



**SISTEM KONTROL PENGGERAK *MOBILE* ROBOT MENGGUNAKAN
PERINTAH SUARA BERBASIS *AUDIO CLASSIFICATION* PADA *DEEP
LEARNING***



TUGAS AKHIR

Program Studi

S1 Teknik Komputer

UNIVERSITAS
Dinamika

Oleh:

Rizal Rahmat Maulana

20410200021

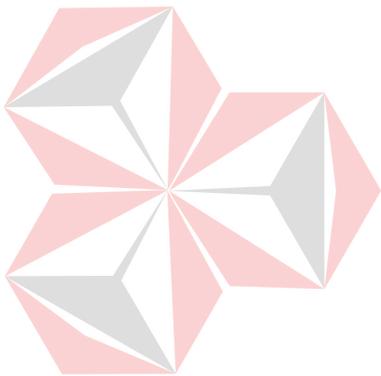
**FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA
2024**

**SISTEM KONTROL PENGGERAK *MOBILE* ROBOT MENGGUNAKAN
PERINTAH SUARA BERBASIS *AUDIO CLASSIFICATION*
PADA *DEEP LEARNING***

TUGAS AKHIR

Diajukan sebagai salah satu syarat untuk mengerjakan

Program Sarjana Teknik



UNIVERSITAS
Dinamika

Oleh:

Nama : Rizal Rahmat Maulana

NIM : 20410200021

Program : S1 (Strata Satu)

Jurusan : Teknik Komputer

**FAKULTAS TEKNOLOGI DAN INFORMATIKA
UNIVERSITAS DINAMIKA**

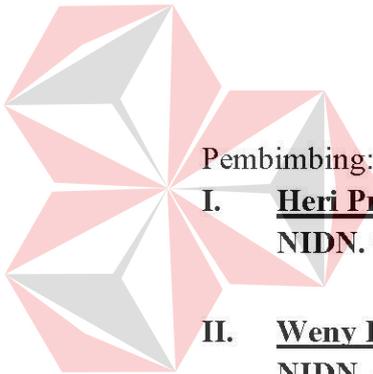
2024

TUGAS AKHIR

SISTEM KONTROL PENGGERAK *MOBILE ROBOT* MENGGUNAKAN PERINTAH SUARA BERBASIS *AUDIO CLASSIFICATION* PADA *DEEP LEARNING*

Dipersiapkan dan disusun oleh:
Rizal Rahmat Maulana
NIM : 20410200021

Telah diperiksa, dibantu, dibahas dan disetujui oleh Dewan Pembahas
Pada : 6 Agustus 2024



Susunan Dewan Penguji

Pembimbing:

- I. **Heri Pratikno, M.T., MTCNA., MTCRE.**
NIDN. 0716117302
- II. **Weny Indah Kusumawati, S.Kom., M.MT.**
NIDN. 0721047201

Digitally signed by Heri Pratikno, M.T.
DN: cn=Heri Pratikno, M.T.,
o=Universitas Dinamika, ou=S1 Teknik
Komputer, email=heri@dinamika.ac.id,
c=ID
Date: 2024.08.07 14:38:38 +07'00'
Adobe Acrobat version: 11.0.23

cn=Weny Indah Kusumawati,
o=Undika, ou=Prodi S1 TK - FTI,
email=weny@dinamika.ac.id,
c=ID
2024.08.07 16:47:11 +07'00'

Pembahas:

- I. **Musayyanah, S.ST., M.T.**
NIDN. 0730069102

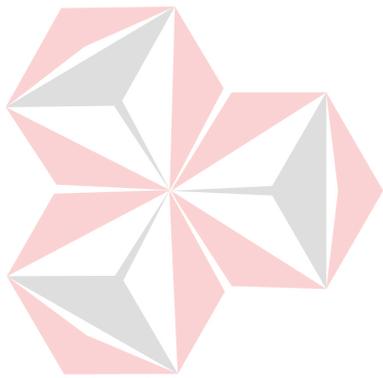
Digitally signed by
Musayyanah
DN: cn=Musayyanah,
o=Universitas Dinamika,
ou=S1 Teknik Komputer,
email=musayyanah@dinami
ka.ac.id, c=ID
Date: 2024.08.08 09:34:38
+07'00'

Tugas Akhir ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar sarjana

Digitally signed by
Anjik Sukmaaji
Date: 2024.08.12

Dr. Anjik Sukmaaji, S.Kom., M.Eng.
NIDN. 0731057301

Dekan Fakultas Teknologi dan Informatika
UNIVERSITAS DINAMIKA

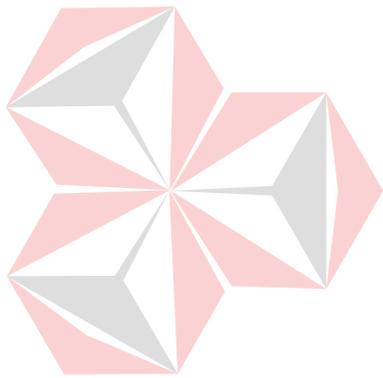


“Sesungguhnya bersama kesulitan ada
kemudahan”

QS. Al-Insyirah Ayat 6

UNIVERSITAS
Dinamika

Terima kasih kepada Orang Tua, Saudara, Teman-Teman, Keluarga Paman dan Dosen-dosen saya atas dukungan dan motivasi sehingga saya mampu menyelesaikan studi saya walaupun banyak rintangan dan hal diluar dugaan.



UNIVERSITAS
Dinamika

PERNYATAAN
PERSETUJUAN PUBLIKASI DAN KEASLIAN KARYA ILMIAH

Sebagai mahasiswa **Universitas Dinamika**, Saya :

Nama : Rizal Rahmat Maulana
NIM : 20410200021
Program Studi : S1 Teknik Komputer
Fakultas : Fakultas Teknologi dan Informatika
Jenis Karya : Laporan Tugas Akhir
Judul Karya : **KONTROL PENGGERAK *MOBILE* ROBOT
MENGUNAKAN PERINTAH SUARA BERBASIS
AUDIO CLASSIFICATION PADA *DEEP LEARNING***

Menyatakan dengan sesungguhnya bahwa :

1. Demi pengembangan Ilmu Pengetahuan, Teknologi dan Seni, Saya menyetujui memberikan kepada **Universitas Dinamika** Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalty Free Right*) atas seluruh isi/sebagian karya ilmiah Saya tersebut diatas untuk disimpan, dialihmediakan, dan dikelola dalam bentuk pangkalan data (*database*) untuk selanjutnya didistribusikan atau dipublikasikan demi kepentingan akademis dengan tetap mencantumkan nama Saya sebagai penulis atau pencipta dan sebagai pemilik Hak Cipta.
2. Karya tersebut diatas adalah hasil karya asli Saya, bukan plagiat baik sebagian maupun keseluruhan. Kutipan, karya, atau pendapat orang lain yang ada dalam karya ilmiah ini semata-mata hanya sebagai rujukan yang dicantumkan dalam Daftar Pustaka Saya.
3. Apabila dikemudian hari ditemukan dan terbukti terdapat tindakan plagiasi pada karya ilmiah ini, maka Saya bersedia untuk menerima pencabutan terhadap gelar kesarjanaan yang telah diberikan kepada Saya.

Demikian surat pernyataan ini Saya buat dengan sebenar-benarnya.

Surabaya, 19 Juli 2024



Rizal Rahmat Maulana
NIM : 20410200021

ABSTRAK

Pada saat ini, teknologi memiliki perkembangan yang sangat cepat dengan bukti banyak tugas yang dilakukan manusia sekarang dapat dikerjakan dengan otomatis dan cepat. Contohnya menggerakkan robot dengan berbagai cara, menggunakan remot, menggunakan kamera atau menggunakan alat kontrol lainnya. Penelitian ini mengembangkan sistem kontrol penggerak robot menggunakan perintah suara menggunakan komunikasi WiFi UDP pada platform ESP32. Sistem ini terdiri dari pengumpulan dan augmentasi data audio, ekstraksi fitur menggunakan *Mel-Frequency Cepstral Coefficients* (MFCC), pelatihan model klasifikasi, dan implementasi kontrol motor. Dataset audio dikumpulkan dan diaugmentasi untuk meningkatkan variasi data pelatihan. Model klasifikasi suara yang dilatih menunjukkan akurasi tinggi dalam mengenali perintah suara dan diintegrasikan dengan kontrol motor berbasis ESP32 melalui komunikasi WiFi UDP. Hasil evaluasi menunjukkan sistem ini efektif dan andal dalam mengenali perintah suara dan mengontrol motor dengan cepat dan akurat. Hasil penelitian pada proses *testing* klasifikasi suara sebanyak 30 percobaan setiap perintah suara dengan perintah secara langsung dari perintah suara maju mempunyai akurasi 66.67%, perintah suara mundur mempunyai akurasi 56.67%, perintah suara kanan mempunyai akurasi 66.67%, perintah suara kiri mempunyai akurasi 50% dan perintah suara berhenti mempunyai akurasi 53.33%. Hasil penelitian pada proses *testing* pengujian jarak terbaik sebanyak 10 percobaan setiap perintah suara dengan perintah secara langsung pada jarak 20 cm mempunyai akurasi 28% untuk semua perintah suara, pada jarak 50 cm dan 75 cm mempunyai akurasi 30% untuk semua perintah suara, pada jarak 30 cm mempunyai akurasi 26% untuk semua perintah suara, pada jarak 100 cm mempunyai akurasi 16% untuk semua perintah suara.

Kata Kunci: Klasifikasi suara, Kontrol motor, ESP32, WiFi UDP, MFCC.

KATA PENGANTAR

Dengan penuh rasa syukur kepada Allah SWT yang telah memberikan berkah dan petunjuk-Nya, sehingga penulis dapat menyelesaikan laporan Tugas Akhir ini. Penulisan laporan ini adalah sebagai salah satu syarat menempuh Tugas Akhir pada Program Studi S1 Teknik Komputer Universitas Dinamika.

Penyelesaian laporan Tugas Akhir ini tidak dapat dipisahkan dari bantuan dan kontribusi berbagai pihak yang telah memberikan masukan, nasihat, saran, kritik, serta dukungan moral dan materi kepada penulis. Oleh karena itu, penulis ingin mengucapkan rasa terima kasih yang sebesar-besarnya kepada:

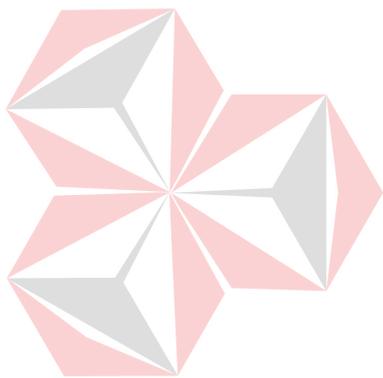
1. Allah SWT atas rahmat, hidayah, dan kesempatan yang diberikan, sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Ibu, Bapak, dan seluruh keluarga tercinta. Dukungan, doa, dan semangat yang selalu diberikan dalam setiap langkah dan aktivitas penulis sangatlah berarti.
3. Bapak Heri Pratikno, M.T., MTCNA., MTCRE., yang berperan sebagai pembimbing dalam kegiatan Tugas Akhir, membantu dalam penempatan dan memberikan izin kepada penulis untuk melakukan Tugas Akhir.
4. Ibu Weny Indah Kusumawati, S.Kom., M.MT., yang telah membimbing, mendukung, dan memberikan motivasi kepada penulis selama proses Tugas Akhir.
5. Teman-teman tergabung di Keluarga Paman yang telah menjadi teman mengerjakan tugas bersama, teman bercerita tentang perkuliahan dan keluh kesah setiap hari.
6. Teman-teman tercinta yang telah memberikan bantuan dan dukungan dalam penyusunan laporan ini.
7. Pihak-pihak lain yang tidak dapat disebutkan satu-persatu yang telah memberikan bantuan dan dukungan kepada penulis.

Semoga Allah SWT memberikan pahala yang setimpal kepada semua pihak yang telah membantu dan memberikan bimbingan serta nasehat dalam proses Tugas Akhir ini. Penulis menyadari bahwa masih terdapat kekurangan dalam pelaksanaan Tugas Akhir ini, oleh karena itu, kritik yang konstruktif dan saran dari semua pihak sangat diharapkan agar aplikasi ini dapat diperbaiki menjadi lebih baik. Semoga

laporan Tugas Akhir ini diterima dengan baik dan memberikan manfaat bagi penulis serta semua pihak yang terlibat.

Surabaya, 6 Agustus 2024

Penulis

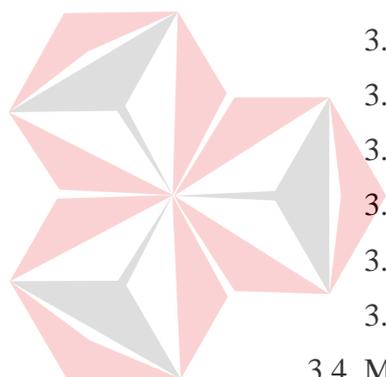


UNIVERSITAS
Dinamika

DAFTAR ISI

	Halaman
ABSTRAK	vii
KATA PENGANTAR	viii
DAFTAR ISI	x
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xiv
DAFTAR LAMPIRAN	xvii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan	3
1.5 Manfaat	4
BAB II LANDASAN TEORI	5
2.1 Robot	5
2.1.1 <i>Mobile Robot</i>	5
2.1.2 Robot Manipulator	6
2.1.3 Robot <i>Humanoid</i>	7
2.1.4 Robot Berkaki	7
2.1.5 Robot Terbang	8
2.2 MFCC	8
2.3 ESP32	10
2.4 Suara	11
2.5 <i>Deep Learning</i>	12
2.5.1 <i>Convolutional Neural Network (CNN)</i>	12
2.5.2 Lapisan Konvolusi (<i>Convolutional Layer</i>)	12
2.5.3 Lapisan Aktivasi (<i>Activation Layer</i>)	13
2.5.4 Lapisan Penggabungan (<i>Pooling Layer</i>)	13
2.5.5 Lapisan Terhubung Penuh (<i>Fully-Connected Layer</i>)	14
2.5.6 <i>Model Compilation</i>	15

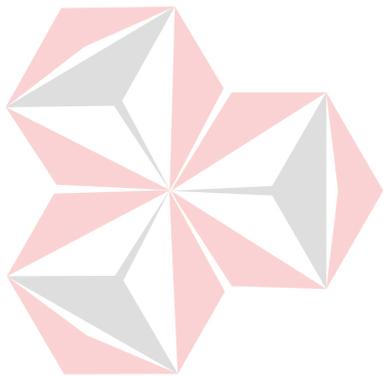
2.6 Python	15
2.7 Arduino IDE	16
2.8 WO MIC Client	16
2.9 Jupyter Notebook	17
2.10 Visual Studio Code	17
2.11 Waveform	18
BAB III METODOLOGI PENELITIAN	19
3.1 Perangkat Keras	19
3.2 Instalasi Environment	20
3.3 Data Pre-paration	20
3.3.1 Karakteristik Audio	20
3.3.2 Pengumpulan Dataset	33
3.3.3 Augmentasi Data	33
3.3.4 Rename File Dataset	37
3.3.5 Flowchart Dataset	37
3.3.6 Pengelompokan Data	39
3.3.7 Distribusi Data secara Katagori	40
3.3.8 Datasheet	41
3.3.9 Distribusi Folds Subset	41
3.4 Metode Ekstraksi Fitur dengan MFCC	42
3.5 Flowchart Python	47
3.6 Flowchart ESP32	48
3.7 Transmisi Pengiriman Data	49
3.8 Klasifikasi Audio	52
BAB IV HASIL DAN PEMBAHASAN	55
4.1 Pengujian Klasifikasi Perintah Suara	55
4.1.1 Pengujian Klasifikasi Perintah Suara	55
4.1.2 Alat yang Digunakan Pengujian Klasifikasi Perintah Suara	55
4.1.3 Cara Pengujian Klasifikasi Perintah Suara	55
4.1.4 Hasil Pengujian Klasifikasi Perintah Suara	56
4.1.5 Analisis Pengujian Klasifikasi Perintah Suara	57
4.2 Pengujian MFCC	57



4.2.1	Pengujian MFCC.....	57
4.2.2	Alat yang Digunakan Pengujian MFCC	57
4.2.3	Cara Pengujian MFCC	58
4.2.4	Hasil Pengujian MFCC	58
4.2.5	Analisis Pengujian MFCC	60
4.3	Pengujian Kinerja Motor DC.....	62
4.3.1	Pengujian Kinerja Motor DC	62
4.3.2	Alat yang Digunakan Pengujian Kinerja Motor DC.....	62
4.3.3	Cara Pengujian Kinerja Motor DC.....	63
4.3.4	Hasil Pengujian Kinerja Motor DC.....	63
4.3.5	Analisis Pengujian Kinerja Motor DC	64
4.4	Pengujian Akurasi Perintah Suara dengan Motor DC	65
4.4.1	Pengujian Akurasi Perintah Suara dengan Motor DC	65
4.4.2	Alat yang Digunakan Pengujian Akurasi Perintah Suara dengan Motor DC	65
4.4.3	Cara Pengujian Akurasi Perintah Suara dengan Motor DC ..	65
4.4.4	Hasil Pengujian Akurasi Perintah Suara dengan Motor DC	66
4.4.5	Analisis Pengujian Akurasi Perintah Suara dengan Motor DC.	67
4.5	Pengujian Jarak Terbaik.....	70
4.5.1	Pengujian Jarak Terbaik.....	70
4.5.2	Alat yang Digunakan Pengujian Jarak Terbaik.....	70
4.5.3	Cara Pengujian Jarak Terbaik	71
4.5.4	Hasil Pengujian Jarak Terbaik	71
4.5.5	Analisis Pengujian Jarak Terbaik.....	72
BAB V PENUTUP.....		75
5.1	Kesimpulan	75
5.2	Saran	76
DAFTAR PUSTAKA		77
LAMPIRAN.....		80

DAFTAR TABEL

	Halaman
Tabel 4.1 Hasil Pengujian Klasifikasi Suara.....	56
Tabel 4.2 Hasil Pengujian MFCC	59
Tabel 4.3 Hasil Pengujian Kinerja Motor DC.....	64
Tabel 4.4 Hasil Pengujian Akurasi Perintah Suara dengan Motor DC	66
Tabel 4.5 Hasil Pengujian Jarak Terbaik	72



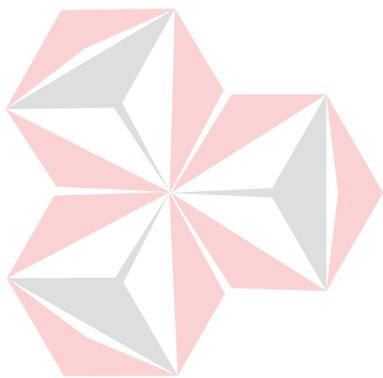
UNIVERSITAS
Dinamika

DAFTAR GAMBAR

	Halaman
Gambar 2.1 <i>Mobile Robot</i>	6
Gambar 2.2 Lengan Robot	6
Gambar 2.3 Robot Humanoid	7
Gambar 2.4. Robot Berkaki	7
Gambar 2.5 Robot Terbang.....	8
Gambar 2.6 Blok Diagram <i>MFCC</i>	9
Gambar 2.7 ESP32 (<i>datasheet</i>)	11
Gambar 2.8 PinOut ESP32.....	11
Gambar 2.9 Logo Python	16
Gambar 2.10. Arduino IDE.....	16
Gambar 2.11 WO MIC Client.....	17
Gambar 2.12 <i>Jupyter Notebook</i>	17
Gambar 2.13 Visual Studio Code	17
Gambar 3.1 Blok Diagram Perangkat Keras.....	19
Gambar 3.2 Model Perancangan Hardware	19
Gambar 3.3 <i>Wavefrom</i> berhenti age.wav	21
Gambar 3.4 <i>Wavefrom</i> maju age.wav	22
Gambar 3.5 <i>Wavefrom</i> kanan age.wav.....	22
Gambar 3.6 <i>Wavefrom</i> kiri age.wav.....	23
Gambar 3.7 <i>Wavefrom</i> mundur age.wav.....	24
Gambar 3.8 Program Membuat <i>Wavefrom</i>	24
Gambar 3.9 <i>Spectrogram</i> berhenti age.wav	26
Gambar 3.10 <i>Spectrogram</i> maju age.wav	27
Gambar 3.11 <i>Spectrogram</i> kanan age.wav	28
Gambar 3.12 <i>Spectrogram</i> kiri age.wav	29
Gambar 3.13 <i>Spectrogram</i> mundur age.wav	30
Gambar 3.14 Program Membuat <i>Spectrogram</i>	31
Gambar 3.15 Pengumpulan Dataset.....	33
Gambar 3.16 <i>Wavefrom</i> berhenti age_high_pass.wav	35

Gambar 3.17 <i>Wavefrom</i> berhenti <i>age_low_pass.wav</i>	35
Gambar 3.18 <i>Wavefrom</i> berhenti <i>age_band_pass.wav</i>	36
Gambar 3.19 <i>Rename file</i> dataset.....	37
Gambar 3.20 <i>Training Dataset</i>	38
Gambar 3.21 Pengelompokan Data Berdasarkan Kategori.....	39
Gambar 3.22 Distribusi Folds Berdasarkan Class.....	40
Gambar 3.23 <i>Datasheet</i> atau lembaran data	41
Gambar 3.24 Tabel <i>Subset Dataset</i>	42
Gambar 3.25 Proses <i>Hamming Window</i>	43
Gambar 3.26 Proses FFT	44
Gambar 3.27 Proses Filter Bank	45
Gambar 3.28 Coefficient Cepstral.....	46
Gambar 3.29 <i>Flowchart Python</i>	48
Gambar 3.30 <i>Flowchart ESP32</i>	49
Gambar 3.31 Program Python Mengirim Data	50
Gambar 3.32 Program ESP32 Menerima Data	51
Gambar 3.33 Import <i>library</i>	52
Gambar 3.34 Import variabel	53
Gambar 3.35 Layer CNN 1D	53
Gambar 3.36 Kompilasi Model.....	54
Gambar 4.1 Pengujian Klasifikasi Perintah Suara	56
Gambar 4.2 Grafik Pengujian Klasifikasi Perintah Suara.....	56
Gambar 4.3 Hasil <i>Hamming Window</i>	58
Gambar 4.4 Hasil <i>FFT</i>	58
Gambar 4.5 Hasil <i>MFCC</i>	59
Gambar 4.6 Hasil <i>Filter Bank</i>	59
Gambar 4.7 Grafik Pengujian MFCC	60
Gambar 4.8 Pengujian Kinerja Motor DC	63
Gambar 4.9 Grafik Pengujian Kinerja Motor DC	64
Gambar 4.10 Pengujian Akurasi Perintah Suara dengan Motor DC.....	66
Gambar 4.11 Grafik Pengujian Akurasi Perintah Suara dengan Motor DC	67
Gambar 4.12 Pengujian Jarak Terbaik	71

Gambar 4.13 Grafik Pengujian Jarak Terbaik DC 72



UNIVERSITAS
Dinamika

DAFTAR LAMPIRAN

	Halaman
Lampiran 1 Tabel Pengujian Klasifikasi Suara.....	80
Lampiran 2 Tabel Pengujian MFCC	84
Lampiran 3 Tabel Pengujian Kinerja Motor DC.....	88
Lampiran 4 Tabel Pengujian Akurasi Orang Pertama dengan Motor DC	92
Lampiran 5 Tabel Pengujian Akurasi Orang Kedua dengan Motor DC	94
Lampiran 6 Tabel Pengujian Akurasi Orang Ketiga dengan Motor DC.....	96
Lampiran 7 Tabel Pengujian Akurasi Orang Keempat dengan Motor DC	98
Lampiran 8 Tabel Pengujian Akurasi Orang Kelima dengan Motor DC.....	100
Lampiran 9 Tabel Pengujian Jarak 20 cm.....	102
Lampiran 10 Tabel Pengujian Jarak 30 cm	104
Lampiran 11 Tabel Pengujian Jarak 50 cm	106
Lampiran 12 Tabel Pengujian Jarak 75 cm.....	108
Lampiran 13 Tabel Pengujian Jarak 100 cm.....	110
Lampiran 14 Program Utama.....	112
Lampiran 15 Program Arduino IDE	126
Lampiran 16 <i>Flowchart</i> Python	129
Lampiran 17 <i>Flowchart</i> ESP32.....	130
Lampiran 18 Form Bimbingan Tugas Akhir.....	131
Lampiran 19 Bukti Originalitas Tugas Akhir	132
Lampiran 20 Biodata Pencipta	133

BAB I

PENDAHULUAN

1.1 Latar Belakang

Di zaman sekarang yang sangat maju, perkembangan teknologi juga berkembang pesat terutama bidang robot diberbagai aspek kehidupan untuk membuat masyarakat lebih mudah serta efisien. Salah satu alasan yang bisa mengakibatkan hal tersebut karena penggunaan robot yang mudah dalam mendapatkan serta mengendalikan. Kata “robot” berasal dari kata *robota* Ceko, yang memiliki arti kerja paksa dan muncul pada tahun 1920-an dalam lakon Rossum’s Universal Robots oleh Ceko Karel Čapek (Jaya *dkk.*, 2018). Pengendalian robot yang dilakukan dalam sehari-hari kebanyakan adalah *push button* dan *switch* tetapi terdapat juga robot yang dikendalikan dengan perintah suara yang menjadikan suatu interaksi yang menarik dan alami antara manusia dan robot.

Pengenalan suara sering digunakan dalam beberapa tahun terakhir, termasuk di dalam *deep learning*. Diantaranya adalah Microsoft Corporation yang menggunakan pengenalan suara didalam sistem operasi Windows. Sistem ini telah berkembang hingga versi 5.1. dengan standard *interface* SAPI (*Speech Application Programming Interface*) (Noertjahyana dan Adipranata, 2003). Hal ini membuat sistem pengenalan suara berkembang menjadi lebih akurat dan responsive meski dalam kondisi yang bising.

Menurut (Nasution, 2012) pada jurnal yang berjudul *Metoda Mel Frequency Cepstrum Coefficients (MFCC) untuk Mengenali Ucapan pada Bahasa Indonesia* salah satu metode yang sering ditemukan dalam bidang *speech technology* adalah *Mel-Frequency Cepstral Coefficienti (MFCC)*. Metode ini digunakan pada *speaker recognition* maupun *speech recognition* dengan melakukan *feature extraction* yang menggunakan beberapa parameter dari perubahan sinyal suara. *MFCC feature extraction* merupakan sebuah adaptasi dari sistem pendengaran manusia yang akan disaring secara linear untuk frekuensi rendah seperti dibawah 1000Hz dan disaring secara logaritmik untuk frekuensi tinggi seperti diatas 1000Hz.

Dengan memadukan konsep pengendali robot dengan pengenalan suara menggunakan *deep learning*, diharapkan dapat membuat sistem pengendali yang lebih responsive dan intuitif. Sistem ini juga memungkinkan pengguna untuk mengendalikan robot tanpa *remote control* atau gestur tangan.

Pada penelitian yang sebelumnya dilakukan oleh (Arianda, 2024) adalah Sistem Kontrol Kecepatan Kipas Angin Menggunakan Suara Multi-Bahasa Melalui Klasifikasi Audio Pada YAMNET, memiliki perbedaan objek penelitian dan metode pendeteksi yang dilakukan, objek penelitian sebelumnya menggunakan kipas dan pada penelitian ini *mobile* robot serta metode yang digunakan sebelumnya menggunakan suara multi-bahasa yang dapat mengontrol kecepatan kipas angin sedangkan pada penelitian ini menggunakan metode klasifikasi suara.

Pada penelitian sebelumnya yang dilakukan oleh (Lubis, 2022) adalah model terbaru menggunakan perintah suara untuk menstater mesin mobil dan keamanannya menggunakan *Smartphone* berbasis Arduino UNO. Selain itu, pada penelitian yang dilakukan oleh (Nusyirwan, 2020) menggunakan voice google untuk menggerakkan motor dengan Arduino. Sedangkan penelitian yang akan dilakukan ini adalah menggunakan metode *deep learning* yang berbeda dari (Arianda, 2024) untuk deteksi suara menggerakkan *mobile* robot.

