada Gambar 4.3 terlihat grafik sinyal perintah suara yang sama namun disampaikan pada jarak yang berbeda. Dari gambar tersebut dapat diketahui bahwa semakin jauh jarak penyampaian perintah suara, maka sinyal yang terekam akan memiliki nilai amplitudo yang semakin kecil yang dapat mengakibatkan sinyal perintah suara yang diterima tidak sampai secara utuh atau terpotong. Hal ini dapat dilihat pada gambar (d). Pada gambar tersebut terlihat bahwa sinyal yang terekam mengalami perpotongan dibagian awal karena amplitudo yang terekam memiliki nilai yang lebih kecil dari nilai ambang batas yang telah ditentukan. Setiap kata perintah yang diucapkan, masing-masing memiliki karakteristiknya sendiri-sendiri, seperti yang dapat dilihat pada Gambar 4.4.



(a)



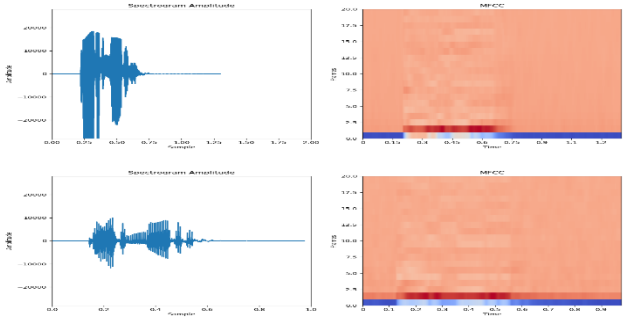
(b)

Gambar 4.4 Grafik Spektrum Frekeunsi Perintah (a)Corner (b)Kalibrasi Data *Learning*

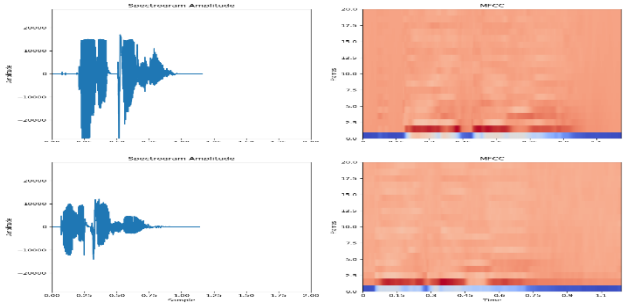
Dari Gambar 4.4 dapat dilihat spektrum frekuensi dari kata perintah kalibrasi dan *corner*. Spektrum frekuensi tersebut diambil dari lima orang yang berbeda. Suara kelima orang tersebut merupakan suara yang digunakan untuk proses *learning* penegenalan suara. Pada gambar tersebut dapat dikatakan bahwa setiap kata perintah memiliki karakteristik spektrum frekuensinya sendiri dan setiap orang memiliki spektrum yang berbeda untuk setiap kata perintah yang diucapkan.

4.2 Ekstraksi Fitur MFCC

Ekstraksi fitur merupakan proses yang dilakukan sebelum data suara yang diterima dimasukan kedalam CNN. Pada proses ektraksi fitur ini, suara yang akan diambil fiturnya merupakan suara yang terikam pada 1.2 detik pertama. Jadi apabila terdapat rekaman yang memiliki nilai lebih dari 1.2 detik, maka rekaman tersebut akan dipotong. Dan untuk rekaman yang kurang dari 1.2 detik, maka rekaman tersebut akan ditambahkan durasinya dengan nilai 0 sampai pada durasi 1.2 detik. Untuk hasil dari ekstraksi fitur MFCC dapat dilihat pada Gambar 4.5.



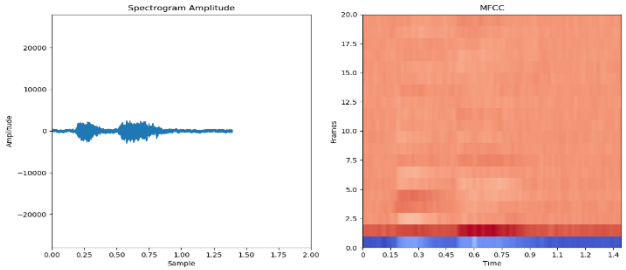
(a)



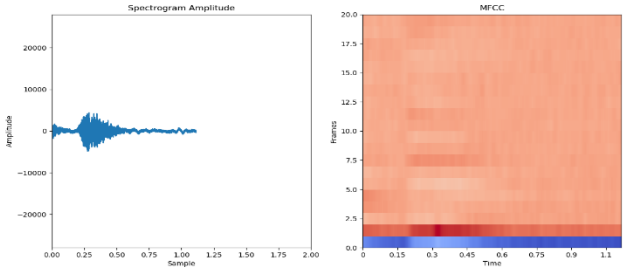
(b)

Gambar 4.5 Hasil Ekstraksi Firtur MFCC (a)*Corner* (b)Kalibrasi Data *Learning*

Pada Gambar 4.5 dapat dilihat pada gambar (a) dan (b) merupakan suara yang dihasilkan oleh dua orang yang berbeda. Pada bagian kiri Gambar 4.5 merupakan sinyal suara yang ditangkap oleh mikrofon, sedangkan sebelah kanan merupakan hasil dari ekstraksi fitur MFCC. Dari gambar tersebut dapat terlihat bahwa, meskipun sinyal suara yang ditangkap oleh mikrofon cukup berbeda, namun hasil dari ekstraksi fitur MFCC masih memiliki pola yang hampir mirip untuk setiap kata perintah yang sama. Sedangkan untuk hasil ektraksi fitur oleh MFCC antara suara perintah yang utuh dan terpotong dapat dilihat pada Gambar 4.6.



(a)



(b)

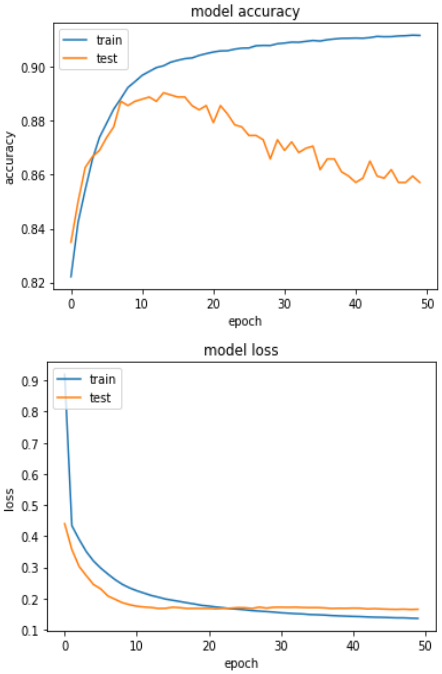
Gambar 4.6 Hasil Ekstraksi Fitur MFCC Kickoff (a)Utuh (b)Terpotong

Berdasarkan Gambar 4.6 terlihat bahwa pada sinyal perintah yang utuh dan terpotong akan menghasilkan ekstraksi fitur yang cukup berbeda. Hal ini terjadi karena pada sinyal yang terpotong, informasi yang didapat tidak

secara kesulurhan. Dan hal ini dapat mempengaruhi hasil dari sistem pengenalan suara yang telah dibuat.

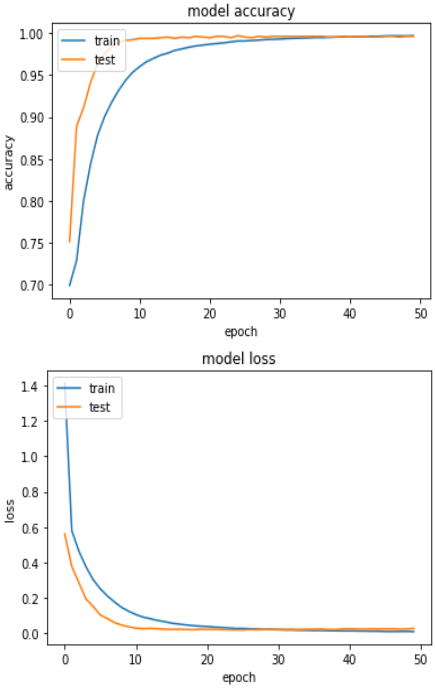
4.3 Pengujian *Learning* model CNN

Didalam pengujian learning model CNN, dilakukan perbandingan akurasi dari penggunaan fungsi aktifasi softmax dan sigmoid pada model CNN yang digunakan. Total data yang digunakan *learning* CNN ini adalah 525 rekaman suara pada jarak yang sama. Data tersebut diambil dari suara 5 orang yang berbeda dan masing-masing orang tersebut mengucapakan 7 kata perintah yang sama, jadi total label yang digunakan dalam proses *learning* data adalah 12 label. Data *learning* tersebut dibagi menjadi dua bagian, yaitu sebagai data pelatihan dan data validasi. Data yang digunakan dalam proses pelatihan adalah 80% dari total. Sedangkan sisanya untuk data validasi.



Gambar 4.7 Grafik Hasil Learning Dengan Fungsi Softmax

Proses *learning* menggunakan fungsi softmax dapat dilihat pada Gambar 4.7. Pada proses *learning* data tersebut terlihat bahwa akurasi prediksi sistem pada data validasi atau *test* semakin mengalami penurunan sedangkan untuk data pelatihan atau *train* semakin mengalami peningkatkan. Fungsi softmax kurang efektif unutk mengklasifikasikan data suara yang memiliki lebih dari satu label.



Gambar 4.8 Grafik Hasil Learning Dengan Fungsi Sigmoid

Berdasarkan Gambar 4.7 dan Gambar 4.8 dapat terlihat jelas bahwa fungsi softmax dan fungsi sigmoid memiliki pengaruh yang besar didalam proses *learning* data. Dapat dilihat pada Gambar 4.8 bahwa model CNN dengan fungsi sigmoid memiliki niali akurasi yang cukup tinggi untuk melakukan prediksi pada data yang dilatihkan dan data yang digunakan dalam tes. Selain itu nilai *loss* dengan menggunakan fungsi sigmoid

memiliki niali yang lebih kecil daripada menggunakan fungsi softmax. Didalam dua proses *learnig* tersebut didapat rata-rata nilai akurasi akhir 85% untuk fungsi softmax dan 99% untuk fungsi sigmoid. Jadi fungsi yang lebih efektif untuk melakukan klasifikasi pada data yang memiliki lebih dari satu label yaitu fungsi sigmoid.

4.4 Pengujian Pengenalan Suara

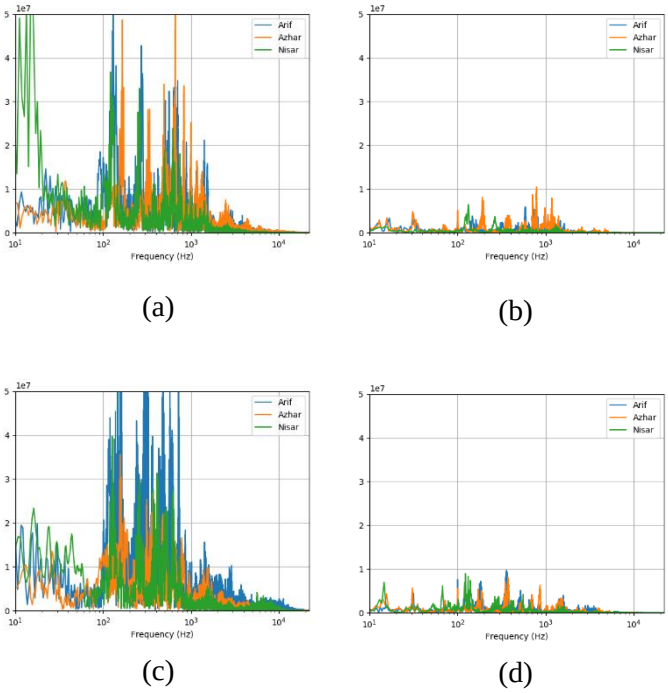
Pengujian prediksi penegenalan suara yang diucapkan, dilakukan pada 9 orang yang berbeda. Pada pengujian ini terdapat 3 orang laki-laki yang suaranya digunakan untuk *leraning*, 3 orang laki-laki-laki dan 3 perempuan yang suaranya tidak digunakan untuk *learning*. Pengujian ini menggunakan data hasil *learnig* dengan fungsi sigmoid, karena model tersebut memiliki tingkat akurasi yang lebih tinggi. Untuk pengujian ini masing-masing orang mengucapkan 7 kata perintah digunakan sebanyak 5 kali percobaan pada 4 jarak yang berbeda.

