

**IMPLEMENTASI KEYWORD SPOTTING UNTUK MENGERAKKAN ROBOT
STRANDBEEST BERTEKNOLOGI TINYML**

SKRIPSI

ALVIN DAELI

181402054



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

**IMPLEMENTASI KEYWORD SPOTTING UNTUK MENGERAKKAN ROBOT
STRANDBEEST BERTEKNOLOGI TINYML**

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana
Teknologi Informasi

ALVIN DAELI

181402054



PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

MEDAN

2024

PERSETUJUAN

Judul : IMPLEMENTASI KEYWORD SPOTTING UNTUK
MENGERAKKAN ROBOT STRANDBEEST
BERTEKNOLOGI TINYML

Kategori : SKRIPSI

Nama Mahasiswa : Alvin Daeli

Nomor Induk Mahasiswa : 181402054

Program Studi : Sarjana (S-1) Teknologi Informasi

Fakultas : Ilmu Komputer dan Teknologi Informasi
Universitas Sumatera Utara

Medan, 4 Juli 2024

Komisi Pembimbing:

Pembimbing 2

Dr. Niskarto Zendrato S.Kom., M.Kom
NIP. 198909192018051001

Pembimbing 1

Dr. Baihaqi Siregar S.Si., M.T.
NIP. 197901082012121002

Diketahui/disetujui oleh

Program Studi S1 Teknologi Informasi

Ketua,

Dedy Arisandi S.T., M.Kom.
NIP. 197908312009121002

PERNYATAAN**IMPLEMENTASI KEYWORD SPOTTING UNTUK MENGERAKKAN ROBOT
STRANDBEEST BERTEKNOLOGI TINYML****SKRIPSI**

Saya mengakui bahwa skripsi ini merupakan hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 4 Juli 2024

Alvin Daeli

181402054

UCAPAN TERIMA KASIH

Puji dan syukur penulis sampaikan kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya penulis dapat menyelesaikan penulisan skripsi ini dimana merupakan salah satu syarat yang harus dipenuhi guna memperoleh gelar Sarjana pada Program Studi S1 Teknologi Informasi di Universitas Sumatera Utara. Penulis mampu menyelesaikan skripsi tidak hanya dengan kemampuan pribadi, melainkan dari berbagai pihak yang terlibat dalam mendoakan, membimbing serta mendukung penulis dari mulai kuliah hingga sampai menyelesaikan skripsi. Penulis ingin mengucapkan terima kasih kepada kedua orang tua penulis, Bapak Frans Fele Daeli dan Ibu Linda Wati yang tiada henti mendoakan, mendukung dan memberi semangat hingga sampai tahap penyelesaian skripsi ini. Penulis juga ingin mengucapkan terima kasih terkhusus kepada:

1. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
2. Bapak Dedy Arisandi S.T., M.Kom. dan Bapak Ivan Jaya S.Si., M.Kom., selaku Ketua Program Studi dan Sekretaris Program Studi S1 Teknologi Informasi Universitas Sumatera Utara.
3. Bapak Baihaqi Siregar S.Si., M.T. selaku Dosen Pembimbing I dan Bapak Niskarto Zendrato S.Kom., M.Kom selaku Dosen Pembimbing II yang telah membantu penulis dalam penyelesaian skripsi ini baik dari segi bimbingan hingga kritik dan saran yang diberikan selama proses penyusunan skripsi.
4. Ibu Dr. Marischa Elveny S.Ti., M.Kom. selaku Dosen Pembimbing I dan Ibu Ade Sarah Huzaifah S.Kom., M.Kom selaku Dosen Pembimbing II yang telah memberi banyak masukan untuk penyempurnaan skripsi penulis.
5. Para Dosen Program Studi S1 Teknologi Informasi yang telah mendidik dan mengajarkan penulis selama kuliah dari tahun 2018.
6. Bapak Indra Aulia S.Ti., M.Kom. yang telah membimbing, mengajar dan memberi masukan kepada penulis selama penyusunan skripsi.
7. Teman-teman seperjuangan, Ammar Rafi Afandi Hasibuan, Rasyid Hafiz, Jimmy Widiyanto, Willi Nardo, Ali Akbar Sinaga, Tiara Amalia dan Ruth

Calista Paulina Sianipar yang telah menemani penulis dalam perkuliahan dan penyusunan skripsi.

8. Para abang dan kakak senior penulis, Bang Fitra Nurmayadi, Bang Muhammad Putra Yuszar, Bang Muhibuddin, Bang Widang Muttaqin dan Kak Syarifah Atika yang telah membantu penulis dalam menyelesaikan skripsi.
9. Seluruh pegawai dan staf Program Studi S1 Teknologi Informasi dan Fakultas Ilmu Komputer – Teknologi Informasi yang telah membantu berbagai kebutuhan akademik maupun administrasi penulis.
10. Teman-teman seangkatan 2018 Teknologi Informasi yang telah bersama-sama menempuh pendidikan kuliah dari awal hingga sampai penyusunan skripsi.
11. Berbagai pihak yang telah membantu penulis dalam penyusunan skripsi baik secara langsung maupun tidak langsung yang penulis tidak dapat sebutkan satu-persatu.

Medan, 4 Juli 2024

Penulis

ABSTRAK

Perkembangan teknologi robotik memberikan banyak kontribusi yang signifikan di bidang yang penuh ragam seperti industri bahkan sampai pada kegiatan harian dimana disediakan otomatisasi yang efektif dan efisien yang membantu pekerjaan manusia. Adapun tantangan dalam robotik adalah membuat metode kontrol robot yang lebih mudah, seperti kontrol suara yang dimana dapat meningkatkan interaksi antar manusia dengan robot. Penelitian ini mengusulkan tentang implementasi teknologi Tiny Machine Learning (TinyML) untuk Keyword Spotting pada robot Strandbeest. TinyML memungkinkan penerapan kecerdasan buatan pada sistem dengan sumber daya terbatas, dimana sistem ini sering diimplementasikan pada robotik yang portabel. Penelitian ini melatih empat kata kunci: “maju”, “mundur”, “kiri”, dan “kanan” untuk mengatur arah gerak robot. Robot yang digunakan adalah robot Strandbeest dengan rangkaian ESP32 sebagai mesin utama robot, dirangkai pada L298N Motorshield dan Arduino Nano 33 BLE Sense Lite sebagai mikrofon. Dataset yang digunakan merupakan dataset suara penulis yang direkam dan terdiri atas berbagai variasi pengucapan kata kunci untuk memastikan model yang kokoh dengan akurasi yang optimal. Pada proses dataset suara, digunakan algoritma Mel-Frequency Cepstral Coefficients (MFCC) untuk membentuk model yang selanjutnya diproses oleh TensorFlow Lite sehingga menjadi *inference* dan akan ditanamkan pada Arduino Nano 33 BLE Sense Lite. Hasil yang diperoleh untuk algoritma MFCC berupa akurasi sebesar 99,34% dengan akurasi pada *real-time testing* sebesar 73%

Kata Kunci: *Keyword Spotting, TinyML, TensorFlow Lite, Mel-Frequency Cepstral Coefficients (MFCC), Strandbeest*

***IMPLEMENTATION OF KEYWORD SPOTTING TO MOBILIZE
STRANDBEEST ROBOT WITH TINYML TECHNOLOGY***

ABSTRACT

The development of robotic technology makes many significant contributions in various fields, both in industry and day-to-day activities which provide effectiveness and efficient automations that assist the human tasks. The challenge in robotics is to make robot control methods easier, such as voice control, which can increase interaction between humans and robots. This research proposes the implementation of Tiny Machine Learning (TinyML) technology for Keyword Spotting on Strandbeest robots. TinyML enables the application of artificial intelligence to resource-constrained systems, which are often implemented in portable robotics. This research trained four keywords: “*maju*” (forward), “*mundur*” (backward), “*kiri*” (left), and “*kanan*” (right) to set the robot's motion direction. The robot used is a Strandbeest robot with an ESP32 circuit as the main engine of the robot, connected to an L298N Motorshield and an Arduino Nano 33 BLE Sense Lite as a microphone. The dataset used is the author's recorded voice dataset and consists of various variations of keyword pronunciation to ensure a robust model with optimal accuracy. In the voice dataset process, the MFCC algorithm or widely known as Mel-Frequency Cepstral Coefficients is used to form a model which is then processed by TensorFlow Lite so that it becomes inference and will be embedded in the *edge devices*. The results obtained for the MFCC algorithm are an accuracy of 99.34% and 73% accuracy for real-time testing.

Keyword: Keyword Spotting, TinyML, TensorFlow Lite, Mel-Frequency Cepstral Coefficients (MFCC), Strandbeest

DAFTAR ISI

PERSETUJUAN	i
PERNYATAAN	ii
UCAPAN TERIMA KASIH	iii
ABSTRAK	v
ABSTRACT	vi
DAFTAR ISI	vii
DAFTAR TABEL	ix
DAFTAR GAMBAR	x
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan Penelitian	4
1.4 Batasan Masalah	4
1.5 Manfaat Penelitian	4
1.6 Metodologi Penelitian	4
1.7 Sistematika Penulisan	5
BAB 2 LANDASAN TEORI	7
2.1 <i>Strandbeest</i>	7
2.2 TinyML (<i>Tiny Machine Learning</i>)	8
2.3 MFCC (<i>Mel-Frequency Cepstral Coefficients</i>)	11
2.4 <i>Keyword Spotting</i>	15
2.5 Arduino Nano 33 BLE Sense Lite	17
2.6 ESP32	17
2.7 Driver Motor L298N	18
2.8 <i>TensorFlow</i>	19
2.9 Penelitian Terdahulu	21

BAB 3	ANALISIS DAN PERANCANGAN SISTEM	26
3.1	Arsitektur Umum	26
3.1.1	<i>Pre-Processing</i>	27
3.1.2	<i>Post-Processing</i>	37
3.1.3	Data yang Digunakan	37
3.2	Perancangan Sistem	39
3.2.1	Komponen <i>Robot Strandbeest</i>	39
3.2.2	<i>Flowchart</i> Sistem Robot Strandbeest	39
3.2.3	Rangkaian Sistem <i>Robot Strandbeest</i>	41
3.3	Evaluasi Model	44
BAB 4	IMPLEMENTASI DAN PENGUJIAN SISTEM	45
4.1	Implementasi Sistem	45
4.1.1	Spesifikasi Sistem <i>Hardware</i>	45
4.1.2	Spesifikasi Sistem <i>Software</i>	45
4.2	Pengujian Sistem	46
4.2.1	Pengujian Fungsional	46
4.2.2	Pengujian Sistem <i>Keyword Spotting</i>	47
BAB 5	KESIMPULAN DAN SARAN	52
5.1	Kesimpulan	52
5.2	Saran	52
DAFTAR PUSTAKA		53

DAFTAR TABEL

Tabel 2.1. Penelitian Terdahulu	23
Tabel 2.2. Penelitian Penulis	25
Tabel 3.1. Tabel Komponen Alat	39
Tabel 4.1. Tabel Spesifikasi Hardware	45
Tabel 4.2. Tabel Spesifikasi Software	46
Tabel 4.3. Tabel Library	46
Tabel 4.4. Pengujian Fungsional	47
Tabel 4.5. Data Testing	48
Tabel 4.6. Pengujian Model pada Data Testing	48
Tabel 4.7. Data Uji Coba Langsung	50
Tabel 4.8. Pengujian Model dengan Data Uji Coba Langsung	51

DAFTAR GAMBAR

Gambar 2.1. <i>Strandbeest</i>	8
Gambar 2.2. Arduino Nano 33 BLE Sense	17
Gambar 2.3. ESP32	18
Gambar 2.4. Driver Motor L298N	19
Gambar 3.1. Arsitektur Umum	26
Gambar 3.2. <i>Pseudocode Resampling Audio</i>	27
Gambar 3.3. <i>Pseudocode Segmentation Audio</i>	28
Gambar 3.4. <i>Pseudocode AugmentAudio</i> penambahan <i>noise</i>	29
Gambar 3.5. <i>Pseudocode AugmentAudio</i> pengubahan <i>pitch</i>	30
Gambar 3.6. <i>Pseudocode ExtractMFCC</i>	30
Gambar 3.7. <i>Pseudocode ExtractMFCC</i> Tahap <i>Pre-emphasis</i>	31
Gambar 3.8. <i>Pseudocode ExtractMFCC</i> Tahap <i>Framing</i>	32
Gambar 3.9. <i>Pseudocode ExtractMFCC</i> Tahap <i>Windowing</i>	32
Gambar 3.10. <i>Pseudocode ExtractMFCC</i> Tahap <i>FFT dan Power Spectrum</i>	33
Gambar 3.11. <i>Pseudocode ExtractMFCC</i> Tahap <i>Mel Filterbank</i>	33
Gambar 3.12. <i>Pseudocode ExtractMFCC</i> Tahap <i>Log Mel Spectrum</i>	33
Gambar 3.13. <i>Pseudocode ExtractMFCC</i> Tahap <i>Discrete Cosine Transform</i>	34
Gambar 3.14. <i>Pseudocode ExtractMFCC</i> (lanjutan)	34
Gambar 3.15. <i>Pseudocode NeuralNetworkTraining</i>	35
Gambar 3.16. <i>Pseudocode NeuralNetworkTraining</i> (lanjutan)	36
Gambar 3.17. <i>Flowchart</i> proses sistem <i>Robot Strandbeest</i>	40
Gambar 3.18. Desain <i>Robot Strandbeest</i>	41
Gambar 3.19. Desain Kaki <i>Robot Strandbeest</i>	42
Gambar 3.20. <i>Robot Strandbeest</i> yang sudah dirangkai	43
Gambar 3.21. <i>Robot Strandbeest</i> tampak belakang	44
Gambar 4.1. Hasil komunikasi UART antara Arduino Nano 33 dengan ESP32	47
Gambar 4.2. Hasil uji coba <i>model</i> dengan terdeteksi <i>noise</i>	49
Gambar 4.3. Hasil uji coba <i>model</i> dengan terdeteksi <i>unknown</i>	49
Gambar 4.4. Hasil uji coba <i>model</i> dengan terdeteksi <i>maju</i>	50
Gambar 4.5. Hasil uji coba <i>model</i> dengan terdeteksi <i>mundur</i>	50

Gambar 4.6. Hasil uji coba <i>model</i> dengan terdeteksi <i>kanan</i>	50
Gambar 4.7. Hasil uji coba <i>model</i> dengan terdeteksi <i>kiri</i>	50

BAB I

PENDAHULUAN

1.1 Latar Belakang

Melihat kemajuan teknologi terutama di bidang kecerdasan buatan berkembang semakin pesat akhir-akhir ini. Baik dalam industri maupun dalam aktivitas kehidupan sehari-hari. Kecerdasan buatan juga sudah mulai diterapkan pada berbagai sistem dan alat yang dapat membantu pekerjaan manusia. Adapun salah satu implementasi yang diterapkan adalah *Deep Learning*, yang merupakan konsep kecerdasan buatan yang didasari oleh *Artificial Neural Networks*. Untuk beberapa pengaplikasian, model yang dibangun berdasarkan *deep learning* lebih unggul dibandingkan dengan model pembelajaran mesin ataupun pendekatan analisis data yang tradisional dan umum digunakan. Adapun *Artificial Neural Networks* tersusun atas konsep matematis dari unit pemrosesan yang terhubung antar satu sama lain yang disebut dengan *neuron* buatan. Mengambil prinsip pemrosesan informasi didalam sistem biologis, setiap koneksi antar *neuron* dapat mengirim sinyal dengan kekuatan yang dapat diatur kuat lemahnya dengan *weight* yang seiring proses *learning* akan selalu disesuaikan sesuai kebutuhan (Janiesch et al., 2021).

