Sistem Pengendalian Perangkat Smart Home Dengan Perintah Suara Menggunakan Raspberry Pi dan Smartphone

Laporan Tugas Akhir

Oleh:
Erik Chaniago (4242211007)
Billy Hagas Tarigan (4242211037)

Program Studi Teknologi Rekayasa Elektronika
Jurusan Teknik Elektro
Politeknik Negeri Batam
2026

Lembar Pengesahan

Tugas Akhir disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Terapan Teknik (S.Tr.T)

Disusun oleh:
Erik Chaniago (4242211007)
Billy Hagas Tarigan (4242211037)

Tanggal Sidang: 15 Januari 2026

Disetujui oleh :

1. Ika Karlina Laila Nur Suciningtyas, S.Si.,M.Si
NIK: 119219

1. Illa Aryeni,S.T., M.T.
NIK:122255

2. Ririn Humaera, M.Pd
NIK:124322

Sistem Pengendalian Perangkat *Smart Home* dengan perintah Suara menggunakan *Raspberry Pi* dan *Smartphone*

Abstrak

Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem pengendalian perangkat *smart home* berbasis perintah suara secara *offline* menggunakan *Raspberry Pi* dan *smartphone*. Sistem dikembangkan untuk mengendalikan perangkat elektronik rumah tangga seperti lampu, kipas angin, televisi, dan pendingin ruangan (AC) melalui kombinasi perintah suara, kendali *web*, penjadwalan otomatis, serta komunikasi infrared dan *relay*. Metode pengenalan suara menggunakan model *Faster-Whisper* yang dioptimalkan untuk pemrosesan lokal tanpa ketergantungan koneksi internet, sedangkan komunikasi data antar perangkat menggunakan protokol *MQTT*. Pengujian dilakukan untuk mengevaluasi performa sistem dari beberapa aspek, meliputi akurasi pengenalan suara, *latency* pemrosesan, pengaruh *noise* lingkungan, *Quality of Service (QoS) MQTT*, serta keandalan *database* penyimpanan *data log*. Hasil pengujian menunjukkan bahwa model *Faster-Whisper* INT8 memberikan performa terbaik dengan waktu pemrosesan paling cepat dan penggunaan memori paling rendah dibandingkan model lain. Sistem pengenalan suara mencapai tingkat keberhasilan rata-rata sebesar 98% dari total 600 percobaan, dengan rata-rata *latency* sekitar 4,45 detik pada kondisi lingkungan normal. Pengujian *noise* menunjukkan bahwa peningkatan kebisingan menyebabkan penurunan akurasi dan peningkatan *latency*, namun sistem masih bekerja dengan baik pada tingkat *noise* sedang. Pengujian *MQTT* menunjukkan tingkat keberhasilan pengiriman pesan sebesar 100% pada seluruh level *QoS*, dengan *latency* rata-rata terendah pada *QoS* 0. Selain itu, sistem *database* mampu menyimpan hingga 1000 *data log* dengan *success rate* 100% dan waktu *insert* serta *query* yang relatif cepat. Berdasarkan hasil tersebut, sistem yang dikembangkan terbukti andal, efisien, dan layak diterapkan sebagai solusi *smart home* berbasis pengenalan suara *offline*, khususnya untuk meningkatkan kenyamanan dan aksesibilitas pengguna.

Kata kunci: *Smart Home*, Perintah Suara, *Raspberry Pi, Faster-Whisper*, *MQTT*, IoT.

Smart Home Device Control System with Voice Commands Using Raspberry Pi and Smartphone

Abstract

This research aims to design and implement an offline voice-controlled smart home system using a Raspberry Pi and a smartphone. The system is developed to control household electronic devices such as lights, fans, televisions, and air conditioners through a combination of voice commands, web-based control, automated scheduling, and infrared and relay-based actuation. The speech recognition method employs the Faster-Whisper model optimized for local processing, eliminating dependency on internet connectivity, while inter-device communication is handled using the MQTT protocol. Several performance evaluations were conducted, including voice recognition accuracy, processing latency, the impact of environmental noise, MQTT Quality of Service (QoS), and database reliability for activity logging. Experimental results show that the Faster-Whisper INT8 model provides the most optimal performance, offering the fastest processing time and lowest memory usage compared to other models. The voice command system achieved an overall success rate of 98% across 600 test trials, with an average processing latency of approximately 4.45 seconds under normal environmental conditions. Noise testing indicates that higher noise levels reduce recognition accuracy and increase latency; however, the system remains reliable under moderate noise conditions typically found in residential environments. MQTT performance testing demonstrates a 100% message delivery success rate across all QoS levels, with the lowest average latency observed at QoS level 0. Furthermore, the database logging system successfully stored up to 1000 log entries with a 100% success rate, maintaining efficient insert and query times. Based on these results, the proposed system is proven to be reliable, efficient, and suitable as an offline voice-based smart home solution, particularly in enhancing user convenience and accessibility.

Keywords: Smart Home, Voice Command, Raspberry Pi, Faster-Whisper, MQTT, Internet of Things.

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab 1. Pendahuluan

1.1 Latar Belakang

Perkembangan teknologi *Internet of Things (IoT)* telah membuka peluang besar dalam menciptakan lingkungan rumah yang lebih cerdas dan responsif terhadap kebutuhan penghuninya. *Smart home* menjadi salah satu implementasi nyata dari teknologi ini, dimana berbagai perangkat rumah tangga dapat dikendalikan secara otomatis dan jarak jauh. Salah satu metode interaksi yang semakin diminati adalah pengendalian berbasis suara karena dinilai lebih praktis dan intuitif. Sistem kendali berbasis suara sangat membantu penyandang disabilitas ganda, terutama mereka yang tidak memiliki tangan atau kaki, dalam melakukan aktivitas sehari-hari. Hal ini menunjukkan bahwa teknologi pengenalan suara tidak hanya relevan dalam konteks kenyamanan, tetapi juga memiliki dampak besar dalam mendukung kemandirian individu berkebutuhan khusus. Penyandang disabilitas seringkali menghadapi keterbatasan dalam berinteraksi dengan perangkat rumah tangga yang umum digunakan, sehingga diperlukan sistem yang mampu menjembatani kebutuhan tersebut. *Smart home* berbasis suara menjadi solusi potensial untuk menjawab tantangan tersebut [1].

Selain untuk penyandang disabilitas fisik, teknologi *smart home* berbasis suara juga sangat bermanfaat bagi penyandang disabilitas sensorik, seperti tunanetra. Teknologi pengenalan suara memiliki potensi luas dalam meningkatkan kualitas hidup penyandang disabilitas dengan berbagai jenis kebutuhan khusus. Aksesibilitas menjadi salah satu tantangan utama dalam penerapan teknologi modern, namun dengan pendekatan desain yang inklusif, hambatan tersebut dapat diatasi. Penerapan teknologi suara yang tepat sasaran memungkinkan terwujudnya lingkungan rumah yang tidak hanya cerdas, tetapi juga adaptif terhadap kebutuhan semua kalangan. Melalui sistem semacam ini, pengguna tunanetra tidak hanya dapat mengendalikan perangkat rumah, tetapi juga merasa lebih aman, percaya diri, dan mandiri dalam ruang pribadinya [2].

Namun demikian, sebagian besar sistem pengenalan suara yang tersedia di pasaran masih bergantung pada koneksi internet dan layanan *cloud* untuk dapat memproses perintah pengguna. Ketergantungan ini menjadi salah satu hambatan besar, terutama di daerah dengan keterbatasan jaringan atau bagi pengguna yang tidak memiliki akses internet secara stabil. Untuk mengatasi permasalahan tersebut, diperlukan pengembangan sistem pengenalan suara *offline*. Sistem ini mampu mengenali perintah suara secara lokal tanpa perlu mengirimkan data ke *server* eksternal, sehingga proses pengenalan menjadi lebih cepat, aman, dan tidak tergantung pada ketersediaan jaringan. Sistem *offline* seperti ini memungkinkan pengguna tetap dapat menggunakan fitur *smart home* meskipun dalam kondisi tanpa internet. Bagi penyandang disabilitas, hal ini menjadi sangat penting karena mereka memerlukan sistem yang dapat diandalkan setiap saat.

Selain itu, penggunaan sistem lokal juga membantu menjaga privasi pengguna karena data suara tidak dikirimkan ke *server* pihak ketiga. Oleh karena itu, pengembangan sistem pengenalan suara lokal menjadi langkah strategis dalam mewujudkan *smart home* yang lebih inklusif dan adaptif [3].

Berdasarkan uraian di atas, dapat disimpulkan bahwa teknologi *smart home* berbasis pengenalan suara memiliki potensi besar untuk meningkatkan kualitas hidup, khususnya bagi penyandang disabilitas yang memiliki keterbatasan dalam mobilitas maupun penglihatan. Namun, tantangan seperti ketergantungan pada koneksi internet dan keterbatasan akurasi dalam kondisi nyata masih menjadi perhatian penting. Oleh karena itu, tugas akhir ini bertujuan untuk merancang dan mengimplementasikan sistem *smart home* berbasis pengenalan suara *offline* menggunakan perangkat *Raspberry Pi*, yang mampu mengenali perintah suara lokal secara akurat, cepat, dan andal, bahkan dalam kondisi lingkungan yang kurang ideal. Diharapkan sistem ini dapat menjadi solusi yang inklusif dan bermanfaat bagi penyandang disabilitas, serta menjadi kontribusi nyata dalam pengembangan teknologi yang lebih manusiawi dan berkeadilan.

1.2 Rumusan Masalah

Adapun rumusan masalah yang diperoleh pada penelitian ini adalah sebagai berikut :

1. Bagaimana menghubungkan perangkat keras dan perangkat lunak dalam sistem *smart hom*e menggunakan *Raspberry Pi* dan *Smartphone*?
2. Bagaimana mengimplementasikan pengenalan suara dalam sistem pengendalian perangkat *smart home* menggunakan *Raspberry Pi* dan *Smartphone*?
3. Bagaimana mengimplementasikan protokol komunikasi *MQTT* dalam sistem perangkat *smart home* menggunakan *Raspberry Pi* dan *Smartphone*?

1.3 Tujuan

Adapun tujuan pada penelitian ini antara lain:

1. Merancang dan menghubungkan perangkat keras serta perangkat lunak pada sistem *smart home* berbasis *Raspberry Pi* dan *smartphone* agar dapat berfungsi secara optimal.
2. Mengimplementasikan sistem pengenalan suara untuk mengendalikan perangkat *smart home* secara efisien melalui *Raspberry Pi* dan *smartphone*.
3. Menerapkan protokol komunikasi *MQTT* dalam sistem *smart home* untuk mendukung komunikasi data yang ringan, stabil, dan efektif antara perangkat.

1.4 Manfaat

Adapun manfaat yang diperoleh dari penelitian ini adalah sebagai berikut :

1. Meningkatkan kenyamanan pengguna sistem perangkat *smart home* untuk tetap dapat mengendalikan perangkat elektronik di rumah tanpa harus bergantung pada jarigan internet.
2. Mengurangi biaya energi listrik ketika pengguna sistem perangkat *smart home* lupa mematikan perangkat elektronik ketika meninggalkan rumah.
3. Mengembangkan sistem pengenalan suara *(speech-to-text)* secara lokal tanpa jaringan internet menggunakan *Raspberry Pi*.
4. Membantu para penyandang disabilitas khususnya tuna netra untuk melakukan sesuatu didalam lingkungan tempat tinggal.

1.5 Batasan

Adapun batasan masalah agar penelitian yang akan dilakukan lebih fokus dan memberikan hasil yang akurat adalah sebagai berikut :

1. Penelitian ini hanya dirancang untuk dapat mengendalikan perangkat elektronik di rumah pada bagian saklar (lampu) dan perintah *infrared* (televisi dan pendingin ruangan).
2. Penelitian ini akan diimplementasikan dalam bentuk perangkat keras mikrokontroler *Raspberry Pi*, *relay* dan modul *infrared*.
3. Penelitian ini hanya berfokus pada akurasi pengenalan suara *(speech-to- text)* yang di proses secara lokal oleh perangkat lunak yang dipasang pada *Raspberry Pi.*
4. Penelitian ini tidak membahas tentang gangguan suara latar belakang ataupun proses filter suara.
5. Penelitian ini menggunakan protokol komunikasi *MQTT* dan berfokus pada kecepatan komunikasi dan perintah dari suara pengguna hingga aktual perintah telah di eksekusi ke masing-masing perangkat elektronik.
6. Penelitian ini hanya berfokus pada pengendalian lampu, kipas angin, pendingin ruangan (*AC*) dan televisi.
7. Penelitian pada *smartphone* bukan dengan suara, tetapi menggunakan fitur klik/tombol pada *website* yang telah disediakan.

1.6 *Work Breakdown Structure*

Work Breakdown Structure (WBS) adalah cara untuk membagi sebuah proyek besar menjadi bagian-bagian kecil sehingga dapat dengan mudah diselesaikan. Proyek dipecah secara bertingkat, mulai dari tujuan utama, lalu menjadi pekerjaan besar, kemudian diuraikan lagi menjadi tugas-tugas detail. Dengan WBS, tim bisa lebih jelas melihat apa saja yang harus dilakukan, siapa yang bertanggung jawab,

serta lebih mudah mengatur waktu dan biaya agar proyek berjalan terarah dan tidak ada pekerjaan yang terlewat. Berikut merupakan pembagian tugas dan tanggung jawab pada penelitian ini.

Tabel 1. *Work Breakdown Structure*

No	Nama	Tugas dan Tanggung Jawab dalam Tim
1	Erik Chaniago	1. Studi literatur dan mencari referensi untuk metode pengenalan suara (*speech-to-text*). 2. Mengembangkan perangkat lunak pengenalan suara (*speech-to-text*) menggunakan *Raspberry Pi.* 3. Merakit perangkat keras dan rangkaian komponen yang diperlukan. 4. Pengujian metode pengenalan suara (*speech-to-text*). 5. Pengambilan data hasil pengujian pengenalan suara (*speech-to-text*). 6. Analisis hasil pengujian pengenalan suara (*speech-to-text*)..
2	Billy Hagas Tarigan	1. Studi literatur dan mencari referensi untuk metode komunikasi dengan protokol *MQTT*. 2. Mengembangkan perangkat lunak metode komunikasi dengan protokol *MQTT*. 3. Merakit perangkat keras dan rangkaian komponen yang diperlukan. 4. Pengujian metode komunikasi dengan protokol *MQTT*. 5. Pengambilan data hasil pengujian metode komunikasi dengan protokol *MQTT*. 6. Analisis hasil pengujian metode komunikasi dengan protokol *MQTT*.

Bab 2. Tinjauan Pustaka

2.1 Penelitian Terkait

Penelitian yang dilakukan oleh Kusuma dan Wahyudi membahas perancangan sistem *smart home* berbasis pengenalan suara menggunakan *Google Assistant* dan ESP8266 sebagai pengendali utama. Sistem ini dirancang untuk memudahkan pengguna dalam mengendalikan perangkat rumah seperti lampu, kipas angin, dan televisi hanya melalui perintah suara. Komunikasi antara *Google Assistant* dan perangkat dikembangkan menggunakan *platform IFTTT (If This Then That)* yang menjembatani perintah suara dengan *server cloud* yang akan memberikan perintah *HTTP request* kepada *NodeMCU*. *NodeMCU* selanjutnya akan memicu *relay* untuk mengaktifkan atau menonaktifkan perangkat elektronik. Pengujian dilakukan dengan beberapa jenis perintah suara dan dalam kondisi koneksi internet yang berbeda. Hasil pengujian menunjukkan bahwa perintah suara dapat diterjemahkan dan dijalankan dengan tingkat keberhasilan di atas 90% saat jaringan internet stabil. Namun, terdapat penurunan performa ketika koneksi internet lambat atau terputus. Penelitian ini membuktikan bahwa integrasi pengenalan suara berbasis *cloud* dapat diimplementasikan secara efektif pada sistem *smart home* sederhana [4].

Penelitian lain oleh Putri dan Handayani berfokus pada pengembangan sistem *smart home* berbasis perintah suara tanpa koneksi internet, yang sangat berguna untuk daerah dengan jaringan terbatas. Sistem ini menggunakan *Voice Recognition Module V3* yang menyimpan kata kunci secara lokal dan menghubungkannya dengan *Arduino Uno*

sebagai mikrokontroler. Perintah suara diproses langsung oleh modul tanpa harus dikirim ke *server* eksternal, sehingga responsnya jauh lebih cepat. Dalam implementasinya, sistem dapat mengendalikan berbagai perangkat rumah seperti lampu, televisi, dan kipas angin dengan menggunakan perintah-perintah yang telah diprogram sebelumnya. Pengujian dilakukan di dalam ruangan dengan kondisi suara stabil. Hasilnya menunjukkan bahwa akurasi pengenalan suara mencapai 85% jika kata kunci diucapkan dengan jelas dan sesuai intonasi saat perekaman awal. Kelemahan dari sistem ini adalah keterbatasan jumlah kata yang bisa disimpan dalam modul dan sensitivitas terhadap suara bising. Meski demikian, sistem ini dinilai cocok untuk pengguna rumahan yang menginginkan solusi lokal tanpa ketergantungan terhadap internet atau *cloud service* [5].