Tabel 4.1 Frekuensi dan Penguatan Suara Perintah *Corner*

Nama	Frekuensi rata-rata (Hz)	Penguatan Suara (dB)			
		5cm	25cm	50cm	100cm
Arif	237	-22	-12	-12	-12
Azhar	741	-19	-14	-13	-13
Nisar	74	-22	-12	-11	-11
Jauhar	384	-27	-14	-13	-11
Muklis	405	-19	-12	-12	-12
Mukramin	650	-20	-12	-12	-12
Azza	551	-17	-11	-10	-11
Melvy	565	-18	-11	-11	-11
Yani	333	-21	-12	-10	-12

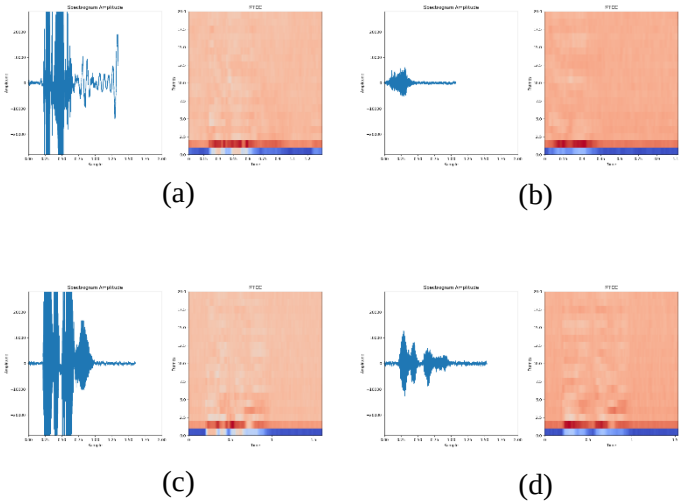
4.4.1 Pengujian Suara *Learning*

Pada pengujian ini suara dari 5 orang yang digunakan untuk *learning* diambil 3 orang untuk menguji sistem pengenalan suara ini mampu mengenali suara. Berbeda dengan data suara pada proses *learning* yang hanya diambil pada satu jarak saja. Pada pengujian ini, suara yang diambil pada 3 orang tersebut dibagi menjadi 4 jarak yang berbeda.



Gambar 4.9 Spektrum Frekuensi Perintah (a)*Corner* 5cm (b)*Corner* 50cm (c)Kalibrasi 5cm (d)Kalibrasi 50cm Data Pengujian Suara *Learning*

Pada Gambar 4.9 terlihat bahwa jarak dapat mempengaruhi frekuensi sinyal yang didapat, semakin jauh jarak antara mikrofon dengan sumber suara maka, spektrum frekuensi suara tersebut akan semakin kecil. Sedangkan untuk hasil dari ekstraksi MFCC pada data pengujian ini dapat dilihat pada Gambar 4.10.



Gambar 4.10 Hasil Ekstraksi MFCC Perintah (a)*Corner* 5cm (b) *Corner* 50cm (c)Kalibrasi 5cm (d)Kalibrasi 50cm Data Pengujian Suara *Learning*

Pada Gambar 4.10 terlihat bahwa ketika jarak semakin jauh, maka sinyal yang ditangkap juga akan semakin mengecil. Namun untuk hasil ekstraksi yang didapat oleh MFCC masih memiliki kemiripan, sehingga masih terdapat kemungkinan untuk mengenalai suara perintah pada jarak yang berbeda. Pada pengujian sistem pengenalan suara, untuk mengenali suara orang yang tidak di *learning*, maka ditambahkan sebuah label untuk orang yang tidak dikenal. Label tersebut akan diberikan kepada suara yang menghasilkan nilai prediksi pengenalan kurang dari nilai ambang batas yang ditentukan. Pada pengujian ini dilakukan pada tiga nilai ambang batas yang berbeda untuk mengetahui niali yang lebih cocok untuk sistem pengenalan usara yang telah dibuat ini.

Tabel 4.2 Hasil Pengujian pada Suara *Learning*

Nama	Jarak (cm)	Keberhasilan (%)			
		Perintah	Orang (0.40)	Orang (0.70)	Orang (0.95)
Arif	5	100	88.6	88.6	71.4
	25	100	68.6	62.9	51.4
	50	97.1	80	74.3	68.6
	100	88.6	48.6	48.6	42.9
Azhar	5	97.1	85.7	82.9	77.1
	25	100	91.4	88.6	88.6
	50	97.1	94.3	94.3	82.9
	100	100	51.4	31.4	2.9
Nisar	5	100	71.4	65.7	48.6
	25	80	100	97.1	91.4
	50	91.4	100	91.4	82.9
	100	80	91.4	91.4	80

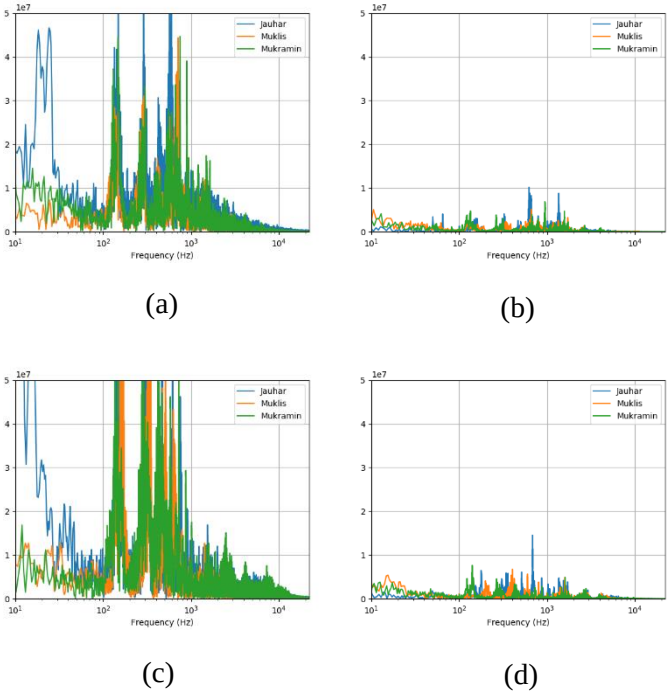
Berdasarkan Tabel 4.2 terlihat bahwa jarak dapat memperngaruhi kemampuan pengenalan dari siistem yang telah dibuat. Semakin dekat sumber suara dengan mikrofon juga dapat mengurangi kemampuan sistem dalam memprediksi suara yang masuk, hal ini terjadi karena jika dibandingkan spektrum frekuensi antara jarak 5cm dan 50cm maka spektrum frekuensi yang lebih mirip yaitu pada jarak 50cm, sehingga untuk pengenalan orang pada jarak 50cm memiliki nilai probabilitas yang lebih stabil dan lebih baik apabila dibandingkan dengan jarak 5cm. Jarak yang memiliki hasil prediksi tertinggi pada pengenalan perintah dan orang yaitu pada jarak 50cm dengan keberhasilan 95% untuk penegenalan perintah dan 78% untuk pengenalan orang dan nilai ambang batas pengenalan orang 0.4.

4.4.2 Pengujian Suara *Non-learning*

Untuk pengujian data suara *non-learning* ini dilakukan pada 3 suara laki-laki dan 3 suara perempuan. Hal ini dilakukan untuk mengetahui seberapa besar kemampuan sistem pengenalan yang dibuat dengan data suara 5 orang laki-laki.

4.4.2.1 Pengujian Suara Laki-laki

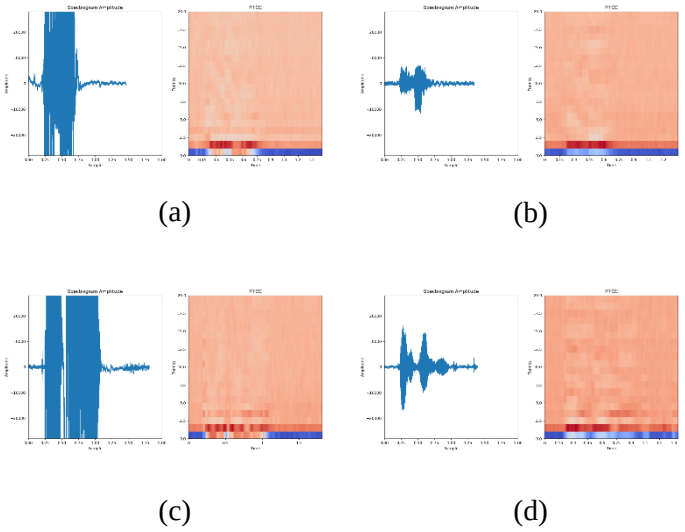
Pada pengujian ini, data yang diambil yaitu suara 3 orang laki-laki yang suaranya tidak pernah di *learning*. Karena suara 3 orang ini tidak pernah di latihkan, maka sistem harus dapat mengenalinya sebagai orang yang tidak dikenal. Spektrum frekuensi dari 3 laki-laki ini dapat dilihat pada Gambar 4.11



Gambar 4.11 Spektrum Frekuensi Perintah (a)*Corner* 5cm (b)*Corner* 50cm (c)Kalibrasi 5cm (d) Kalibrasi 50cm Data Pengujian Suara Laki-laki Non-*learning*

Pada Gambar 4.11 ini terlihat bahwa spektrum frekuensi yang dihasilkan oleh 3 orang laki-laki ini memiliki nilai spektrum yang mirip dengan suara

learning. Untuk hasil pengambilan fitur MFCC dari suara 3 orang laki-laki ini dapat dilihat pada Gambar 4.12



(a) (b)

(c) (d)

Gambar 4.12 Hasil Ekstraksi MFCC Perintah (a)*Corner* 5cm (b) *Corner* 50cm (c)Kalibrasi 5cm (d)Kalibrasi 50cm Data Pengujian Suara Laki-laki Non-*learning*

Dapat dilihat pada Gambar 4.12 dan Gambar 4.10, walaupun suara perintah diucapkan oleh yang berbeda namun hasil ekstraksi fitur MFCC yang didapat masih memiliki kemiripan. Meskipun hasil suara yang terekam tampak berbeda cukup jauh tetapi fitur yang didapat tidak berbeda jauh pada perintah yang sama. Karena fitur yang didapat pada suara 3 orang laki-laki ini memiliki kemiripan dengan suara yang di latihkan, maka suara 3 orang ini kemungkinan besar masih dapar dikenali oleh sistem pengenalan suara ini.