Perkembangan teknologi robotik juga sudah mencapai pada tahap robot dapat dikendalikan baik menggunakan alat kendali ataupun melalui perintah suara. Robot dengan kontrol suara adalah sebuah contoh praktis dari pengendalian gerakan suatu robot sederhana dengan memberikan perintah suara yang sering digunakan sehari-hari (Chaudhry et al., 2019). Kehadiran robot cukup membantu pekerjaan manusia yang sifatnya repetitif, memiliki resiko yang cukup tinggi, maupun pekerjaan sederhana yang memiliki potensi untuk dikerjakan secara otomatis. Konsep rumah pintar (*Smart Home*) merupakan salah satu hal yang sering menggunakan robot ataupun perangkat Internet of Things (IoT) untuk membantu manusia dalam berbagai tugas sehari-hari. Adapun robot atau perangkat IoT yang dapat dikendalikan dengan perintah suara merupakan salah satu poin penting dalam meningkatkan kemudahan interaksi antar manusia dengan robot ataupun perangkat IoT yang digunakan (Erol et al., 2018).

Dalam pengendalian robot dengan kontrol suara, diperlukan suatu sistem yang diimplementasi pada mesin robot tersebut. TinyML (*Tiny Machine Learning*) merupakan versi ringan dari sistem *Machine Learning* yang dikembangkan terkhusus untuk implementasi pada sistem dengan sumber daya terbatas, seperti mikrokontroler. Dengan penggunaan daya hanya beberapa miliwatt atau kurang, TinyML memberi peluang untuk perangkat tertanam berbasis IoT untuk dapat menggunakan sistem berdaya rendah dengan berbagai modul manajemen daya yang canggih. (Ray, 2022)

Keyword Spotting (KWS) merupakan salah satu pendekatan yang dilakukan dalam pengembangan robot dengan kontrol suara yang diimplementasi dengan teknologi TinyML. KWS dapat diartikan sebagai proses identifikasi suatu kata kunci dari suara yang mengandung berbagai kata. Dalam implementasinya diperlukan *speech feature extraction* untuk mengekstrak *parameter-parameter* penting dari data yang kemudian akan dibuat menjadi *model* untuk mendeteksi perintah suara sesuai dengan yang diharapkan. *Mel-Frequency Cepstral Coefficients* (MFCCs) merupakan salah satu *speech feature extraction* yang sampai sekarang masih berkompeten dan paling optimal digunakan terkhususnya untuk kasus KWS dengan kata kunci yang berasal dari ucapan manusia (Lopez-Espejo et al., 2022).

Robot Strandbeest yang digunakan dalam penelitian ini merupakan robot kinetik yang diciptakan oleh Theo Jansen, seorang seniman dan insinyur di Belanda. Robot ini dikembangkan pada awal 1990 dengan tujuan membuat struktur yang mampu bergerak sendiri dengan menggunakan energi angin. Tujuan Robot Strandbeest digunakan dalam penelitian yang berkaitan dengan robot berkaki karena desain kaki Strandbeest yang dapat melewati berbagai permukaan seperti permukaan cekung maupun tanjakan (Kavlak & Yongul, 2022).

Terdapat beberapa penelitian yang mengarah pada pengembangan robot kendali suara maupun konsep mengolah suaranya, diantaranya oleh (Wang & Li, 2022), penulis membangun sistem Keyword Spotting (KWS) yang diimplementasi pada mikrokontroler STM32F7 dengan inti Cortex-M7 @216MHz dan RAM statis sebesar 512KB. Dengan arsitektur CNN yang berhasil menyederhanakan jumlah

operasi yang diperlukan untuk KWS agar dapat dijalankan pada *edge devices*. Hasil yang dicapai adalah sistem dapat mencetak hasil klasifikasi tiap ~37ms secara terus menerus.

Berikutnya oleh (Chaudhry et al., 2019) dimana penulis mengembangkan sistem robot dengan kendali suara berbasis Arduino ATmega. Sistem ini dibangun dengan aplikasi *android* sebagai medium untuk menyalurkan perintah manusia ke mikrokontroller. Kontroller dihubungkan dengan *module* Bluetooth melalui protokol UART, perintah yang diterima akan di proses oleh *module* suara lalu dikonversikan ke teks. Teks tersebut akan diproses oleh mikrokontroller dan hasilnya akan digunakan untuk mengambil keputusan yang akan menggerakkan robot. Penelitian oleh (Gouda et al., 2018) melakukan implementasi KWS melalui *Image Processing* dengan berbagai metode CNN ditambah dengan metode regularisasi *Virtual Adversarial Training* yang berhasil mencapai 92% akurasi dalam validasi 20% sampel data random dari data input.

Berdasarkan penelitian diatas, dapat dilihat bahwa terdapat beberapa penelitian yang sudah melakukan implemetasi *machine learning* (dalam kasus ini, *Keyword Spotting*) pada sistem dengan sumber daya terbatas seperti Arduino. Bahkan beberapa sistem di-implementasi pada *robot* dengan kendali suara berbasis Android. Namun belum terdapat sistem *robot Strandbeest* dengan kendali suara yang berbasis mikrokontroler sehingga penulis melaksanakan penelitian yang berjudul “Implementasi Keyword Spotting Untuk Menggerakkan Robot Strandbeest Berteknologi TinyML”

1.2 Rumusan Masalah

Robot Strandbeest merupakan robot yang dirangkai dari banyak rangkaian segitiga yang dihubungkan dengan engsel untuk menciptakan kaki-kaki yang dapat bergerak pada lingkungan terjal maupun permukaan cekung. Namun tidak banyak penelitian yang menggunakan *Robot Strandbeest* dengan perintah suara dalam kasus menggerakkan *robot* dengan teknologi *TinyML* berbasis *Keyword Spotting*.

1.3 Tujuan Penelitian

Penulis melakukan penelitian ini dengan tujuan untuk implementasi *Tiny Machine Learning* pada *robot Strandbeest* guna menggerakkan *robot* berbasis *Keyword Spotting*.

1.4 Batasan Masalah

Adapun diterapkan berbagai batasan masalah guna mencegah perluasan ruang lingkup penelitian, antara lain:

1. Kata kunci yang akan dilatih untuk menggerakkan robot ada empat (4), yaitu maju, mundur, kiri dan kanan.
2. Uji coba yang akan dilakukan bersifat *real-time*, sehingga akurasi yang dicapai pada tahap *testing* dapat berbeda dengan *real-time testing*.
3. Jarak antar sumber suara dengan mikrofon pada saat uji coba adalah lima (5) hingga lima belas (15) cm.
4. Lingkungan tempat uji coba berada dalam ruangan dengan *noise* sedikit.

1.5 Manfaat Penelitian

Terdapat juga beberapa manfaat penelitian yang telah dilakukan, dijabarkan sebagai berikut.

1. Meningkatkan ilmu pengetahuan di bidang implementasi *Keyword Spotting* dengan lingkungan sistem yang memiliki sumber daya terbatas seperti Arduino Nano 33 BLE Sense Lite.
2. Dihasilkan luaran yaitu *robot Strandbeest* sebagai prototipe untuk pembelajaran *Tiny Machine Learning* di lingkungan dengan sumber daya terbatas (Mikrokontroler).
3. Landasan dari penelitian lanjut mengenai pengendalian berbagai mikrokontroler dengan perintah suara.

1.6 Metodologi Penelitian

Adapun tahap yang akan dilaksanakan pada penelitian terdiri dari:

1. Studi Literatur

Merupakan proses dimana penulis akan melakukan studi berbagai referensi yang dapat digunakan sebagai dasar acuan dari *Keyword Spotting*, ESP32 dan Arduino Nano 33

BLE Sense Lite. Adapun referensi didapat melalui buku, jurnal, maupun artikel ilmiah yang relevan dengan topik penelitian.

2. Pengumpulan Data

Untuk tahap ini, penulis akan melakukan perekaman berbagai suara yang nantinya digunakan sebagai *data training* yang akan digunakan sebagai *dataset* untuk melatih robot.

3. Analisis Permasalahan

Merupakan tahap dimana penulis melakukan analisis data dan studi berbagai literatur dimana sudah dikumpulkan pada tahap sebelumnya.

4. Perancangan Sistem

Bagian dimana dilakukan perancangan suatu sistem terdiri dari rancang arsitektur, perakitan robot serta alat kendali robot sesuai dengan analisis yang sudah dilaksanakan pada tahap sebelumnya.

5. Implementasi

Penulis melakukan implementasi sistem robot yang sudah dirakit sedemikian rupa hingga tercapai suatu prototipe robot yang memenuhi tujuan penelitian yang didasari oleh studi berbagai literatur serta analisis yang sudah dilakukan oleh penulis.

6. Pengujian

Merupakan tahap percobaan atau pengujian sistem yang telah dibangun pada tahap sebelumnya guna memastikan sistem dapat berfungsi secara optimal.

7. Dokumentasi dan Penyusunan Laporan

Tahap dimana hasil penelitian akan didokumentasi dan disusun dalam bentuk laporan yang berdasarkan hasil dari analisis dan implementasi sistem serta pengujian sistem.

1.7 Sistematika Penulisan

Guna memberikan gambaran jelas dari penulisan ini, penulis menyusun skripsi ini berdasarkan bagian yang telah disajikan sebagai berikut.

Bab I: Pendahuluan

Tahap berikut ini akan dijabarkan latar belakang penelitian ini, rumusan masalah yang ingin diselesaikan peneliti, tujuan penelitian yang menjadi acuan penelitian, batasan masalah agar mencegah lingkup penelitian yang meluas ke arah yang tidak relevan,

serta manfaat penelitian yang akan diterima oleh pembaca dan sistematika penulisan laporan penelitian ini.

Bab II: Landasan Teori

Tahap ini dijelaskan berbagai teori yang akan melandasi penelitian ini, berupa *Keyword Spotting*, *Tiny Machine Learning*, *Mel-Frequency Cepstral Coefficient* (MFCC) dan *TensorFlow*.

Bab III: Metodologi Penelitian

Bagian berikut diberikan penjelasan mengenai data yang digunakan pada penelitian, arsitektur umum dari sistem yang akan dibuat serta pendekatan yang dilakukan untuk penerapan algoritma dalam kasus *Keyword Spotting*.

Bab IV: Implementasi dan Pengujian Sistem

Pada bagian ini akan dipaparkan tentang implementasi algoritma ke sistem dan serta pengujian suatu sistem yang telah dibangun.

Bab V: Kesimpulan dan Saran

Pada bagian ini diberikan hasil yang dicapai dari penelitian ini, disusun menjadi kesimpulan serta ditambahkan beberapa saran dari penulis yang dapat menjadi acuan untuk peningkatan penelitian yang serupa di waktu yang mendatang.

BAB II

LANDASAN TEORI

2.1 Strandbeest

Strandbeest merupakan patung kinetik (*kinetic sculpture*) yang dibuat oleh Theo Jansen, seorang artis asal Belanda. Nama Strandbeest dapat diartikan sebagai "hewan pantai" dalam bahasa Belanda. Strandbeest ini mulai dikembangkan dari awal 1990 hingga sekarang. Konsep Strandbeest ini sering dikembangkan oleh para peneliti khususnya dalam kasus penelitian yang berkaitan dengan robot berkaki karena desainnya yang mampu menanggulangi berbagai rintangan yang cekung maupun tanjakan (Kavlak & Yongul, 2022).

Adapun desain dan struktur dari Strandbeest ini berupa pipa PVC, botol plastik dan juga berbagai bahan sederhana lainnya. Dengan angin sebagai sumber energi, Strandbeest ini bergerak mulus dan stabil berkat botol-botol plastik yang terpasang pada struktur dimana berfungsi sebagai pompa udara dan penyimpanan energi. Saat angin bertiup, tekanan udara di dalam botol meningkat dan memicu mekanisme gerakan kaki-kaki Strandbeest. Mekanisme gerakan kaki-kaki Strandbeest memungkinkan untuk pergerakan di lingkungan yang sulit seperti terjal, berbatu, dan sebagainya (Kavlak & Kartal, 2021).

Sejak awal 1990, Jansen terus mengembangkan desain Strandbeest dengan berbagai inovasi seperti sistem sensor untuk deteksi air yang berguna agar Strandbeest menghindari pantai yang basah. Tujuan dan konsep dari Strandbeest ini tidak hanya sekadar karya seni kinetik, melainkan sebagai bentuk kehidupan yang baru dimana Jansen berharap untuk dapat menciptakan makhluk yang sepenuhnya mandiri dan mampu bertahan di pantai tanpa campur tangan manusia.

Strandbeest telah dipamerkan di berbagai galeri seni dan pameran dan juga telah menginspirasi berbagai orang dalam berbagai bidang seperti seni, desain, teknik, maupun pendidikan. Dengan konsep dan mekanisme yang unik, Strandbeest sering digunakan sebagai bahan studi untuk mempelajari berbagai prinsip kinetik dan biomekanika.

Adapun contoh dari Strandbeest dapat dilihat pada Gambar 2.1



Gambar 2.1. Strandbeest (Sumber: <https://www.strandbeest.com/strandbeest/2010-siamesis>)

2.2 TinyML (*Tiny Machine Learning*)

TinyML merupakan cabang dari pembelajaran mesin yang berfokus kepada implementasi model *Machine Learning* (ML) pada sistem dengan konsumsi daya serta *resources* yang terbatas seperti mikrokontroler. TinyML memungkinkan perangkat *edge* berbasis IoT untuk menggunakan daya yang lebih rendah dengan penggabungan berbagai modul manajemen daya sehingga tercapai konsumsi daya sekitar beberapa milliwatt atau lebih sedikit (Ray, 2022). Menurut (Abadade et al., 2023) terdapat keuntungan dari penggunaan TinyML dipaparkan sebagai berikut.

1. Efisiensi Energi

Dimana TinyML memungkinkan implementasi algoritma *Machine Learning* pada perangkat yang memiliki konsumsi daya sangat rendah, seperti mikrokontroler.

2. *Latency* Rendah

Dimana *model* yang dijalankan langsung pada perangkat akan mengurangi waktu latensi karena tidak perlu mengirim data ke *server cloud* untuk diproses.

3. Keamanan dan Privasi

Karena data diproses langsung pada perangkat, sehingga mengurangi resiko pelanggaran privasi dan meningkatkan keamanan data

4. Operasional Offline

TinyML dapat beroperasi pada perangkat tanpa konektivitas internet yang merupakan hal penting untuk implementasi sistem di daerah terpencil ataupun dengan konektivitas yang tidak stabil.

TinyML pada umumnya digunakan pada berbagai komponen seperti Mikrokontroler, yang dimana merupakan unit pemrosesan dengan daya rendah yang dapat menjalankan model pembelajaran mesin yang telah dikecilkan. Contohnya seperti Arduino dan ESP32. Adapun TinyML merupakan model pembelajaran mesin yang terbentuk berdasarkan algoritma yang telah dilatih menggunakan *dataset* tertentu, guna melaksanakan tugas yang diberikan seperti klasifikasi (*Classification*), deteksi objek (*Object Detection*), pemrosesan bahasa alami (*Natural Language Processing*) dan juga pengenalan suara (*Speech Recognition*). TinyML biasanya dikembangkan menggunakan *tools* dan *library* yang memudahkan dalam pengembangan dan implementasi model TinyML seperti *TensorFlow Lite for Microcontrollers*, *Edge Impulse*, dan *MicroTensor*. Sejauh ini, TinyML telah berfokus pada pengembangan *mobile inference* dimana telah dihasilkan berbagai kemajuan dalam model pembelajaran mesin seperti *pruning*, *sparsity* dan *quantization* (Banbury et al., 2020).

Proses pengembangan TinyML meliputi berbagai hal, namun proses yang dilakukan secara umum akan diuraikan sebagai berikut.

1. Pengumpulan Data

Proses ini melaksanakan pengumpulan berbagai data yang relevan dan dibutuhkan untuk proses pelatihan model.

2. Pelatihan Model

Proses ini melakukan pelatihan model dari data yang telah dikumpulkan menggunakan perangkat seperti PC ataupun menggunakan *server cloud*.

3. Optimasi dan Kompresi Model

Proses ini dilakukan setelah pelatihan model dimana model yang sudah dibangun akan di optimasi dan di kompres sehingga dapat di implementasi pada perangkat yang dituju.

4. *Deploy* dan Inferensi

Proses ini akan melakukan integrasi model ke perangkat keras seperti mikrokontroler agar perangkat tersebut dapat melakukan proses inferensi dan menjalankan model yang sebelumnya sudah dilatih di perangkat tersebut.

Dalam penerapannya, berdasarkan (Sanchez-Iborra & Skarmeta, 2020) TinyML dapat diaplikasikan pada berbagai bidang yang akan dipaparkan sebagai berikut.