Selanjutnya, penelitian oleh Setiawan dkk. mengembangkan sistem pengendalian perangkat rumah berbasis suara yang dikombinasikan dengan aplikasi Android dan mikrokontroler ESP32. Dalam penelitian ini, perintah suara diberikan melalui

aplikasi Android yang mengintegrasikan *Google Speech Recognition API*. Suara pengguna yang direkam melalui mikrofon ponsel akan diterjemahkan menjadi teks, kemudian dikirim ke ESP32 melalui jaringan *Wi-Fi* menggunakan protokol komunikasi HTTP. Mikrokontroler ESP32 kemudian memproses perintah tersebut dan mengaktifkan *relay* untuk mengendalikan perangkat seperti lampu atau kipas angin. Penelitian ini juga menyediakan antarmuka grafis sederhana di aplikasi Android untuk memantau status perangkat. Hasil uji coba dilakukan dalam kondisi jaringan stabil dan tidak stabil. Dalam kondisi jaringan stabil, tingkat keberhasilan mencapai 92%, namun turun signifikan pada kondisi jaringan lambat. Penelitian ini menyimpulkan bahwa sistem *voice recognition* berbasis Android dan mikrokontroler ESP32 memiliki potensi besar untuk diterapkan pada sistem *smart home*, terutama karena kemudahan pengembangan antarmuka dan integrasi *API* yang sudah tersedia secara bawaan pada sistem operasi Android [6].

Secara keseluruhan, dapat dilihat bahwa perkembangan perangkat *smart home* terjadi cukup signifikan, karena perubahan gaya hidup dan trend manusia yang semakin minat terhadap hal yang praktis dan efisien. Mayoritas penelitian perangkat smart home masih mengandalkan jaringan internet sebagai komunikasi utama ataupun proses pengolahan suara, maka dari itu diharapkan penelitian yang akan dilakukan ini dapat memberikan kelebihan yang berbeda dari perangkat- perangkat smart home yang ada saat ini.

2.2 Landasan Teori

2.2.1 *Internet of Things*

Internet of Things (IoT) adalah konsep dimana berbagai perangkat fisik saling terhubung melalui jaringan internet untuk saling bertukar data dan melakukan pengendalian secara otomatis tanpa intervensi manusia secara langsung. Perangkat-perangkat ini dilengkapi dengan sensor, aktuator, dan teknologi komunikasi yang memungkinkan mereka untuk mengumpulkan dan mengirimkan informasi ke sistem pusat atau antarperangkat lainnya. IoT telah diterapkan di berbagai sektor, mulai dari industri manufaktur, kesehatan, pertanian, hingga sistem *smart home.*

Gambar 1. *Internet of Things*

Dalam konteks sistem *smart home*, IoT berperan sebagai fondasi utama dalam mengintegrasikan berbagai perangkat rumah tangga seperti lampu, kamera keamanan, *thermostat*, dan peralatan elektronik lainnya. Setiap perangkat terhubung melalui jaringan lokal atau internet dan pengendaliannya melalui aplikasi *smartphone* atau antarmuka berbasis *web*. Dengan adanya IoT, sistem dapat diatur untuk merespons kondisi lingkungan secara otomatis, misalnya menyalakan lampu saat penghuni masuk ruangan atau mengirim peringatan saat sensor mendeteksi aktivitas mencurigakan [11].

Penerapan IoT tidak lepas dari tantangan, seperti masalah keamanan data, interoperabilitas antarperangkat, dan konsumsi daya. Oleh karena itu, pengembangan sistem IoT dalam tugas akhir memerlukan perencanaan arsitektur yang matang, pemilihan protokol komunikasi yang tepat seperti *MQTT* atau HTTP, serta pertimbangan efisiensi energi dari perangkat yang digunakan. Dengan pendekatan yang tepat, IoT dapat memberikan solusi inovatif yang berdampak nyata pada kehidupan sehari-hari dan mendukung pengembangan teknologi yang berkelanjutan [7].

2.2.2 *Smart Home*

Smart home adalah konsep hunian yang mengintegrasikan teknologi informasi dan komunikasi untuk mengendalikan berbagai perangkat rumah secara otomatis dan efisien. Dengan menggunakan jaringan internet dan sensor-sensor pintar, sistem *smart home* memungkinkan pengendalian lampu, suhu ruangan, keamanan, hingga peralatan elektronik lainnya melalui *smartphone* menggunakan *web* atau aplikasi. Teknologi ini tidak hanya memberikan kenyamanan, tetapi juga meningkatkan efisiensi energi dan keamanan penghuni rumah [1].

Gambar 2. Smart Home

Dalam pengembangan sistem *smart home*, berbagai teknologi *seperti Internet of Things* (IoT), *Artificial Intelligence* (AI), dan *cloud computing* memainkan peran penting. Sistem ini dirancang agar mampu beradaptasi dengan kebiasaan pengguna serta memberikan respons secara *real-time* berdasarkan data yang

dikumpulkan dari lingkungan. Implementasi *smart home* sangat relevan dengan perkembangan kota pintar *(smart city),* dimana integrasi teknologi pada skala rumah tangga menjadi bagian dari ekosistem perkotaan yang lebih luas dan berkelanjutan [10].

2.2.3 *Raspberry Pi*

Raspberry Pi adalah komputer mini berukuran kecil yang dikembangkan oleh *Raspberry Pi* Foundation dengan tujuan awal untuk pendidikan dalam bidang komputer dan pemrograman. Meskipun ukurannya kompak dan harganya terjangkau, *Raspberry Pi* memiliki kemampuan yang cukup mumpuni untuk menjalankan berbagai aplikasi, mulai dari komputasi dasar hingga proyek-proyek berbasis *Internet of Things* (IoT), otomasi, dan kecerdasan buatan. Perangkat ini dilengkapi dengan prosesor ARM, port USB, HDMI, GPIO *(General Purpose Input Output)*, serta konektivitas *Wi-Fi* dan *Bluetooth*, yang menjadikannya sangat fleksibel untuk berbagai eksperimen dan pengembangan sistem tertanam *(embedded systems)* [13].

Gambar 3. *Raspberry Pi* 4 Model B

Dalam konteks tugas akhir, *Raspberry Pi* dapat digunakan sebagai otak dari sistem otomatisasi seperti *smart home*, sistem *monitoring* lingkungan, atau pengenalan suara. Dukungan terhadap berbagai bahasa pemrograman seperti *Python* dan *C++*, serta kompatibilitas dengan sistem operasi *Linux* seperti *Raspbian*, menjadikan *Raspberry Pi* sangat ideal untuk penelitian dan pengembangan sistem terintegrasi. Selain itu, komunitas pengguna *Raspberry Pi* yang luas dan dokumentasi yang lengkap menjadi keunggulan tersendiri yang mempermudah proses belajar dan implementasi proyek [14].

2.2.4 *Relay Module*

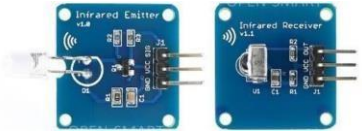
Relay adalah output yang berfungsi sebagai saklar untuk perangkat lain. *Relay* dikendalikan dengan tegangan dari pin *Raspberry Pi* sehingga dapat melakukan *switch*. Ada 3 koneksi utama yaitu COM sebagai input dari perangkat lain, NC *(Normally Close)* pada keadaan biasa COM akan terhubung ke pin NC, dan NO *(Normally Open)* pada keadaan biasa tidak terhubung, namun saat *relay* mendapat tegangan dari *arduino* maka COM akan berpindah dari NC dan terhubung dengan NO [15].

Gambar 4. *Relay Module*

Karena *relay* merupakan salah satu jenis dari saklar, maka istilah *Pole* dan *Throw* yang dipakai dalam saklar juga berlaku pada *relay* [16]. *Pole* merupakan banyaknya kontak *(contact)* yang dimiliki oleh sebuah *relay*, sedangakan *throw* adalah banyaknya kondisi yang dimiliki oleh sebuah kontak *(contact).*

2.2.5 *Infra-Red Module*

Modul infrared (IR) adalah perangkat elektronik yang menggunakan gelombang elektromagnetik inframerah untuk mendeteksi objek, mengirimkan data, atau membaca sinyal dari *remote control*. Modul ini umumnya terdiri dari dua komponen utama, yaitu *IR transmitter* (pemancar) dan *IR receiver* (penerima). Pemancar mengirimkan sinyal cahaya inframerah dalam bentuk pulsa, yang kemudian dapat diterima oleh penerima jika tidak terhalang oleh objek. Karena sinyal inframerah tidak terlihat oleh mata manusia dan memiliki jangkauan pendek, modul ini banyak digunakan untuk komunikasi jarak dekat serta sistem deteksi objek atau Gerakan [17].



Gambar 5. *Infra-Red Module*

Dalam proyek tugas akhir, modul IR banyak diaplikasikan untuk sistem otomatisasi, seperti deteksi kehadiran, penghitung orang, atau kendali perangkat melalui *remote*. Modul ini bekerja dengan prinsip pantulan: ketika sinyal IR yang dipancarkan mengenai suatu objek, sinyal akan dipantulkan kembali dan diterima oleh sensor. Data dari sensor tersebut kemudian dapat diolah oleh mikrokontroler seperti *Arduino* atau *Raspberry Pi* [18]. Keunggulan modul IR adalah konsumsi daya rendah, harga terjangkau, dan kemudahan integrasi dalam sistem *embedded*.

2.2.6 *Faster Whisper*

Faster Whisper adalah versi teroptimasi dari model *Whisper*, yaitu model kecerdasan buatan untuk mengubah suara menjadi teks *(speech-to-text).* Dibuat menggunakan *CTranslate2, Faster Whisper* dirancang agar lebih cepat, lebih ringan, dan lebih efisien dibandingkan *Whisper* asli. Dengan optimasi ini, proses transkripsi dapat berjalan dengan kecepatan lebih tinggi, bahkan ketika menggunakan perangkat dengan spesifikasi tidak terlalu kuat seperti CPU standar.

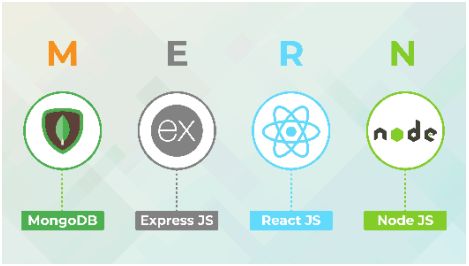
Gambar 6. *Speech to Text*

Selain kecepatan, *Faster Whisper* tetap mempertahankan kualitas hasil transkripsi yang akurat dan mendukung banyak bahasa. Teknologi ini banyak digunakan pada aplikasi AI, layanan transkripsi otomatis, serta sistem yang membutuhkan pemrosesan audio *real-time*. Karena lebih hemat memori dan lebih mudah di-*deploy*, *Faster Whisper* menjadi pilihan populer untuk produksi maupun proyek personal yang membutuhkan konversi audio yang cepat dan stabil.

2.2.7 *MERN Stack*

MERN Stack adalah kumpulan teknologi pengembangan *web* yang terdiri dari *MongoDB*, *Express.js*, *React.js*, dan *Node.js*. Keempatnya digunakan bersama untuk membangun aplikasi *web full-stack* berbasis JavaScript. *MongoDB* berperan sebagai *database* NoSQL, *Express.js* sebagai *framework backend*, *React.js* sebagai

library frontend, dan *Node.js* sebagai *runtime* yang menjalankan JavaScript di sisi *server*. Karena semuanya menggunakan JavaScript, proses pengembangan menjadi lebih konsisten dan efisien.



Gambar 7. *MERN Stack*

Stack ini populer karena kemudahan *scaling*, komunitas yang besar, serta kemampuan untuk membangun aplikasi modern seperti *dashboard*, sistem *e-commerce*, dan aplikasi *real-time*. MERN juga mendukung arsitektur *API* yang fleksibel, performa cepat, dan kemampuan untuk membuat UI interaktif menggunakan *React*. Kombinasi ini menjadikan *MERN Stack* sebagai salah satu pilihan utama untuk developer yang ingin membangun aplikasi *web* lengkap dari *frontend* hingga *backend* dengan satu bahasa pemrograman.

2.2.8 *MQTT Protocol*

MQTT (*Message Queuing Telemetry Transport*) adalah protokol komunikasi ringan yang digunakan untuk mengirim dan menerima data antar perangkat, terutama pada sistem *Internet of Things* (IoT). *MQTT* dirancang agar efisien, cepat, dan tetap bekerja dengan baik pada jaringan yang lambat atau tidak stabil. Karena sangat hemat *bandwidth*, *MQTT* ideal untuk perangkat kecil seperti sensor, modul IoT, dan mikrokontroler.

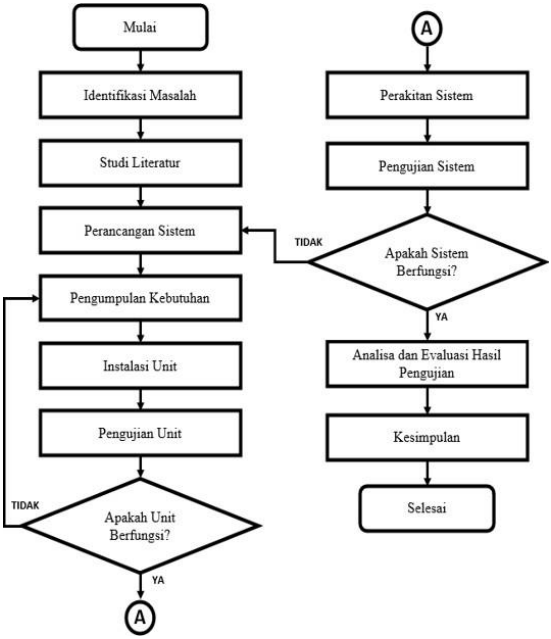
Gambar 8. *MQTT*

Cara kerja *MQTT* menggunakan konsep *publish–subscribe*. Artinya, perangkat *publisher* mengirim pesan ke sebuah "topik", lalu perangkat *subscriber* menerima pesan tersebut jika mereka berlangganan topik yang sama. Semua proses ini diatur oleh *MQTT broker* (misalnya *Mosquitto*), yang bertugas menyalurkan pesan antar perangkat. Dengan sistem ini, komunikasi antar perangkat menjadi lebih fleksibel, sederhana, dan *scalable* untuk banyak perangkat IoT.

Bab 3. Metodologi Penelitian

3.1 Alur Penelitian

Dalam merancang sebuah sistem, diperlukan suatu metode pelaksanaan agar rancangan penelitian yang dibuat terstruktur dan jelas. Tahapan-tahapan dalam pelaksanaan ini tentunya harus efektif dan efisien. sehingga tidak membuang banyak biaya dan waktu secara percuma. Gambar 9 merupakan *Flowchart* alur Penelitian ini.



Gambar 9. Alur Penelitian

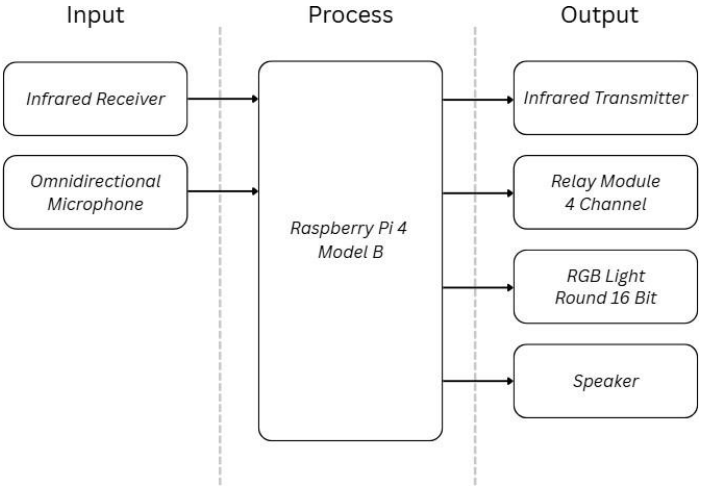
Perancangan penelitian ini akan dijelaskan secara lengkap pada bagian ini. Sistem *smart home* pada penelitian ini menggunakan *Raspberry Pi* 3 Model B+ sebagai mikrokontroler yang akan memproses semua perintah untuk mengendalikan perangkat elektronik secara saklar (*relay*) ataupun perintah *remote control* (*infrared*). *Raspberry Pi* ini juga akan digunakan sebagai *server* untuk *website* yang akan digunakan sebagai antarmuka

antara mikrokontroler dengan manusia dengan konsep lokal sehingga tidak bergantung pada jaringan internet.

Beberapa komponen pendukung yang akan digunakan pada penelitian ini adalah Modul *Relay* 5V sebagai saklar elektronik, Modul *Infrared Transmitter* sebagai media perintah *remote control* ke perangkat elektronik, Modul *Infrared Receiver* untuk merekam perintah *remote control* dari masing-masing perangkat elektronik yang akan diberikan perintah

3.2 Blok Diagram Sistem

Sistem bekerja dengan menerima berbagai masukan dari lingkungan melalui dua perangkat input utama, yaitu *Infrared Receiver* dan *Omnidirectional Microphone*. Infrared *Receiver* berfungsi menangkap sinyal IR dari *remote* perangkat elektronik seperti TV atau AC, kemudian *Raspberry Pi* melakukan *decoding* untuk mengenali jenis perintah yang diterima. Sementara itu, mikrofon *omnidirectional* digunakan untuk menangkap suara atau perintah berbasis audio yang mungkin digunakan untuk mengaktifkan fungsi tertentu. Kedua masukan ini dipantau secara terus-menerus oleh *Raspberry Pi* sehingga sistem selalu siap merespons setiap sinyal yang masuk.