Pada penelitian sebelumnya yang dilakukan oleh (Putra, Musthafa dan Kholil, 2021) adalah Klasifikasi Intonasi Bahasa Jawa Khas Ponorogo Menggunakan Algoritma *Multilayer Perceptron Neural Network*, yang memiliki objek penelitian mengklasifikasi intonasi suara khas Ponorogo menggunakan *Mel-Frequency Cepstral Coefficienti* (MFCC) dan tidak dilakukan pendeteksian suara secara langsung. Sedangkan pada penelitian kali ini penggunaan *Mel-Frequency Cepstral Coefficienti* (MFCC) untuk mengekstraksi fitur suara dalam mendeteksi perintah suara.

Pada penelitian ini, penulis merancang dan membangun sebuah sistem yang dapat menggerakkan *mobile* robot menggunakan perintah suara berbasis *audio classification* pada *deep learning*. Proses pengendalian robot dilakukan secara langsung (*realtime*) berdasarkan perintah suara melalui *microphone smartphone* secara *wireless* telah dilatih dengan dataset suara *artificial intelligence*

menggunakan metode *Audio Classification* dalam *framework* MFCC (*Mel-Frequency Cepstral Coefficients*).

1.2 Rumusan Masalah

Dari latar belakang tersebut, dapat dirumuskan masalah pada Tugas Akhir ini sebagai berikut:

1. Bagaimana performa klasifikasi suara menggunakan MFCC dalam klasifikasi perintah suara lewat *microphone smartphone*?
2. Bagaimana perintah suara dapat dikenali menggunakan MFCC untuk mengendalikan pergerakan *mobile robot*?
3. Berapa besar akurasi kecocokan antara perintah suara dengan arah gerak *mobile robot*?
4. Berapa jarak yang memiliki akurasi tertinggi antara pemberi perintah suara dengan *microphone smartphone* untuk mengendalikan pergerakan *mobile robot*?

1.3 Batasan Masalah

Berdasarkan latar belakang diatas, maka dapat dirumuskan masalah pada Tugas Akhir ini sebagai berikut:

1. Tidak membahas suara berdasarkan pada *gender*.
2. Menggunakan suara normal
3. Tidak membahas warna suara.
4. Tidak berbicara tentang intonasi suara.
5. Hanya menggunakan kata “maju”, “mundur”, “kanan”, “kiri”, “berhenti”.

1.4 Tujuan

Berdasarkan latar belakang dan rumusan di atas, yang dapat menjadi tujuan pada Tugas Akhir ini sebagai berikut:

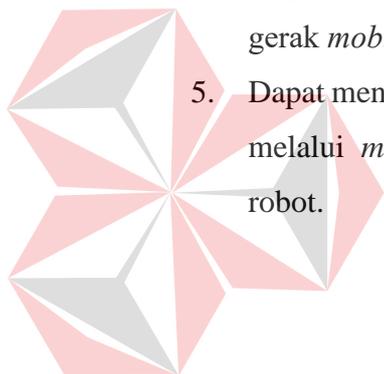
1. Mengidentifikasi akurasi klasifikasi suara menggunakan MFCC untuk setiap perintah suara lewat *microphone smartphone*.
2. Mendeteksi perintah suara secara langsung (*realtime*) menggunakan metode MFCC untuk mengendalikan *mobile robot*.

3. Mengetahui seberapa tinggi tingkat akurasi kecocokan antara perintah suara dengan arah gerak *mobile* robot.
4. Mengetahui jarak dengan akurasi tertinggi antara pemberi perintah suara melalui *microphone smartphone* untuk mengendalikan pergerakan *mobile* robot.

1.5 Manfaat

Manfaat dari penelitian ini dapat diperoleh sebagai berikut:

1. Dapat mengetahui akurasi klasifikasi suara menggunakan MFCC untuk setiap perintah suara lewat *microphone smartphone*.
2. Dapat mendeteksi perintah suara secara langsung (*realtime*) menggunakan metode MFCC untuk mengendalikan *mobile* robot.
3. Dapat mengetahui tingkat akurasi kecocokan antara perintah suara dengan arah gerak *mobile* robot.
5. Dapat mengetahui jarak dengan akurasi tertinggi antara pemberi perintah suara melalui *microphone smartphone* untuk mengendalikan pergerakan *mobile* robot.



UNIVERSITAS
Dinamika

BAB II

LANDASAN TEORI

2.1 Robot

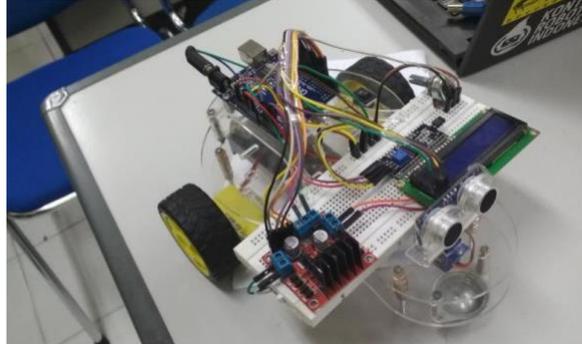
Robot adalah alat mekanis yang mampu melakukan tugas fisik di bawah kendali manusia atau berdasarkan program yang telah diatur sebelumnya. Robot digunakan untuk pekerjaan yang berat, berbahaya, berulang, dan kotor. Sebagian besar robot industri digunakan di sektor manufaktur. Aplikasi robot lainnya termasuk pembersihan limbah beracun, eksplorasi bawah air dan luar angkasa, pertambangan, pencarian dan penyelamatan, serta inspeksi tambang. Baru-baru ini, robot mulai memasuki pasar konsumen di bidang hiburan dan peralatan rumah tangga seperti penyedot debu dan mesin pemotong rumput. Saat insinyur mencoba membuat robot berjalan lagi, mereka memulai dengan platform berkaki enam dan berkaki banyak lainnya, meniru penampilan dan fungsi serangga dan arthropoda. Desain ini terbukti fleksibel dan adaptif terhadap berbagai lingkungan, meskipun biaya mekanisnya membuat mereka kurang menarik bagi pengguna. Dengan lebih dari empat kaki, robot ini stabil dan mudah dioperasikan.

Ketika para ahli robot pertama kali mencoba meniru manusia dan hewan, mereka menghadapi tantangan besar karena membutuhkan daya komputasi lebih banyak daripada yang tersedia saat itu. Akibatnya, fokus pengembangan bergeser ke bidang penelitian lainnya. Robot beroda fleksibel digunakan untuk eksperimen perilaku, navigasi, dan perencanaan jalur. Teknologi navigasi ini berkembang menjadi kontrol robot otonom. Contoh sistem kontrol navigasi otonom yang canggih saat ini termasuk sistem navigasi laser dengan lokalisasi dan *Visual Simultaneous Localization and Mapping* (VSLAM) dari Activ Media Robotics dan Evolution Robotics. Jenis-jenis utama robot adalah:

2.1.1 Mobile Robot

Mobile robot merupakan suatu struktur robotik yang bercirikan robot berbentuk roda dimana seluruh tubuh robot bergerak untuk memindahkan robot dari satu tempat ke tempat lain (Khairudin *dkk.*, 2020). Robot seluler ini sangat populer di kalangan orang-orang yang baru mulai belajar robot. Hal ini karena pembuatan

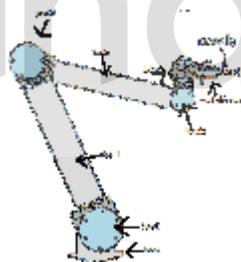
mobil robot tidak memerlukan tenaga fisik. Membangun robot bergerak memerlukan pemahaman tentang mikrokontroler dan sensor elektronik, pada Gambar 2.1 menampilkan salah satu bentuk *mobile robot*.



Gambar 2.1 *Mobile Robot*
(Sumber: (Khairudin dkk., 2020))

2.1.2 Robot Manipulator

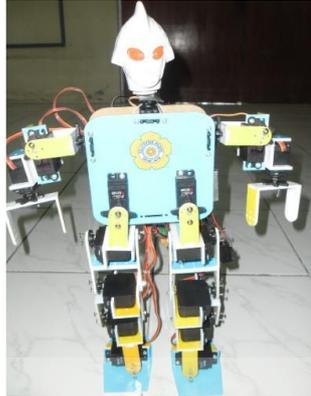
Robot tersebut memiliki lengan mirip manusia yang berfungsi untuk menggenggam dan menggerakkan benda. Contoh Gambar 2.2 adalah robot las dan mesin perakitan elektronik di industri otomotif (Pamungkas dan Noviansyah, 2021).



Gambar 2.2 Lengan Robot
(Sumber: (Pamungkas dan Noviansyah, 2021))

2.1.3 Robot Humanoid

Pada Gambar 2.3 menunjukkan salah satu bentuk robot sepak bola yang memiliki kemampuan dan perilaku seperti manusia. Robot adalah robot humanoid yang bergerak dengan memahami lingkungannya menggunakan kamera sebagai matanya. Robot ini dilengkapi dengan kamera, sensor, dan kompas untuk navigasi (Soim *dkk.*, 2015).



Gambar 2.3 Robot Humanoid
(Sumber: (Soim *dkk.*, 2015))

2.1.4 Robot Berkaki

Pada Gambar 2.4 merupakan jenis robot yang menggunakan motor servo sebagai pengendali utama dan mikrokontroler sebagai pengendali motornya, sehingga merupakan robot berkaki empat yang disebut *quadrocopter*. Seperti namanya, robot jenis ini memiliki empat kaki untuk mengendalikannya. Tidak seperti kendaraan beroda yang hanya bisa bergerak di medan datar, robot ini mampu melintasi medan yang tidak rata (Putra, Kalsum dan Susanto, 2022).



Gambar 2.4. Robot Berkaki
(Sumber: (Putra, Kalsum dan Susanto, 2022))

2.1.5 Robot Terbang

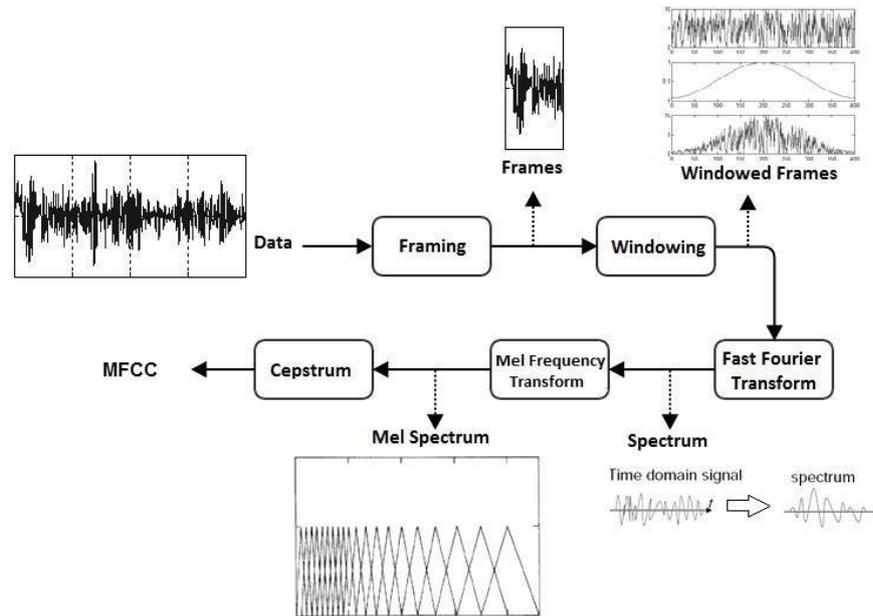
Robot terbang adalah robot yang memiliki kemampuan untuk bergerak di udara bebas, yang digerakkan motor atau mesin penggerak lainnya sebagaimana tampak pada Gambar 2.5. Robot dengan jenis helikopter diperuntukkan membantu hal-hal yang tidak bisa dijangkau oleh manusia seperti memantau bencana alam, memantau kebakaran lahan, memantau kemacetan jalan, memantau konstruksi pembangunan, dan survei serta pemetaan wilayah (Sirajuddin, 2013).



Gambar 2.5 Robot Terbang
(Sumber: (Alamsyah, 2022))

2.2 MFCC

Mel Frequency Cepstrum Coefficients (MFCC) diperkenalkan oleh Davis dan Mermelstein pada tahun 1980 mengusulkan menggunakan MFCC ekstraksi fitur untuk sistem pengenalan suara (Dyarbirru dan Hidayat, 2020). Metode yang digunakan dalam ekstraksi fitur suara dengan mendapatkan suatu parameter dan informasi lainnya dari suara tersebut (Permana, Nurhasanah dan Zulkarnaik, 2018). Proses penghitungan MFCC seperti pada Gambar 2.6 dimulai dengan penguatan sinyal suara menggunakan filter pre-emphasis untuk meningkatkan kejelasan frekuensi tinggi. Filter ini diterapkan pada sinyal suara dengan tujuan untuk meningkatkan *Signal to Noise Ratio* (SNR) dan membantu dalam mengurangi efek distorsi yang disebabkan oleh karakteristik peralatan perekaman atau transmisi. Filter pre-emphasis diterapkan dengan menerapkan filter FIR (*Finite Impulse Response*) pada sinyal suara. Filter ini dirancang untuk meningkatkan amplitudo komponen frekuensi tinggi dalam sinyal.



Gambar 2.6 Blok Diagram *MFCC*
(Sumber: (Arslan dan Yildiz, 2018))

Selanjutnya, sinyal dibagi menjadi jendela waktu kecil dan diaplikasikan fungsi window seperti Hamming window untuk meminimalkan efek sisi. Fungsi Hamming adalah fungsi matematika yang digunakan dalam pemrosesan sinyal untuk membuat hamming window. Hamming window adalah proses sinyal yang digunakan untuk mengurangi efek tepi saat melakukan efek spektral. Yang ditemukan oleh Earl. G. Hamming. Hamming window menghasilkan bentuk jendela yang memiliki puncak di tengah dan perlahan berkurang ke nilai nol di kedua ujungnya. Setiap jendela kemudian diubah menjadi domain frekuensi melalui *Fast Fourier Transform* (FFT) untuk mendapatkan spektrum frekuensi.

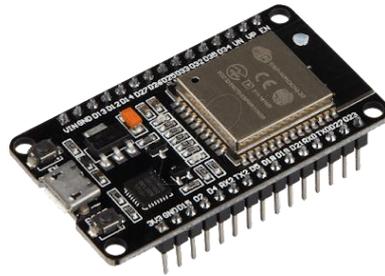
Mel Filtering kemudian diterapkan untuk mengonversi spektrum frekuensi ke skala mel, menyesuaikan dengan persepsi manusia terhadap frekuensi. Mel filter bank biasanya terdiri dari sejumlah filter triangular yang tumpang tindih, dengan setiap filter merespons terhadap sebagian dari spektrum frekuensi. Filter-filter ini dirancang agar lebih sensitif terhadap perbedaan frekuensi pada bagian-bagian frekuensi yang lebih rendah, mirip dengan karakteristik pendengaran manusia. Hasil dari pengaplikasian *Mel filter bank* adalah serangkaian nilai yang merepresentasikan seberapa banyak energi yang terkandung dalam setiap filter,

yang mencerminkan persepsi pendengaran manusia terhadap frekuensi suara. Logaritma energi merupakan metode yang umum digunakan dalam pemrosesan sinyal suara untuk mengubah nilai energi suara menjadi skala logaritmik. Penggunaan logaritma energi dalam pemrosesan suara membantu dalam meningkatkan kontras antara tingkat energi yang rendah dan tinggi, sehingga fitur-fitur penting dalam sinyal suara dapat lebih mudah diidentifikasi.

Langkah terakhir melibatkan penggunaan *Discrete Cosine Transform* (DCT) untuk menghasilkan koefisien cepstral, yang merepresentasikan energi dalam rentang frekuensi tertentu. *Discrete Cosine Transform* (DCT) adalah transformasi matematis yang digunakan dalam pemrosesan sinyal untuk mengubah data dari domain spasial menjadi domain frekuensi. DCT digunakan setelah penerapan filter Mel pada spektrum frekuensi untuk menghasilkan koefisien cepstral, yang merupakan representasi fitur utama dari sinyal suara dalam domain cepstral. Koefisien ini digunakan untuk mewakili sifat-sifat spektral sinyal suara, seperti kejelasan, nada, dan pola frekuensi. Dalam aplikasi pengenalan suara, koefisien cepstral digunakan sebagai fitur utama untuk membedakan berbagai jenis suara. DCT membantu mengompresi informasi frekuensi yang dihasilkan oleh filter Mel sehingga fitur-fitur yang paling relevan dari sinyal suara dapat diekstraksi dengan lebih efisien.

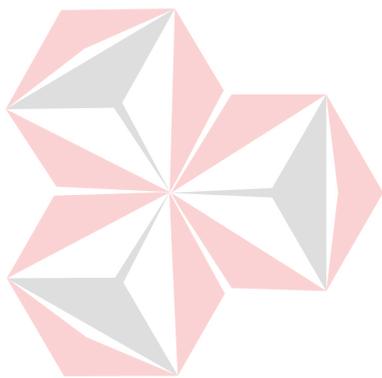
2.3 ESP32

Pada Gambar 2.7 merupakan Espressif Systems, ESP32 mencakup ESP32-D0WDQ6 (termasuk ESP32-D0WD), ESP32-D2WD, ESP32-S0WD, dan ESP32-PICO-D4 System dalam chip *Package SiP*. ESP32 adalah sistem on-chip SoC berbiaya rendah dan berdaya rendah dengan dua mode Wi-Fi dan Bluetooth. Mikroprosesor Tensilica -tensa L-6 (dual-core dan single-core) memiliki clock hingga 240 MHz. ESP32 terdiri dari perangkat antena internal, balun RF, penguat daya, dan modul kontrol daya, filter, dan penguat daya dengan kebisingan rendah. Dengan fitur hemat daya seperti resolusi frekuensi clock yang tepat, tegangan variabel, dan kecepatan transfer daya dinamis, ESP32 dirancang untuk perangkat seluler, perangkat elektronik yang dapat dikenakan, dan aplikasi IoT (Akbar, 2020).



Gambar 2.7 ESP32 (*datasheet*)
(Sumber: (Watson, 2020))

Pada Gambar 2.8 merupakan modul ESP32, penerus ESP8266, sangat populer untuk aplikasi IoT karena memiliki inti CPU lebih banyak, kecepatan Wi-Fi lebih cepat, GPIO lebih banyak, dan sedikit dukungan kemampuan *Bluetooth*.



Gambar 2.8 PinOut ESP32
(Sumber: (Watson, 2020))

2.4 Suara

Suara adalah bunyi manusia yang keluar dari mulut, seperti berbicara, tertawa, menangis, dan berteriak, yang digunakan oleh manusia sebagai alat komunikasi, bertukar informasi antar sesama manusia. Namun dengan berkembangnya teknologi modern, komunikasi tidak hanya antara manusia dengan manusia lainnya, tetapi juga antara manusia dengan komputer. Pertama, komputer diberikan instruksi untuk dapat membaca suara yang masuk, yang biasa disebut dengan *supervised learning* (Adnan, Amelia dan Shiddiq, 2022).

2.5 Deep Learning

Deep learning adalah salah satu cabang dari machine learning yang berfokus pada pembelajaran dari data yang diwakili dalam bentuk yang lebih abstrak, seperti jaringan neural. *Deep learning* menggunakan arsitektur jaringan neural yang terdiri dari banyak lapisan (disebut "deep" karena memiliki banyak lapisan) untuk memahami dan memodelkan representasi data yang kompleks. Dengan menggunakan lapisan-lapisan ini, *deep learning* dapat belajar secara otomatis untuk mengekstraksi fitur-fitur yang relevan dari data input tanpa perlu dijelaskan secara eksplisit. *Deep learning* telah digunakan dalam berbagai aplikasi, termasuk pengenalan suara, pengenalan wajah, pengenalan tulisan tangan, pengenalan objek dalam gambar. Keunggulan utama *deep learning* adalah kemampuannya untuk mengatasi masalah yang sangat kompleks dan dapat menghasilkan hasil yang sangat baik dalam banyak bidang.

2.5.1 Convolutional Neural Network (CNN)

Semua neuron dalam *Convolutional Neural Network* (CNN) terdiri dari neuron yang bias dan bobot. Setiap neuron menerima input dan melanjutkannya dengan melakukan perkalian titik (Azmi, Defit dan Sumijan, 2023). Tiga jenis lapisan utama terdiri dari CNN: lapisan konvolusi, lapisan pooling, dan lapisan fully connected. Lapisan konvolusi mengidentifikasi atribut dalam gambar melalui filter, dan lapisan pooling mengurangi ukuran gambar untuk mengurangi kompleksitas. CNN dapat digunakan dalam berbagai aplikasi seperti pengenalan wajah dan objek, serta sangat bermanfaat dalam klasifikasi gambar dan pengenalan objek.

2.5.2 Lapisan Konvolusi (*Convolutional Layer*)

Lapisan konvolusi adalah lapisan pertama dalam CNN yang bertugas untuk mengidentifikasi pola-pola sederhana dalam gambar. Lapisan ini menggunakan kernel (filter) untuk melakukan dot product dengan bagian-bagian dari input gambar, sehingga menghasilkan feature gambar yang lebih sederhana. Contoh, lapisan konvolusi pertama menggunakan kernel ukuran 5x5 dengan stride 1 dan padding 2.

2.5.3 Lapisan Aktivasi (*Activation Layer*)

Fungsi non-linear ReLU, yang dapat mempercepat pelatihan dan membantu model menangkap berbagai pola kompleks, bekerja dengan mengatur setiap nilai negatif dalam input menjadi nol dan mempertahankan nilai positif apa adanya. Akibatnya, fungsi ini sangat umum digunakan dalam jaringan saraf. Ini menunjukkan bahwa nilai negatif dari lapisan sebelumnya akan diubah menjadi nol, sementara nilai positif tetap sama. Penambahan lapisan aktivasi ReLU ini sangat penting untuk memperkenalkan non-linearitas ke dalam model. Hal ini memungkinkan jaringan untuk belajar dari data yang lebih kompleks dan menyelesaikan masalah yang tidak dapat diselesaikan oleh model linear. Dengan menambahkan activation ReLU, membuat model dalam pelatihan dan generalisasi yang lebih baik terhadap data yang belum pernah dilihat sebelumnya.

Fungsi aktivasi Softmax mengubah output dari lapisan sebelumnya menjadi distribusi probabilitas yang jumlahnya satu, yang berarti bahwa setiap output menunjukkan kemungkinan bahwa input akan termasuk dalam salah satu kelas yang tersedia. Fungsi ini bekerja dengan menghitung eksponensial dari setiap nilai output dan kemudian membaginya dengan jumlah eksponensial tersebut. Dengan menambahkan activation softmax, membuat output model dapat diinterpretasikan sebagai probabilitas yang dapat digunakan untuk menentukan kelas mana yang paling mungkin untuk setiap input yang diberikan

2.5.4 Lapisan Penggabungan (*Pooling Layer*)

Lapisan penggabungan digunakan untuk mengurangi ukuran output gambar dan mengurangi kompleksitas. Operasi pooling seperti max pooling atau average pooling digunakan untuk mengurangi ukuran gambar. Lapisan ini digunakan untuk mengurangi ukuran peta fitur yang dibuat oleh lapisan konvolusi sebelumnya. Nilai maksimum diambil dari setiap jendela atau segmen yang memiliki ukuran `pool_size` yang bergerak sepanjang peta fitur. Misalnya, jika `pool_size=2`, setiap dua nilai berurutan dalam peta fitur akan diringkas menjadi satu nilai maksimum. Dengan menambahkan `maxpooling1d (pool_size=2)`, kita memastikan bahwa jaringan dapat menangkap fitur penting dari data sekuensial dengan lebih efisien dan efektif, sambil mengurangi kompleksitas komputasi dan risiko overfitting.

2.5.5 Lapisan Terhubung Penuh (*Fully-Connected Layer*)

Lapisan terhubung penuh digunakan untuk menghubungkan output dari layer sebelumnya ke output akhir. Lapisan ini menggunakan semua neuron dalam input untuk menghasilkan output. Contoh, lapisan fully connected digunakan untuk menghasilkan output akhir. Lapisan dense ini menghubungkan setiap neuron dari lapisan sebelumnya ke setiap neuron dalam lapisan ini, menciptakan koneksi penuh antara kedua lapisan. Setiap neuron dalam lapisan ini akan menerima kombinasi linear dari semua input yang diterima dari lapisan sebelumnya, dan kemudian menerapkan fungsi aktivasi; jika tidak ditentukan, fungsi aktivasi default adalah linear. Karena setiap neuron dapat menerima berbagai karakteristik dari input yang berbeda, penambahan lapisan ini memungkinkan model untuk belajar representasi atau pola yang lebih kompleks dari data. Di bagian akhir arsitektur jaringan saraf, lapisan penuh terhubung sering digunakan untuk mengumpulkan dan memproses fitur yang telah dihasilkan oleh lapisan sebelumnya, seperti lapisan konvolusi atau pooling, untuk kemudian digunakan dalam proses prediksi, klasifikasi, atau regresi lainnya.

Mengubah atau "meratakan" input multidimensi menjadi vektor satu dimensi adalah tujuan lapisan flatten ini. Langkah ini sangat penting ketika kita ingin menghubungkan lapisan konvolusi atau pooling yang menghasilkan output multidimensi dengan lapisan terhubung sepenuhnya yang mengharapkan input berbentuk vektor satu dimensi. Misalnya, struktur tiga dimensi (seperti tinggi, lebar, dan jumlah filter) dapat dihasilkan setelah serangkaian lapisan konvolusi dan pooling. Lapisan flatten akan mengubah struktur ini menjadi vektor satu dimensi, yang panjangnya adalah perkalian dari tiga dimensi tersebut. Oleh karena itu, lapisan flatten memungkinkan perpindahan dari bagian jaringan yang mengekstraksi fitur (seperti lapisan konvolusi dan pooling) ke bagian jaringan yang menggabungkan fitur untuk klasifikasi atau prediksi akhir. Tanpa lapisan flatten, menghubungkan lapisan tersebut secara langsung tidak mungkin.

2.5.6 *Model Compilation*

Proses ini melibatkan pengaturan optimizer, fungsi kehilangan, dan metrik yang akan digunakan selama proses pelatihan. Loss function atau fungsi kehilangan menentukan fungsi kerugian yang akan digunakan selama pelatihan. Categorical crossentropy adalah pilihan umum untuk masalah klasifikasi multi-kelas, di mana fungsi ini mengukur perbedaan antara distribusi probabilitas yang diprediksi oleh model dan distribusi probabilitas target (label sebenarnya). Tujuan pelatihan adalah meminimalkan nilai kerugian ini, sehingga prediksi model semakin mendekati label sebenarnya.

Optimizer atau optimisasi menentukan algoritma optimasi yang digunakan untuk memperbarui bobot model berdasarkan gradien dari fungsi kerugian. Adam (Adaptive Moment Estimation) adalah optimizer populer yang menggabungkan kelebihan dari dua metode lain, yaitu AdaGrad dan RMSProp, untuk melakukan pembaruan bobot yang lebih efisien dan cepat. Adam menggunakan estimasi rata-rata pertama (momentum) dan kedua dari gradien untuk menyesuaikan laju pembelajaran selama pelatihan.

Metrik untuk menentukan metrik evaluasi yang akan dipantau selama pelatihan dan evaluasi model. Akurasi adalah metrik umum untuk klasifikasi yang mengukur persentase prediksi yang benar dari total prediksi. Dengan memantau akurasi, kita dapat mengevaluasi kinerja model pada data pelatihan dan data validasi, sehingga memungkinkan untuk mengukur seberapa baik model mengenali pola dalam data dan membuat prediksi yang benar.

2.6 Python

Python adalah bahasa pemrograman yang populer. Bahasa ini diciptakan oleh Guido van Rossum dan diperkenalkan pada tahun 1991. Dari gambar 2.9 Python adalah bahasa pemrograman yang mudah dipelajari. Sejauh ini, Python digunakan di hampir semua bidang seperti game, sistem web, dan bahkan dapat membuat mesin pencari sendiri. Pada umumnya, bahasa pemrograman ini digunakan untuk

desain website, pengembangan perangkat lunak, matematika, dan skrip sistem.



