Prototipe Asisten Suara untuk Informasi Keseharian dan Kendali *Smart Room* Berbasis Mikrokontroler ESP32

Irsyad Alief Pratama; Umi Fadillah,
Program Studi Teknik Elektro, Fakultas Teknik,
Universitas Muhammadiyah Surakarta

Abstrak

Penelitian ini dilatarbelakangi oleh perkembangan teknologi *Internet of Things (IoT)* yang mendorong terciptanya sistem otomatis untuk meningkatkan kenyamanan dan efisiensi aktivitas manusia, salah satunya melalui penerapan *smart room* dengan kontrol berbasis suara yang lebih alami dan praktis dibandingkan saklar konvensional. Berbagai penelitian *smart home* berbasis ESP32 yang menunjukkan bahwa telah mampu mengendalikan perangkat seperti lampu dan kipas menggunakan perintah suara, namun sebagian besar masih terbatas pada fungsi *ON/OFF* dan belum berperan sebagai asisten informasi yang interaktif. Selain itu, pengembangan *voice assistant* pada *embedded system* masih terbatas dan umumnya belum terintegrasi secara menyeluruh dengan sistem *smart home*. Oleh karena itu, penelitian ini bertujuan merancang dan mengimplementasikan sistem asisten suara kendali *smart room* berbasis ESP32 yang mampu menerima, memproses, dan merespons perintah suara pengguna secara andal, serta terintegrasi dengan sistem smart room. Sistem menggunakan *Python* yang diprogram di laptop sebagai output suara untuk mendukung interaksi dua arah. Fungsi sistem meliputi pemberian informasi keseharian seperti waktu dan tanggal, percakapan sederhana, serta pengendalian perangkat smart home seperti lampu, kipas, dan pintu. Penelitian dibatasi pada metode pengenalan suara berbasis *rule-based* menggunakan logika *if–else*. Metode penelitian menggunakan pendekatan rekayasa eksperimental melalui tahapan studi literatur, perancangan perangkat keras dan perangkat lunak, implementasi sistem, serta pengujian kinerja untuk memastikan sistem mampu bekerja sesuai fungsi yang dirancang sekaligus menghasilkan prototipe *smart room* berbasis *voice assistant* yang hemat daya dan ekonomis. Hasil pengujian menunjukkan tingkat keberhasilan pengenalan perintah suara sebesar 92,5% pada program *Python* dan 95% pada antarmuka web, dengan rata-rata waktu respon masing-masing 4,96 detik dan 2,60 detik, sehingga sistem dapat dikatakan layak digunakan sebagai solusi kendali *smart room* berbasis suara yang responsif.

Kata Kunci: *Embedded System, ESP32, IoT, Smart Room, Voice Control.*

Abstract

This research is motivated by the rapid development of Internet of Things (IoT) technology and artificial intelligence, which has driven the creation of automated systems to improve human comfort and efficiency, particularly through the implementation of smart rooms with voice-based control that are more natural and practical than conventional switches. Previous studies on ESP32-based smart home systems have demonstrated the ability to control devices such as lights and fans using voice commands; however, most are still limited to ON/OFF functions and have not yet functioned as interactive information assistants, while voice assistant development on embedded systems remains limited and not fully integrated with smart home systems. Therefore, this research aims to design and implement an ESP32-based voice assistant system for smart room control that can reliably receive, process, and respond to user voice commands while being fully integrated with the smart room system. The system uses Python programmed on a laptop as voice output to support two-way interaction, with functions including providing daily information such as time and date, simple conversations, and controlling smart home devices such as lights, fans, and doors. This study is limited to a rule-based voice recognition method using if–else logic and employs an experimental engineering approach consisting of literature review, hardware and

software design, system implementation, and performance testing to ensure the system operates as intended while producing an energy-efficient and cost-effective smart room prototype based on a voice assistant. The test results show a success rate of voice command recognition of 92.5% in the Python program and 95% in the web interface, with an average response time of 4.96 seconds and 2.60 seconds respectively, so the system can be said to be suitable for use as a responsive voice-based smart room control solution.

***Keywords**: Embedded System, ESP32, IoT, Smart Room, Voice Control.*

1. PENDAHULUAN

Perkembangan teknologi *Internet of Things* (IoT) dalam beberapa tahun terakhir telah membuka banyak peluang untuk menciptakan sistem yang dapat memudahkan aktivitas manusia sehari-hari. Salah satu wujud nyata dari perkembangan tersebut adalah sistem *smart home*, yakni sistem yang memungkinkan berbagai perangkat dalam rumah untuk dikendalikan secara otomatis dan terintegrasi melalui jaringan internet (Hanani & Hariyadi, 2020). Konsep ini kemudian berkembang ke cakupan yang lebih kecil dan spesifik, yaitu *smart room*. Berbeda dengan *smart home* yang mencakup seluruh rumah, *smart room* hanya berfokus pada satu ruangan tertentu sehingga lebih sederhana, hemat daya, dan lebih mudah diimplementasikan sebagai prototipe berbasis *embedded system*.

Salah satu cara untuk berinteraksi dengan sistem *smart room* adalah melalui perintah suara atau *voice control*. Pendekatan ini dinilai lebih praktis dan natural dibandingkan cara konvensional seperti menekan tombol atau saklar, karena pengguna cukup berbicara untuk mengoperasikan perangkat tanpa harus menyentuh apapun (Prasetyana & Fadlilah, 2021), Kemudahan inilah yang membuat pengendalian berbasis suara semakin banyak dilirik untuk diterapkan pada sistem *smart room*, terutama untuk mengendalikan perangkat seperti lampu, kipas angin, maupun sistem keamanan pintu. Hal ini juga dibuktikan oleh (Nur Afiyat & Hakim, 2021) yang mengembangkan *prototype* sistem pengendalian perangkat elektronik berbasis IoT menggunakan *voice control* dan Blynk, di mana sistem tersebut berhasil mengendalikan perangkat melalui perintah suara namun masih terbatas pada fungsi kendali ON/OFF tanpa adanya kemampuan sistem untuk merespons atau memberikan informasi balik kepada pengguna.

Beberapa penelitian sebelumnya telah membuktikan bahwa sistem kendali berbasis suara menggunakan mikrokontroler ESP32 mampu bekerja dengan baik untuk mengontrol perangkat secara *real-time*. Namun, sebagian besar sistem yang dikembangkan masih sebatas menjalankan perintah ON/OFF sederhana dan belum dikembangkan lebih jauh menjadi asisten suara yang bisa berinteraksi secara dua arah dengan pengguna (Surani et al., 2025). Beberapa penelitian lain memang sudah mencoba menambahkan fitur sensor lingkungan dan integrasi *cloud*, tetapi

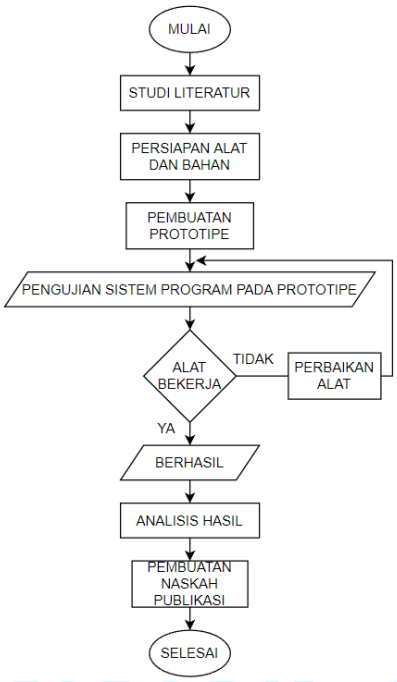
interaksi suaranya masih satu arah dan belum mampu memberikan informasi seperti waktu, tanggal, maupun pengingat aktivitas harian (Bhaganagare et al., 2025a).

Penelitian yang berfokus pada *voice assistant* juga sudah cukup banyak dikembangkan, misalnya untuk sistem keamanan pintu, *wearable device*, hingga aplikasi berbasis kecerdasan buatan. Akan tetapi, sistem-sistem tersebut umumnya masih berdiri sendiri dan belum terintegrasi langsung dengan kendali perangkat dalam satu ruangan (Kurniadi & Setiawan, 2024). Selain itu, membangun *voice assistant* pada *embedded system* yang bisa sekaligus mengendalikan perangkat dan memberikan informasi bukan hal yang mudah karena membutuhkan integrasi antara pemrosesan suara, komunikasi data, dan kendali *hardware* secara bersamaan. Pendekatan menggunakan arsitektur *hybrid* yang menggabungkan Python untuk pemrosesan suara, ESP32 sebagai pengendali perangkat, dan Blynk sebagai platform cloud sebenarnya menawarkan solusi yang cukup menjanjikan, namun sistem dengan arsitektur seperti ini yang sekaligus berfungsi sebagai asisten informasi keseharian masih jarang ditemukan dalam penelitian sebelumnya (Bhaganagare et al., 2025b; Cruz-Morales, 2024).

Berdasarkan kajian terhadap penelitian-penelitian terdahulu tersebut, masih ada celah yang cukup jelas dalam penelitian di bidang ini. Sebagian besar penelitian hanya berfokus pada kendali perangkat atau voice assistant secara terpisah, dan belum mengintegrasikan keduanya dalam satu sistem *smart room* yang mampu memberikan informasi keseharian seperti waktu, tanggal, serta aktivitas harian pengguna melalui interaksi suara yang natural dan dua arah. Oleh karena itu, diperlukan pengembangan prototipe asisten suara dengan sistem *smart room* yang berbasis mikrokontroler ESP32 yang tidak hanya berfungsi sebagai pengendali perangkat ruangan, tetapi juga sebagai penyedia informasi keseharian secara interaktif.

2. METODE

Penelitian ini bertujuan mengembangkan prototipe sistem asisten suara yang terintegrasi dengan kendali perangkat *smart room* berbasis mikrokontroler ESP32. Sistem dirancang menggunakan arsitektur *hybrid* yang menggabungkan pemrosesan suara berbasis Python pada komputer, mikrokontroler ESP32 sebagai pengendali perangkat keras, serta platform *cloud* Blynk sebagai jembatan komunikasi antara perangkat lunak dan perangkat keras. Metode yang digunakan dalam penelitian ini meliputi perancangan sistem, implementasi perangkat keras dan perangkat lunak, serta pengujian sistem secara menyeluruh. Gambar 1 adalah alur serta penjelasan mengenai tahapan penelitian yang akan dilakukan.