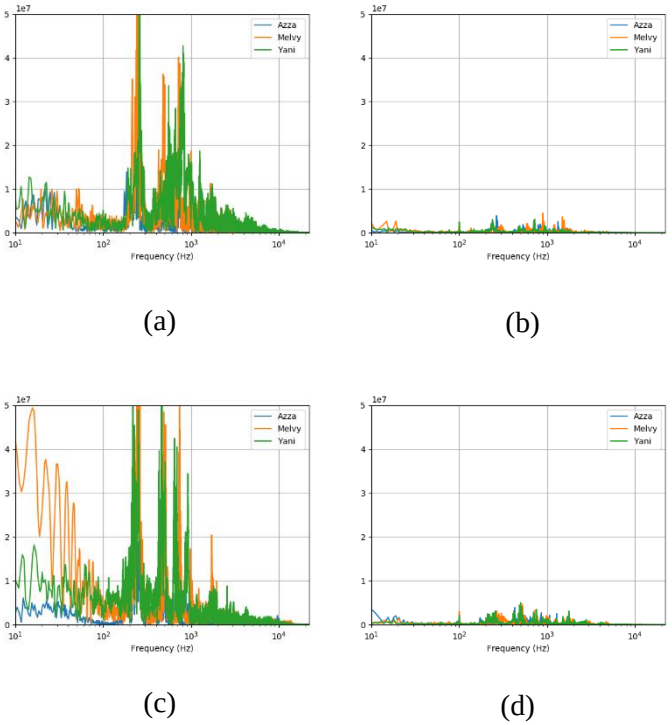
Tabel 4.3 Hasil Pengujian pada Suara Laki-laki *Non-learning*

Nama	Jarak (cm)	Keberhasilan (%)			
		Perintah	Orang (0.40)	Orang (0.70)	Orang (0.95)
Jauhar	5	74.3	31.4	51.4	71.4
	25	100	25.7	40	62.9
	50	97.143	28.6	45.77	62.9
	100	94.3	57.1	65.7	85.7
Muklis	5	100	22.9	57.1	88.5
	25	74.3	14.3	31.4	57.1
	50	88.6	0	20	45.7
	100	57.1	31.4	40	60
Mukramin	5	100	14.2	31.4	68.5
	25	94.3	11.4	22.9	34.3
	50	82.9	31.4	37.1	54.3
	100	71.4	17.1	25.7	62.9

Berdasarkan hasil pengujian yang didapat pada Tabel 4.3 dapat dikatakan bahwa keberhasilan prediksi untuk mengenali perintah memiliki presentase yang hampir sama dengan pengujian pada data yang dilatihkan namun untuk mengenali orang terdapat penurunan keberhasilan. Jadi untuk mengenali suara laki-laki *non-learning* didapatkan presentase keberhasilan tertinggi yaitu 91.4% untuk pengenalan perintah dan 76.2% untuk pengenalan orang pada jarak 100cm dengan nilai ambang batas pengenalan orang 0.95.

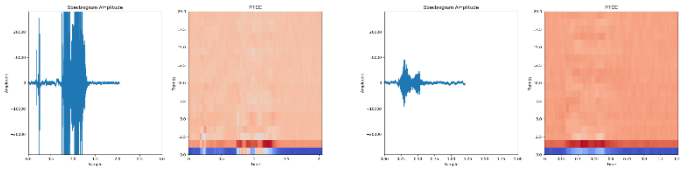
4.4.2.2 Pengujian Suara Perempuan

Pada pengujian dengan suara perempuan ini, suara yang digunakan adalah 3 suara perempuan berbeda yang suaranya tidak pernah dialtihkan, sedangkan data yang dilatihkan hanya suara 5 orang laki-laki. Spektrum Frekuensi dari suara 3 orang perempuan tersebut dapat dilihat pada Gambar 4.13

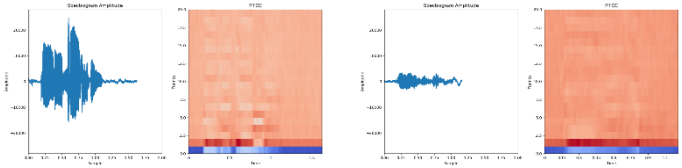


Gambar 4.13 Spektrum Frekuensi Perintah (a)*Corner* 5cm (b)*Corner* 50cm (c)Kalibrasi 5cm (d) Kalibrasi 50cm Data Pengujian Suara Perempuan Non-*learning*

Setiap suara perempuan memiliki pola spektrum frekuensinya sendiri. Apabila dibandingkan dengan spektrum frekuensi suara laki-laki pada Gambar 4.11, spektrum fekuensi pada perempuan relatif lebih kecil, semakin jauh jarak sumber suara, maka spektrum frekuensi akan tampak sangat kecil sekali dan kemungkinan berhasil dalam mengenali suara lebih kecil karena pada data yang dilatihkan hanya ada suara laki-laki dengan jarak yang dekat.



(a) (b)



(c) (d)

Gambar 4.14 Hasil Ekstraksi MFCC Perintah (a)*Corner* 5cm (b) *Corner* 50cm (c)Kalibrasi 5cm (d)Kalibrasi 50cm Data Pengujian Suara Perempuan *Non-learning*

Gambar 4.14 menunjukan hasil ekstraksi fitur MFCC pada suara perempuan yang dijadikan sebagai pengujian. Berdasarakan gambar tersebut dapat dilihat bahwa hasil ektraksi fitur pada suara perempuan memiliki perbedaan apabila dibandingkan dengan suara yang digunakan pada latihan, dapat dilihat pada Gambar 4.5. Sinyal suara yang ditangkap oleh mikrofon pada suara perempuan memiliki nilai amplitudo yang relatif lebih kecil dibandingkan pada suara laki-laki.

Tabel 4.4 Hasil Pengujian pada Suara Perempuan *Non-learning*

Nama	Jarak (cm)	Keberhasilan (%)			
		Perintah	Orang (0.40)	Orang (0.70)	Orang (0.95)
Azza	5	68.6	28.6	45.7	68.6
	25	57.1	42.9	62.9	77.1
	50	51.4	31.4	57.1	82.9
	100	45.7	22.9	34.3	48.6
Melvy	5	40	25.7	51.4	80
	25	42.9	48.9	74.3	91.4
	50	51.4	31.4	48.6	80
	100	54.3	25.7	48.6	65.7
Yani	5	82.9	28.6	42.9	71.4
	25	60	31.4	42.9	77.1
	50	65.7	28.6	42.9	68.6
	100	57.143	28.6	45.7	68.6

Tabel 4.4 menunjukan hasil dari pengujian sistem pengenalan suara yang dibuat menggunakan suara perempuan *non-learning*. Dari hasil yang didapatkan dapat dikatakan kemampuan mengenali suara pada suara perempuan sangat lemah, dilihat dari presentase keberhasilan yang didapat bila dibandingkan dengan presentase keberhasilan pengenalan suara laki-laki lebih tinggi. Hal ini terjadi karena didalam sistem pengenalan yang dibuat tidak ada sama sekali data suara seorang perempuan. Apabila dilihat pada spektrum frekuensi dan ekstraksi fitur MFCC dapat dilihat bahwa perbedaan pada suara perempuan ini lebih terlihat dibandingkan pada suara laki-laki. Jadi hasil terbaik yang didapat pada pengujian ini yaitu 63.8% untuk pengenalan perintah dan 73.3% untuk pengenalan orang pada jarak 100cm dengan nilai ambang batas pengenalan orang 0.95.

BAB 5
PENUTUP

5.1 Kesimpulan

Jadi kesimpulan yang dapat penulis ambil dari serangkaian pembuatan penelitian tugas akhir ini, didapatkan beberapa hasil kesimpulan sebagai berikut:

1. Pengambilan data menggunakan *voice activity detector* (VAD) dengan metode ambang batas kurang efektif untuk orang yang memiliki suara yang bernada rendah. Sehingga kata yang diucapkan terkadang tidak terekam secara keseluruhan dan membuat prediksi dari sistem menjadi salah
2. Pada prose *learning* data yang menggunakan lebih dari satu label untuk klasifikasi, maka *leraning* model menggunakan fungsi aktivasi sigmoid lebih efektif dan memberikan hasil akurasi yang lebih tinggi yaitu sekitar 99%. Sedangkan menggunakan aktivasi softmax hanya dapat memberikan akurasi sekitar 85%.
3. Sistem pengenalan perintah suara yang dibuat ini memiliki akurasi yang cukup tinggi untuk mengenali jenis kata perintah yang diucapkan. Dari hasil pengujian, didapat nilai akurasi rata-rata terbesar 95% untuk suara orang dilatihkan, 91% untuk suara laki-laki dan 64% untuk suara perempuan yang tidak dilatihkan.
4. Untuk penegenalan orang yang memberikan perintah didapat rata-rata akurasi sekitar 91% untuk suara orang yang dilatihkan, sedangkan untuk mengenali suara laki-laki 76% dan perempuan 73% yang tidak dilatihkan.

5.2 Saran

Berikut adalah saran beberapa saran yang diberikan oleh penulis agar sistem pengenalan suara pada penelitian tugas akhir ini dapat dikembangkan menjadi sistem yang lebih baik:

1. Metode ambang batas dari *voice activity detector* (VAD) untuk pengambillan data suara dapat ditambah dengan sistem penguatan suara secara otomatis, sehingga data suara tetap dapat terekam meskipun suara tersebut memiliki nada yang rendah.

2. Agar kemampuan pengenalan suara meningkat untuk suara perempuan sebaiknya terdapat suara perempuan yang dimasukan proses *learning*.
3. Fitur suara yang digunakan untuk *learning* dapat ditambahkan atau diganti untuk meningkatkan kemampuan pengenalan suara pemberi perintah, sehingga karakteristik suara setiap orang dapat terlihat lebih jelas.

DAFTAR PUSTAKA

[1] Heru, E. Pitowarno, and K. Mutijarsa, “Buku Panduan Kontes Robot Sepakbola Indonesia Beroda,” Dec. 2018.

[2] M. Asada *et al.*, “MSL Technical Committee 1997–2019,” p. 107, Dec. 2018.

[3] D. T. Wibowo, “Perancangan Alat Automatic Voice Control Untuk Radio FM,” 2017.

[4] L. Sialuzan, “Transmitter Pada Microphonewireless Dengan Tampilan Seven Segment Sebagai Indikator Keluaran,” 2014.

[5] W. Q. Ong and A. W. C. Tan, “Robust voice activity detection using gammatone filtering and entropy,” in *2016 International Conference on Robotics, Automation and Sciences (ICORAS)*, Melaka, Malaysia, 2016, pp. 1–5.

[6] G. Meoni, L. Pilato, and L. Fanucci, “A low power Voice Activity Detector for portable applications,” in *2018 14th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Prague, 2018, pp. 41–44.

[7] T. Nasution, “Metoda Mel Frequency Cepstrum Coefficients (MFCC) untuk Mengenali Ucapan pada Bahasa Indonesia,” vol. 1, no. 1, p. 10, 2012.

[8] M. Bezoui, A. Elmoutaouakkil, and A. Beni-hssane, “Feature extraction of some Quranic recitation using Mel-Frequency Cepstral Coeficients (MFCC),” in *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, Marrakech, Morocco, 2016, pp. 127–131.

[9] A. Winursito, R. Hidayat, A. Bejo, and M. N. Y. Utomo, “Feature Data Reduction of MFCC Using PCA and SVD in Speech Recognition System,” in *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, Shah Alam, 2018, pp. 1–6.

[10] W. S. Eka Putra, “Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101,” *J. Tek. ITS*, vol. 5, no. 1, Mar. 2016.

[11] E. N. Arrofiqoh and H. Harintaka, “Implementasi Metode Convolutional Neural Network Untuk Klasifikasi Tanaman Pada Citra Resolusi Tinggi,” *GEOMATIKA*, vol. 24, no. 2, p. 61, Nov. 2018.

[12] C. K. Dewa, A. L. Fadhilah, and A. Afiahayati, “Convolutional Neural Networks for Handwritten Javanese Character Recognition,” *IJCCS Indones. J. Comput. Cybern. Syst.*, vol. 12, no. 1, p. 83, Jan. 2018.

[13] T. Phaladisailoed and T. Numnonda, “Machine Learning Models Comparison for Bitcoin Price Prediction,” in *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Kuta, 2018, pp. 506–511.