1. *eHealth*

Merupakan bagian dengan potensi pertumbuhan yang sangat besar, contoh sederhananya seperti pemantauan kesehatan (*health monitoring*), bantuan penglihatan (*visual assistance*), bantuan pendengaran (*hearing aids*), penginderaan individu (*personal sensing*) dan pengenalan aktivitas (*activity recogniton*) dapat dijalankan dengan sistem TinyML.

2. *Smart Spaces*

Alat monitoring dan pengawasan yang berbasis IoT sekarang, dapat berevolusi menjadi suatu entitas yang cerdas dan otonom yang memiliki kemampuan untuk mengambil keputusan yang terdesentralisasi dan cepat. Contohnya seperti pengawasan lalu-lintas, polusi udara, ataupun sensor kerumunan (*crowdsensing*). Dengan implementasi yang simpel dan tidak tergantung pada jaringan listrik memungkinkan penempatan sistem ini pada daerah terpencil dan pedesaan sehingga membentuk ruang pintar (*smart spaces*) dengan jangkauan tanpa batas.

3. *Vehicular Services*

Implementasi sistem ini juga dapat dilakukan pada transportasi publik dengan tujuan menyediakan pelayanan kendaraan standar seperti pengamanan berkendara, monitoring status kendaraan, perencanaan rute perjalanan dan sebagainya. Namun transportasi pribadi belum dipertimbangkan dalam implementasi TinyML dikarenakan sistemnya yang terbaru dengan kebutuhan masing-masing individu yang sangat beragam.

4. *Industry 4.0*

Dampak dari TinyML sangat berpotensi pada bidang industri dan sektor manufaktur dimana bidang tersebut sedang mengalami proses digitalisasi dibawah naungan Revolusi Industri 4.0. Dengan pemrosesan yang sangat beragam dari segi komputasi dan kebutuhan memori, TinyML memungkinkan pemrosesan ini dipertimbangkan dan kemudian dilaksanakan di sistem yang ada ataupun diarahkan ke sistem *edge* ataupun *cloud*. Sedangkan untuk penyempurnaan proses produksi, pengambilan keputusan ataupun pelacakan aset akan dibantu oleh integrasi intelijen disepanjang rantai produksi.

5. *Smart Agriculture and Farming*

Dengan mengadopsi sistem kontrol dan monitoring berbasis *Machine Learning*, sektor ini akan meningkatkan efisiensi dan kualitas dari hasil panen tumbuhan dan hewan yang dimana akan meningkatkan pendapatan karena kualitas yang meningkat.

Terdapat juga tantangan dan batasan dalam implementasi TinyML, diantaranya adalah Keterbatasan *Hardware*, dimana TinyML ditujukan untuk implementasi pada sistem dengan sumber daya yang terbatas dari segi memori dan kapasitas pemrosesan sehingga menuntut model TinyML yang sangat efisien. Keterbatasan Daya, dimana sistem *edge* harus dapat mengelola konsumsi daya yang optimal sehingga memperpanjang masa pakai baterai. Dan yang terakhir yaitu Kompleksitas Pengembangan, dimana diperlukan pengetahuan tentang pembelajaran mesin dan pemrograman tertanam (*embedded*) yang dimana merupakan tantangan untuk pengembang sistem TinyML.

2.3 MFCC (*Mel-Frequency Cepstral Coefficients*)

Teknik yang paling umum digunakan dalam proses pengolahan suara dan pengenalan ucapan adalah dengan menggunakan MFCC atau yang biasa dikenal sebagai *Mel-Frequency Cepstral Coefficients*. Adapun proses yang dilakukan oleh MFCC ini berupa pengubahan sinyal suara menjadi serangkaian fitur yang menyerupai persepsi cara pendengaran manusia. Adapun Skala Mel yang menjadi konsep dasar dari MFCC dirumuskan pada persamaan (2.1) berikut

$$m = 2595 \log_{10} \left(1 + \frac{f}{700} \right)$$

Persamaan *Mel Scale* (2.1)

Adapun proses pembuatan MFCC dilakukan dengan cara *Pre-emphasis*, *Framing*, *Windowing*, *Fast Fourier Transform* (FFT), *Mel Filter Bank*, *Logarithm*, dan *Discrete Cosine Transform* (DCT). Penjelasan lebih lengkap dipaparkan sebagai berikut.

1. *Pre-emphasis*

Dimana sinyal suara akan melewati *filter high-pass* untuk memperkuat energi pada frekuensi tinggi, yang akan membantu dalam mengimbangi penurunan energi yang terjadi pada frekuensi tinggi selama perekaman suara. Adapun perhitungan dirumuskan pada persamaan (2.2)

$$y[t] = x[t] - \alpha x[t - 1]$$

Rumus *Pre-emphasis* (2.2)

Dimana $y[t]$ merupakan sinyal setelah *pre-emphasis*, $x[t]$ adalah sinyal asli dan α adalah koefisien *pre-emphasis* yang umumnya sebesar 0,97.

2. *Framing*

Sinyal akan dipotong menjadi *frame-frame* yang lebih pendek dengan tujuan memastikan bahwa sinyal tetap stasioner dalam *frame* tersebut. Adapun perhitungan dirumuskan pada persamaan (2.3)

$$x_i[n] = x[n + i \cdot (M - O)]$$

Rumus *Framing* (2.3)

Dimana n merupakan panjang sinyal suara dengan panjang *frame* adalah M dan overlap O . Perumusan ini dilakukan dengan kondisi $0 \leq n < M$.

3. *Windowing*

Setiap *frame* akan dikalikan dengan jendela (*Hamming Window*) dengan tujuan mengurangi diskontinuitas pada tepi *frame* yang dapat menyebabkan kebocoran spektral. Adapun perhitungan dirumuskan pada persamaan (2.4)

$$w[n] = 0.54 - 0.46 \cos\left(\frac{2\pi n}{M-1}\right)$$

Rumus *Hamming Window* (2.4)

Dimana $w[n]$ adalah jendela (*window*).

4. *Fast Fourier Transform* (FFT)

Pada tahap ini, FFT akan diterapkan untuk mengubah setiap *frame* dari domain waktu ke domain frekuensi yang dimana akan menghasilkan spektrum frekuensi.

5. *Mel Filter Bank*

Spektrum yang dihasilkan dari FFT akan dilewatkan melalui serangkaian *filter* segitiga yang terdistribusi merata pada skala Mel. Dimana *filter* ini akan menghitung energi dalam berbagai rentang frekuensi. Adapun perhitungan dirumuskan pada persamaan (2.5)

$$E_m = \sum_{k=f_{m-1}}^{f_{m+1}} |X[k]|^2 H_m[k]$$

Rumus *Mel Filter Bank* (2.5)

Dimana $H_m[k]$ merupakan respons *filter* Mel ke- m .

6. *Logarithm*

Energi dari setiap *filter* Mel akan diambil logaritmanya dengan tujuan menyesuaikan dengan persepsi manusia yang lebih sensitif terhadap perubahan relatif dalam intensitas suara daripada perubahan secara absolut. Dirumuskan pada persamaan (2.6)

$$\log(E_m)$$

Rumus *Logarithm* (2.6)

7. *Discrete Cosine Transform* (DCT)

DCT diterapkan pada hasil logaritma *filter bank* untuk menghasilkan koefisien cepstral dengan tujuan menghasilkan fitur yang tidak berkorelasi

dan data yang lebih ringan. Proses ini akan menghasilkan MFCC dengan perhitungan dirumuskan pada persamaan (2.7)

$$MFCC[n] = \sum_{m=1}^M \log(E_m) \cos\left[\frac{\pi n(m - 0.5)}{M}\right]$$

Rumus *Discrete Cosine Transform* (2.7)

Dalam penerapannya, (Abdul & Al-Talabani, 2022) mengemukakan bahwa MFCC dapat diterapkan pada Analisis Akustik berupa:

1. *Speech Analysis*, contohnya *Automatic Speech Recognition*, *Speech Emotion Recognition*, dan *Language and Dialect Recognition*
2. *Biometric Application*, contohnya *Speaker Recognition* dan *Gender Recognition Over the Phone Call*
3. *Digital Forensic*, contohnya *Sentiment Speech Analysis* dan *Fake Speech Detection*

Pada *Medical Applications* berupa:

1. *EEG Analysis*
2. *ECG Analysis*
3. *Disease Detection Application*, seperti *Parkinson Disease*, *Voice Pathology Disorder Detection* dan *Covid-19 Detection Via Cough* (Bansal et al., 2020).

Pada Analisis Industri berupa:

1. *Gear Health Monitoring*
2. *Bearing Health Monitoring*
3. *Turbine Health Monitoring*
4. *Pump Health Monitoring*

Adapun kelebihan dari MFCC adalah teknik ini lebih sesuai dengan persepsi pendengaran manusia, efektif digunakan dalam berbagai sistem pengolahan suara serta lebih mudah untuk diimplementasikan. Sedangkan untuk kekurangan dari MFCC ini adalah teknik ini sensitif terhadap *noise*, sensitif terhadap perbedaan intonasi dan aksentuasi dalam pengenalan ucapan (*Keyword Spotting*) serta kurang optimal digunakan dalam kasus pengenalan suara tanpa kata (*non-speech*).

2.4 Keyword Spotting

Keyword Spotting dapat diartikan sebagai tugas untuk identifikasi berbagai kata kunci dari suara yang mengandung berbagai kata (Momeni et al., 2020). Konsep ini diimplementasi pada fitur untuk aktivasi aplikasi sistem asisten suara (Voice Assistants) seperti Cortana dari Microsoft maupun Google Assistant, dimana aplikasi sistem ini akan aktif apabila mendengarkan kata kunci tertentu. Sehingga *Keyword Spotting* ini menggunakan daya komputasi yang lebih sedikit dibandingkan *Automatic Speech Recognition (ASR)*. Selain sebagai alat untuk aktivasi sistem asisten suara, *Keyword Spotting* juga diaplikasikan pada berbagai hal seperti *mining* data suara, *indexing* berbagai data audio dan rute panggilan telepon (Lopez-Espejo et al., 2022).

Adapun fungsi dan implementasi *Keyword Spotting* akan dipaparkan sebagai berikut.

1. Perintah Suara (*Voice Commands*)
 - Asisten Virtual, contoh umumnya adalah Siri, Google Assistant dan Alexa dimana untuk mengenali perintah suara yang spesifik yang dimulai dengan kata kunci seperti “Hey Siri” atau “OK Google”.
 - Kontrol Perangkat, dimana digunakan untuk mengontrol perangkat pintar dengan perintah suara, seperti menghidupkan lampu, memainkan musik ataupun mengatur termostat.
2. Keamanan dan Otentikasi
 - Pengenalan Kata Sandi Suara, dimana digunakan kata kunci sebagai bagian dari sistem otentikasi untuk mengidentifikasi pengguna.
 - Monitoring Panggilan, dimana digunakan untuk mendeteksi kata-kata sensitif atau ancaman dalam panggilan telepon. Biasanya diimplementasikan pada aplikasi keamanan atau layanan pelanggan.
3. Transkripsi dan Analisis
 - Analisis Data Audio, digunakan dalam analisis besar data audio untuk mengekstraksi informasi penting berdasarkan kata kunci yang muncul dalam percakapan.
 - Penandaan Konten, dimana digunakan untuk menandai bagian-bagian tertentu dari audio atau video berdasarkan kata kunci yang muncul.

Adapun cara kerja dari *Keyword Spotting* dipaparkan sebagai berikut.

1. Preprocessing Audio

- Noise Reduction, mengurangi kebisingan suara latar belakang untuk meningkatkan kejelasan audio.
- Feature Extraction, mengekstraksi fitur audio seperti Mel-Frequency Cepstral Coefficients (MFCC) yang digunakan untuk mewakili karakteristik akustik.

2. Modelling

- Model Akustik, model yang dilatih untuk mengenali pola akustik dari kata-kata kunci spesifik.
- Neural Networks, digunakan untuk meningkatkan akurasi deteksi yang dimana merupakan *deep learning*. Contohnya Convolutional Neural Networks (CNN) atau Recurrent Neural Networks (RNN).

3. Keyword Detection

- Sliding Window, menggunakan pendekatan jendela geser untuk menganalisis potongan-potongan kecil dari sinyal audio dan mendeteksi keberadaan kata kunci.
- Thresholding, menentukan ambang batas probabilitas untuk memutuskan apakah kata kunci telah terdeteksi.

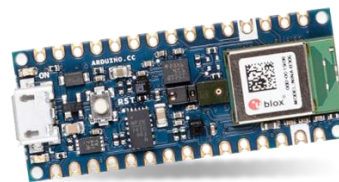
4. Post-processing

- Validation, memastikan bahwa deteksi kata kunci bukan hasil dari kebisingan atau kesalahan sistem.
- Action Triggering, jika kata kunci telah terdeteksi, sistem dapat memicu tindakan yang telah diprogram, seperti memulai sesi interaksi dengan pengguna atau mengirim notifikasi.

Terdapat juga tantangan dalam implementasi Keyword Spotting, yaitu sistem KWS harus dapat berfungsi dengan baik dalam berbagai lingkungan dengan tingkat kebisingan yang berbeda. Penggunaan energi juga harus efisien, terutama untuk implementasi pada perangkat portabel atau yang selalu aktif. Sistem juga harus mampu mengenali kata kunci yang diucapkan dengan berbagai aksen, intonasi dan kecepatan bicara. Serta sistem KWS harus dapat menjaga keamanan dan privasi pengguna dengan tidak menyimpan atau menyebarkan suara yang didengar.

2.5 Arduino Nano 33 BLE Sense Lite

Adalah suatu mikrokontroler yang dirakit sebagai turunan dari Arduino Nano 33 BLE Sense. Dibangun dengan mikrokontroller nRF52840 dan Sistem Arm® Mbed™, Arduino ini juga memiliki berbagai sensor seperti LSM9DS1 untuk mendeteksi pergerakan, MP34DT05 sebagai Mikrofon, APDS9960 untuk deteksi bentuk Gerakan (*gesture*) tertentu. Namun untuk versi Sense Lite tidak terdapat sensor untuk suhu, kelembaban dan tekanan (V et al., 2022). Mikrokontroler ini memiliki konektivitas Bluetooth dengan daya rendah (BLE) dan komunikasi bidang pendek atau dikenal sebagai NFC dengan Memori sebesar 256 KB RAM dan 1 MB Flash. Terdapat 14 pin untuk input dan output digital, 8 pin untuk input analog, PWM, I2C, SPI, UART dan GPIO. Daya yang dibutuhkan untuk mengoperasikan mikrokontroler ini adalah sebesar 3.3V, 5V (melalui VIN pin) dan juga dapat diberi daya melalui konektor micro-USB.



Gambar 2.2. Arduino Nano 33 BLE Sense

2.6 ESP32

ESP32 merupakan mikrokontroller SoC (*System on Chip*) yang memiliki modul Wi-Fi dan *Bluetooth* yang memungkinkan komunikasi ke mikrokontroller secara *wireless*. ESP32 diproduksi dalam bentuk prototipe yang dapat digunakan dalam berbagai alat rumah tangga pintar (*smart home applications*), dalam otomasi sistem, aplikasi IoT berbasis *cloud* dan sebagainya (Babiuch et al., 2019). ESP32 merupakan penerus dari ESP8266 dengan performa lebih tinggi yang dan fitur yang lebih lengkap. Modul ini banyak digunakan dalam berbagai proyek Internet of Things (IoT), *Smart Home* dan lainnya.

ESP32 tersusun atas prosesor dual-core Xtensa LX6 yang memiliki performa hingga 240 MHz, RAM dengan kapasitas 520 KB, dan Flash memory mencapai 4

MB. Dengan koneksi Wi-Fi yang memiliki mode AP, STA dan dual mode serta Bluetooth 4.2 maupun Bluetooth Low Energy (BLE). Terdapat 25 GPIO pins yang dapat dikonfigurasi untuk berbagai keperluan seperti digital I/O, PWM, ADC, DAC dan sebagainya. Antarmuka komunikasi seperti UART dan PWM hardware untuk kontrol motor dan LED. Terdapat juga Ultra Low Power (ULP) co-processor untuk hemat energi. Secure boot dan flash encryption untuk pengamanan data, pin header yang kompatibel dengan breadboard, Port Micro USB, tombol reset dan tombol boot untuk memudahkan pengoperasian. ESP32 juga dapat menerima program yang tersusun dari berbagai bahasa pemrograman dan *environment* seperti Arduino IDE dan Espressif IDF serta Micro Python.