Gambar 1. Diagram Alur Penelitian

a. Studi Literatur

Pada tahap ini dilakukan pengumpulan referensi dari jurnal, buku, dan sumber ilmiah lainnya yang berkaitan dengan *Internet of Things* (IoT), *voice assistant*, ESP32, *smart room*, serta sistem kendali berbasis suara. Studi literatur bertujuan untuk memahami konsep kerja sistem, mengetahui metode yang digunakan pada penelitian sebelumnya, serta menentukan komponen dan teknologi yang akan digunakan.

b. Persiapan alat dan bahan

Pada tahap ini dilakukan identifikasi, pengumpulan, dan persiapan seluruh komponen yang dibutuhkan dalam pembuatan sistem. Komponen *hardware* yang dipersiapkan meliputi mikrokontroler ESP32 sebagai pusat kendali, modul relay sebagai saklar elektronik untuk mengontrol lampu, kipas, dan kunci pintu, dengan sumber tegangan sistem 220V. Selain itu dipersiapkan juga perangkat komputer atau laptop yang akan digunakan untuk menjalankan program *voice assistant* berbasis Python. Dari sisi *software*, dipersiapkan Arduino IDE untuk pemrograman ESP32, *Visual Studio Code* untuk pemrograman Python dan HTML, *library* pendukung seperti *SpeechRecognition*, gTTS, *pygame*, dan *requests*, serta aplikasi Blynk sebagai platform komunikasi antara sistem *voice assistant* dan ESP32.

c. Pembuatan Prototipe

Pada tahap ini dilakukan perakitan rangkaian hardware sesuai dengan skema yang telah dirancang, yaitu menghubungkan ESP32 dengan modul relay yang berfungsi untuk mengontrol beban listrik seperti lampu, kipas, dan kunci pintu. Selain itu dilakukan pembuatan program pada ESP32 menggunakan Arduino IDE agar ESP32 dapat terhubung dengan jaringan WiFi dan menerima perintah dari server *Blynk*. Selanjutnya dibuat program *voice assistant* menggunakan *Python* yang berjalan pada laptop untuk menerima input suara pengguna, mengubahnya menjadi teks menggunakan *Google Speech Recognition*, memproses perintah menggunakan logika program, dan mengirimkan perintah ke ESP32 melalui platform Blynk. Sebagai opsi kontrol tambahan, dikembangkan pula antarmuka web berbasis HTML, CSS, dan JavaScript yang dapat diakses melalui *browser* di perangkat apapun termasuk *smartphone*. Antarmuka web ini menyediakan dua cara kontrol sekaligus, yaitu melalui tombol manual yang dapat disentuh langsung di layar HP, serta melalui perintah suara menggunakan fitur *Web Speech API* yang tersedia di *browser Google Chrome*. Dengan adanya antarmuka web ini, pengguna tidak harus selalu berada di depan laptop untuk mengoperasikan sistem, cukup membuka halaman web melalui HP dan sistem dapat dikendalikan dari mana saja selama terhubung ke internet.

d. Pengujian sistem program pada prototipe

Setelah prototipe selesai dibuat, tahap selanjutnya adalah melakukan pengujian sistem secara menyeluruh. Pengujian dilakukan untuk memastikan bahwa setiap bagian sistem dapat bekerja sesuai dengan fungsi yang diharapkan. Pengujian meliputi pengujian koneksi ESP32 dengan jaringan WiFi, pengujian komunikasi antara *Python* dan *Blynk*, pengujian respons sistem terhadap perintah suara pengguna baik melalui Python maupun antarmuka web, serta pengujian kemampuan sistem dalam mengontrol lampu, kipas, dan kunci pintu. Pengujian juga dilakukan terhadap tombol kontrol manual pada antarmuka web untuk memastikan fungsinya berjalan dengan baik ketika diakses melalui *smartphone*. Selain itu dilakukan juga pengujian stabilitas sistem dan waktu respons terhadap perintah yang diberikan.

e. Analisis hasil

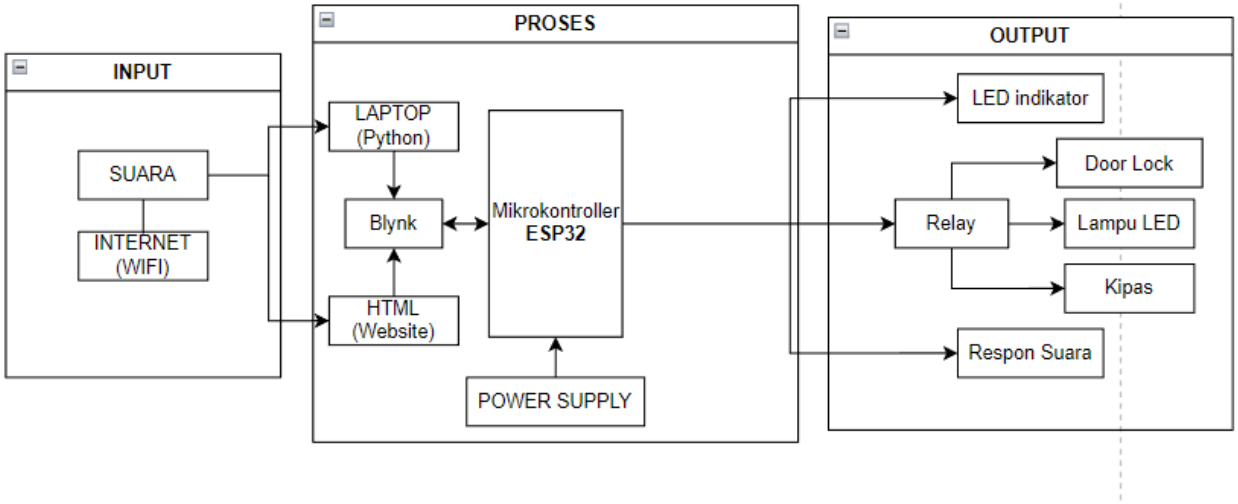
Pada tahap ini dilakukan evaluasi terhadap hasil pengujian untuk memastikan apakah sistem sudah dapat bekerja secara fungsional. Parameter yang diperiksa meliputi apakah ESP32 dapat terhubung ke jaringan, apakah relay dapat aktif dan nonaktif sesuai perintah, apakah sistem *voice assistant* dapat mengirimkan perintah dengan benar, serta apakah antarmuka web dapat diakses dan dioperasikan dengan baik melalui *smartphone*. Jika sistem tidak bekerja, maka dilakukan

identifikasi penyebab masalah, baik dari sisi hardware seperti kesalahan rangkaian, maupun dari sisi *software* seperti kesalahan program. Setelah perbaikan dilakukan, sistem kembali diuji hingga dapat bekerja secara fungsional.

f. Pembuatan naskah publikasi

Laporan berisi latar belakang penelitian, landasan teori, metode penelitian, perancangan sistem, implementasi *hardware* dan *software*, hasil pengujian, analisis hasil, serta kesimpulan dan saran. Laporan ini disusun sebagai bentuk pertanggungjawaban ilmiah dan sebagai referensi untuk penelitian selanjutnya.

2.1. Skema Blok Penelitian

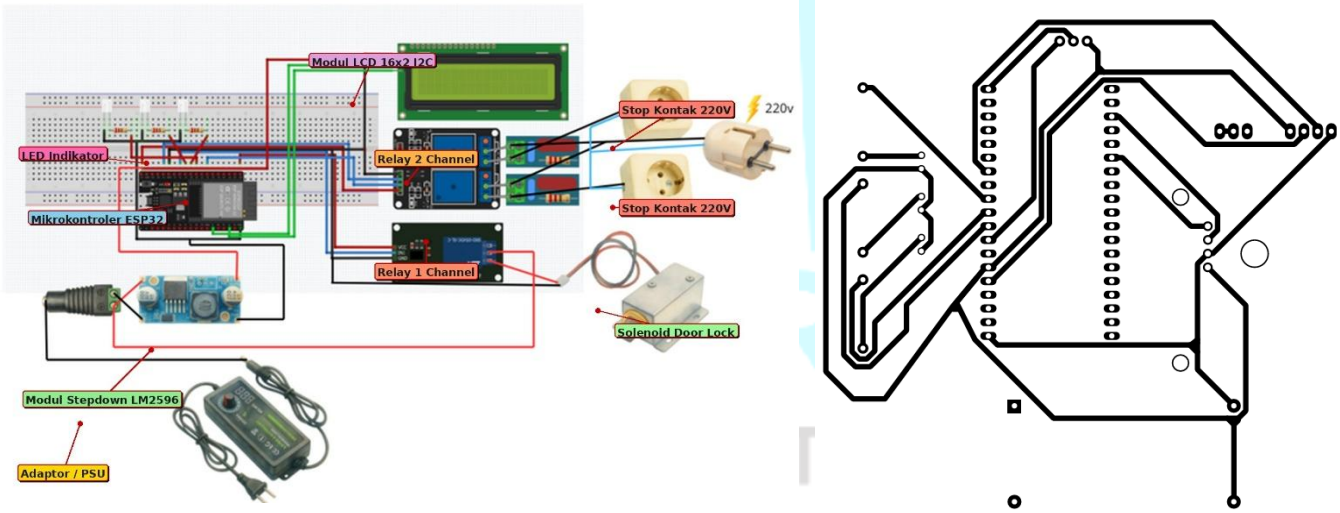


Gambar 2. Skema blok penelitian

Berdasarkan Gambar 2 ialah sistem prototipe asisten suara untuk informasi keseharian dan kendali *smart room* berbasis ESP32 ini bekerja melalui tiga tahap utama, yaitu input, proses, dan output. Pada tahap input, terdapat dua cara yang bisa digunakan untuk memberikan perintah ke sistem. Cara pertama adalah melalui suara, di mana pengguna cukup berbicara di depan mikrofon laptop dan suara tersebut akan langsung ditangkap oleh program *Python* untuk diproses menggunakan *Google Speech Recognition*. Cara kedua adalah melalui antarmuka web berbasis HTML yang bisa dibuka di *browser*, melaui *smartphone*, sehingga pengguna bisa memberikan perintah secara manual melalui tombol yang tersedia di halaman web tanpa harus berbicara. Kedua cara input ini sama-sama membutuhkan koneksi internet melalui WiFi agar bisa berkomunikasi dengan sistem. Pada tahap proses, perintah yang sudah dikenali oleh sistem *Python* maupun antarmuka web akan dikirimkan ke platform *Blynk* melalui internet. *Blynk* di sini berperan sebagai jembatan komunikasi antara laptop dan mikrokontroler ESP32. Setelah menerima perintah dari *Blynk*, ESP32 yang sudah terhubung ke

jaringan WiFi akan memproses perintah tersebut sesuai program yang telah ditanamkan melalui Arduino IDE, kemudian mengaktifkan atau menonaktifkan pin output yang sesuai. ESP32 juga membutuhkan *power supply* sebagai sumber daya agar dapat bekerja. Pada tahap output, sinyal dari ESP32 diteruskan ke modul relay yang berfungsi sebagai saklar elektronik. Relay ini yang kemudian mengontrol perangkat listrik bertegangan tinggi seperti lampu LED, kipas angin, dan *solenoid door lock*. Setiap kali relay aktif, LED indikator juga akan menyala sebagai tanda bahwa relay sedang dalam kondisi bekerja. Sementara itu, apabila perintah yang diberikan berupa pertanyaan informasi keseharian seperti jam, hari, atau tanggal, sistem tidak akan mengaktifkan relay melainkan langsung menghasilkan jawaban dalam bentuk suara melalui fitur *text-to-speech* pada *Python* yang dikeluarkan lewat speaker laptop.

2.2. *Wiring diagram system*

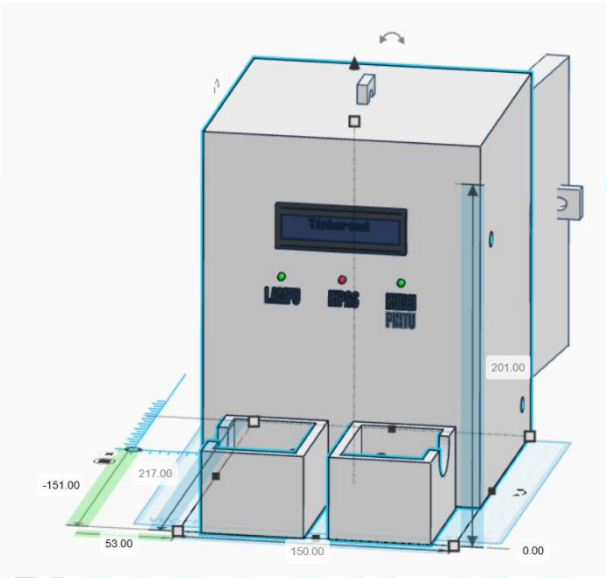


(a) *Wiring System* (b) Desain PCB

Gambar 3. *Wiring diagram*

Pada Gambar 3 tersebut menggambarkan hubungan antar komponen pada alat *smart room* yang terhubung dengan Python dan HTML, ESP32 berfungsi sebagai pusat pengendali yang terhubung dengan beberapa output yaitu Relay, Lampu LED sebagai indikator dan LCD I2C, Sumber daya utama sistem berasal dari adaptor yang menghasilkan tegangan DC. Tegangan tersebut kemudian diturunkan menggunakan modul *step-down* LM2596 menjadi tegangan yang sesuai untuk menyuplai ESP32 dan komponen *Solenoid Door Lock*, ESP32 dihubungkan ke modul relay 2 channel yang digunakan untuk mengontrol dua stop kontak bertegangan 220V. Stop kontak pertama digunakan untuk menghubungkan perangkat lampu, sedangkan stop kontak kedua digunakan untuk menghubungkan perangkat kipas angin. Selain modul relay 2 channel, terdapat juga modul relay 1

channel yang dihubungkan secara terpisah dan digunakan khusus untuk mengontrol *solenoid door lock* sebagai sistem pengunci pintu. Selain itu, ESP32 juga dihubungkan ke modul LCD 16x2 yang dipasang di bagian atas rangkaian. LCD ini berfungsi untuk menampilkan status sistem secara langsung pada perangkat, seperti informasi perintah yang sedang dijalankan maupun kondisi perangkat yang sedang aktif. Dan menambahkan output LED sebagai indikator tambahan bahwa sistem bisa berjalan dengan baik.