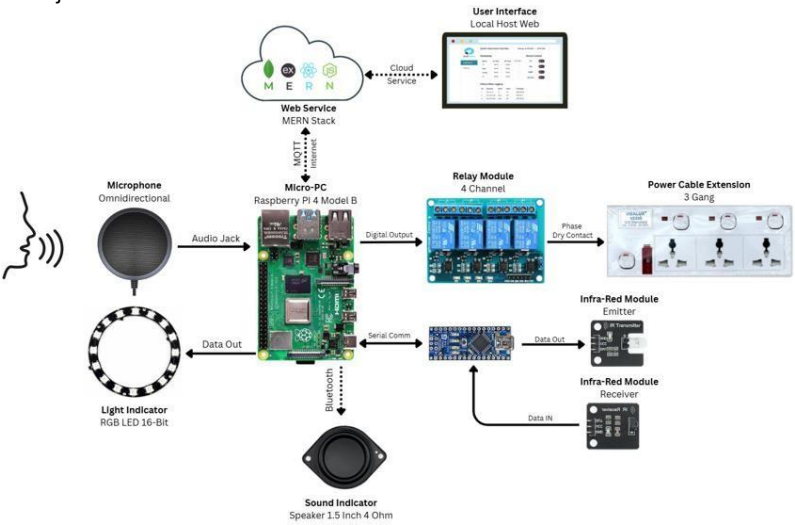
Gambar 10. Blok Diagram Sistem

Setelah *Raspberry Pi* berhasil memproses dan mengidentifikasi perintah, sistem akan mengaktifkan output yang sesuai. Jika perintah berupa pengendalian perangkat elektronik berbasis IR, *Raspberry Pi* mengirimkan kembali sinyal IR

melalui Infrared *Transmitter*. Jika perintah memerlukan kendali perangkat listrik fisik, modul *Relay* 4 *Channel* digunakan sebagai saklar elektronik untuk mengaktifkan atau menonaktifkan perangkat tersebut. Selain itu, RGB *Light Round* 16 Bit digunakan sebagai indikator visual untuk menampilkan status sistem, sedangkan speaker memberikan umpan balik audio. Dengan demikian, *Raspberry Pi* bertindak sebagai pusat kendali yang menghubungkan input dengan output dan memastikan setiap perintah dieksekusi dengan tepat.

3.3 Sistem Kendali

Sistem kendali pada gambar tersebut menunjukkan arsitektur terintegrasi yang memanfaatkan *Raspberry Pi 4* sebagai pusat pengolahan data. *Raspberry Pi* menerima input dari berbagai sensor, seperti mikrofon omnidireksional untuk menangkap suara dan modul *infra-red (emitter dan receiver)* untuk membaca kondisi lingkungan. Data dari sensor ini kemudian diproses dan diteruskan ke perangkat output, termasuk lampu indikator RGB LED serta speaker sebagai indikator suara. Komunikasi antarperangkat dilakukan melalui jalur data seperti *audio jack*, *serial communication*, dan *Bluetooth*, sehingga setiap komponen dapat bekerja secara sinkron.



Gambar 11. Sistem Kendali

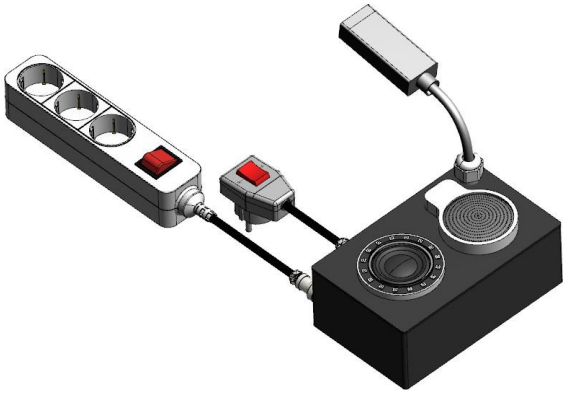
Selain itu, *Raspberry Pi* juga mengendalikan modul *relay 4-channel* untuk melakukan *switching* pada beban listrik yang terhubung ke *power cable extension.* Sistem ini terhubung dengan layanan *web* berbasis *MERN stack*, memungkinkan pengendalian dan pemantauan melalui antarmuka *web* lokal. Dengan demikian,

pengguna dapat mengakses status sistem maupun memberikan perintah melalui *dashboard* berbasis *cloud-service*, menjadikan keseluruhan rangkaian sebagai sistem kendali yang cerdas, terpusat, dan mudah dioperasikan.

3.4 Perangkat Keras

3.4.1 Desain Mekanikal

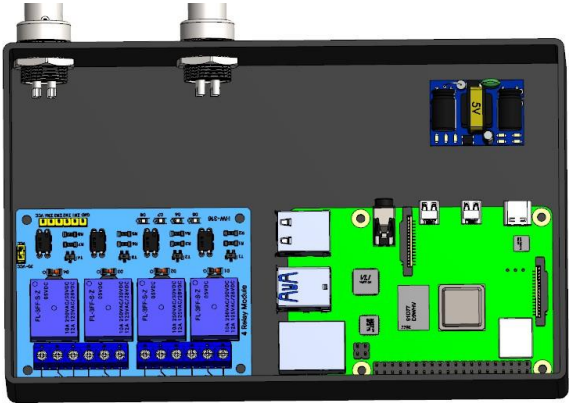
Gambar 12 menunjukkan desain mekanikal dari *box* unit terbuat dari plastik ABS yang memiliki dimensi 18x11.5x6.5cm, berfungsi sebagai wadah fisik seluruh perangkat sistem kendali. Pada bagian atas *box* terlihat beberapa komponen antarmuka luar, seperti ring LED RGB sebagai indikator visual, sebuah speaker sebagai indikator suara, serta modul sensor inframerah yang dipasang pada dudukan khusus. Desain ini dibuat agar setiap komponen dapat bekerja optimal sekaligus memberikan tampilan perangkat yang rapi, kokoh, dan ergonomis untuk penggunaan di lingkungan nyata.



Gambar 12. Mekanikal Box Unit

Gambar 13 memperlihatkan bagian dalam box unit, yaitu susunan perangkat elektronik yang ditempatkan dengan rapi di dalam *casing*. Terlihat *Raspberry Pi* sebagai pusat pengendali sistem yang dipasang di sisi kiri, sementara modul *relay* 4-*channel* berada di sisi kanan untuk melakukan *switching* beban listrik. Kabel-kabel masuk dari bagian bawah box dan dihubungkan langsung ke komponen utama, memastikan koneksi yang aman, terstruktur, dan mudah untuk

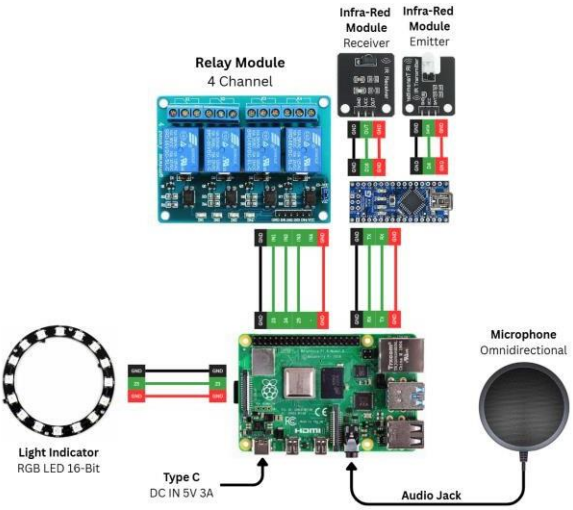
pemeliharaan. Desain interior ini mendukung keandalan sistem serta memudahkan instalasi maupun *troubleshooting*.



Gambar 13. Bagian Dalam Box Unit

3.4.2 Desain Electrikal

Gambar 14 memperlihatkan desain elektrikal dari sistem kendali, yaitu hubungan antar komponen elektronik yang terhubung ke *Raspberry Pi* sebagai pengendali utama. Pada bagian kiri, *light indicator* berupa *ring* RGB LED 16-bit disambungkan ke pin GPIO *Raspberry Pi* melalui jalur data, VCC, dan GND untuk menghasilkan indikator visual sesuai instruksi program.



Gambar 14. Desain Elektrikal

Tabel 2. Konfigurasi Pin

No	Perangkat	Fungsi	GPIO Raspi	Pin Fisik
1	*Relay 4 Channel*	IN1	GPIO23	Pin 16
2	*Relay 4 Channel*	IN2	GPIO24	Pin 18
3	*Relay 4 Channel*	IN3	GPIO25	Pin 22
4	*Relay 4 Channel*	IN4	GPIO26	Pin 37
5	*Relay 4 Channel*	VCC	5V	Pin 2
6	*Relay 4 Channel*	GND	GND	Pin 6
7	*IR Receiver*	OUT	GPIO20	Pin 38
8	*IR Receiver*	VCC	3.3V	Pin 1
9	*IR Receiver*	GND	GND	Pin 6
10	*IR Emitter*	IN	GPIO21	Pin 40
11	*IR Emitter*	VCC	3.3V	Pin 1
12	*IR Emitter*	GND	GND	Pin 6
13	*Arduino Nano*	TX (Raspi)	GPIO14	Pin 8
14	*Arduino Nano*	RX (Raspi)	GPIO15	Pin 10

15	*Arduino Nano*	GND	GND	Pin 6
16	*RGB LED Ring (WS2812)*	DATA	GPIO22	Pin 16
17	*RGB LED Ring*	VCC	5V	Pin 4
18	*RGB LED Ring*	GND	GND	Pin 6

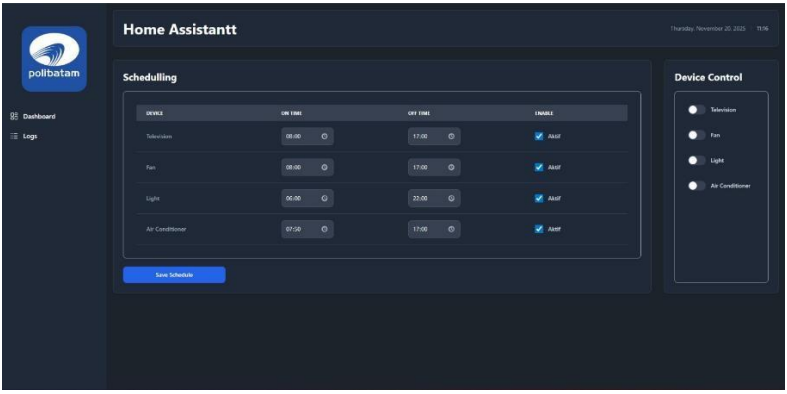
Di bagian kanan, mikrofon *omnidirectional* terhubung melalui *audio jack* untuk menangkap input suara. *Raspberry Pi* juga mendapatkan suplai daya melalui port USB Type-C dengan tegangan DC 5V 3A agar seluruh rangkaian bekerja stabil.

Pada bagian atas diagram, modul *relay 4-channel* terhubung ke beberapa pin GPIO sebagai output digital yang digunakan untuk mengendalikan perangkat listrik eksternal. Selain itu, terdapat dua modul *infra-red (receiver dan emitter)* yang masing-masing dihubungkan ke *Raspberry Pi* melalui jalur data serta catu daya 5V dan GND. Modul ini bekerja sama untuk mendeteksi objek atau pergerakan menggunakan sinyal infra-merah. Secara keseluruhan, gambar ini menunjukkan skema wiring lengkap yang memastikan setiap sensor, aktuator, dan modul bekerja selaras dalam sistem kendali berbasis *Raspberry Pi*.

3.5 Perangkat Lunak

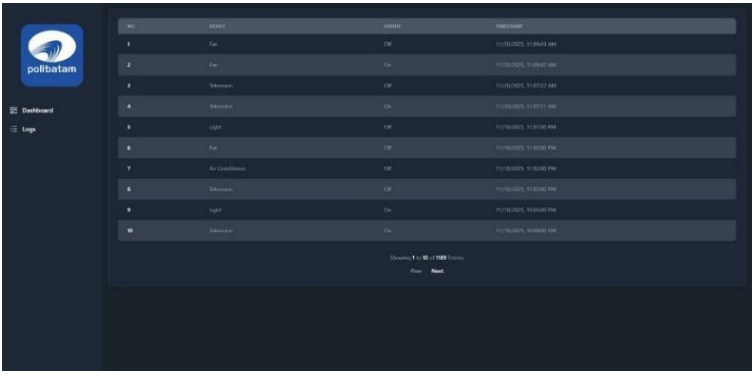
3.5.1 *Web Application*

Aplikasi *web* pada gambar tersebut menampilkan antarmuka *Home Assistant* yang berfungsi untuk mengatur penjadwalan serta kendali perangkat rumah secara otomatis. Pada halaman *dashboard* di Gambar 15, pengguna dapat mengatur waktu *ON* dan *OFF* untuk berbagai perangkat seperti televisi, kipas, lampu, dan AC. Setiap perangkat juga dilengkapi dengan opsi *enable* untuk mengaktifkan atau menonaktifkan jadwal yang telah ditentukan. Selain penjadwalan, terdapat pula panel *Device Control* di sisi kanan yang memungkinkan pengguna menyalakan atau mematikan perangkat secara manual melalui tombol toggle. Berikut link untuk mengakses *web* yang telah kami buat http://polibatamassistant.xyz/



Gambar 15. Tampilan *Dashboard Web*

Sementara itu, halaman riwayat di Gambar 16 menampilkan daftar aktivitas perangkat yang tersimpan dalam bentuk tabel. Informasi yang ditampilkan meliputi nama perangkat, status (*ON* atau *OFF*), serta timestamp ketika perubahan terjadi. Riwayat ini membantu pengguna memantau kapan suatu perangkat dinyalakan atau dimatikan, sehingga memudahkan evaluasi penggunaan energi dan aktivitas sistem secara keseluruhan. Dengan tampilan yang rapi dan konsisten, aplikasi *web* ini memberikan kemudahan dalam mengelola otomatisasi rumah secara efisien.

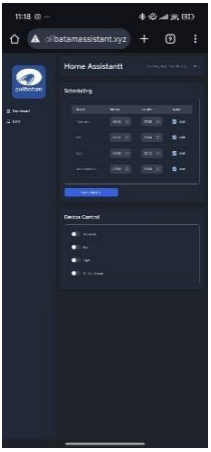


Gambar 16. Tampilan Halaman Riwayat

3.5.2 *Web Application Mobile Version*

Web mobile version adalah versi tampilan sebuah aplikasi atau situs *web* yang telah dioptimalkan khusus untuk perangkat *mobile* seperti *smartphone* dan tablet. Versi ini dibuat agar antarmuka tetap nyaman digunakan pada layar yang lebih kecil, dengan tata letak yang lebih sederhana, tombol yang lebih besar, serta navigasi yang lebih mudah diakses menggunakan sentuhan (*touchscreen*). Tujuan utama *web mobile version* adalah memastikan pengguna mendapatkan pengalaman yang sama efektifnya seperti saat mengakses melalui komputer, meskipun ruang tampilan lebih terbatas. Biasanya, versi ini menggunakan desain responsif yang secara otomatis menyesuaikan tampilan sesuai ukuran layar perangkat.

Pada versi *mobile* aplikasi *web* Gambar 17, tampilan *dashboard* dirancang agar tetap responsif dan mudah digunakan pada layar yang lebih kecil. Elemen-elemen utama seperti penjadwalan perangkat, tombol *Save Schedule*, serta panel *Device Control* tetap tersusun rapi dan proporsional sehingga pengguna dapat melakukan pengaturan dengan nyaman. Meskipun ruang tampilan lebih terbatas, informasi tetap disajikan secara jelas dan fungsi utama tetap dapat diakses tanpa hambatan.



Gambar 17. Tampilan *Dashboard* Mobile Version

Halaman riwayat pada versi *mobile* Gambar 18 juga menampilkan data aktivitas perangkat dalam format tabel yang disesuaikan untuk layar *smartphone*. Pengguna dapat melihat daftar perangkat, status, serta waktu perubahan secara ringkas dan mudah dibaca. Adaptasi *layout* ini memastikan bahwa seluruh fitur pada versi desktop tetap dapat diakses secara optimal melalui perangkat *mobile*,

sehingga pengguna dapat memonitor dan mengendalikan sistem otomatisasi rumah kapan saja dan di mana saja.

Gambar 18. Tampilan Halaman Riwayat Mobile Version

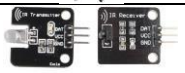
3.6 Anggaran Biaya, Alat dan Bahan

Anggaran biaya tersebut disusun untuk pengadaan perangkat pintar *home assistant* yang mencakup komponen pemrosesan, input, output, serta pendukung kelistrikan dan instalasi. Sistem ini menggunakan unit pemroses utama sebagai pusat kendali, didukung oleh mikrokontroler tambahan untuk menangani fungsi tertentu. Perangkat input seperti mikrofon dan sensor digunakan untuk menerima perintah dan mendeteksi kondisi lingkungan, sementara perangkat output seperti speaker, lampu indikator, dan modul *relay* berfungsi untuk memberikan respons dan mengendalikan peralatan rumah tangga. Selain itu, anggaran juga mencakup modul komunikasi audio, penyimpanan data, catu daya, serta berbagai kabel, konektor, dan aksesori instalasi guna memastikan sistem dapat bekerja dengan stabil, aman, dan terintegrasi dengan baik dalam lingkungan rumah pintar.

