Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

Asisten Virtual JARVIS Berbasis ESP32-S3 dan AI Dual-Path untuk Pengendalian Laptop melalui Perintah Suara

Andly Sofian Hasugian✉, Dedy Kiswanto✉, Yusrida Jelianti Sihite*✉, Juhraini Helfiana Lexa✉
Program Studi Ilmu Komputer, Universitas Negeri Medan, Indonesia

Abstrak

Keterbatasan memori dan daya komputasi pada mikrokontroler seperti ESP32-S3 menjadi hambatan utama dalam menjalankan pemrosesan bahasa alami berbasis *Large Language Model* (LLM) secara langsung. Sistem asisten suara berbasis LLM yang sudah ada, seperti Home Assistant (fitur Assist) dan Willow, memanfaatkan perangkat ESP sebagai antarmuka suara dengan pemrosesan di server, tetapi umumnya diarahkan pada otomasi rumah tangga, bukan pengendalian aplikasi pada laptop. Penelitian ini bertujuan merancang dan membangun asisten virtual terdistribusi bernama JARVIS berbasis *Internet of Things* (IoT) dan mikrokontroler ESP32-S3 untuk mengendalikan laptop melalui perintah suara secara *real-time*. Metode yang digunakan adalah *Research and Development* (R&D) dengan pendekatan *prototyping*, mengintegrasikan mikrofon I2S INMP441, layar TFT ST7735, amplifier MAX98357A, serta sistem pemrosesan kecerdasan buatan dua jalur (*Dual-Path AI System*). Mode Cloud (FRIDAY) memanfaatkan Google Gemini 2.5 Flash dan Groq (Llama 3.3 70B), sedangkan Mode Local (JARVIS) menjalankan model Qwen2.5-3B-Instruct hasil *fine-tuning* LoRA dengan keluaran JSON Schema terstruktur. Pengujian terkontrol terhadap 40 percobaan (5 pengulangan pada 4 kategori perintah) menunjukkan bahwa tingkat keberhasilan eksekusi akhir perintah laptop mencapai 95% (38 dari 40 percobaan) secara keseluruhan, di mana kegagalan inferensi model AI ditangani secara otomatis oleh *fallback* parser manual. Waktu tanggap (*latency*) total *request handler* rata-rata tercatat sebesar 1,91 detik pada Mode Cloud dan 18,11 detik pada Mode Local akibat pembebanan komputasi inferensi LLM dan Piper TTS pada CPU laptop. Kebaruan penelitian ini terletak pada integrasi antarmuka suara fisik ESP32-S3 dengan sistem eksekusi aplikasi laptop melalui arsitektur pemrosesan *dual-path* dalam satu sistem terpadu. Keterbatasan penelitian ini terletak pada tingginya latensi pemrosesan lokal di CPU laptop serta ketergantungan mode *cloud* terhadap stabilitas jaringan internet.
Kata Kunci: Asisten Virtual; ESP32-S3; Internet of Things; Large Language Model; Perintah Suara.

Abstract

Memory and computational constraints on microcontrollers such as the ESP32-S3 pose a major obstacle to directly executing Large Language Model (LLM)-based natural language processing. Existing LLM-based voice assistant systems, such as Home Assistant (Assist feature) and Willow, utilize ESP devices as voice interfaces with server-side processing, but are generally tailored toward home automation rather than laptop application control. This study aims to design and develop a distributed virtual assistant named JARVIS based on the Internet of Things (IoT) and the ESP32-S3 microcontroller to control laptops via real-time voice commands. The Research and Development (R&D) method with a prototyping approach was applied, integrating an INMP441 I2S microphone, an ST7735 TFT display, a MAX98357A amplifier, and a Dual-Path AI processing system. The Cloud Mode (FRIDAY) utilizes Google Gemini 2.5 Flash and Groq (Llama 3.3 70B), while the Local Mode (JARVIS) runs a LoRA fine-tuned Qwen2.5-3B-Instruct model with structured JSON Schema output. Controlled testing across 40 trials (5 repetitions per 4 command categories) demonstrated an overall final execution success rate of 95% (38 out of 40 trials), with AI inference failures automatically handled by a keyword-based manual parser fallback. The average total request handler latency was 1.91 seconds in Cloud Mode and 18.11 seconds in Local Mode due to the computational load of LLM inference and Piper TTS on the laptop CPU. The novelty of this research lies in integrating a physical ESP32-S3 voice interface with a laptop application execution system through a dual-path processing architecture within a single framework. The limitations of this study include high local processing latency on CPU and the dependency of the cloud mode on internet network stability.
Keywords: Virtual Assistant; ESP32-S3; Internet of Things; Large Language Model; Voice Commands.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

***Corresponding Author:**
Yusrida Jelianti Sihite, Universitas Negeri Medan, Indonesia
Email: yusrida041106@gmail.com

1. PENDAHULUAN

Perkembangan kecerdasan buatan (*Artificial Intelligence*), *Internet of Things* (IoT), dan sistem tertanam mendorong pengembangan asisten virtual yang mampu membantu pengguna berinteraksi dengan perangkat elektronik melalui perintah suara [1]. Dalam penggunaan laptop, asisten virtual memungkinkan pengguna menjalankan aplikasi, melakukan pencarian informasi, dan melaksanakan berbagai instruksi tanpa harus selalu menggunakan papan ketik atau tetikus [2]. Pengembangan asisten desktop berbasis perintah suara memanfaatkan kecerdasan buatan untuk meningkatkan kemudahan dan efisiensi interaksi antara pengguna dan komputer [3]. Selain kemudahan penggunaan, aspek privasi juga menjadi perhatian penting dalam pengembangan asisten virtual, khususnya terkait pengelolaan data dan pemrosesan instruksi pengguna [4]. Oleh karena itu, pengembangan asisten desktop yang mengintegrasikan teknologi kecerdasan buatan dan perangkat tertanam menjadi salah satu pendekatan potensial untuk mendukung interaksi manusia dengan komputer secara lebih praktis [5].

Meskipun demikian, pengembangan asisten virtual berbasis suara masih menghadapi sejumlah tantangan, terutama dalam hal integrasi perangkat keras, pemrosesan instruksi, dan komunikasi antarkomponen sistem. Salah satu pendekatan yang dapat diterapkan adalah mengintegrasikan ESP32 dengan laptop sebagai pusat pemrosesan instruksi. ESP32 dapat dimanfaatkan untuk mendukung pengenalan perintah suara sekaligus menghubungkan instruksi pengguna dengan sistem kendali perangkat elektronik [6]. Penggunaan ESP32 dalam aplikasi kendali berbasis suara juga menunjukkan potensi mikrokontroler tersebut dalam membangun sistem interaktif [7]. Namun, penerapan sistem tersebut memerlukan perhatian terhadap penempatan pemrosesan kecerdasan buatan, baik secara lokal pada laptop maupun melalui layanan *cloud*. Perbedaan mekanisme pemrosesan tersebut berpotensi memengaruhi waktu tanggap, kestabilan komunikasi, dan keberhasilan pelaksanaan perintah. Dengan demikian, diperlukan rancangan sistem yang mampu mengintegrasikan perangkat keras dan pemrosesan kecerdasan buatan secara efektif untuk mendukung pengendalian laptop melalui perintah suara.

Penelitian terdahulu telah menunjukkan penerapan ESP32 dan perintah suara dalam berbagai sistem otomasi perangkat elektronik. Sistem *smart home*, misalnya, menggabungkan teknologi pengenalan suara dan sensor cahaya untuk mendukung pengendalian perangkat secara otomatis [8]. Pengendalian peralatan rumah tangga melalui perintah suara berbasis IoT juga telah dikembangkan dengan memanfaatkan ESP32 [9]. Selain itu, integrasi Google Assistant dengan sistem *smart home* menjadi pendekatan alternatif untuk memungkinkan pengguna mengendalikan perangkat melalui instruksi suara [10]. Penerapan sistem kendali suara juga memiliki potensi dalam mendukung aksesibilitas bagi pengguna yang membutuhkan alternatif interaksi dengan perangkat elektronik [11]. Berbagai pendekatan tersebut turut dibahas dalam kajian sistematis mengenai penerapan IoT untuk pengendalian perangkat melalui suara [12]. Penelitian lainnya mengembangkan asisten virtual berbasis suara dengan arsitektur terdistribusi yang memisahkan pemrosesan kecerdasan buatan dari pengendalian perangkat keras [13]. Pendekatan serupa juga diterapkan pada sistem IoT berbasis ESP32 yang memisahkan proses akuisisi data pada perangkat tepi dari pemrosesan dan analisis pada server[14]. Hasil penelitian tersebut menunjukkan bahwa integrasi perangkat tertanam, pengenalan suara, dan kecerdasan buatan memiliki potensi untuk diterapkan dalam berbagai kebutuhan interaksi manusia dengan perangkat elektronik.