[14] T. Carneiro, R. V. Medeiros Da Nobrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, “Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications,” *IEEE Access*, vol. 6, pp. 61677–61685, 2018.

[15] M. B. Ahmad, Saifullah, M. A. Raja, M. W. Asif, and K. Khurshid, “i-Riter: Machine learning based novel eye tracking and calibration,” in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, Houston, TX, USA, 2018, pp. 1–5.

[16] Y. Mardiana and J. Sahputra, “Analisa Performansi Protokol TCP, UDP dan SCTP Pada Lalu Lintas Multimedia,” vol. 13, no. 2, p. 12, 2017.

[17] D. Hadiyuwono and Pambudi, “User Datagram Protocol (UDP),” 2011.

[18] R. Mishra and A. Javed, “ROS based service robot platform,” in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, Auckland, 2018, pp. 55–59.

[19] A. Jalil, “Robot Operating System (ROS) Dan Gazebo Sebagai Media Pembelajaran Robot Interaktif,” *Ilk. J. Ilm.*, vol. 10, no. 3, p. 284, Dec. 2018.

[20] M. Agus, S. W. Widyanto, S. Wisnugroho, and S. Asuhadi, “Automatci Identification System (AIS) Bebasis Mikrokontroler Untuk Pengawasan Nelayan Di Wakatobi,” p. 7.

[21] A. Darmawan, “Sistem Pembacaan Bola Menggunakan Omnidirectional Camera Untuk Pergerakan Robot Sepak Bola Beroda,” Oct. 2018.

[22] Feng Zhu and Liancheng Su, “Omnidirectional Depth Estimation by a Single Perspective Camera,” in *2006 6th World Congress on Intelligent Control and Automation*, Dalian, China, 2006, pp. 9854–9857.

[23] W. S. Pambudi, “Rancang Bangun 3 Wheels Omni-directional Mobile Robot Menggunakan Sensor Position Sensitive Device (PSD) Serta Sensor Vision Dengan Metode Kendali Fuzzy Logic Controller (FLC) Untuk Menghindari Halangan,” p. 14, 2011.

[24] D. Irawan, “Rancang Bangun Prototipe Lift Barang Menggunakan Motor Arus Searah Dengan Perintah Smartphone Android,” 2016.

[25] L. Zi-Yi, W. Sew-Kin, P. Wai-Leong, and O. Chee-Pun, “The design of DC motor driver for solar tracking applications,” in *2012 10th IEEE International Conference on Semiconductor Electronics (ICSE)*, Kuala Lumpur, Malaysia, 2012, pp. 556–559.

.....Halaman ini sengaja dikosongkan.....

LAMPIRAN

Program *Learning*

```
import librosa
import os
from sklearn.model_selection import
train_test_split
from sklearn.preprocessing import
MultiLabelBinarizer
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout,
Flatten, Conv2D, MaxPooling2D
from keras.utils import to_categorical
import matplotlib.pyplot as plt

DATA_PATH =
'/content/godrive/data_train/data_train_baru/'

def wav2mfcc(file_path, max_len=11):
    wave, sr = librosa.load(file_path,
                mono=True, sr=None)
    mfcc = librosa.feature.mfcc(wave, sr=44100,
           n_mfcc=20, hop_length= 64)
    if (max_len > mfcc.shape[1]):
        pad_width = max_len - mfcc.shape[1]
        mfcc = np.pad(mfcc, pad_width=((0, 0),
                (0, pad_width)), mode='constant')
    else:
        mfcc = mfcc[:, :max_len]
    return mfcc

def get_labels(path=DATA_PATH):
    labels = os.listdir(path)
    return labels

def save_data_to_array(path=DATA_PATH,
                       max_len=11):
```

```
    labels = get_labels(path)
    for label in labels:
        mfcc_vectors = []
        wavfiles = [path + label + '/' + wavfile
                    for wavfile in os.listdir
                    (path + '/' + label)]
        for wavfile in wavfiles:
            mfcc = wav2mfcc(wavfile, max_len=
                            max_len)
            mfcc_vectors.append(mfcc)
        np.save(label + '.npy', mfcc_vectors)

def get_train_test(split_ratio=0.80,
                   random_state=35):
    labels = get_labels(DATA_PATH)
    label_all = []
    for label in labels:
        data_label = label.split("_")
        label_all.append(data_label)
    mlb = MultiLabelBinarizer()
    mlb.fit(label_all)
    label_data = mlb.classes_
    print(label_data)
    X = np.load(labels[0] + '.npy')
    y = mlb.transform([labels[0].split("_")])
    for i in range(len(X)-1):
        y = np.vstack((y, mlb.transform(
            [labels[0].split("_")])))
    for label in labels[1:]:
        x = np.load(label + '.npy')
        X = np.vstack((X, x))
        for i in range(len(x)):
            y = np.vstack((y, mlb.transform(
                [label.split("_")])))
    assert X.shape[0] == len(y)
    return train_test_split(X, y, test_size=(1 -
           split_ratio), random_state =
           random_state, shuffle=True)
```

```
save_data_to_array(path=DATA_PATH, max_len=820)
X_train,X_test,y_train,y_test= get_train_test()
X_train=X_train.reshape(X_train.shape[0],20,820,
                        1)
X_test=X_test.reshape(X_test.shape[0],20,820,1)

model = Sequential()
model.add(Conv2D(32, kernel_size=(4, 4),
activation='relu', input_shape=(20, 820, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(32, kernel_size=(2, 2),
          activation='relu'))
model.add(MaxPooling2D(pool_size=(1, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(12, activation='sigmoid'))

model.compile(loss=keras.losses.binary_crossentr
opy,optimizer=keras.optimizers.Adam(lr=0.00005),
metrics=['accuracy'])

print(model.summary())
history = model.fit(X_train, y_train,batch_size=
          None,steps_per_epoch=50,epochs=50,
          verbose=1,validation_data=
          (X_test, y_test),validation_steps=100)

model.save('/content/godrive/data_train/
           Save_Model/model.h5')

print(history.history.keys())
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
```

```
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

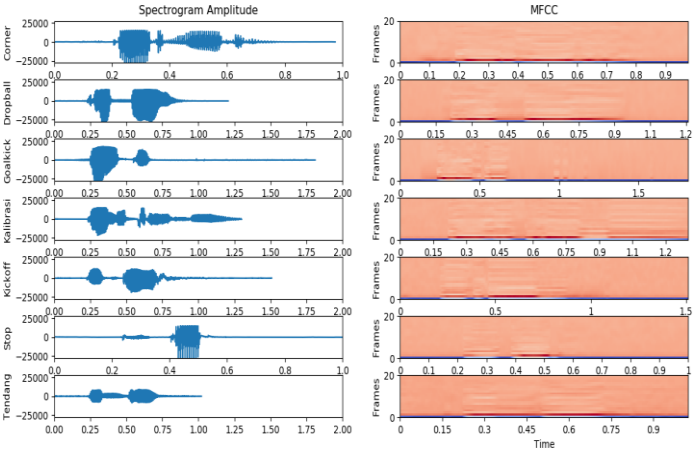
Program Pengujian

```
import os
import librosa
import numpy as np
import keras
from keras.models import Sequential
from keras.models import load_model

def wav2mfcc(file_path, max_len=11):
    wave, sr = librosa.load(file_path,
               mono=True, sr=None)
    mfcc = librosa.feature.mfcc(wave, sr=44100,
           n_mfcc=20, hop_length= 64)
    if (max_len > mfcc.shape[1]):
        pad_width = max_len - mfcc.shape[1]
        mfcc = np.pad(mfcc, pad_width=((0, 0),
               (0, pad_width)), mode='constant')
    else:
        mfcc = mfcc[:, :max_len]
    return mfcc

speech=['Corner','Dropball','Goalkick',
        'Kalibrasi','Kickoff','Stop','Tendang']
speaker=['Arif','Azhar','Habib','Nisar','Revo']

model=Sequential()
model=load_model('Model_Hasil_Train/
                  multilabel_new_V7.h5')
path = 'data_record/BARU/'
nama = os.listdir(path)
i = 0
j = 0
hasil_perintah = [0,0,0,0,0]
hasil_orang = [0,0,0,0,0]