Gambar 2.3. ESP32

2.7 Driver Motor L298N

Merupakan suatu modul yang sering digunakan untuk kendali motor DC dan motor stepper. Modul ini terdiri atas IC L298N yang dimana merupakan jembatan H ganda (dual H-bridge) yang dapat mengendalikan dua motor DC atau satu motor stepper dengan ketepatan dan kontrol yang optimal. Adapun voltase yang digunakan untuk kendali motor berkisar antara 5V hingga 35V, dapat mengendalikan arus hingga 2A, dilengkapi dengan pendingin untuk mencegah kerusakan apabila terjadi *overheating* dan terdapat dioda yang digunakan untuk melindungi IC dari arus balik yang dihasilkan oleh motor.

Modul ini memiliki 4 input pins (IN1, IN2, IN3, IN4) yang digunakan untuk kontrol arah putaran motor, 2 enable pins (ENA, ENB) yang digunakan untuk

pengendalian aktivasi H-bridge serta dapat dihubungkan ke PWM untuk mengendalikan kecepatan motor, terdapat 4 output pins (OUT1, OUT2, OUT3, OUT4) yang terhubung ke terminal motor, Power Supply pins untuk memenuhi tegangan motor dan Ground untuk menyelesaikan rangkaian listrik.



Gambar 2.4. Driver Motor L298N

2.8 TensorFlow

TensorFlow merupakan library yang fleksibel dimana dapat digunakan untuk mengembangkan program dan melatih *neural network* serta berbagai model pembelajaran mesin lainnya. Algoritma inti yang digunakan pada TensorFlow tersusun atas bahasa C++ dan CUDA (Compute Unified Device Architecture) yang sangat dioptimalkan, platform computing yang parallel dan API yang dibuat oleh NVIDIA (Pang et al., 2020). *Library* ini dikembangkan oleh Google Brain Team yang berbasis *open-source* sehingga dapat digunakan dimodifikasi maupun didistribusikan oleh siapa saja. *Library* ini juga mendukung berbagai jenis model *machine learning*, *neural networks*, *decision trees* dan *linear models*, juga dapat digunakan untuk riset dan produksi baik skala kecil (seperti pada *edge devices*) maupun skala besar seperti menggunakan *server* dan *cloud*.

Adapun arsitektur dan komponen utama dari TensorFlow ini berupa Tensors, Graphs, Sessions, dan Eager Execution. Tensor merupakan representasi data dalam bentuk array multidimensi dimana Tensors adalah unit dasar dari data dalam TensorFlow. Tensor ini memungkinkan berbagai operasi matematis seperti matriks multiplikasi, penjumlahan, pengurangan dan sebagainya. Graphs atau computational graphs digunakan untuk mendefinisikan aliran data dan operasi yang akan dilakukan. Dengan node dalam graph yang berfungsi mewakili operasi matematika, sedangkan

edges mewakili tensors. Graphs memungkinkan eksekusi yang efisien dan optimisasi serta memfasilitasi distribusi komputasi di berbagai perangkat keras (CPU, GPU, TPU). Sessions digunakan sebelum TensorFlow 2.0 untuk menjalankan Graph. Sessions menyediakan konteks untuk menjalankan operasi dan mengevaluasi tensors. Setelah TensorFlow 2.0, digunakan Eager Execution sebagai pengganti Sessions. Eager Executions memungkinkan untuk eksekusi langsung dari operasi tensor tanpa membangun computational graph yang dimana membuat debugging dan pengembangan lebih mudah dan intuitif (Singh & Manure, 2020).

Terdapat Ekosistem TensorFlow yaitu TensorFlow Core, Keras, TensorFlow Extended (TFX), TensorFlow Lite, TensorFlow.js dan TensorFlow Hub. Berikut penjelasan secara detail.

1. TensorFlow Core

Merupakan pustaka inti yang dimana menyediakan API dasar untuk membangun dan melatih model pembelajaran mesin.

2. Keras

Merupakan API tingkat tinggi yang dibangun di TensorFlow. Memiliki sintaks yang lebih sederhana untuk membangun dan melatih model neural network dan juga mendukung prototyping secara cepat.

3. TensorFlow Extended (TFX)

Menyediakan komponen dan alat untuk membangun dan mengelola pipeline pembelajaran mesin produksi. Juga memiliki komponen seperti TensorFlow Data Validation, TensorFlow Transform, TensorFlow Model Analysis, dan TensorFlow Serving.

4. TensorFlow Lite

Merupakan versi TensorFlow yang dioptimalkan untuk perangkat mobile dan tertanam (embedded). Dimana memungkinkan implementasi model pembelajaran mesin pada perangkat dengan sumber daya terbatas.

5. TensorFlow.js

Menyediakan API untuk menjalankan dan melatih model TensorFlow di dalam browser atau pada Node.js.

6. TensorFlow Hub

Merupakan pustaka yang menyediakan model siap pakai untuk berbagai tugas pembelajaran mesin, yang dapat diintegrasikan ke dalam aplikasi dengan mudah.

Adapun aplikasi dan penggunaan dari TensorFlow sering digunakan pada Computer Vision, seperti pengenalan objek, deteksi wajah, segmentasi gambar dan lainnya. Pada Natural Language Processing, seperti pemrosesan bahasa alami, terjemahan mesin, analisis sentimen dan lainnya. Pada Speech Recognition dengan tujuan pengenalan suara ataupun konversi ucapan ke teks. Dan pada Reinforcement Learning yang ditujukan untuk pengembangan sistem yang dapat belajar dari interaksi dengan lingkungan. Keunggulan dari TensorFlow ini dari skala skalabilitas dimana TensorFlow mendukung eksekusi distribusi pada beberapa perangkat keras, seperti CPU, GPU dan TPU. Dengan sifatnya yang berupa proyek open-source, TensorFlow dikembangkan oleh komunitas dan juga dukungan aktif dari Google. TensorFlow juga menawarkan beragam alat dan pustaka untuk berbagai kebutuhan pembelajaran mesin.

Tantangan dari menggunakan TensorFlow adalah butuh kemampuan atau pemahaman yang mendalam untuk seorang pemula dalam pembelajaran mesin. Dan dalam optimasi model dan komputasi dapat memiliki tingkat kesulitan yang kompleks sehingga diperlukan pemahaman tentang perangkat keras dan sistem yang akan diimplementasi TensorFlow.

2.9 Penelitian Terdahulu

Telah dilakukan berbagai penelitian yang terkait dengan Keyword Spotting, dan implementasi *machine learning* pada mikrokontroller. Diantaranya penelitian dari (V et al., 2022) tentang implementasi Gesture dan Speech Recognition dengan model yang dibangun berdasarkan TinyML yang kemudian model tersebut akan di-*deploy* pada Arduino Nano 33 BLE Sense dengan bantuan Edge Impulse. Pada Gesture Recognition, dilakukan uji coba deteksi pergerakan tangan manusia dengan *data testing* yang dibuat sebelumnya lalu dilakukan identifikasi berdasarkan pada model yang telah dibuat dengan akurasi yang mencapai 100%. Pada Speech Recognition, dilakukan uji coba deteksi *keyword* dengan *data testing* yang telah dibuat sebelumnya. Lampu RGB pada Arduino Nano 33 BLE Sense akan hidup apabila *keyword* yang

dideteksi oleh mikrofon Arduino sesuai dengan *keyword* yang telah dilatih sebelumnya dimana tercapai akurasi sebesar 98,75%.

Berikutnya oleh (Gouda et al., 2018), dimana penulis menggunakan berbagai model CNN pada data gambar yang dibentuk dengan cara mengubah klip suara menjadi spectrograms. Tercapai akurasi validasi sebesar 92% pada 20% sampel random dari data input setelah implementasi metode regularisasi "Virtual Adversarial Training".

Pada penelitian yang telah dilakukan oleh (Chaudhry et al., 2019), penulis merakit suatu sistem robot yang dimana dikontrol menggunakan suara berbasis Arduino. Peralatan yang digunakan berupa Arduino ATmega dan ponsel Android sebagai mikrofon. Sistem robot ini memiliki proses dimana kontroller dihubungkan dengan module Bluetooth melalui protokol UART, kemudian suara yang diterima oleh kontroller akan diproses oleh dan dikonversikan ke teks, lalu teks tersebut oleh diproses oleh mikrokontroller dan hasilnya akan menentukan keputusan yang akan diambil untuk arah gerak robot.

Penelitian (Sakr et al., 2020) menyajikan sebuah framework pembelajaran mesin yang bernama Edge Learning Machine (ELM) dimana dapat mengelola tahap *training* pada komputer dan melaksanakan proses *inferences* pada mikrokontroller. Framework tersebut mengimplementasikan tiga algoritma pembelajaran mesin yaitu *Support Vector Machine* (SVM), algoritma k-NN (*k-Nearest Neighbors*) dan DT (*Decision Tree*) dan menggunakan STM X-Cube-AI untuk implementasi ANNs atau dikenal sebagai *Artificial Neural Networks* pada STM32 Nucleo. Didapati hasil dimana ANN memiliki performa lebih baik dibandingkan algoritma lain pada banyak kasus, tanpa memandang perbedaan *device*. Peningkatan *depth* dari suatu *Neural Network* dapat juga meningkatkan performa pada level saturasi. k-NN berfungsi hampir sama dengan ANN dan pada kasus tertentu dapat memiliki performa lebih baik namun membutuhkan semua *training set* untuk disimpan pada tahap *inference* sehingga akan membutuhkan memory yang lebih signifikan. Sedangkan performa DT sangat bervariasi tergantung pada *dataset* yang digunakan.

Pada penelitian (Ding et al., 2018), dimana menggunakan metode gabungan audio dan visual untuk deteksi kata kunci (AV-KWS) berdasarkan dimensi multi suatu

jaringan neural konvolusi (MCNN) untuk menyelesaikan permasalahan KWS, yang berupa kesulitan mendeteksi kata tertentu pada kondisi lingkungan yang ribut. Dimulai dengan ekstraksi *log mel-spectrogram* dan area bibir baik dari aliran suara dan gambar, lalu dipakai sebagai *data input* pada network neural suatu *audioVisual*. Kemudian *audioVisual neural network* yang berbasis MCNN, memiliki 2D CNN serta 3D CNN digunakan untuk membuat model fitur waktu-frekuensi dari *log mel-spectrogram* dan fitur *spatiotemporal* dari area bibir. Hasil dari *audio* dan *visual networks* akan digabung untuk KWS berdasarkan *decision fusion*. Diperoleh hasil berdasarkan pada *database* audio-video yang kompleks dimana metode ini mencapai performa yang lebih baik dibandingkan metode umum lainnya.

Tabel 2.1. Penelitian Terdahulu

No.	Judul Penelitian	Peneliti	Tahun	Keterangan
1.	Implementation Of Tiny Machine Learning Models on Arduino 33 BLE For Gesture And Speech Recognition	V et al.	2022	Menyajikan implementasi TinyML pada Arduino Nano 33 BLE Sense tentang Gesture dan Speech Recognition. Pada proses Gesture Recognition dilakukan uji coba dengan data testing dan hasil akurasi yang dicapai adalah 100%. Pada Speech Recognition dilakukan uji coba dengan data testing dan hasil akurasi yang dicapai adalah 98,75%.
2.	Speech Recognition: Key Word Spotting through Image Recognition	Gouda et al.	2018	Menyajikan metode regularisasi "Virtual Adversarial Training" untuk meningkatkan efektivitas dalam penggunaan metode CNN pada kasus Speech Recognition dimana tercapai akurasi validasi sebesar 92% pada 20% sampel random dari data input.

3. Arduino Based Chaudhry 2019
Voice Controlled et al.
Robot
Menyajikan sistem robot berbasis Arduino ATmega dengan kontroller suara dimana ponsel Android berfungsi sebagai mikrofon. Dimana suara yang diterima oleh mikrofon akan diproses oleh module suara lalu dikonversi ke teks yang kemudian teks tersebut akan diproses oleh mikrokontroller untuk menentukan arah gerak robot.
4. Machine Sakr et al. 2020
Learning on
Mainstream
Microcontrollers
Menyajikan Framework ML, Edge Learning Machine (ELM) yang mengimplementasikan SVM, k-NN dan Decision Tree serta menggunakan STM X-Cube-AI untuk implementasi Artificial Neural Networks (ANNs). Dengan hasil berupa ANN memiliki performa yang lebih baik tanpa memandang device yang digunakan, sedangkan k-NN membutuhkan memory yang lebih banyak agar dapat memiliki performa serupa dengan ANN. Untuk performa DT bervariasi tergantung pada dataset yang digunakan.

5.	Audio-Visual Keyword Spotting Based on Multidimensional Convolutional Neural Network	Ding et 2018 al.	Menyajikan metode Audio-Visual Key Word Spotting (AV-KWS) berbasis Multidimensional Convolutional Neural Network (MCNN) untuk menyelesaikan permasalahan KWS yang sulit mendeteksi keyword pada kondisi lingkungan yang ribut. Diperoleh performa yang lebih baik dibandingkan metode umum lainnya pada kasus data audio-video yang kompleks.
----	--	---------------------	--

Sedangkan perbedaan penelitian yang dilakukan penulis dengan penelitian-penelitian terdahulu disajikan pada tabel 2.2 berikut.

Tabel 2.2. Penelitian Penulis

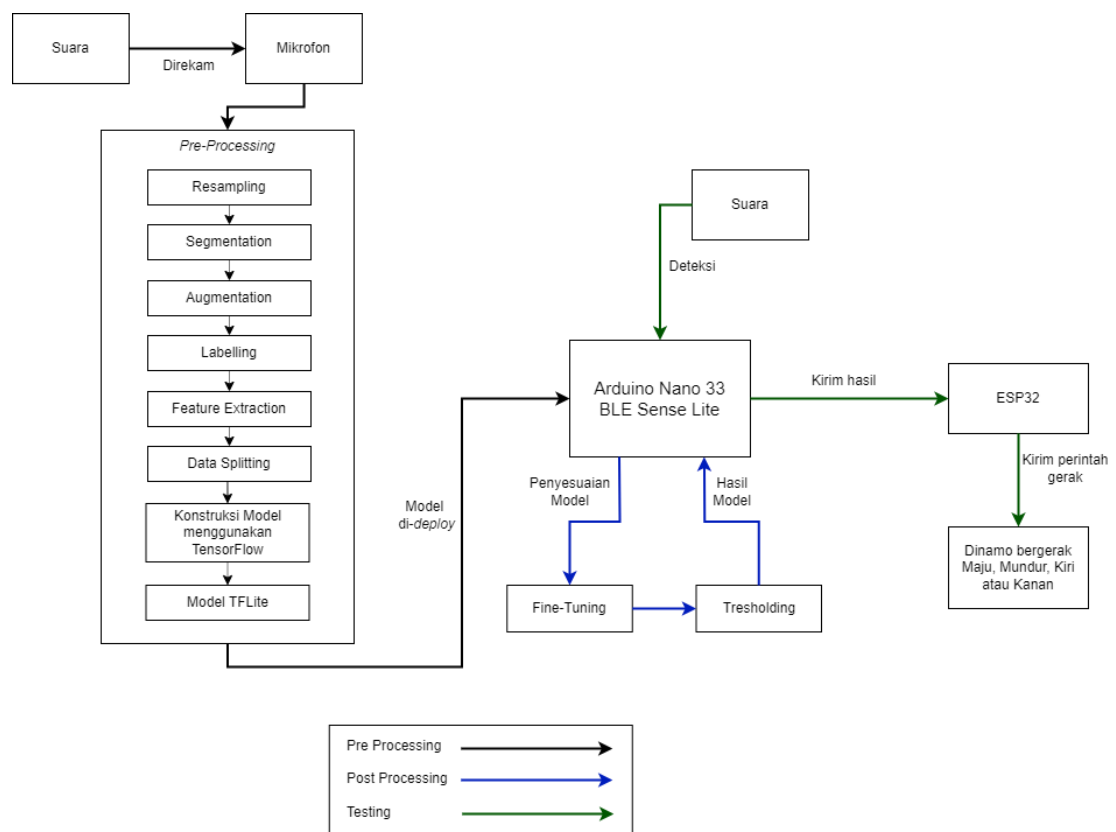
No.	Judul Penelitian	Peneliti	Tahun	Keterangan
1.	Implementasi Keyword Spotting untuk Menggerakkan Robot Strandbeest Berteknologi TinyML	Alvin Daeli	2024	Menyajikan implementasi Keyword Spotting pada Robot Strandbeest dengan menggunakan Arduino Nano 33 BLE Sense Lite sebagai mikrofon dan perangkat tertanam yang menjalankan model TinyML serta ESP32 sebagai mesin untuk mengendalikan Robot Strandbeest. Didapati hasil pengujian model dengan data training sebanyak 3115 dan data testing sebanyak 773, akurasi tercapai 99,34%. Akurasi pada uji coba real-time adalah 73%.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Arsitektur Umum

Penulis akan melakukan proses deteksi kata kunci menggunakan Arduino Nano 33 BLE Sense Lite, lalu suara yang telah terdeteksi akan diklasifikasi sesuai model yang telah dibuat. Hasil klasifikasi akan dikirim ke ESP32 melalui komunikasi UART dan akan diproses untuk menentukan gerakan dinamo yang terpasang pada robot.