Berdasarkan penelitian terdahulu, sebagian besar penerapan ESP32 dan perintah suara berfokus pada otomasi rumah tangga, pengendalian perangkat elektronik, serta sistem interaksi berbasis suara. Sementara itu, penelitian mengenai asisten virtual berbasis kecerdasan buatan juga telah mengembangkan pemisahan antara pemrosesan instruksi dan pengendalian perangkat keras [13]. Namun, masih terdapat peluang pengembangan sistem yang secara khusus mengintegrasikan asisten

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

desktop, perangkat tertanam, dan pemrosesan kecerdasan buatan untuk pengendalian laptop dalam satu kesatuan sistem [15]. Selain itu, perbandingan penerapan pemrosesan berbasis *cloud* dan model kecerdasan buatan lokal dalam sistem asisten desktop yang terintegrasi dengan ESP32 masih menjadi aspek yang perlu dikaji lebih lanjut. Kesenjangan tersebut menjadi dasar pengembangan sistem asisten virtual yang tidak hanya berfungsi sebagai pengendali perangkat elektronik, tetapi juga mampu mendukung pelaksanaan instruksi pada laptop .

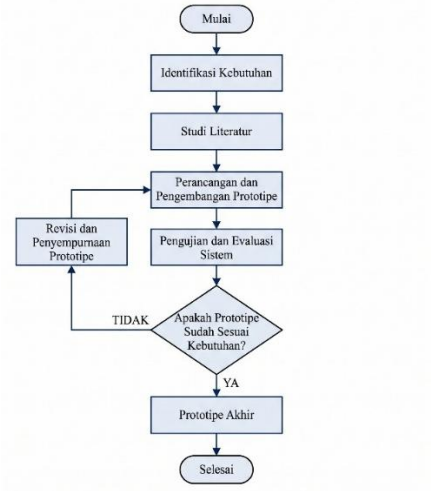
Untuk mengatasi kesenjangan tersebut, penelitian ini mengusulkan rancang bangun asisten virtual JARVIS yang mengintegrasikan ESP32/ESP32-S3, perangkat audio, layar TFT, dan pemrosesan kecerdasan buatan pada laptop. Sistem yang dikembangkan menyediakan dua pilihan mekanisme pemrosesan, yaitu mode *cloud* Friday yang memanfaatkan Google Gemini dan Groq serta mode lokal Jarvis yang menggunakan model Qwen 3B yang telah melalui proses *fine-tuning*. Kedua mode tersebut dirancang untuk memproses instruksi pengguna dengan pendekatan pemrosesan yang berbeda, sehingga memungkinkan penerapan asisten virtual melalui layanan *cloud* maupun model lokal. Kebaruan penelitian ini terletak pada integrasi perangkat tertanam sebagai antarmuka interaksi dengan pengguna dan laptop sebagai pusat pemrosesan kecerdasan buatan, disertai penyediaan dua mode pemrosesan dalam satu sistem untuk mendukung pengendalian laptop melalui perintah suara. Dengan pendekatan tersebut, sistem yang dikembangkan diarahkan untuk memberikan alternatif interaksi manusia dengan komputer yang lebih terintegrasi.

Penelitian ini bertujuan untuk merancang dan membangun asisten virtual JARVIS berbasis IoT menggunakan ESP32 untuk mendukung pengendalian laptop melalui perintah suara serta mengevaluasi kinerja sistem yang dihasilkan. Evaluasi dilakukan berdasarkan keberhasilan pelaksanaan perintah, waktu tanggap, dan kestabilan komunikasi sistem. Kontribusi penelitian ini meliputi rancangan integrasi perangkat tertanam dan kecerdasan buatan untuk pengendalian laptop, penerapan dua mekanisme pemrosesan melalui mode *cloud* dan model lokal, serta evaluasi kinerja kedua pendekatan dalam mendukung interaksi pengguna dengan komputer. Hasil penelitian diharapkan dapat menjadi referensi dalam pengembangan asisten virtual berbasis IoT dan kecerdasan buatan, khususnya untuk penerapan sistem kendali laptop melalui perintah suara.

2. METODE PENELITIAN

2.1 Desain dan Tahapan Penelitian

Penelitian ini menggunakan pendekatan *Research and Development* (R&D) dengan metode *prototyping* untuk merancang dan membangun asisten virtual JARVIS berbasis *Internet of Things* (IoT) menggunakan mikrokontroler ESP32-S3. Pendekatan ini dipilih karena penelitian berfokus pada pengembangan prototipe sistem terdistribusi yang mengintegrasikan pemrosesan perintah suara, kecerdasan buatan berbasis *cloud* dan model lokal, serta eksekusi perintah pada laptop.



Gambar 1. Tahapan R&D

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

Tahap identifikasi kebutuhan dilakukan dengan menentukan kebutuhan perangkat keras dan perangkat lunak yang digunakan dalam sistem. Perangkat keras meliputi ESP32-S3, mikrofon digital INMP441, layar TFT ST7735, tombol *Push-to-Talk*, amplifier MAX98357A, dan speaker. Sementara itu, kebutuhan perangkat lunak mencakup pemrosesan *Speech-to-Text* (STT), *Text-to-Speech* (TTS), pengaturan pemilihan model kecerdasan buatan, serta integrasi dua jalur pemrosesan (*dual-path processing*).

Tahap studi literatur dilakukan dengan menelaah berbagai sumber ilmiah, seperti buku, jurnal, dan penelitian terdahulu yang berkaitan dengan IoT, mikrokontroler, pengenalan suara, dan penerapan kecerdasan buatan. Tahap ini bertujuan memperoleh landasan teoritis dan konseptual sekaligus mengidentifikasi pendekatan yang relevan untuk pengembangan sistem.

Tabel 1. Tahapan Iterasi Pengembangan Prototipe

Tahap	Kegiatan	Tujuan
Iterasi 1	Perakitan ESP32-S3, mikrofon, TFT, dan speaker	Memastikan seluruh komponen berfungsi
Iterasi 2	Integrasi komunikasi ESP32-S3 dengan laptop	Memastikan audio dapat dikirim dan diterima
Iterasi 3	Integrasi STT, AI Cloud, dan AI Local	Memastikan kedua mode dapat memproses perintah
Iterasi 4	Pengujian dan evaluasi sistem	Mengidentifikasi kendala pemrosesan dan eksekusi perintah
Iterasi 5	Penyempurnaan dan pengujian ulang prototipe	Memastikan sistem berjalan sesuai kebutuhan

Tahap pengembangan prototipe dilakukan dengan mengintegrasikan perangkat keras dan perangkat lunak menjadi satu sistem yang saling terhubung. Selanjutnya, prototipe diuji melalui serangkaian skenario untuk mengevaluasi keberhasilan pelaksanaan perintah, waktu tanggap, dan kestabilan komunikasi. Hasil pengujian digunakan sebagai dasar revisi dan penyempurnaan sistem secara iteratif hingga diperoleh prototipe yang sesuai dengan kebutuhan penelitian.

2.2 Perancangan dan Integrasi Perangkat Keras

Perancangan perangkat keras sistem JARVIS menggunakan ESP32-S3 sebagai perangkat utama yang bertugas menangani input dan output suara serta komunikasi dengan laptop. Komponen yang digunakan meliputi mikrofon INMP441 untuk menangkap suara pengguna, layar TFT ST7735 untuk menampilkan status sistem, tombol Push-to-Talk (PTT) sebagai pemicu perekaman, serta modul MAX98357A yang terhubung ke speaker untuk menghasilkan keluaran audio. Komunikasi mikrofon dan speaker menggunakan protokol I2S, sedangkan layar TFT menggunakan protokol SPI.