for namas in nama:
    jaraks = [path + namas + '/' + jarak for
jarak in os.listdir(path + '/' + namas)]
    for jarak in jaraks:
```

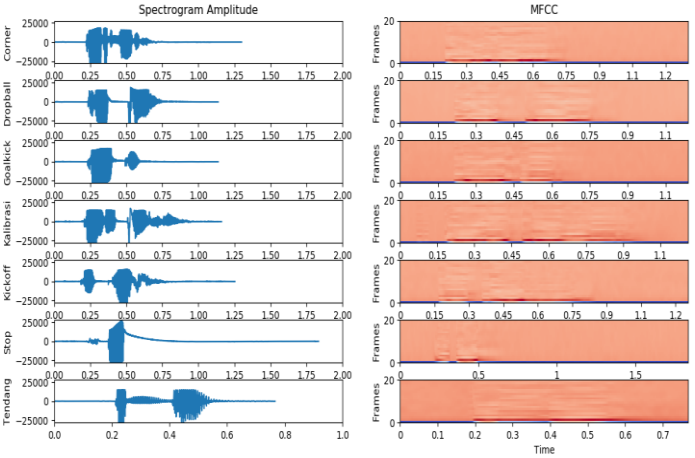
```
rekamans=[jarak + '/' + rekaman for
          rekaman in os.listdir(jarak)]
for rekaman in rekamans:
    files=[rekaman + '/' + file for
           file in os.listdir(rekaman)]
    for file in files:
        perintah = speech[i]
        if namas=='Jauhar' or
           namas=='Muklis' or
           namas=='Mukramin' or
           namas=='Azza'or
           namas=='Yani' or
           namas=='Melvy':
            namas = 'unknown'
        sample=wav2mfcc(file,max_len=
        820)
        sample_reshaped=sample.reshape
        (1, 20, 820, 1)
        prediksi=model.predict
        (sample_reshaped)
        prediksi_speech=prediksi[:,
        np.r_[2:5, 6:8, 10:12]]
        prediksi_speaker = prediksi[:,
        np.r_[0:2, 5, 8:10]]
        data_speech=speech[np.argmax
        (prediksi_speech)]
        data_speech_raw=prediksi_speech
        [:, np.argmax(prediksi_speech)]
        data_speaker=speaker[np.argmax
        (prediksi_speaker)]