Gambar 3.1 Arsitektur Umum

Pada Gambar 3.1, terdapat arsitektur umum dari sistem yang akan dibangun dengan penjelasan sebagai berikut.

3.1.1 Pre-Processing

Dimana akan dilakukan proses perekaman suara sebagai data yang akan digunakan pada *training*. Suara yang direkam mengandung 4 kata kunci yaitu, “maju”, “mundur”, “kiri”, dan “kanan”. Dilakukan juga beberapa perekaman suara *random* untuk menghasilkan data *noise* yang akan juga di-*training* dengan tujuan membuat model suara lebih kokoh dan dapat membedakan kata kunci yang telah dilatih dengan suara lingkungan yang bersifat *random*. Adapun proses yang akan dilakukan berupa:

1. Resampling

Dimana akan dilakukan modifikasi *sample rate* audio sehingga tercapai nilai yang konstan dan seragam pada semua data audio yaitu sebesar 16 kHz. Adapun pseudocode dipaparkan pada Gambar 3.2 berikut

```

Algorithm ResampleAudio
  Input: audio_path, target_sample_rate (default = 16000)
  Output: resampled_audio, target_sample_rate

  # Load audio file
  audio, sr = librosa.load(audio_path, sr=None)

  # Check if resampling is needed
  If sr is not equal to target_sample_rate then
    # Resample audio to target sample rate
    audio = librosa.resample(audio, orig_sr=sr,
                             target_sr=target_sample_rate)
  EndIf

  # Return the resampled audio and the target sample rate
  Return audio, target_sample_rate
End Algorithm

```

Gambar 3.2 *Pseudocode Resampling Audio*

Pada Gambar 3.2, dijelaskan proses *resampling* audio adalah dengan menggunakan library *librosa* dimana *sample rate* audio diperiksa terlebih dahulu dan apabila bernilai tidak sama dengan 16000 Hz, maka *librosa* akan *resample* audio tersebut menjadi 16000 Hz dimana *sample rate* tersebut sesuai dengan *sample rate* dari audio yang dapat diterima oleh mikrofon Arduino Nano 33 BLE

Sense Lite. Proses *resampling* ini dilakukan sehingga menyelaraskan kondisi percobaan model dengan *data testing* serta percobaan model secara *real-time*.

2. Segmentation

Proses dimana data audio akan dibagi menjadi segmen-segmen yang lebih kecil atau selama 1 detik dimana audio 1 detik tersebut dipastikan mengandung hanya 1 kata kunci. Dengan *pseudocode* diberikan pada Gambar 3.3

```

Algorithm SegmentAudio
  Input: audio, segment_length (default = 16000)
  Output: segments
  Initialize empty list segments
  For start from 0 to length of audio with step segment_length
  do
    end = start + segment_length
    If end is less than or equal to length of audio then
      Append audio[start:end] to segments
    EndIf
  EndFor
  Return segments
End Algorithm

```

Gambar 3.3 *Pseudocode Segmentation Audio*

Pada Gambar 3.3 dijelaskan proses *segmentation audio*. Diberikan panjang segmen default sebesar 16000 karena suara yang akan dibagi memiliki *sample rate* sebesar 16000 yang dimana panjang segmen merupakan laju pengambilan sampel dari data audio. Audio yang memiliki panjang segmen 16000 dengan *sample rate* sebesar 16000 Hz memiliki durasi sebesar 1 detik. Proses yang dilakukan berupa inisialisasi daftar kosong bernama *segments* untuk menyimpan potongan-potongan audio. Kemudian dilakukan iterasi dari awal data audio sampai akhir data dengan langkah sebesar panjang segmen yang ditentukan. Pada setiap langkah perulangan, dihitung indeks akhir dari segmen saat ini sebagai nilai awal ditambah panjang segmen. Jika indeks akhir masih dalam batas panjang audio maka akan diambil potongan audio dari indeks awal hingga indeks akhir dan ditambahkan ke daftar *segments*. Proses tersebut dilakukan hingga seluruh data audio diproses. Dan terakhir, daftar *segments* yang berisi potongan-potongan

audio akan menjadi *output* dari program dimana berisi potongan-potongan audio yang berdurasi selama 1 detik.

3. Augmentation

Merupakan proses yang dilakukan untuk membuat dataset lebih kokoh yaitu dengan implementasi beberapa transformasi pada audio seperti penambahan *noise* pada dataset. Diberikan *pseudocode* pada Gambar 3.4 dan Gambar 3.5. Algoritma *AugmentAudio* dirancang untuk modifikasi data audio dengan cara menambahkan kebisingan (*noise*) dan mengubah nada (*pitch*).

```

Algorithm AugmentAudio
  Input: audio, sample_rate, noise_factor (default = 0.005),
  pitch_factor (default = 0.7)
  Output: augmented_audio, pitch_shifted_audio
  # Step 1: Add Noise to Audio
  Procedure AddNoise
    # Generate random noise
    noise = random values of length audio
    # Add noise to audio
    augmented_audio = audio + noise_factor * noise
  Return augmented_audio
End Procedure

```

Gambar 3.4 *Pseudocode AugmentAudio* penambahan *noise*

Pada Gambar 3.4, diberikan *pseudocode* untuk menambahkan kebisingan pada audio. Algoritma akan menghasilkan data *noise* secara acak dari kumpulan data *noise* yang telah diberikan lalu data *noise* acak tersebut akan ditambah ke audio asli dengan tingkat suara sebesar *noise_factor*. Hasil berupa data audio yang telah ditambahkan *noise* yang terletak pada *augmented_audio*.

```

# Step 2: Change Pitch of Audio
Procedure ChangePitch
    # Change the pitch of the audio
    pitch_shifted_audio = pitch_shift audio by pitch_factor
    with sample_rate
    Return pitch_shifted_audio
End Procedure

# Perform the augmentations
augmented_audio = AddNoise(audio, noise_factor)

pitch_shifted_audio = ChangePitch(audio, sample_rate,
pitch_factor)

Return augmented_audio, pitch_shifted_audio
End Algorithm

```

Gambar 3.5 *Pseudocode AugmentAudio* perubahan *pitch*

Pada Gambar 3.5 diberikan *pseudocode* untuk mengubah nada audio. Audio asli diubah berdasarkan *pitch_factor* dengan *sample-rate*. Hasil dari proses ini berupa data audio yang memiliki tingkat nada yang berbeda dari data sebelumnya.

4. Labelling

Merupakan tahap dimana dilakukan penamaan data (pemberian label) pada data audio yang telah diproses sebelumnya dengan tujuan dapat identifikasi hasil dari *testing* yang dilakukan. Contohnya apabila data audio berisi kata “maju”, akan diberi label “maju”.

5. Feature Extraction

Bagian dimana audio akan di-transformasi menjadi suatu format yang dapat digunakan dalam training model dan inference pada pembelajaran mesin. Pada bagian ini akan digunakan teknik MFCC dengan *pseudocode* dipaparkan pada sebagai berikut.

```

Algorithm ExtractMFCC
    Input: audio, sample_rate
    Output: mfccs
    Constants:
        n_mfcc = 13

```

Gambar 3.6 *Pseudocode ExtractMFCC*

```

    frame_length = 0.02 # in seconds
    frame_stride = 0.02 # in seconds
    n_mels = 32
    n_fft = 256
    normalization_window_size = 101
    low_freq = 0
    high_freq = 8000
    pre_emphasis_coeff = 0.98
# Step 1: Pre-emphasis
Procedure PreEmphasis
    emphasized_audio = array of same length as audio
    emphasized_audio[0] = audio[0]
    For i from 1 to length of audio - 1 do
        emphasized_audio[i] = audio[i] - pre_emphasis_coeff *
audio[i - 1]
    EndFor
    Return emphasized_audio
End Procedure

```

Gambar 3.7 *Pseudocode ExtractMFCC Tahap Pre-emphasis*

Pada Gambar 3.7 dijelaskan proses inisialisasi algoritma dan proses *pre-emphasis* dengan tujuan memperkuat frekuensi tinggi dalam audio untuk mengimbangi penurunan energi pada frekuensi tersebut. Dibentuk array *emphasized_audio* dengan panjang yang sama dengan data audio asli dimana nilai pertama array tersebut sama dengan nilai pertama audio asli, sedangkan nilai-nilai berikutnya dihitung dengan cara mengurangi audio saat ini dengan hasil kali koefisien *pre-emphasis* dan audio sebelumnya.

```

# Step 2: Framing

Procedure FrameAudio(emphasized_audio, sample_rate,
frame_length, frame_stride)

    frame_length_samples = frame_length * sample_rate
    frame_stride_samples = frame_stride * sample_rate

    num_frames = floor((length of emphasized_audio -
frame_length_samples) / frame_stride_samples) + 1

    frames = array of size (num_frames, frame_length_samples)

    For i from 0 to num_frames - 1 do
        start_idx = i * frame_stride_samples
        end_idx = start_idx + frame_length_samples
        frames[i] = emphasized_audio[start_idx:end_idx]
    EndFor

    Return frames

End Procedure

```

Gambar 3.8 Pseudocode ExtractMFCC Tahap Framing

Pada Gambar 3.8 dijelaskan tahap *framing* dimana audio yang telah diberi *pre-emphasis* akan dibagi menjadi segmen-segmen kecil dengan panjang tertentu. Panjang frame ditentukan oleh *frame_length* dan jarak antar frame ditentukan oleh *frame_stride*. Dihitung jumlah frame yang diperlukan untuk kemudian di isi ke dalam array *frames* dengan potongan-potongan audio sesuai dengan indeks yang telah dihitung.

```

# Step 3: Windowing

Procedure ApplyWindow(frames, normalization_window_size)

    hamming_window =
create_hamming_window(normalization_window_size)

    windowed_frames = frames * hamming_window

    Return windowed_frames

End Procedure

```

Gambar 3.9 Pseudocode ExtractMFCC Tahap Windowing

Pada Gambar 3.9 dijelaskan tahap *windowing* dimana *frame* yang telah dihasilkan pada tahap sebelumnya akan dikalikan dengan *Hamming Window* untuk mengurangi diskontinuitas di tepi frame. Algoritma membuat *hamming window*

dengan ukuran *normalization_window_size* dan dikalikan setiap *frame* dengan *window* tersebut untuk menghasilkan *windowed_frames*.

```
# Step 4: FFT and Power Spectrum

Procedure ComputePowerSpectrum(windowed_frames, n_fft)

    power_spectrum = |FFT(windowed_frames, n_fft)|^2 / n_fft

    Return power_spectrum

End Procedure
```

Gambar 3.10 *Pseudocode ExtractMFCC Tahap FFT dan Power Spectrum*

Pada Gambar 3.10 dijelaskan tahap perhitungan *Fast Fourier Transform* (FFT) dari setiap *frame* dengan *window* untuk mendapatkan spektrum frekuensi. *Power Spectrum* dihitung dengan cara mengkuadratkan magnitudo hasil FFT dan membaginya dengan *n_fft*.

```
# Step 5: Mel Filterbank

Procedure MelFilterbank(power_spectrum, n_mels, sample_rate,
low_freq, high_freq)

    mel_filterbank = create_mel_filterbank(n_mels, n_fft,
sample_rate, low_freq, high_freq)

    mel_spectrum = dot_product(power_spectrum, mel_filterbank)

    Return mel_spectrum

End Procedure
```

Gambar 3.11 *Pseudocode ExtractMFCC Tahap Mel Filterbank*

Pada Gambar 3.11 dijelaskan tahap *Mel Filterbank* dengan cara menerapkan *mel-filterbank* pada *power spectrum* untuk mendapatkan representasi spektral dalam skala Mel. *Mel Filterbank* ini dibuat berdasarkan jumlah filter *n_mels*, laju pengambilan sampel dan batas frekuensi rendah dan tinggi yang ditentukan.

```
# Step 6: Log Mel Spectrum

Procedure LogMelSpectrum(mel_spectrum)

    log_mel_spectrum = log(mel_spectrum + epsilon)

    Return log_mel_spectrum

End Procedure
```

Gambar 3.12 *Pseudocode ExtractMFCC Tahap Log Mel Spectrum*

Pada Gambar 3.12 dijelaskan tahap *Log Mel Spectrum* dengan cara mengambil logaritma dari spektrum Mel untuk menghasilkan *log_mel_spectrum* yang lebih sesuai untuk representasi karakteristik suara manusia.

```
# Step 7: Discrete Cosine Transform (DCT)

Procedure ComputeMFCC(log_mel_spectrum, n_mfcc)

    mfccs = DCT(log_mel_spectrum, type=2)[: , :n_mfcc]

    Return mfccs

End Procedure
```

Gambar 3.13 *Pseudocode ExtractMFCC Tahap Discrete Cosine Transform*

Pada Gambar 3.13 dijelaskan tahap perhitungan *Discrete Cosine Transform* (DCT) dimana DCT diterapkan pada *log_mel_spectrum* untuk mengurangi korelasi antara fitur-fitur dan menghasilkan MFCC.

```
# Perform the extraction

emphasized_audio = PreEmphasis(audio, pre_emphasis_coeff)

frames = FrameAudio(emphasized_audio, sample_rate,
frame_length, frame_stride)

windowed_frames = ApplyWindow(frames,
normalization_window_size)

power_spectrum = ComputePowerSpectrum(windowed_frames, n_fft)

mel_spectrum = MelFilterbank(power_spectrum, n_mels,
sample_rate, low_freq, high_freq)

log_mel_spectrum = LogMelSpectrum(mel_spectrum)

mfccs = ComputeMFCC(log_mel_spectrum, n_mfcc)

Return mfccs

End Algorithm
```

Gambar 3.14 *Pseudocode ExtractMFCC (lanjutan)*

Pada Gambar 3.14 dijelaskan proses *ExtractMFCC* yang dimulai dari *pre-emphasis* audio hingga membentuk *mel frequency cepstral coefficients* (mfccs) yang dimana dapat digunakan untuk berbagai aplikasi pemrosesan suara, seperti pengenalan ucapan (*Keyword Spotting*) atau analisis audio.

6. Data Splitting

Proses dimana hasil dataset dari proses sebelumnya akan dibagi menjadi dua bagian yaitu *data training* dan *data testing*. Dengan tujuan dapat menilai

kemampuan model sebelum diuji coba secara *real-time*. Adapun pembagian diterapkan *data* untuk *training* 80% dan *data* untuk *testing* 20%. Dengan total data sebanyak 3888 data audio, didapati *data training* sebesar 3115 data audio dan *data testing* sebesar 773 data audio.