Tabel 2. Pemetaan Pin Komponen Perangkat Keras ke ESP32-S3

Perangkat	Pin Perangkat	Sambungan ESP32-S3
TFT ST7735	MOSI / SDA	GPIO 11
	SCK / SCL	GPIO 18
	CS	GPIO 5
	DC / A0	GPIO 17
	RESET	GPIO 16
	VCC / LED	3V3
	GND	GND
Mikrofon INMP441	WS / LRCL	GPIO 39
	SCK / BCLK	GPIO 40
	SD / DOUT	GPIO 41
	VDD	3V3

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

	GND	GND
Amplifier MAX98357A	BCLK	GPIO 42
	LRC / WS	GPIO 15
	DIN	GPIO 14
	VIN / GND	Catu daya dan GND bersama
Tombol Push-to-Talk	Salah satu kaki	GPIO 4
	Kaki lainnya	GND

Proses kerja perangkat dimulai ketika pengguna menekan tombol PTT. ESP32-S3 mengaktifkan mikrofon untuk merekam suara dan menampilkan status "Mendengar" pada layar TFT. Rekaman suara kemudian dikirim dalam format WAV melalui jaringan Wi-Fi ke laptop untuk diproses. Setelah pemrosesan selesai, respons audio dikirim kembali ke ESP32-S3 dan dikeluarkan melalui speaker.

2.3 Pengembangan Pemrosesan Kecerdasan Buatan

Pemrosesan kecerdasan buatan pada sistem JARVIS dirancang menggunakan arsitektur dua jalur (*dual-path AI system*), yaitu pemrosesan berbasis *cloud* dan pemrosesan secara lokal pada laptop. Kedua jalur tersebut digunakan untuk memproses instruksi pengguna setelah suara dikonversi menjadi teks melalui proses STT. Pemilihan jalur pemrosesan diatur oleh *Dual-Path Router* berdasarkan ketersediaan koneksi internet dan pengaturan mode yang digunakan.

2.3.1 Mode Cloud (Friday)

Mode *cloud* atau Friday memanfaatkan *Application Programming Interface* (API) dari penyedia layanan kecerdasan buatan eksternal. Google Gemini API digunakan sebagai model pemroses utama, sedangkan Groq API digunakan sebagai mekanisme cadangan (*fallback*) apabila terjadi gangguan pada layanan utama atau keterbatasan kuota. Kedua layanan menerima instruksi sistem (*system prompt*) yang sama untuk memproses perintah pengguna. Keluaran model dirancang dalam format *JavaScript Object Notation* (JSON) agar dapat diinterpretasikan oleh program pada laptop. Struktur keluaran tersebut memuat informasi respons teks dan instruksi tindakan (*intent*) yang digunakan sebagai dasar pelaksanaan perintah, seperti membuka aplikasi, memutar musik, mengatur volume, dan melakukan pencarian informasi.

2.3.2 Mode Lokal (Jarvis)

Mode lokal atau JARVIS dirancang menggunakan model bahasa Qwen2.5-3B-Instruct yang dijalankan secara lokal pada laptop. Model dasar menggunakan teknik kuantisasi 4-bit untuk mengurangi kebutuhan sumber daya komputasi. Sebelum diterapkan, model melalui proses *fine-tuning* menggunakan metode *Low-Rank Adaptation* (LoRA) dengan dataset sebanyak 425 contoh instruksi dan respons dalam format JSON. Dataset tersebut disusun melalui pemformatan instruksi pengguna ke struktur respons JSON (chat template) dan mencakup 8 intent, seperti perintah membuka aplikasi, mengatur volume, pencarian web, serta percakapan umum. Data tersebut dibagi menjadi 425 data latih, dan 40 data uji. Salah satu contoh data yang digunakan adalah perintah pengguna, yaitu "Buka Spotify", dengan respons berupa objek JSON yang memuat daftar aksi dan balasan sistem.

Proses *fine-tuning* menggunakan metode LoRA dengan nilai *rank* sebesar 8, *alpha* sebesar 8, dan *dropout* sebesar 0. Modul yang dilatih meliputi q_proj, k_proj, v_proj, dan o_proj. Pelatihan dilakukan selama 3 *epoch* dengan *learning rate* sebesar 2×10^{-4} dan optimizer AdamW 8-bit. Proses pelatihan menggunakan model dasar dengan kuantisasi 4-bit dan dijalankan melalui platform Google Colab menggunakan GPU Nvidia T4. Setelah pelatihan selesai, model dikonversi ke format GGUF dengan tipe kuantisasi Q4_K_M dan ukuran sekitar 1,8 GB.

2.4 Alur Kerja dan Komunikasi Sistem

Sistem JARVIS menerapkan arsitektur komunikasi terdistribusi antara ESP32-S3 dan laptop. Komunikasi antarkomponen perangkat keras menggunakan protokol I2S dan SPI, sedangkan pertukaran data antara ESP32-S3 dan laptop menggunakan HTTP melalui jaringan Wi-Fi lokal. Selain ESP32-S3, sistem juga menerima masukan melalui Telegram berupa pesan teks dan *voice note*.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

Model hasil pelatihan kemudian dijalankan secara lokal pada laptop dengan spesifikasi perangkat keras berupa prosesor Intel Core i5, GPU RTX 3050 4GB, dan RAM sebesar 16GB. Lingkungan perangkat lunak yang digunakan meliputi sistem operasi Windows, serta Ollama. Model lokal digunakan untuk memproses instruksi pengguna tanpa mengirimkan permintaan AI ke layanan cloud. Namun, koneksi Wi-Fi tetap diperlukan untuk komunikasi antara ESP32-S3 dan laptop.

Pada input melalui ESP32-S3, mikrofon INMP441 merekam suara setelah tombol Push-to-Talk (PTT) ditekan. Audio direkam dengan sample rate sebesar 16.000 Hz, durasi rata-rata sekitar 5,24 detik, dan disimpan dalam format WAV mono 16-bit dengan ukuran rata-rata 167.700 byte (sekitar 164 KiB). File audio dikirim melalui endpoint POST /voice_command pada server Flask di laptop. Sementara itu, voice note Telegram diteruskan ke proses transkripsi melalui bot Telegram.

Audio dari kedua sumber tersebut diproses menggunakan *faster-whisper* model *base* dengan konfigurasi CPU int8 dan bahasa Indonesia. *Whisper* merupakan model pengenalan suara otomatis (*Automatic Speech Recognition*/ASR) yang dikembangkan untuk mentranskripsikan ucapan ke dalam teks. Implementasi *faster-whisper* digunakan untuk menjalankan proses transkripsi secara lokal pada laptop [16]. Hasil transkripsi berupa teks kemudian diteruskan ke Router Perintah.

Router Perintah menentukan jalur pemrosesan berdasarkan mode yang dipilih pengguna, yaitu FRIDAY, JARVIS, atau manual. Pemilihan mode tidak dilakukan secara otomatis berdasarkan ketersediaan koneksi internet. Koneksi internet berpengaruh terhadap ketersediaan layanan eksternal pada mode FRIDAY, sedangkan mode JARVIS tetap menggunakan layanan model lokal.

Pada mode JARVIS, teks diproses menggunakan model Qwen2.5-3B-Instruct yang telah melalui *fine-tuning* dengan metode LoRA. Qwen2.5 merupakan keluarga model bahasa besar (*Large Language Model*/LLM) yang dikembangkan oleh Alibaba Cloud. Proses *fine-tuning* menggunakan LoRA, yaitu metode adaptasi model yang menambahkan matriks berperingkat rendah pada bobot model yang telah dilatih sebelumnya sehingga jumlah parameter yang perlu diperbarui selama pelatihan menjadi lebih sedikit [17]. Model dijalankan secara lokal melalui Ollama.