        data_speaker_raw =
        prediksi_speaker[:,np.argmax
        (prediksi_speaker)]
        if data_speaker_raw < 0.95:
            data_speaker = 'unknown'
```

Sinyal Suara dan Fitur MFCC *Learning*

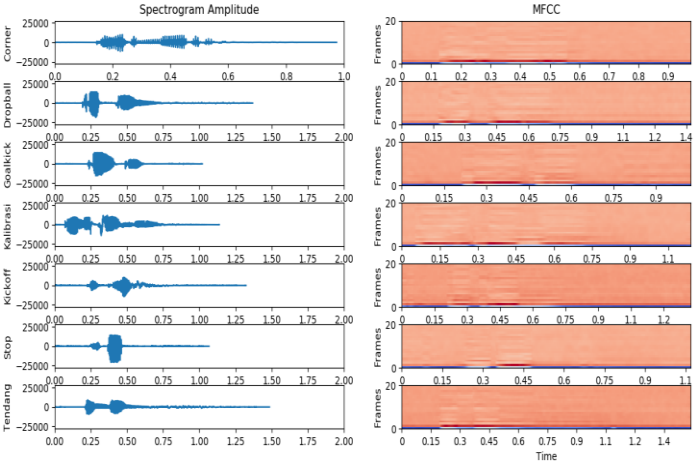
1. Arif



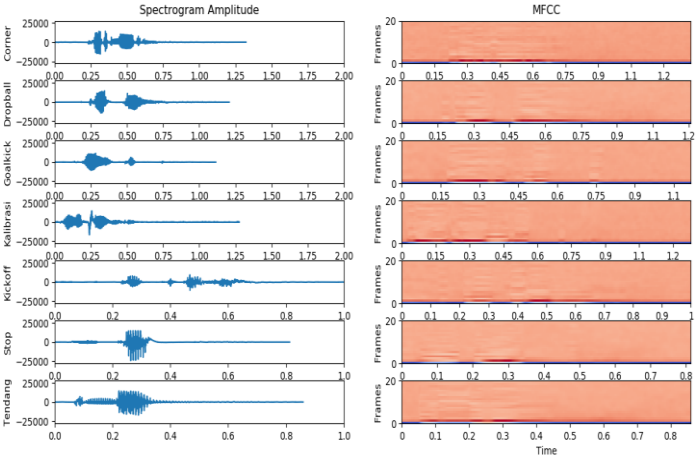
2. Azhar



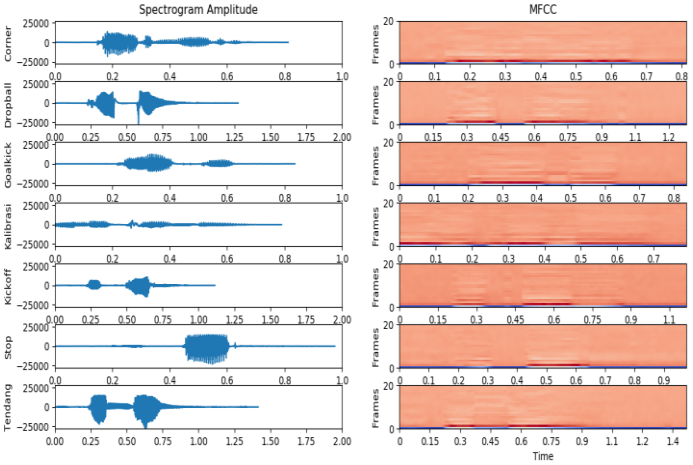
3. Habib



4. Nisar

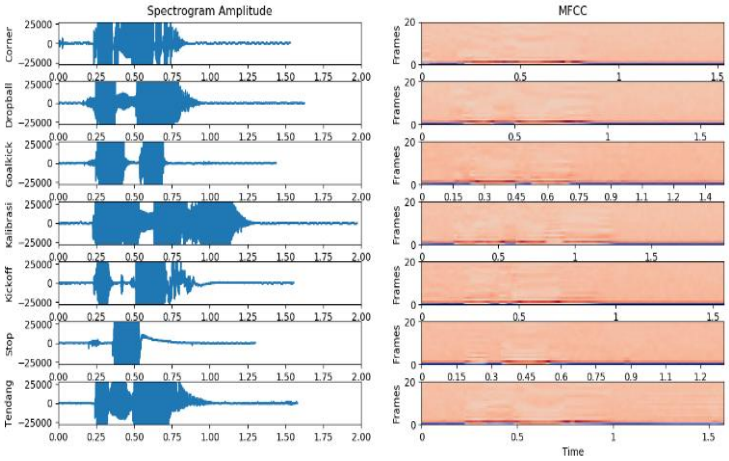


5. Revo

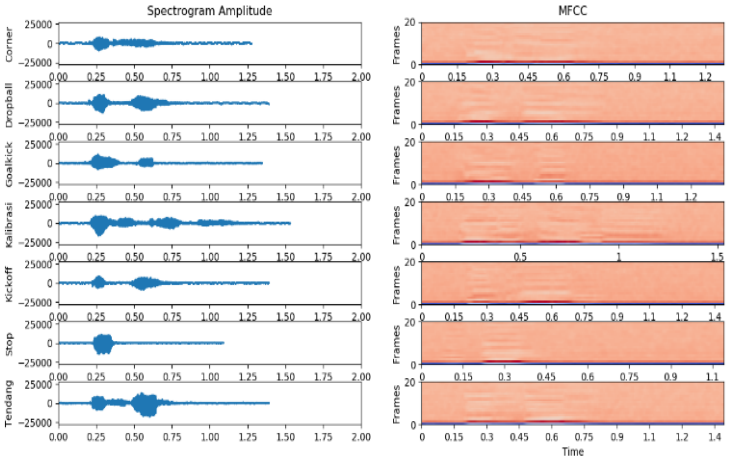

Spectrogram Amplitude
MFCC
Corner
Dropball
Goalkick
Kalibrasi
Kickoff
Stop
Tendang
Frames
Time

Sinyal Suara dan Fitur MFCC Pengujian

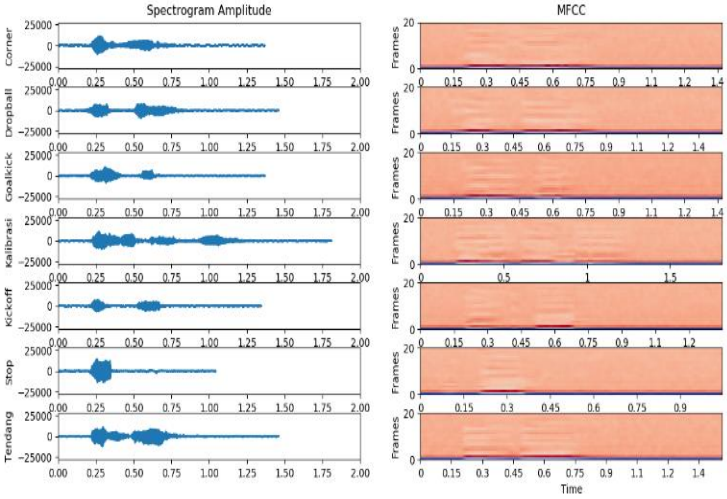
1. Arif (5cm)



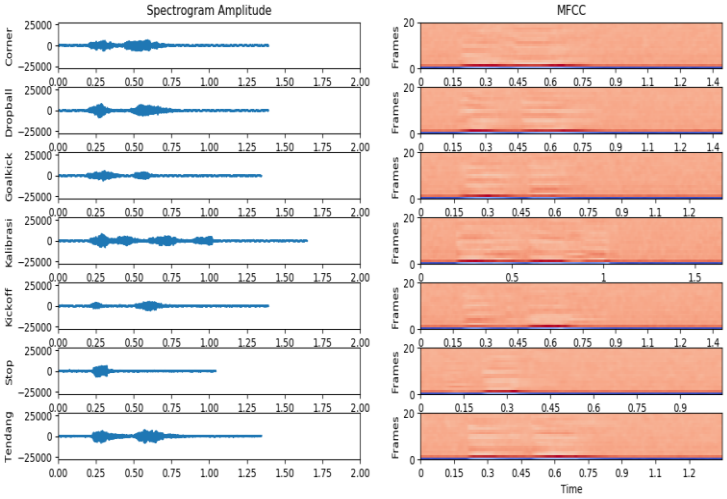
2. Arif (25cm)



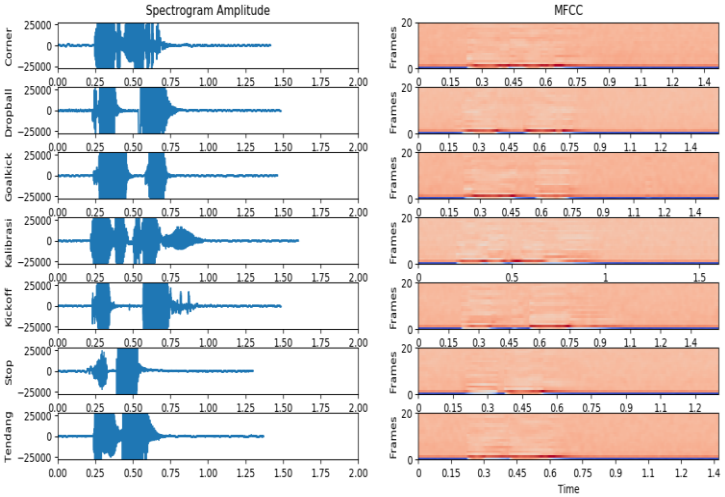
3. Arif (50cm)



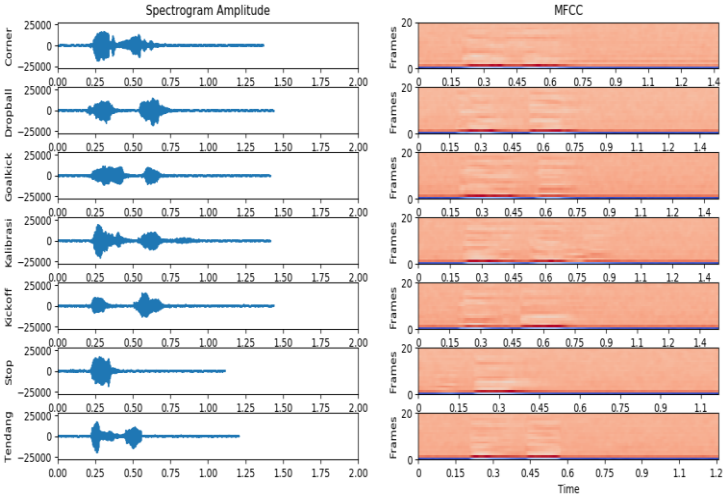
4. Arif (100cm)



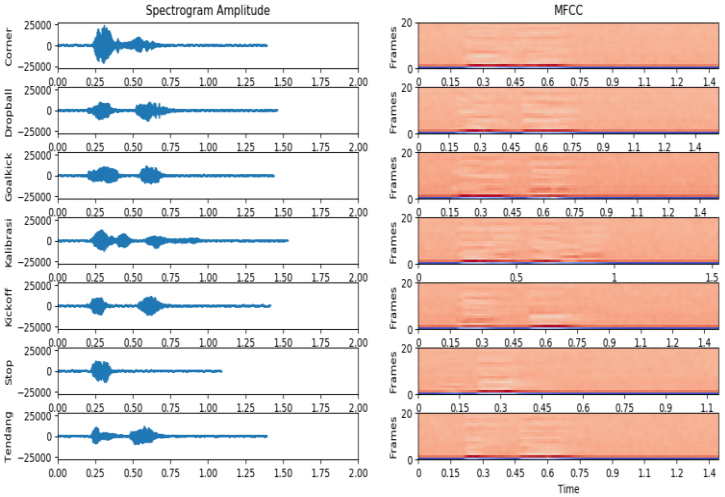
5. Azhar (5cm)



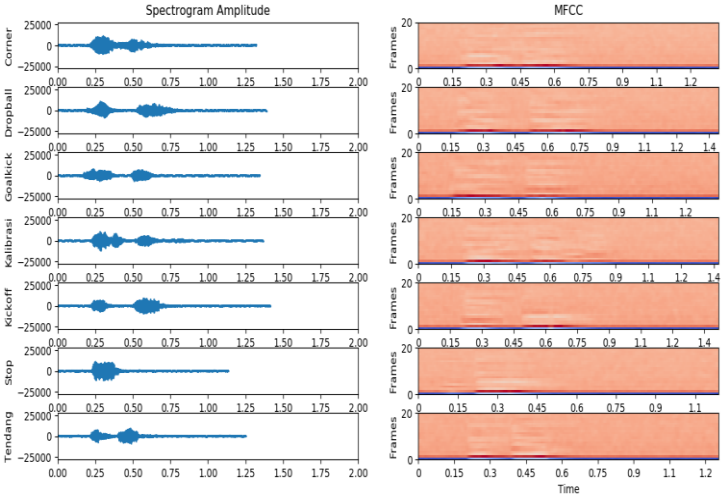
6. Azhar (25cm)



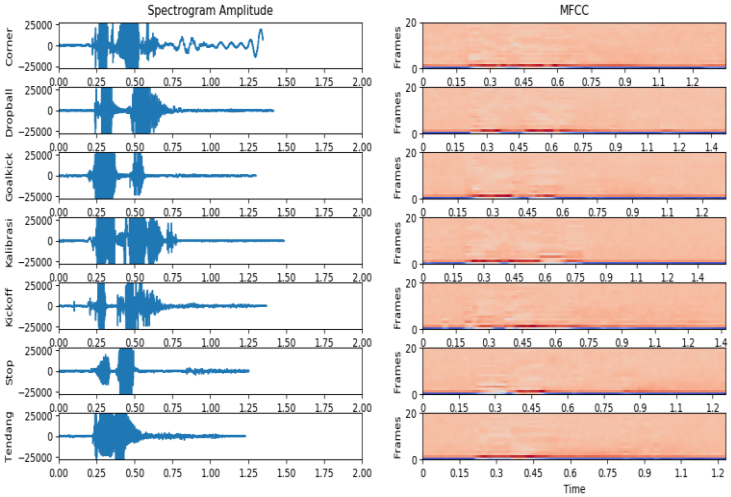
7. Azhar (50cm)



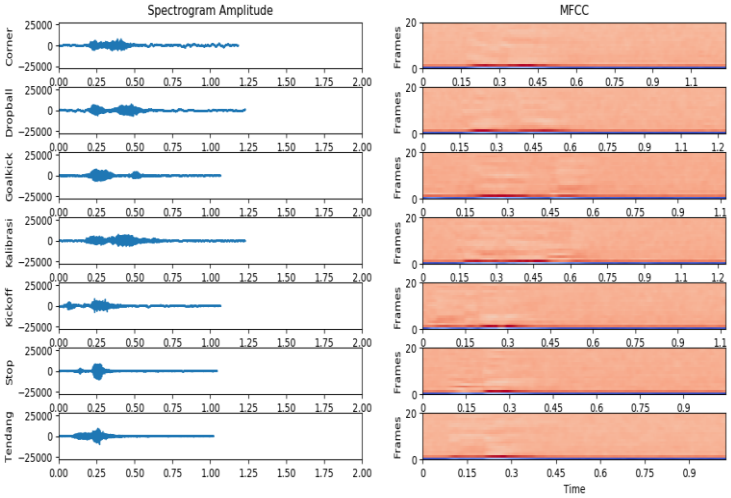
8. Azhar (100cm)



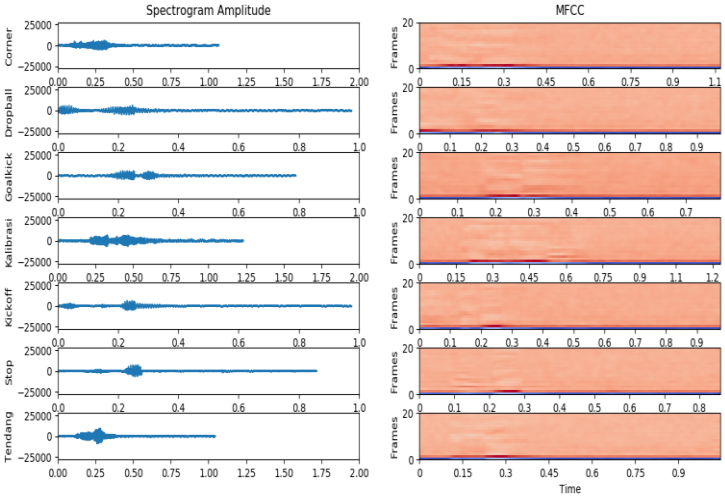
9. Nisar (5cm)