Gambar 4. Desain 3D wadah

Desain 3D dari wadah alat menggunakan software desain *Tinkercad*. Desain ini dibuat untuk memberikan gambaran awal tentang tata letak komponen dan dimensi wadah sebelum benar-benar diwujudkan. Berdasarkan Gambar 4, wadah alat dirancang berbentuk kotak dengan dimensi 217 × 150 × 201 mm. Bagian depan wadah dirancang untuk menampilkan layar LCD 16x2 di bagian atas, diikuti tiga buah LED indikator di bawahnya yang masing-masing diberi label Lampu, Kipas, dan Kunci Pintu. Di bagian bawah wadah disediakan dua ruang terpisah yang digunakan sebagai tempat penyimpanan kabel dan komponen internal agar tidak berantakan. Sisi kanan wadah dilengkapi engsel dan kait pintu sehingga bagian dalam bisa dibuka dan ditutup dengan mudah untuk keperluan perawatan atau perbaikan.

2.3 Perancangan *software*

Pengembangan *software* pada penelitian ini dibagi menjadi tiga bagian utama, yaitu program ESP32 menggunakan Arduino IDE, program *voice assistant* berbasis Python, dan antarmuka web

berbasis HTML, CSS, serta JavaScript. Ketiga bagian ini saling terintegrasi dan bekerja bersama melalui *platform Blynk* sebagai penghubung antara *software* dan *hardware*.

2.3.1 Program ESP32

ESP32 diprogram menggunakan Arduino IDE sebagai pengendali utama pada sisi *hardware*. Tugas utama program ini adalah menghubungkan ESP32 ke jaringan WiFi rumah dan mempertahankan koneksinya dengan server *Blynk* agar perangkat selalu dalam kondisi siap menerima perintah kapanpun dibutuhkan. Saat ada perintah masuk dari *Blynk*, ESP32 akan membaca nilai yang dikirimkan melalui pin virtual dan menentukan apakah relay harus diaktifkan atau dinonaktifkan berdasarkan nilai tersebut. Dalam sistem ini digunakan tiga pin virtual yaitu V0 yang difungsikan untuk mengontrol relay lampu, V1 untuk relay kipas angin, dan V3 untuk relay *solenoid door lock* sebagai pengunci pintu. Di samping fungsi kendali relay, program pada ESP32 juga mengatur tampilan layar LCD 16x2 yang dipasang pada wadah alat, di mana LCD ini menampilkan kondisi terkini dari setiap perangkat yang terhubung sehingga pengguna bisa langsung melihat status sistem tanpa harus membuka aplikasi atau laptop.

2.3.2. Program *python*

Program voice assistant dikembangkan menggunakan bahasa pemrograman *Python* yang dijalankan langsung pada laptop. Pemilihan Python sebagai bahasa pemrograman didasarkan pada ketersediaan *library* yang lengkap dan kemudahan integrasinya dengan berbagai layanan berbasis internet. Terdapat beberapa library utama yang digunakan dalam program ini, masing-masing memiliki peran yang berbeda dalam menjalankan sistem secara keseluruhan.

Library SpeechRecognition dimanfaatkan untuk menangkap suara yang masuk melalui mikrofon laptop dan mengubahnya menjadi teks menggunakan layanan *Google Speech Recognition*. Sebelum mulai mendengarkan, program terlebih dahulu melakukan kalibrasi kebisingan sekitar selama 0,5 detik menggunakan fungsi *adjust_for_ambient_noise* agar mikrofon bisa menyesuaikan sensitivitasnya dengan kondisi ruangan saat itu. Teks hasil pengenalan suara kemudian dicocokkan dengan daftar perintah yang sudah didefinisikan sebelumnya menggunakan operator in pada *Python*, sehingga sistem tetap bisa mengenali perintah meskipun pengguna menambahkan kata lain di depan atau belakang kata kunci.

Untuk memberikan respon balik kepada pengguna, digunakan *library gTTS* atau *Google Text-to-Speech* yang bertugas mengubah teks jawaban menjadi file audio, kemudian diputar menggunakan *library pygame*. Proses pengiriman perintah ke *hardware* dilakukan melalui *library requests* yang mengirimkan sinyal ke server *Blynk* menggunakan HTTP API, selanjutnya *Blynk* meneruskan perintah tersebut ke ESP32 untuk mengaktifkan relay yang sesuai.

Selain fungsi utama tersebut, program *Python* juga dilengkapi dengan fitur pengukur waktu respons atau delay yang bekerja secara otomatis setiap kali perintah diproses. Pengukuran dilakukan dengan mencatat waktu menggunakan fungsi *time.time()* tepat setelah suara selesai dikenali, kemudian mencatatnya kembali setelah seluruh proses perintah selesai dijalankan. Selisih antara kedua waktu tersebut menjadi nilai *delay* yang ditampilkan di terminal. Seluruh data perintah beserta hasil dan nilai *delay* nya kemudian disimpan secara otomatis ke dalam *file Excel* menggunakan *library openpyxl* ketika program dihentikan, baik melalui perintah suara maupun melalui *keyboard*.

2.3.3. Antarmuka Web (HTML)

Selain program *Python*, sistem ini juga dilengkapi dengan antarmuka web sebagai opsi kontrol tambahan yang bisa diakses melalui smartphone maupun laptop. Antarmuka web ini dikembangkan menggunakan tiga file terpisah yaitu *index.html* sebagai kerangka struktur halaman, *style.css* untuk mengatur tampilan visual, dan *app.js* sebagai tempat seluruh logika program dijalankan. Pemisahan ketiga *file* ini dilakukan agar kode lebih terorganisir dan mudah diperbaiki jika sewaktu-waktu ada perubahan yang perlu dilakukan pada salah satu bagiannya saja.

Untuk kontrol perangkat secara manual, tersedia tombol *toggle* yang bisa digunakan untuk menyalakan dan mematikan lampu serta kipas angin, tombol untuk membuka dan mengunci pintu, serta tombol untuk menyalakan atau mematikan semua perangkat secara bersamaan hanya dengan satu kali tekan. Selain kontrol manual, antarmuka web juga dilengkapi dengan fitur kontrol suara menggunakan *Web Speech API* yang sudah tersedia di browser *Google Chrome*, sehingga pengguna bisa memberikan perintah suara langsung dari HP tanpa perlu menggunakan laptop.

Setiap kali perintah dijalankan, baik melalui tombol manual maupun suara, sistem akan langsung menampilkan hasilnya pada kotak status yang ada di halaman web. Di bawah kotak status tersebut juga terdapat penghitung *delay* otomatis yang dihitung menggunakan fungsi *performance.now()* pada *JavaScript*, yang menunjukkan berapa lama waktu yang dibutuhkan sistem untuk memproses perintah dari awal hingga selesai. Seluruh riwayat perintah yang sudah dijalankan juga tersimpan dalam log yang ditampilkan di bagian bawah halaman, lengkap dengan informasi waktu, jenis perintah, hasil, dan nilai *delay* setiap percobaannya.

Untuk keperluan pencatatan data pengujian, antarmuka web dilengkapi dengan fitur ekspor data ke format *Excel* menggunakan *library SheetJS*. Ketika tombol *export* ditekan, sistem akan secara otomatis mengunduh *file Excel* yang berisi dua *sheet*, yaitu *sheet* pertama yang memuat *log* seluruh perintah secara lengkap dan *sheet* kedua yang berisi ringkasan statistik pengujian seperti total percobaan, jumlah perintah yang berhasil dan gagal, persentase keberhasilan, serta rata-rata *delay* keseluruhan.

2.4. Perintah yang Didukung Sistem

Sistem ini dirancang untuk bisa mengenali dua jenis perintah, yaitu perintah untuk mengontrol perangkat ruangan dan perintah untuk meminta informasi keseharian dengan rentang kebisingan yang digunakan dalam pengujian ini berkisar antara 50 hingga 80 dB karena berdasarkan hasil pengukuran langsung di lokasi pengujian, tingkat kebisingan ruangan yang terukur memang berada pada rentang tersebut. Semua perintah menggunakan Bahasa Indonesia dan sengaja dibuat dengan beberapa variasi kata kunci agar pengguna tidak perlu mengucapkan kata yang persis sama setiap kali memberikan perintah.

Tabel 1. Perintah smart room

No	Perintah	Fungsi
1	"nyalakan lampu" / "nyalain lampu" / "lampu nyala" / "lampu on"	Menyalakan lampu
2	"matikan lampu" / "lampu mati" / "lampu off"	Mematikan lampu
3	"nyalakan kipas" / "nyalain kipas" / "kipas nyala" / "kipas on" / "hidupkan kipas"	Menyalakan kipas
4	"matikan kipas" / "kipas mati" / "kipas off"	Mematikan kipas
5	"buka pintu" / "pintu buka" / "buka kunci pintu"	Membuka kunci pintu
6	"kunci pintu" / "pintu kunci"	Mengunci pintu
7	"nyalakan semua" / "nyalain semuanya" / "nyalakan lampu dan kipas"	Menyalakan lampu dan kipas
8	"matikan semua" / "matiin semuanya"	Mematikan lampu dan kipas

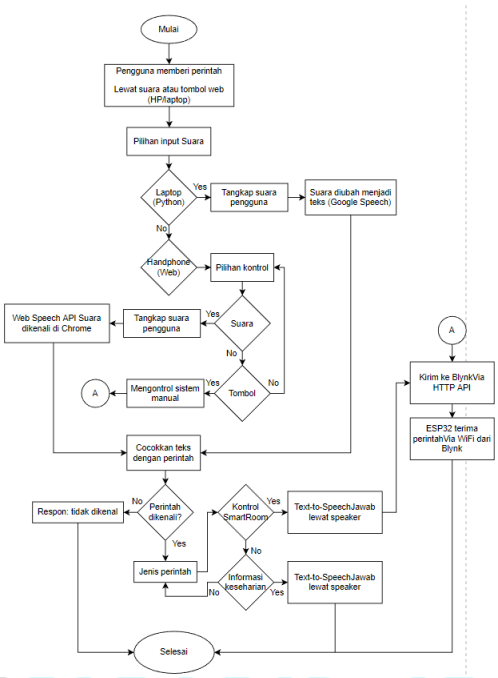
Selain perintah kendali perangkat, sistem juga bisa merespons pertanyaan seputar informasi keseharian. Berbeda dengan perintah kendali yang mengaktifkan relay, perintah jenis ini langsung diproses oleh program Python maupun antarmuka web untuk kemudian dijawab melalui *fitur text-to-speech* tanpa melibatkan ESP32 sama sekali.