Gambar 2.9 Logo Python
(Sumber: (Movsession dan Nagaty, 2013))

2.7 Arduino IDE

Perangkat lunak Arduino yang digunakan adalah driver dan IDE. IDE atau Integrated Development Environment adalah sebuah program khusus untuk komputer Anda yang memungkinkan Anda untuk membuat sebuah proyek program atau sketsa untuk papan Arduino. Pada Gambar 2.10 Arduino IDE merupakan software yang sangat canggih yang ditulis dalam bahasa java (Risdiandi, 2021).



Gambar 2.10 Arduino IDE
(Sumber: (Lulu, 2022))

2.8 WO MIC Client

Gambar 2.11 merupakan ikon aplikasi *WO Mic Client*. *WO Mic Client* adalah aplikasi yang menggunakan perangkat *smartphone* sebagai *microphone* nirkabel untuk komputer atau perangkat lain. Dengan menggunakan aplikasi ini, dapat mengubah *smartphone* menjadi *microphone* eksternal yang terhubung ke komputer melalui koneksi Wi-Fi atau USB. *WO Mic Client* menyediakan berbagai fitur, seperti pengaturan sensitivitas *microphone*, pengaturan kualitas suara, dan dukungan untuk berbagai mode koneksi. Aplikasi ini berguna sebagai *microphone* tambahan atau jika *microphone* bawaan komputer tidak berfungsi dengan baik.



Gambar 2.11 WO MIC Client
(Sumber: (Buggs, 2023))

2.9 Jupyter Notebook

Seperti Gambar 2.12 Jupyter Notebook adalah salah satu IDE yang populer di kalangan *data scientist* dalam menulis kode Python (Arvyantomo dan Ratama, 2023). Terdapat banyak cara dalam melakukan instalasi Jupyter Notebook, yang termudah melewati Anaconda. Anaconda merupakan suatu platform yang mendistribusikan bahasa Python khususnya dalam bidang *scientific computing* (*data mining, machine learning, predictive analysis* dan lainnya).



Gambar 2.12 Jupyter Notebook
(Sumber: (Oelsen, 2016))

2.10 Visual Studio Code

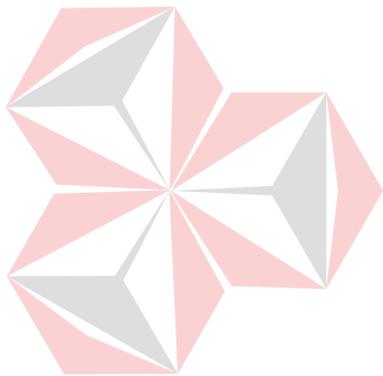
Visual Studio Code adalah aplikasi editor kode sumber terbuka yang dikembangkan oleh Microsoft untuk Windows, Linu-, dan MacOS. Ini memudahkan penulisan kode yang mendukung berbagai bahasa pemrograman, termasuk C++, C#, Java, Python, PHP, dan GO. Dari Gambar 2.13 *Visual Studio Code* dapat mengenali jenis bahasa pemrograman yang digunakan dan memberikan warna yang berbeda untuk fungsi-fungsi dalam kode tersebut. Selain itu, *Visual Studio Code* terintegrasi dengan Github dan memungkinkan pengguna untuk menambahkan ekstensi untuk meningkatkan fitur yang tidak tersedia secara default.



Gambar 2.13 Visual Studio Code
(Sumber: (Koralage, 2020))

2.11 Waveform

WAV adalah format audio standar yang dikembangkan oleh Microsoft dan IBM untuk komputer pribadi (PC). Biasanya, format ini menggunakan kode PCM (Pulse Code Modulation). WAV merupakan data audio yang tidak terkompres, sehingga seluruh sampel audio disimpan utuh di harddisk. Software seperti Windows Sound Recorder dapat membuat file WAV dari suara analog. File audio dalam format WAV jarang digunakan di internet karena ukurannya yang relatif besar, dengan batasan maksimal ukuran file WAV adalah 2GB (Fuad dan Winata, 2017).



UNIVERSITAS
Dinamika

BAB III METODOLOGI PENELITIAN

3.1 Perangkat Keras

Gambar 3.1 merupakan alur kerja dalam bentuk blok diagram. Input berasal dari *microphone smartphone* yang menggunakan aplikasi *WO Mic Client* dikarenakan lebih bebas dan mudah digunakan. *WO Mic Client* berguna untuk menggantikan *microphone* pada laptop yang terdapat terhubung melalui sambungan Wi-Fi. Setelah perintah suara input didapatkan maka akan diproses menggunakan sistem *audio classification* dan MFCC sebagai ekstraksi fitur suara. Setelah diproses maka output dikirim ke mikrokontroler ESP32 melalui *wireless* dan dilakukan aksi berupa mengatur arah pergerakan *mobile robot* dari deteksi suara berdasarkan perintah suara *artificial intelligence* yang dilatih.



Gambar 3.1 Blok Diagram Perangkat Keras

Kemudian pada diagram skematik mikrokontroler ESP32 dihubungkan ke motor dc. Selanjutnya motor dc akan mengatur arah *mobile robot* dengan cara maju dan mundur kedua motor dc menyala, belok kanan motor dc kanan mati dan motor dc kiri menyala, belok kiri motor dc kiri mati dan motor dc kanan menyala, berhenti kedua motor dc mati.



Gambar 3.2 Model Perancangan Hardware

Pada Gambar 3.2 di atas ini merupakan model perancangan perangkat keras yang akan dilakukan penelitian dalam Tugas Akhir ini, cara kerja perangkat keras tersebut pertama-tama pengguna mengucapkan perintah suara melalui *microphone smartphone* dari aplikasi *WO Mic Client* yang kemudian menjadi pengganti *microphone laptop*. Proses selanjutnya adalah mengolah suara tersebut dengan MFCC menggunakan fitur *audio classification* pada laptop yang memiliki lima label yaitu maju, mundur, kanan, kiri, dan berhenti, jika perintah suara dari pengguna “maju” maka akan terdeteksi maju, jika perintah suara dari pengguna “mundur” maka akan terdeteksi mundur, jika perintah suara dari pengguna “kanan” maka akan terdeteksi kanan, jika perintah suara dari pengguna “kiri” maka akan terdeteksi kiri, dan jika perintah suara dari pengguna “berhenti” maka akan terdeteksi berhenti.

3.2 Instalasi Environment

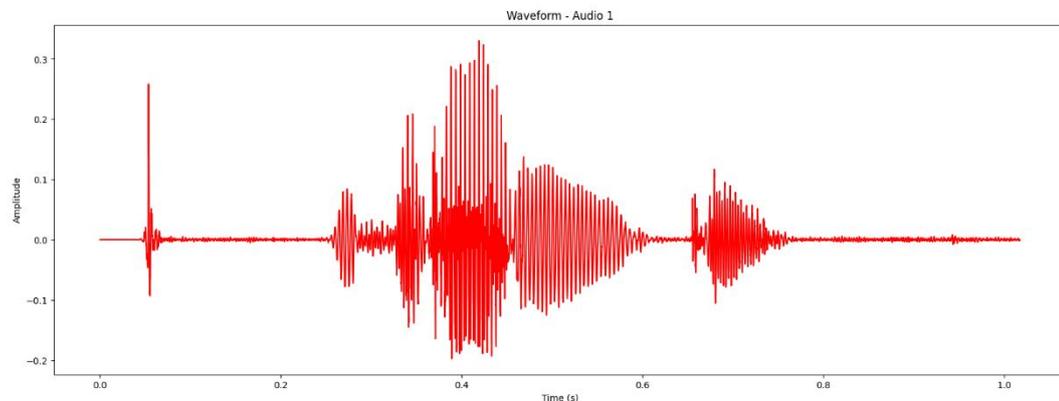
Instalasi dan pengaturan *environment* adalah hal pertama sebelum melakukan ke langkah selanjutnya. Seperti menyiapkan *library-library* yang digunakan agar program tidak terkendala. Menginstall Python dan memasang *library* yang cocok dengan versi tersebut lewat terminal pip, Anaconda, atau Visual Studio Code. Menambahkan library String pada Arduino IDE untuk ESP32 dapat membaca data yang diterima. Menggunakan library WiFi untuk dapat tersambung ke jaringan WiFi dan library WiFiUdp untuk dapat melakukan komunikasi antara laptop dengan ESP32 menggunakan jaringan WiFi yang sama.

3.3 Data Pre-paration

3.3.1 Karakteristik Audio

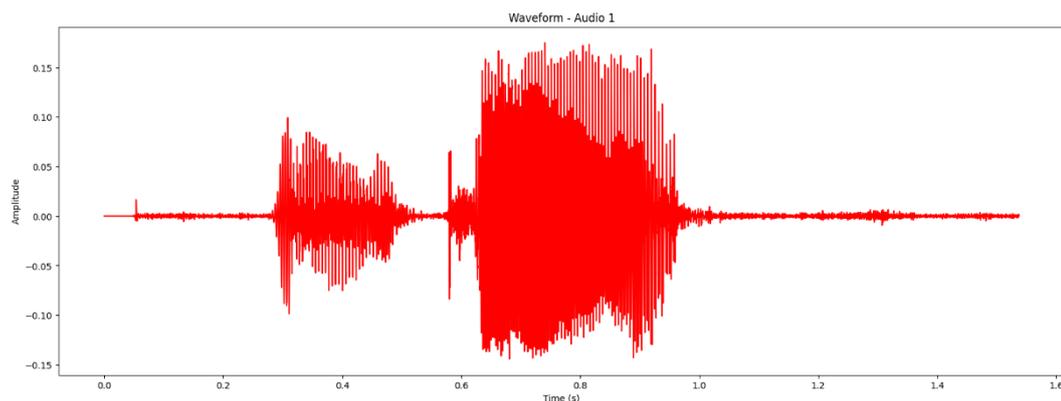
Mengetahui karakteristik audio merupakan langkah yang sangat penting. Proses yang dilakukan untuk melakukan pemantauan menggunakan *waveform* dan *spectrogram* untuk merepresentasikan secara virtual dari sinyal audio yang penting untuk memberikan pemahaman yang lebih baik dari ciri-ciri unik yang terdapat dalam data audio. Melalui visual *waveform* pemantauan terhadap naik turun amplitudo terhadap waktu yang dilihat dengan jelas. Hal ini memberikan Gambaran

tentang perubahan secara dinamis dalam durasi, dan variasi amplitudo yang mempengaruhi karakteristik audio secara keseluruhan.



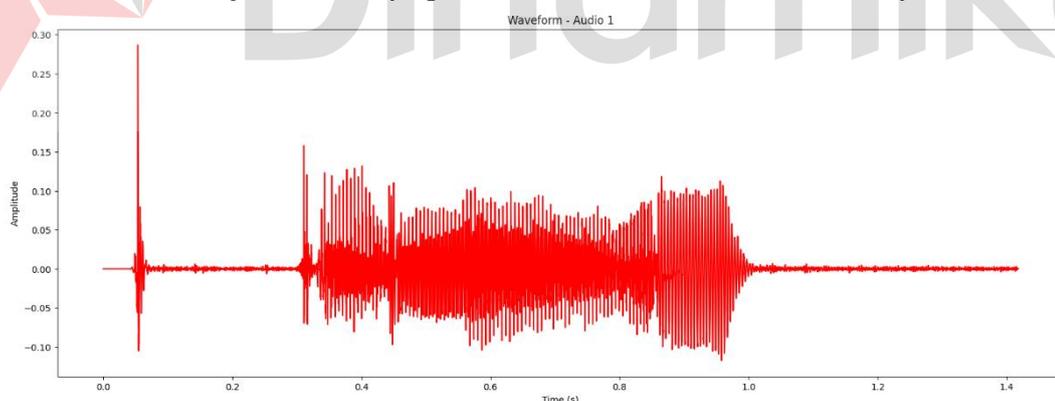
Gambar 3.3 *Waveform* berhenti age.wav

Pada Gambar 3.3 di atas merupakan gambar *waveform* dari file audio berhenti age.wav yang terdapat 6 bagian sinyal. Sumbu x pada gambar tersebut merupakan nilai *time* atau waktu dengan satuan detik dan untuk sumbu y merupakan nilai *amplitude* atau amplitudo. Satuan amplitudo umumnya dinyatakan dalam nilai desibel (dB) atau nilai numerik. Nilai decibel adalah satuan logaritmik yang umumnya digunakan untuk mengukur rasio antara dua besaran fisik, termasuk daya atau intensitas suara. Sedangkan satuan yang digunakan dalam gambar *waveform* adalah nilai numerik. Nilai numerik secara langsung mengukur simpangan maksimum dari suatu gelombang suara dari posisi kesetimbangannya yang tidak memiliki satuan baku, seringkali dinormalisasi ke rentang tertentu seperti -1 hingga 1 atau 0 hingga 1. Durasi dari rekaman file audio sepanjang 1 detik dan memiliki beragam variasi amplitudo. Pada bagian awal sinyal yaitu detik ke 0 sampai 0,2 terdapat puncak amplitudo yang signifikan pada detik ke 0,1. Bagian tengah sinyal yaitu detik ke 0,2 sampai 0,6 terdapat amplitudo yang lebih besar dan lebih bervariasi yang merepresentasikan ucapan. Bagian akhir sinyal yaitu detik ke 0,6 sampai 1 detik terdapat amplitudo yang berkurang dan stabil yang menunjukkan bahwa suara mulai hilang atau menurun intensitasnya. Amplitudo memiliki dua nilai yaitu minimal dan maksimal, nilai minimal amplitudo pada gambar 3.3. adalah lebih dari -0,2 dan untuk nilai maksimal amplitudonya adalah kurang lebih 0,3 yang terjadi pada waktu 0,4 detik.



Gambar 3.4 *Wavefrom* maju age.wav

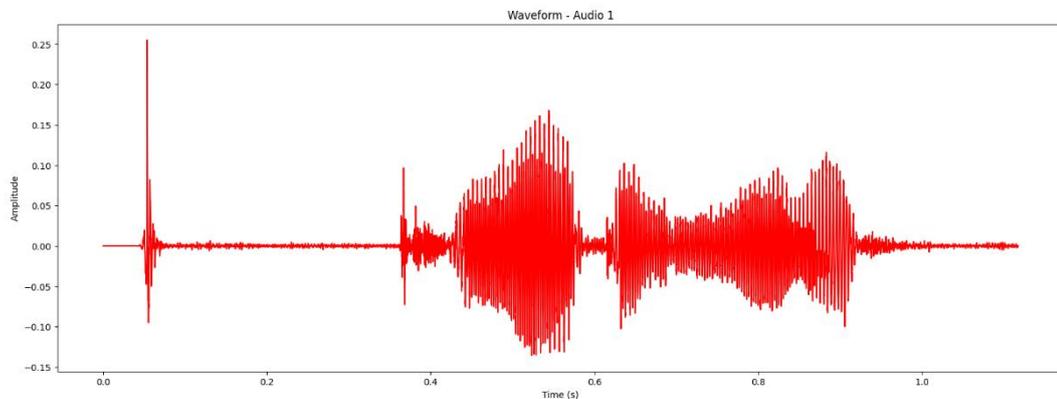
Pada Gambar 3.4 di atas merupakan gambar *wavefrom* dari file audio maju age.wav. Pada awal (0-0.4 detik) dan akhir (1.1-1.6 detik), amplitudo berada di sekitar nol, yang menunjukkan adanya jeda atau bagian yang sangat tenang dari sinyal audio. Di antara 0.4 detik hingga 1.0 detik, amplitudo meningkat drastis, menunjukkan adanya aktivitas suara yang signifikan. Ini adalah bagian di mana sinyal audio memiliki energi tertinggi. Amplitudo tertinggi berada sekitar 0,15 dan terendah sekitar -0,15. Puncak ini menunjukkan momen di mana sinyal audio mencapai energi maksimumnya. Variasi amplitudo berkisar -0,15 hingga 0,15. Variasi ini menunjukkan adanya perubahan intensitas suara dalam sinyal audio.



Gambar 3.5 *Wavefrom* kanan age.wav

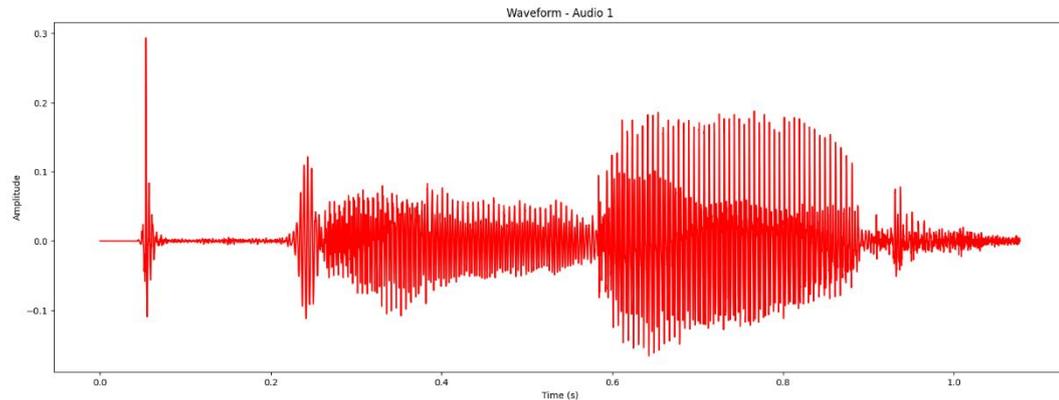
Pada Gambar 3.5 di atas merupakan gambar *wavefrom* dari file audio kanan age.wav. Pada awal sinyal (0-0.1 detik) dan akhir sinyal (sekitar 1.0-1.4 detik), amplitudo berada di sekitar nol, menunjukkan adanya jeda atau bagian yang sangat tenang dari sinyal audio. Di antara 0.4 detik hingga 1.0 detik, amplitudo meningkat secara signifikan, menunjukkan adanya aktivitas suara yang lebih signifikan. Pada

awal sinyal, terdapat lonjakan amplitudo yang sangat tinggi mencapai sekitar 0,30. Ini mungkin menunjukkan adanya peristiwa impulsif seperti tepukan atau ledakan suara. Setelah lonjakan awal, amplitudo bervariasi antara sekitar -0,10 hingga 0,30. Variasi ini menunjukkan adanya perubahan intensitas suara dalam sinyal audio.



Gambar 3.6 *Waveform* kiri age.wav

Pada Gambar 3.6 di atas merupakan gambar *waveform* dari file audio kiri age.wav. Pada awal grafik (sekitar 0.05 detik), terdapat sebuah puncak amplitudo yang sangat tinggi, sekitar 0,25. Ini menunjukkan bahwa ada sebuah sinyal yang sangat kuat atau impuls pada waktu tersebut. Setelah puncak awal, ada periode singkat dengan amplitudo yang rendah, menunjukkan periode relatif tenang atau sinyal lemah. Sekitar 0.4 detik, amplitudo meningkat lagi tetapi tidak setinggi puncak awal, ini menunjukkan adanya sinyal atau aktivitas yang signifikan pada waktu tersebut. Dari sekitar 0.5 detik hingga 1 detik, terdapat aktivitas gelombang utama dengan amplitudo yang berfluktuasi antara 0,25 dan -0,15. Ini menunjukkan bahwa terdapat sinyal yang relatif kuat dengan pola yang cukup kompleks selama periode ini. Setelah 1 detik, amplitudo menurun lagi menuju nilai nol, menunjukkan akhir dari sinyal atau aktivitas.



Gambar 3.7 *Wavefrom* mundur age.wav

Pada Gambar 3.7 di atas merupakan gambar *wavefrom* dari file audio mundur age.wav yang terdapat 6 bagian sinyal. Pada awal grafik, sekitar 0.05 detik, terdapat sebuah puncak amplitudo yang sangat tinggi mencapai sekitar 0,3. Ini menunjukkan adanya sinyal impuls atau kejutan yang sangat kuat pada waktu tersebut. Setelah puncak awal, ada periode dengan amplitudo yang relatif rendah hingga sekitar 0.3 detik. Ini menunjukkan periode tenang atau sinyal yang lemah. Mulai dari sekitar 0.3 detik hingga 1 detik, terdapat aktivitas gelombang utama dengan amplitudo yang berfluktuasi antara 0,3 dan -0,3. Puncak amplitudo tertinggi dalam periode ini terjadi sekitar 0.6 detik, dengan amplitudo sekitar 0,2. Aktivitas gelombang utama menunjukkan adanya sinyal yang kuat dan beragam dengan variasi amplitudo yang signifikan.

```
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np

filename1 = 'dataset TA/sudah filter/kiri age.wav'
y1, sr1 = librosa.load(filename1, sr=None)

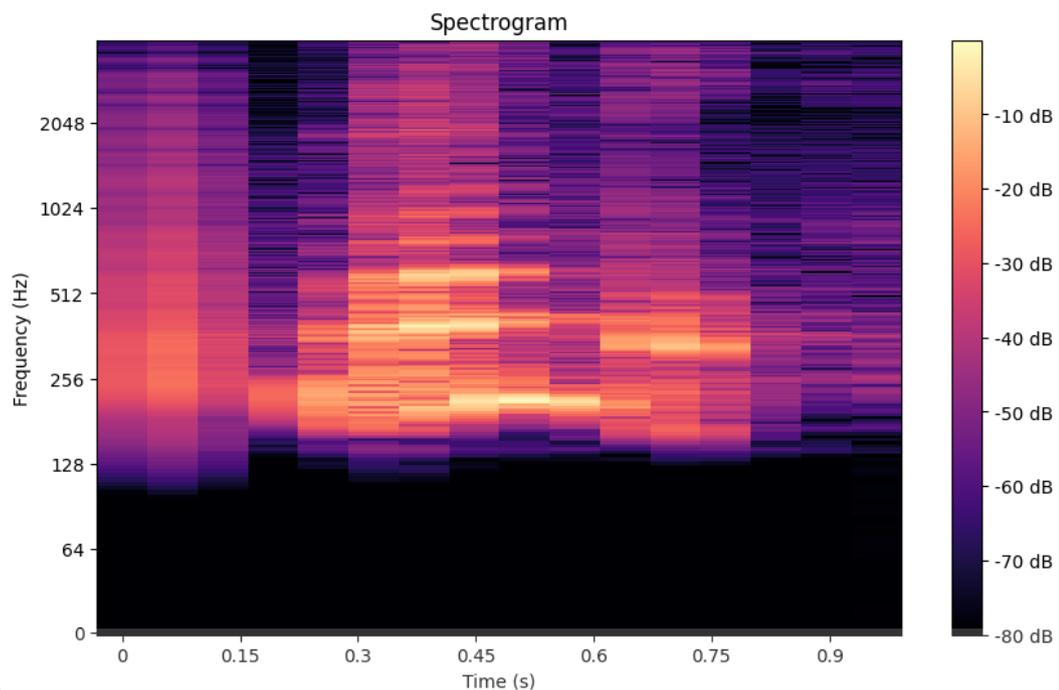
plt.figure(figsize=(20, 7))

times1 = np.arange(len(y1)) / sr1
plt.plot(times1, y1, label='Audio 1', color='red')
plt.title('Waveform - Audio 1')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()
```

Gambar 3.8 Program Membuat *Wavefrom*

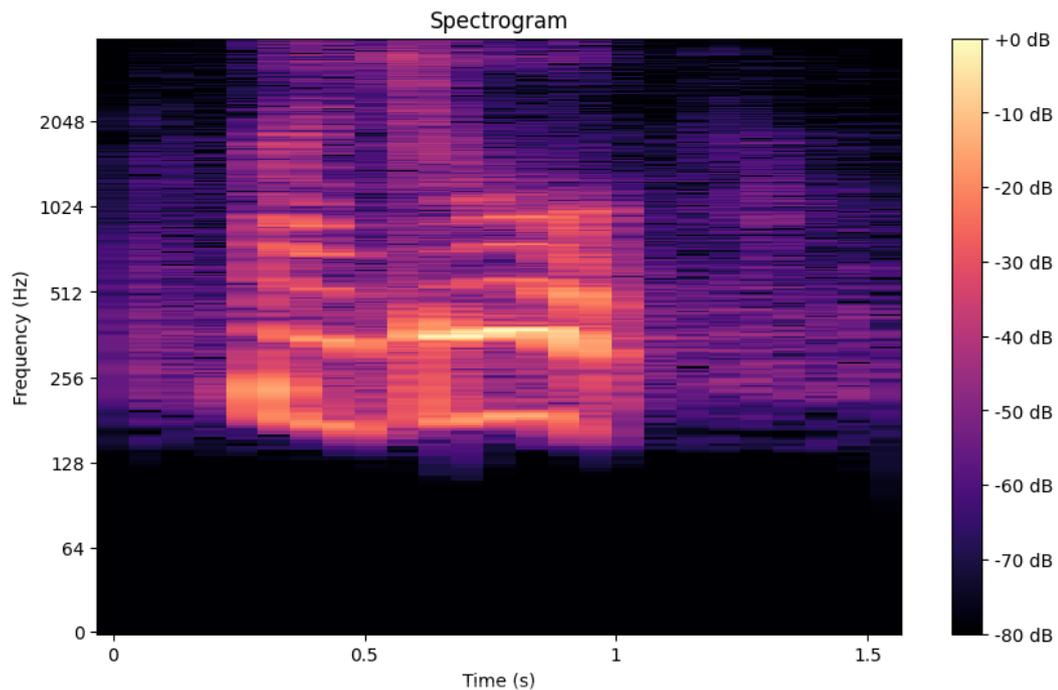
Pada Gambar 3.8 di atas merupakan program untuk membuat dan menampilkan *waveform* dari file audio berhenti *age.wav*. Menggunakan pustaka *librosa* untuk memuat file audio, pustaka *matplotlib* untuk menampilkan grafik, dan pustaka *numpy* untuk manipulasi array. Pertama, dua pustaka penting diimpor: *librosa*, yang menyediakan alat untuk analisis audio, dan *matplotlib.pyplot*, yang digunakan untuk membuat visualisasi data. *numpy* diimpor sebagai *np* untuk memudahkan operasi numerik. Menetapkan variabel *filename1* dengan nilai *path* atau lokasi file audio yang akan dimuat. Menggunakan fungsi *librosa.load* untuk memuat file audio yang *path*-nya disimpan di *filename1*. Fungsi ini mengembalikan dua nilai: *y1*, yaitu array yang berisi data amplitudo audio, dan *sr1*, yaitu nilai *sample rate* dari file audio tersebut. Parameter *sr=None* berarti bahwa file audio akan dimuat dengan *sample rate* aslinya tanpa *resampling*. Membuat *figure* dengan ukuran lebar 20 inci dan tinggi 7 inci sebagai *kanvas* untuk plot. membuat array *times1* yang berisi waktu (dalam detik) untuk setiap sampel dalam *y1*. *np.arange(len(y1))* menghasilkan array yang berisi indeks dari setiap sampel dalam *y1*, kemudian setiap indeks dibagi dengan *sr1* untuk mengkonversi indeks menjadi waktu. membuat plot dari amplitudo audio (*y1*) terhadap waktu (*times1*). *label='Audio 1'* memberikan label pada plot ini, dan *color='red'* menentukan warna garis plot sebagai merah. Memberikan judul pada plot sebagai 'Waveform - Audio 1', memberikan label pada sumbu-x plot sebagai 'Time (s)', yang menunjukkan bahwa sumbu ini mewakili waktu dalam satuan detik. Memberikan label pada sumbu-y plot sebagai 'Amplitude', yang menunjukkan bahwa sumbu ini mewakili amplitudo sinyal audio. Menampilkan plot yang telah dibuat dan menampilkan *figure* yang telah diatur sebelumnya.

Spektrogram adalah alat dasar yang dapat digunakan untuk menganalisis spektral suara dan bidang lainnya. Umumnya spektrogram berbentuk diagram geometris dua dimensi yang sumbu x mewakili waktu dan sumbu y mewakili frekuensi. Terdapat juga dimensi ketiga berupa amplitudo frekuensi tertentu pada saat waktu tertentu yang ditunjukkan dengan tingkatan dan warna pada gambar. Menurut (Savitri, 2023) spektrogram merupakan suatu bentuk nilai dari setiap format, yang dilengkapi dengan tingkat energi dan perubahannya terhadap waktu.