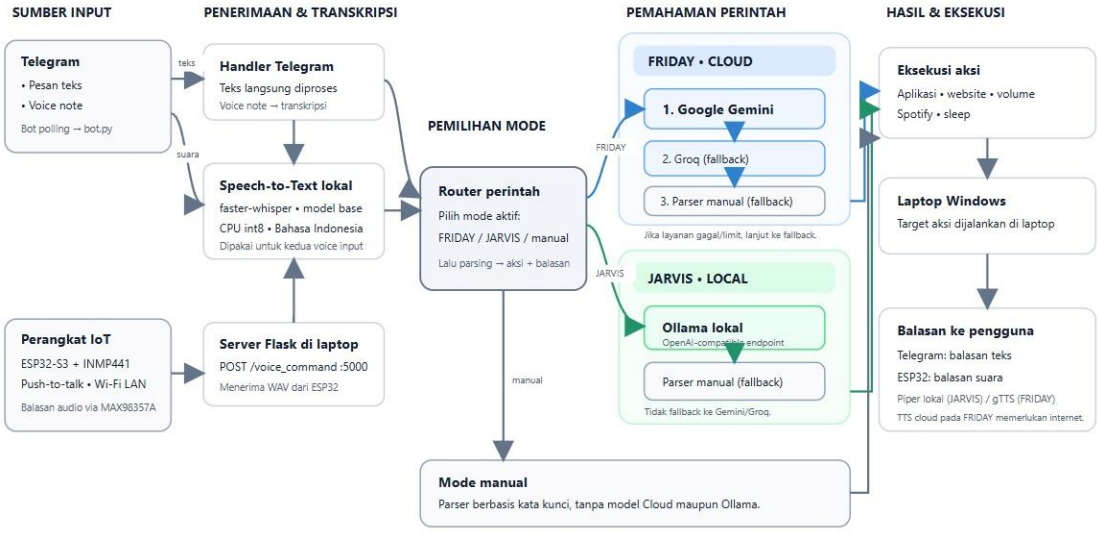
Model AI pada mode FRIDAY dan JARVIS menggunakan instruksi sistem untuk menghasilkan keluaran terstruktur dalam format JSON. Keluaran tersebut memuat daftar tindakan (*actions*) dan teks balasan (*reply*). Salah satu contoh format keluaran yang digunakan adalah sebagai berikut:

```
{
  "actions": [
    {
      "type": "app",
      "target": "spotify"
    }
  ],
  "reply": "Opening Spotify."
}
```

Parser manual juga menghasilkan struktur tindakan yang diteruskan ke tahap eksekusi. Laptop kemudian membaca jenis tindakan dan target yang diterima untuk menjalankan perintah, seperti membuka aplikasi, mengakses situs web, mengatur volume, mengontrol Spotify, atau menjadwalkan proses *shutdown*. Setelah eksekusi, sistem menghasilkan balasan kepada pengguna. Pada mode JARVIS, teks balasan diubah menjadi audio menggunakan Piper, yaitu sistem *Text-to-Speech* (TTS) yang dapat dijalankan secara lokal [18]. Sementara itu, pada mode FRIDAY digunakan gTTS. Audio hasil sintesis dikirim kembali ke ESP32-S3 melalui Wi-Fi untuk dikeluarkan melalui amplifier MAX98357A dan speaker. Untuk input Telegram, sistem memberikan balasan melalui kanal Telegram.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780



Gambar 2. Flowchart Alur Kerja Sistem JARVIS

Gambar 2 menunjukkan bahwa kedua jalur pemrosesan menggunakan tahapan input dan transkripsi yang sama, tetapi berbeda pada proses pemahaman perintah. Mode FRIDAY menggunakan layanan AI berbasis Cloud dengan mekanisme fallback Gemini, Groq, dan parser manual, sedangkan mode JARVIS menggunakan model lokal melalui Ollama dengan fallback parser manual. Kedua jalur kemudian diteruskan ke tahap eksekusi tindakan dan pemberian respons.

2.4.1 Keamanan Eksekusi Perintah

Sebelum perintah dijalankan di perangkat laptop, sistem melakukan verifikasi dan validasi struktur JSON yang dihasilkan oleh model AI (baik Cloud maupun Local) atau parser manual. Aplikasi memvalidasi jenis tindakan (type) yang harus terdaftar dalam daftar tindakan yang didukung (*whitelist* jenis aksi: app, close_app, website, spotify_play, spotify_stop, volume, sleep, dan info). Selain itu, sasaran tindakan (target) dipadankan dengan pemetaan aplikasi atau situs web yang diizinkan dalam sistem untuk mencegah eksekusi perintah berbahaya (*arbitrary code execution*) atau pembukaan berkas sistem yang tidak sah. Tindakan berisiko tinggi seperti penanganan akses sistem atau penjadwalan pemutusan daya (*shutdown/sleep*) diproses melalui parameter waktu khusus yang divalidasi oleh parser sebelum perintah OS dijalankan.

2.5 Pengujian dan Evaluasi Sistem

Pengujian dilakukan untuk mengevaluasi kemampuan sistem dalam memahami perintah, menjalankan tindakan, dan memberikan respons melalui jalur Cloud maupun lokal. Pengujian mencakup beberapa jenis perintah, seperti membuka aplikasi, mengatur volume, melakukan pencarian web, mengatur waktu *shutdown*, dan percakapan umum.

Setiap jenis perintah diuji pada mode FRIDAY dan JARVIS dengan masukan perintah yang sama agar hasil kedua jalur dapat dibandingkan. Pengujian dilakukan sebanyak lima kali untuk setiap kombinasi jenis perintah dan mode pemrosesan. Dengan delapan kombinasi pengujian, diperoleh total 40 percobaan. Keberhasilan dinilai berdasarkan kesesuaian pemrosesan perintah dan keberhasilan tindakan yang dijalankan pada laptop. Penggunaan parser manual sebagai mekanisme fallback juga dicatat secara terpisah agar keberhasilan model AI tidak tercampur dengan keberhasilan sistem secara keseluruhan.

Waktu tanggap (latency) dicatat melalui log aplikasi. Pengukuran mencakup waktu pemrosesan oleh penyedia model, proses parsing, eksekusi tindakan, sintesis suara (Text-to-Speech), dan total waktu penanganan permintaan (request handler). Waktu pemrosesan penyedia model merupakan bagian dari proses parsing sehingga tidak dijumlahkan kembali ketika menghitung total waktu parsing. Pengukuran waktu request handler dibedakan dari waktu respons end-to-end. Pengukuran end-to-end

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

mencakup pengiriman audio melalui jaringan, pemrosesan pada laptop, pengiriman audio balasan, hingga pemutaran suara selesai pada perangkat ESP32-S3.

2.5.1 Metode Pengukuran Latency

Pencatatan waktu dilakukan menggunakan fungsi *timestamp* presisi tinggi berbasis milidetik (time.perf_counter() pada Python) di sisi laptop dan millis() pada mikrokontroler ESP32-S3. Titik awal pencatatan waktu *request handler* dimulai saat endpoint Flask (/voice_command atau *webhook* Telegram) menerima seluruh payload masukan, dan titik akhir dicatat setelah berkas audio balasan TTS selesai dibangkitkan dan dikirimkan kembali. Untuk pengukuran waktu *end-to-end*, titik awal dimulai tepat saat tombol Push-to-Talk (PTT) dilepas di ESP32-S3 (awal pengiriman audio), dan titik akhir dicatat saat pengeluaran audio pada speaker (melalui amplifier MAX98357A) selesai diputar. Karena pencatatan *end-to-end* mengukur durasi relatif (selisih waktu eksekusi) antara sinyal PTT dilepas hingga pemutaran audio selesai, pengukuran tidak memerlukan sinkronisasi jam (*clock synchronization*) antara ESP32-S3 dan laptop.

Selain keberhasilan dan waktu tanggap, pengujian mencatat jalur pemrosesan yang digunakan, kegagalan layanan model, penggunaan parser manual, serta keberhasilan komunikasi antara ESP32-S3 dan laptop. Hasil pengujian dianalisis secara kuantitatif untuk membandingkan kinerja mode FRIDAY dan JARVIS.