Tabel 2. Perintah keseharian

No	Perintah	Fungsi
1	"jam berapa" / "pukul berapa"	Menyebutkan jam saat ini
2	"hari apa" / "hari ini apa"	Menyebutkan hari saat ini
3	"tanggal berapa" / "tanggal hari ini"	Menyebutkan tanggal saat ini
4	"siapa yang membuatmu" / "siapa pembuatmu"	Menyebutkan nama pembuat sistem
5	"tujuanmu apa" / "apa tujuanmu"	Menyebutkan tujuan sistem

Cara sistem mengenali perintah-perintah di Tabel 2 adalah dengan menggunakan metode pencocokan *substring*, bukan pencocokan kalimat penuh. Pada program *Python* digunakan operator in, sedangkan pada antarmuka web digunakan *method includes()* di *JavaScript*. Prinsip kerjanya cukup sederhana, yaitu sistem hanya perlu menemukan kata kunci di dalam kalimat yang diucapkan, tidak peduli ada tambahan kata lain di depan atau belakangnya. Misalnya ketika pengguna mengucapkan "tolong nyalakan lampu", sistem tetap bisa mengenali perintah tersebut karena di dalam kalimat itu terdapat kata kunci "nyalakan lampu".

2.5. Perancangan sistem kerja



Gambar 5. Sistem kerja

Sistem ini bekerja melalui satu alur utama yang mencakup dua fungsi sekaligus, yaitu kendali perangkat dan informasi keseharian. Alur kerja lengkap sistem digambarkan pada Gambar 5. Sistem dimulai ketika pengguna memberikan perintah melalui salah satu dari dua jalur input yang tersedia. Jalur pertama adalah melalui suara yang ditangkap mikrofon dan diproses menggunakan *Google Speech Recognition* pada *Python* atau *Web Speech API* pada *browser Chrome*. Jalur kedua adalah melalui klik tombol manual pada antarmuka web yang bisa diakses dari HP maupun laptop. Kedua jalur ini akan menghasilkan teks perintah yang kemudian dicocokkan dengan daftar kata kunci menggunakan operator in pada Python atau *method includes()* pada *JavaScript*.

Jika perintah tidak dikenali, sistem langsung memberikan respon bahwa perintah tidak dapat dipahami. Jika perintah berhasil dikenali, sistem akan menentukan ke mana tindakan harus diarahkan berdasarkan jenis perintahnya. Untuk perintah kendali perangkat, sistem mengirimkan sinyal ke *platform Blynk* melalui HTTP API. *Blynk* kemudian meneruskan sinyal tersebut ke mikrokontroler ESP32 melalui koneksi WiFi, dan ESP32 akan mengaktifkan atau menonaktifkan relay yang sesuai untuk mengendalikan lampu, kipas angin, maupun *solenoid door lock*.

Untuk perintah informasi keseharian, sistem tidak melibatkan ESP32 maupun relay sama sekali. Sistem hanya perlu menentukan jenis informasi yang diminta. Jika pertanyaan menyangkut jam, hari, atau tanggal, sistem mengambil data langsung dari jam dan kalender perangkat. Jika pertanyaan menyangkut pembuat atau tujuan sistem, jawaban sudah tersimpan sebagai teks tetap di dalam program. Setelah jawaban ditentukan, teks tersebut langsung diubah menjadi suara

menggunakan *gTTS* pada *Python* atau *Web Speech API* pada *browser* dan diputar melalui *speaker* HP ataupun Laptop.

2.6. Metode Pengujian

Setelah sistem selesai dibuat, tahap selanjutnya adalah melakukan serangkaian pengujian untuk mengetahui sejauh mana sistem bisa bekerja dengan baik. Pengujian dilakukan secara langsung menggunakan prototipe yang sudah dirakit dengan kondisi pengujian yang bervariasi agar data yang diperoleh bisa mencerminkan performa sistem secara menyeluruh.

Pengujian pertama adalah pengujian perintah suara yang bertujuan untuk mengetahui seberapa andal sistem dalam mengenali dan menjalankan perintah yang diberikan. Dengan pengujian kebisingan dilakukan untuk mengetahui seberapa besar pengaruh tingkat kebisingan lingkungan terhadap kemampuan sistem dalam mengenali perintah suara. Tingkat kebisingan diukur menggunakan aplikasi *Decibel X* pada *smartphone* yang diletakkan di dekat mikrofon selama pengujian berlangsung. Setiap perintah diucapkan sebanyak 10 kali dan dicatat apakah sistem berhasil merespons dengan benar atau tidak. Pengujian ini dilakukan terhadap seluruh perintah yang didukung sistem, baik perintah kendali perangkat seperti lampu, kipas, dan pintu, maupun perintah informasi keseharian seperti menanyakan jam, hari, dan tanggal. Tingkat keberhasilan kemudian dihitung menggunakan Persamaan 1.

$$\text{Persentase Keberhasilan} = \frac{\sum Keberhasilan}{\sum Pengujian}\ x\ 100\% \quad (1)$$

Pengujian berikutnya adalah pengujian tombol manual pada antarmuka web yang dilakukan untuk memastikan seluruh tombol kontrol yang tersedia bisa berfungsi dengan baik ketika diakses melalui *smartphone* maupun laptop. Setiap tombol ditekan satu per satu dan diamati apakah relay pada *hardware* merespons dengan benar sesuai fungsinya. Bersamaan dengan itu juga dilakukan pengujian variasi kata kunci untuk mengetahui seberapa fleksibel sistem dalam mengenali perintah yang diucapkan dengan susunan kalimat yang berbeda. Beberapa variasi ucapan dicoba untuk setiap kata kunci, misalnya dengan menambahkan kata tambahan di depan perintah atau menggunakan susunan kalimat yang sedikit berbeda namun tetap mengandung kata kunci yang sama, untuk mengevaluasi apakah metode pencocokan *substring* yang digunakan sudah cukup fleksibel dalam menangani variasi ucapan pengguna.

Pengujian terakhir adalah pengujian waktu respons atau *delay* yang bertujuan untuk mengukur berapa lama waktu yang dibutuhkan sistem dari perintah selesai diucapkan hingga sistem selesai merespons. Pada program *Python* pengukuran dilakukan secara otomatis menggunakan fungsi *time.time()* yang mencatat waktu mulai dan selesai setiap perintah diproses dan hasilnya langsung

ditampilkan di terminal. Pada antarmuka web pengukuran dilakukan menggunakan fungsi *performance.now()* yang hasilnya ditampilkan di halaman web setelah setiap perintah selesai dijalankan. Setiap perintah diuji sebanyak 10 kali percobaan dan nilai rata-rata *delay* dihitung menggunakan Persamaan 2.

$$\bar{D} = \frac{D1+D2+\ldots+Dn}{n} \quad (2)$$

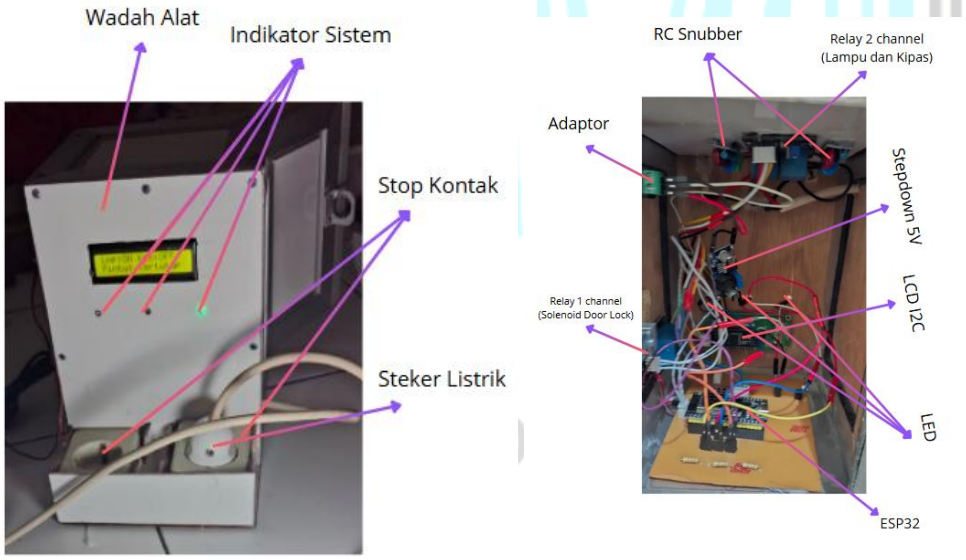
$\bar{D}$ = rata-rata delay (detik)

$D1 + D2+\ldots+Dn$ = nilai delay setiap percobaan (detik)

n = jumlah percobaan

3. HASIL DAN PEMBAHASAN

3.1. Hasil pengujian *hardware*



(b) Tampak Luar (c) Tampak Dalam

Gambar 6. Tampilan *Hardware*

Hasil implementasi *Hardware* ditunjukkan pada kedua Gambar 6. Seluruh komponen ditempatkan di dalam wadah berbahan akrilik putih berbentuk kotak yang dirancang agar tampilan alat terlihat rapi sekaligus melindungi komponen elektronik di dalamnya.

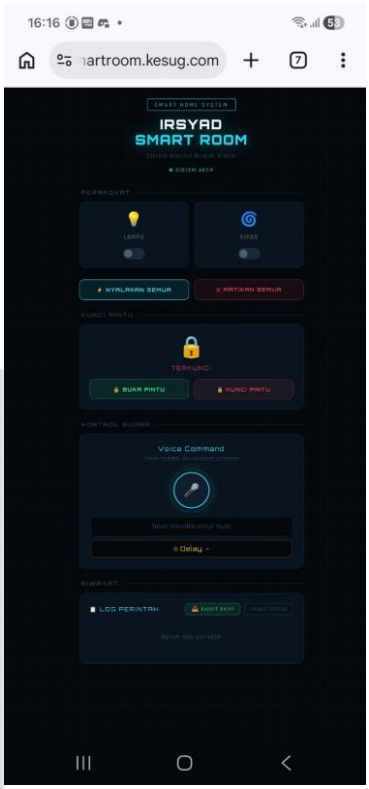
Hasil pengujian tabel dapat dilihat pada Tabel 3.

Tabel 3. Hasil pengujian komponen *hardware*

No	Komponen	Status	Keterangan
1	Mikrokontroler ESP32	Normal	Terhubung ke jaringan WiFi dan server *Blynk* dengan stabil
2	Relay 2 *Channel*	Normal	Mampu mengaktifkan dan menonaktifkan lampu serta kipas sesuai perintah
3	Relay 1 *Channel*	Normal	Mampu mengunci dan membuka kunci solenoid door lock sesuai perintah
4	LCD 16x2 I2C	Normal	Menampilkan splash screen dan status perangkat secara *real-time*
5	LED Indikator	Normal	Menyala bersamaan dengan relay sebagai penanda visual status perangkat
6	*Stepdown* LM2596	Normal	Tegangan output stabil dan tidak ada gangguan daya selama pengujian

Sebelum pengujian sistem secara keseluruhan dilakukan, terlebih dahulu dilakukan pengujian pada masing-masing komponen perangkat keras untuk memastikan semuanya bisa bekerja dengan baik. Dari hasil pengujian tersebut ditunjukkan pada Tabel 3. seluruh komponen yang terpasang dalam wadah alat menunjukkan kondisi yang normal dan tidak ditemukan kendala apapun selama pengujian berlangsung. Bagian dalam wadah memuat mikrokontroler ESP32 sebagai pusat kendali sistem, modul stepdown 5V untuk menurunkan tegangan dari adaptor, modul relay 2 channel untuk mengontrol lampu dan kipas angin, modul relay 1 channel khusus untuk solenoid door lock, modul LCD I2C, LED indikator, serta RC Snubber yang dipasang untuk meredam lonjakan tegangan saat relay bekerja. Sementara itu pada bagian luar wadah, layar LCD dan tiga LED indikator dipasang di bagian depan agar status sistem bisa langsung terlihat tanpa harus membuka wadah. Di sisi kanan terdapat dua stop kontak sebagai titik keluaran daya untuk perangkat lampu dan kipas, sedangkan di bagian bawah terdapat steker listrik yang menghubungkan seluruh sistem ke sumber listrik PLN 220V.