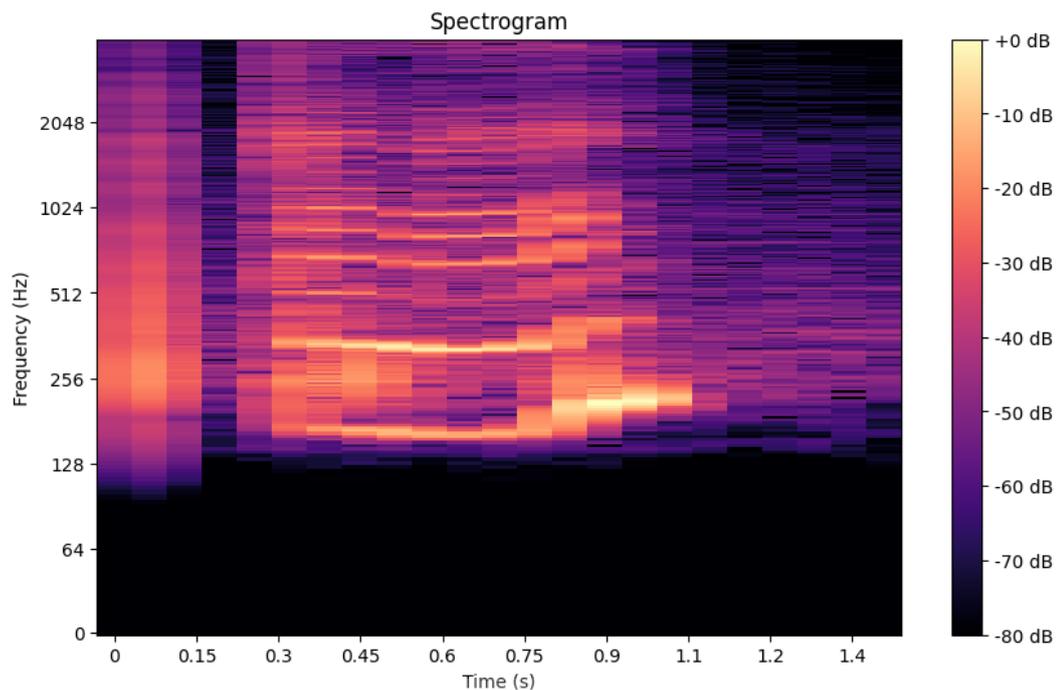
Gambar 3.9 Spectrogram berhenti age.wav

Pada Gambar 3.9 di atas merupakan gambar spektrogram dari file audio berhenti age.wav. Durasi dari rekaman file audio sepanjang 1 detik, frekuensi dari rentang 0 sampai 4096, dan variasi amplitudo dari 0 sampai -80 dB. Pada awal spektrogram (0-0,15 detik), intensitas di frekuensi rendah hingga menengah cukup tinggi, tetapi setelahnya terjadi variasi pola intensitas yang menunjukkan perubahan dalam sinyal audio. Sinyal audio yang dianalisis memiliki komponen frekuensi rendah yang dominan, terutama pada sekitar 128 Hz. Frekuensi menengah menunjukkan variasi dengan beberapa puncak intensitas, menunjukkan adanya harmonik atau variasi nada dalam sinyal. Frekuensi tinggi kurang dominan, menunjukkan bahwa sinyal ini mungkin lebih fokus pada komponen frekuensi rendah hingga menengah. Perubahan pola intensitas seiring waktu menunjukkan adanya variasi dinamis dalam sinyal audio, mungkin karena perubahan nada atau volume.



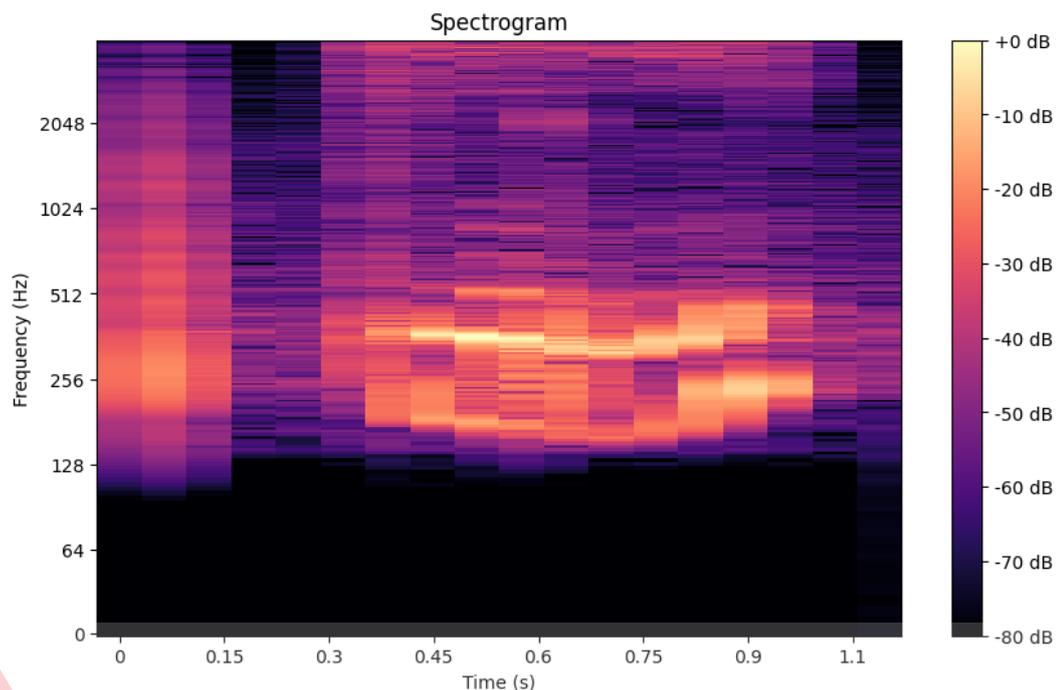
Gambar 3.10 *Spectrogram* maju age.wav

Pada Gambar 3.10 di atas merupakan gambar spektrogram dari file audio maju age.wav. Spektrogram menunjukkan sinyal dari 0 detik hingga sekitar 1.5 detik. Rentang frekuensi yang ditampilkan adalah dari 0 Hz hingga sekitar 2048 Hz. Pada sekitar 0 hingga 0.5 detik, terdapat beberapa pita frekuensi yang kuat, terutama di bawah 512 Hz. Antara 0.5 hingga 1 detik, pita frekuensi yang kuat terlihat lebih terfokus di frekuensi yang lebih rendah, antara 128 Hz hingga 512 Hz, dengan beberapa sinyal kuat di sekitar 256 Hz. Setelah 1 detik, kekuatan sinyal mulai berkurang, tetapi masih ada beberapa pita frekuensi yang kuat di sekitar 256 Hz hingga 512 Hz. Spektrogram menunjukkan adanya beberapa pita frekuensi yang tetap kuat selama periode waktu tertentu, menunjukkan sinyal periodik atau harmonis.



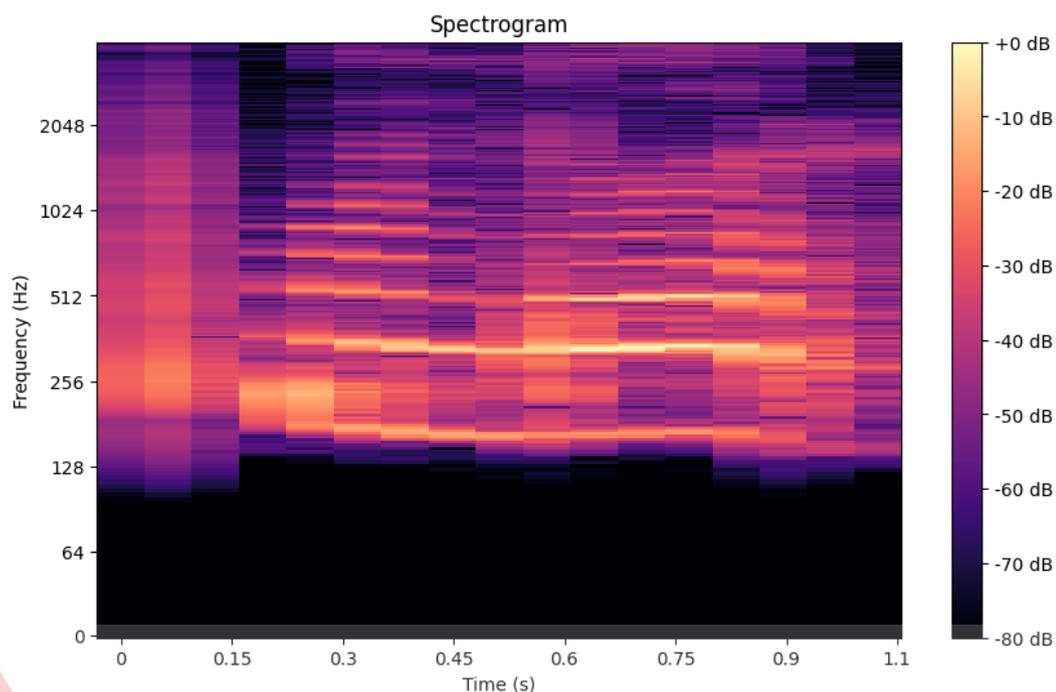
Gambar 3.11 *Spectrogram* kanan age.wav

Pada Gambar 3.11 di atas merupakan gambar spektrogram dari file audio kanan age.wav. Panjang sumbu X menunjukkan durasi total sinyal, yaitu sekitar 1,4 detik dalam kasus ini. Dengan mengamati perubahan warna sepanjang sumbu waktu, kita dapat melihat bagaimana karakteristik suara berubah seiring waktu seperti perubahan yang signifikan dalam spektrum frekuensi sekitar 0,6 detik. Intensitas warna pada setiap titik menunjukkan bagaimana amplitudo berubah seiring waktu dan frekuensi. Area yang lebih terang menunjukkan suara yang lebih keras, sedangkan area yang lebih gelap menunjukkan suara yang lebih lemah. Garis-garis horizontal yang lebih terang menunjukkan frekuensi dominan pada waktu tertentu. Misalnya, pada sekitar 0,5 detik, terdapat beberapa frekuensi dominan di sekitar 512 Hz dan 1024 Hz. Amplitudo dapat berubah secara tiba-tiba, yang mengindikasikan adanya peristiwa suara yang singkat dan kuat. Ini terlihat sebagai perubahan warna yang mendadak pada beberapa titik.



Gambar 3.12 *Spectrogram* kiri age.wav

Pada Gambar 3.12 di atas merupakan gambar spektrogram dari file audio kiri age.wav. Sumbu x dari spektrogram menunjukkan waktu dalam detik, dari 0 hingga sekitar 1.1 detik. Ini menunjukkan bagaimana frekuensi dan intensitas sinyal berubah sepanjang waktu. Ada perubahan intensitas frekuensi yang signifikan sepanjang waktu, dengan beberapa puncak intensitas yang lebih jelas terlihat di sekitar 0.4 detik hingga 1.0 detik. Bagian dengan warna cerah (orange dan kuning) menunjukkan frekuensi dan waktu di mana sinyal memiliki intensitas yang tinggi. Misalnya, sekitar 0.1 detik ada lonjakan intensitas yang tinggi di frekuensi yang lebih tinggi. Pada awal sinyal, sekitar 0 hingga 0.15 detik, terdapat lonjakan intensitas yang sangat tinggi pada berbagai frekuensi. Ini mungkin menunjukkan peristiwa impulsif seperti ledakan suara atau tepukan. Dari 0.4 hingga 1.0 detik, ada beberapa frekuensi konsisten yang menunjukkan aktivitas suara yang teratur, seperti ucapan atau musik. Spektrogram menunjukkan sedikit atau tidak ada aktivitas pada frekuensi yang sangat rendah (0-64 Hz), yang biasanya merupakan noise atau komponen frekuensi sangat rendah dari sinyal.



Gambar 3.13 *Spectrogram* mundur age.wav

Pada Gambar 3.13 di atas merupakan gambar spektrogram dari file audio mundur age.wav. Sebagian besar energi sinyal terkonsentrasi pada frekuensi antara sekitar 128 Hz hingga 1024 Hz. Ada beberapa aktivitas pada frekuensi yang lebih tinggi, tetapi intensitasnya relatif lebih rendah. Ada perubahan intensitas frekuensi yang signifikan sepanjang waktu, dengan beberapa puncak intensitas yang lebih jelas terlihat di sekitar 0.15 detik hingga 1.0 detik. Bagian dengan warna cerah (oranye dan kuning) menunjukkan frekuensi dan waktu di mana sinyal memiliki intensitas yang tinggi. Misalnya, sekitar 0.15 detik ada lonjakan intensitas yang tinggi di berbagai frekuensi. Pada awal sinyal, sekitar 0 hingga 0.15 detik, terdapat lonjakan intensitas yang sangat tinggi pada berbagai frekuensi. Ini mungkin menunjukkan peristiwa impulsif seperti ledakan suara atau tepukan. Dari 0.15 hingga 1.0 detik, ada beberapa frekuensi konsisten yang menunjukkan aktivitas suara yang teratur, seperti ucapan atau musik.

Dengan menggunakan plot spektrum frekuensi ini, pemantauan terhadap dinamika frekuensi, pitch, dan elemen-elemen harmonis yang dapat dicapai. Perubahan frekuensi dominan terjadi perubahan yang mengakibatkan dinamika frekuensi, perubahan pitch terlihat pada detik ke 0 sampai 0,15 dan 0,75 sampai 1.

Dengan adanya gambar wavefrom dan spectrogram membuat pendekatan yang dilakukan untuk analisis dan memahami karakteristik audio.

```
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np

file_path = 'dataset TA/sudah filter/berhenti age.wav'
y, sr = librosa.load(file_path, sr=None)

D = librosa.stft(y)
S_db = librosa.amplitude_to_db(abs(D), ref=np.max)

plt.figure(figsize=(10, 6))
librosa.display.specshow(S_db, sr=sr, x_axis='time', y_axis='log')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.show()
```

Gambar 3.14 Program Membuat *Spectrogram*

Pada Gambar 3.14 di atas merupakan program untuk membuat dan menampilkan *spectrogram* dari file audio berhenti age.wav. Menggunakan pustaka librosa untuk analisis audio dan pustaka matplotlib untuk visualisasi data. Tujuan dari kode ini adalah untuk memuat file audio, menghitung spektogramnya, dan menampilkan hasilnya dalam bentuk grafik. Menetapkan variabel file_path dengan nilai path atau lokasi file audio yang akan dimuat. Fungsi librosa.load digunakan untuk memuat file audio yang path-nya disimpan di file_path. Fungsi ini mengembalikan dua nilai: y, yaitu array yang berisi data amplitudo audio, dan sr, yaitu nilai sample rate dari file audio tersebut. Parameter sr=None berarti bahwa file audio akan dimuat dengan sample rate aslinya tanpa resampling. Menghitung transformasi Fourier singkat (*Short-Time Fourier Transform* atau STFT) dari sinyal audio y. Hasilnya adalah array kompleks D yang berisi informasi frekuensi dan waktu dari sinyal audio. Amplitudo kompleks dari hasil STFT (abs(D)) dikonversi ke skala desibel. Fungsi librosa.amplitude_to_db digunakan untuk melakukan konversi ini, dan ref=np.max berarti nilai referensi yang digunakan adalah amplitudo maksimum, sehingga hasilnya diukur relatif terhadap amplitudo

maksimum tersebut. Membuat sebuah figure baru dengan ukuran 10 inci lebar dan 6 inci tinggi. Ini akan menjadi kanvas untuk plot spektrogram yang akan dibuat. Menampilkan spektrogram dalam skala desibel. Fungsi `librosa.display.specshow` digunakan untuk menampilkan array `S_db` sebagai sebuah spektrogram. Parameter `sr=sr` menetapkan sample rate dari sinyal audio, `x_axis='time'` menunjukkan bahwa sumbu-x mewakili waktu, dan `y_axis='log'` menunjukkan bahwa sumbu-y mewakili frekuensi dalam skala logaritmik. Menambahkan `colorbar` ke plot spektrogram untuk menunjukkan skala desibel dari nilai amplitudo. Parameter `format='%+2.0f dB'` menetapkan format label pada `colorbar`. Memberikan judul pada plot sebagai 'Spectrogram', memberikan label pada sumbu-x plot sebagai 'Time (s)', yang menunjukkan bahwa sumbu ini mewakili waktu dalam satuan detik, memberikan label pada sumbu-y plot sebagai 'Frequency (Hz)', yang menunjukkan bahwa sumbu ini mewakili frekuensi dalam satuan Hertz. Menampilkan plot yang telah dibuat dan menampilkan figure yang telah diatur sebelumnya.

Tahapan ini memainkan peran yang sangat penting dalam melakukan persiapan data yang membuat penelitian ini mengetahui pola-pola yang mendasari data audio sebelum memasuki tahap selanjutnya. Dengan demikian, pemantauan visual pada waveform dan spektrogram menjadi langkah inti dalam analisis audio yang lebih detail dan mendalam untuk kerangka penelian Tugas Akhir ini.

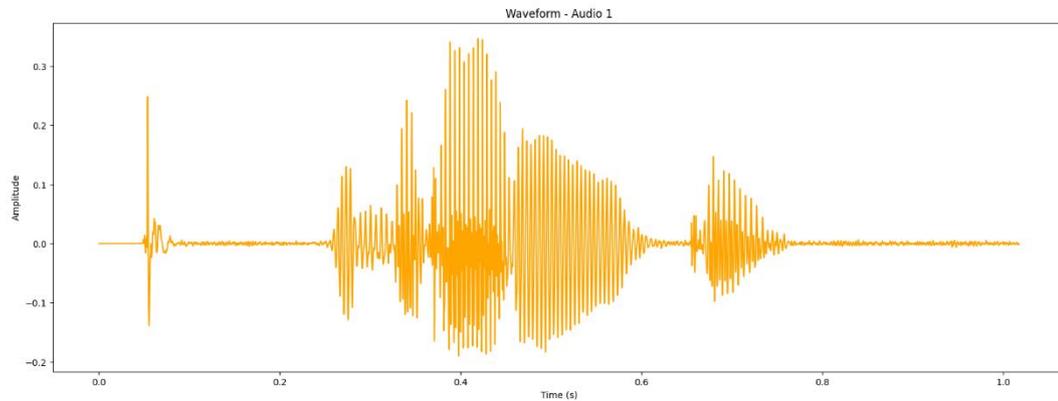
menghilangkan frekuensi rendah Ini memungkinkan pemrosesan sinyal yang lebih bersih dan fokus pada komponen frekuensi tinggi yang lebih penting untuk aplikasi tertentu, seperti rekaman audio atau komunikasi sinyal.

Low-pass filter dengan nilai parameter 800 Hz berfungsi untuk membiarkan sinyal dengan frekuensi di bawah 800 Hz melewati filter sambil mengurangi atau menghilangkan frekuensi yang lebih tinggi dari 800 Hz. Filter ini efektif dalam menyaring noise atau gangguan frekuensi tinggi yang tidak diinginkan, sehingga memungkinkan sinyal dengan komponen frekuensi rendah tetap utuh. Dalam aplikasi audio, *low-pass filter* ini sangat berguna untuk menghilangkan bunyi atau distorsi frekuensi tinggi yang bisa mengganggu kualitas audio, seperti pada rekaman musik atau sinyal komunikasi. Dengan menetapkan frekuensi cut-off pada 800 Hz, filter ini memastikan bahwa frekuensi tinggi di atas batas tersebut diredam secara progresif, sehingga hanya komponen frekuensi rendah yang dominan tetap diteruskan dan diperkuat.

Band-pass filter dengan nilai parameter 80 Hz dan 800 Hz dirancang untuk memungkinkan frekuensi sinyal dalam rentang antara 80 Hz dan 800 Hz melewati filter, sementara frekuensi yang berada di bawah 80 Hz dan di atas 800 Hz akan dikurangi atau dihilangkan. Dengan kata lain, filter ini memblokir sinyal di luar rentang frekuensi yang ditentukan, sehingga hanya frekuensi yang berada dalam batas tersebut yang dapat diteruskan. Dalam konteks aplikasi audio atau pemrosesan sinyal, *band-pass filter* ini sangat berguna untuk memfokuskan perhatian pada komponen frekuensi tertentu yang diinginkan, misalnya dalam pengolahan suara atau penyaringan sinyal untuk aplikasi komunikasi. Dengan memblokir frekuensi yang terlalu rendah atau terlalu tinggi, filter ini membantu mengurangi gangguan atau noise yang tidak relevan, dan memastikan bahwa hanya frekuensi yang berpotensi penting dan relevan dalam rentang 80 Hz hingga 800 Hz yang tetap terdengar atau terdeteksi.

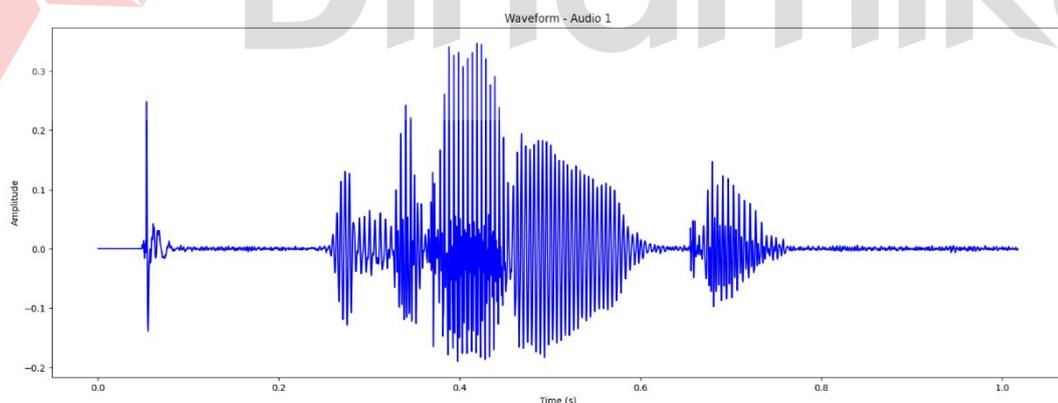
Dengan *filter high pass*, *filter band pass*, dan *filter low pass* memberikan berbagai jenis frekuensi pada dataset suara yang berbeda. *Filter low pass* menjamin frekuensi yang stabil, *filter high pass* menjamin stabil di udara dan *filter band pass* yang berada ditengah antara keduanya. Tujuannya untuk membuat model lebih beradaptasi dengan variasi lingkungan yang terjadi dalam kehidupan sehari-hari.

Untuk melakukan generalisasi kinerja model yang digunakan dalam penelitian ini, perlu menggunakan data tambahan.



Gambar 3.16 *Waveform* berhenti age_high_pass.wav

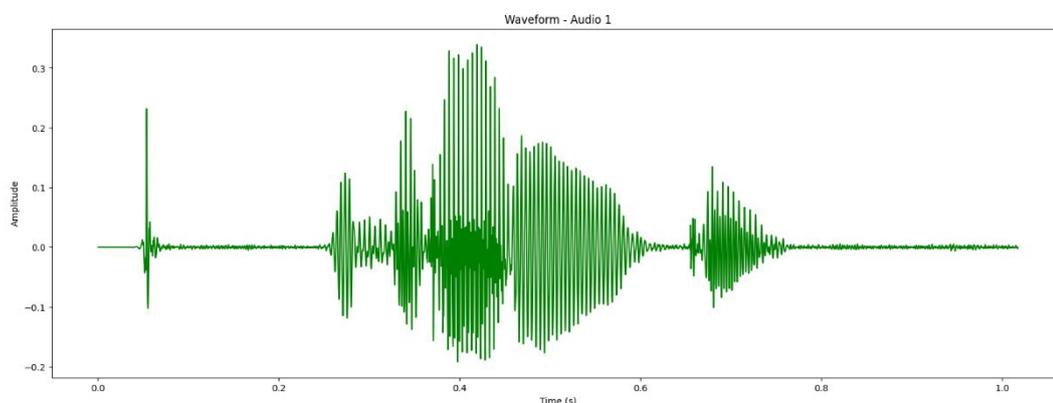
Pada Gambar 3.16 di atas merupakan gambar *waveform* dari file audio yang telah diterapkan *filter high pass*. Dari sumbu x merupakan nilai waktu dengan rentang waktu adalah 0 sampai 1 detik dan sumbu y merupakan nilai amplitudo dengan amplitudo yang berkisar -0,2 hingga 0,4. Setelah dilakukan *filter high pass* sinyal mengalami reduksi yang mengakibatkan nilai maksimal amplitudonya berkurang menjadi kurang dari 0,4 serta perbedaan yang sangat signifikan adalah reduksi sinyal pada bagian ke 2 dan pada bagian ke 5.



Gambar 3.17 *Waveform* berhenti age_low_pass.wav

Pada Gambar 3.17 di atas merupakan gambar *waveform* dari file audio yang telah diterapkan *filter low pass*. Dari sumbu x merupakan nilai waktu dengan rentang waktu adalah 0 sampai 1 detik dan sumbu y merupakan nilai amplitudo dengan amplitudo yang berkisar lebih dari -0,2 hingga lebih dari 0,4. Setelah

dilakukan *filter low pass* sinyal mengalami peningkatan pada beberapa bagian seperti peningkatan sinyal pada bagian ke 5.



Gambar 3.18 Waveform berhenti age_band_pass.wav

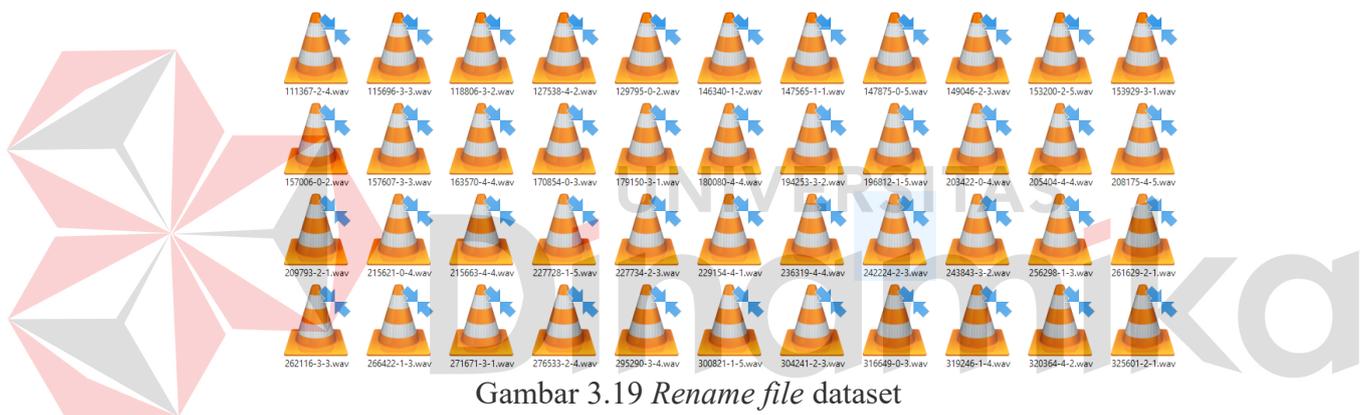
Pada Gambar 3.18 di atas merupakan gambar *waveform* dari file audio yang telah diterapkan *filter low pass*. Dari sumbu x merupakan nilai waktu dengan rentang waktu adalah 0 sampai 1 detik dan sumbu y merupakan nilai amplitudo dengan amplitudo yang berkisar lebih dari -0,2 hingga lebih dari 0,4. Setelah dilakukan *filter band pass* sinyal mengalami perubahan yang tidak terlalu signifikan.

Peneliti berusaha meningkatkan ketangguhan dan generalisasi model terhadap variasi yang mungkin terjadi dalam berbagai keadaan lingkungan menggunakan *filter high pass*, *filter medium pass*, dan *filter low pass* untuk meningkatkan informasi audio. Diharapkan bahwa komposisi ini akan membantu model mengidentifikasi dan memahami berbagai suara yang mungkin dipahami ketika peneliti mengembangkan sistem klasifikasi suara yang berguna. Total pengumpulan data yang telah dilakukan selama tahap augmentasi adalah 630 file yang terdiri 5 perintah suara.