7. Konstruksi model dengan TensorFlow

Dimana proses *training dataset* akan dilakukan menggunakan TensorFlow yang akan membentuk model TFLite dan model tersebut akan di-deploy pada Arduino Nano 33 BLE Sense Lite. Adapun *pseudocode* dipaparkan pada Gambar 3.15

```

Algorithm NeuralNetworkTraining
  Input: train_dataset, validation_dataset, input_length,
  classes, args

  Output: Trained neural network model

  # Set parameters

  EPOCHS = args.epochs or 100
  LEARNING_RATE = args.learning_rate or 0.005
  ENSURE_DETERMINISM = args.ensure_determinism
  BATCH_SIZE = args.batch_size or 32

  # Shuffle train dataset if determinism is not ensured
  If not ENSURE_DETERMINISM then
    Shuffle train_dataset with buffer size = BATCH_SIZE * 4
  EndIf

  # Batch the datasets

  Batch train_dataset with BATCH_SIZE, drop_remainder = False
  Batch validation_dataset with BATCH_SIZE, drop_remainder =
  False

  # Model architecture

  Initialize model as Sequential

  Add Reshape layer to model with shape (input_length / 13, 13)
  and input_shape = (input_length,)

  Add Conv1D layer to model with 8 filters, kernel_size = 3,
  padding = 'same', activation = 'relu'

  Add MaxPooling1D layer to model with pool_size = 2, strides =
  2, padding = 'same'

  Add Dropout layer to model with rate = 0.25

  Add Conv1D layer to model with 16 filters, kernel_size = 3,
  padding = 'same', activation = 'relu'

```

Gambar 3.15 *Pseudocode NeuralNetworkTraining*


```

    Add MaxPooling1D layer to model with pool_size = 2, strides =
    2, padding = 'same'

    Add Dropout layer to model with rate = 0.25

    Add Flatten layer to model

    Add Dense layer to model with number of units = classes, name
    = 'y_pred', activation = 'softmax'

    # Set optimizer

    Initialize optimizer as Adam with learning_rate =
    LEARNING_RATE, beta_1 = 0.9, beta_2 = 0.999

    # Add custom callback

    Append BatchLoggerCallback to callbacks with BATCH_SIZE,
    train_sample_count, epochs = EPOCHS, ensure_determinism =
    ENSURE_DETERMINISM

    # Compile the model

    Compile model with loss = 'categorical_crossentropy',
    optimizer = opt, metrics = ['accuracy']

    # Train the model

    Fit model with train_dataset, epochs = EPOCHS,
    validation_data = validation_dataset, verbose = 2, callbacks =
    callbacks

    # Optional: Flag to disable per-channel quantization

    disable_per_channel_quantization = False

End Algorithm

```

Gambar 3.16 *Pseudocode NeuralNetworkTraining (lanjutan)*

Pada Gambar 3.15 dan 3.16 dijelaskan Algoritma *NeuralNetworkTraining* dengan tujuan untuk konstruksi model dengan *data training* dan *data validation* yang telah ditentukan sebelumnya. Algoritma dimulai dengan penetapan beberapa parameter seperti jumlah *epoch*, laju pembelajaran (*learning_rate*), *flag* untuk memastikan determinisme (*ensure_determinism*) dan ukuran *batch* (*batch_size*). Jika determinisme tidak diaktifkan maka dataset pelatihan diacak dengan ukuran buffer tertentu. Kemudian, dataset pelatihan dan validasi dibagi menjadi *batch* sesuai ukuran *batch* yang ditentukan.

Selanjutnya, algoritma melakukan inisialisasi model sebagai model *Sequential* dengan menambahkan beberapa lapisan seperti lapisan *Reshape* yang bertujuan untuk mengubah bentuk *input*, lapisan *Conv1D* dan *MaxPooling1D* untuk ekstraksi fitur, lapisan *Dropout* untuk mencegah *overfitting*, lapisan *Flatten* serta

Dense untuk klasifikasi akhir. *Optimizer Adam* diatur dengan parameter yang telah ditentukan dan *callback* khusus ditambahkan untuk mencatat informasi *batch* selama pelatihan.

Setelahnya, model dikompilasi dengan menggunakan *loss categorical_crossentropy*, *optimizer Adam* dan metrik *accuracy*. Model kemudian dilatih menggunakan dataset pelatihan dengan jumlah epoch yang telah ditentukan, sementara dataset validasi digunakan untuk validasi selama pelatihan. Algoritma ini juga memungkinkan penggunaan *flag* opsional untuk menonaktifkan kuantisasi per saluran. Akhirnya, model yang telah terlatih dikembalikan sebagai *output*.

3.1.2 Post-Processing

Pada tahap *post-processing*, dilakukan proses *Fine Tuning* dan *Thresholding* dengan tujuan untuk meningkatkan kinerja model pada deteksi kata kunci tertentu yang tidak sepenuhnya tercakup dalam dataset asli. Adapun proses *Fine Tuning* dilakukan dengan cara mengambil model Keyword Spotting yang telah dilatih sebelumnya sebagai model dasar. Lalu ditambahkan lapisan tambahan atau modifikasi arsitektur untuk penyesuaian dengan dataset baru. Kemudian model dilatih dengan dataset baru menggunakan teknik *transfer learning* yaitu melatih ulang lapisan-lapisan terakhir model dengan data baru sambil membekukan lapisan-lapisan awal yang sudah dilatih. Pada proses *Thresholding*, dilakukan penyesuaian ambang batas probabilitas untuk menentukan apakah suara tertentu berisi kata kunci atau tidak.

3.1.3 Data yang Digunakan

Adapun data yang digunakan pada penelitian ini berupa data yang penulis rekam secara langsung ditambah dengan beberapa data dari Google Speech Commands Dataset. Data tersebut akan diproses pada tahap *pre-processing* (pada 3.1.1) sehingga menghasilkan jumlah data sebanyak 3888 data suara yang masing-masing berdurasi 1 detik. Adapun pembagian antara *data training* dan *data testing* dengan rasio 80:20 yang menghasilkan jumlah *data training* sebesar 3115 yang terdiri atas 565 kata kunci “kanan”, 582 kata kunci “kiri”, 553 kata kunci “maju”, 614 kata kunci “mundur”, 401 kata “noise” dan 400 kata “unknown”. Sedangkan jumlah *data testing* sebesar 773 terdiri atas 136 kata kunci “kanan”, 139 kata kunci “kiri”, 153 kata kunci “maju”, 146 kata kunci “mundur”, 99 kata “noise” dan 100 kata “unknown”.

Kata “unknown” ini merupakan kata yang tidak ditujukan sebagai kata kunci pada kasus Keyword Spotting. Adapun tujuan pelatihan kata yang bukan kata kunci target akan dipaparkan sebagai berikut.

- Meningkatkan Generalisasi Model

Generalisasi adalah kemampuan model untuk bekerja dengan baik pada data baru yang belum pernah dilihat. Melatih model dengan kata-kata yang tidak dikenal membantu model belajar membedakan antara kata kunci target dan berbagai input audio lain yang tidak relevan.

- Mengurangi *False Positives*

False Positives terjadi ketika model salah melakukan identifikasi data yang bukan target sebagai kata kunci. Dengan melatih model pada berbagai kata yang tidak dikenal, model belajar untuk lebih cermat dan mengurangi kemungkinan model mendeteksi kata kunci secara keliru saat kata lain diucapkan.

- Meningkatkan Ketahanan

Lingkungan audio dunia nyata sangat bervariasi dan berisik. Dengan melatih kata-kata yang tidak dikenal akan membantu model menjadi lebih tahan terhadap variasi pola bicara, aksen, dan kebisingan latar belakang. Ketahanan ini penting untuk memastikan sistem KWS bekerja dengan baik di berbagai pengguna dan lingkungan.

- Menangani Variabilitas Akustik

Ucapan memiliki variabilitas akustik yang signifikan karena perbedaan dalam suara, pengucapan, kecepatan, dan intonasi penutur. Dengan memasukkan kata-kata yang tidak dikenal dalam proses pelatihan, model belajar untuk fokus pada fitur spesifik dari kata kunci target dan menjadi kurang sensitif terhadap variasi yang tidak relevan. Ini membantu mempertahankan akurasi meskipun ada variabilitas akustik yang signifikan dalam input.

- Augmentasi Data

Memasukkan kata-kata yang tidak dikenal dapat dianggap sebagai bentuk augmentasi data. Dimana hal ini meningkatkan keragaman data pelatihan, yang dapat membantu mencegah model dari *overfitting* pada kata kunci target. *Overfitting* terjadi ketika model bekerja dengan baik pada data pelatihan tetapi

gagal dalam menggeneralisasi data baru. Dengan augmentasi data pelatihan dengan kata-kata yang tidak dikenal, model belajar fitur yang lebih kuat dan umum.

3.2 Perancangan Sistem

Tahap ini penulis akan menjelaskan tahap-tahap yang akan dilaksanakan dalam merancang sistem. Adapun perlengkapan yang diperlukan dalam membangun sistem robot ini adalah Arduino Nano 33 BLE Sense Lite sebagai inti dari program Keyword Spotting dan ESP32 sebagai inti dari sistem robot yang akan dikendalikan sesuai dengan keyword yang terdeteksi. 2 dinamo dibutuhkan untuk menggerakkan kaki-kaki robot yang sebelumnya dirakit menggunakan mesin 3d printer. Adapun Motor Shield dan baterai juga tersusun untuk membentuk robot yang dapat berfungsi dengan baik.

3.2.1 Komponen Robot Strandbeest

Dalam proses membangun Robot Strandbeest diperlukan berbagai komponen yang relevan dan dibutuhkan guna melaksanakan penelitian dengan optimal. Adapun Robot Strandbeest dirangkai dengan ragam komponen yang disajikan pada tabel 3.1.

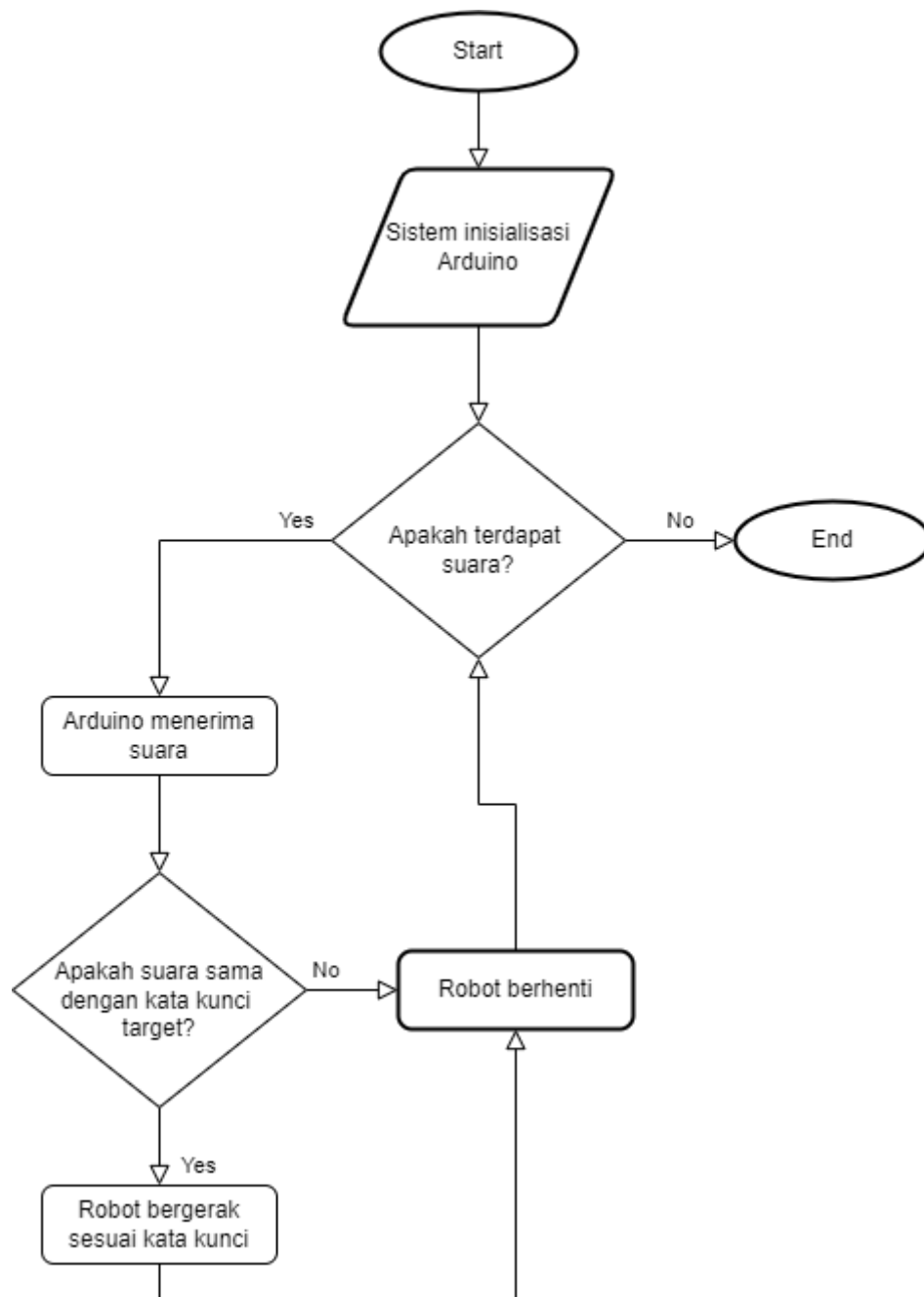
Tabel 3.1. Tabel Komponen Alat

No	Alat	Jumlah	Fungsi
1	Arduino Nano 33 BLE Sense Lite	1 Unit	Sebagai mikrofon dan perangkat yang memproses Keyword Spotting
2	ESP32 DEVKIT V1	1 Unit	Sebagai mesin utama yang mengendalikan robot
3	Driver Motor L298N	1 Unit	Modul penghubung antar ESP32 dengan motor penggerak robot
4	Motor DC Gearbox	2 Unit	Motor yang menggerakkan kaki-kaki robot

3.2.2 Flowchart Sistem Robot Strandbeest

Dalam membuat sistem robot diperlukan *flowchart* untuk menjelaskan dengan detail alur kerja sistem tersebut. Proses yang berlangsung dimulai dengan mengaktifkan

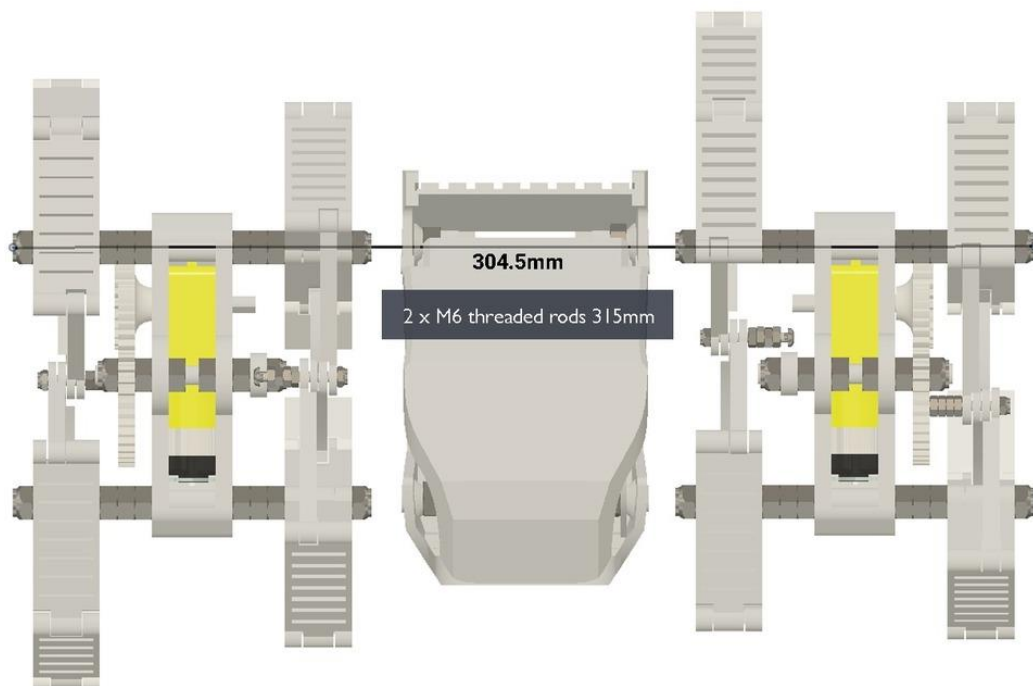
robot, lalu menunggu sebentar proses inialisasi Arduino, kemudian berlanjut pada pengambilan keputusan apakah robot mendeteksi suara, apabila suara terdeteksi maka mikrofon pada arduino akan menerima suara dan diproses apakah suara sesuai dengan kata kunci yang dilatih, apabila sesuai maka robot bergerak sesuai dengan kata kunci. Proses kembali ke bagian apakah suara ada dan akan berhenti apabila sistem dimatikan. Adapun *flowchart* sistem robot Strandbeest akan dipaparkan pada Gambar 3.17.