3.2. Pengujian Tombol Manual Antarmuka Web



Gambar 7. Tampilan Web

Tampilan antarmuka web sistem Irsyad Smart Room ditunjukkan pada Gambar 7. Antarmuka ini diakses melalui browser smartphone pada alamat smartroom.kesug.com dengan tampilan bertema gelap futuristik yang dirancang agar nyaman digunakan di layar kecil. Terdapat beberapa bagian utama pada halaman ini. Di bagian atas terdapat header yang menampilkan nama sistem beserta indikator status sistem aktif. Di bawahnya terdapat bagian kontrol perangkat yang memuat dua tombol toggle untuk menyalakan dan mematikan lampu serta kipas angin secara individual, serta dua tombol untuk menyalakan dan mematikan semua perangkat sekaligus. Dilakukan pengujian pada masing-masing tombol pada antarmuka Web untuk memastikan semuanya bisa bekerja dengan baik. Dari hasil pengujian tersebut ditunjukkan pada Tabel 4. seluruh tombol yang tersedia pada antarmuka web menunjukkan kondisi yang berhasil dan tidak ditemukan kendala apapun selama pengujian berlangsung**.**

Tabel 4. Pengujian Tombol Manual Antarmuka Web

No	Komponen	Status	Keterangan
1	*Toggle* Lampu *ON*	Berhasil	Relay lampu aktif, LED indikator menyala
2	*Toggle* Lampu *OFF*	Berhasil	Relay lampu nonaktif, LED indikator mati
3	*Toggle* Kipas *ON*	Berhasil	Relay kipas aktif, LED indikator menyala
4	*Toggle* Kipas *OFF*	Berhasil	Relay kipas nonaktif, LED indikator mati
5	Buka Pintu	Berhasil	Relay *door lock* aktif, solenoid membuka kunci
6	Kunci Pintu	Berhasil	Relay *door lock* nonaktif, solenoid mengunci
7	Nyalakan Semua	Berhasil	Relay lampu dan kipas aktif bersamaan
8	Matikan Semua	Berhasil	Relay lampu dan kipas nonaktif bersamaan

Setiap tombol yang ditekan berhasil mengirimkan sinyal ke *Blynk* dan diteruskan ke ESP32 untuk mengaktifkan atau menonaktifkan relay yang sesuai. Hal ini dibuktikan dengan menyalanya LED indikator pada wadah alat setiap kali relay aktif, yang menandakan bahwa komunikasi antara antarmuka web, *Blynk*, dan ESP32 berjalan dengan lancar. Dari hasil pengujian ini dapat disimpulkan bahwa antarmuka web dapat dijadikan sebagai alternatif kontrol yang handal selain perintah suara, terutama ketika pengguna berada di kondisi yang tidak memungkinkan untuk memberikan perintah melalui suara.

3.3. Hasil Pengujian Perintah Suara kontrol *SmartRoom*

Pengujian perintah suara dilakukan sebanyak 10 kali percobaan untuk setiap perintah, baik pada program *Python* maupun antarmuka web HTML. Hasil pengujian pada masing-masing platform ditunjukkan pada Tabel 4 dan Tabel 5. Rumus rata-rata hasil pengujian dapat dilihat pada Persamaan 1.

Contoh penggunaan Persamaan 1 pada Tabel 5 :

$$\text{Persentase Keberhasilan} = \frac{\sum Keberhasilan}{\sum Pengujian}\ x\ 100\% = \frac{74}{80}\ x\ 100\%$$

$$= 92{,}5\%$$

Tabel 5. Pengujian perintah suara melalui *Python*

No	Perintah	Berhasil	Gagal	% Keberhasilan
1	Nyalakan Lampu	9	1	90%
2	Matikan Lampu	9	1	90%
3	Nyalakan Kipas	10	0	100%
4	Matikan Kipas	7	3	70%
5	Buka Pintu	10	0	100%
6	Kunci Pintu	10	0	100%
7	Nyalakan Semua	9	1	90%
8	Matikan Semua	10	0	100%
Total		**74**	**6**	**92,5%**

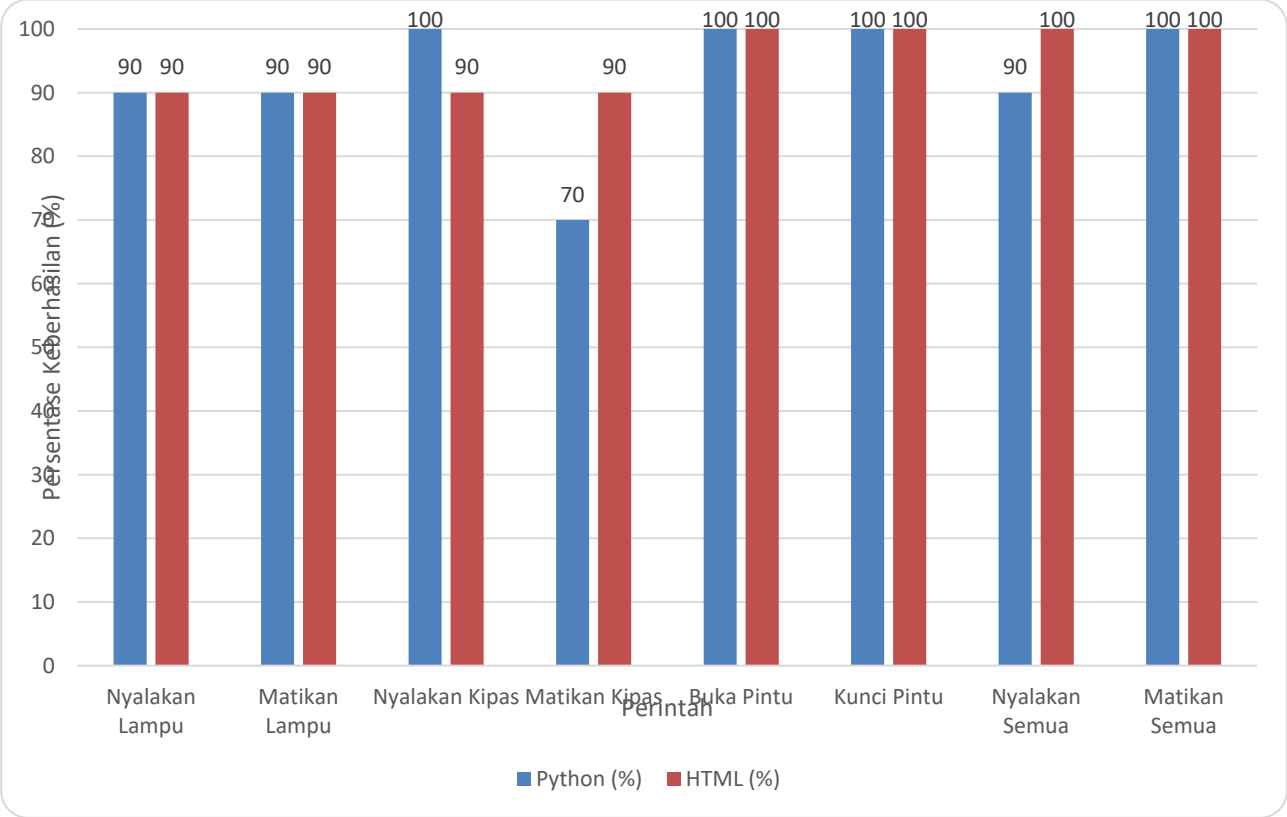
Contoh penggunaan Persamaan 1 pada Tabel 6 :

$$\text{Persentase Keberhasilan} = \frac{\sum Keberhasilan}{\sum Pengujian} \; x \; 100\% = \frac{76}{80} \; x \; 100\%$$

$$= 95\%$$

Tabel 6. Pengujian perintah suara melalui HTML

No	Perintah	Berhasil	Gagal	% Keberhasilan
1	Nyalakan Lampu	9	1	90%
2	Matikan Lampu	9	1	90%
3	Nyalakan Kipas	9	1	90%
4	Matikan Kipas	9	1	90%
5	Buka Pintu	10	0	100%
6	Kunci Pintu	10	0	100%
7	Nyalakan Semua	10	0	100%
8	Matikan Semua	10	0	100%
Total		**76**	**4**	**95%**



Gambar 8. Grafik persentase keberhasilan

Berdasarkan data pada Gambar 8 tingkat keberhasilan program Python sebesar 92,5% sedangkan antarmuka web HTML mencapai 95%. Perbedaan ini terjadi karena pada *Python*, perintah matikan kipas mengalami tiga kali kegagalan yang disebabkan oleh sistem hanya menangkap kata "matikan" atau "matikan ki" tanpa melanjutkan kata "kipas" secara utuh, sehingga tidak cocok dengan kata kunci yang sudah didefinisikan. Hal yang sama terjadi pada beberapa perintah lain di *Python* di mana *Google Speech Recognition* hanya menangkap sebagian kata dari perintah yang diucapkan. Sementara pada antarmuka web HTML, kegagalan lebih merata di beberapa perintah dengan masing-masing hanya satu kali gagal, yang kemungkinan disebabkan oleh kondisi penangkapan suara sesaat yang kurang optimal. Secara keseluruhan kedua platform menunjukkan performa yang cukup baik dengan tingkat keberhasilan di atas 90%.

3.4. Hasil Pengujian Perintah Suara Informasi Keseharian

Pengujian perintah informasi keseharian dilakukan untuk mengetahui kemampuan sistem dalam merespons pertanyaan seputar waktu, tanggal, hari, serta informasi tentang sistem itu sendiri. Setiap perintah diuji sebanyak 10 kali percobaan pada dua platform, yaitu program *Python* dan antarmuka web HTML.

Contoh penggunaan Persamaan 1 pada Tabel 7 :

$$\text{Persentase Keberhasilan} = \frac{\sum Keberhasilan}{\sum Pengujian}\ x\ 100\% = \frac{94}{50}\ x\ 100\%$$

$$= 94\%$$

Tabel 7. Pengujian Perintah Informasi Keseharian melalui *Python*

No	Perintah	Berhasil	Gagal	% Keberhasilan
1	Jam berapa / Pukul berapa	10	0	90%
2	Hari apa / Hari ini apa	10	0	90%
3	Tanggal berapa / Tanggal hari ini	9	1	90%
4	Siapa yang membuatmu / Siapa pembuatmu	9	1	90%
5	Tujuanmu apa / Apa tujuanmu	9	1	100%
Total		**47**	**3**	**94%**