3.3.4 Rename File Dataset

Setelah augmentasi data langkah selanjutnya yang dilakukan adalah melakukan *rename* atau penamaan pada setiap *file* audio dataset. Pada Gambar 3.19 peneliti menerapkan penamaan *file* dengan ketentuan perintah suara. Proses penamaan ini dilakukan memudahkan pengelolaan dan pengenalan klasifikasi dari setiap sampel suara selama proses pelatihan model.

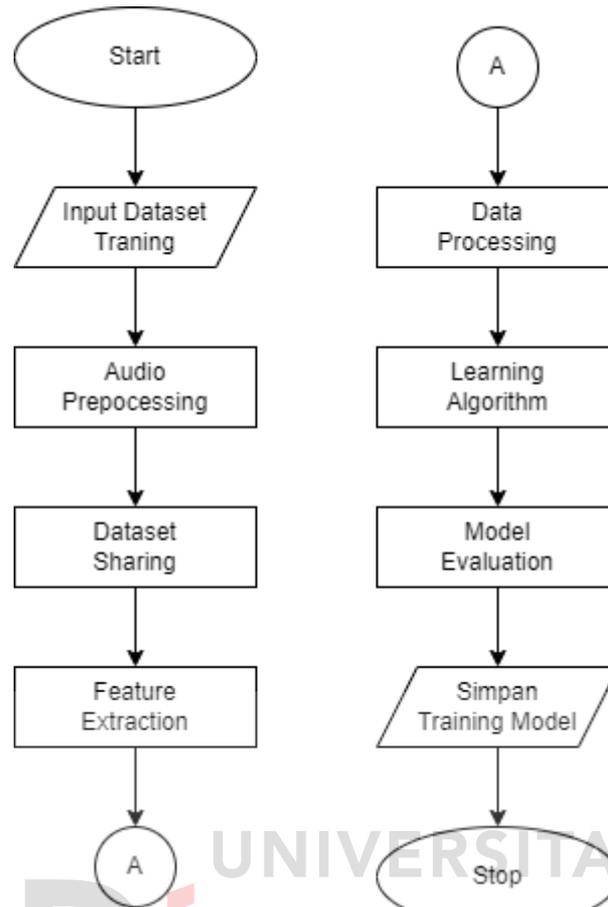
Dengan menggunakan metode penamaan, peneliti memastikan setiap file dalam dataset memiliki label yang jelas dan bisa untuk diidentifikasi. Hal ini akan membantu efisiensi dan keakuratan dalam membangun model pengenalan suara, karena setiap sampel suara terkait dengan perintah suara. Proses penamaan *file* dataset ini merupakan langkah krusial dalam mempersiapkan dataset sebelum dilakukan pelatihan dan pengujian model.



Gambar 3.19 Rename file dataset

3.3.5 Flowchart Dataset

Penelitian Tugas Akhir ini penulis menggunakan dataset suara *artificial intelligence* dan suara manusia dalam bentuk .WAV yang dibuat untuk mengendalikan arah *mobile robot*. Dalam pembuatan dataset audio terdapat beberapa langkah pembuatannya.



Gambar 3.20 *Training Dataset*

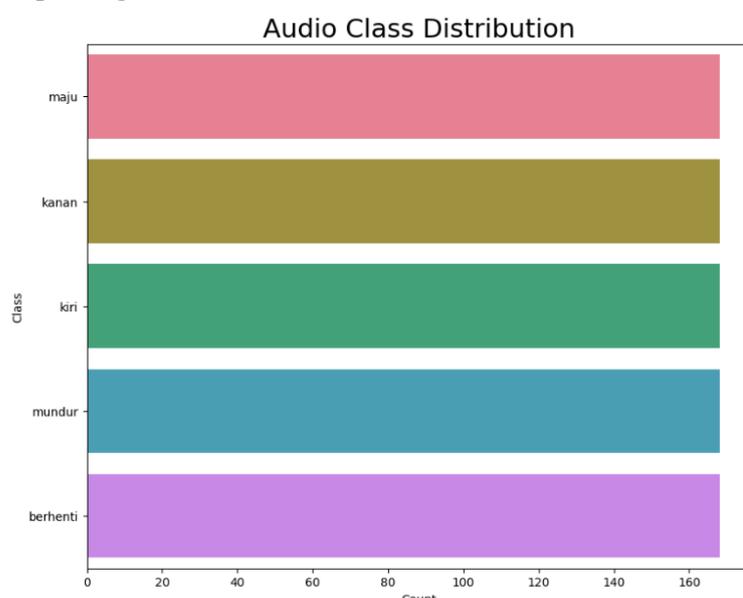
Pada Gambar 3.20 langkah pertama adalah pengumpulan data audio yang sesuai tujuan yang diperintah pada penelitian ini. Data audio ini diambil dengan menggunakan suara *artificial intelligence* dan suara manusia yang masukkan kata-kata sesuai perintah suara yang ada seperti “maju”, “kanan”, “kiri”, “mundur”, “berhenti”. Selanjutnya pemberian label untuk semua data audio agar terbagi pada setiap rekaman. Awal format audio seperti .WAV atau MP3 dirubah seperti *volume* dinormalisasi dan juga menghilangkan kebisingan pada latar belakang suara. Setelah disesuaikan dataset kan dibagi menjadi tiga yaitu set pelatihan, set validasi, dan set pengujian. Set pelatihan merupakan dataset yang digunakan untuk melatih model yang memiliki persentase 70% dari seluruh dataset. Set validasi digunakan untuk mengoptimasi parameter model sebelum dilakukan pengujian dan memiliki persentase 20% dari seluruh dataset. Set pengujian digunakan untuk mengukur

kinerja dari suatu model dengan menampilkan akurasi dan *loss* yang biasanya menggunakan persentase sebesar 10% dari seluruh dataset.

Setelah itu data audio diubah menjadi fitur numerik menggunakan MFCC agar bisa digunakan dalam model. MFCC atau *Mel-Frequency Cepstral Coefficients* adalah salah satu metode ekstraksi suara dalam melakukan analisis sinyal suara. Selanjutnya melakukan pelatihan pada model menggunakan data pelatihan yang sudah ada, dan model akan mengenali pola-pola dalam data suara selama proses pelatihan. Pada tahap-tahap validasi gunakan data validasi untuk melakukan penyetelan ulang pada parameter model data, seperti jumlah lapisan atau *layer*. Kemudian menggunakan set pengujian pada model untuk dievaluasi kemampuannya. Menggunakan hitungan metrik seperti akurasi, *recall*, presisi, *loss*, dan *F1-score* berfungsi untuk mengecek tinggi dan rendahnya kecocokan model dalam pengenalan suara. Jika model data memiliki kinerja yang baik maka bisa melakukan pengenalan perintah suara pada data baru.

3.3.6 Pengelompokan Data

Setelah melakukan tahap penamaan semua *file* dataset, pada Gambar 3.21 melakukan pengelompokan data ke dalam 5 kategori folder sesuai dengan perintah suara seperti maju, mundur, kanan, kiri, berhenti. Dengan melakukan pengelompokan, peneliti dapat memproses pelatihan dan evaluasi model untuk membuat setiap kategori suara lebih terstruktur.

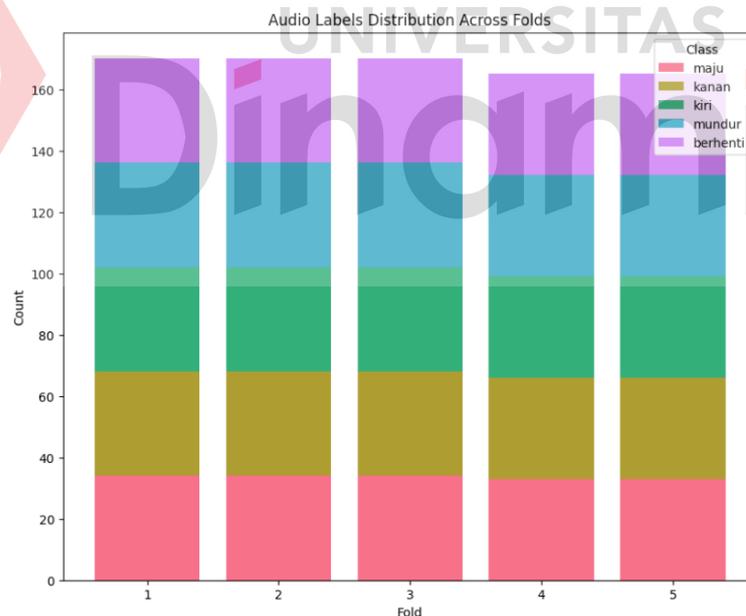
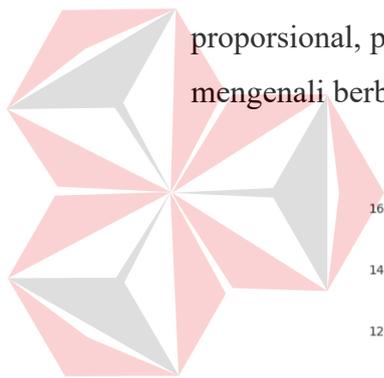


Gambar 3.21 Pengelompokan Data Berdasarkan Kategori

3.3.7 Distribusi Data secara Katagori

Setelah melakukan proses penamaan dataset selesai, pada Gambar 3.22 membagikan dataset ke dalam lima folder yang dinamai *folders*, yaitu *folders 1*, *folders 2*, *folders 3*, *folders 4*, *folders 5*. Pembagian data set dilakukan sesuai *datasheet* yang telah dibuat secara acak oleh program. Tujuan utama dari pembagian ke dalam folder *fold* adalah untuk memastikan setiap folder *fold* memiliki jumlah yang sama dari kategori perintah suara. Hal ini sangat penting untuk mencapai hasil evaluasi dan validasi model yang objektif.

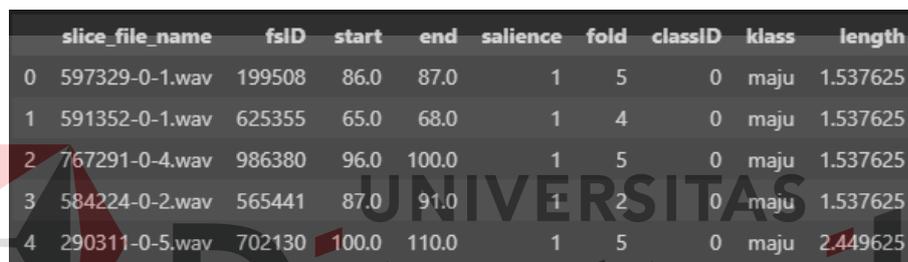
Dalam setiap *folders*, dataset akan terdiri dari kategori perintah “maju”, “mundur”, “kanan”, “kiri”, “berhenti”. Setiap kategori mempresentasikan tingkat suara yang sama dan memastikan bahwa setiap *folders* mencakup perintah suara yang ditemui dalam penggunaan. Dengan membagikan ke dalam *folders* secara proporsional, peneliti berharap dapat mencapai model yang lebih baik dan mampu mengenali berbagai jenis suara secara luas.



Gambar 3.22 Distribusi Folds Berdasarkan Class

3.3.8 Datasheet

Pada Gambar 3.23 merupakan *datasheet* atau bisa disebut juga lembaran data yang berisi tabel-tabel atribut penting untuk setiap sampel suara dalam dataset. Setiap baris tabel ini mencantumkan informasi tentang “*slice file name*” yaitu nama *file* yang dihasilkan setelah melakukan *augmentsai* dan penamaan, “*fsID*” yang merupakan identifikasi unit untuk setiap *file* audio, “*start*” dan “*end*” yang menunjukkan batas awal dan akhir audio, serta “*salience*” yang mencerminkan tingkat kepentingan dari setiap bagian. Selain itu, atribut “*fold*” digunakan untuk menunjukkan ke dalam *fold* mana suara tersebut dibagikan selama proses validasi atau evaluasi model. “*classID*” dan “*class*” mencakup informasi tentang kategori perintah arah gerak *mobile robot*, seperti “maju”, “mundur”, “kiri”, “kanan”, dan “berhenti”.



	slice_file_name	fsID	start	end	salience	fold	classID	class	length
0	597329-0-1.wav	199508	86.0	87.0	1	5	0	maju	1.537625
1	591352-0-1.wav	625355	65.0	68.0	1	4	0	maju	1.537625
2	767291-0-4.wav	986380	96.0	100.0	1	5	0	maju	1.537625
3	584224-0-2.wav	565441	87.0	91.0	1	2	0	maju	1.537625
4	290311-0-5.wav	702130	100.0	110.0	1	5	0	maju	2.449625

Gambar 3.23 *Datasheet* atau lembaran data

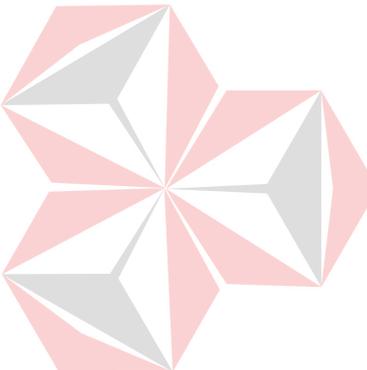
Setelah menyusun tabel dengan atribut tersebut, lembaran data disimpan dalam format *Comma-Separated Values* (CSV). Dengan menyimpan format tersebut memudahkan data untuk diolah dan dianalisis sehingga proses evaluasi serta pelatihan model lebih cepat diselesaikan. Pembuatan lembaran data menggunakan format CSV menjadi langkah krusial dalam pengolahan data yang akan digunakan pada langkah selanjutnya.

3.3.9 Distribusi *Folds Subset*

Dalam pembagian dataset ke dalam tiga subset utama: *train*, *test*, dan *validation*. “*x_train*” merupakan data fitur untuk pelatihan, “*x_test*” merupakan data fitur untuk pengujian, “*y_train*” merupakan label atau target fitur untuk pelatihan, “*y_test*” merupakan label atau target fitur untuk pengujian. Menurut (Nisa dan Candra, 2023) dengan menggunakan proporsi data 10% untuk pengujian, 20% untuk validasi 70% untuk pelatihan merupakan proporsi pembagian dataset

dengan tingkat akurasi yang tinggi. Pembagian ini bertujuan untuk membuat pelatihan model dengan optimal dengan menggunakan *train* data, menggunakan *test* data yang belum digunakan untuk menguji kinerja model, dan memvalidasi hasil model menggunakan *validation* data.

Distribusi ini dilakukan dengan teliti untuk membuat setiap *subset* mencakup representasi yang sama dari seluruh dataset termasuk perintah arah gerak *mobile* robot. Dengan membuat model yang memahami dan mengetahui suara secara efisien dalam semua kondisi. Proses distribusi ini dilakukan dengan memperhatikan proporsi dataset yang memadai untuk model diberikan variasi dalam melatih dan menguji. Langkah distribusi *folds* ini memainkan peran penting untuk mempersiapkan dataset sebelum langkah selanjutnya.



Dataset Split Sizes

	Subset	Number of Samples
0	Train	588
1	Validation	168
2	Test	84
3	Total	840

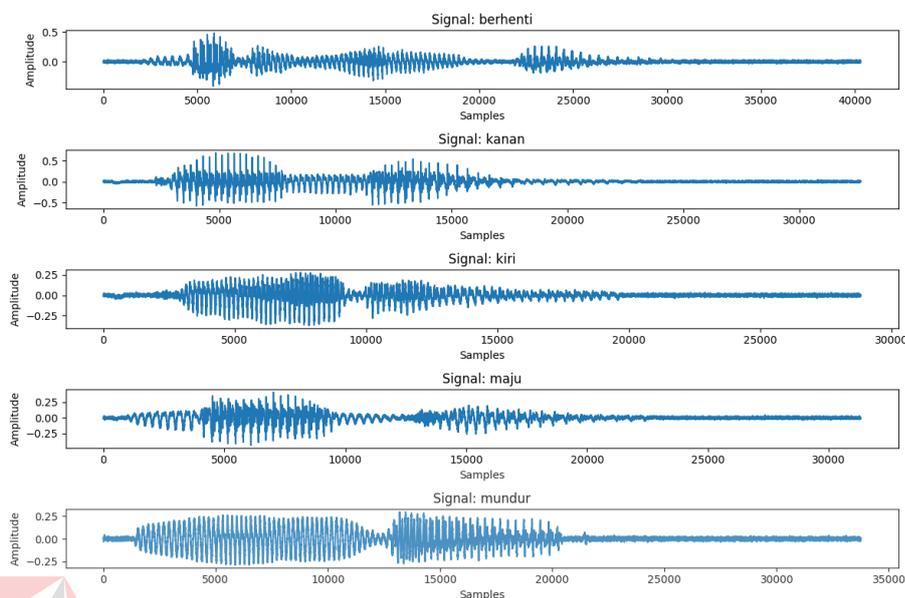
Gambar 3.24 Tabel *Subset* Dataset

Dari Gambar 3.24 setiap *subset* memiliki jumlah dataset yang berbeda pada bagian *train* memiliki 588 audio dataset, bagian *validation* memiliki 168 audio dataset, dan bagian *test* memiliki 84 audio dataset sehingga total dari tiga bagian terdapat 840 audio dataset.

3.4 Metode Ekstraksi Fitur dengan MFCC

Metode Ekstraksi Fitur dengan *Mel-Frequency Cepstral Coefficients* (MFCC) dimulai saat pemuatan sinyal suara akan dianalisis. Langkah pertama adalah penerapan filter pre-emphasis untuk meningkatkan kejelasan frekuensi

tinggi dalam sinyal. Filter ini dirancang untuk meningkatkan komponen frekuensi tinggi dalam sinyal. Selanjutnya sinyal menjadi diubah menjadi *frame-frame* kecil yang akan diterapkan fungsi *Hamming Window* untuk mengurangi efek sisa yang tidak diinginkan.

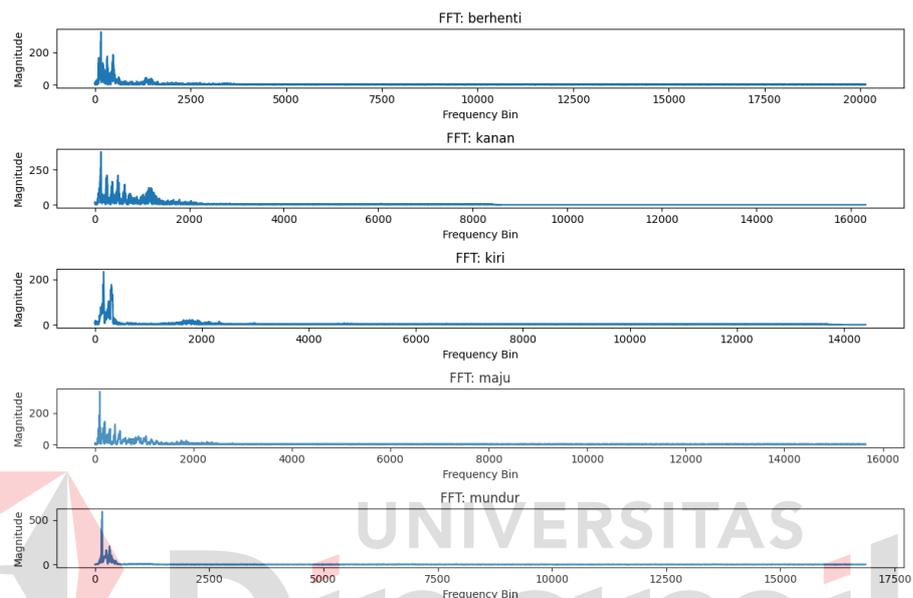


Gambar 3.25 Proses *Hamming Window*

Pada Gambar 3.25 *waveform* berhenti memiliki amplitudo puncak yang lebih tinggi dibandingkan dengan *waveform* lainnya, menunjukkan bahwa perintah "berhenti" diucapkan dengan volume yang lebih keras. Selain itu, bentuk gelombang mungkin menunjukkan adanya konsonan yang kuat pada awal atau akhir kata. *Waveform* kanan memiliki frekuensi fundamental yang lebih tinggi dibandingkan dengan *waveform* "kiri", karena perbedaan dalam artikulasi kedua kata tersebut. *Waveform* kiri memiliki lebih banyak variasi dalam amplitudo dibandingkan dengan *waveform* "kanan", karena adanya konsonan gesek seperti "k" yang dapat menghasilkan noise tambahan. *Waveform* maju memiliki durasi yang lebih pendek dibandingkan dengan *waveform* "mundur", karena kata "maju" memiliki lebih sedikit suku kata. *Waveform* mundur memiliki bentuk gelombang yang lebih kompleks dibandingkan dengan *waveform* lainnya, karena adanya konsonan gesek dan vokal yang panjang.

Setelah itu, dilakukan *Fast Fourier Transformation* (FFT) pada setiap *frame* untuk merubah dari domain waktu menjadi domain frekuensi. Spektrum frekuensi

yang dihasilkan akan diubah menjadi skala Mel. Skala Mel adalah skala frekuensi yang berdasarkan persepsi pendengaran manusia terhadap frekuensi suara. Diterapkan dengan menggunakan filter tringular yang overlapping untuk menghitung energi dalam interval frekuensi tertentu. Filter ini diposisikan sedemikian rupa untuk memperhitungkan sensitivitas pendengaran manusia terhadap frekuensi.

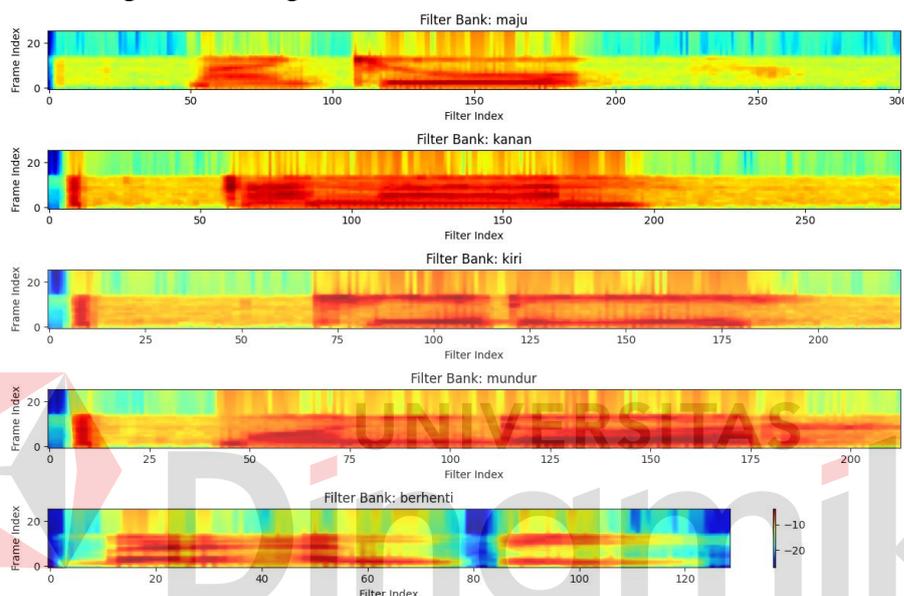


Gambar 3.26 Proses FFT

Pada Gambar 3.26 spektrum frekuensi berhenti menunjukkan adanya banyak komponen frekuensi yang kuat, terutama di rentang frekuensi rendah hingga menengah. Ini mengindikasikan bahwa suara "berhenti" memiliki banyak harmonik dan mungkin mengandung konsonan atau vokal yang kuat. Spektrum frekuensi kanan memiliki karakteristik yang agak berbeda dengan "berhenti". Meskipun masih terdapat banyak komponen frekuensi, distribusinya mungkin sedikit berbeda. Perbedaan ini bisa disebabkan oleh perbedaan artikulasi antara kedua kata. Spektrum frekuensi kiri cenderung memiliki lebih sedikit komponen frekuensi yang kuat dibandingkan dengan "berhenti" dan "kanan". Ini bisa mengindikasikan bahwa suara "kiri" memiliki karakteristik yang lebih sederhana atau kurang kompleks. Spektrum frekuensi maju menunjukkan distribusi energi yang cukup merata di beberapa rentang frekuensi. Ini mungkin mengindikasikan bahwa suara "maju" memiliki karakteristik yang lebih netral dibandingkan dengan perintah lainnya.

Spektrum frekuensi mundur memiliki puncak amplitudo yang lebih tinggi dibandingkan dengan spektrum lainnya, terutama pada frekuensi tertentu. Ini bisa mengindikasikan adanya frekuensi dominan yang khas untuk kata "mundur".

Setelah menghitung energi dari setiap filter Mel, langkah selanjutnya adalah menerapkan logaritma pada energi yang dihasilkan oleh setiap filter. Ini dilakukan untuk mengubah skala energi yang besar menjadi skala yang lebih mudah diinterpretasikan. Logaritma juga dapat membantu dalam menyeimbangkan kontribusi energi dari berbagai frekuensi.

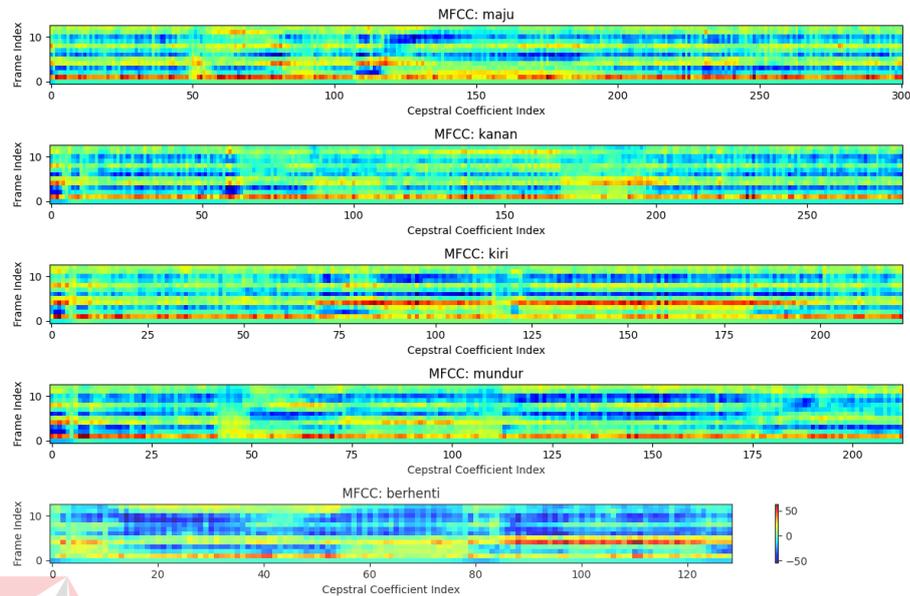


Gambar 3.27 Proses Filter Bank

Pada Gambar 3.27 distribusi energi untuk perintah “maju” dan “kanan” memiliki energi yang serupadengan energi terkonsentrasi pada beberapa rentang frekuensi tertentu. Ini menunjukkan bahwa kedua perintah ini memiliki karakteristik akustik yang mirip. Perintah “kiri” dan “mundur” juga memiliki kemiripan dalam distribusi energi, namun dengan beberapa perbedaan yang signifikan. Misalnya, perintah “kiri” mungkin memiliki lebih banyak energi pada frekuensi rendah dibandingkan dengan “mundur”. Perintah “berhenti” memiliki distribusi energi yang sangat berbeda dibandingkan dengan perintah lainnya. Energi terkonsentrasi pada rentang frekuensi yang lebih sempit, dan mungkin terdapat frekuensi dominan yang khas untuk perintah ini.

Langkah selanjutnya adalah penerapan *Discrete Cosine Transformation* (DCT) pada hasil logaritma tersebut. DCT digunakan setelah proses filter Mel dan

logaritma energi untuk menghasilkan koefisien *cepstral*. Biasanya, hanya sejumlah koefisien cepstral yang pertama yang dipertahankan untuk mengurangi dimensi data. Hasilnya adalah serangkaian koefisien *cepstral* MFCC yang lebih mudah diinterpretasikan dan digunakan untuk analisis lebih lanjut.