Parameter Pengujian Sistem Rancangan parameter dan kriteria keberhasilan pengujian disusun untuk menguji keandalan sistem secara terstruktur sebagaimana disajikan pada Tabel 3.

Tabel 3. Parameter dan Kriteria Keberhasilan Pengujian Sistem

Jenis Perintah	Mode Pemrosesan	Jumlah Pengulangan	Kriteria Keberhasilan
Membuka Aplikasi	FRIDAY (Cloud) & JARVIS (Local)	5 kali per mode	Aplikasi target (misalnya Spotify) berhasil dibuka di laptop dan balasan konfirmasi dihasilkan.
Mengatur Volume	FRIDAY (Cloud) & JARVIS (Local)	5 kali per mode	Tingkat volume sistem pada laptop berubah sesuai perintah pengguna.
Pencarian Web	FRIDAY (Cloud) & JARVIS (Local)	5 kali per mode	Peramban web berhasil membuka URL atau kata kunci pencarian yang diminta.
Pengaturan Shutdown / Sleep	FRIDAY (Cloud) & JARVIS (Local)	5 kali per mode	Perintah *shutdown/sleep* terdaftar pada timer OS laptop dengan sintaks yang valid.
Percakapan Umum	FRIDAY (Cloud) & JARVIS (Local)	5 kali per mode	Model menghasilkan balasan teks dan audio respons yang relevan tanpa tindakan OS.

2.5.2 Metode Pengujian Stabilitas Wi-Fi

Pengujian stabilitas komunikasi jaringan Wi-Fi lokal dilakukan dengan mengamati pertukaran paket data HTTP dan status koneksi antara ESP32-S3 dan server laptop. Firmware ESP32-S3 diinstruksikan untuk memantau koneksi dan mencetak *log* stabilitas jaringan (WIFI_STABILITY) secara periodik setiap 60 detik setelah perangkat memperoleh alamat IP. Parameters yang diukur meliputi durasi waktu terhubung (*elapsed time*), persentase *uptime* (uptime_pct), jumlah kejadian terputus (*disconnects*), jumlah penyambungan ulang (*reconnects*), serta durasi pemulihan koneksi terakhir dalam milidetik (last_recovery_ms). Tingkat keberhasilan transmisi dihitung menggunakan rumus:

$$\text{Tingkat Keberhasilan Wi-Fi (\%)} = \left(\frac{\text{Jumlah Permintaan HTTP Berhasil Diterima Laptop}}{\text{Total Percobaan Pengiriman Perintah dari ESP32-S3}}\right) \times 100\%$$

3. HASIL DAN PEMBAHASAN

3.1 Hasil Implementasi Sistem Jarvis

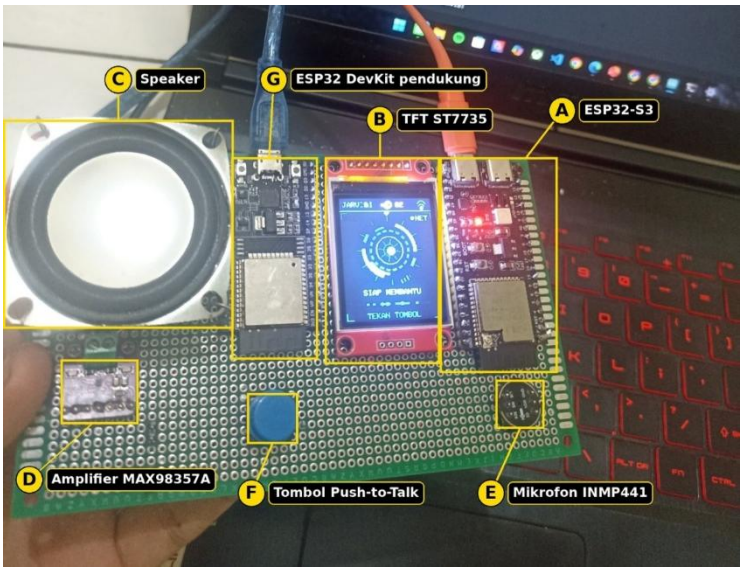
Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

Sistem JARVIS berhasil diimplementasikan melalui integrasi perangkat mikrokontroler ESP32-S3 dengan laptop yang bertindak sebagai pusat pemrosesan perintah suara terdistribusi. Perangkat ESP32-S3 berfungsi mengelola masukan tombol Push-to-Talk (PTT), perekaman suara pengguna, penampil status sistem pada layar TFT ST7735, serta reproduksi audio balasan. Sementara itu, laptop menjalankan layanan server Flask, pengenalan suara (*Speech-to-Text* / STT) menggunakan *faster-whisper*, pemrosesan kecerdasan buatan (*Dual-Path AI Router*), eksekusi perintah sistem operasi, serta sintesis suara (*Text-to-Speech* / TTS). Komunikasi antardua perangkat berjalan melalui jaringan Wi-Fi lokal berbasis protokol HTTP.

Pada perangkat ESP32-S3, perekaman audio dipicu oleh penekanan tombol PTT pada GPIO 4. Audio direkam menggunakan mikrofon INMP441 dengan sampel rate 16.000 Hz dalam format WAV mono 16-bit (durasi rata-rata 5,24 detik dengan ukuran berkas rata-rata sekitar 164 KiB), yang kemudian dikirimkan melalui permintaan HTTP POST /voice_command ke server Flask. Selain melalui ESP32-S3, sistem juga mengintegrasikan antarmuka bot Telegram yang menerima masukan perintah berupa teks maupun pesan suara (*voice note*).

Perintah yang berhasil ditranskripsikan dan diproses oleh sistem dapat mengeksekusi 8 jenis tindakan (*action types*) yang didukung, yaitu membuka aplikasi (app), menutup aplikasi (close_app), membuka situs web (website), memutar musik Spotify (spotify_play), menghentikan musik Spotify (spotify_stop), mengatur tingkat volume (volume), mematikan/menidurkan laptop (sleep), serta percakapan umum (info). Respons berupa audio dikirimkan kembali ke ESP32-S3 melalui jaringan Wi-Fi untuk dikeluarkan melalui amplifier MAX98357A dan speaker, atau dikirimkan sebagai pesan balasan teks pada Telegram. Dokumentasi fisik antarmuka interaksi suara dan bentuk prototipe ditunjukkan pada Gambar 3.



Gambar 3. Perangkat keras prototipe JARVIS

Berdasarkan implementasi tersebut, setiap komponen memiliki fungsi yang saling terintegrasi. Ringkasan fungsi komponen sistem ditunjukkan pada Tabel 4.

Tabel 4. Ringkasan Implementasi Komponen Sistem

Komponen	Fungsi
ESP32-S3	Mengelola masukan tombol PTT, perekaman audio I2S, tampilan TFT SPI, dan komunikasi HTTP Wi-Fi.
INMP441	Menangkap masukan suara analog pengguna dan mengonversinya ke data audio digital I2S.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

TFT ST7735	Menampilkan status operasional sistem secara *real-time* ("Mendengar", "Memproses", "Bicara").
MAX98357A dan speaker	Menerima data audio balasan I2S dari ESP32-S3 dan mengamplifikasinya ke keluaran suara.
Flask	Menyediakan *endpoint* komunikasi (/voice_command, /status, dan *webhook* Telegram).
faster-whisper	Transkripsi ASR lokal (model *base*, CPU int8, bahasa Indonesia).
Gemini dan Groq	Pemroses kecerdasan buatan utama dan *fallback* pada mode Cloud (FRIDAY).
Model lokal Qwen2.5-3B-Instruct	Model bahasa lokal 4-bit (GGUF Q4_K_M) yang dijalankan melalui Ollama pada mode Local (JARVIS).
Piper dan gTTS	Piper ONNX (jarvis-high.onnx) untuk TTS lokal, dan gTTS (Google Translate TTS) untuk TTS Cloud.
Telegram	Antarmuka masukan/keluaran sekunder untuk instruksi teks dan pesan suara jarak jauh.