Tabel 3. Anggaran Biaya

No.	*Item*	*Qty*	*Unit*	*Unit Price*	*Sub Total*	*Picture*
1	*Raspberry Pi 4*	1	Pcs	Rp2.000.000	Rp2.000.000	

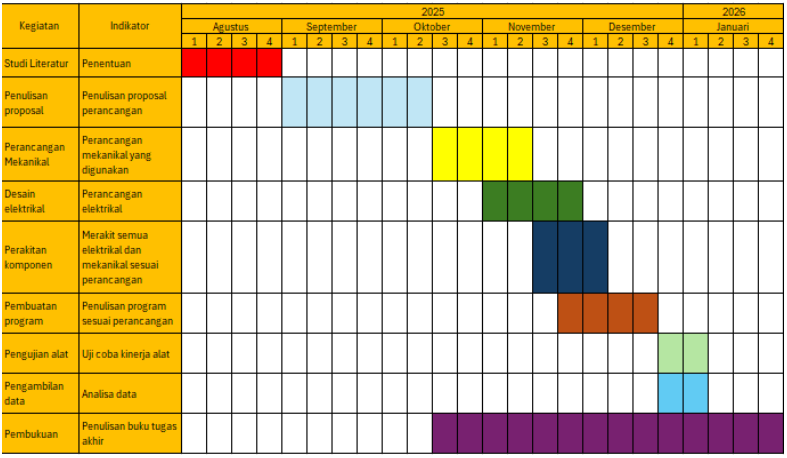
2	*Omnidirectional Microphone*	1	Pcs	Rp100.000	Rp100.000	
3	*Mp3 Module*	1	Pcs	Rp30.000	Rp30.000	
4	*RGB Light*	1	Pcs	Rp25.000	Rp25.000	
5	*Connector GX16*	2	Pcs	Rp10.000	Rp20.000	
6	*Speaker 1.5 Inch*	1	Pcs	Rp50.000	Rp50.000	
7	*Universal Box X6*	1	Pcs`	Rp20.000	Rp20.000	
8	*Relay Module 4 Channel*	1	Pcs	Rp40.000	Rp40.000	
9	Power Steker	1	Pcs	Rp10.000	Rp10.000	
10	Baut	1	Lot	Rp9.000	Rp9.000	
11	*Extension* 4 Gang	1	Pcs	Rp65.000	Rp65.000	
12	*Flexible Fitting Hose*	1	Pcs	Rp18.000	Rp18.000	
13	*Cable AWG22*	3	Meter	Rp3.000	Rp9.000	
14	*Cable 0.75mm 4 Core*	3	Meter	Rp25.000	Rp75.000	
15	Micro SD Card	1	Pcs	Rp100.000	Rp100.000	

16	*Cable Gland*	1	Pcs	Rp3.000	Rp3.000	
17	*Audio Jack to USB Converter*	1	Pcs	Rp100.000	Rp100.000	
18	*Power Adaptor 5V 3A*	1	Pcs	Rp25.000	Rp25.000	
19	*IR Module*	2	Pcs	Rp15.000	Rp30.000	
20	*Arduino Nano*	1	Pcs	Rp45.000	Rp45.000	
21	*3D Print Part*	1	Lot	Rp50.000	Rp50.000	
				Total	Rp2.824.000	

3.7 Jadwal Pelaksanaan

Berikut adalah jadwal pelaksanaan proyek akhir dengan judul Sistem Pengendalian Perangkat *Smart home* Dengan Perintah Suara Menggunakan *Raspberry Pi dan Smartphone*

Tabel 4. Jadwal Pelaksanaan



Bab 4. Hasil dan Pembahasan

Gambar 19 menampilkan perangkat utama hasil penelitian yang tampak terdiri dari sebuah modul kendali dengan *box* hitam, kabel *extension* dan steker listrik. *Box control unit* tampak dilengkapi dengan *speaker*, indikator berbentuk *RGB Light 16-bit* bulat, serta IR *holder* dengan stand yang fleksible untuk perubah arah heading.

Gambar 19. Hasil Penelitian

Dua modul yang tidak terhubung yang ditampilkan dalam gambar menunjukkan bahwa sistem ini memerlukan koneksi fisik untuk menghubungkan modul utama dengan unit-unit yang dikendalikannya. Pada bagian kanan bawah terdapat steker berwarna krem yang dapat digunakan sebagai koneksi sumber listrik. Secara keseluruhan, Gambar 19 menunjukkan bentuk fisik prototipe atau unit hasil penelitian yang sudah dapat beroperasi dan telah terintegrasi dengan komponen-komponen penunjangnya.

Gambar 20 memperlihatkan tampilan keseluruhan *(full unit)* dari sistem yang telah dirakit lengkap. Modul kendali berwarna hitam terlihat sudah terhubung langsung dengan kabel *extension* melalui *connector*, menandakan bahwa konfigurasi pada gambar ini adalah representasi dari kondisi perangkat saat siap

digunakan. Semua konektor terpasang pada port yang sesuai, sehingga antarmuka pengguna dan modul aktuator dapat bekerja sebagai satu sistem terpadu.

Gambar 20. Tampilan Full Unit

Pada gambar ini terlihat bahwa perangkat sudah berada dalam bentuk final atau mendekati bentuk final, di mana setiap komponen berada pada tempatnya dan sistem terlihat rapi, fungsional, serta siap diuji atau dioperasikan. Hal ini menunjukkan bahwa proses perakitan sudah selesai dan prototipe telah memenuhi struktur fisik sebagaimana direncanakan dalam desain penelitian.

Gambar 21. Tampilan Pengakabelan Unit

Pada Gambar 21 terlihat *Raspberry Pi 4 Model B* sebagai pusat kendali, terhubung dengan modul *relay* 4-channel untuk mengatur beban listrik eksternal seperti stop-kontak atau perangkat rumah tangga. Kabel-kabel daya dan sinyal menghubungkan *relay* ke catu daya internal dan komponen lain. Pada bagian penutup kotak, tampak sebuah speaker kecil sebagai indikator suara, serta sebuah *omnidirectional microphone* yang terhubung ke *Raspberry*. Ada driver audio yang mengendalikan speaker untuk memainkan suara mp3. Secara keseluruhan, kotak elektronik ini merupakan prototipe sistem kendali pintar yang mengintegrasikan audio input/output, kendali *relay*, komunikasi IR, dan komputasi berbasis *Raspberry Pi* dalam satu unit tertutup.

4.1 Pengujian Sistem

4.1.1 Infra-Red

Pengujian sistem infrared pada perangkat AC dilakukan dengan merekam sinyal infrared dari *remote* AC menggunakan modul *infrared receiver*, kemudian mengirimkan kembali sinyal tersebut melalui *infrared emitter* saat perintah diberikan. *Arduino* mampu memproses dan memancarkan ulang kode infrared dengan format yang sesuai sehingga unit AC dapat merespons perintah dengan baik. Hasil pengujian menunjukkan bahwa AC berhasil dinyalakan secara konsisten selama emitter diarahkan ke sensor AC dan berada dalam jarak efektif tanpa halangan.

Gambar 22. Pengujian *Infra-Red* AC

Pengujian *infrared* pada TV dilakukan dengan metode yang sama, yaitu menangkap kode sinyal tombol daya dari *remote* TV menggunakan infrared receiver dan memancarkannya kembali melalui infrared emitter. *Arduino* berhasil mengirimkan sinyal infrared yang dikenali oleh TV sehingga perangkat dapat menyala sesuai perintah. Dari hasil pengujian, sistem bekerja secara stabil dan responsif pada berbagai percobaan selama posisi emitter dan jarak pengujian berada dalam kondisi optimal.

Gambar 23. Pengujian *Infra-Red* TV

4.1.2 *Relay* Control

Pengujian modul *relay* menggunakan *Raspberry Pi* dilakukan dengan menghubungkan pin GPIO *Raspberry Pi* ke input *relay* dan memberikan sinyal logika *HIGH* atau *LOW* melalui program. Setiap kanal *relay* diuji secara bergantian untuk memastikan bahwa *relay* dapat aktif dan nonaktif sesuai perintah yang diberikan. Pada saat GPIO berada pada kondisi aktif, *relay* berhasil menghubungkan beban listrik, sedangkan pada kondisi nonaktif *relay* memutuskan arus dengan baik. Hasil pengujian menunjukkan bahwa modul *relay* dapat dikendalikan secara stabil oleh *Raspberry Pi* dan mampu berfungsi sesuai dengan logika kendali yang telah dirancang.

Pengujian modul *relay* menggunakan *Raspberry Pi* dilakukan dengan memanfaatkan dua kanal (gang) *relay*, yaitu satu gang untuk mengendalikan kipas angin dan satu gang untuk mengendalikan lampu. *Raspberry Pi* memberikan sinyal logika melalui pin GPIO untuk mengaktifkan dan menonaktifkan masing-masing

relay sesuai perintah yang diberikan. Hasil pengujian menunjukkan bahwa *relay* mampu menyalakan dan mematikan kipas angin serta lampu secara terpisah dan sesuai dengan perintah sistem. Selama pengujian, *relay* bekerja secara stabil tanpa gangguan, dan beban listrik dapat dikendalikan dengan baik.

Gambar 24. Pengujian *Relay* Output Lampu

Berdasarkan hasil pengujian, penggunaan dua gang *relay* untuk mengendalikan kipas angin dan lampu terbukti berjalan dengan baik menggunakan *Raspberry Pi*. Sistem ini menunjukkan bahwa *Raspberry Pi* dapat diandalkan sebagai pengendali perangkat listrik dalam sistem otomasi berbasis IoT, serta mampu meningkatkan efisiensi dan fleksibilitas pengendalian perangkat elektronik.

Gambar 25. Pengujian *Relay* Output Kipas

4.1.3 Voice Command

Sistem *voice command* pada perangkat diuji untuk memastikan kemampuan sistem dalam menerima, memproses, dan mengeksekusi perintah suara dari pengguna. Pada tahap awal, sistem berada pada kondisi siaga untuk menerima perintah suara yang ditandai dengan indikator *LED* berwarna biru. Pada kondisi ini, mikrofon aktif dan sistem melakukan proses listening untuk menangkap suara pengguna secara *real-time.*

Gambar 26. Sistem *Listening*

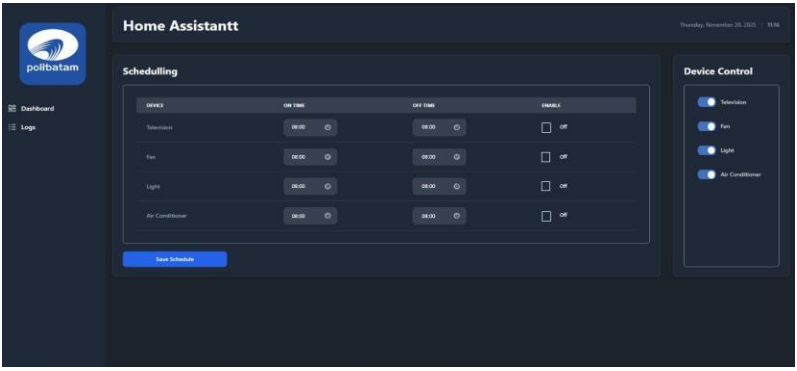
Setelah perintah suara berhasil diterima, sistem masuk ke tahap pemrosesan yang ditandai dengan perubahan warna *LED* menjadi hijau. Pada tahap ini, data suara diproses oleh sistem untuk dilakukan pengenalan dan interpretasi perintah sesuai dengan fungsi yang telah diprogram. Proses ini melibatkan analisis sinyal suara hingga sistem menentukan aksi yang akan dijalankan berdasarkan perintah yang diterima.

Gambar 27. Sistem *Execute*

Berdasarkan hasil pengujian, sistem *voice command* mampu beralih status dari mode listening ke mode proses secara responsif dan akurat. Indikator *LED* berfungsi dengan baik sebagai penanda visual kondisi sistem, sehingga pengguna dapat dengan mudah mengetahui status perangkat. Dengan demikian, sistem *voice command* dapat diandalkan sebagai metode interaksi yang efektif dalam mendukung pengendalian perangkat berbasis IoT.

4.1.4 *Web* Control

Pengujian *web control* dilakukan untuk memastikan sistem dapat mengendalikan perangkat elektronik melalui antarmuka *web* menggunakan tombol *toggle ON/OFF*. Perangkat yang dikendalikan meliputi kipas angin, lampu, AC, dan TV. Setiap *toggle switch* pada halaman web terhubung dengan sistem kendali yang berjalan pada *Raspberry Pi*, sehingga perubahan status *toggle* akan mengirimkan perintah secara *real-time* untuk mengaktifkan atau menonaktifkan perangkat terkait.

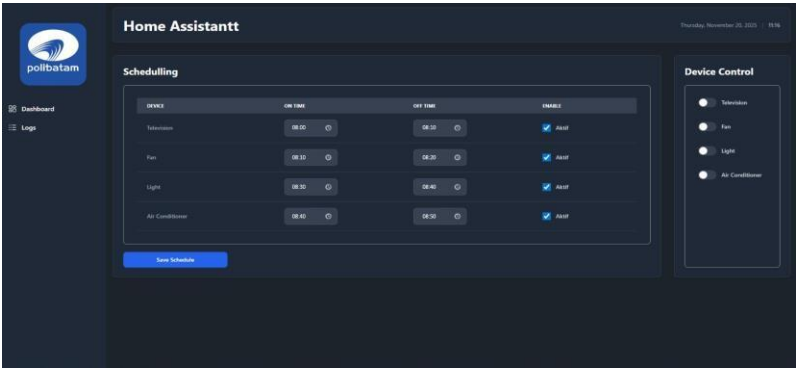


Gambar 28. *Web* Control

Berdasarkan hasil pengujian, seluruh *toggle switch* pada *web control* berfungsi dengan baik dan mampu mengendalikan kipas angin, lampu, AC, dan TV sesuai dengan perintah pengguna. Sistem menunjukkan respon yang cepat dan stabil, serta status perangkat dapat ditampilkan secara akurat pada antarmuka *web*. Hal ini membuktikan bahwa *web control* efektif digunakan sebagai media pengendalian perangkat dalam sistem otomasi berbasis IoT.

4.1.5 Web Scheduling

Pengujian fitur *web scheduling* dilakukan untuk memastikan sistem mampu mengatur waktu operasional perangkat elektronik secara otomatis melalui antarmuka *web*. Penjadwalan diterapkan pada empat perangkat, yaitu kipas angin, lampu, AC, dan TV. Pengguna dapat menentukan waktu *ON* dan *OFF* untuk masing-masing perangkat, kemudian data jadwal tersebut diproses oleh sistem dan dieksekusi secara otomatis oleh *Raspberry Pi* sesuai waktu yang telah ditentukan.

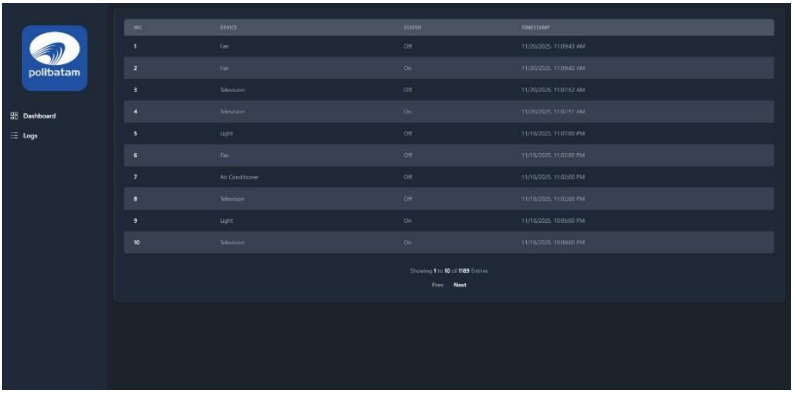


Gambar 29. *Web Scheduling*

Berdasarkan hasil pengujian, sistem *web scheduling* bekerja dengan baik dan mampu menyalakan serta mematikan kipas angin, lampu, AC, dan TV secara tepat waktu sesuai jadwal yang diatur. Sistem berjalan stabil tanpa gangguan, dan perubahan jadwal dapat diterapkan dengan baik melalui antarmuka *web*. Hal ini menunjukkan bahwa fitur *web scheduling* efektif dalam mendukung otomasi perangkat dan meningkatkan efisiensi pengendalian dalam sistem IoT.

4.1.6 Web Logging

Pengujian fitur *web logging* dilakukan untuk memastikan sistem mampu mencatat setiap aktivitas pengendalian perangkat melalui antarmuka *web*. Aktivitas yang dicatat meliputi perintah *ON* dan *OFF* pada kipas angin, lampu, AC, dan TV, baik yang dilakukan secara manual melalui *web control* maupun otomatis melalui fitur *scheduling*. *Data log* disimpan oleh sistem dan ditampilkan pada halaman *web* dalam bentuk informasi waktu, jenis perangkat, dan status perintah.