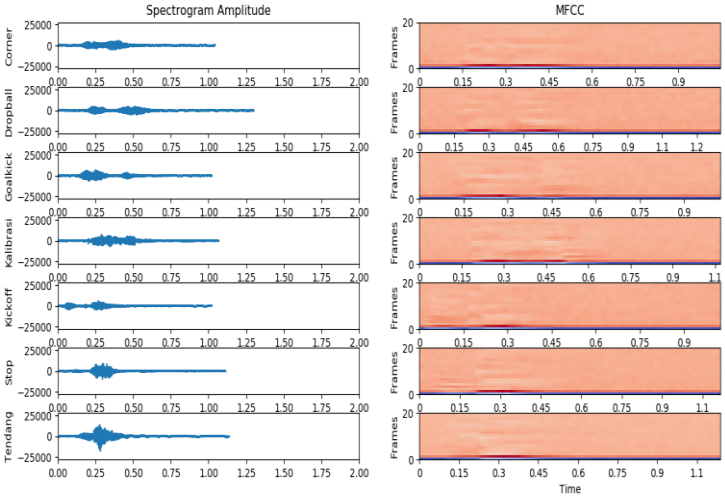
10. Nisar (25cm)



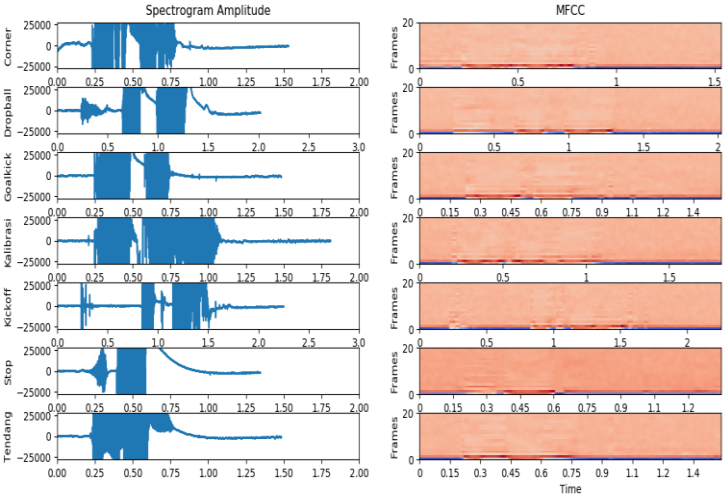
11. Nisar (50cm)



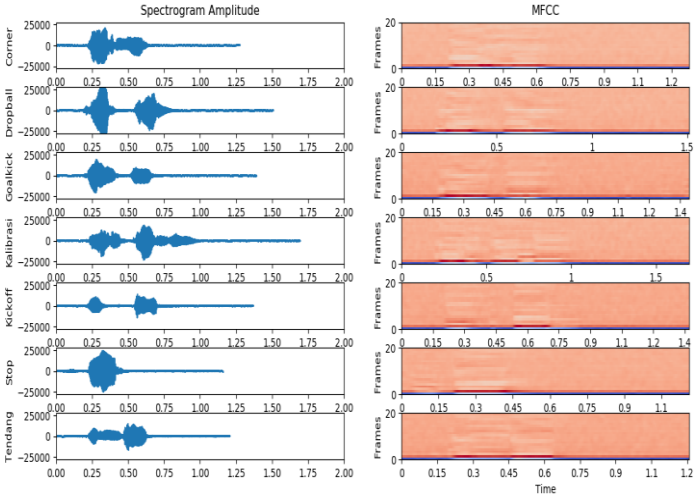
12. Nisar (100cm)



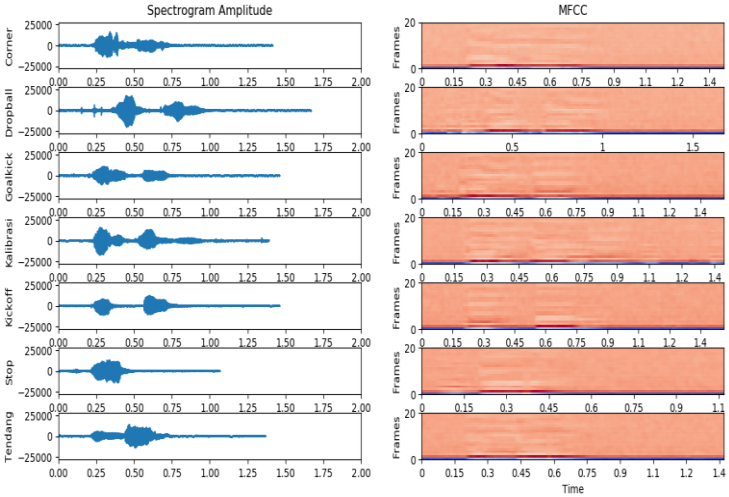
13. Jauhar (5cm)



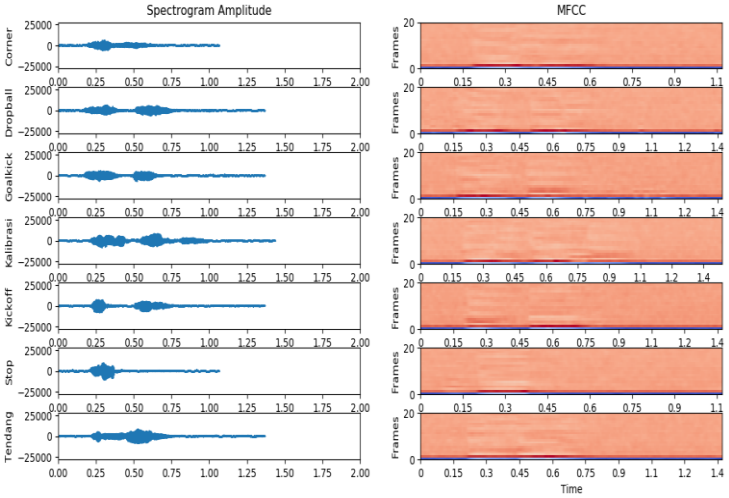
14. Jauhar (25cm)



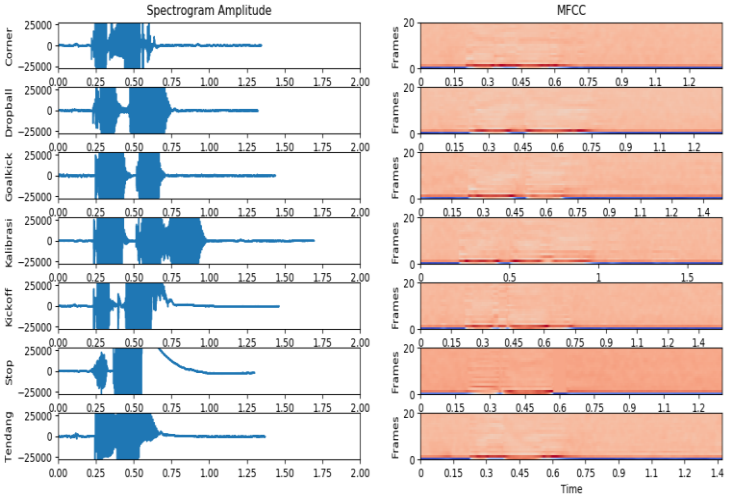
15. Jauhar (50cm)



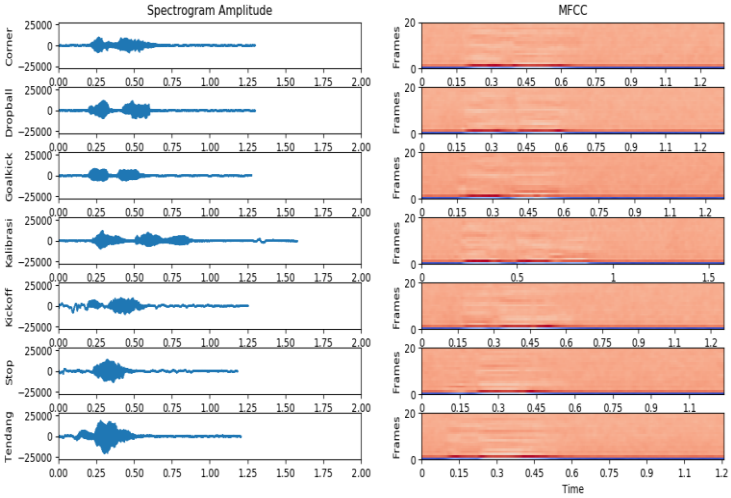
16. Jauhar (100cm)



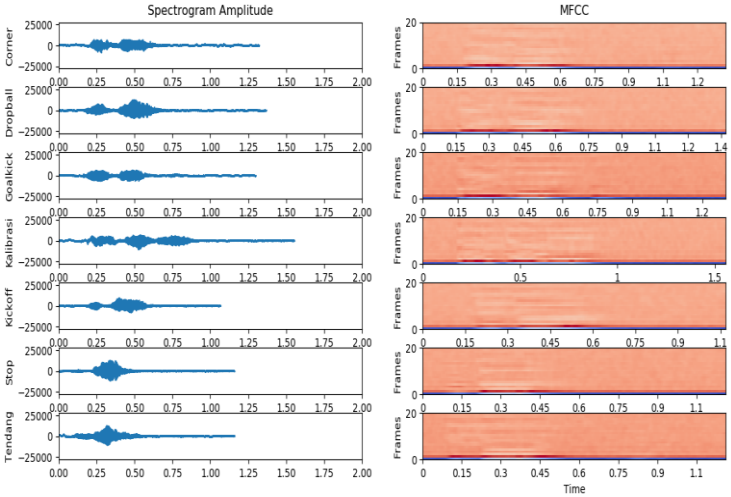
17. Muklis (5cm)



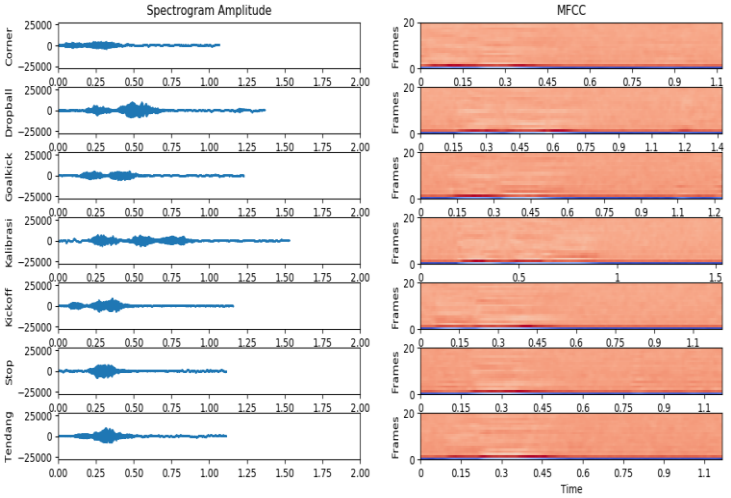
18. Muklis (25cm)



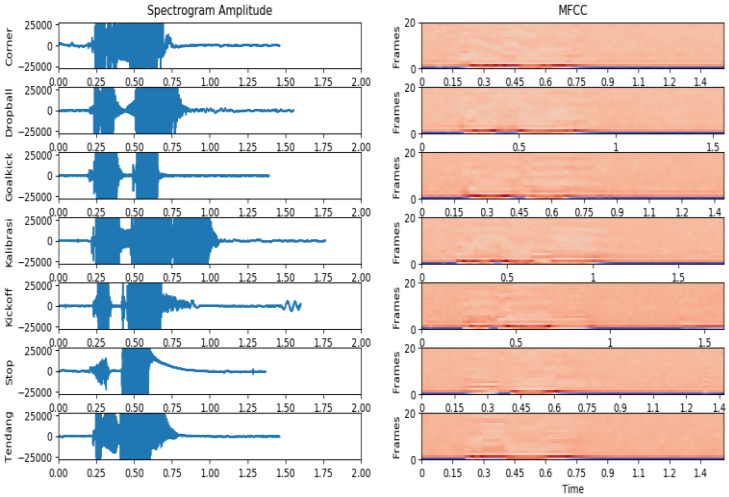
19. Muklis (50cm)



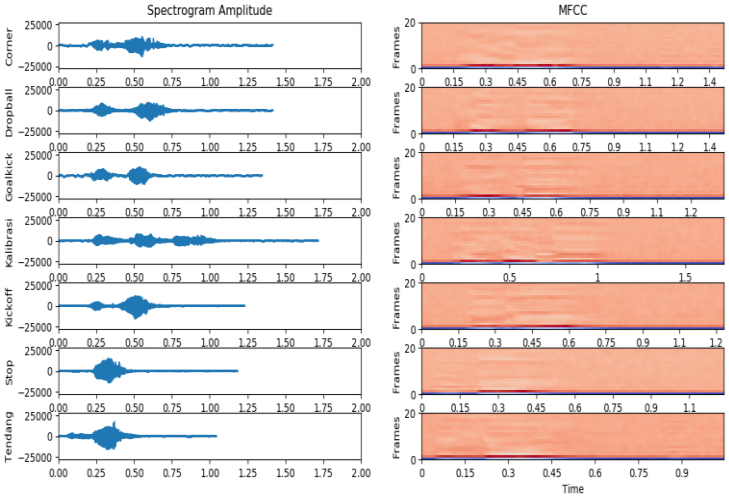
20. Muklis (100cm)



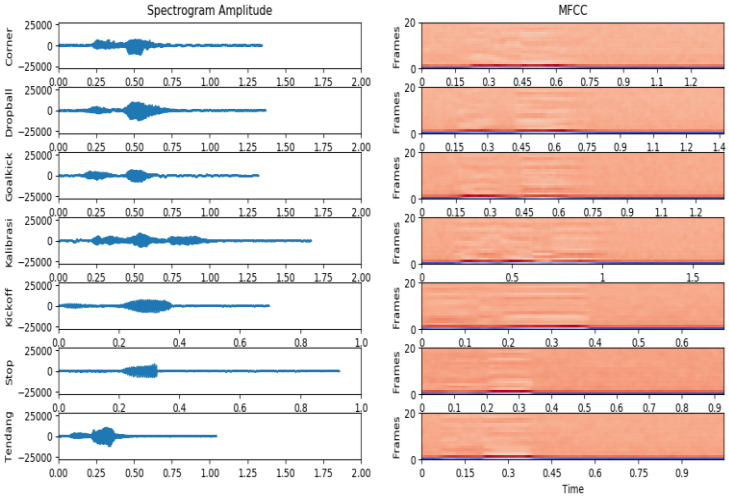
21. Mukramin (5cm)



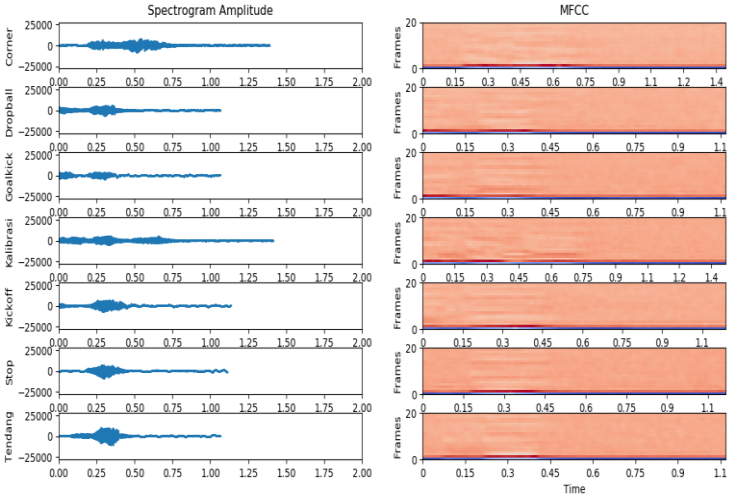
22. Mukramin (25cm)



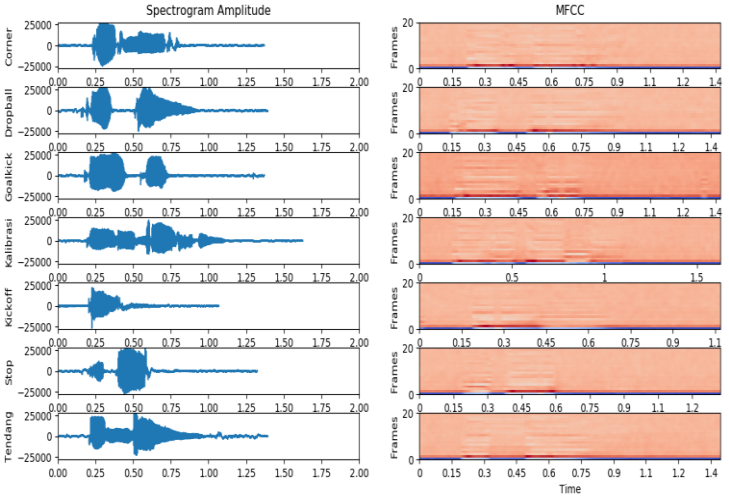
23. Mukramin (50cm)



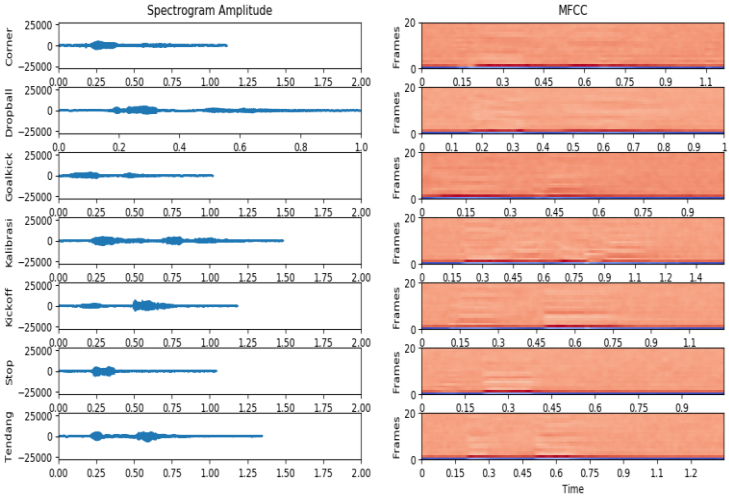
24. Mukramin (100cm)



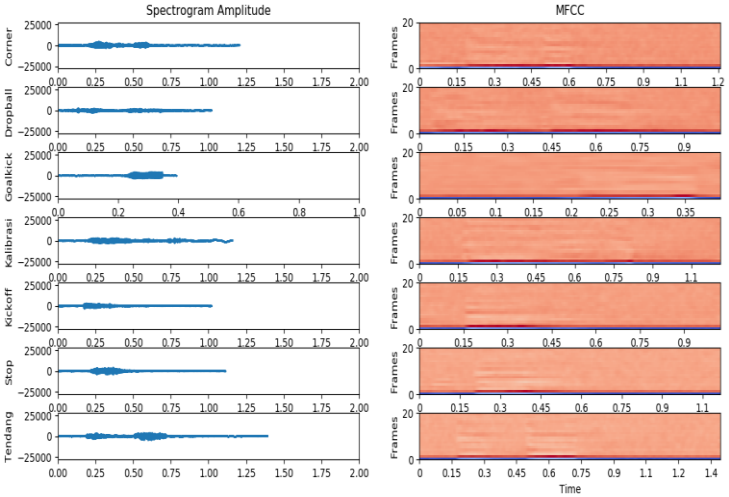
25. Azza (5cm)



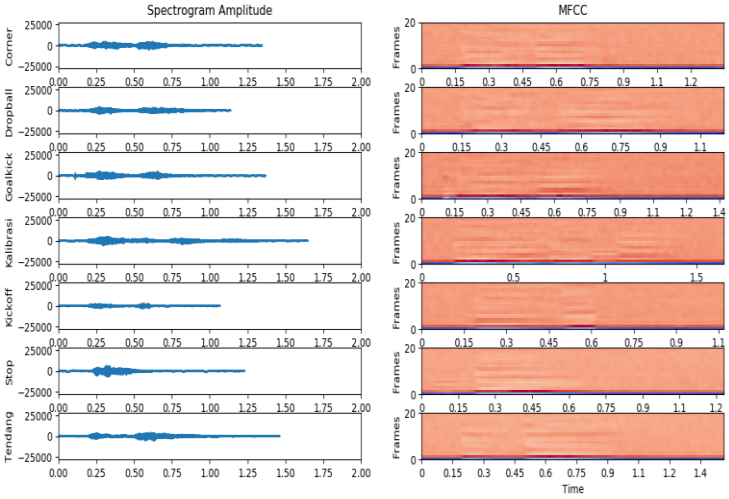
26. Azza (25cm)



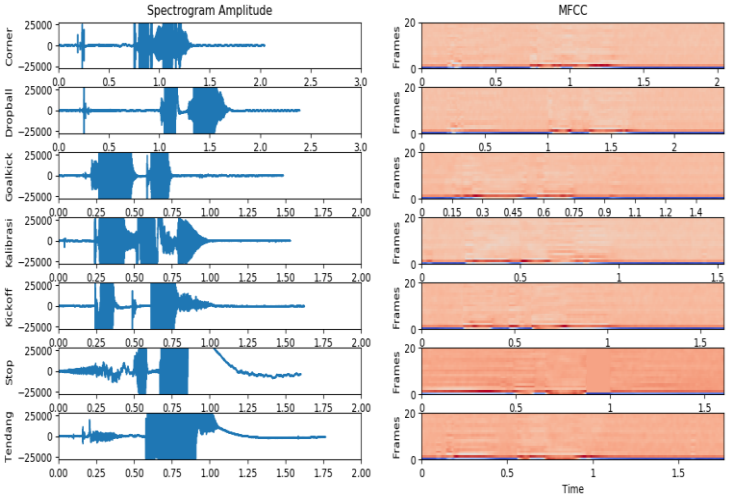
27. Azza (50cm)