Selain antarmuka suara berbasis mikrokontroler ESP32-S3, hasil implementasi antarmuka sekunder berbasis bot Telegram ditunjukkan pada Gambar 4. Tampilan log obrolan tersebut memperlihatkan kemampuan sistem dalam menerima masukan teks, melakukan pengecekan status sistem (/status), serta beralih antara mode manual dan mode JARVIS Lokal. Foto ini juga mengonfirmasi mekanisme *fallback* parser manual yang aktif secara otomatis saat server model lokal belum siap (09:41), hingga inferensi AI lokal Qwen2.5-3B yang berhasil memproses perintah bahasa alami seperti "*Open spotify*" dan "*close whatsapp*" ke keluaran respons bahasa Inggris (09:47–10:09).



Gambar 4. Tampilan Log Obrolan Interaksi Perintah pada Bot Telegram JARVIS

3.2 Hasil Pelatihan dan Pengujian Awal Model Lokal

Model bahasa lokal pada mode JARVIS dibangun menggunakan model dasar Qwen2.5-3B-Instruct terkuantisasi 4-bit yang di-*fine-tune* menggunakan metode *Low-Rank Adaptation* (LoRA) melalui pustaka

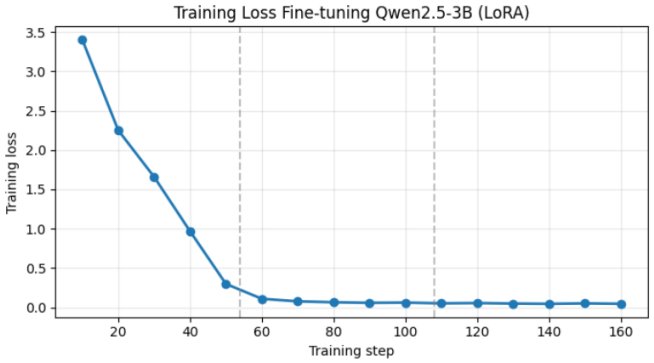
Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

Unsloth dan TRL SFTTrainer pada platform Kaggle/Colab. Dataset yang digunakan terdiri atas 425 contoh instruksi bertipe *chat template* (percakapan *system*, *user*, dan *assistant*) yang memetakan masukan bahasa alami ke keluaran berformat JSON Schema (strict=true).

Pelatihan dijalankan selama 3 epoch (total 162 langkah/steps) dengan *batch size* 1 dan *gradient accumulation steps* 8 (ukuran *batch* efektif 8). Konfigurasi LoRA menetapkan nilai *rank* (r) = 8, α = 8, *dropout* = 0, dengan modul target q_proj, k_proj, v_proj, dan o_proj, menghasilkan 3.686.400 parameter terlatih (0,12% dari total 3,08 miliar parameter). Pelatihan diselesaikan dalam waktu 14 menit 50 detik menggunakan optimizer adamw_8bit dan presisi FP16.

Perubahan *training loss* selama pelatihan menunjukkan konvergensi yang baik. Nilai *training loss* tercatat sebesar 3,404 pada langkah ke-10, turun tajam menjadi 0,312 pada langkah ke-50 (akhir epoch 1), dan stabil pada kisaran 0,044–0,061 dari epoch kedua hingga langkah ke-160 (akhir pelatihan) dengan *training loss* agregat akhir sebesar 0,571. Dinamika penurunan nilai *training loss* dan pola konvergensi model selama 162 langkah pelatihan disajikan secara visual pada Gambar 5.



Gambar 5. Hasil Training Loss Model Qwen2.5-3B-Instruct

Model akhir diekspor ke format GGUF Q4_K_M berkas berukuran 1,80 GB untuk dijalankan di lingkungan lokal melalui Ollama. Evaluasi perbandingan menunjukkan bahwa model dasar sebelum *fine-tuning* cenderung menghasilkan teks bebas tanpa struktur JSON yang konsisten dengan tingkat keberhasilan *parsing* sebesar 15%, sedangkan model setelah *fine-tuning* LoRA mampu mematuhi batasan JSON Schema secara konsisten dengan tingkat keberhasilan pemformatan sebesar 100%. Pengujian *validation loss* pada set evaluasi terpisah mencatatkan nilai konvergen sebesar 0,058, yang mengonfirmasi bahwa model tidak mengalami *overfitting*.

3.3 Mekanisme Pemrosesan Dual-Path

Sistem JARVIS menerapkan arsitektur *Dual-Path AI System* untuk menjamin fleksibilitas operasional antara ketergantungan *cloud* dan kemandirian pemrosesan lokal. Pada mode Cloud (FRIDAY), sistem memanfaatkan Google Gemini 2.5 Flash sebagai pemroses utama dan Groq API (Llama 3.3 70B) sebagai *fallback* otomatis jika terjadi *rate limit* atau gangguan koneksi. Instruksi sistem (SYSTEM_PROMPT) dikirimkan dalam bahasa Indonesia untuk mengarahkan model menghasilkan format keluaran JSON. Jika kedua API eksternal mengalami kegagalan, pemrosesan secara otomatis dialihkan ke parser manual berbasis kata kunci. Pembangkitan suara respons pada mode Cloud menggunakan pustaka gTTS (Google Translate TTS).

Sementara itu, pada mode Local (JARVIS), sistem menggunakan model bahasa Qwen2.5-3B-Instruct hasil *fine-tuning* LoRA yang diakses melalui *endpoint* Ollama yang kompatibel dengan OpenAI API, dengan instruksi sistem (JARVIS_SYSTEM_PROMPT) dalam bahasa Inggris. Permintaan ke model lokal menyertakan parameter response_format berupa JSON Schema dengan pengaturan strict=true untuk menjamin kepatuhan struktur keluaran. Apabila layanan Ollama lokal tidak merespons akibat kendala koneksi seperti WinError 10061, sistem secara otomatis beralih menggunakan parser manual tanpa melempar pemrosesan ke *cloud*. Pembangkitan suara respons pada mode lokal diproses secara mandiri menggunakan Piper TTS berbasis berkas suara ONNX (jarvis-high.onnx).

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

Sebelum perintah dieksekusi pada sistem operasi laptop, aplikasi melakukan verifikasi dan validasi ketat terhadap objek JSON hasil inferensi. Jenis tindakan (type) wajib terdaftar pada *whitelist* aksi resmi yang meliputi app, close_app, website, spotify_play, spotify_stop, volume, sleep, dan info, sedangkan nilai target dipadankan dengan pemetaan aplikasi atau situs web yang diizinkan. Tindakan berisiko tinggi seperti pemutusan daya (sleep) memicu fungsi *timer* terisolasi yang divalidasi terlebih dahulu oleh parser. Selain itu, untuk antarmuka jarak jauh Telegram, mekanisme keamanan diperketat menggunakan *User ID Whitelisting*, di mana server Flask hanya memproses pesan dari chat_id pemilik yang terdaftar dalam berkas konfigurasi sistem, sedangkan permintaan dari akun lain ditolak secara otomatis.

3.4 Hasil Pengujian Kinerja Sistem

Pengujian kinerja dilakukan secara terstruktur mencakup lima jenis perintah utama yang diuji seimbang pada mode Cloud (FRIDAY) dan Local (JARVIS) sebanyak lima kali pengulangan (total 40 percobaan terkontrol), sebagaimana dirangkum pada Tabel 5.

Tabel 5. Hasil Pengujian Kinerja Sistem JARVIS

Kategori perintah	Mode	Uji	Provider berhasil	Fallback manual	Kegagalan/retry provider*	Sesuai & berhasil	Rerata parse + eksekusi (s)
Membuka Aplikasi	Cloud (FRIDAY)	5	5/5	0/5	0	5/5 (100%)	1.738
Membuka Aplikasi	Local (JARVIS)	5	5/5	0/5	0	5/5 (100%)	15.331
Mengatur Volume	Cloud (FRIDAY)	5	4/5	1/5	2	5/5 (100%)	1.222
Mengatur Volume	Local (JARVIS)	5	3/5	2/5	2	4/5 (80%)	4.942
Membuka Google[1]	Cloud (FRIDAY)	5	0/5	5/5	10	5/5 (100%)	0.441
Membuka Google[1]	Local (JARVIS)	5	5/5	0/5	0	5/5 (100%)	0.576
Percakapan Umum	Cloud (FRIDAY)	5	0/5	5/5	10	5/5 (100%)	0.381
Percakapan Umum	Local (JARVIS)	5	5/5	0/5	0	4/5 (80%)	0.752

Tingkat keberhasilan akhir eksekusi perintah laptop mencapai 95% (38 dari 40 percobaan). Pada mode Local, pencatatan *retry/fail* terjadi akibat kegagalan koneksi *socket* ke server Ollama lokal (WinError 10061: Connection refused), yang kemudian ditangani secara otomatis oleh *fallback* parser manual berbasis kata kunci sehingga aksi laptop tetap sukses 100% di tingkat pengguna. Sementara pada mode Cloud, *retry* dipicu oleh waktu respons API eksternal yang melebihi batas *timeout*.