Gambar 30. *Web Logging*

Hasil pengujian menunjukkan bahwa sistem *web logging* bekerja dengan baik dan mampu merekam seluruh aktivitas pengendalian perangkat secara akurat dan berurutan. Informasi *log* dapat diakses dengan mudah melalui antarmuka web dan digunakan sebagai bahan pemantauan maupun evaluasi kinerja sistem. Dengan adanya fitur *web logging*, sistem menjadi lebih terkontrol, transparan, dan andal dalam mendukung pengelolaan perangkat berbasis IoT.

4.2 Hasil Pengujian Sistem

4.2.1 Infra-Red

Tabel 5 menunjukkan hasil pengujian sistem kendali perangkat menggunakan sinyal *infra-red* (IR). Pada pengujian ini, beberapa perintah seperti TV*_POWER*, AC*_ON*, dan AC*_OFF* diuji untuk melihat apakah perangkat menerima sinyal sesuai harapan. Kolom "Aksi yang Diharapkan" menjelaskan respon ideal dari perangkat, sedangkan kolom "*Status LED IR*" menunjukkan apakah modul IR mengirim sinyal (LED berkedip) atau tidak. Kolom "Respon Serial" memberikan informasi apakah perintah dikenali dan diproses oleh sistem, sehingga dapat diketahui apakah sinyal IR terkirim dengan benar.

Tabel 5. Pengujian Infra-Red

No	Device	Perintah	Aksi yang Diharapkan	Status LED IR	Respon Serial	Hasil Pengujian
1	TV	*TV_POWER*	*Arduino* mengirim raw IR untuk power on	LED IR berkedip	"Action: TV POWER"	*TV On/Off (toggle)*
2		*TV_PREVIOUS*	*Arduino* mengirim raw IR untuk previous sebelumnya	LED IR berkedip	"Action: TV NEXT"	*TV Next Channel*

3		*TV_NEXT*	*Arduino* mengirim raw IR untuk next channel	LED IR berkedip	"Action: TV PREVOIUS"	*TV Previous Channel*
4		Perintah tidak dikenal	Tidak mengirim IR	LED IR tidak berkedip	"Error: Unknown command."	Tidak ada reaksi
5	AC	*AC_ON16*	*Arduino* mengirim raw IR untuk power on	LED IR berkedip	"Action: TURNING AC ON AND SET TO 16 CELCIUS"	*AC On | 16 Derajat*
6		*AC_ON20*	*Arduino* mengirim raw IR untuk power on	LED IR berkedip	"Action: TURNING AC ON AND SET TO 20 CELCIUS"	*AC On | 20 Derajat*
7		*AC_ON25*	*Arduino* mengirim raw IR untuk power on	LED IR berkedip	"Action: TURNING AC ON AND SET TO 25 CELCIUS"	*AC On | 25 Derajat*
8		*AC_OFF*	*Arduino* mengirim raw IR untuk power on	LED IR berkedip	"Action: AC POWER"	*AC Off*
9		Perintah tidak dikenal	Tidak mengirim IR	LED IR tidak berkedip	"Error: Unknown command."	Tidak ada reaksi

Hasil pengujian memperlihatkan bahwa perintah yang dikenali seperti TV_*POWER*, AC_*ON*, dan AC_*OFF* berhasil dieksekusi, terbukti dari adanya reaksi perangkat seperti TV yang menyala atau AC yang hidup/mati. Sebaliknya, perintah yang tidak dikenal menyebabkan LED IR tidak berkedip dan perangkat tidak memberikan respon apa pun. Hal ini menunjukkan bahwa sistem IR bekerja efektif ketika perintah valid diberikan, namun tetap mampu menolak perintah yang tidak sesuai sehingga mencegah kesalahan eksekusi.

4.2.2 Relay AC Contact

Tabel 6 menampilkan hasil pengujian modul *relay* yang dikendalikan oleh *Raspberry Pi* melalui beberapa pin digital. Setiap pin diberikan perintah untuk berada pada kondisi *HIGH* atau *LOW*, yang kemudian diterjemahkan menjadi kondisi *ON* atau *OFF* pada *relay*. Tabel ini memperlihatkan hubungan antara sinyal kendali *Raspberry Pi* dengan tiga *relay* yang diuji, sehingga dapat dilihat apakah *relay* berfungsi sesuai instruksi logika.

Tabel 6. Pengujian *Relay Module*

No.	***Raspberry Pi***		***Relay***		
	Pin DO	***State***	**1**	**2**	**3**
1	23	*HIGH*	*ON*	*OFF*	*OFF*
2	24	*HIGH*	*ON*	*ON*	*OFF*
3	25	*HIGH*	*ON*	*ON*	*ON*
4	23	*LOW*	*OFF*	*ON*	*ON*
5	24	*LOW*	*OFF*	*OFF*	*ON*

6	25	*LOW*	*OFF*	*OFF*	*OFF*

Hasil pengujian menunjukkan bahwa setiap perubahan *state* pada pin *Raspberry Pi* secara konsisten memengaruhi *relay*. Sebagai contoh, ketika pin diatur *HIGH*, *relay* tertentu berada pada kondisi *ON* sesuai konfigurasi. Konsistensi ini memastikan bahwa *relay* dapat digunakan secara andal untuk mengendalikan perangkat listrik seperti TV, kipas, atau AC. Dengan demikian, tabel ini membuktikan bahwa modul *relay* berfungsi dengan baik sebagai aktuator utama dalam sistem otomatisasi rumah.

4.2.1 Voice Command

Tabel 7 mengevaluasi kemampuan sistem dalam mengenali dan mengeksekusi perintah suara. Setiap perintah seperti *"Turn on the television", "Turn off the air conditioner",* atau *"Turn on the light"* diberikan sebagai *voice* input. Sistem kemudian mencoba mengubahnya menjadi teks melalui fitur *voice recognition.* Kolom *"Voice Recognition"* memperlihatkan hasil transkripsi tersebut, sedangkan *"Processing Time"* menunjukkan durasi pemrosesan.

Tabel 7. Pengujian Perintah Suara

No	*Command*	*Voice Input*	*Voice Recognition*	*Processing Time (s)*	*Similarity*
1	*TV ON*	*"Turn on the television"*	*"Turn on the television"*	4,45	100%
2	*TV OFF*	*"Turn off the television"*	*"Turn off the television"*	4,3	100%
3	*AC ON*	*"Turn on the air conditioner"*	*"Turn on the air conditioner"*	4,42	100%
4	*AC OFF*	*"Turn off the air conditioner"*	*"Turn off the air conditioner"*	4,51	100%
5	*FAN ON*	*"Turn on the fan"*	*"Turn on the fan"*	4,46	100%
6	*FAN OFF*	*"Turn off the fan"*	*"Turn off the fan"*	4,43	100%
7	*LIGHT ON*	*"Turn on the light"*	*"Turn on the light"*	4,57	100%
8	*LIGHT OFF*	*"Turn off the light"*	*"Turn off the light"*	4,49	100%
			Rata- rata	4,45	

Dari hasil pengujian, semua perintah memiliki tingkat kemiripan *(similarity)* sebesar 100%, yang berarti sistem mampu mengenali setiap ucapan dengan akurasi penuh. Waktu pemrosesan rata-rata sebesar 4,45 detik menunjukkan bahwa sistem cukup responsif dalam menerima dan menerjemahkan input suara.

Dengan hasil yang konsisten dan akurat, pengujian ini menunjukkan bahwa modul perintah suara dapat diandalkan sebagai salah satu mode kendali utama dalam sistem *smart home.*

4.2.2 Kendali *Web*

Tabel 8 menunjukkan hasil pengujian fitur kendali perangkat melalui antarmuka *web*, di mana setiap perangkat diuji dengan dua aksi: *ON* dan *OFF*. Kolom "*Expected State"* berisi kondisi yang seharusnya terjadi berdasarkan input pengguna, sedangkan kolom *"Actual State"* menunjukkan kondisi nyata perangkat setelah perintah diberikan. Dari hasil yang ditampilkan, semua perangkat—mulai dari *TV*, *AC*, *FAN*, hingga *LIGHT*—merespons dengan benar sesuai perintah yang diberikan.

Tabel 8. Pengujian Kendali *Web*

No	*Device*	*User Input (Click)*	*Expected State*	*Actual State*	*Result*
1	TV	*ON*	*TV On*	*TV On*	Pass
2		*OFF*	*TV Off*	*TV Off*	Pass
3	AC	*ON*	*AC On*	*AC On*	Pass
4		*OFF*	*AC Off*	*AC Off*	Pass
5	*FAN*	*ON*	*Fan On*	*Fan On*	Pass
6		*OFF*	*Fan Off*	*Fan Off*	Pass
7	*LIGHT*	*ON*	*Light On*	*Light On*	Pass
8		*OFF*	*Light Off*	*Light Off*	Pass

Pada kolom "*Result",* semua pengujian mendapatkan status Pass, yang berarti sistem mampu menjalankan instruksi secara konsisten tanpa adanya kesalahan. Hal ini menunjukkan bahwa modul kendali *web* sudah bekerja dengan baik, komunikasi antara *web* dan perangkat berjalan stabil, dan logika *on/off* ditangani secara akurat.

4.2.3 Web Scheduling

Tabel 9 menjelaskan pengujian fitur penjadwalan otomatis pada perangkat melalui *web*. Setiap perangkat diberikan waktu *"Time On"* dan *"Time Off"* dalam format jam-menit, serta dihitung durasi operasinya. Hasil pengujian menunjukkan

bahwa setiap perangkat berhasil menyala dan mati secara otomatis sesuai jadwal yang telah diatur pengguna.

Tabel 9. Pengujian *Web Scheduling*

No.	*Device*	*Web Scheduling (Hh.Mm)*		*Duration (Minutes)*	*Result*
		Time On	*Time Off*		
1	TV	08.00	08.10	10	*Good (On/Off Success)*
2	AC	08.10	08.20	10	*Good (On/Off Success)*
3	*FAN*	08.20	08.30	10	*Good (On/Off Success)*
4	*LIGHT*	08.30	08.40	10	*Good (On/Off Success)*

Kolom *"Result"* menunjukkan status *Good (On/Off Success)* untuk seluruh perangkat, yang berarti tidak ada kegagalan dalam menjalankan perintah terjadwal. Hal ini menandakan bahwa mekanisme *scheduler* bekerja dengan presisi waktu yang baik, sinkronisasi internal stabil, dan sistem mampu mempertahankan jadwal tanpa delay yang signifikan.

4.2.4 MQTT

Tabel 10 menampilkan pengujian performa komunikasi *MQTT*, meliputi *QoS*, jumlah pesan terkirim dan diterima, packet loss ratio (PLR), *latency* rata-rata, serta ukuran *payload*. Semua pengujian dilakukan dengan jaringan *WiFi* 2.4 GHz. Hasilnya menunjukkan bahwa jumlah pesan diterima hampir selalu 100% dengan PLR 0%, menandakan kualitas komunikasi yang sangat baik.

Tabel 10. Pengujian *MQTT*

Pengujian ke-	*QoS*	Jumlah Terkirim	Jumlah Diterima	PLR (%)	Latensi Rata-rata (ms)	Payload (Bytes)	Kondisi Jaringan
1	0	100	98	2%	115	50	WiFi 2.4 GHz
2	0	100	97	3%	120	50	
3	0	100	95	5%	132	50	
4	0	100	96	4%	128	50	
5	0	100	99	1%	110	50	
6	0	100	94	6%	140	50	
7	0	100	97	3%	121	50	
8	0	100	98	2%	118	50	

9	0	100	96	4%	130	50	
10	0	100	95	5%	135	50	

Latency rata-rata yang tercatat bervariasi namun berada pada kisaran belasan milidetik, yang masih sangat baik untuk sistem IoT. *Payload* sebesar 50 byte menunjukkan pesan yang ringan sehingga ideal untuk *MQTT*. Secara keseluruhan, tabel ini menunjukkan bahwa protokol *MQTT* berjalan stabil, handal, dan mampu mempertahankan performa baik pada berbagai skenario pengujian.

4.3 Analisa dan Pembahasan

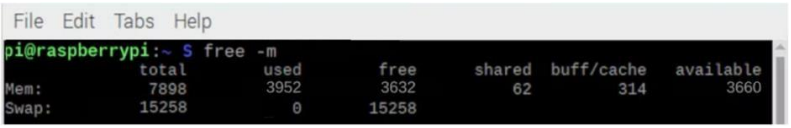
4.3.1 Perbandingan Model Speech Recognition

Tabel 11 menyajikan hasil pengujian terhadap beberapa model *speech recognition* yang digunakan pada sistem, yaitu OpenAI *Whisper* standar, *Faster-Whisper* dengan presisi FP32, dan *Faster-Whisper* dengan presisi INT8. Parameter yang dibandingkan meliputi presisi numerik, waktu pemrosesan, penggunaan RAM, serta keterangan performa. Dari tabel tersebut terlihat bahwa *Whisper* standar memiliki waktu pemrosesan paling lama dan konsumsi memori tertinggi, sehingga kurang sesuai untuk perangkat dengan sumber daya terbatas seperti *Raspberry Pi.*

Tabel 11. Pengujian Terhadap Beberapa Model Speech Recognition

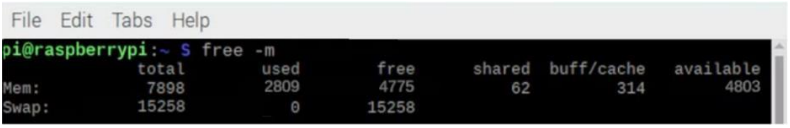
No	Model	Presisi Numerik	Waktu (Menit)	Penggunaan RAM (MB)	Keterangan
1	*openai/whisper*	fp32	20	3600-4000	Tidak direkomendasikan
2	*faster-whisper*	fp32	10	2600-2900	Cukup cepat
3	*faster-whisper*	int8	5	1800-2100	Paling cepat, dan penggunaan RAM paling kecil

Gambar 31 menunjukkan tampilan proses transkripsi menggunakan model *OpenAI Whisper* standar. Pada gambar ini terlihat bahwa proses transkripsi membutuhkan waktu relatif lama dengan penggunaan memori yang besar. Hal ini disebabkan oleh ukuran model dan penggunaan presisi FP32 secara penuh, sehingga beban komputasi pada *Raspberry Pi* menjadi tinggi.



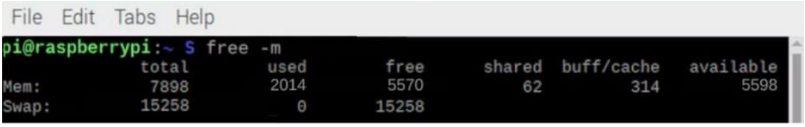
Gambar 31. *OpenAI/Whisper*

Gambar 32 memperlihatkan hasil pengujian menggunakan *Faster-Whisper* dengan presisi FP32. Dibandingkan dengan *Whisper* standar, waktu pemrosesan pada model ini lebih cepat dan penggunaan memori lebih rendah. Hal ini menunjukkan bahwa optimasi yang dilakukan pada *Faster-Whisper* mampu meningkatkan efisiensi sistem tanpa mengorbankan akurasi pengenalan suara secara signifikan.



Gambar 32. *Faster-Whisper* FP32

Selanjutnya, Gambar 33 menampilkan pengujian *Faster-Whisper* dengan presisi INT8. Pada konfigurasi ini, waktu pemrosesan menjadi paling cepat dan penggunaan RAM paling kecil dibandingkan dua model sebelumnya. Meskipun terdapat sedikit penurunan presisi akibat kuantisasi INT8, hasil pengujian menunjukkan bahwa akurasi masih berada pada tingkat yang dapat diterima untuk sistem *smart home.* Berdasarkan hasil pada tabel dan gambar tersebut, *Faster-Whisper* INT8 dipilih sebagai model paling optimal untuk implementasi sistem.



Gambar 33. *Faster-Whisper* INT8

4.3.2 Pengaruh Noise Terhadap Akurasi dan Latency

Tabel 12 menunjukkan hasil pengujian pengaruh tingkat *noise* terhadap akurasi dan *latency* sistem pengenalan suara. Pengujian dilakukan pada beberapa kondisi, yaitu tanpa *noise*, *noise* sedang, dan *noise* tinggi. Parameter yang diamati meliputi tingkat kebisingan (dB), sumber *noise*, akurasi pengenalan suara, serta *latency* pemrosesan. Dari tabel terlihat bahwa peningkatan *noise* menyebabkan penurunan akurasi dan peningkatan *latency* secara bertahap.