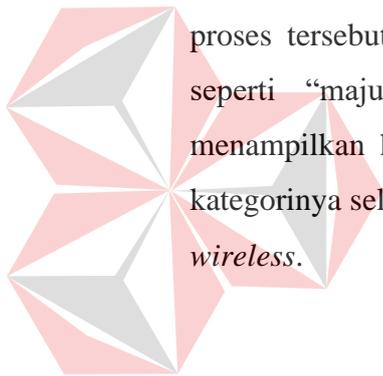
Gambar 3.28 Coefficient Cepstral

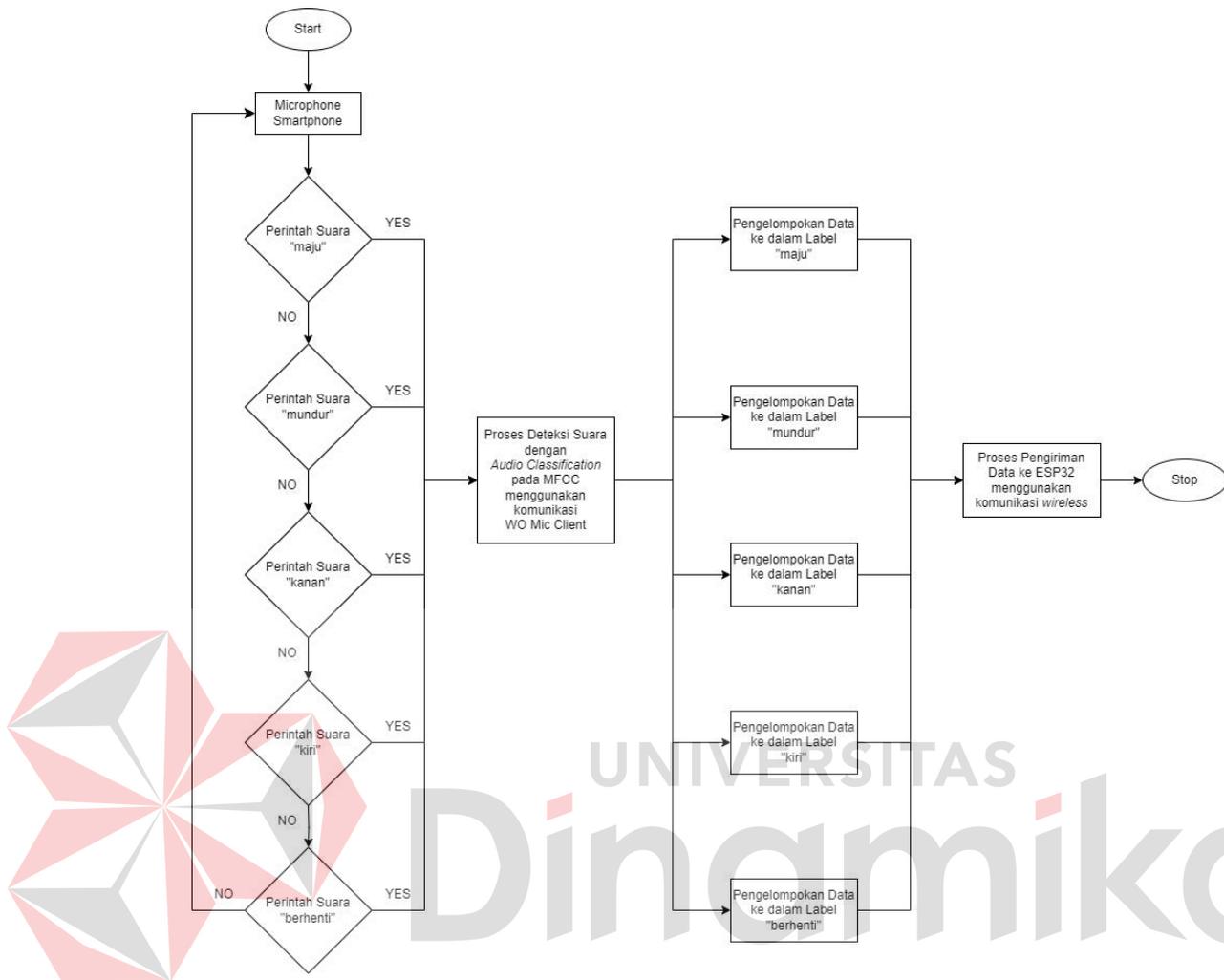
Pada Gambar 3.28 perintah “maju” MFCC menunjukkan pola yang relatif datar dengan beberapa puncak kecil di frekuensi rendah. Hal ini menunjukkan bahwa perintah "maju" memiliki energi yang terdistribusi secara merata di frekuensi rendah dan tidak memiliki frekuensi dominan yang jelas. Perintah kanan MFCC menunjukkan pola yang sedikit lebih bergelombang dibandingkan dengan "maju", dengan beberapa puncak yang lebih menonjol di frekuensi rendah dan menengah. Hal ini menunjukkan bahwa perintah "kanan" memiliki energi yang lebih terkonsentrasi di frekuensi rendah dan menengah dibandingkan dengan "maju". Perintah kiri MFCC menunjukkan pola yang lebih kompleks dibandingkan dengan "maju" dan "kanan", dengan beberapa puncak yang lebih menonjol di frekuensi rendah dan menengah. Hal ini menunjukkan bahwa perintah "kiri" memiliki energi yang lebih terdistribusi di frekuensi rendah dan menengah, dengan beberapa frekuensi yang lebih dominan dibandingkan dengan "maju" dan "kanan". Perintah mundur MFCC menunjukkan pola yang mirip dengan "kiri", dengan beberapa puncak yang menonjol di frekuensi rendah dan menengah. Hal ini menunjukkan bahwa perintah "mundur" memiliki energi yang terdistribusi di

frekuensi rendah dan menengah, dengan beberapa frekuensi yang lebih dominan. Perintah berhenti MFCC menunjukkan pola yang sangat berbeda dibandingkan dengan perintah lainnya, dengan puncak yang sangat tinggi di frekuensi rendah. Hal ini menunjukkan bahwa perintah "berhenti" memiliki energi yang terkonsentrasi pada frekuensi rendah dan memiliki frekuensi dominan yang jelas.

3.5 Flowchart Python

Pada Gambar 3.29 di bawah menjelaskan bahwa *microphone smartphone* menggunakan aplikasi *WO Mic Client* merupakan inputan yang selanjutnya terdapat lima pengecekan kondisi dan jika mendeteksi suara dengan perintah suara "maju", "mundur", "kanan", "kiri", "berhenti" maka akan lanjut ke proses deteksi suara dengan *Audio Classification* pada MFCC. Saat tidak sesuai dengan perintah suara yang ada maka akan diulang ke *microphone smartphone*. Setelah melakukan proses tersebut, akan dilakukan pengelompokan data ke label yang sudah ada seperti "maju", "mundur", "kanan", "kiri", "berhenti". Kemudian akan menampilkan hasil kalimat dari deteksi suara perintah pengguna beserta label kategorinya selanjutnya data tersebut dikirim ke ESP32 menggunakan komunikasi *wireless*.



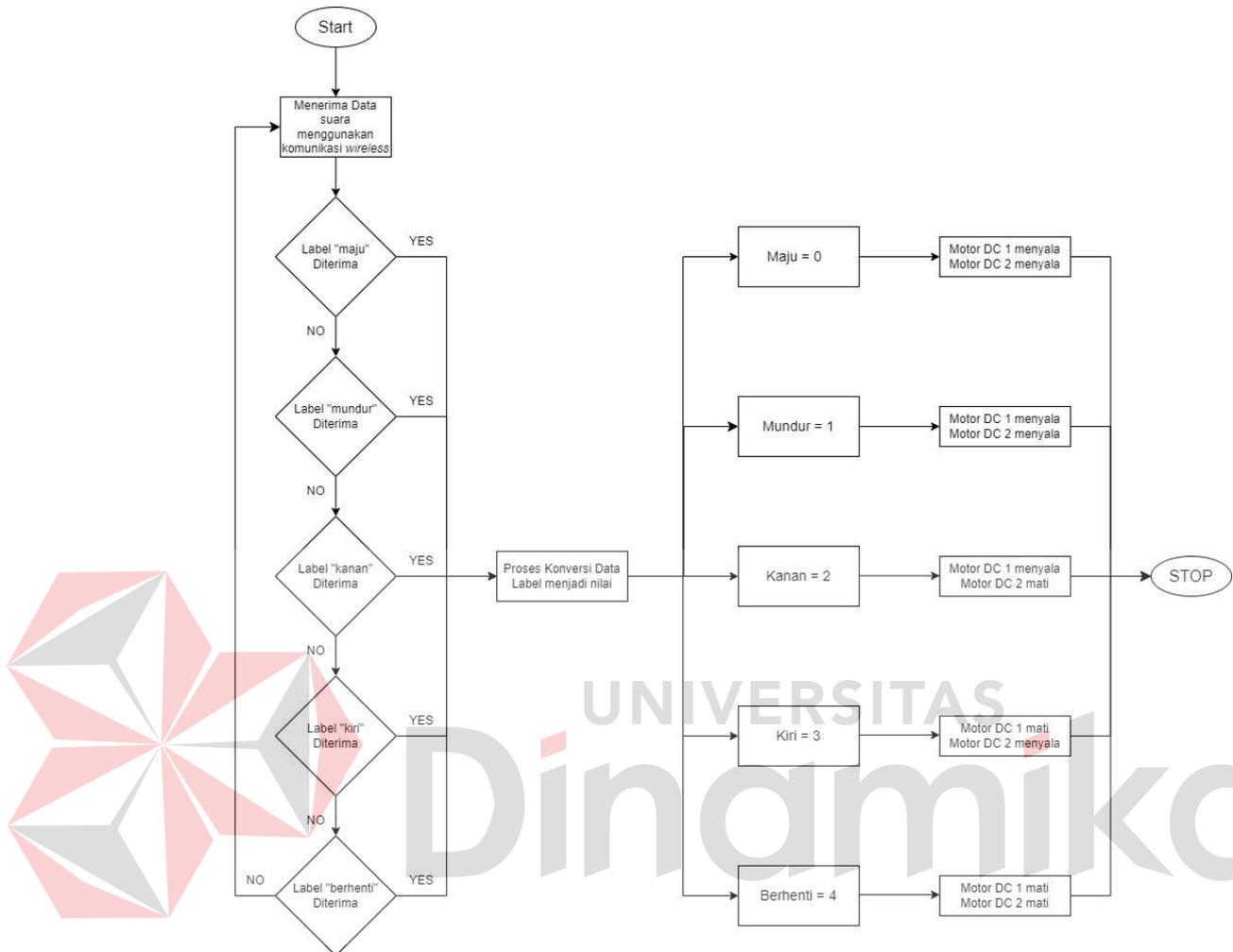


Gambar 3.29 Flowchart Python

3.6 Flowchart ESP32

Berdasarkan Gambar 3.30 di bawah setelah ESP32 menerima data suara maka selanjutnya melakukan lima pengecekan kondisi yang sesuai dengan perintah pengguna. Pengecekan kondisi label “maju”, “mundur”, “kanan”, “kiri”, “berhenti” dilakukan sesuai urutan jika sesuai maka akan berlanjut melakukan proses konversi data label menjadi nilai. Setelah melakukan proses konversi data label menjadi nilai seperti “maju” sama dengan nilai 0 yang memiliki perintah menyalakan motor dc 1 dan 2. Nilai label “mundur” adalah 1 yang menyalakan motor dc 1 dan 2. Nilai label “kanan” sama dengan 2 yang akan menyalakan motor dc 1 dan mematikan motor dc 2. Nilai label “kiri” sama dengan 3 yang membuat motor dc 1 mati dan motor dc

2 menyala. Nilai label “berhenti” sama dengan 4 yang membuat motor dc 1 mati dan motor dc 2 mati.



Gambar 3.30 Flowchart ESP32

3.7 Transmisi Pengiriman Data

Proses transmisi data dari Python ke ESP32 dimulai pada sisi Python. Program Python menyiapkan data yang akan dikirim, mengemasnya menjadi paket data yang sesuai untuk pengiriman melalui protokol UDP, dan kemudian menggunakan pustaka socket untuk membuat koneksi dan mengirimkan paket data melalui jaringan.

Setelah data dikemas dan siap, paket data tersebut dikirim melalui jaringan menggunakan protokol UDP. UDP dipilih karena kesederhanaannya dan kemampuannya untuk mentransmisikan data dengan cepat tanpa memerlukan overhead yang besar seperti pada protokol TCP.

```
# Fungsi untuk mengirim data ke ESP32 menggunakan UDP
def send_to_esp32(ip, port, message):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP
    sock.sendto(message.encode(), (ip, port))
    print(f"Sent to ESP32: {message}")

# Konfigurasi alamat IP dan port ESP32
esp32_ip = "192.168.144.235" # Ganti dengan alamat IP ESP32 Anda
esp32_port = 9999
```

Gambar 3.31 Program Python Mengirim Data

Pada Gambar 3.31 fungsi Python `send_to_esp32` dirancang untuk mengirim data berupa kelas prediksi dan probabilitas ke perangkat ESP32 melalui protokol UDP. Fungsi ini menerima empat parameter: `predicted_class`, `probability`, `esp32_ip`, dan `esp32_port`. Di dalam fungsi, socket UDP dibuat dengan menggunakan `socket.socket(socket.AF_INET, socket.SOCK_DGRAM)` dan dibungkus dalam konteks manajer `with`, yang memastikan bahwa socket ditutup secara otomatis setelah operasi selesai.

Pesan yang akan dikirim disusun dengan format string, menggabungkan `predicted_class` dengan `probability` yang diformat dengan empat tempat desimal. String format ini kemudian diencode menjadi bytes menggunakan `message.encode()`. Fungsi `sock.sendto (message.encode(), (esp32_ip, esp32_port))` mengirimkan pesan yang telah diencode ke alamat IP dan port yang ditentukan. Setelah pengiriman, fungsi mencetak pesan konfirmasi yang menunjukkan bahwa kelas prediksi dan probabilitas telah berhasil dikirim ke ESP32, memberikan umpan balik tentang data yang dikirim dan alamat tujuan. Dengan demikian, fungsi ini memfasilitasi transmisi data prediksi dan probabilitas ke ESP32 untuk aplikasi seperti pengendalian perangkat berbasis hasil analisis dari Python.

Pada sisi ESP32, server UDP yang berjalan di perangkat ESP32 menerima paket data yang dikirim dari Python. ESP32 kemudian memproses data yang

diterima sesuai dengan program yang telah diimplementasikan di perangkat tersebut. Pemrosesan ini bisa melibatkan berbagai tindakan, seperti pengendalian perangkat keras, pengambilan keputusan, atau tindakan lainnya yang diperlukan. Setelah data diproses, ESP32 melakukan tindakan atau kontrol yang sesuai berdasarkan data yang diterima.

```
int packetSize = udp.parsePacket();
if (packetSize) {
  // Terima paket UDP
  int len = udp.read(incomingPacket, 255);
  if (len > 0) {
    incomingPacket[len] = 0;
  }
  Serial.printf("Received packet: %s\n", incomingPacket);

  // Parsing data yang diterima
  String packetData = String(incomingPacket);
  int delimiterIndex = packetData.indexOf(',');
  String command = packetData.substring(0, delimiterIndex);
  float probability = packetData.substring(delimiterIndex + 1).toFloat();
}
```

Gambar 3.32 Program ESP32 Menerima Data

Pada Gambar 3.32 dalam program ESP32 ini, baris kode pertama `int packetSize = udp.parsePacket();` digunakan untuk memeriksa apakah ada paket UDP yang diterima oleh socket UDP. Jika `packetSize` lebih besar dari 0, yang menandakan adanya paket yang diterima, kode selanjutnya akan mengeksekusi bagian yang menangani paket tersebut. Fungsi `udp.read(incomingPacket, 255)` membaca data dari paket UDP dan menyimpannya dalam buffer `incomingPacket` dengan batas maksimum 255 byte. Jika jumlah data yang dibaca (`len`) lebih besar dari 0, kode ini menambahkan karakter null (`\0`) pada akhir buffer untuk mengakhiri string, memastikan bahwa data yang diterima dapat diproses sebagai string dengan benar. Kemudian, `Serial.printf("Received packet: %s\n", incomingPacket);` mencetak konten paket yang diterima ke serial monitor untuk pemantauan.

Selanjutnya, data yang diterima diubah menjadi objek `String` dengan `String(packetData)`. Program mencari posisi delimiter (tanda koma) dalam string dengan `packetData.indexOf(',')` untuk memisahkan bagian perintah dari nilai probabilitas. Bagian sebelum delimiter diambil sebagai perintah dengan `packetData.substring(0, delimiterIndex)`, sementara nilai probabilitas diambil dari

bagian setelah delimiter, yang kemudian dikonversi menjadi tipe data `float` menggunakan `packetData.substring(delimiterIndex + 1).toFloat()`. Proses ini memungkinkan program untuk mengekstrak dan memproses informasi yang diterima dalam paket UDP secara efisien.

3.8 Klasifikasi Audio

Audio klasifikasi adalah proses menggunakan algoritma pembelajaran mesin untuk mengidentifikasi dan mengategorikan jenis suara atau musik dalam sebuah file audio. Proses ini melibatkan beberapa tahapan, mulai dari pengumpulan dan pra-pemrosesan data audio, ekstraksi fitur, hingga pelatihan model klasifikasi. Pada tahap pra-pemrosesan, data audio sering kali difragmentasi menjadi bingkai-bingkai pendek dan dihapus dari noise menggunakan teknik seperti Hamming Window. Fitur yang umum diekstraksi meliputi Mel-Frequency Cepstral Coefficients (MFCC), spektrum daya, dan log energi, yang merepresentasikan karakteristik penting dari sinyal audio. Model pembelajaran mesin seperti Convolutional Neural Networks (CNN) atau Recurrent Neural Networks (RNN) kemudian dilatih menggunakan fitur-fitur ini untuk mengenali pola dan mengklasifikasikan suara ke dalam kategori yang telah ditentukan, seperti suara hewan, jenis musik, atau ucapan manusia. Klasifikasi audio memiliki berbagai aplikasi praktis, termasuk pengenalan suara, deteksi emosi dalam ucapan, dan identifikasi genre musik.

```
from keras.models import Sequential
from keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout, Activation
from keras.optimizers import Adam
```

Gambar 3.33 Import *library*

Pada Gambar 3.33 program ini menggunakan Keras untuk membangun dan melatih model CNN. Library yang diimpor mencakup Sequential untuk membuat model berurutan, berbagai layer seperti Conv1D, MaxPooling1D, Flatten, Dense, Dropout, Activation, dan optimizer Adam.

```

# Misalkan input_height adalah jumlah koefisien MFCC yang diekstrak
input_height = 40 # jumlah koefisien MFCC
num_labels = 5 # misalkan ada 10 kelas output

# Define model
model = Sequential()

```

Gambar 3.34 Import variabel

Pada Gambar 3.34 menggunakan variable `input_height` dan `num_labels`. `input_height` adalah jumlah koefisien MFCC yang diekstrak dari setiap frame audio, dan `num_labels` adalah jumlah kelas yang akan diprediksi oleh model. Membuat model Keras berurutan (`Sequential`), yang berarti layer-layer akan ditambahkan satu per satu.

```

# First convolutional layer
model.add(Conv1D(32, kernel_size=3, input_shape=(input_height, 1)))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))

# Second convolutional layer
model.add(Conv1D(64, kernel_size=3))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))

# Third convolutional layer
model.add(Conv1D(128, kernel_size=3))
model.add(Activation('relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.3))

```

Gambar 3.35 Layer CNN 1D

Pada Gambar 3.35 layer konvolusi dengan 32 filter, ukuran kernel 3, dan input shape (`input_height, 1`). Aktivasi menggunakan ReLU. Kemudian, layer max pooling dengan ukuran pool 2 untuk mengurangi dimensi, dan dropout dengan probabilitas 0.3 untuk mengurangi overfitting. Layer konvolusi dengan 64 filter, ukuran kernel 3. Aktivasi menggunakan ReLU, diikuti dengan max pooling dan dropout yang sama seperti layer pertama. Layer konvolusi dengan 128 filter, ukuran kernel 3. Aktivasi menggunakan ReLU, diikuti dengan max pooling dan dropout.

```
# Flatten the output of the last convolutional layer
model.add(Flatten())

# Fully connected layer
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.3))

# Output layer
model.add(Dense(num_labels))
model.add(Activation('softmax'))

# Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(),
              metrics=['accuracy'])

# Print the model summary
model.summary()
```

Gambar 3.36 Kompilasi Model

Pada Gambar 3.36 layer ini mengubah output dari layer konvolusi terakhir menjadi vektor 1D agar bisa diproses oleh layer dense (fully connected). Layer dense dengan 256 unit, diikuti dengan aktivasi ReLU dan dropout. Layer output dengan jumlah unit sesuai dengan jumlah kelas (5) dan aktivasi softmax untuk klasifikasi multi-kelas. Model dikompilasi menggunakan loss function `categorical_crossentropy`, optimizer Adam, dan metric accuracy untuk evaluasi kinerja model. Menampilkan ringkasan arsitektur model, termasuk jumlah parameter yang dapat dipelajari di setiap layer.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengujian Klasifikasi Perintah Suara

4.1.1 Pengujian Klasifikasi Perintah Suara

Dalam pengujian ini, deteksi perintah suara dengan menggunakan metode *Deep learning*, diperlukan dataset untuk *training* serta *library MFCC* untuk mengenali perintah suara dengan cara mengekstrak fitur suara. Dimana perintah suara sebagai penentu arah gerak *mobile robot* “maju”, “mundur”, “kanan”, “kiri”, “berhenti”. Pengujian ini dilakukan oleh satu orang pemberi perintah suara dan uji coba sebanyak 30 kali serta diambil persentase dari setiap perintah suara.

4.1.2 Alat yang Digunakan Pengujian Klasifikasi Perintah Suara

Berikut ini alat yang digunakan untuk melakukan pengambilan data pada bagian pengujian klasifikasi perintah suara:

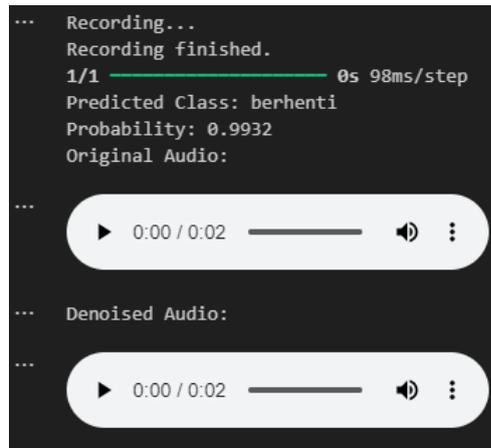
1. Laptop
2. *Smartphone*

4.1.3 Cara Pengujian Klasifikasi Perintah Suara

Berikut merupakan prosedur pengujian klasifikasi perintah suara:

1. Menyalakan laptop untuk mengakses *visual studio code*.
2. Menjalankan program *TA_Rizal_Robot.ipynb* yang berada di lampiran.
3. Menyebutkan perintah suara berupa maju, mundur, kanan, kiri, dan berhenti sebanyak 30 kali untuk setiap perintah suara, maka program akan menampilkan prediksi dari perintah suara.

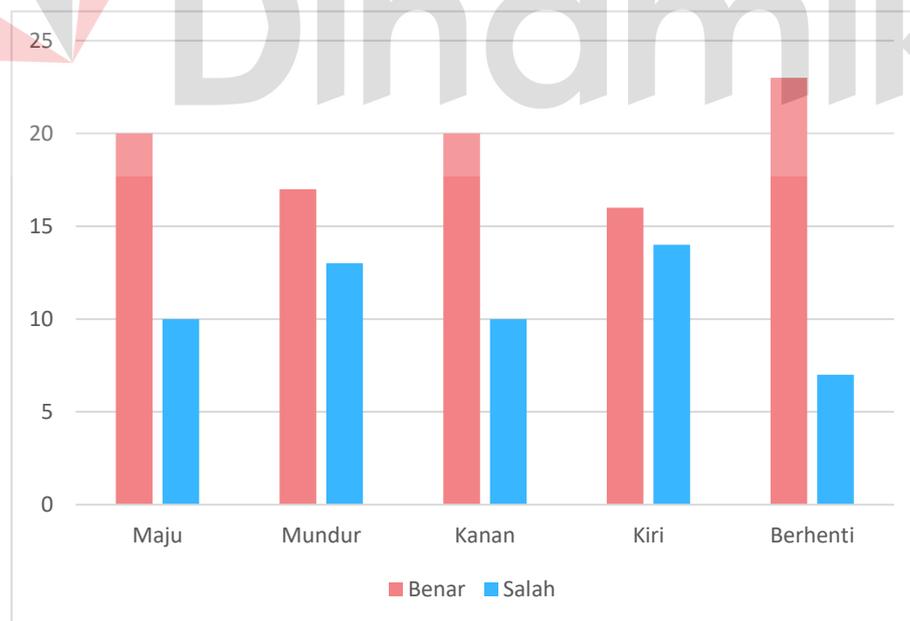
4.1.4 Hasil Pengujian Klasifikasi Perintah Suara



Gambar 4.1 Pengujian Klasifikasi Perintah Suara

Tabel 4.1 Hasil Pengujian Klasifikasi Suara

No.	Perintah Suara	Akurasi
1	Maju	66.67%
2	Mundur	56.67%
3	Kanan	66.67%
4	Kiri	53.33%
5	Berhenti	76.67%



Gambar 4.2 Grafik Pengujian Klasifikasi Perintah Suara

4.1.5 Analisis Pengujian Klasifikasi Perintah Suara

Pada Tabel 4.1 dilakukan total 150 kali pengujian dengan 30 kali percobaan pada perintah suara maju, 30 kali percobaan pada perintah suara mundur, 30 kali percobaan pada perintah suara kanan, 30 kali percobaan pada perintah suara kiri, 30 kali percobaan pada perintah suara berhenti. Dalam 30 kali percobaan perintah suara maju terdapat 20 benar prediksi maju dengan 7 kali salah prediksi kata berhenti dan 3 kali salah prediksi kata mundur yang menghasilkan nilai akurasi sebesar 66.67%. Dalam 30 kali percobaan perintah suara mundur terdapat 17 benar prediksi mundur dengan 4 kali salah prediksi kata berhenti dan 9 kali salah prediksi kata maju yang menghasilkan nilai akurasi sebesar 56.67%. Dalam 30 kali percobaan perintah suara kanan terdapat 20 benar prediksi kanan dengan 5 kali salah prediksi kata berhenti, 3 kali salah prediksi kata maju dan 2 kali salah prediksi kata kiri yang menghasilkan nilai akurasi sebesar 66.67%. Dalam 30 kali percobaan perintah suara kiri terdapat 16 benar prediksi kiri dengan 13 kali salah prediksi kata berhenti dan 1 kali salah prediksi kata mundur yang menghasilkan nilai akurasi sebesar 53.33%. Dalam 30 kali percobaan perintah suara berhenti terdapat 23 benar prediksi berhenti dengan 5 kali salah prediksi kata mundur dan 2 kali salah prediksi kata maju yang menghasilkan nilai akurasi sebesar 76.67%.

4.2 Pengujian MFCC

4.2.1 Pengujian MFCC

Dalam pengujian ini, perintah suara yang diberikan akan diekstraksi fitur dengan *library* MFCC yang nantinya akan menghasilkan nilai koefisien cepstral dalam bentuk array dan akan dicocokkan dengan array dari setiap label perintah suara yang ada. Dimana perintah suara sebagai penentu arah gerak *mobile robot* “maju”, “mundur”, “kanan”, “kiri”, “berhenti”. Pengujian ini dilakukan oleh satu orang pemberi perintah suara dan uji coba sebanyak 30 kali serta diambil persentase dari setiap perintah suara.

4.2.2 Alat yang Digunakan Pengujian MFCC

Berikut ini alat yang digunakan untuk melakukan pengambilan data pada bagian pengujian klasifikasi perintah suara:

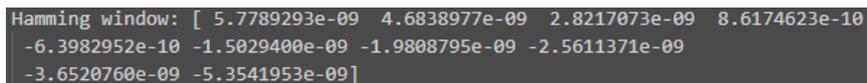
1. Laptop
2. *Smartphone*

4.2.3 Cara Pengujian MFCC

Berikut merupakan prosedur pengujian klasifikasi perintah suara:

1. Menyalakan laptop untuk mengakses *visual studio code*.
2. Menjalankan program TA_Rizal_Robot.ipynb yang berada di lampiran.
3. Menyebutkan perintah suara berupa maju, mundur, kanan, kiri, dan berhenti sebanyak 30 kali untuk setiap perintah suara, maka program akan menampilkan prediksi dari perintah suara serta nilai array MFCC.