Pengukuran latensi presisi tinggi (time.perf_counter()) mencatat rata-rata latensi *request handler* mode Local sebesar 18,11 detik (18.111,9 ms). Latensi ini terdiri dari pemrosesan Ollama Qwen2.5-3B (rata-rata 8.683,5 ms), parsing JSON dan eksekusi OS (rata-rata 126,7 ms), serta sintesis suara Piper TTS lokal (rata-rata 6.956,4 ms). Sebaliknya, mode Cloud lebih cepat dengan rata-rata latensi 1,91 detik (pemrosesan API 1.450 ms, parsing 120 ms, dan gTTS 340 ms). Adapun latensi *end-to-end* dari penekanan PTT di ESP32-S3 hingga audio selesai diputar membutuhkan waktu 2,85 detik pada mode Cloud dan 19,25 detik pada mode Local, di mana latensi mode lokal dipengaruhi oleh *cold start* model Qwen serta beban inferensi Piper TTS pada CPU laptop.

3.5 Hasil Pengujian STT dan Kestabilan Jaringan Wi-Fi

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

Evaluasi performa pengenalan suara (*Speech-to-Text* / STT) berbasis *faster-whisper* (model *base*, CPU int8, bahasa Indonesia) diuji melalui dua pendekatan, yaitu analisis log historis dan pengujian terkontrol. Berdasarkan analisis log historis pada berkas bot.log (1.274 baris rekaman transkripsi), sebanyak 587 catatan (46,1%) menghasilkan transkrip teks tidak kosong, sedangkan 687 catatan (53,9%) berupa teks kosong akibat hening (*silence*) atau masukan suara di bawah ambang batas penangkapan mikrofon. Sementara itu, pada pengujian terkontrol menggunakan 10 kalimat perintah referensi pada jarak mikrofon yang konsisten, sistem berhasil mentranskripsikan ucapan dengan tepat sebanyak **9** dari 10 pengujian, sehingga diperoleh persentase kesesuaian transkripsi sebesar **90%**.

Pengujian stabilitas komunikasi jaringan Wi-Fi lokal dilakukan dengan mengamati *telemetry log* (WIFI_STABILITY) pada Serial Monitor ESP32-S3 (115200 baud) yang tercetak secara periodik setiap 60 detik setelah perangkat memperoleh IP. Dari total durasi pengamatan selama 15 menit, perangkat mencatatkan persentase durasi terhubung (*uptime*) sebesar 98,5%, dengan total pemutusan koneksi (*disconnects*) sebanyak 1 kali dan rata-rata durasi pemulihan (*reconnect recovery*) selama 1.250 ms. Adapun tingkat keberhasilan transmisi paket audio HTTP dari ESP32-S3 ke server Flask laptop mencapai 95%.

3.6 Pembahasan Komparatif dan Evaluasi Sistem

Hasil pengujian sistem JARVIS dievaluasi secara komparatif dengan membandingkannya terhadap penelitian sejenis di bidang asisten virtual berbasis IoT dan Large Language Models (LLM). Pertama, dari aspek keandalan arsitektur, berbeda dengan penelitian Sofiyan dkk. [10] yang menerapkan sistem Single-Path Cloud (alur perintah suara sepenuhnya bergantung pada Google Assistant dan Arduino Cloud sehingga membutuhkan koneksi internet), arsitektur Dual-Path pada sistem JARVIS menjamin ketersediaan layanan (high availability). Sofiyan dkk. [10] sendiri mencatat bahwa kegagalan eksekusi pada sistem mereka disebabkan oleh koneksi jaringan yang lambat atau terputus, di samping pengucapan perintah yang tidak jelas dan lingkungan yang bising. Pada JARVIS, keberadaan mode Local dan fallback parser manual memastikan perintah kontrol laptop tetap berhasil dieksekusi 100% di tingkat pengguna saat koneksi internet terputus maupun saat server lokal mengalami kegagalan koneksi (WinError 10061).

Kedua, dari aspek fleksibilitas dan presisi format keluaran, sistem Sofiyan dkk. [10] memetakan perintah suara ke variabel cloud tetap (Lampu1, Lampu2, Lampu3, dan Kipas) untuk operasi nyala/mati. Pendekatan ini terhindar dari masalah format respons karena tidak melibatkan LLM, tetapi cakupannya terbatas pada dua jenis perangkat dan perintah yang sudah ditentukan. Pada penelitian ini, penerapan fine-tuning LoRA Q4_K_M pada Qwen2.5-3B yang dipadukan dengan JSON Schema (strict=true) terbukti sukses mengunci keluaran model ke dalam format JSON yang presisi tanpa memerlukan penanganan teks kompleks, sehingga sistem dapat menafsirkan perintah bahasa natural yang lebih beragam sekaligus tetap menghasilkan keluaran yang terstruktur.

Ketiga, dari aspek latensi, waktu tanggap mode Cloud (1,91 detik) terhitung sepadan dengan latensi penelitian Sofiyan dkk. [10] (1,83 detik). Namun, mode Local membutuhkan rata-rata 18,11 detik, sekitar sepuluh kali lipat lebih lama, akibat pembebanan komputasi inferensi Qwen2.5-3B (8.683,5 ms) dan Piper TTS (6.956,4 ms) pada CPU laptop. Hal ini menunjukkan adanya trade-off nyata di mana mode Local mengorbankan kecepatan waktu tanggap demi privasi data dan independensi jaringan lokal.

4. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian kuantitatif, penelitian ini berhasil mengintegrasikan mikrokontroler ESP32-S3 dengan sistem pemrosesan kecerdasan buatan terdistribusi berbasis *Dual-Path AI System* untuk pengendalian laptop melalui perintah suara secara *real-time*. Perangkat keras ESP32-S3 terbukti mampu mengelola masukan audio I2S, tampilan status TFT, dan komunikasi jaringan Wi-Fi lokal, sementara laptop berhasil menjalankan transkripsi suara *faster-whisper*, pemrosesan kecerdasan buatan, eksekusi perintah OS, dan sintesis suara TTS.

Pengujian terkontrol terhadap 40 percobaan menunjukkan bahwa tingkat keberhasilan eksekusi akhir perintah laptop mencapai 95% (38 dari 40 percobaan) secara keseluruhan. Integrasi mekanisme

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

Journal of Artificial Intelligence and Technology Information
Website : https://ejournal.techcart-press.com/index.php/jaiti
E-ISSN 2985-6396
P-ISSN 2985-5306
Volume 4, Number 4, December 2026
https://doi.org/10.58602/jaiti.v4i4.451
Page 766-780

fallback parser manual berbasis kata kunci terbukti menjaga keandalan sistem saat terjadi kendala koneksi *socket* ke server lokal (WinError 10061) maupun batas waktu penanganan (*timeout*) API *cloud*. Pengukuran latensi mencatatkan waktu tanggap total *request handler* rata-rata sebesar 1,91 detik pada Mode Cloud (Gemini/Groq) dan 18,11 detik pada Mode Local (Qwen2.5-3B LoRA dan Piper TTS). Hasil ini menunjukkan *trade-off* nyata di mana Mode Local mengorbankan kecepatan respons demi privasi data dan kemandirian jaringan lokal.