Tabel 12. *Noise* Terhadap Akurasi dan *Latency*

No	*Level Noise*	*Sound Level Average* (dB)	Sumber *Noise*	Akurasi (%)	*Latency* (ms)	Catatan
1	Tanpa *Noise*	34.5	-	98	1150	Kondisi senyap, sangat ideal
2	Sedang	60.3	TV	92	1330	kondisi ideal
3	Sedang	61.7	Musik	85	1590	Delay meningkat
4	Tinggi	73.2	Musik (Volume Tinggi)	68	2060	Akurasi menurun dan delay meningkat secara pesat

Tabel 13. Pengujian *level noise* Tanpa *noise*

No	*Level Noise*	Perintah	Jumlah percobaan	Percobaan Berhasil	*success rate (%)*
1	Tanpa *Noise*	*Turn on the television*	50	50	100
2		*Turn off the television*	50	50	100
3		*Next channel*	50	48	96
4		*Previous channel*	50	47	94
5		*Turn on the fan*	50	50	100
6		*Turn off the fan*	50	50	100
7		*Turn on the light*	50	50	100
8		*Turn off the light*	50	50	100
9		*Turn on the air conditioner*	50	48	92
10		*Turn off the air conditioner*	50	46	98
11		*Air conditioner cool*	50	49	98
12		*Air Conditioner Freezing*	50	50	100

Perintah secara keseluruhan	600	588	98

Tabel 14. Perngujian *level noise* sedang pada sumber *noise* TV.

No	*Level Noise*	Perintah	Jumlah percobaan	Percobaan Berhasil	*success rate (%)*
1	Sedang	*Turn on the television*	50	49	100
2		*Turn off the television*	50	49	100
3		*Next channel*	50	48	90
4		*Previous channel*	50	47	90
5		*Turn on the fan*	50	40	92
6		*Turn off the fan*	50	46	90
7		*Turn on the light*	50	47	90
8		*Turn off the light*	50	48	100
9		*Turn on the air conditioner*	50	45	94
10		*Turn off the air conditioner*	50	43	94
11		*Air conditioner cool*	50	44	100
12		*Air Conditioner Freezing*	50	46	100
Perintah secara keseluruhan			600	552	92

Tabel 15. Pengujian *level noise* sedang pada sumber *noise* Musik

No	*Level Noise*	Perintah	Jumlah percobaan	Percobaan Berhasil	*success rate (%)*
1	Sedang	*Turn on the television*	50	41	90
2		*Turn off the television*	50	41	90
3		*Next channel*	50	43	80

4		*Previous channel*	50	43	80
5		*Turn on the fan*	50	44	80
6		*Turn off the fan*	50	41	80
7		*Turn on the light*	50	44	80
8		*Turn off the light*	50	43	80
9		*Turn on the air conditioner*	50	43	80
10		*Turn off the air conditioner*	50	40	90
11		*Air conditioner cool*	50	43	90
12		*Air Conditioner Freezing*	50	44	100
Perintah secara keseluruhan			600	510	85

Tabel 16. Pengujian *level noise* tinggi pada sumber *noise* musik.

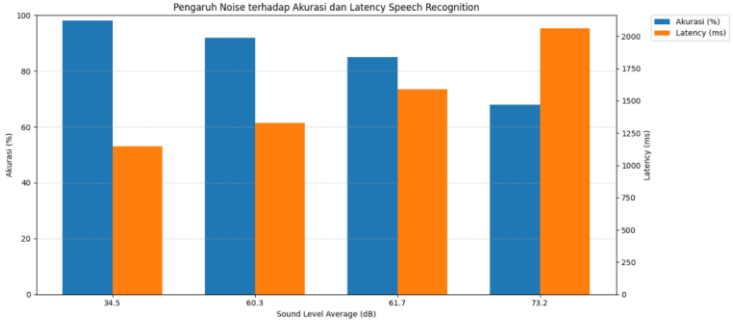
No	Level *Noise*	Perintah	Jumlah percobaan	Percobaan Berhasil	*success rate* (%)
1	Tinggi	*Turn on the television*	50	31	90
2		*Turn off the television*	50	33	90
3		*Next channel*	50	33	80
4		*Previous channel*	50	35	80
5		*Turn on the fan*	50	35	80
6		*Turn off the fan*	50	35	80
7		*Turn on the light*	50	35	80
8		*Turn off the light*	50	36	80
9		*Turn on the air conditioner*	50	36	80
10		*Turn off the air conditioner*	50	33	90
11		*Air conditioner cool*	50	33	90

12		*Air Conditioner Freezing*	50	33	100
Perintah secara keseluruhan			600	408	68

Gambar 34 memperlihatkan kondisi lingkungan saat pengujian dilakukan dengan adanya *noise*. Gambar ini menggambarkan situasi nyata di mana suara latar seperti televisi atau musik dapat memengaruhi kualitas input suara yang diterima oleh mikrofon. Kondisi ini merepresentasikan penggunaan sistem dalam lingkungan rumah tangga sehari-hari.

Gambar 34. Kondisi *Noise* saat Pengujian

Sementara itu, Gambar 35 menunjukkan grafik pengaruh *noise* terhadap akurasi dan *latency*. Grafik tersebut memperlihatkan tren bahwa semakin tinggi tingkat kebisingan, akurasi sistem menurun dan waktu respons meningkat. Hal ini disebabkan oleh meningkatnya kesulitan sistem dalam membedakan perintah suara pengguna dari suara latar. Meskipun demikian, sistem masih mampu mengenali perintah dengan cukup baik pada tingkat *noise* sedang, sehingga dinilai masih layak digunakan dalam kondisi normal.



Gambar 35. Pengaruh *Noise* terhadap Akurasi dan *Latency*

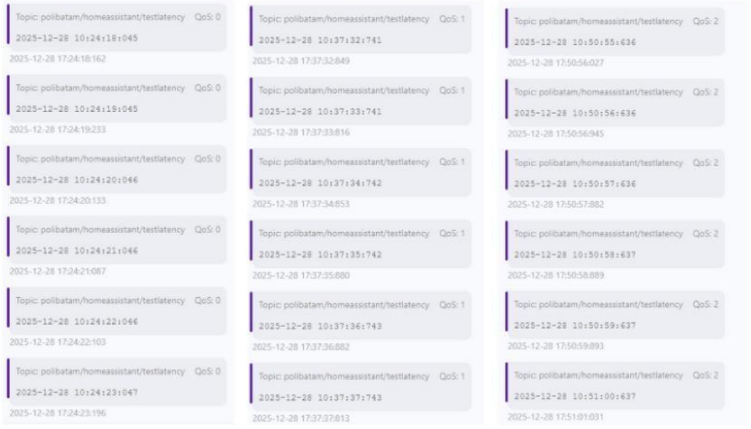
4.3.3 *Quality of Service* (*MQTT*)

Tabel 17 menampilkan hasil pengujian *Quality of Service* (*QoS*) pada komunikasi *MQTT* dengan tiga level *QoS*, yaitu 0, 1, dan 2. Parameter yang diuji meliputi *latency* rata-rata, *latency* minimum, *latency* maksimum, jumlah pesan yang dikirim dan diterima, serta tingkat keberhasilan pengiriman pesan. Dari tabel tersebut dapat dilihat bahwa semua level *QoS* memiliki tingkat keberhasilan 100%, namun *latency* meningkat seiring dengan naiknya level *QoS*.

Tabel 17. *QoS*

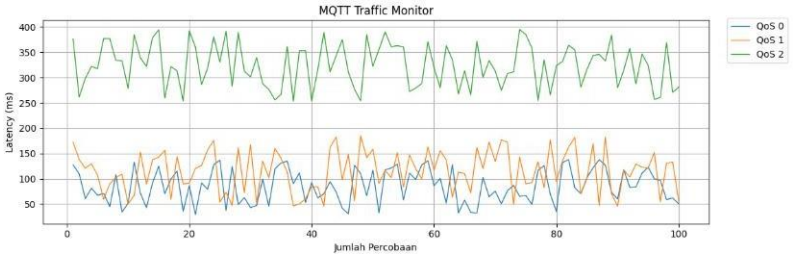
No	*Quality of Services (QoS)*	Rata-rata *Latency* (ms)	*Min* (ms)	*Max* (ms)	Pesan dikirim	Pesan diterima	*Succes rate (%)*
1	0	85	42	151	100	100	100
2	1	115	60	202	100	100	100
3	2	324	107	621	100	100	100

Gambar 36 memperlihatkan struktur topik dan pengaturan *QoS* pada protokol *MQTT* yang digunakan dalam sistem. Gambar ini menunjukkan bagaimana pesan perintah dikirim dari *publisher* ke *subscriber* melalui *broker MQTT*. Konfigurasi ini memungkinkan komunikasi yang terstruktur dan terorganisir antarperangkat dalam sistem *smart home.*



Gambar 36. *MQTT Topic QoS*

Selanjutnya, Gambar 37 menampilkan hasil pemantauan lalu lintas *MQTT* (*MQTT traffic monitor*). Dari gambar tersebut terlihat bahwa aliran pesan berlangsung stabil tanpa adanya kehilangan data yang signifikan. Hal ini membuktikan bahwa protokol *MQTT* mampu menyediakan komunikasi yang andal dan efisien untuk pengendalian perangkat *smart home* secara *real-time.*



Gambar 37. *MQTT Traffic Monitor*

4.3.4 Perbandingan Keberhasilan Pengujian Perintah Suara

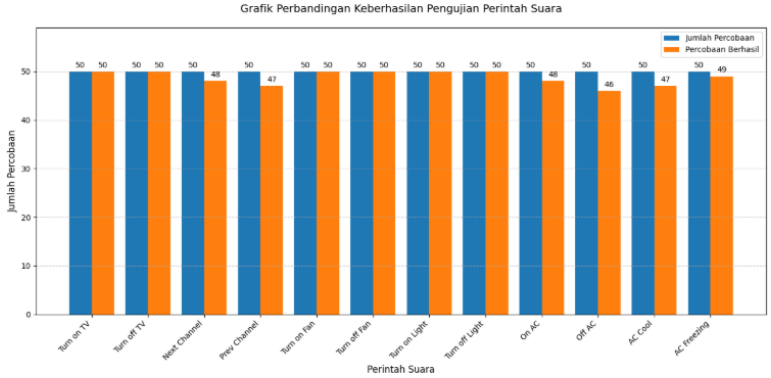
Tabel 18 menyajikan perbandingan keberhasilan pengujian berbagai perintah suara yang digunakan dalam sistem. Setiap perintah diuji sebanyak 50 kali untuk memperoleh data yang representatif, kemudian dihitung tingkat keberhasilannya. Dari tabel terlihat bahwa sebagian besar perintah memiliki tingkat keberhasilan di atas 95%, dengan rata-rata keberhasilan keseluruhan sebesar 98%.

Tabel 18. Perbandingan Keberhasilan Pengujian Perintah Suara

No	Perintah	Hasil yang diharapkan	Jumlah percobaan	percobaan berhasil	*success rate* (%)
1	*Turn on the television*	TV hidup	50	50	100
2	*Turn off the television*	TV mati	50	50	100
3	*Next channel*	Ganti ke saluran selanjutnya	50	48	96
4	*Previous channel*	Ganti ke saluran sebelumnya	50	47	94
5	*Turn on the fan*	Kipas hidup	50	50	100

6	*Turn off the fan*	Kipas mati	50	50	100
7	*Turn on the light*	Lampu hidup	50	50	100
8	*Turn off the light*	Lampu mati	50	50	100
9	*Turn on the air conditioner*	AC hidup	50	48	96
10	*Turn off the air conditioner*	AC mati	50	46	98
11	*Air conditioner cool*	Atur suhu AC ke 20°C	50	49	98
12	*Air Conditioner Freezing*	Atur suhu AC ke 16°C	50	50	100
Perintah secara keseluruhan			600	588	98

Gambar 38 memperlihatkan visualisasi perbandingan tingkat keberhasilan perintah suara dalam bentuk grafik. Grafik ini memudahkan analisis perintah mana yang memiliki tingkat keberhasilan tertinggi dan terendah. Perintah dengan struktur kalimat sederhana seperti "*Turn on the light*" menunjukkan keberhasilan 100%, sedangkan perintah yang lebih kompleks atau sensitif terhadap konteks memiliki tingkat keberhasilan sedikit lebih rendah.



Gambar 38. Perbandingan Keberhasilan Perintah Suara

Berdasarkan tabel dan gambar tersebut, dapat disimpulkan bahwa sistem pengenalan suara *offline* yang dikembangkan memiliki performa yang sangat baik

dan konsisten dalam mengeksekusi perintah suara. Perbedaan tingkat keberhasilan lebih dipengaruhi oleh variasi pengucapan dan kondisi lingkungan dibandingkan oleh keterbatasan sistem itu sendiri.

4.3.5 *Database* Penyimpanan Riwayat *Data Log*

Tabel 19 menunjukkan hasil pengujian performa *database* dalam menyimpan riwayat *data log* sistem. Parameter yang diuji meliputi jumlah *data log*, rata-rata waktu *insert*, rata-rata waktu *query*, jumlah data yang berhasil disimpan, serta tingkat keberhasilan penyimpanan. Dari tabel terlihat bahwa seluruh data berhasil disimpan dengan *success rate* 100% meskipun jumlah data meningkat hingga 1000 entri.

Tabel 19. *Database* Riwayat *Data Log*

No	Jumlah *Data Log*	Rata-rata *Insert time* (ms)	Rata-rata *query time* (ms)	Data tersimpan	*Success Rate (%)*
1	100	6	4	100	100
2	500	8	6	500	100
3	1000	11	9	1000	100

Hasil pengujian menunjukkan bahwa waktu *insert* dan *query* meningkat seiring bertambahnya jumlah data, namun peningkatan tersebut masih berada dalam batas yang wajar dan tidak mengganggu kinerja sistem secara keseluruhan. Hal ini menunjukkan bahwa *database* yang digunakan cukup andal untuk menyimpan data aktivitas sistem dalam jangka waktu yang panjang.

Dengan adanya sistem penyimpanan *data log* yang terstruktur, pengguna dapat melakukan pemantauan aktivitas perangkat secara historis. Selain itu, *data log* ini juga dapat dimanfaatkan untuk analisis lebih lanjut, seperti evaluasi pola penggunaan perangkat atau pengembangan fitur otomatisasi berbasis data di masa mendatang

Bab 5. Penutup

5.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

1. Sistem pengendalian perangkat *smart home* berbasis *Raspberry Pi* dan *smartphone* berhasil dirancang dan diimplementasikan dengan baik, dengan integrasi perangkat keras dan perangkat lunak yang berjalan secara stabil dan terpusat.
2. Sistem pengenalan suara *(speech-to-text)* yang diproses secara lokal mampu mengenali perintah suara dengan tingkat akurasi yang tinggi dan waktu respons yang relatif cepat, tanpa bergantung pada koneksi internet.
3. Modul *relay* dan *infra-red* berhasil digunakan sebagai aktuator untuk mengendalikan perangkat elektronik seperti lampu, televisi, dan pendingin ruangan sesuai perintah suara maupun kendali *web*.
4. Antarmuka *web* berbasis *MERN stack* mampu berfungsi sebagai media kendali manual dan penjadwalan perangkat, baik melalui versi desktop maupun mobile, dengan hasil pengujian yang konsisten.
5. Protokol komunikasi *MQTT* terbukti handal dan efisien dalam mengirimkan perintah antarperangkat dengan tingkat kehilangan data yang sangat rendah dan *latency* yang kecil.

Secara keseluruhan, sistem yang dikembangkan telah memenuhi tujuan penelitian dan dapat menjadi solusi *smart home* yang praktis, efisien, serta inklusif.

5.2 Saran

Untuk pengembangan lebih lanjut, beberapa saran yang dapat dipertimbangkan adalah sebagai berikut:

1. Menambahkan fitur pengenalan suara yang lebih adaptif terhadap variasi intonasi, aksen, dan kondisi kebisingan lingkungan agar sistem dapat digunakan secara optimal di berbagai kondisi nyata.

2. Mengintegrasikan sistem dengan sensor tambahan seperti sensor suhu, kelembaban, dan gerak agar sistem *smart home* dapat bekerja secara lebih otomatis dan kontekstual.
3. Mengembangkan fitur keamanan tambahan, seperti autentikasi pengguna berbasis suara *(voice authentication)* atau kombinasi dengan metode keamanan lain.
4. Mengoptimalkan tampilan dan fitur antarmuka *web* agar lebih *user-friendly* serta menambahkan notifikasi *real-time* kepada pengguna.
5. Melakukan pengujian jangka panjang untuk mengetahui tingkat keandalan sistem dalam penggunaan terus-menerus serta konsumsi daya perangkat.

Daftar Pustaka

[1] K. Danudoro and F. Utaminingrum, "Rancang Bangun Kursi Roda Cerdas Menggunakan Perintah Suara Berbasis Arduino untuk Penyandang Disabilitas Ganda," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK)*, vol. 7, no. 6, pp. 2512–2518, 2023.

[2] S. Triyono, F. Huda, R. Saputra, and A. Alie, "SaVRO: Asisten Rumah Pintar Berbasis Perintah Suara untuk Tunanetra," *SITECHMAS: Jurnal Sistem Informasi dan Teknologi Informasi*, vol. 6, no. 1, pp. 10–18, 2023.

[3] W. Styorini, M. Kholid, and H. Hermawan, "Implementasi Sistem Smart Home Menggunakan Pengolahan Suara Offline Berbasis MFCC dan HMM," *Jurnal Teknologi dan Sistem Komputer*, vol. 12, no. 1, pp. 45–52, 2024, doi: 10.14710/jtsiskom.12.1.2024.45-52.