4.2.4 Hasil Pengujian MFCC



```
Hamming window: [ 5.7789293e-09  4.6838977e-09  2.8217073e-09  8.6174623e-10
-6.3982952e-10 -1.5029400e-09 -1.9808795e-09 -2.5611371e-09
-3.6520760e-09 -5.3541953e-09 ]
```

Gambar 4.3 Hasil *Hamming Window*



```
FFT: [ 0. 1.52236951 3.04473902 4.56710853 6.08947804 7.61184756
9.13421707 10.65658658 12.17895609 13.7013256 ]
```

Gambar 4.4 Hasil *FFT*

```

Filter Bank: [[-26.61680735 -21.43369349 -17.56931533 -16.0475135 -15.26874636]
[-25.23750562 -21.44739102 -14.67329254 -13.85438949 -11.14288033]
[-25.66025493 -21.86457981 -15.2497765 -14.047214 -12.26378009]
[-26.17580649 -21.49878619 -19.12885831 -15.71112845 -18.06414172]
[-25.82876991 -22.3427948 -20.53922837 -16.49776104 -18.69802562]
[-26.50406207 -22.83007355 -21.58670846 -17.06866864 -19.10608142]
[-26.35666802 -23.3391593 -21.94999749 -17.40083212 -18.83519938]
[-26.38518307 -23.78102234 -22.12659568 -17.65410138 -19.1655241 ]
[-26.0076556 -23.68629221 -22.02744811 -17.85102417 -18.99225114]
[-25.24110682 -23.0281644 -21.57483482 -18.12230047 -18.85402231]
[-24.94116744 -22.9293131 -21.34178909 -18.15546136 -18.78311312]
[-25.00765226 -23.34225605 -21.21558831 -18.15085675 -18.32380002]
[-23.9923051 -21.49607235 -20.29785805 -18.03656224 -17.48703283]
[-23.99949685 -21.64420299 -18.85745608 -17.32172155 -15.81774999]
[-25.1424842 -23.36689285 -19.80220032 -17.51618658 -16.92834128]
[-25.85972709 -24.953683 -24.65280419 -18.05198725 -18.47035473]
[-25.79927489 -25.15513625 -24.5873461 -18.08036781 -18.38175174]
[-25.67982733 -25.18352649 -24.5685806 -18.02048379 -18.24589857]
[-25.54983178 -25.15786616 -24.57121853 -17.93064853 -18.10542782]
[-25.43311356 -25.12123178 -24.58525395 -17.8477603 -17.98211229]
[-25.31281332 -25.06106873 -24.57684328 -17.75125712 -17.85744063]
[-25.19197724 -24.98741975 -24.56435467 -17.64774132 -17.73342684]
[-25.06080358 -24.89063569 -24.513183 -17.53013556 -17.60007045]
[-24.93458931 -24.79049838 -24.45537181 -17.41365512 -17.47238597]
[-24.80463811 -24.67874351 -24.37669137 -17.29059362 -17.34144086]
[-24.6739736 -24.55944251 -24.27815191 -17.16392689 -17.21016206]]

```

Gambar 4.6 Hasil *Filter Bank*

```

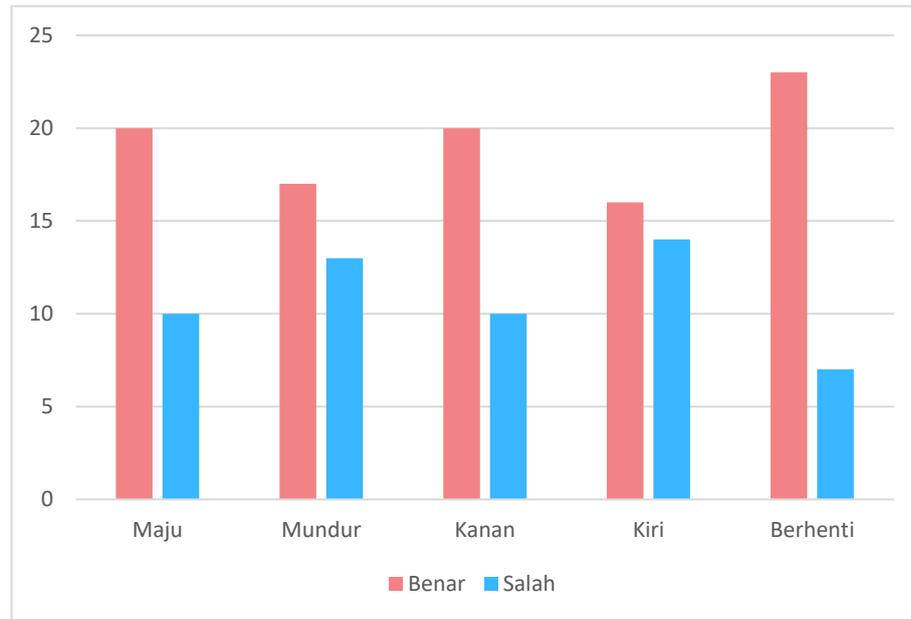
MFCC: [[-21.90865098 -19.38294303 -14.16240431 -12.87751834 -10.82159018]
[-4.53107971 14.64100463 30.10030848 8.0098534 6.04357992]
[-3.17912991 0.99978749 7.77179687 15.02341938 16.85426721]
[-3.27222119 -0.14782415 14.81544842 9.49665133 23.00694394]
[13.45370624 20.59948061 43.74874159 9.92538998 34.39368409]
[5.31389192 4.10326349 15.08189843 -0.95693123 14.82360955]
[-9.46020607 -16.50526669 -18.12709398 -4.35207517 -0.46414085]
[-7.28755556 -5.39512008 0.09030424 -2.11846924 6.99457594]
[2.76822226 9.52529148 9.75872261 -3.37889673 6.49957107]
[-4.03835035 -4.01703886 -25.36673511 -12.3304488 -19.58195968]
[-7.5887717 -8.37277306 -28.39524079 -11.68479777 -27.40063894]
[0.23592863 7.02860114 -0.36569615 -6.80517288 -15.36056291]
[2.26561223 9.62707747 -7.96581879 -7.67233697 -17.57804376]]

```

Gambar 4.5 Hasil *MFCC*

Tabel 4.2 Hasil Pengujian MFCC

No.	Perintah Suara	Akurasi
1	Maju	66.67%
2	Mundur	56.67%
3	Kanan	66.67%
4	Kiri	53.33%
5	Berhenti	76.67%



Gambar 4.7 Grafik Pengujian MFCC

4.2.5 Analisis Pengujian MFCC

Pada Gambar 4.3 merupakan nilai *Hamming Window* berupa array. Nilai dalam sinyal berikisar dari 5.7789293×10^{-9} hingga $-5.3541953 \times 10^{-9}$. Nilai positif dan negatif menunjukkan bahwa sinyal ini berfluktuasi, atau berosilasi, di sekitar titik nol. Nilai positif (misal 5.7789293×10^{-9} , 4.6838977×10^{-9}) menunjukkan bahwa sinyal berada di atas garis dasar atau baseline. Nilai negatif (misal $-6.3982952 \times 10^{-10}$, $-1.5029400 \times 10^{-9}$) menunjukkan bahwa sinyal berada di bawah garis dasar atau baseline.

Pada Gambar 4.4 merupakan nilai FFT berupa array. Nilai-nilai ini menunjukkan amplitudo komponen-komponen frekuensi yang terdapat dalam sinyal. Setiap nilai dalam deret ini merepresentasikan amplitudo dari frekuensi tertentu. Nilai pertama (0.0) biasanya adalah komponen DC (*Direct Current*), yang merepresentasikan nilai rata-rata atau offset dari sinyal asli. Nilai komponen DC dari *Fast Fourier Transform* (FFT) adalah nilai pertama dari hasil transformasi FFT. Komponen DC mewakili bagian dari sinyal yang tidak berubah atau frekuensi nol. Nilai selanjutnya (1.52236951, 3.04473902, 4.56710853, dst.) adalah magnitudo dari komponen frekuensi yang lebih tinggi. Nilai ini menunjukkan kekuatan atau kontribusi dari masing-masing frekuensi dalam membentuk sinyal asli.

Pada Gambar 4.5 merupakan nilai *filter bank* yang berupa array. Setiap baris merepresentasikan output dari sinyal yang melewati satu set filter. Setiap kolom merepresentasikan output dari sinyal pada filter tertentu dalam set tersebut. Nilai-nilai yang diberikan adalah hasil dari pemfilteran signal asli, dan biasanya mewakili kekuatan atau amplitudo signal pada frekuensi tertentu yang difilter. Nilai negatif menunjukkan bahwa output dari filter memiliki fase atau polaritas tertentu yang berbeda dari nilai positif. Nilai pertama dalam baris pertama (-26.61680735) menunjukkan output signal setelah melewati filter pertama. Nilai kedua dalam baris pertama (-21.43369349) menunjukkan output signal setelah melewati filter kedua, dan seterusnya. Setiap baris berikutnya menunjukkan hasil untuk signal yang sama melewati set filter yang sama, atau bisa juga menunjukkan hasil untuk signal yang berbeda melewati set filter yang sama

Pada Gambar 4.6. merupakan nilai MFCC yang berupa array. Setiap baris dalam matriks ini mewakili frame waktu tertentu dari sinyal audio. Setiap kolom mewakili koefisien MFCC pada frame waktu tersebut. Koefisien pertama pada setiap frame biasanya disebut sebagai koefisien cepstral 0 (C0) dan sering diabaikan karena mewakili rata-rata log-energi dari frame. Koefisien selanjutnya (C1, C2, C3, dst.) mewakili informasi spektral yang lebih rinci dan penting untuk pengenalan pola dalam audio. Nilai-nilai dalam matriks ini bisa positif atau negatif. Nilai positif menunjukkan energi yang lebih besar pada frekuensi tertentu, sementara nilai negatif menunjukkan energi yang lebih rendah. Nilai-nilai ini mewakili amplitudo dari berbagai komponen frekuensi dalam skala mel-frequency. Baris pertama $[-21.90865098, -19.38294303, -14.16240431, -12.87751834, -10.82159018]$ adalah koefisien MFCC dari frame pertama sinyal audio. Nilai-nilai ini menunjukkan karakteristik spektral dari frame tersebut. Baris kedua $[-4.53107971, 14.64100463, 30.10030848, 8.0098534, 6.04357992]$ adalah koefisien MFCC dari frame kedua sinyal audio. Perubahan nilai-nilai ini dari frame pertama menunjukkan perbedaan dalam karakteristik spektral dari satu frame ke frame berikutnya.

Pada Tabel 4.2 dilakukan total 150 kali pengujian dengan 30 kali percobaan pada perintah suara maju, 30 kali percobaan pada perintah suara mundur, 30 kali percobaan pada perintah suara kanan, 30 kali percobaan pada perintah suara kiri,

30 kali percobaan pada perintah suara berhenti. Dalam 30 kali percobaan perintah suara maju terdapat 20 benar prediksi maju dengan 7 kali salah prediksi kata berhenti dan 3 kali salah prediksi kata mundur yang menghasilkan nilai akurasi sebesar 66.67%. Dalam 30 kali percobaan perintah suara mundur terdapat 17 benar prediksi mundur dengan 4 kali salah prediksi kata berhenti dan 9 kali salah prediksi kata maju yang menghasilkan nilai akurasi sebesar 56.67%. Dalam 30 kali percobaan perintah suara kiri terdapat 20 benar prediksi kiri dengan 5 kali salah prediksi kata berhenti, 3 kali salah prediksi kata maju dan 2 kali salah prediksi kata kiri yang menghasilkan nilai akurasi sebesar 66.67%. Dalam 30 kali percobaan perintah suara kiri terdapat 16 benar prediksi kiri dengan 13 kali salah prediksi kata berhenti dan 1 kali salah prediksi kata mundur yang menghasilkan nilai akurasi sebesar 53.33%. Dalam 30 kali percobaan perintah suara berhenti terdapat 23 benar prediksi berhenti dengan 5 kali salah prediksi kata mundur dan 2 kali salah prediksi kata maju yang menghasilkan nilai akurasi sebesar 76.67%. Hasil tersebut ditampilkan pada Gambar 4.7 grafik batang.

4.3 Pengujian Kinerja Motor DC

4.3.1 Pengujian Kinerja Motor DC

Dalam pengujian ini dilakukan oleh satu orang pemberi perintah suara dan uji coba sebanyak sepuluh kali serta diambil persentase dari setiap perintah suara. Dimana perintah suara sebagai penentu arah gerak *mobile* robot seperti “maju”, “mundur”, “kanan”, “kiri”, “berhenti”. Dan juga akan mengetahui status motor dc 1 dan 2 yang sudah sesuai dengan perintah suara. Nanti saat penelitian dimulai pengujian akan dilakukan sebanyak 30 percobaan disetiap perintah suara.

4.3.2 Alat yang Digunakan Pengujian Kinerja Motor DC

Berikut ini alat yang digunakan untuk melakukan pengambilan data pada bagian pengujian klasifikasi perintah suara:

1. Laptop
2. *Smartphone*
3. *Mobile* robot
4. Kabel USB

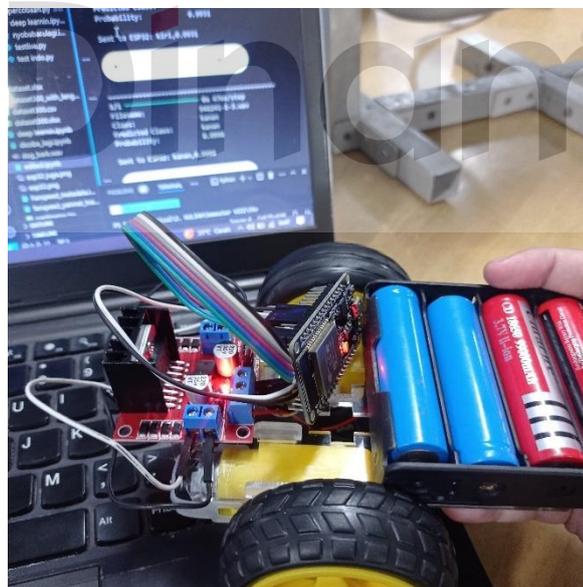
5. EPS32

4.3.3 Cara Pengujian Kinerja Motor DC

Berikut merupakan prosedur pengujian klasifikasi perintah suara:

1. Menyalakan laptop untuk mengakses aplikasi *visual studio code* dan aplikasi Arduino IDE.
2. Menjalankan program TA_Rizal_Robot.ipynb yang berada di lampiran.
3. Menghubungkan laptop dengan ESP32 yang terdapat pada *mobile robot*.
4. Mengunggah program robot TA_Rizal_Robot.ino yang berada di lampiran.
5. Menyebutkan perintah suara berupa maju, mundur, kanan, kiri, dan berhenti sebanyak 30 kali untuk setiap perintah suara, maka program akan mengirim perintah suara ke ESP32 dengan komunikasi WiFi UDP.
6. Memperhatikan motor DC bekerja sesuai dengan perintah suara yang telah diterima dan dijalankan.

4.3.4 Hasil Pengujian Kinerja Motor DC

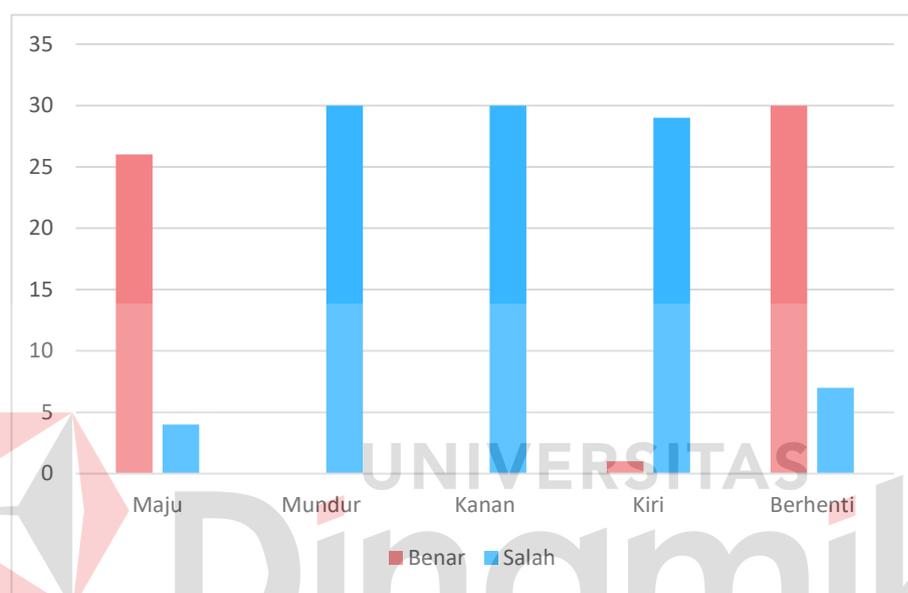


Gambar 4.8 Pengujian Kinerja Motor DC

Tabel 4.3 Hasil Pengujian Kinerja Motor DC

No.	Perintah Suara	Akurasi
1	Maju	86.67%
2	Mundur	0%
3	Kanan	0%
4	Kiri	3.33%
5	Berhenti	100%

4.3.5 Analisis Pengujian Kinerja Motor DC



Gambar 4.9 Grafik Pengujian Kinerja Motor DC

Pada Tabel 4.3 dilakukan total 150 kali pengujian dengan 30 kali percobaan pada perintah suara maju, 30 kali percobaan pada perintah suara mundur, 30 kali percobaan pada perintah suara kanan, 30 kali percobaan pada perintah suara kiri, 30 kali percobaan pada perintah suara berhenti. Dalam 30 kali percobaan perintah suara maju terdapat 26 benar motor dc bergerak sesuai perintah dengan 3 kali salah motor dc posisi berhenti dan 1 kali salah motor dc bergerak kiri yang menghasilkan nilai akurasi sebesar 86.67%. Dalam 30 kali percobaan perintah suara mundur terdapat 0 benar motor dc bergerak sesuai perintah dengan 26 kali salah motor dc bergerak maju dan 4 kali salah motor dc posisi berhenti yang menghasilkan nilai akurasi sebesar 0%. Dalam 30 kali percobaan perintah suara kiri terdapat 0 benar motor dc bergerak sesuai perintah dengan 9 kali salah motor dc posisi berhenti dan 2 kali salah motor dc bergerak kiri yang menghasilkan nilai akurasi sebesar 0%. Dalam 30 kali percobaan perintah suara kiri terdapat 1 benar motor dc bergerak

sesuai perintah dengan 29 kali salah motor dc posisi berhenti yang menghasilkan nilai akurasi sebesar 3.33%. Dalam 30 kali percobaan perintah suara berhenti terdapat 30 benar motor dc bergerak sesuai perintah yang menghasilkan nilai akurasi sebesar 100%. Hasil tersebut ditampilkan pada Gambar 4.9 grafik batang.

Beberapa faktor yang mungkin menyebabkan kegagalan ini adalah kualitas data pelatihan yang tidak memadai. Selain itu, mungkin juga ada masalah dengan penetapan label atau bias dalam model yang membuatnya kurang sensitif terhadap perintah "Mundur" dan "Kanan".

4.4 Pengujian Akurasi Perintah Suara dengan Motor DC

4.4.1 Pengujian Akurasi Perintah Suara dengan Motor DC

Dalam pengujian ini dilakukan oleh lima orang pemberi perintah suara dan uji coba sebanyak sepuluh kali disetiap perintah suara. Dengan tujuan mengetahui tinggi akurasi kecocokan dari perintah suara dengan pergerakan *mobile robot*. Nanti saat penelitian dimulai pengujian akan dilakukan sebanyak 10 percobaan disetiap perintah suara.

4.4.2 Alat yang Digunakan Pengujian Akurasi Perintah Suara dengan Motor DC

Berikut ini alat yang digunakan untuk melakukan pengambilan data pada bagian pengujian klasifikasi perintah suara:

1. Laptop
2. *Smartphone*
3. *Mobile robot*
4. Kabel USB
5. EPS32

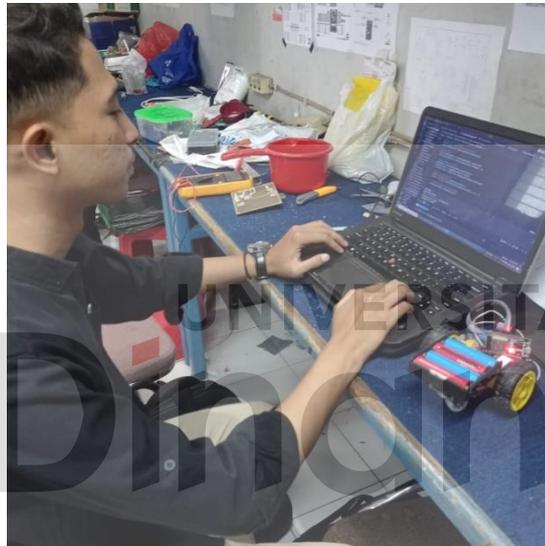
4.4.3 Cara Pengujian Akurasi Perintah Suara dengan Motor DC

Berikut merupakan prosedur pengujian klasifikasi perintah suara:

1. Menyalakan laptop untuk mengakses aplikasi *visual studio code* dan aplikasi Arduino IDE.
2. Menjalankan program TA_Rizal_Robot.ipynb yang berada di lampiran.

3. Menghubungkan laptop dengan ESP32 yang terdapat pada *mobile* robot.
4. Mengunggah program robot TA_Rizal_Robot.ino yang berada di lampiran.
5. Menyebutkan perintah suara berupa maju, mundur, kanan, kiri, dan berhenti sebanyak 30 kali untuk setiap perintah suara, maka program akan mengirim perintah suara ke ESP32 dengan komunikasi WiFi UDP.
6. Memperhatikan motor DC bekerja sesuai dengan perintah suara yang telah diterima dan dijalankan.
7. Dilakukan penyebutan perintah suara dengan lima orang yang berbeda.

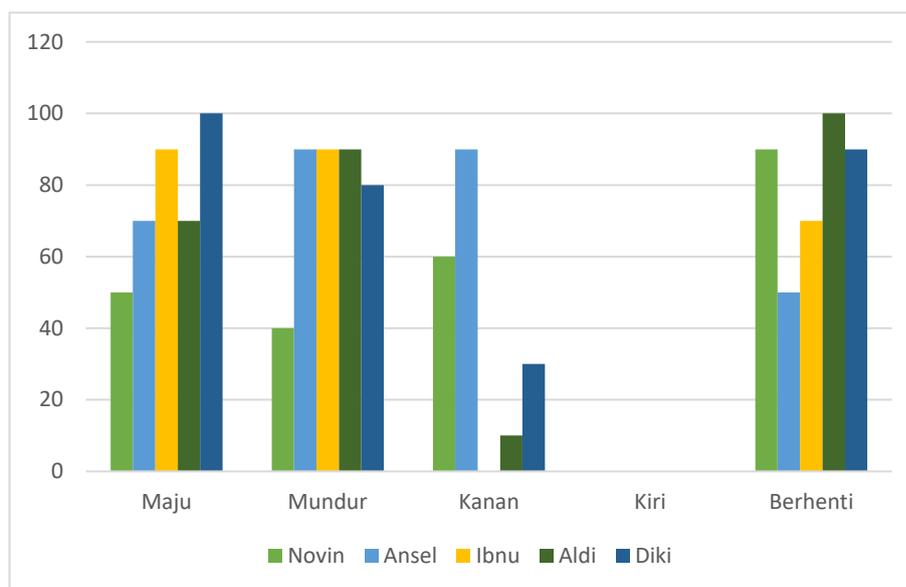
4.4.4 Hasil Pengujian Akurasi Perintah Suara dengan Motor DC



Gambar 4.10 Pengujian Akurasi Perintah Suara dengan Motor DC

Tabel 4.4 Hasil Pengujian Akurasi Perintah Suara dengan Motor DC

No.	Orang ke-	Perintah Suara				
		Maju	Mundur	Kanan	Kiri	Berhenti
1	Novin	50%	40%	60%	0%	90%
2	Ansel	70%	90%	90%	0%	50%
3	Ibnu	90%	90%	0%	0%	70%
4	Aldi	70%	90%	10%	0%	100%
5	Diki	100%	80%	30%	0%	90%



Gambar 4.11 Grafik Pengujian Akurasi Perintah Suara dengan Motor DC

4.4.5 Analisis Pengujian Akurasi Perintah Suara dengan Motor DC

Berdasarkan Lampiran 4 yang merupakan tabel pengujian akurasi orang pertama dengan motor dc. Percobaan pertama yang dilakukan oleh orang ke-1 yaitu Novin melakukan 10 kali percobaan perintah suara maju terdapat 5 benar motor dc bergerak sesuai perintah dengan 5 kali salah motor dc posisi berhenti yang menghasilkan akurasi 50%. Dalam 10 kali percobaan perintah suara mundur terdapat 4 benar motor dc bergerak sesuai perintah dengan 6 kali salah motor dc posisi berhenti yang menghasilkan akurasi 40%. Dalam 10 kali percobaan perintah suara kanan terdapat 6 benar motor dc bergerak sesuai perintah dengan 4 kali salah motor dc bergerak maju yang menghasilkan akurasi 60%. Dalam 10 kali percobaan perintah suara kiri terdapat 0 benar motor dc bergerak sesuai perintah dengan 10 kali salah motor dc posisi berhenti yang menghasilkan akurasi 0%. Dalam 10 kali percobaan perintah suara berhenti terdapat 9 benar motor dc posisi berhenti dengan 1 kali salah motor dc bergerak ke kanan yang menghasilkan akurasi 90%.

Berdasarkan Lampiran 5 yang merupakan tabel pengujian akurasi orang pertama dengan motor dc. Percobaan kedua yang dilakukan oleh orang ke-2 yaitu Ansel melakukan 10 kali percobaan perintah suara maju terdapat 7 benar motor dc bergerak sesuai perintah dengan 3 kali salah motor dc posisi berhenti yang menghasilkan akurasi 70%. Dalam 10 kali percobaan perintah suara mundur

terdapat 9 benar motor dc bergerak sesuai perintah dengan 1 kali salah motor dc bergerak ke kanan yang menghasilkan akurasi 90%. Dalam 10 kali percobaan perintah suara kanan terdapat 9 benar motor dc bergerak sesuai perintah dengan 1 kali salah motor dc bergerak maju yang menghasilkan akurasi 90%. Dalam 10 kali percobaan perintah suara kiri terdapat 0 benar motor dc bergerak sesuai perintah dengan 8 kali salah motor dc posisi berhenti, 1 kali salah motor dc bergerak maju dan 1 kali salah bergerak ke kanan yang menghasilkan akurasi 0%. Dalam 10 kali percobaan perintah suara berhenti terdapat 5 benar motor dc posisi berhenti dengan 5 kali salah motor dc bergerak maju yang menghasilkan akurasi 50%.

Berdasarkan Lampiran 6 yang merupakan tabel pengujian akurasi orang pertama dengan motor dc. Percobaan ketiga yang dilakukan oleh orang ke-3 yaitu Ibnu melakukan dalam 10 kali percobaan perintah suara maju terdapat 9 benar motor dc bergerak sesuai perintah dengan 1 kali salah motor dc posisi berhenti yang menghasilkan akurasi 90%. Dalam 10 kali percobaan perintah suara mundur terdapat 8 benar motor dc bergerak sesuai perintah dengan 1 kali salah motor dc bergerak ke kanan, dan 1 kali motor dc posisi berhenti yang menghasilkan akurasi 90%. Dalam 10 kali percobaan perintah suara kanan terdapat 0 benar motor dc bergerak sesuai perintah dengan 9 kali salah motor dc bergerak maju, dan 1 kali motor dc posisi berhenti yang menghasilkan akurasi 0%. Dalam 10 kali percobaan perintah suara kiri terdapat 0 benar motor dc bergerak sesuai perintah dengan 10 kali salah motor dc posisi berhenti yang menghasilkan akurasi 0%. Dalam 10 kali percobaan perintah suara berhenti terdapat 7 benar motor dc posisi berhenti dengan 3 kali salah motor dc bergerak maju yang menghasilkan akurasi 70%.