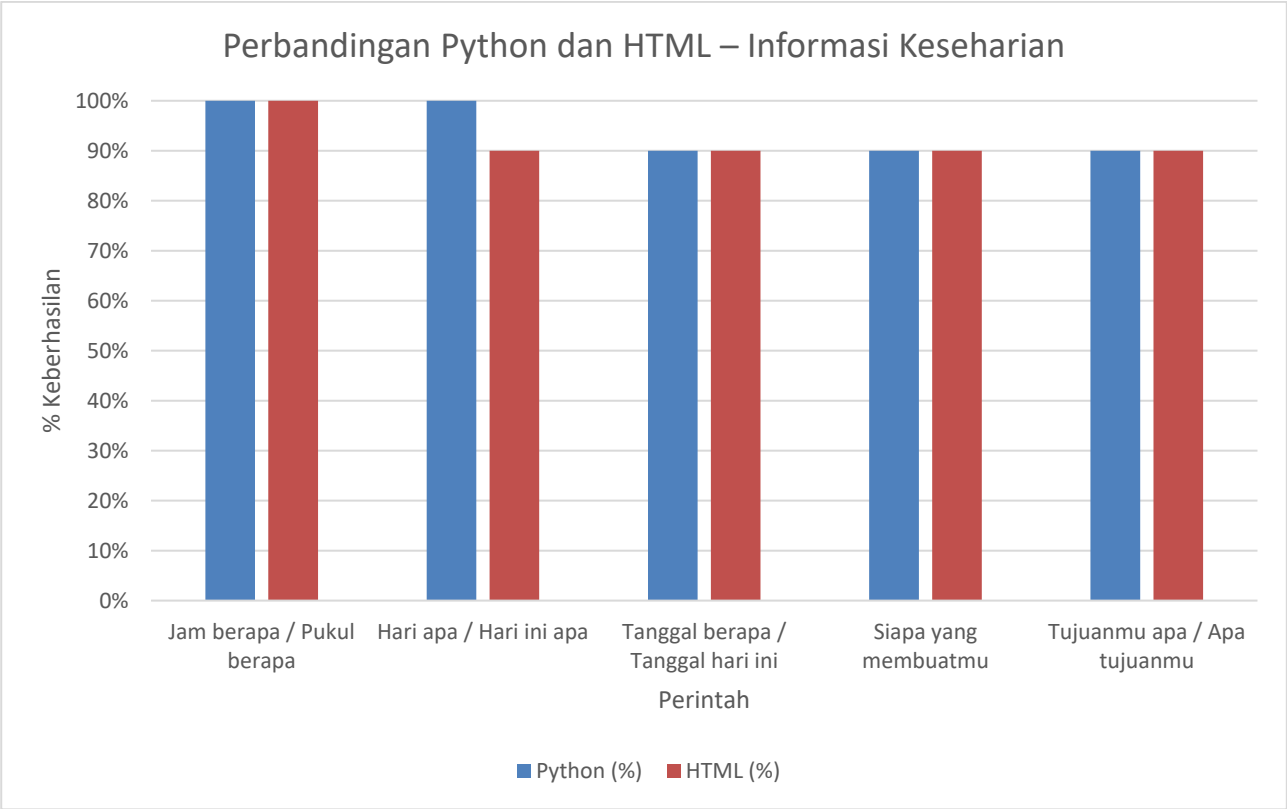
Contoh penggunaan Persamaan 1 pada Tabel 8 :

$$\text{Rata rata} = \frac{Berhasil}{Pengujian}\ x\ 100\% = \frac{96}{50}\ x\ 100\% \quad (4)$$

$$= 92\%$$

Tabel 8. Pengujian Perintah Informasi Keseharian melalui HTML

No	Perintah	Berhasil	Gagal	% Keberhasilan
1	Jam berapa / Pukul berapa	10	0	90%
2	Hari apa / Hari ini apa	9	1	90%
3	Tanggal berapa / Tanggal hari ini	9	1	90%
4	Siapa yang membuatmu / Siapa pembuatmu	9	1	90%
5	Tujuanmu apa / Apa tujuanmu	9	1	100%
Total		**46**	**4**	**92%**



Gambar 9. Perbandingan Python dan HTML untuk informasi keseharian

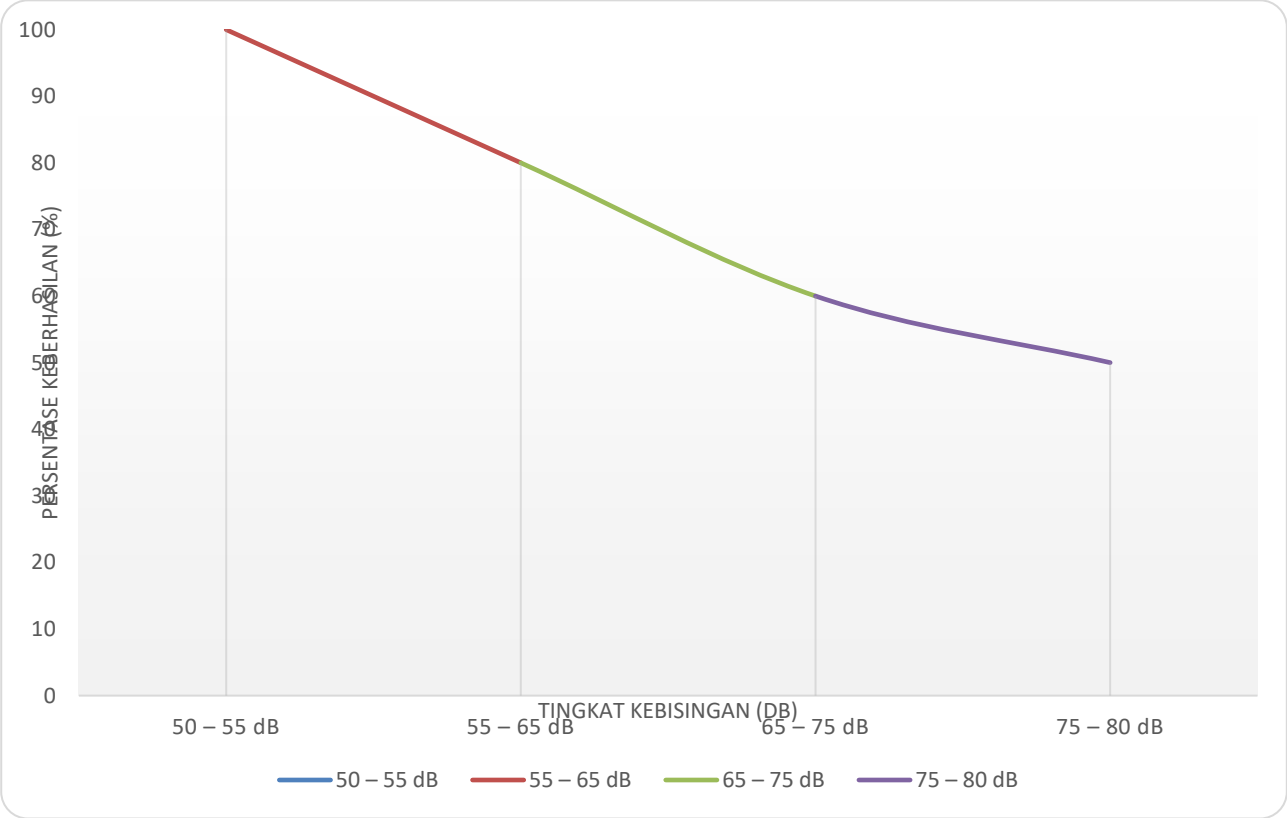
Program *Python* berhasil mencapai tingkat keberhasilan sebesar 94% sedangkan antarmuka web HTML sedikit di bawahnya yaitu 92%. Meskipun terdapat selisih kecil di antara keduanya, kedua platform tetap menunjukkan performa yang baik karena sama-sama berada di atas angka 90%. kata kunci "jam berapa / pukul berapa" menjadi perintah yang paling mudah dikenali oleh sistem pada kedua platform. Hal ini kemungkinan karena kata-kata tersebut memiliki pengucapan yang cukup khas dan tidak mudah tertukar dengan kata lain sehingga proses pengenalan suaranya lebih konsisten. Sementara itu, kegagalan yang terjadi pada perintah seperti "tanggal berapa", "siapa pembuatmu", dan "tujuanmu apa" lebih banyak disebabkan oleh mikrofon yang hanya menangkap sebagian kata dari kalimat yang diucapkan, misalnya hanya menangkap "tanggal" tanpa kata "berapa" di belakangnya, sehingga sistem tidak bisa mencocokkannya dengan kata kunci yang sudah terdaftar.

3.5. Hasil Pengujian Kebisingan

Pengujian kebisingan dilakukan untuk mengetahui seberapa besar pengaruh tingkat kebisingan lingkungan terhadap kemampuan sistem dalam mengenali perintah suara. Hasil pengujian ditunjukkan pada Tabel 9**.**

Tabel 9. Pengujian Kebisingan

No.	Kebisingan (dB)	∑Keberhasilan	∑Kegagalan	% Keberhasilan
1.	50 – 55 dB	10	0	100
2.	55 – 65 dB	8	2	80
3.	65 – 75 dB	6	4	60
4.	75 – 80 dB	5	5	50



Gambar 10. Grafik Pengujian Kebisingan

Sistem menunjukkan performa terbaik pada kondisi kebisingan di bawah 55 dB dengan tingkat keberhasilan 100%. Pada rentang 55 hingga 65 dB tingkat keberhasilan turun menjadi 80%, yang menunjukkan bahwa kebisingan ringan seperti percakapan di sekitar sudah mulai mempengaruhi kemampuan mikrofon dalam menangkap perintah secara utuh. Penurunan lebih lanjut terjadi pada rentang 65 hingga 75 dB dengan tingkat keberhasilan 60%, di mana suara latar seperti kipas yang berisik atau musik mulai cukup dominan sehingga menyulitkan *Google Speech Recognition* dalam memilah suara perintah. Pada kondisi paling berisik yaitu rentang 75 hingga 80 dB, tingkat

keberhasilan hanya mencapai 50% yang berarti separuh dari percobaan gagal dikenali karena kebisingan sudah sangat mengganggu proses penangkapan suara oleh mikrofon.

3.6. Hasil Pengujian *Delay Python* dan HTML

Pengujian waktu respon dilakukan untuk mengukur berapa lama waktu yang dibutuhkan sistem dari perintah selesai diucapkan hingga sistem selesai merespons. Hasil pengujian *delay* pada program *Python* dan antarmuka web HTML ditunjukkan pada Tabel 10 dan Tabel 11.

Contoh penggunaan Persamaan 2 pada Tabel 10 :

- Perintah 1 : $\overline{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{4{,}10+3{,}99+6{,}73+5{,}10+3{,}62+4{,}31+5{,}70+3{,}69+4{,}31+3{,}72}{10} = \frac{45{,}27}{10} = 4{,}53$
- Perintah 2 : $\overline{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{4{,}34+3{,}68+3{,}53+4{,}76+3{,}83+3{,}52+4{,}58+3{,}77+3{,}79+3{,}77}{10} = \frac{39{,}57}{10} = 3{,}96$
- Perintah 3 : $\overline{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{3{,}90+3{,}72+4{,}84+4{,}11+4{,}98+5{,}42+4{,}48+4{,}91+3{,}78+3{,}91}{10} = \frac{44{,}05}{10} = 4{,}41$
- Perintah 4 : $\overline{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{4{,}31+3{,}95+3{,}98+4{,}26+4{,}87+5{,}95+4{,}80+4{,}26+4{,}87+4{,}80}{10} = \frac{46{,}05}{10} = 4{,}60$
- Perintah 5 : $\overline{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{4{,}47+6{,}66+5{,}18+4{,}40+4{,}29+4{,}30+4{,}22+4{,}03+3{,}99+6{,}02}{10} = \frac{47{,}56}{10} = 4{,}76$
- Perintah 6 : $\overline{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{5{,}03+4{,}90+3{,}59+4{,}40+4{,}29+4{,}30+4{,}22+4{,}03+3{,}99+6{,}02}{10} = \frac{44{,}77}{10} = 4{,}48$
- Perintah 7 : $\overline{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{7{,}25+8{,}53+5{,}87+5{,}13+5{,}65+8{,}31+4{,}90+5{,}78+7{,}22+7{,}22}{10} = \frac{65{,}86}{10} = 6{,}59$
- Perintah 8 : $\overline{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{8{,}44+5{,}73+7{,}32+5{,}13+5{,}65+5{,}62+8{,}92+5{,}87+5{,}43+5{,}67}{10} = \frac{63{,}78}{10} = 6{,}38$