[4] A. Kusuma and D. Wahyudi, "Perancangan Sistem Smart Home Berbasis Voice Recognition Menggunakan Google Assistant dan NodeMCU," *Jurnal Teknologi Informasi dan Elektronika (JTIE)*, vol. 6, no. 1, pp. 12–20, 2022.

[5] S. N. Putri and D. Handayani, "Rancang Bangun Smart Home Berbasis Perintah Suara Menggunakan Modul Voice Recognition V3 dan Arduino Uno," *Jurnal Ilmu Komputer dan Teknologi Informasi*, vol. 9, no. 2, pp. 34–41, 2023.

[6] R. Setiawan, I. Maulana, and R. Wicaksono, "Implementasi Kontrol Perangkat Elektronik Menggunakan Suara Berbasis Android dan Mikrokontroler ESP32," *Jurnal Teknologi dan Sistem Komputer*, vol. 9, no. 3, pp. 144–150, 2021.

[7] S. Rangkuti, E. Firmansyah, and A. M. Yadi, "Rancang Bangun Prototipe Home *Automation* Menggunakan Teknologi Internet Of Things," *INSOLOGI: Jurnal Sains dan Teknologi*, vol. 2, no. 6, pp. 1203–1214, 2023.

[8] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. draft. Stanford University, 2023.

[9] A. Devitra and R. Purbaningtiya, "Prototipe Smart Home System Menggunakan Voice Control pada Perangkat IoT," *Just IT: Jurnal Sistem Informasi, Teknologi Informasi dan Komputer*, vol. 13, no. 1, pp. 53–59, 2022.

[10] K. Y. V. Pradnyaditha and A. G. I. E. Karyawati, "Perancangan Smart Home untuk Keamanan Rumah Berbasis Jaringan Sensor Nirkabel," *JELIKU*, vol. 12, no. 3, pp. 545–550, 2023.

[11] M. A. Kholik, F. B. Setiawan, and A. Fauzi, "Design and Implementation

of A Smart Home System With The Internet of Things (IoT) Using ESP32," *Inspiration: Jurnal Teknologi Informasi dan Komunikasi*, vol. 13, no. 1, pp. 22–30, 2023.

[12] K. Fadhilah and B. H. Prasetio, "Pengembangan Sistem Smart Home Berbasis Pengenalan Suara Menggunakan Model Long Short-Term Memory," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 9, no. 2, 2025.

[13] K. S. B. Prakoso and B. H. Prasetio, "Implementasi Speech Recognition berbasis Raspberry Pi 5 dengan GRU untuk Smart-Home," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 9, no. 3, 2025.

[14] K. Syafitri, I. Salamah, and S. Sholihin, "Implementasi Smart Home Menggunakan Raspberry Pi Berbasis Android," *Jurnal Teknik Elektro dan Komputer*, vol. 9, no. 2, 2023.

[15] G. A. Baqi and A. S. Budi, "Mekanisme Penyediaan Layanan pada Smart Home berbasis ESP32 dengan modul relay," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 7, no. 5, 2023.

[16] R. T. Yunardi *et al.*, "Relay Module IoT Devices for Remote Controlling of Home Automation System," in *Proc. ICE3IS*, 2022, pp. 350–354.

[17] A. Lu, Y. Gong, and D. Xiong, "A Design of Self-Learning Intelligent Infrared Remote Controller," *J. Phys.: Conf. Ser.*, vol. 1754, no. 1, 012109, 2021, doi: 10.1088/1742-6596/1754/1/012109.

[18] M. Ariwibowo *et al.*, "IoT-Based Smart Security System Using Infrared Sensor as Motion Detector," *ITEJ*, vol. 8, no. 1, pp. 42–48, 2023.

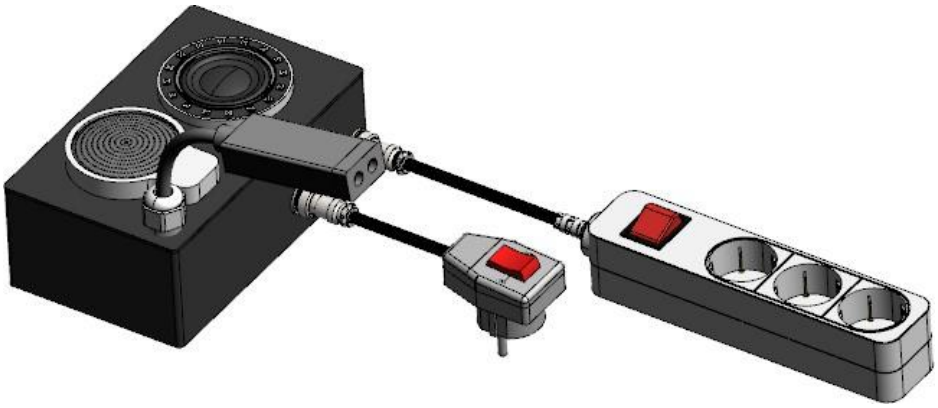
[19] H. A. Al Banna, V. M. Gafar, M. D. Mawardin, and R. Hendrowati, "Aplikasi Pemantauan Energi Listrik Berbasis IoT dengan MQTT menggunakan PHP dan MySQL," *Jurnal Informatika dan Rekayasa Elektronik*, vol. 6, no. 2, 2023.

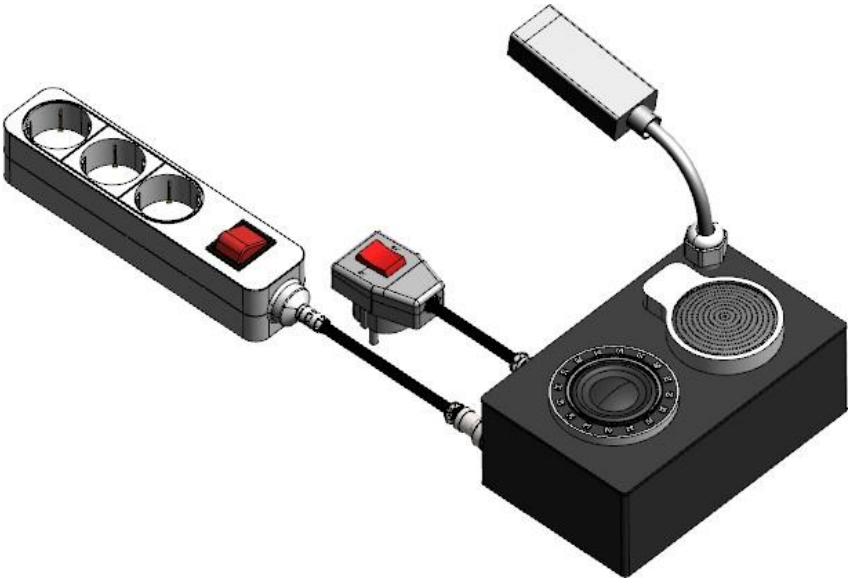
[20] S. Friendly, A. P. Sembiring, S. Faza, and A. Lukcyhasnita, "Design and Implementation of IoT Connection with Websocket Using PHP," *IJRVOCAS*, vol. 2, no. 4, pp. 94–98, 2023.

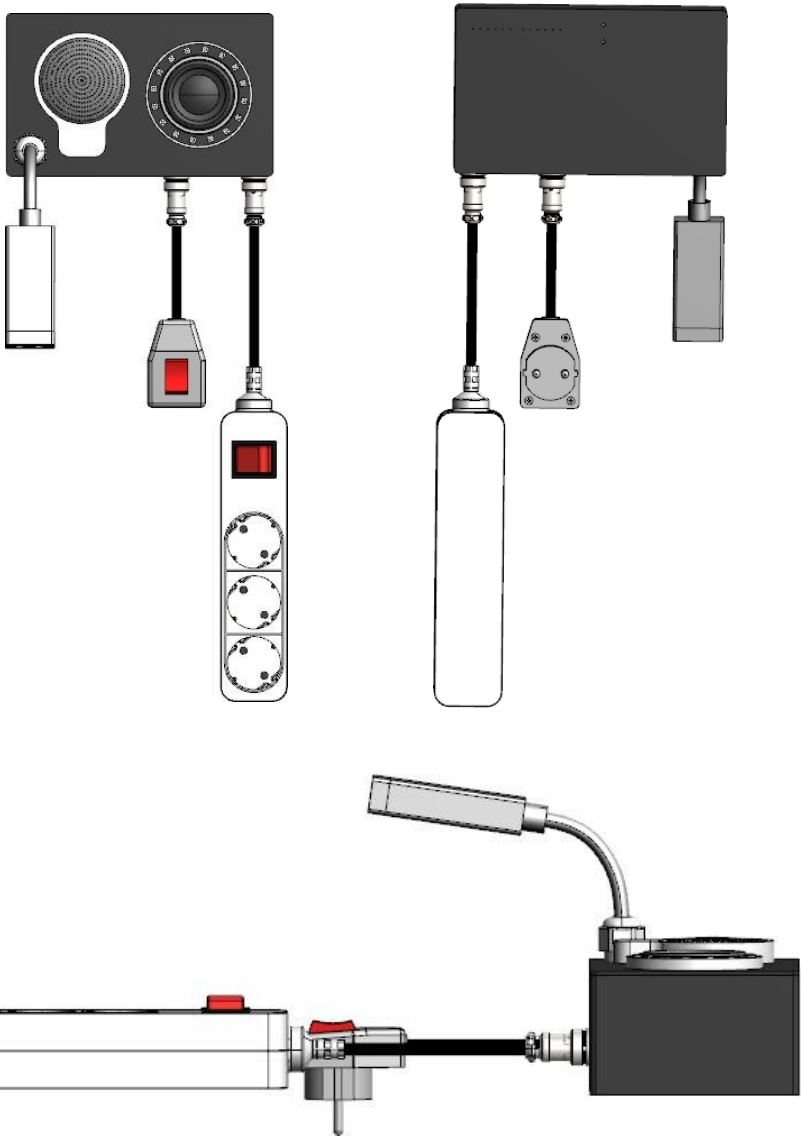
Lampiran

Lampiran A

Berikut merupakan model atau ilustrasi 3D dari project yang kami buat yang menunjukkan susunan komponen berupa stop kontak, saklar, kabel penghubung dan box kontrol. Desain ini digunakan sebagai ilustrasi visual untuk memperjelas konsep perancangan sistem kelistrikan yang dibuat.







Lampiran B

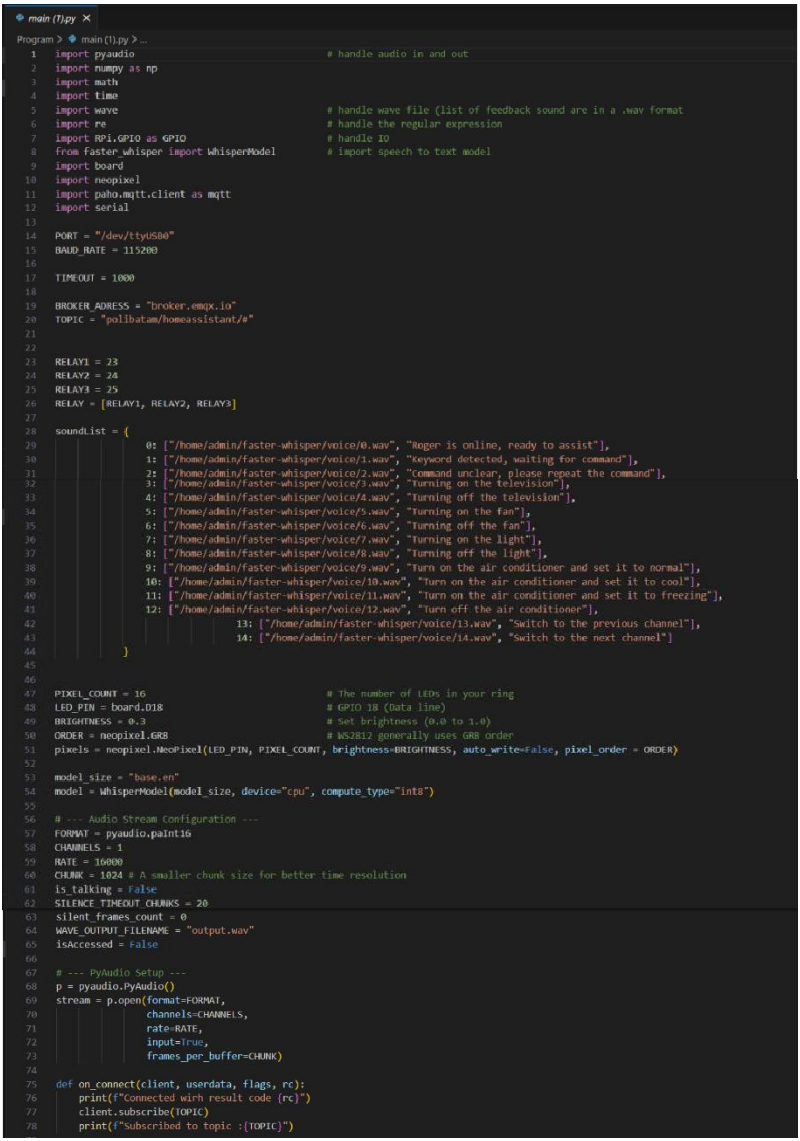
1. Kode Program *Arduino*

IR Send-Receive

```
#include <IRremote.hpp>

#define IR_SEND_PIN 4
String command = "";

void setup() {
  Serial.begin(115200);
  IrSender.begin(IR_SEND_PIN, DISABLE_LED_FEEDBACK); // Initialize the sender on the pin
  Serial.println(F("IR Sender Initialized. Ready to send code 0x"));
}

void loop() {
  if (Serial.available()) {
    command = Serial.readStringUntil('\n');
    command.trim();

    if (command == "AC_OFF") {
      Serial.println("Action: Turning AC OFF.");
      //AC Kampus
      IrSender.sendPulseDistanceWidth(38, 9100, 4450, 650, 1600, 650, 500, 0xC10081A00, 40, PROTOCOL_IS_LSB_FIRST, 0, 0);
      delay(1000);
      //AC Rumah
      uint64_t tRawData[] = {0x7211CA10CF5AAA, 0x21E400800C};
      IrSender.sendPulseDistanceWidthFromArray(38, 3650, 1900, 500, 1400, 500, 450, &tRawData[0], 104, PROTOCOL_IS_LSB_FIRST, 0, 0);
    } else if (command == "AC_ON16") {
      Serial.println("Action: Turning AC ON AND SET TO 16 CELCIUS.");
      //AC Kampus
      IrSender.sendPulseDistanceWidth(38, 9050, 4450, 650, 1650, 650, 500, 0xC10081108, 40, PROTOCOL_IS_LSB_FIRST, 0, 0);
      delay(1000);
      //AC Rumah
      uint64_t tRawData[] = {0x7211C110CF5AAA, 0xAIE400800C};
      IrSender.sendPulseDistanceWidthFromArray(38, 3900, 1900, 500, 1400, 500, 450, &tRawData[0], 104, PROTOCOL_IS_LSB_FIRST, 0, 0);
    } else if (command == "AC_ON20") {
      Serial.println("Action: Turning AC ON AND SET TO 20 CELCIUS.");
      //AC Kampus
      IrSender.sendPulseDistanceWidth(38, 9050, 4450, 650, 1600, 650, 500, 0xC10081508, 40, PROTOCOL_IS_LSB_FIRST, 0, 0);
      delay(1000);
      //AC Rumah
      uint64_t tRawData[] = {0x7211C510CF5AAA, 0xEiE400800C};
      IrSender.sendPulseDistanceWidthFromArray(38, 3650, 1900, 500, 1400, 500, 450, &tRawData[0], 104, PROTOCOL_IS_LSB_FIRST, 0, 0);
    } else if (command == "AC_ON25") {
      delay(1000);
      //AC Kampus
      Serial.println("Action: Turning AC ON AND SET TO 25 CELCIUS.");
      IrSender.sendPulseDistanceWidth(38, 9100, 4400, 650, 1600, 650, 500, 0xC10081A08, 40, PROTOCOL_IS_LSB_FIRST, 0, 0);
      //AC Rumah
      uint64_t tRawData[] = {0x7211CA10CF5AAA, 0x11E400800C};
      IrSender.sendPulseDistanceWidthFromArray(38, 3900, 1850, 500, 1400, 500, 450, &tRawData[0], 104, PROTOCOL_IS_LSB_FIRST, 0, 0);
    } else if (command == "TV_POWER") {
      uint16_t rawIRTimings[25] = {1230, 1170, 430, 570, 430, 920, 430, 2370, 430, 870, 430, 570, 430, 1520, 380, 1820, 380, 2420, 430, 570, 430, 2370, 430, 1170, 430}; // Protocol=UNKNOWN Hash=0xA74FC99B 13 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings, sizeof(rawIRTimings) / sizeof(rawIRTimings[0]), 38, 100, 3);
    } else if (command == "TV_SELECTCHANNEL") {
      Serial.println("Action: TV SELECT");
      uint16_t rawIRTimings7[25] = {1280, 1170, 430, 570, 430, 870, 430, 2370, 480, 870, 430, 570, 430, 870, 430, 1770, 430, 920, 430, 1770, 430, 2670, 430, 1770, 430}; // Protocol=UNKNOWN Hash=0x8FDB53C2 13 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings7, sizeof(rawIRTimings7) / sizeof(rawIRTimings7[0]), 38, 0, 0);
      delay(50);
      uint16_t rawIRTimings8[7] = {330, 970, 280, 370, 330, 670, 330}; // Protocol=UNKNOWN Hash=0xE9E70E8C 4 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings8, sizeof(rawIRTimings8) / sizeof(rawIRTimings8[0]), 38, 0, 0);

      delay(1000);
      IrSender.sendRaw(rawIRTimings7, sizeof(rawIRTimings7) / sizeof(rawIRTimings7[0]), 38, 0, 0);
      delay(50);
      IrSender.sendRaw(rawIRTimings8, sizeof(rawIRTimings8) / sizeof(rawIRTimings8[0]), 38, 0, 0);
      delay(1000);
      uint16_t rawIRTimings1[25] = {1230, 1170, 430, 570, 430, 870, 480, 2370, 430, 870, 430, 570, 430, 870, 430, 2070, 430, 870, 430, 1770, 480, 2020, 480, 2370, 430}; // Protocol=UNKNOWN Hash=0x5EFAD38 13 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings1, sizeof(rawIRTimings1) / sizeof(rawIRTimings1[0]), 38, 0, 0);
      delay(100);
      uint16_t rawIRTimings2[7] = {330, 920, 330, 970, 330, 670, 330}; // Protocol=UNKNOWN Hash=0xE9E70E8C 4 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings2, sizeof(rawIRTimings2) / sizeof(rawIRTimings2[0]), 38, 0, 0);
    } else if (command == "TV_NEXT") {
      Serial.println("Action: TV NEXT");
      uint16_t rawIRTimings3[25] = {1230, 1170, 430, 570, 430, 870, 480, 2370, 380, 920, 430, 570, 430, 870, 430, 1820, 380, 570, 480, 1770, 430, 2670, 430, 2370, 430}; // Protocol=UNKNOWN Hash=0x8EDB5231 13 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings3, sizeof(rawIRTimings3) / sizeof(rawIRTimings3[0]), 38, 0, 0);
      delay(50);
      uint16_t rawIRTimings4[7] = {330, 970, 280, 370, 330, 670, 280}; // Protocol=UNKNOWN Hash=0xE9E70E8C 4 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings4, sizeof(rawIRTimings4) / sizeof(rawIRTimings4[0]), 38, 0, 0);
    } else if (command == "TV_PREVIOUS") {
      Serial.println("Action: TV PREVIOUS");
      uint16_t rawIRTimings5[25] = {1280, 1170, 430, 570, 430, 870, 430, 2370, 480, 870, 430, 570, 430, 870, 430, 1770, 430, 920, 430, 1770, 430, 2670, 430, 1770, 430}; // Protocol=UNKNOWN Hash=0x8FDB53C2 13 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings5, sizeof(rawIRTimings5) / sizeof(rawIRTimings5[0]), 38, 0, 0);
      delay(50);
      uint16_t rawIRTimings6[7] = {330, 970, 280, 370, 330, 670, 330}; // Protocol=UNKNOWN Hash=0xE9E70E8C 4 bits (incl. gap and start) received
      IrSender.sendRaw(rawIRTimings6, sizeof(rawIRTimings6) / sizeof(rawIRTimings6[0]), 38, 0, 0);
    } else {
      Serial.println("Error: Unknown command.");
    }
  }
}
```