Gambar 3.17. *Flowchart* proses sistem Robot Strandbeest

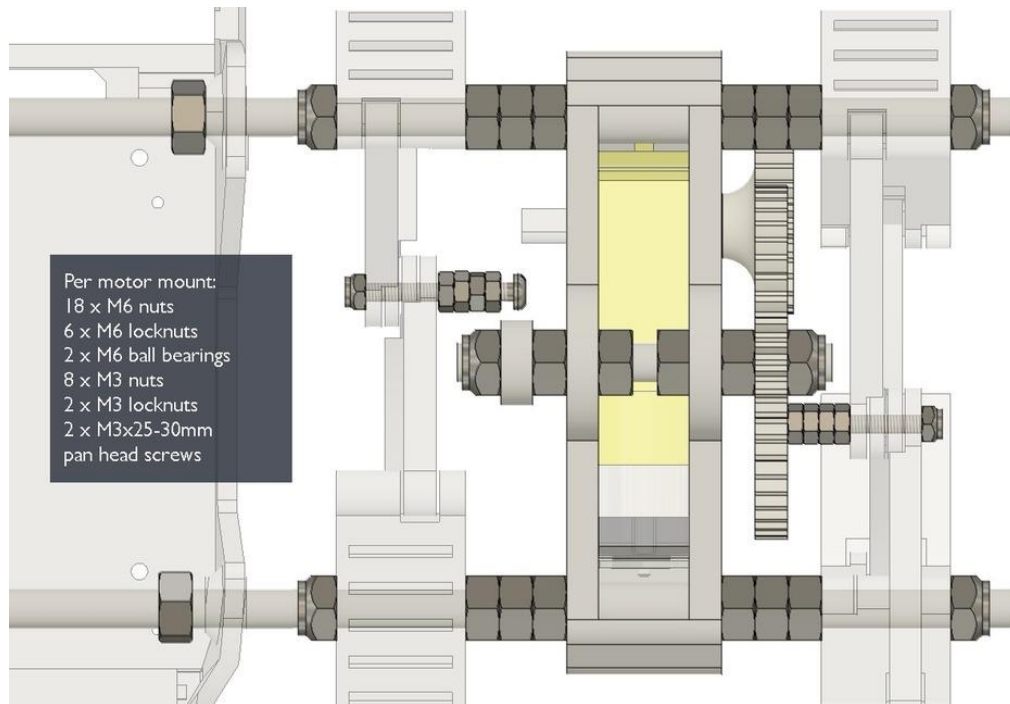
3.2.3 Rangkaian Sistem Robot Strandbeest

Rangkaian dari Robot Strandbeest yaitu, Kerangka Robot dirangkai sedemikian rupa dengan Dinamo dipasang pada *gear* robot untuk menggerakkan kaki robot. Motorshield dihubungkan pada Motor penggerak kaki robot dan juga terhubung pada ESP32 yang berfungsi sebagai mesin utama robot. Arduino Nano 33 yang berfungsi sebagai mikrofon juga terhubung pada ESP32. Arduino disini akan menerima suara dari pengguna, memproses suara tersebut dan diidentifikasi apakah suara tersebut merupakan perintah gerak yang sudah dilatih sebelumnya. Lalu hasil tersebut dikirim ke ESP32 agar diproses dan pada akhirnya robot bergerak sesuai perintah gerak tersebut. Tampilan desain robot dapat dilihat pada Gambar 3.18.



Gambar 3.18 Desain Robot Strandbeest

Adapun dalam perakitan robot diperlukan 2 batang berulir dengan ukuran M6 dengan panjang 315 mm. Dan untuk merakit kaki-kaki robot diperlukan setidaknya 18 mur ukuran M6, 6 mur kunci ukuran M6, 2 *ball bearings* ukuran M6, 8 mur ukuran M3, 2 mur kunci ukuran M3 dan 2 sekrup *pan head* ukuran 25-30 mm. Dan bahan-bahan tersebut diperlukan untuk tiap motor, dipaparkan pada Gambar 3.19.



Gambar 3.19 Desain Kaki Robot Strandbeest

Setelah perakitan robot, diperlukan penjelasan skema rangkaian komponen elektronika yang terdiri atas Arduino Nano 33 BLE Sense Lite, ESP32, Driver Motor L298N, 2 Motor DC Gearbox dan 3 Baterai 18650 Li-ion 3.7V. Adapun penjelasan rangkaian sebagai berikut.

1. Arduino Nano 33 BLE Sense Lite

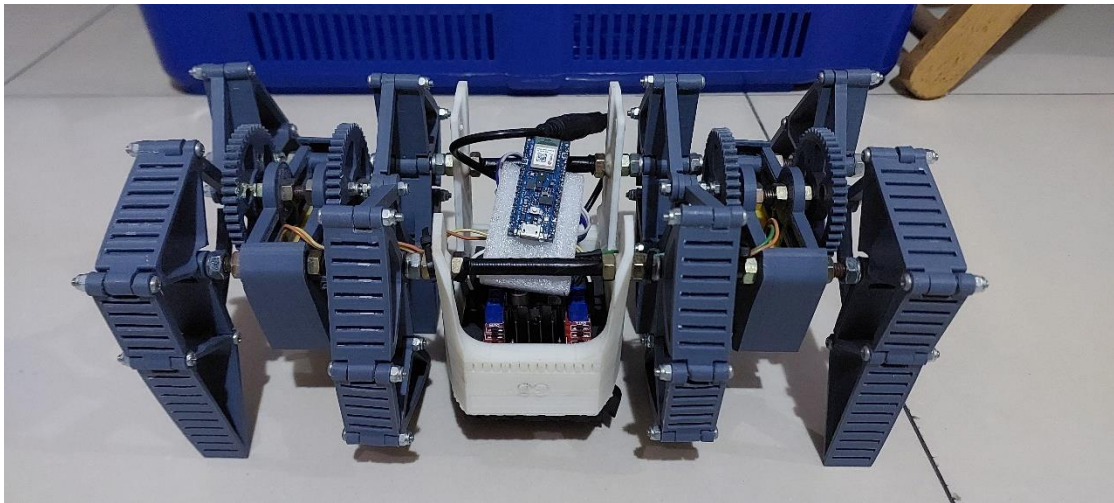
- Pin RX dihubungkan ke Pin GPIO17 ESP32
- Pin TX dihubungkan ke Pin GPIO16 ESP32
- Pin GND dihubungkan ke Pin GND ESP32
- Pin VIN dihubungkan ke Pin 5V Driver Motor L298N

2. ESP32

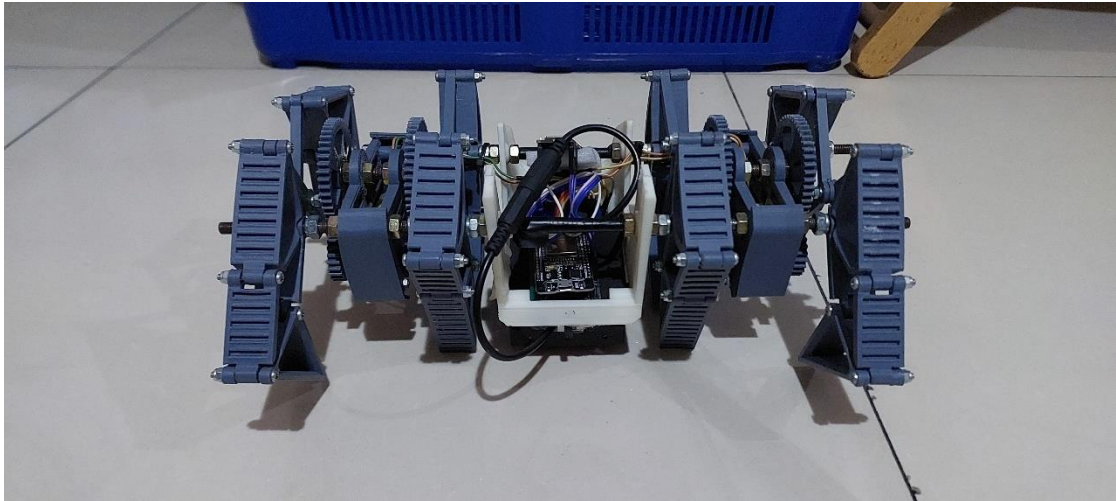
- Pin GND dihubungkan ke Pin GND Driver Motor L298N
- Pin VIN dihubungkan ke Pin 5V Driver Motor L298N
- Pin GPIO26 dihubungkan ke Pin IN1 Driver Motor L298N
- Pin GPIO27 dihubungkan ke Pin IN2 Driver Motor L298N
- Pin GPIO25 dihubungkan ke Pin ENA Driver Motor L298N
- Pin GPIO15 dihubungkan ke Pin IN3 Driver Motor L298N
- Pin GPIO2 dihubungkan ke Pin IN4 Driver Motor L298N

- Pin GPIO4 dihubungkan ke Pin ENB Driver Motor L298N
3. Driver Motor L298N
 - Pin OUT1 dihubungkan ke kutub positif Motor DC Gearbox pertama
 - Pin OUT2 dihubungkan ke kutub negatif Motor DC Gearbox pertama
 - Pin OUT3 dihubungkan ke kutub positif Motor DC Gearbox kedua
 - Pin OUT4 dihubungkan ke kutub negatif Motor DC Gearbox kedua
 4. Motor DC Gearbox
 - Disambungkan pada *gear* robot Strandbeest yang telah dirakit
 5. Baterai 18650 Li-ion 3.7V
 - Dipasang pada *holder battery* dengan kabel merah dihubungkan ke Pin 12V Driver Motor L298N dan kabel hitam dihubungkan pada Pin GND Driver Motor L298N.

Hasil dari perakitan dapat dilihat pada Gambar 3.20 untuk tampak depan dan pada Gambar 3.21 untuk tampak belakang



Gambar 3.20 Robot Strandbeest yang sudah dirangkai



Gambar 3.21 Robot Strandbeest tampak belakang

3.3 Evaluasi Model

Setelah proses perancangan sistem dan perakitan robot *Strandbeest*, akan dilakukan tahap Evaluasi Model. Evaluasi model diperlukan untuk mengukur kinerja model agar dapat disimpulkan model dapat bekerja secara optimal dengan tingkat akurasi yang diukur menggunakan persamaan (3.1)

$$Akurasi = \frac{Jumlah\ Prediksi\ Benar}{Total\ Jumlah\ Prediksi} \times 100\%$$

Rumus Akurasi (3.1)

BAB IV

IMPLEMENTASI DAN PENGUJIAN SISTEM

Pada bagian ini akan dijelaskan mengenai hasil penerapan dari suatu sistem yang penulis sudah rancang pada Bab 3, terdapat juga berbagai pengujian yang dilakukan terkait modul-modul yang digunakan.

4.1 Implementasi Sistem

4.1.1. Spesifikasi Sistem Hardware

Berikut disajikan spesifikasi dari *hardware* yang akan digunakan dalam penelitian ini, pada Tabel 4.1

Tabel 4.1 Tabel Spesifikasi Hardware

No	Hardware	Spesifikasi
1	Laptop HP 240 G8 Notebook	Processor Intel Core i5-1035G1 CPU @ 1.00GHz (8 CPUs), ~1.2GHz RAM 8GB SSD 256GB
2	ESP32 DEVKIT V1	Processor Dual-core Xtensa 32-bit LX6 Frekuensi Clock 160 MHz (up to 240 MHz) RAM 520 KB SRAM Memori Flash 4 MB
3	Arduino Nano 33 BLE Sense Lite	Processor ARM Cortex-M4 32-bit Frekuensi Clock 64 MHz RAM 256 KB SRAM Memori Flash 1 MB

4.1.2. Spesifikasi Sistem Software

Dan berikut disajikan spesifikasi dari *software* yang digunakan dalam penelitian ini, dipaparkan di Tabel 4.2 serta *library* yang akan digunakan pada penelitian terdapat pada Tabel 4.3

Tabel 4.2 Tabel Spesifikasi Software

No	Hardware	Software
1	Laptop HP 240 G8 Notebook	Windows 10 Home Single Language 64-bit (10.0, Build 19045) Arduino IDE 2.3.2 Visual Studio Code 1.90 Anaconda 23.7.4 Python 3.11.5 Librosa 0.10.1 Numpy 1.24.3 Soundfile 0.12.1

Tabel 4.3 Tabel Library

No	Packages	Keterangan
1	PDM.h	Berfungsi untuk penggunaan modul mikrofon pada Arduino Nano 33
2	HardwareSerial.h	Berfungsi untuk komunikasi UART antara ESP32 dan Arduino Nano 33
3	KeywordSpotting_inferencing.h	Merupakan Library hasil dari deploy model ke Arduino Nano 33

4.2 Pengujian Sistem

Tahap ini akan melaksanakan pengujian sistem. Adapun pengujian berupa uji coba Arduino Nano dapat mendeteksi kata kunci yang diucapkan, uji coba robot bergerak dengan ESP32, uji coba Arduino Nano mengirim data ke ESP32, uji coba robot gerak maju, gerak mundur, bergerak ke arah kiri dan bergerak ke arah kanan dan uji coba jarak antara sumber suara dengan mikrofon juga dilakukan. Pengujian ini akan terbagi atas pengujian secara fungsional dan uji secara kinerja sistem.

4.2.1 Pengujian Fungsional

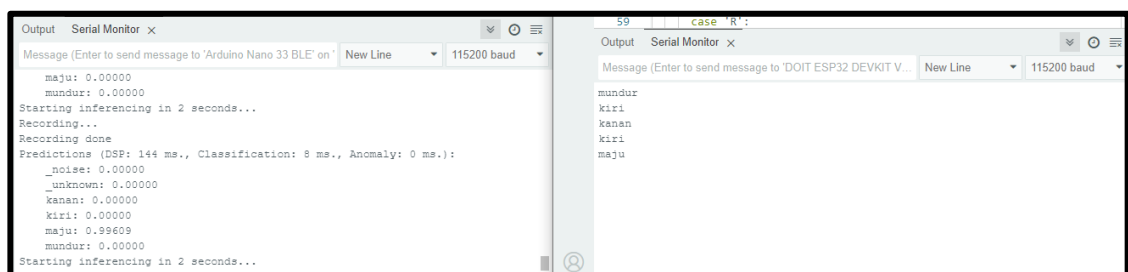
Pengujian Fungsional merupakan proses uji coba tiap-tiap fitur pada sistem Robot Strandbeest guna memastikan tidak terjadi kerusakan ataupun kinerja fitur yang gagal dimana dapat mengakibatkan alur kerja sistem yang tidak berjalan dengan sempurna.

Adapun pengujian secara fungsional dari robot ini akan dipaparkan di Tabel 4.4 berikut.

Tabel 4.4 Pengujian Fungsional

No	Fungsi	Hasil
1	Kendali motor dengan ESP32 untuk pergerakan robot	Berjalan
2	Komunikasi UART antara Arduino Nano 33 BLE Sense Lite dengan ESP32 DEVKIT V1	Berjalan, Lihat Gambar 4.1
3	Sinkronisasi antara perintah suara yang diucapkan dengan pengiriman perintah gerak ke ESP32	Berhasil

Dilakukan uji coba komunikasi UART antara Arduino Nano 33 dan ESP32 guna memastikan agar robot bergerak sinkron dengan perintah suara diterima. Dipaparkan pada Gambar 4.1



Gambar 4.1 Hasil komunikasi UART antara Arduino Nano 33 dengan ESP32

4.2.2 Pengujian Sistem Keyword Spotting

Setelah pengujian fungsional dan memastikan tiap-tiap fitur pada sistem dapat berjalan dengan baik, maka dilakukan pengujian sistem yang berupa uji coba model dengan *data testing* untuk mendapatkan akurasi kinerja model pada *data* yang telah diolah. Pengujian secara langsung (*real-time*) juga dilakukan dimana model akan dihadapkan pada suara manusia langsung yang diterima dari modul mikrofon yang terdapat pada Arduino Nano 33 BLE Sense Lite untuk mendapatkan akurasi model.