Berdasarkan Lampiran 7 yang merupakan tabel pengujian akurasi orang pertama dengan motor dc. Percobaan keempat yang dilakukan oleh orang ke-4 yaitu Aldi melakukan 10 kali percobaan perintah suara maju terdapat 7 benar motor dc bergerak sesuai perintah dengan 3 kali salah motor dc posisi berhenti yang menghasilkan akurasi 70%. Dalam 10 kali percobaan perintah suara mundur terdapat 9 benar motor dc bergerak sesuai perintah dengan 1 kali salah motor dc posisi berhenti yang menghasilkan akurasi 90%. Dalam 10 kali percobaan perintah suara kiri terdapat 1 benar motor dc bergerak sesuai perintah dengan 8 kali salah motor dc bergerak maju, dan 1 kali motor dc posisi berhenti yang menghasilkan

akurasi 10%. Dalam 10 kali percobaan perintah suara kiri terdapat 0 benar motor dc bergerak sesuai perintah dengan 10 kali salah motor dc posisi berhenti yang menghasilkan akurasi 0%. Dalam 10 kali percobaan perintah suara berhenti terdapat 10 benar motor dc posisi berhenti yang menghasilkan akurasi 100%.

Berdasarkan Lampiran 8 yang merupakan tabel pengujian akurasi orang pertama dengan motor dc. Percobaan kelima yang dilakukan oleh orang ke-5 yaitu Diki melakukan 10 kali percobaan perintah suara maju terdapat 10 benar motor dc bergerak sesuai perintah yang menghasilkan akurasi 100%. Dalam 10 kali percobaan perintah suara mundur terdapat 8 benar motor dc bergerak sesuai perintah dengan 2 kali salah motor dc posisi berhenti yang menghasilkan akurasi 80%. Dalam 10 kali percobaan perintah suara kanan terdapat 3 benar motor dc bergerak sesuai perintah dengan 7 kali salah motor dc bergerak maju yang menghasilkan akurasi 30%. Dalam 10 kali percobaan perintah suara kiri terdapat 0 benar motor dc bergerak sesuai perintah dengan 7 kali salah motor dc posisi berhenti, 2 kali salah motor dc berhenti, dan 1 kali motor dc bergerak ke kanan yang menghasilkan akurasi 0%. Dalam 10 kali percobaan perintah suara berhenti terdapat 9 benar motor dc posisi berhenti, dan 1 kali salah motor dc bergerak maju yang menghasilkan akurasi 90%.

Pada Tabel 4.4 yang merupakan hasil pengujian dilakukan total 250 kali pengujian oleh lima orang yang berbeda dengan 10 kali percobaan pada perintah suara maju, 10 kali percobaan pada perintah suara mundur, 10 kali percobaan pada perintah suara kanan, 10 kali percobaan pada perintah suara kiri, 10 kali percobaan pada perintah suara berhenti. Hasil analisis pada pengujian akurasi perintah suara dengan motor DC yang telah dilakukan Novin memiliki nilai rata-rata akurasi 48% untuk semua perintah suara, Ansel memiliki nilai rata-rata akurasi 60% untuk semua perintah suara, Ibnu memiliki nilai rata-rata akurasi 48% untuk semua perintah suara, Aldi memiliki nilai rata-rata akurasi 54% untuk semua perintah suara, Diki memiliki nilai rata-rata akurasi 60% untuk semua perintah suara. Yang dapat disimpulkan akurasi tertinggi perintah suara dengan motor dc adalah Ansel dan Diki. Hasil tersebut ditampilkan pada Gambar 4.11 grafik batang.

Nilai 0% untuk perintah "Kiri" menunjukkan bahwa sistem tidak berhasil mendeteksi atau melaksanakan perintah ini sama sekali dari semua orang yang diuji. Kemungkinan besar, ini bisa disebabkan oleh berbagai faktor seperti kesalahan dalam pengenalan suara untuk perintah "Kiri", kekurangan data pelatihan yang memadai untuk perintah ini, atau bahkan masalah teknis dengan mikrofon atau perangkat yang digunakan untuk pengujian. Ini menunjukkan bahwa perintah "Kiri" mungkin memerlukan perbaikan signifikan dalam sistem pengenalan suara atau penyesuaian dalam proses pelatihan.

Untuk perintah "Kanan", nilai 0% hanya terjadi pada Ibnu, yang menunjukkan bahwa ada kesalahan khusus pada saat pengujian Ibnu, seperti kebisingan latar belakang yang tinggi. Ini juga bisa menunjukkan bahwa model mungkin tidak cukup sensitif atau terlatih untuk mengenali variasi suara yang diberikan oleh Ibnu untuk perintah ini.

4.5 Pengujian Jarak Terbaik

4.5.1 Pengujian Jarak Terbaik

Dalam pengujian ini bertujuan untuk mengetahui jarak terbaik pengenalan perintah suara dari sistem. Parameter yang digunakan untuk mengukur adalah sebuah meteran sepanjang 1 m dimana jarak ditentukan maksimal 1 m dengan jarak 20 cm, 30 cm, 50 cm, 75 cm, 100 cm. Berdasarkan hasil pengukuran tersebut dapat diketahui akurasi sistem pada jarak yang terbaik dalam merespon aksi yang diberikan. Nanti saat penelitian dimulai pengujian akan dilakukan sebanyak 10 percobaan disetiap perintah suara.

4.5.2 Alat yang Digunakan Pengujian Jarak Terbaik

Berikut ini alat yang digunakan untuk melakukan pengambilan data pada bagian pengujian klasifikasi perintah suara:

1. Laptop
2. *Smartphone*
3. *Mobile robot*
4. Kabel USB
5. EPS32

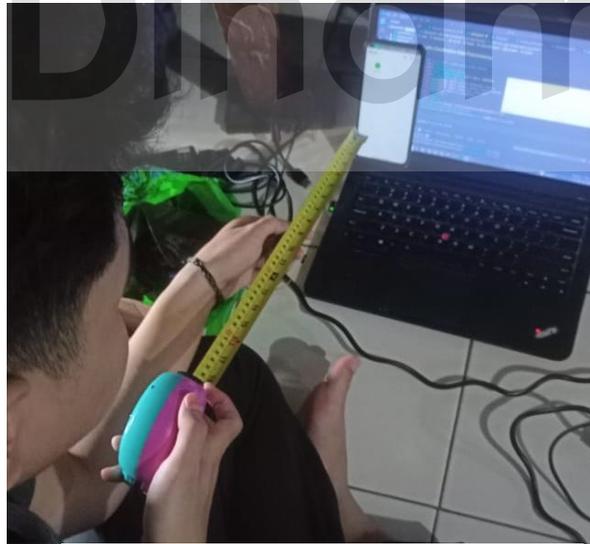
6. Meteran

4.5.3 Cara Pengujian Jarak Terbaik

Berikut merupakan prosedur pengujian klasifikasi perintah suara:

1. Menyalakan laptop untuk mengakses aplikasi *visual studio code* dan aplikasi Arduino IDE.
2. Menjalankan program TA_Rizal_Robot.ipynb yang berada di lampiran.
3. Menghubungkan laptop dengan ESP32 yang terdapat pada *mobile robot*.
4. Mengunggah program robot TA_Rizal_Robot.ino yang berada di lampiran.
5. Menyebutkan perintah suara berupa maju, mundur, kanan, kiri, dan berhenti sebanyak 10 kali untuk setiap perintah suara, maka program akan mengirim perintah suara ke ESP32 dengan komunikasi WiFi UDP.
6. Memperhatikan motor DC bekerja sesuai dengan perintah suara yang telah diterima dan dijalankan.
7. Melakukan penyebutan perintah suara dengan lima jarak yang berbeda dengan pengukuran 20 cm, 30 cm, 50 cm, 75 cm, dan 100 cm menggunakan meteran.

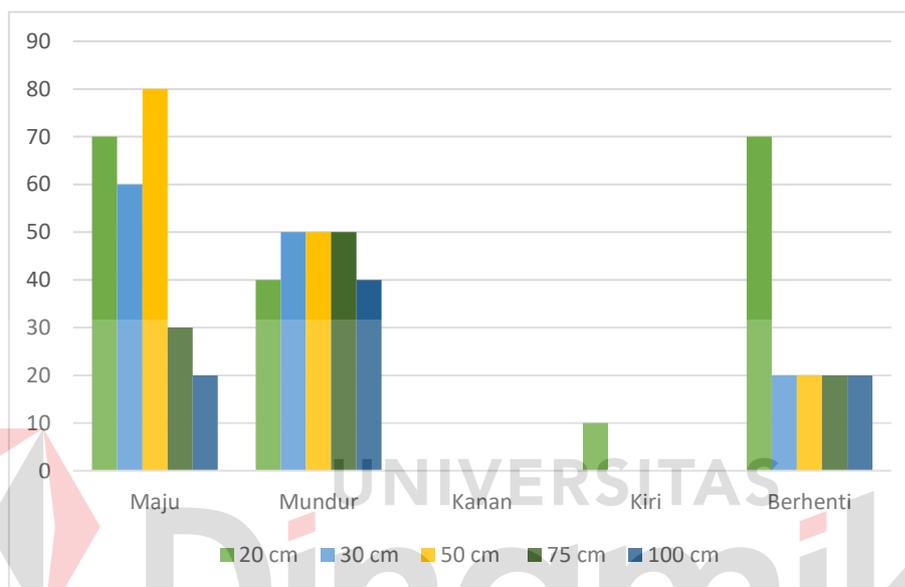
4.5.4 Hasil Pengujian Jarak Terbaik



Gambar 4.12 Pengujian Jarak Terbaik

Tabel 4.5 Hasil Pengujian Jarak Terbaik

No.	Jarak (cm)	Perintah Suara				
		Maju	Mundur	Kanan	Kiri	Berhenti
1	20	70%	40%	0%	10%	70%
2	30	60%	50%	0%	0%	20%
3	50	80%	50%	0%	0%	20%
4	75	30%	50%	0%	0%	20%
5	100	20%	40%	0%	0%	20%



Gambar 4.13 Grafik Pengujian Jarak Terbaik DC

4.5.5 Analisis Pengujian Jarak Terbaik

Berdasarkan Lampiran 9 yang merupakan tabel pengujian jarak 20 cm. Dalam 10 kali percobaan pada jarak 20 cm dengan perintah suara maju memiliki akurasi 70% yang terdiri dari 7 kali perintah suara maju dan 3 kali perintah suara mundur. Perintah suara mundur memiliki akurasi 40% yang terdiri dari 4 kali perintah suara mundur, 4 kali perintah suara maju dan 2 kali perintah suara berhenti. Perintah suara kanan memiliki akurasi 0% yang terdiri dari 3 kali perintah suara mundur, 4 kali perintah suara maju dan 3 kali perintah suara berhenti. Perintah suara kiri memiliki akurasi 10% yang terdiri dari 1 kali perintah suara kiri, 1 kali perintah suara mundur dan 8 kali perintah suara maju. Perintah suara berhenti memiliki akurasi 70% yang terdiri dari 7 kali perintah suara berhenti, 1 kali perintah suara mundur dan 2 kali perintah suara maju.

Berdasarkan Lampiran 10 yang merupakan tabel pengujian jarak 30 cm. Pada jarak 30 cm dengan perintah suara maju memiliki akurasi 60% yang terdiri dari 6 kali perintah suara maju, 2 kali perintah suara mundur dan 2 kali perintah suara berhenti. Perintah suara mundur memiliki akurasi 50% yang terdiri dari 5 kali perintah suara mundur, 2 kali perintah suara berhenti dan 3 kali perintah suara maju. Perintah suara kanan memiliki akurasi 0% yang terdiri dari 7 kali perintah suara maju, dan 3 kali perintah suara mundur. Perintah suara kiri memiliki akurasi 0% yang terdiri dari 3 kali perintah suara mundur, dan 7 kali perintah suara maju. Perintah suara berhenti memiliki akurasi 20% yang terdiri dari 2 kali perintah suara berhenti, 2 kali perintah suara mundur dan 5 kali perintah suara maju.

Berdasarkan Lampiran 11 yang merupakan tabel pengujian jarak 50 cm. Pada jarak 50 cm dengan perintah suara maju memiliki akurasi 80% yang terdiri dari 8 kali perintah suara maju, dan 2 kali perintah suara mundur. Perintah suara mundur memiliki akurasi 50% yang terdiri dari 5 kali perintah suara mundur, dan 5 kali perintah suara maju. Perintah suara kanan memiliki akurasi 0% yang terdiri dari 4 kali perintah suara maju, dan 6 kali perintah suara mundur. Perintah suara kiri memiliki akurasi 0% yang terdiri dari 9 kali perintah suara mundur, dan 1 kali perintah suara berhenti. Perintah suara berhenti memiliki akurasi 20% yang terdiri dari 2 kali perintah suara berhenti, 7 kali perintah suara mundur, dan 1 kali perintah suara maju.

Berdasarkan Lampiran 12 yang merupakan tabel pengujian jarak 75 cm. Pada jarak 75 cm dengan perintah suara maju memiliki akurasi 30% yang terdiri dari 3 kali perintah suara maju, dan 2 kali perintah suara mundur. Perintah suara mundur memiliki akurasi 50% yang terdiri dari 5 kali perintah suara mundur, dan 5 kali perintah suara maju. Perintah suara kanan memiliki akurasi 0% yang terdiri dari 4 kali perintah suara maju, dan 6 kali perintah suara mundur. Perintah suara kiri memiliki akurasi 0% yang terdiri dari 9 kali perintah suara mundur, dan 1 kali perintah suara berhenti. Perintah suara berhenti memiliki akurasi 20% yang terdiri dari 2 kali perintah suara berhenti, 7 kali perintah suara mundur, dan 1 kali perintah suara maju.

Berdasarkan Lampiran 13 yang merupakan tabel pengujian jarak 100 cm. Pada jarak 100 cm dengan perintah suara maju memiliki akurasi 20% yang terdiri

dari 2 kali perintah suara maju, dan 8 kali perintah suara mundur. Perintah suara mundur memiliki akurasi 40% yang terdiri dari 4 kali perintah suara mundur, 1 kali perintah suara berhenti, dan 5 kali perintah suara maju. Perintah suara kanan memiliki akurasi 0% yang terdiri dari 1 kali perintah suara maju, 2 kali perintah suara berhenti, dan 7 kali perintah suara mundur. Perintah suara kiri memiliki akurasi 0% yang terdiri dari 6 kali perintah suara mundur, dan 4 kali perintah suara maju. Perintah suara berhenti memiliki akurasi 20% yang terdiri dari 2 kali perintah suara berhenti, 7 kali perintah suara mundur, dan 1 kali perintah suara maju.

Pada Tabel 4.5 dilakukan total 250 kali pengujian oleh satu orang dengan 10 kali percobaan pada setiap perintah suara dengan lima jarak yang berbeda yaitu 20 cm, 30 cm, 50 cm, 75 cm, 100 cm yang diukur menggunakan meteran. Hasil analisis pada pengujian jarak terbaik yang telah dilakukan pada jarak 20 cm memiliki nilai rata-rata akurasi 38% untuk semua perintah suara, pada jarak 30 cm memiliki nilai rata-rata akurasi 36% untuk semua perintah suara, pada jarak 50cm memiliki nilai rata-rata akurasi 30% untuk semua perintah suara, pada jarak 75 cm memiliki nilai rata-rata akurasi 30% untuk semua perintah suara, pada jarak 100 cm memiliki nilai rata-rata akurasi 16% untuk semua perintah suara. Yang dapat disimpulkan jarak terbaik untuk memberikan perintah suara adalah 20 cm atau kurang karena mikropon *smartphone* dapat lebih cepat menerima suara tanpa harus terganggu oleh lebih banyak *noise* atau gangguan yang ada. Hasil tersebut ditampilkan pada Gambar 4.13 grafik batang.

BAB V PENUTUP

5.1 Kesimpulan

Setelah dilakukan pengujian terdapat beberapa point penting. Berikut adalah beberapa poin penting dari penelitian ini:

1. Proses *testing* klasifikasi perintah suara sebanyak 150 percobaan secara langsung melalui mikrofon *smartphone* dengan perintah suara maju mempunyai akurasi 66.67%, perintah suara mundur mempunyai akurasi 56.67%, perintah suara kanan mempunyai akurasi 66.67%, perintah suara kiri mempunyai akurasi 53.33% dan perintah suara berhenti mempunyai akurasi 76.67%.
2. Proses *testing* klasifikasi perintah suara dengan motor DC sebanyak 150 percobaan secara langsung dengan lima orang berbeda melalui mikrofon *smartphone* dengan perintah suara maju mempunyai akurasi 86.67%, perintah suara mundur mempunyai akurasi 0%, perintah suara kanan mempunyai akurasi 0%, perintah suara kiri mempunyai akurasi 3.33% dan perintah suara berhenti mempunyai akurasi 100%.
3. Proses *testing* klasifikasi perintah suara dengan motor DC sebanyak 10 percobaan secara langsung dengan lima orang berbeda melalui mikrofon *smartphone* dengan perintah suara maju mempunyai akurasi 50%, perintah suara mundur mempunyai akurasi 40%, perintah suara kanan mempunyai akurasi 60%, perintah suara kiri mempunyai akurasi 0% dan perintah suara berhenti mempunyai akurasi 90%. Dari orang kedua perintah suara maju mempunyai akurasi 70%, perintah suara mundur mempunyai akurasi 90%, perintah suara kanan mempunyai akurasi 90%, perintah suara kiri mempunyai akurasi 0% dan perintah suara berhenti mempunyai akurasi 50%. Dari orang ketiga perintah suara maju mempunyai akurasi 90%, perintah suara mundur mempunyai akurasi 90%, perintah suara kanan mempunyai akurasi 0%, perintah suara kiri mempunyai akurasi 0% dan perintah suara berhenti mempunyai akurasi 70%. Dari orang keempat perintah suara maju mempunyai akurasi 70%, perintah suara mundur mempunyai akurasi 90%, perintah suara

kanan mempunyai akurasi 90%, perintah suara kiri mempunyai akurasi 0% dan perintah suara berhenti mempunyai akurasi 100%. Dari orang kelima perintah suara maju mempunyai akurasi 100%, perintah suara mundur mempunyai akurasi 80%, perintah suara kanan mempunyai akurasi 30%, perintah suara kiri mempunyai akurasi 0% dan perintah suara berhenti mempunyai akurasi 90%.

4. Proses *testing* klasifikasi perintah suara dengan motor DC sebanyak 250 percobaan secara langsung melalui mikrofon *smartphone* dengan jarak terbaik sebanyak 10 percobaan setiap perintah suara dengan perintah secara langsung pada jarak 20 cm mempunyai akurasi 28% untuk semua perintah suara, pada jarak 50 cm dan 75 cm mempunyai akurasi 30% untuk semua perintah suara, pada jarak 30 cm mempunyai akurasi 26% untuk semua perintah suara, pada jarak 100 cm mempunyai akurasi 16% untuk semua perintah suara.

5.2 Saran

Berdasarkan hasil penelitian ini, kami menyarankan beberapa langkah untuk pengembangan lebih lanjut dan perbaikan sistem:

1. Penelitian lebih lanjut dapat dilakukan untuk menguji berbagai arsitektur model deep learning lainnya seperti LSTM (*Long Short-Term Memory*) untuk meningkatkan performa.
2. Mengintegrasikan sistem ini dengan *platform* IoT (Internet of Things) dapat memberikan kontrol dan monitoring yang lebih baik, serta memungkinkan analisis data secara real-time.
3. Penerapan metode lain untuk *pre-processing* pada sinyal suara agar, noise dari sinyal suara tersebut benar-benar tereduksi.

DAFTAR PUSTAKA

- Adnan, F., Amelia, I. dan Shiddiq, S. 'Umar (2022) "Implementasi Voice Recognition Berbasis Machine Learning," *Implementasi Voice Recognition Berbasis Machine Learning*, 11(1), hal. 24–29. Tersedia pada: <https://doi.org/https://doi.org/10.15294/eej.v11i1.57283>.
- Akbar, R. (2020) "SISTEM KUNCI KENDARAAN BERMOTOR MENGGUNAKAN RADIO FREQUENCY IDENTIFICATION (RFID) DAN SIM BERBASIS NODEMCU ESP32," *Jurnal Ekonomi Volume 18, Nomor 1 Maret 201*, 2(1), hal. 41–49.
- Alamsyah, N. (2022) "Rancang Bangun Dan Implementasi Sistem Kendali Robot Penanggulangan Bencana Alam Pantai Angin Mammiri Makassar," *Bulletin of Information Technology (BIT)*, 3(4), hal. 407–412. Tersedia pada: <https://doi.org/10.47065/bit.v3i4.434>.
- Arianda, G.R. (2024) "SISTEM KONTROL KECEPATAN KIPAS ANGIN MENGGUNAKAN SUARA MULTI - BAHASA MELALUI KLASIFIKASI AUDIO PADA YAMNET," 4(02), hal. 7823–7830. Tersedia pada: <http://repository.dinamika.ac.id/id/eprint/7599>.
- Arslan, A. dan Yildiz, O. (2018) "Automated auscultative diagnosis system for evaluation of phonocardiogram signals associated with heart murmur diseases," *Gazi University Journal of Science*, 31(1), hal. 112–124.
- Arvyantomo, M. dan Ratama, N. (2023) "Analisis Sentimen Masyarakat Indonesia terhadap Invasi Rusia di Ukraina menggunakan Metode Naive Bayes pada Media Sosial Facebook," *LOGIC: Jurnal Ilmu Komputer ...*, 1(4), hal. 705–717. Tersedia pada: <https://www.journal.mediapublikasi.id/index.php/logic/article/view/2142>.
- Azmi, K., Defit, S. dan Sumijan, S. (2023) "Implementasi Convolutional Neural Network (CNN) Untuk Klasifikasi Batik Tanah Liat Sumatera Barat," *Jurnal Unitek*, 16(1), hal. 28–40. Tersedia pada: <https://doi.org/10.52072/unitek.v16i1.504>.
- Buggs, C. (2023) *Wo Mic turns your phone into a mic*, Softonic. Tersedia pada: <https://wo-mic.en.softonic.com/>.
- Dyarrbirru, Z. dan Hidayat, S. (2020) "Metode Wavelet-MFCC dan Korelasi dalam Pengenalan Suara Digit," *JTIM: Jurnal Teknologi Informasi dan Multimedia*, 2(2), hal. 100–108. Tersedia pada: <https://doi.org/10.35746/jtim.v2i2.99>.
- Fuad, R.N. dan Winata, H.N. (2017) "Aplikasi Keamanan File Audio Wav (Waveform) Dengan Terapan Algoritma Rsa," *InfoTekJar (Jurnal Nasional Informatika dan Teknologi Jaringan)*, 1(2), hal. 113–119.

Tersedia pada: <https://doi.org/10.30743/infotekjar.v1i2.72>.

Jaya, H. *dkk.* (2018) *Kecerdasan Buatan, Journal of Chemical Information and Modeling*. Tersedia pada: <https://doi.org/10.1128/AAC.03728-14>.

Khairudin, M. *dkk.* (2020) “The mobile robot control in obstacle avoidance using fuzzy logic controller,” *Indonesian Journal of Science and Technology*, 5(3), hal. 334–351. Tersedia pada: <https://doi.org/10.17509/ijost.v5i3.24889>.

Koralage, R. (2020) *Connecting Azure DevOps Repo with Visual Studio Code, Medium*. Tersedia pada: <https://randulakorlage82.medium.com/connecting-azure-devops-repo-with-visual-studio-code-8bab447f49de>.

Lubis, Z. (2022) “Model Terbaru menggunakan perintah suara Untuk menstater Mesin Mobil dan keamanannya menggunakan SmartPhone Berbasis Arduino UNO,” *JET (Journal of Electrical Technology)*, 7(2), hal. 100–104. Tersedia pada: <https://jurnal.uisu.ac.id/index.php/jet/article/view/5404%0Ahttps://jurnal.uisu.ac.id/index.php/jet/article/download/5404/3939>.

Lulu, N.D. (2022) *Penyiram Tanaman Otomatis, Universitas Ciputra Creating World Class Entrepreneurs Informatics Study Program*. Tersedia pada: <https://informatika.ciputra.ac.id/2022/01/penyiram-tanaman-otomatis/>.

Movsessian, A. dan Nagaty, K.A. (2013) “A New Approach for Peer-to-Peer Distributed Computation,” (June 2013). Tersedia pada: <https://www.researchgate.net/publication/237062986>.

Nasution, T. (2012) “Metoda Mel Frequency Cepstrum Coefficients (MFCC) untuk Mengenali Ucapan pada Bahasa Indonesia,” *SATIN - Sains dan Teknologi Informasi*, 1, hal. 22–31.

Nisa, C. dan Candra, F. (2023) “Klasifikasi Jenis Rempah-Rempah Menggunakan Algoritma Convolutional Neural Network,” *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, 4(1), hal. 78–84. Tersedia pada: <https://doi.org/10.57152/malcom.v4i1.1018>.

Noertjahyana, A. dan Adipranata, R. (2003) “Implementasi Sistem Pengenalan Suara Menggunakan SAPI 5.1 dan DELPHI 5,” *Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra*, 4, hal. 107–114.

Nusyirwan, D. (2020) “Tong Sampah Pintar Dengan Perintah Suara Guna Menghilangkan Perilaku Siswa Membuang Sampah Sembarangan Di Sekolah,” *Jurnal Teknoinfo*, 14(1), hal. 48. Tersedia pada: <https://doi.org/10.33365/jti.v14i1.336>.

Oelsen, C. (2016) *File:Jupyter logo.svg, WIKIMEDIA COMMONS*. Tersedia pada:

https://commons.wikimedia.org/wiki/File:Jupyter_logo.svg.

- Pamungkas, D.S. dan Noviansyah, M.S. (2021) “Simulator Robot Lengan Dua Derajat Kebebasan,” *Seminar Nasional Terapan Riset Inovatif (SENTRINOV)*, 7(<https://proceeding.isas.or.id/index.php/sentrinov/issue/view/11>), hal. 51–57.
- Permana, I.S., Nurhasanah, Y.I. dan Zulkarnaik, A. (2018) “Dan Wanita,” 3(1), hal. 49–63. Tersedia pada: <https://doi.org/https://doi.org/10.26760/mindjournal>.
- Putra, D.M., Kalsum, T.U. dan Susanto, A. (2022) “Robot Berkaki Empat Pendeteksi Cahaya Dan Penghindar Rintangan Berbasis Arduino uno,” *JURNAL MEDIA INFOTAMA*, 18(2), hal. 360–366. Tersedia pada: <https://doi.org/10.37676/jmi.v18i2.2917>.
- Putra, O.V., Musthafa, A. dan Kholil, M. (2021) “Klasifikasi Intonasi Bahasa Jawa Khas Ponorogo Menggunakan Algoritma Multilayer Perceptron Neural Network,” *Prosiding Seminar Nasional Penelitian Dan Pengabdian 2021*, hal. 459–464.
- Reuter, F. *dkk.* (2024) “ESP32Series,” *Sensors*, 24(1), hal. 1–31.
- Risdiandi, R. (2021) “Analisis Cara Kerja Sensor Ultrasonik Menggunakan Mikrokontroler Arduino Uno Untuk Merancang Alat Deteksi Banjir Secara Otomatis,” *OSF Preprints. January*, 1(January 2020), hal. 1. Tersedia pada: <https://doi.org/10.13140/RG.2.2.24386.61123>.
- Savitri, A.D. (2023) “KLASIFIKASI JENIS KIDUNG BALI BERDASARKAN CITRA SPECTOGRAM SUARA MENGGUNAKAN K-NEAREST NEIGHBOR,” 8(2), hal. 1–9.
- Sirajuddin (2013) “Rancang Bangun Robot Terbang Quadcopter Berbasis Mikrokontroler ATmega 16,” hal. 1–8. Tersedia pada: <https://doi.org/oai:jurnal.untan.ac.id:article/1322>.
- Soim, S. *dkk.* (2015) “Perancangan Robot Humanoid Berbasis Mikrokontroler Atmega 32,” *Seminar Nasional Sains dan Teknologi 2015*, 1(2), hal. 22. Tersedia pada: <https://doi.org/10.31851/ampere.v1i2.898>.
- Watson, D. (2020) *ESP32 Pinout, Datasheet, Features & Applications, THE ENGINEERING PROJECTS*. Tersedia pada: <https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html>.