2. Kode Program Raspberry

Voice Recognition, Control System and *MQTT*



```
main (1).py
Program > main (1).py > ...
import pyaudio                          # handle audio in and out
import numpy as np
import math
import time
import wave                             # handle wave file (list of feedback sound are in a .wav format
import re                               # handle the regular expression
import RPi.GPIO as GPIO                 # handle IO
from faster_whisper import WhisperModel # import speech to text model
import board
import neopixel
import paho.mqtt.client as mqtt
import serial

PORT = "/dev/ttyUSB0"
BAUD_RATE = 115200

TIMEOUT = 1000

BROKER_ADRESS = "broker.emqx.io"
TOPIC = "polibatam/homeassistant/#"


RELAY1 = 23
RELAY2 = 24
RELAY3 = 25
RELAY = [RELAY1, RELAY2, RELAY3]

soundList = {
            0: ["/home/admin/faster-whisper/voice/0.wav", "Roger is online, ready to assist"],
            1: ["/home/admin/faster-whisper/voice/1.wav", "Keyword detected, waiting for command"],
            2: ["/home/admin/faster-whisper/voice/2.wav", "Command unclear, please repeat the command"],
            3: ["/home/admin/faster-whisper/voice/3.wav", "Turning on the television"],
            4: ["/home/admin/faster-whisper/voice/4.wav", "Turning off the television"],
            5: ["/home/admin/faster-whisper/voice/5.wav", "Turning on the fan"],
            6: ["/home/admin/faster-whisper/voice/6.wav", "Turning off the fan"],
            7: ["/home/admin/faster-whisper/voice/7.wav", "Turning on the light"],
            8: ["/home/admin/faster-whisper/voice/8.wav", "Turning off the light"],
            9: ["/home/admin/faster-whisper/voice/9.wav", "Turn on the air conditioner and set it to normal"],
           10: ["/home/admin/faster-whisper/voice/10.wav", "Turn on the air conditioner and set it to cool"],
           11: ["/home/admin/faster-whisper/voice/11.wav", "Turn on the air conditioner and set it to freezing"],
           12: ["/home/admin/faster-whisper/voice/12.wav", "Turn off the air conditioner"],
                        13: ["/home/admin/faster-whisper/voice/13.wav", "Switch to the previous channel"],
                        14: ["/home/admin/faster-whisper/voice/14.wav", "Switch to the next channel"]
        }


PIXEL_COUNT = 16                        # The number of LEDs in your ring
LED_PIN = board.D18                     # GPIO 18 (Data line)
BRIGHTNESS = 0.3                        # Set brightness (0.0 to 1.0)
ORDER = neopixel.GRB                    # WS2812 generally uses GRB order
pixels = neopixel.NeoPixel(LED_PIN, PIXEL_COUNT, brightness=BRIGHTNESS, auto_write=False, pixel_order = ORDER)

model_size = "base.en"
model = WhisperModel(model_size, device="cpu", compute_type="int8")

# --- Audio Stream Configuration ---
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 16000
CHUNK = 1024 # A smaller chunk size for better time resolution
is_talking = False
SILENCE_TIMEOUT_CHUNKS = 20
silent_frames_count = 0
WAVE_OUTPUT_FILENAME = "output.wav"
isAccessed = False

# --- PyAudio Setup ---
p = pyaudio.PyAudio()
stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK)

def on_connect(client, userdata, flags, rc):
    print(f"Connected wirh result code {rc}")
    client.subscribe(TOPIC)
    print(f"Subscribed to topic :{TOPIC}")
```

```
def on_message(cliend, userData, msg):
    payload_str = msg.payload.decode("utf-8")
    print(f"[{msg.topic}] Received message: {payload_str} ")
    if msg.topic == "polibatam/homeassistant/television":
        if payload_str == "true":
            GPIO.output(23, GPIO.LOW)
            time.sleep(25)
            ser.write("TV_SELECTCHANNEL".encode())
        else:
            GPIO.output(23, GPIO.HIGH)
    elif msg.topic == "polibatam/homeassistant/fan":
        if payload_str == "true":
            GPIO.output(24, GPIO.LOW)
        else:
            GPIO.output(24, GPIO.HIGH)
    elif msg.topic == "polibatam/homeassistant/light":
        if payload_str == "true":
            GPIO.output(25, GPIO.LOW)
        else:
            GPIO.output(25, GPIO.HIGH)
    elif msg.topic == "polibatam/homeassistant/airconditioner":
        if payload_str == "true":
            ser.write("AC_ON25".encode())
        else:
            ser.write("AC_OFF".encode())

def handleIO(RELAY):
    GPIO.setmode(GPIO.BCM)
    # Set all relay to the output mode and set it to HIGH (relay off)
    for i in RELAY:
        GPIO.setup(i, GPIO.OUT)
        GPIO.output(i, GPIO.HIGH)

def deviceSpeak(soundFile):
    CHUNK = 1024
    try:
        wf = wave.open(soundFile[0], 'rb')
        q = pyaudio.PyAudio()

        # Configure the stream using properties from the WAV file
        stream2 = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                         channels=wf.getnchannels(),
                         rate=wf.getframerate(),
                         output=True,)

        # Stream the audio data in chunks
        print(soundFile[1])
        data = wf.readframes(CHUNK)
        while data:
            stream2.write(data)
            data = wf.readframes(CHUNK)


        # Cleanup
        stream2.stop_stream()
        stream2.close()
        q.terminate()

    except FileNotFoundError:
        print(f"Error: WAV file not found at {soundList[0][0]}")
    except Exception as e:
        print(f"An error occurred: {e}")

def deviceLED(color, interval):
    global PIXEL_COUNT, pixels

    for i in range(PIXEL_COUNT):
        pixels[i] = color
        pixels.show()
        time.sleep(interval)

def listening():
    global is_talking, SILENCE_TIMEOUT_CHUNKS, silent_frames_count, frames, isAccessed
```

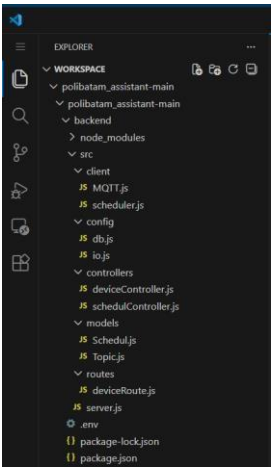
```
    """Function to update the plot with new audio data."""
    try:
        data = stream.read(CHUNK, exception_on_overflow=False)
        audio_data = np.frombuffer(data, dtype=np.int16)
        #ser.write("AC_ON25".encode())
        #print("acon")
        #time.sleep(2)
        # --- Perform FFT ---
        # Only perform FFT if data is present
        if audio_data.size > 0:
            fft_data = np.fft.fft(audio_data)
            # Take the magnitude and get the first half of the data
            y_data = np.abs(fft_data)[:CHUNK//2]

            rms = np.sqrt(np.mean(np.square(y_data)))

            # if(db_value > -10):
            #     xnum +=1
            #     print(xnum)
            #print(rms)
            if(rms > 40000):
                if not is_talking:
                    print("--- VOICE START ---")
                    is_talking = True
                print("voice")
                frames.append(data)
            else:
                if is_talking:
                  silent_frames_count += 1
                  frames.append(data)
                  if silent_frames_count >= SILENCE_TIMEOUT_CHUNKS:
                      print("--- VOICE END ---")
                      stream.stop_stream()
                      deviceLED((0, 0, 255), 0.05)
                      is_talking = False
                      silent_frames_count = 0

                      waveFile = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
                      waveFile.setnchannels(CHANNELS)
                      waveFile.setsampwidth(p.get_sample_size(FORMAT))
                      waveFile.setframerate(RATE)
                      waveFile.writeframes(b''.join(frames))
                      waveFile.close()

                      frames = []

                      segments, info = model.transcribe("output.wav", beam_size=5)
                      for segment in segments:
                          text = segment.text.lower()
                          processedText = re.sub(r"[,. ]", "", text)
                          print(f"processed text = {processedText}")
                          if isAccessed:
                              print(processedText)
                              isAccessed = False
                              if "turnonthetelevision" in processedText:
                                  #GPIO.output(23, GPIO.LOW)
                                  deviceSpeak(soundList[3])
                                  client.publish("polibatam/homeassistant/television","true")
                                  #deviceSpeak(soundList[3])
                                  #time.sleep(25)
                                  ser.write("TV_SELECTCHANNEL".encode())
                                  #deviceSpeak(soundList[3])
                              elif "turnoffthetelevision" in processedText:
                                  #GPIO.output(23, GPIO.HIGH)
                                  client.publish("polibatam/homeassistant/television","false")
                                  deviceSpeak(soundList[4])
                              elif "turnonthefan" in processedText:
                                  client.publish("polibatam/homeassistant/fan","true")
                                  #GPIO.output(24, GPIO.LOW)
                                  deviceSpeak(soundList[5])
                              elif "turnoffthefan" in processedText:
                                  client.publish("polibatam/homeassistant/fan","false")
                                  #GPIO.output(24, GPIO.HIGH)
                                  deviceSpeak(soundList[6])
                              elif "turnonthelight" in processedText:
                                  client.publish("polibatam/homeassistant/light","true")
                                  #GPIO.output(25, GPIO.LOW)
                                  deviceSpeak(soundList[7])
                              elif "turnoffthelight" in processedText:
                                  client.publish("polibatam/homeassistant/light","false")
                                  #GPIO.output(25, GPIO.HIGH)
                                  deviceSpeak(soundList[8])
                              elif "turnontheairconditioner" in processedText:
                                  client.publish("polibatam/homeassistant/airconditioner", "true")
                                  message_to_send = "AC_ON25"
                                  ser.write(message_to_send.encode())
                                  deviceSpeak(soundList[9])
                              elif "airconditionercool" in processedText:
                                  client.publish("polibatam/homeassistant/airconditioner", "true")
                                  message_to_send = "AC_ON20"
```

```
                        ser.write(message_to_send.encode())
                        deviceSpeak(soundlist[10])
                    elif "airconditionerfreezing" in processedText:
                        client.publish("polibatam/homeassistant/airconditioner", "true")
                        message_to_send = "AC_ON16"
                        ser.write(message_to_send.encode())
                        deviceSpeak(soundlist[11])
                    elif "turnofftheairconditioner" in processedText:
                        client.publish("polibatam/homeassistant/airconditioner", "false")
                        message_to_send = "AC_OFF"
                        ser.write(message_to_send.encode())
                        deviceSpeak(soundlist[12])
                    elif "nextchannel" in processedText:
                        ser.write("TV_NEXT".encode())
                        deviceSpeak(soundlist[14])
                    elif "previouschannel" in processedText:
                        ser.write("TV_PREVIOUS".encode())
                        deviceSpeak(soundlist[13])
                    else:
                        deviceSpeak(soundlist[2])
                        isAccessed = True
                elif "hellorogee" in processedText:
                    deviceSpeak(soundList[1])
                    isAccessed = True
                else:
                    print(segment.text)
                    print("Fail Keyword")
            time.sleep(2)
            deviceLED((0, 255, 0), 0.05)
            stream.start_stream()
    except IOError:
        # Handle cases where the audio buffer overflows
        pass

pixels.fill((0, 0, 0))
time.sleep(3)
frames = []

handleIO(RELAY)
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION1)
client.on_connect = on_connect
client.on_message = on_message
client.connect(BROKER_ADRESS, 1883, 60)
client.loop_start()

deviceLED((0, 255, 0), 0.05)
stream.stop_stream()
deviceSpeak(soundList[0])
stream.start_stream()

try:
    ser = serial.Serial(
        port=PORT,
        baudrate=BAUD_RATE,
        parity=serial.PARITY_NONE,
        stopbits=serial.STOPBITS_ONE,
        bytesize=serial.EIGHTBITS,
        timeout=TIMEOUT
    )
    print(f"Connected to {PORT} at {BAUD_RATE} baud.")
    while True:
        listening()
except KeyboardInterrupt:
    client.loop_stop()
    client.disconnect()
```

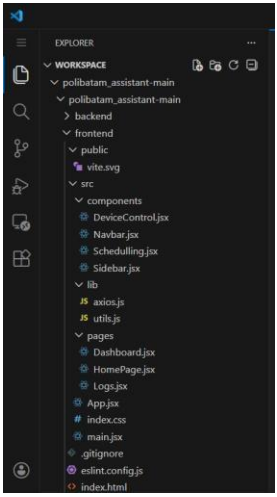
Lampiran C

Program *Web* User Interface
Output Control and *Scheduling*

a. *Backend* Workspace



b. *Frontend* Workspace



Lampiran D

Berikut merupakan link video dari kegiatan simulasi project TA kami :
https://drive.google.com/drive/folders/1nIh6xyYCZVlF6NxjVe8pqLyRje4s-ugI

Lembar Revisi Sidang Tugas Akhir
Program Studi Teknologi Rekayasa Elektronika

Nama : Erik Chaniago dan Billy Hagas Tarigan
NIM : 4242211007 dan 4242211037
Judul Tugas Akhir : Sistem Pengendalian Perangkat *Smart Home* dengan Perintah Suara Menggunakan *Raspberry Pi* dan *Smartphone*
Tanggal Sidang : Kamis, 15 Januari 2026

No	Nama Penguji	Saran Perbaikan	Perbaikan	TTD Penguji
1	Ririn Humaera, M.Pd	1. Perhatikan bahasa asing, masih banyak ditemukan belum berhuruf miring 2. Tabel dan Daftar isi di cek kembali banyak yang tidak sesuai dengan nomor halaman. 3. Perlu ditambahkan data pengujian *noise* dan tanpa noise 4. Perbaiki kalimat sesuai dengan daftar pustaka	Seluruh Halaman, Halaman vi dan vii, Halaman 39, Halaman 7	
2	Ika Karlina Laila Nur Suciningtyas, S.Si.,M.Si	1. Tambahkan poin batasan masalah penelitian *smartphone* bukan menggunakan suara tetapi menggunakan website fitur klik/tombol	Halaman 3	

Nama Pembimbing

Illa Aryeni,S.T., M.T.
NIK; 122255