Diberikan Tabel 4.6 yang berisi hasil pengujian model pada *data testing* dengan jumlah data dapat dilihat pada Tabel 4.5

Tabel 4.5 Data Testing

Label	Jumlah
_noise	99
_unknown	100
kanan	136
kiri	139
maju	153
mundur	146

Tabel 4.6 Pengujian Model pada Data Testing

Actual \ Prediction	_noise	_unknown	kanan	kiri	maju	mundur
_noise	97	0	0	1	1	0
_unknown	1	96	0	0	0	0
kanan	1	0	135	0	0	0
kiri	0	0	0	138	0	0
maju	1	0	0	0	151	0
mundur	0	0	0	0	0	146

Dari Tabel 4.6, terdapat hasil *uncertain* yang tidak dipaparkan pada tabel hasil pengujian. Adapun data *uncertain* tersebut berupa data *_unknown* sebanyak 3 data, data “kiri” sebanyak 1 data, dan data “maju” sebanyak 1 data. Data *uncertain* ini terbentuk karena model tidak mampu menentukan data tersebut adalah merupakan bagian dari salah satu label yang telah dilatih sebelumnya. Dengan menggunakan persamaan akurasi, didapatkan hasil sebesar 99,34%.

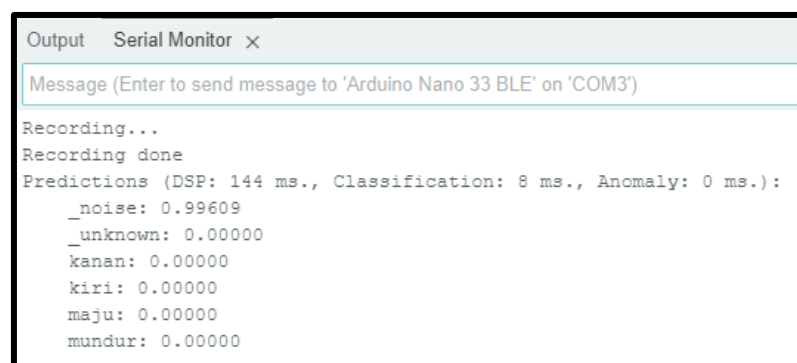
$$Akurasi = \frac{Jumlah\ Prediksi\ Benar}{Total\ Jumlah\ Prediksi} \times 100\%$$

$$Akurasi = \frac{97 + 96 + 135 + 138 + 151 + 146}{99 + 97 + 136 + 138 + 152 + 146} \times 100\%$$

$$Akurasi = \frac{763}{768} \times 100\%$$

$$Akurasi = 99,34\%$$

Selanjutnya akan dilakukan uji coba model secara langsung (*real-time*) dengan suara yang langsung diucapkan dan diterima oleh modul mikrofon Arduino. Diberikan contoh uji coba untuk masing-masing suara dimana dapat dilihat pada Gambar 4.2 sampai Gambar 4.7. Adapun penentuan *thresholding* ditetapkan dengan nilai 0.9 (Bluche et al., 2020) dari rentang 0 hingga 1 dimana bertujuan untuk mengatur keyakinan deteksi (*detection confidence*). Apabila model mendapatkan prediksi setidaknya 0.9 atau lebih, maka prediksi tersebut akan diyakini sebagai nilai yang benar positif (*true positive*).

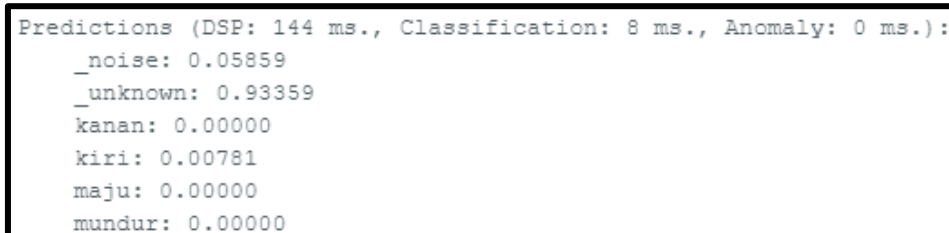


```

Output  Serial Monitor x
Message (Enter to send message to 'Arduino Nano 33 BLE' on 'COM3')
Recording...
Recording done
Predictions (DSP: 144 ms., Classification: 8 ms., Anomaly: 0 ms.):
_noise: 0.99609
_unknown: 0.00000
kanan: 0.00000
kiri: 0.00000
maju: 0.00000
mundur: 0.00000

```

Gambar 4.2 Hasil uji coba model, terdeteksi *noise* karena model memprediksi suara yang diterima dengan tingkat keyakinan (nilai *confidence*) sebesar 0.99609 yang dimana nilai tersebut lebih besar dari nilai threshold yang telah ditentukan yaitu sebesar 0.9.

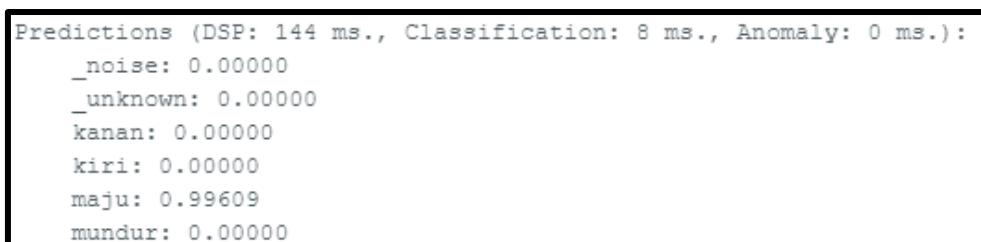


```

Predictions (DSP: 144 ms., Classification: 8 ms., Anomaly: 0 ms.):
_noise: 0.05859
_unknown: 0.93359
kanan: 0.00000
kiri: 0.00781
maju: 0.00000
mundur: 0.00000

```

Gambar 4.3 Hasil uji coba model, terdeteksi *unknown* dengan nilai *confidence* sebesar 0.93359



```

Predictions (DSP: 144 ms., Classification: 8 ms., Anomaly: 0 ms.):
_noise: 0.00000
_unknown: 0.00000
kanan: 0.00000
kiri: 0.00000
maju: 0.99609
mundur: 0.00000

```

Gambar 4.4 Hasil uji coba model, terdeteksi maju dengan nilai *confidence* sebesar 0.99609

```
Recording done
Predictions (DSP: 144 ms., Classification: 8 ms., Anomaly: 0 ms.):
  _noise: 0.00000
  _unknown: 0.00000
  kanan: 0.00000
  kiri: 0.00000
  maju: 0.00000
  mundur: 0.99609
```

Gambar 4.5 Hasil uji coba model, terdeteksi mundur dengan nilai *confidence* sebesar 0.99609

```
Predictions (DSP: 144 ms., Classification: 8 ms., Anomaly: 0 ms.):
  _noise: 0.00000
  _unknown: 0.00000
  kanan: 0.99609
  kiri: 0.00000
  maju: 0.00000
  mundur: 0.00000
```

Gambar 4.6 Hasil uji coba model, terdeteksi kanan dengan nilai *confidence* sebesar 0.99609

```
Predictions (DSP: 144 ms., Classification: 8 ms., Anomaly: 0 ms.):
  _noise: 0.00000
  _unknown: 0.00000
  kanan: 0.00000
  kiri: 0.99609
  maju: 0.00000
  mundur: 0.00000
```

Gambar 4.7 Hasil uji coba model, terdeteksi kiri dengan nilai *confidence* sebesar 0.99609

Berikutnya akan dipaparkan hasil pengujian secara langsung (*real-time*) yang dapat dilihat pada Tabel 4.8 dengan jumlah uji coba dipaparkan pada Tabel 4.7.

Tabel 4.7 Data Uji Coba Langsung

Label	Jumlah
kanan	50
kiri	50
maju	50
mundur	50

Tabel 4.8 Pengujian Model dengan Data Uji Coba Langsung

Actual \ Prediction	_noise	_unknown	kanan	kiri	maju	mundur
kanan	8	2	37	0	3	0
kiri	9	5	0	36	0	0
maju	7	3	4	0	36	0
mundur	11	2	0	0	0	37

Didapati hasil pengujian langsung menunjukkan bahwa performa model tidak seakurat dibanding dengan pada saat pengujian model dengan *data testing*. Hal ini dipengaruhi oleh berbagai faktor diantaranya, pengucapan yang kurang pas dengan proses mikrofon menangkap suara, aksentasi dan intonasi suara yang bervariasi serta lingkungan uji coba yang memiliki *noise*. Dengan menggunakan persamaan akurasi, didapatkan hasil sebesar 73%.

$$Akurasi = \frac{Jumlah\ Prediksi\ Benar}{Total\ Jumlah\ Prediksi} \times 100\%$$

$$Akurasi = \frac{37 + 36 + 36 + 37}{50 + 50 + 50 + 50} \times 100\%$$

$$Akurasi = \frac{146}{200} \times 100\%$$

$$Akurasi = 73\%$$

Terakhir, dipaparkan uji coba pergerakan robot dengan tujuan memberikan visualisasi hasil penelitian dan implementasi sistem Keyword Spotting pada Robot Strandbeest berteknologi TinyML. Demo program disajikan pada video yang dapat diakses di laman berikut: <https://youtu.be/67LTKv8QpZ8>.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang penulis ambil dari penelitian Implementasi *Keyword Spotting* untuk Menggerakkan Robot Strandbeest Berteknologi *TinyML* diuraikan sebagai berikut

- Dapat dilakukan implementasi *TinyML* pada sistem dengan sumber daya (*resource*) yang terbatas, dalam kasus ini yaitu mikrokontroler Arduino Nano 33 BLE Sense Lite.
- Akurasi yang didapat dari pengujian *model Keyword Spotting* mencapai 99.34% dengan jumlah *data testing* sebesar 773 data.
- Akurasi yang didapat dari pengujian secara langsung mencapai 73% dengan jumlah pengujian sebanyak 200 kali dengan 50 kali tiap kata kunci (maju, mundur, kiri, kanan).
- Robot Strandbeest berhasil diimplementasi sistem *Keyword Spotting* dengan teknologi *TinyML*

5.2 Saran

Saran penulis untuk meningkatkan penelitian terkait *Keyword Spotting* dengan teknologi *TinyML* adalah:

- Penambahan ragam suara baik secara pengucapan ataupun intonasi yang dapat meningkatkan keakuratan dalam pendeteksian suara.
- Penggunaan modul mikrokontroler yang terbaru agar penelitian lebih relevan dengan perkembangan teknologi sekarang.
- Menambahkan variasi kata kunci seperti berhenti dan jalan.

Daftar Pustaka

- Abadade, Y., Temouden, A., Bamoumen, H., Benamar, N., Chtouki, Y., & Hafid, A. S. (2023). A Comprehensive Survey on TinyML. *IEEE Access*, *11*, 96892–96922. <https://doi.org/10.1109/ACCESS.2023.3294111>
- Abdul, Z. Kh., & Al-Talabani, A. K. (2022). Mel Frequency Cepstral Coefficient and its Applications: A Review. *IEEE Access*, *10*, 122136–122158. <https://doi.org/10.1109/ACCESS.2022.3223444>
- Babiuch, M., Foltyniek, P., & Smutny, P. (2019). Using the ESP32 microcontroller for data processing. *Proceedings of the 2019 20th International Carpathian Control Conference, ICC 2019, May 2019*. <https://doi.org/10.1109/CarpathianCC.2019.8765944>
- Banbury, C. R., Reddi, V. J., Lam, M., Fu, W., Fazel, A., Holleman, J., Huang, X., Hurtado, R., Kanter, D., Lokhmotov, A., & Patterson, D. (2020). *Benchmarking TinyML Systems: Challenges and Direction*. <https://doi.org/10.48550/arXiv.2003.04821>
- Bansal, V., Pahwa, G., & Kannan, N. (2020). Cough classification for COVID-19 based on audio mfcc features using convolutional neural networks. *2020 IEEE International Conference on Computing, Power and Communication Technologies, GUCON 2020*, 604–608. <https://doi.org/10.1109/GUCON48875.2020.9231094>
- Bluche, T., Primet, M., & Gisselbrecht, T. (2020). *Small-Footprint Open-Vocabulary Keyword Spotting with Quantized LSTM Networks*. <http://arxiv.org/abs/2002.10851>
- Chaudhry, A., Batra, M., Gupta, P., Lamba, S., & Gupta, S. (2019). Arduino Based Voice Controlled Robot. *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 415–417. <https://doi.org/10.1109/ICCCIS48478.2019.8974532>
- Ding, R., Pang, C., & Liu, H. (2018). Audio-Visual Keyword Spotting Based on Multidimensional Convolutional Neural Network. *2018 25th IEEE International Conference on Image Processing (ICIP)*, 4138–4142. <https://doi.org/10.1109/ICIP.2018.8451096>
- Erol, B. A., Wallace, C., Benavidez, P., & Jamshidi, M. (2018). Voice Activation and Control to Improve Human Robot Interactions with IoT Perspectives. *2018 World Automation Congress (WAC)*, 1–5. <https://doi.org/10.23919/WAC.2018.8430412>
- Gouda, S. K., Kanetkar, S., Harrison, D., & Warmuth, M. K. (2018). *Speech Recognition: Keyword Spotting Through Image Recognition*. <http://arxiv.org/abs/1803.03759>

- Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3), 685–695. <https://doi.org/10.1007/s12525-021-00475-2>
- Kavlak, K., & Kartal, I. A. (2021). Design of Mobile Robot with Strandbeest Walking Mechanism to Overcome the Set Type Obstacle. *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 1–4. <https://doi.org/10.1109/HORA52670.2021.9461349>
- Kavlak, K., & Yongul, A. (2022). Design of Quadraped Mobile Robot with Strandbeest Walking Mechanism to Overcome Concave and Ramp Type Obstacles. *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 1–4. <https://doi.org/10.1109/HORA55278.2022.9799951>
- Lopez-Espejo, I., Tan, Z.-H., Hansen, J. H. L., & Jensen, J. (2022). Deep Spoken Keyword Spotting: An Overview. *IEEE Access*, 10, 4169–4199. <https://doi.org/10.1109/ACCESS.2021.3139508>
- Momeni, L., Afouras, T., Stafylakis, T., Albanie, S., & Zisserman, A. (2020). *Seeing wake words: Audio-visual Keyword Spotting*. <http://arxiv.org/abs/2009.01225>
- Pang, B., Nijkamp, E., & Wu, Y. N. (2020). Deep Learning With TensorFlow: A Review. *Journal of Educational and Behavioral Statistics*, 45(2), 227–248. <https://doi.org/10.3102/1076998619872761>
- Ray, P. P. (2022). A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4), 1595–1623. <https://doi.org/10.1016/j.jksuci.2021.11.019>
- Sakr, F., Bellotti, F., Berta, R., & De Gloria, A. (2020). Machine Learning on Mainstream Microcontrollers. *Sensors*, 20(9), 2638. <https://doi.org/10.3390/s20092638>
- Sanchez-Iborra, R., & Skarmeta, A. F. (2020). TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities. *IEEE Circuits and Systems Magazine*, 20(3), 4–18. <https://doi.org/10.1109/MCAS.2020.3005467>
- Singh, P., & Manure, A. (2020). Introduction to TensorFlow 2.0. In *Learn TensorFlow 2.0* (pp. 1–24). Apress. https://doi.org/10.1007/978-1-4842-5558-2_1
- V, V., C, R. A., Prasanna, R., Kakarla, P. C., PJ, V. S., & Mohan, N. (2022). *Implementation Of Tiny Machine Learning Models On Arduino 33 BLE For Gesture And Speech Recognition*. <http://arxiv.org/abs/2207.12866>
- Wang, J., & Li, S. (2022). *Keyword Spotting System and Evaluation of Pruning and Quantization Methods on Low-power Edge Microcontrollers*. <http://arxiv.org/abs/2208.02765>