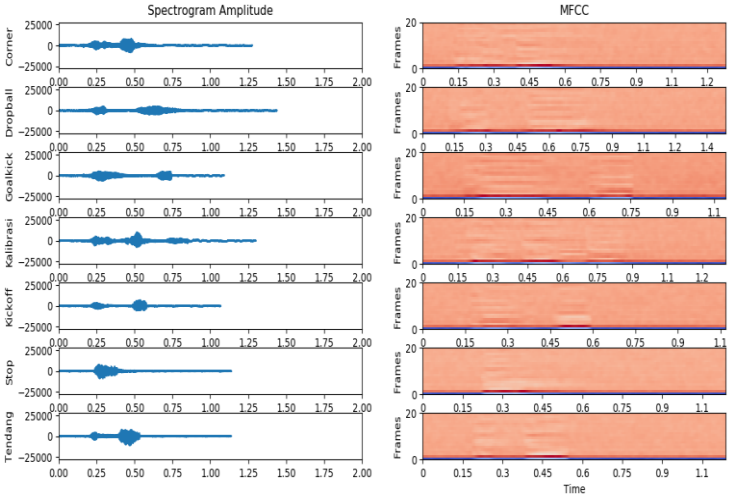
28. Azza (100cm)



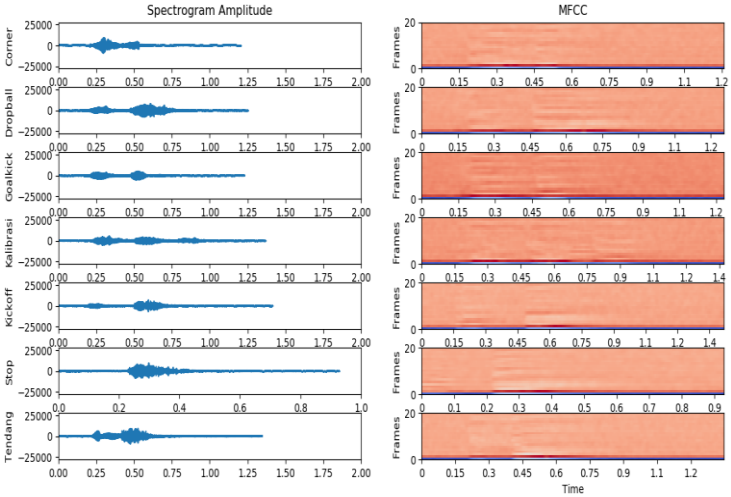
29. Melvy (5cm)



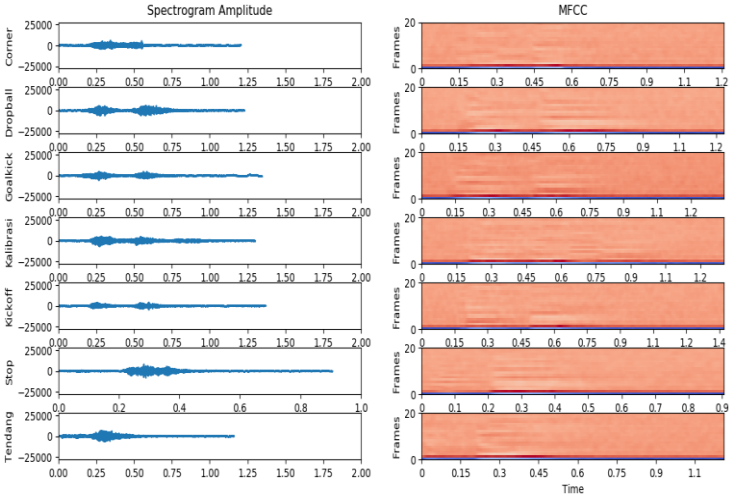
30. Melvy (25cm)



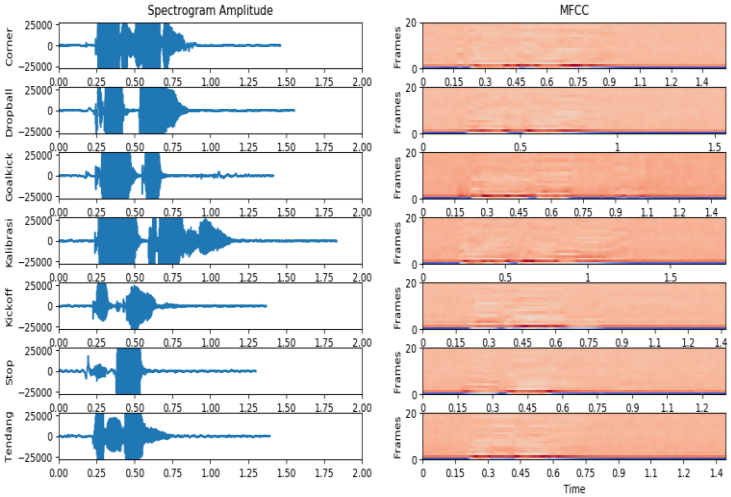
31. Melvy (50cm)



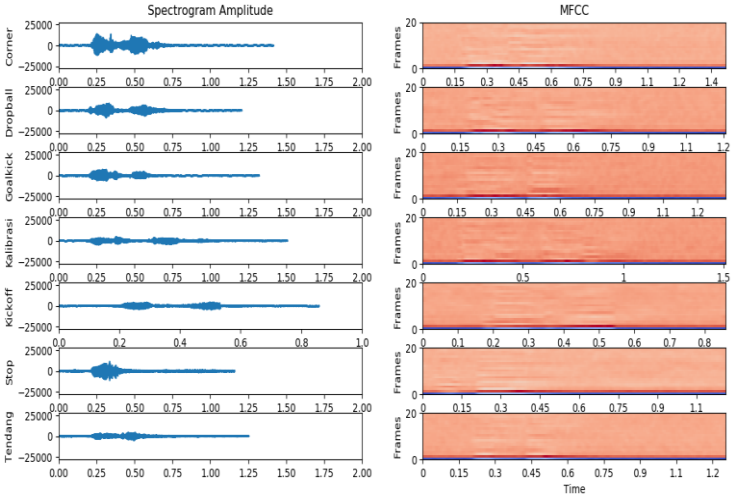
32. Melvy (100cm)



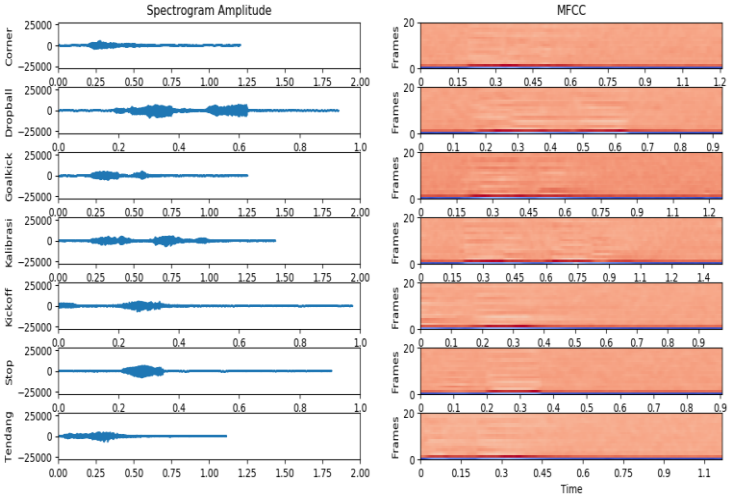
33. Yani (5cm)



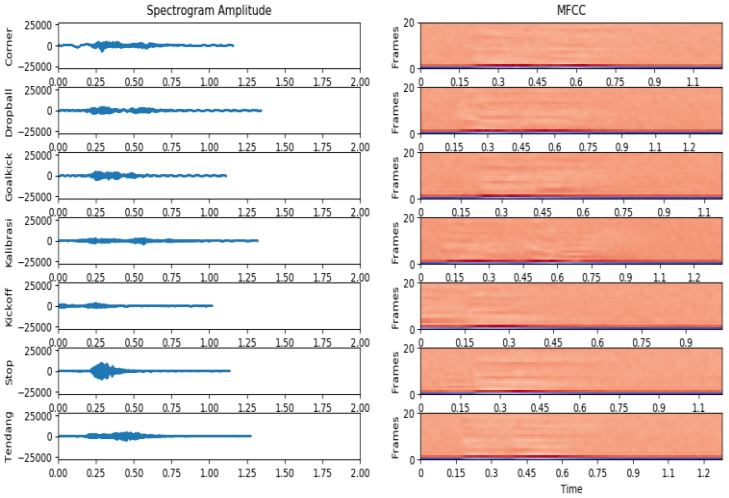
34. Yani (25cm)



35. Yani (50cm)



36. Yani (100cm)



.....Halaman ini sengaja dikosongkan.....

BIODATA PENULIS

Penulis lahir di Kota Tuban pada tanggal 15 April 1998. Penulis mulai menjalani pendidikan dasar di SDN Latsari pada tahun 2004 dialnjutkan dengan pendidikan menengah di SMPN1 Tuban pada tahun 2010 dan MA Amanatul Ummah pada tahun 2013. Penulis menjalankan studi Strata-I di Institut Teknologi Sepuluh Nopember Surabaya pada tahun 2015. Selama menjalani studi Strata-I, penulis aktif sebagai anggota Tim Robotika ITS divisi KRSBI Beroda pada tahun 2016-2019 dan menjadi ketua tim KRSBI Beroda periode 2017-2018. Penulis juga menjadi asisten praktikum di Laboratorium B202 Elektronika Dasar.

.....Halaman ini sengaja dikosongkan.....