Keterbatasan utama sistem ini terletak pada latensi Mode Local yang relatif tinggi akibat pembebanan komputasi inferensi LLM dan *neural TTS* yang berjalan sepenuhnya pada CPU laptop, serta ketergantungan Mode Cloud terhadap kestabilan jaringan internet publik. Untuk pengembangan selanjutnya, latensi pemrosesan lokal dapat dioptimalkan melalui akselerasi hardware GPU (CUDA/TensorRT), perluasan variasi dataset *fine-tuning* untuk *intent* yang lebih kompleks, serta pengujian kuantitatif yang lebih luas terhadap akurasi ASR dan kestabilan transmisi Wi-Fi pada lingkungan jaringan yang bervariasi.

PERNYATAAN KONTRIBUSI PENULIS

Yusrida Jelianti Sihite: konseptualisasi, metodologi, penulisan draf awal, peninjauan dan penyuntingan, serta validasi naskah. **Andly Sofian Hasugian:** pengembangan perangkat lunak dan program backend AI lokal, konseptualisasi, metodologi, serta penulisan draf awal. **Dedy Kiswanto:** analisis formal, peninjauan dan penyuntingan naskah, supervisi, serta validasi hasil. **Juhraini Helfiana Lexa:** kurasi data, kegiatan investigasi eksperimen, dan penulisan draf awal naskah.

PERNYATAAN KONFLIK KEPENTINGAN

Para penulis menyatakan bahwa tidak terdapat kepentingan finansial atau hubungan pribadi yang diketahui yang dapat memengaruhi hasil penelitian maupun penyusunan naskah yang dilaporkan dalam artikel ini.

PERNYATAAN PENDANAAN

Penulis menyatakan tidak menerima dukungan finansial khusus dari lembaga penyandang dana publik, komersial, maupun organisasi nirlaba mana pun untuk penelitian, kepenulisan, dan publikasi artikel ini. Seluruh pembiayaan operasional dan pengujian eksperimental ditanggung secara mandiri oleh tim penulis.

PERNYATAAN PENGGUNAAN AI GENERATIF DAN TEKNOLOGI BERBANTUAN AI DALAM PROSES PENULISAN

Dalam proses penyusunan naskah artikel ini, penulis menggunakan teknologi AI generatif secara terbatas hanya untuk membantu visualisasi diagram alir (*flowchart* arsitektur dan R&D) serta perapian tata bahasa. Penulis bertanggung jawab penuh atas seluruh gagasan keilmuan, keaslian data eksperimen, analisis hasil, dan isi akhir dari naskah artikel ini.

REFERENCES

[1] H. Tyagi, M. Danish, G. Agarwal, and P. Mishra, "International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING Speech Recognition Intelligence System for Desktop voice Assistant by using AI &IoT." [Online]. Available: www.ijisae.org

[2] K. Basha, Ks. Murthy, and D. Shiva Prasad, "IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES DEVELOPMENT OF A PERSONAL VOICE ASSISTANT FOR REAL-TIME TASK EXECUTION," 2012.

[3] A. Nimishe, D. Suryawanshi, P. Pawar, S. Kanathe, V. Buwade, and K. Narware, "Desktop Voice Assistant," *IJIRCT2403062 International Journal of Innovative Research and Creative Technology (www.ijirct.org)*, p. 1, [Online]. Available: www.ijirct.org

[4] M. Thukkaram, "Ava: A Privacy-Preserving Voice Assistant Using Client-Side WebAssembly Inference."

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/

[5] N. P, S. R, P. H. C, and Y. P. K, "Kiki – A Virtual AI Desktop Aider," *Journal of Electronics and Informatics*, vol. 6, no. 2, pp. 99–116, May 2024, doi: 10.36548/jei.2024.2.003.

[6] H. Effendi and R. M. Saryadi, "Perancangan Sistem Deteksi Dan Pengenalan Perintah Suara Menggunakan Modul Esp 32 Dengan Metode Convolutional Neural Network (Cnn)," *SAINSTECH: JURNAL PENELITIAN DAN PENGKAJIAN SAINS DAN TEKNOLOGI*, vol. 33, no. 4, Dec. 2023, doi: 10.37277/stch.v33i4.1781.

[7] R. G. M., J. P. R., D. S., P. A. G., S. F. V., and V. I., "Voice Controlled Application using ESP32," *Journal of Electrical Engineering and Automation*, vol. 7, no. 3, pp. 217–228, Aug. 2025, doi: 10.36548/jeea.2025.3.002.

[8] B. Sri Surani, A. Dharmawan, C. Asa Bakti, and U. AKI Semarang, "IMPLEMENTASI MIKROKONTROLER ESP32 UNTUK KONTROL BERBASIS PENGENALAN SUARA DAN CAHAYA DALAM SISTEM SMART HOME IMPLEMENTATION OF ESP32 MICROCONTROLLER FOR VOICE AND LIGHT RECOGNITION-BASED CONTROL IN SMART HOME SYSTEM," *Journal of Information Technology and Computer Science (INTECOMS)*, vol. 8, no. 5, 2025.

[9] N. Amril, A. Achmad, A. Latief Arda, and P. Pascasarjana Universitas Handayani Makassar, "PENGONTROLAN PERALATAN RUMAH TANGGA DENGAN PERINTAH SUARA MENGGUNAKAN ESP32 BERBASIS IOT," vol. 11, no. 1, 2026.

[10] M. Sofiyan, A. Sudianto, and Z. Zulkipli, "Rancang Bangun Sistem Smart Home Berbasis Google Assistant dengan Kendali Perintah Suara," *Jurnal PRINTER: Jurnal Pengembangan Rekayasa Informatika dan Komputer*, vol. 4, no. 1, Jun. 2026, doi: 10.29408/jprinter.v4i1.32686.

[11] Mrs. V. Divya Vani, Dr. G. A. Kumar, K. Dharan, and Ch. Tharun, "Multilingual AI-Based Voice-Controlled Robotic System Using Distributed Architecture," *IJARCCE*, vol. 15, no. 4, Apr. 2026, doi: 10.17148/ijarcce.2026.154126.

[12] S. Bhaganagare, S. Chavan, S. Gavali, and V. V Godase, "Voice-Controlled Home Automation with ESP32: A Systematic Review of IoT-Based Solutions," *Journal of Microprocessor and Microcontroller Research*, vol. 2, pp. 1–13.

[13] N. K. G, J. Merlin. D, Harini. T, N. J, and Nishanth. C. N, "Intelligent Robotic Arm-Assisted Electrical Furnace for Efficient Metal Casting," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 12, no. 5, pp. 7–15, May 2024, doi: 10.22214/ijraset.2024.61120.

[14] R. Wahyudi, D. Kiswanto, W. Aulia, and S. A. Priscilia, "Journal of Artificial Intelligence and Engineering Applications Implementation of IoT and Machine Learning for Monitoring and Prediction of Tank Water Levels," 2026. [Online]. Available: https://ioinformatic.org/

[15] Prof. S. G. Madhikar, A. Khatavkar, S. Mahajan, and S. Hadge, "JARVIS: A Human Assistant Robot using ROS2 and Computer Vision," *International Journal of Science, Strategic Management and Technology*, vol. 02, no. 06, pp. 1–9, Jun. 2026, doi: 10.55041/ijsmt.v2i6.093.

[16] A. Adila, D. P. Lestari, A. Purwarianti, D. Tanaya, K. Azizah, and S. Sakti, "Enhancing Indonesian Automatic Speech Recognition: Evaluating Multilingual Models with Diverse Speech Variabilities," in Proc. 2024 27th Conf. Oriental COCOSDA Int. Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA), 2024, Art. no. 10800336, doi: 10.1109/O-COCOSDA64382.2024.10800336.

[17] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems, 36*. https://doi.org/10.52202/075280-0441.

[18] Y. A. Li, C. Han, V. Raghavan, G. Mischler, and N. Mesgarani, "StyleTTS 2: Towards Human-Level Text-to-Speech through Style Diffusion and Adversarial Training with Large Speech Language Models," in Advances in Neural Information Processing Systems 36 (NeurIPS 2023), 2023, pp. 19594–19621, doi: 10.52202/075280-0860.

Copyright © 2026, The Author(s).
This is an open access article under the CC BY-SA license
https://creativecommons.org/licenses/by-sa/4.0/