Tabel 10. Pengujian Delay Python

No	Perintah	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Rata rata
1	Nyalakan Lampu	4,10	3,99	6,73	5,10	3,62	4,31	5,70	3,69	4,31	3,72	4,53
2	Matikan Lampu	4,34	3,68	3,53	4,76	3,83	3,52	4,58	3,77	3,79	3,77	3,96
3	Nyalakan Kipas	3,90	3,72	4,84	4,11	4,98	5,42	4,48	4,91	3,78	3,91	4,41
4	Matikan Kipas	4,31	3,95	3,98	4,26	4,87	5,95	4,80	4,26	4,87	4,80	4,60
5	Buka Pintu	4,47	6,66	5,18	4,40	4,29	4,30	4,22	4,03	3,99	6,02	4,76
6	Kunci Pintu	5,03	4,90	3,59	4,40	4,29	4,30	4,22	4,03	3,99	6,02	4,48
7	Nyalakan Semua	7,25	8,53	5,87	5,13	5,65	8,31	4,90	5,78	7,22	7,22	6,59

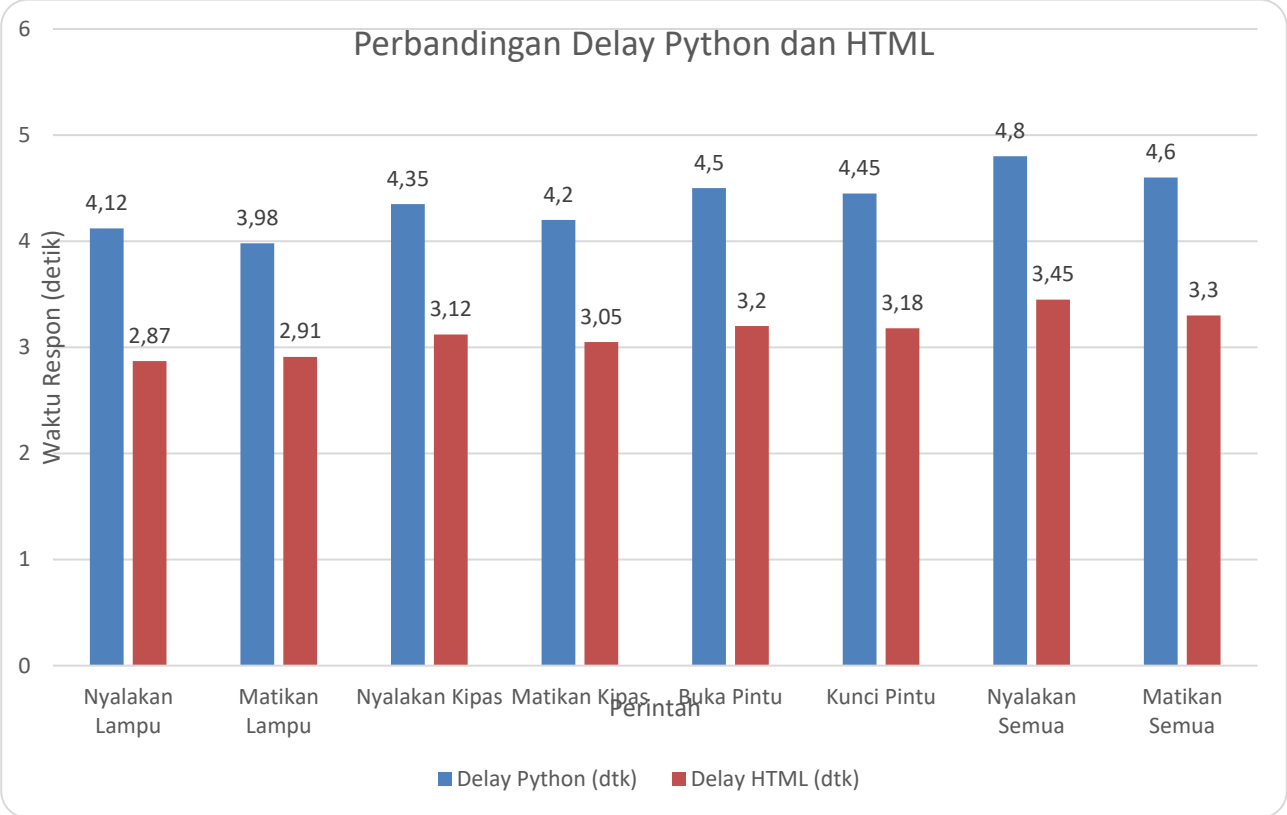
8	Matikan Semua	8,44	5,73	7,32	5,13	5,65	5,62	8,92	5,87	5,43	5,67	6,38

Contoh penggunaan persamaan 2 pada Tabel 11 :

- Perintah 1 : $\bar{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{2,57+2,57+2,72+2,56+2,62+2,70+2,53+2,53+2,67+2,73}{10} = \frac{26,20}{10} = 2,62$
- Perintah 2 : $\bar{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{2,74+2,56+2,65+2,77+2,57+2,96+2,55+2,57+2,53+2,57}{10} = \frac{26,47}{10} = 2,65$
- Perintah 3 : $\bar{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{2,90+2,96+2,67+3,11+2,62+2,74+2,79+2,77+2,75+2,54}{10} = \frac{27,85}{10} = 2,79$
- Perintah 4 : $\bar{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{2,64+2,60+2,66+2,77+2,67+2,80+2,67+2,60+2,54+2,75}{10} = \frac{26,70}{10} = 2,67$
- Perintah 5 : $\bar{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{2,86+2,66+2,64+2,59+2,27+2,43+2,49+2,57+2,60+2,23}{10} = \frac{25,34}{10} = 2,53$
- Perintah 6 : $\bar{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{2,58+2,67+2,49+2,24+2,55+2,51+2,28+2,58+2,24+3,59}{10} = \frac{25,73}{10} = 2,57$
- Perintah 7 : $\bar{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{2,37+2,63+2,60+2,36+2,42+2,74+2,49+2,74+2,41+2,48}{10} = \frac{25,24}{10} = 2,52$
- Perintah 8 : $\bar{D} = \frac{D1+D2+\ldots+Dn}{n} = \frac{2,28+2,45+2,52+2,34+2,45+2,49+2,56+2,36+2,75+2,37}{10} = \frac{24,57}{10} = 2,46$

Tabel 11. Pengujian Delay HTML

No	Perintah	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Rata rata
1	Nyalakan Lampu	2,57	2,57	2,72	2,56	2,62	2,70	2,53	2,53	2,67	2,73	2,62
2	Matikan Lampu	2,74	2,56	2,65	2,77	2,57	2,96	2,55	2,57	2,53	2,57	2,65
3	Nyalakan Kipas	2,90	2,96	2,67	3,11	2,62	2,74	2,79	2,77	2,75	2,54	2,79
4	Matikan Kipas	2,64	2,60	2,66	2,77	2,67	2,80	2,67	2,60	2,54	2,75	2,67
5	Buka Pintu	2,86	2,66	2,64	2,59	2,27	2,43	2,49	2,57	2,60	2,23	2,53
6	Kunci Pintu	2,58	2,67	2,49	2,24	2,55	2,51	2,28	2,58	2,24	3,59	2,57
7	Nyalakan Semua	2,37	2,63	2,60	2,36	2,42	2,74	2,49	2,74	2,41	2,48	2,52
8	Matikan Semua	2,28	2,45	2,52	2,34	2,45	2,49	2,56	2,36	2,75	2,37	2,46



Gambar 11. Perbandingan Delay Python dan HTML

Rata-rata *delay* pada antarmuka web HTML jauh lebih rendah yaitu hanya 2,60 detik. Nilai delay tertinggi pada HTML terjadi pada perintah "nyalakan kipas" sebesar 2,79 detik dan yang paling rendah pada perintah "matikan semua" sebesar 2,46 detik. Perbedaan antar perintah pada HTML tidak terlalu signifikan dan cenderung konsisten di kisaran 2,46 hingga 2,79 detik. Hal ini terjadi karena antarmuka web menggunakan *Web Speech API* yang berjalan langsung di *browser* tanpa perlu membuat *file audio* terlebih dahulu, sehingga proses *text-to-speech* jauh lebih ringan dan cepat dibanding gTTS pada *Python*.

Jika dibandingkan secara keseluruhan, antarmuka web HTML lebih unggul dari segi kecepatan respon dengan selisih rata-rata sekitar 2,36 detik dibanding program *Python*. Meskipun begitu, perbedaan ini tidak serta-merta membuat program Python kalah fungsi karena keduanya tetap mampu menjalankan seluruh perintah dengan benar.

3.7. Hasil Pengujian Variasi Kata Kunci

Pengujian variasi kata kunci dilakukan untuk melihat sejauh mana sistem masih bisa mengenali perintah ketika pengguna mengucapkannya dengan cara yang sedikit berbeda dari kata kunci yang sudah ditentukan. Dari 8 variasi ucapan yang dicoba, hasilnya ditunjukkan pada Tabel 12.

Tabel 12. Variasi Kata Kunci

No.	Kata Kunci Terdaftar	Variasi Ucapan	Status	Keterangan
1	"nyalakan lampu"	"tolong nyalakan lampu"	Berhasil	Mengandung kata kunci
2	"nyalakan lampu"	"bisakah nyalakan lampu"	Berhasil	Mengandung kata kunci
3	"nyalakan lampu"	"aktifkan lampu"	Gagal	Kata kunci tidak terdaftar
4	"nyalakan lampu"	"hidupkan lampu"	Gagal	Kata kunci tidak terdaftar
5	"matikan kipas"	"tolong matikan kipas"	Berhasil	Mengandung kata kunci
6	"buka pintu"	"tolong buka pintu"	Berhasil	Mengandung kata kunci
7	"buka pintu"	"pintunya dibuka"	Gagal	Susunan kata tidak cocok
8	"matikan semua"	"matiin semuanya"	Berhasil	Terdaftar sebagai variasi

Variasi ucapan yang berhasil dikenali semuanya adalah kalimat yang masih memuat kata kunci aslinya secara utuh, hanya saja ada tambahan kata di bagian depan seperti "tolong" atau "bisakah". Karena sistem menggunakan metode pencocokan *substring*, selama kata kuncinya masih ada di dalam kalimat maka sistem tetap bisa memprosesnya dengan benar. Selain itu variasi "matiin semuanya" juga berhasil karena kata tersebut memang sudah didaftarkan sejak awal sebagai salah satu variasi yang dikenali dalam program.

Sementara itu tiga variasi yang gagal punya alasan masing-masing. Ucapan "aktifkan lampu" dan "hidupkan lampu" gagal karena kedua kata tersebut tidak ada dalam daftar kata kunci, meskipun maknanya sama dengan "nyalakan lampu". Sedangkan "pintunya dibuka" gagal bukan karena katanya salah, tapi karena susunannya dibalik sehingga kata kunci "buka pintu" tidak ditemukan dalam urutan yang benar oleh sistem.

3.8. Analisis Keterbatasan Sistem

Tabel 13. Keterbatasan Sistem

No.	Keterbatasan	Dampak	Saran Pengembangan
1	Bergantung penuh pada internet	Sistem mati total saat offline	Tambahkan mode offline dengan speech recognition lokal
2	*Delay Python* rata-rata 4,96 detik	Respon terasa lambat	Ganti gTTS dengan library TTS offline seperti pyttsx3
3	Fleksibilitas kata kunci terbatas	Akurasi turun di atas 65 dB	Tambahkan sinonim dan variasi kata kunci di program atau menggunakan sistem *LLM (Large Language Model)*
4	Sensitif terhadap kebisingan	"hidupkan lampu"	Tambahkan *noise filtering* atau *directional microphone*
5	Tidak ada otentikasi pengguna	Siapapun bisa mengontrol sistem, termasuk buka pintu	Tambahkan verifikasi suara dan enkripsi token Blynk

Keterbatasan pertama adalah ketergantungan sistem terhadap koneksi internet. Seluruh proses pengenalan suara pada program *Python* maupun antarmuka web HTML bergantung sepenuhnya pada layanan *Google Speech Recognition* yang membutuhkan koneksi internet aktif. Hal yang sama juga berlaku pada komunikasi antara program *Python* dan ESP32 yang harus melewati server *Blynk* terlebih dahulu. Akibatnya ketika koneksi internet tidak stabil atau terputus, sistem tidak bisa berfungsi sama sekali meskipun perangkat keras dalam kondisi normal.

Keterbatasan kedua adalah nilai *delay* yang cukup tinggi terutama pada program *Python* dengan rata-rata 4,96 detik per perintah. Tingginya delay ini disebabkan oleh rangkaian proses yang harus dilalui secara berurutan mulai dari pengenalan suara oleh *Google Speech Recognition*, pengiriman sinyal ke server *Blynk*, hingga proses *text-to-speech* menggunakan gTTS yang harus membuat file audio terlebih dahulu sebelum diputar.

Keterbatasan ketiga adalah fleksibilitas pengenalan perintah yang masih terbatas. Sistem hanya bisa mengenali perintah yang mengandung kata kunci yang sudah didefinisikan sebelumnya, sehingga perintah dengan kata yang berbeda meskipun bermakna sama seperti "aktifkan lampu" sebagai pengganti "nyalakan lampu" tidak akan dikenali. Masalah ini sebenarnya muncul karena sistem masih mengandalkan logika if-else sederhana yang sifatnya rule-based, jadi cara kerjanya cuma mencocokkan teks secara harfiah tanpa benar-benar mengerti maksud dari kalimat yang diucapkan. Ke depannya, hal ini bisa jadi salah satu poin yang menarik untuk dikembangkan lebih lanjut, misalnya dengan mencoba menerapkan *Large Language Model (LLM)* sebagai pengganti metode pencocokan kata kunci yang dipakai sekarang. Dengan bantuan LLM, sistem berpotensi bisa memahami maksud pengguna dengan lebih luwes, bahkan ketika kalimatnya disusun berbeda atau menggunakan gaya bahasa yang lebih santai sekalipun, sehingga ke depannya interaksi antara pengguna dengan sistem bisa terasa lebih natural tanpa harus terpaku pada kata kunci yang kaku seperti saat ini.

Keterbatasan keempat adalah pengaruh kebisingan lingkungan yang cukup signifikan terhadap akurasi pengenalan suara. Hasil pengujian menunjukkan bahwa pada kondisi kebisingan di atas 65 dB, tingkat keberhasilan sistem mulai turun secara signifikan hingga hanya mencapai 50% pada kondisi kebisingan 75 hingga 80 dB. Hal ini membuat sistem kurang ideal untuk digunakan di lingkungan yang ramai atau berisik.

Keterbatasan kelima yang perlu disoroti adalah dari sisi keamanan sistem. Sejauh ini, komunikasi antara program Python, antarmuka web, Blynk, dan ESP32 belum dilengkapi dengan sistem otentikasi pengguna seperti login atau verifikasi suara, sehingga siapapun yang mengetahui cara mengakses antarmuka web atau berada cukup dekat dengan mikrofon bisa langsung memberikan perintah ke sistem, termasuk perintah untuk membuka kunci pintu. Hal ini jadi cukup riska, terutama untuk fitur door lock yang berkaitan langsung dengan keamanan fisik ruangan. Selain itu, token autentikasi Blynk yang digunakan untuk komunikasi antara perangkat juga masih ditulis langsung di dalam kode program tanpa enkripsi tambahan, yang sebenarnya kurang ideal kalau sistem ini nantinya dikembangkan untuk penggunaan yang lebih luas atau dipublikasikan kodenya secara terbuka.

4. PENUTUP

Berdasarkan seluruh proses perancangan, implementasi, dan pengujian yang telah dilakukan pada penelitian ini, dapat disimpulkan bahwa sistem asisten suara untuk kendali *smart room* berbasis ESP32 berhasil dibangun dan berfungsi dengan baik sesuai tujuan awal penelitian. Sistem ini mampu menjalankan dua fungsi utama secara bersamaan, yaitu mengendalikan perangkat ruangan seperti lampu, kipas angin, dan kunci pintu, serta memberikan informasi keseharian seperti waktu, hari, dan tanggal melalui interaksi suara dua arah. Dari sisi perangkat keras, seluruh komponen yang digunakan mulai dari ESP32, modul relay, LCD, hingga LED indikator terbukti berfungsi dengan baik tanpa kendala selama pengujian berlangsung. Pada sisi perangkat lunak, sistem dikembangkan melalui dua jalur kontrol yang berbeda, yakni program *Python* yang dijalankan di laptop dan antarmuka web berbasis HTML yang bisa diakses melalui *smartphone*, sehingga memberikan fleksibilitas lebih bagi pengguna dalam memilih cara berinteraksi dengan sistem. Hasil pengujian menunjukkan bahwa tingkat keberhasilan pengenalan perintah suara mencapai 92,5% pada program Python dan 95% pada antarmuka web, dengan rata-rata waktu respon masing-masing 4,96 detik dan 2,60 detik. Antarmuka web HTML menunjukkan performa yang sedikit lebih unggul dibanding *Python*, baik dari segi tingkat keberhasilan maupun kecepatan respon, yang kemungkinan disebabkan oleh penggunaan *Web Speech API* yang lebih ringan dibanding kombinasi *Google Speech Recognition* dan gTTS pada *Python*. Selain itu, pengujian tombol manual pada antarmuka web bisa berfungsi selama internet pada hardware masih terkoneksi, sementara pengujian variasi kata kunci hanya mencapai 62,5% yang menunjukkan bahwa sistem masih memiliki keterbatasan dalam memahami variasi cara bicara pengguna. Faktor kebisingan lingkungan juga terbukti berpengaruh terhadap akurasi sistem, di mana performa terbaik diperoleh pada kondisi ruangan dengan kebisingan di bawah 65 dB. Mengacu pada hasil pengujian dan keterbatasan yang ditemukan selama penelitian berlangsung, ada beberapa hal yang bisa dijadikan masukan untuk pengembangan sistem ini ke depannya. Pertama, sistem saat ini masih sangat bergantung pada koneksi internet untuk menjalankan *Google Speech Recognition* maupun komunikasi dengan *Blynk*, sehingga ke depannya bisa dipertimbangkan untuk menambahkan mode offline menggunakan library speech recognition lokal agar sistem tetap bisa berjalan meskipun tanpa koneksi internet. Kedua, nilai delay pada program *Python* yang masih tergolong cukup tinggi bisa diperbaiki dengan mengganti *library gTTS* yang digunakan saat ini dengan *library text-to-speech offline* seperti *pyttsx3*, sehingga proses pembuatan respon suara tidak perlu lagi melewati server *Google* dan bisa berjalan lebih cepat. Ketiga, keterbatasan dalam hal fleksibilitas pengenalan perintah yang masih menggunakan pendekatan rule-based berbasis logika *if-else* bisa dikembangkan lebih jauh dengan memanfaatkan *Large Language Model (LLM)* sebagai pengganti metode pencocokan kata kunci yang dipakai saat ini. Dengan penerapan LLM, sistem berpotensi mampu memahami maksud

pengguna secara lebih luwes meskipun kalimat yang diucapkan menggunakan susunan atau pilihan kata yang berbeda dari kata kunci yang sudah didaftarkan. Keempat, mengingat akurasi sistem cukup terpengaruh oleh kebisingan lingkungan terutama di atas 65 dB, penelitian selanjutnya bisa mencoba menambahkan mikrofon *directional* atau teknik *noise filtering* agar sistem tetap bisa bekerja dengan baik meskipun digunakan di lingkungan yang lebih ramai. Kelima, dari sisi keamanan, sistem ini belum memiliki mekanisme otentikasi pengguna sehingga perintah sensitif seperti membuka kunci pintu masih bisa dijalankan oleh siapa saja yang berada dekat dengan mikrofon atau mengetahui cara mengakses antarmuka web. Pengembangan selanjutnya disarankan untuk menambahkan fitur verifikasi suara pengguna sebelum menjalankan perintah-perintah yang berkaitan dengan keamanan fisik, serta menyimpan kredensial seperti token Blynk menggunakan metode penyimpanan yang lebih aman agar tidak langsung terlihat di dalam kode program.

PERSANTUNAN

Alhamdulillahirabbil‘alamin, segala puji dan syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat, hidayah, dan kemudahan-Nya sehingga penulisan artikel ilmiah ini dapat terselesaikan dengan baik. Penulis juga menyampaikan terima kasih dan penghargaan yang setinggitingginya kepada berbagai pihak yang telah memberikan dukungan baik teknis maupun non teknis selama proses penyusunan dan penyelesaian penelitian ini. Ucapan terima kasih secara khusus penulis sampaikan kepada :

1. Orang tua dan keluarga tercinta yang senantiasa memberikan doa, dukungan, dan semangat dalam menyelesaikan studi dan penelitian ini.
2. Ibu Umi Fadlilah, S.T., M.Eng, Ph, D selaku dosen pembimbing yang telah memberikan arahan, motivasi, dan bimbingan secara intensif selama proses penelitian dan penyusunan artikel ilmiah.
3. Seluruh dosen dan staf pengajar di Program Studi Teknik Elektro Universitas Muhammadiyah Surakarta yang telah memberikan ilmu dan pengalaman berharga selama masa perkuliahan.
4. Seluruh rekan mahasiswa Teknik Elektro angkatan 2022 yang telah menemani dan memberikan bantuan selama proses perkuliahan hingga tahap akhir penyusunan karya ilmiah ini.

Semoga segala bantuan dan kebaikan yang telah diberikan menjadi amal jariyah dan mendapat balasan yang terbaik dari Allah SWT.

DAFTAR PUSTAKA

Bhaganagare, S., Chavan, S., Gavali, S., & Godase, V. (2025a). *Voice Controlled Home Automation Using ESP32 and IoT-Based Cloud Integration*. 1411–1432. https://doi.org/10.48175/IJARSCT-29478

Bhaganagare, S., Chavan, S., Gavali, S., & Godase, V. V. (2025b). *Voice-Controlled Home Automation with ESP32 : A Systematic Review of IoT-Based Solutions*. *2*(3). https://www.academia.edu/download/124444859/_1_13_Revised_Automation_with_ESP32.pdf

Cruz-Morales, R. (2024). *Design and implementation of a voice assistant to be used in a IoT home automation environment Abstract*. https://www.researchgate.net/publication/375957243_Design_and_implementation_of_a_voice_assistant_to_be_used_in_an_IoT_home_automation_environment

Hanani, A., & Hariyadi, M. A. (2020). *Smart Home Berbasis IoT Menggunakan Suara Pada Google Assistant*. *14*(1), 49–56. https://jurnal.asia.ac.id/index.php/jitika/article/view/456/261

Kurniadi, D., & Setiawan, R. (2024). *Aplikasi Voice Assistant Pada Smartwach Menggunakan Open Artificial Intellegence*. 309–320. https://doi.org/10.33364/algoritma/v.21-2.1499

Nur Afiyat, M. H., & Hakim, M. D. Al. (2021). *Sistem pengendalian perangkat elektronik berbasis IoT (Internet of Things) menggunakan voice control dan Blynk*. *4*(1), 93–104. https://doi.org/https://doi.org/10.31598

Prasetyana, B. E., & Fadlilah, U. (2021). *Smart home menggunakan easy voice recognition berbasis Internet of Things*. *4*, 152–158. https://proceedings.ums.ac.id/rapi/article/view/154

Surani, B. S., Dharmawan, A., & Bakti, C. A. (2025). *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*. *8*. https://journal.ipm2kpe.or.id/index.php/INTECOM/article/view/16363/10748