Year: 2026, Volume: 1, Number: 2, Pages: 87-102

Journal of Advanced Vocational Information and Communication Technology

Homepage: https://journal.isas.or.id/index.php/javict, email: javict@isas.or.id

Voice Command Smart Home Control on Single-board Computer Using Google Speech API and Cosine Similarity

Mochammad Ilham Maulana[1], Reksa Prastama Putra[1], Zaky Wahyu Oktavianto[1]*

[1,*]Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Indonesia

*Corresponding Author: reksaprastama1905@gmail.com

Article Info

Article History:
Submitted 24 Apr 2026
First Revised 7 May 2026
Accepted 21 May 2026
First Available Online 22 May 2026
Publication Date 25 May 2026

Keyword:
Smart home,
speech recognition,
text processing,
voice command,
single board computer

Abstract

The development of technology today is very fast and sophisticated. One example of technological developments is the smart home. A smart home is a house or building equipped with high technology that allows various systems and devices in the home to communicate with one another. This technology is very helpful for parents and people with disabilities in activities at home. One way of controlling smart homes currently being developed is voice control. However, some smart home devices do not yet have control features using voice commands in Indonesian. This study aims to create and test a smart home device that can be controlled by voice commands using the Google Speech-to-Text Application Programming Interface (API) as a medium for capturing and converting voice commands from users into text. The results of the conversion are processed using the Cosine Similarity method to determine what commands are given and which devices the user wants to control. This process is carried out on a Single Board Computer (SBC), specifically the Raspberry Pi. The system achieves up to 100% accuracy in quiet conditions with an average response time of 1001.4 ms. However, this accuracy drops to approximately 80% in noisy environments above 65dB, indicating limitations of the Google Speech API under high noise conditions.

1. Introduction`

The development of technology today is very fast and sophisticated. One example of technological development is Smart Home. Smart home is a house or building that is equipped with high technology that allows various systems and devices in the house to

communicate with one another. With this smart home technology, it will be easier for users to control the use of electronic devices in the home [17]. This is very helpful also for parents and people with disabilities in activities at home. One way to control smart home that is being developed at this time is controlling using sound.

In addition, Speech Recognition is a process for recognizing spoken words that are captured by the microphone and converted into digital data. Many technologies have been created that are able to understand and respond to utterances, such as the Jasper platform, Google Speech API, Alexa Voice Services, and Bing Speech API [1]. With this technology, we can control all electronic equipment through our voice commands. Currently speech recognition technology has been widely used as in Google Assistant, Siri, and Cortana.

In some previous research, none of these studies has implemented the Google Speech Application Programming Interface (API) as a voice recognizer on smart home devices combined with text processing in Indonesian. Recent studies have shown growing interest in voice-controlled smart home systems using various approaches. Iliev and Ilieva [21] proposed an IoT-fog-cloud framework combining cloud-based speech-to-text with natural language processing (NLP) for smart home control, demonstrating that utterance-to-command transformation can be adapted for non-English languages. Irugalbandara et al. [22] developed a secure smart home automation system with offline speech recognition capabilities, addressing internet dependency and cybersecurity vulnerabilities commonly found in cloud-based systems. Alshammri [23] explored deep learning-based source separation to improve voice command recognition accuracy in multi-device smart home environments with background noise. The present research addresses the gap by implementing Google Speech API with Cosine Similarity-based text processing specifically for Indonesian voice commands on a Raspberry Pi platform.

Table 1. Comparison with Previous Research

Research	Smart Home	Voice Command	SBC	Text Processing
[2]	Yes	Amazon Alexa	Raspberry Pi	No
[3]	Yes	Google Home	Raspberry Pi	TF-IDF
[4]	Yes	WebSpeech API	Wemos D1	Dialogflow API
[5]	Yes	Google Asst.	Node MCU	No
[8]	Yes	AMR Voice	Arduino Mega	No
[9]	Yes	MIT App Inv.	Wemos D1	No
[10]	Yes	Voice Module	Arduino	RNN
[18]	Yes	FB Messenger	Raspberry Pi	No
[20]	Yes	Alexa Voice	Raspberry Pi	No
[21]	Yes	Cloud STT + NLP	IoT-Fog-Cloud	NLP Methods
[22]	Yes	Offline ASR	—	No
[23]	Yes	Voice + Deep Learning	—	Deep Learning
Proposed	Yes	Google Speech API	Raspberry Pi	Cosine Similarity

DOI: *https://doi.org/10.52158/javict.v1i2.1545*
ISSN: *3124-0437 (media online)*

From the comparison above, the proposed research is unique in combining Google Speech API with Cosine Similarity-based text processing to handle varied Indonesian voice commands on a Raspberry Pi-based smart home system

2. Methods

This research creates a Smart Home device that can control and monitor electronic devices remotely, implementing a voice command feature using the Google Speech API so users can control home devices using Indonesian. The Google Speech API converts voice commands into text, which is then processed using the Cosine Similarity method to match against the existing command dataset.

2.1. System Design

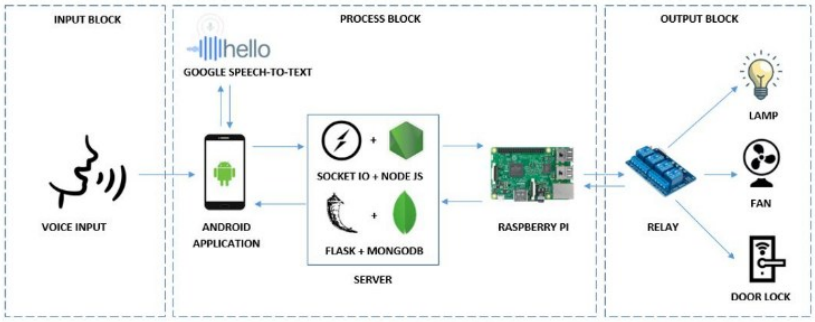

Figure 1. System Design of this research

Users open the application and give voice commands to control home devices such as turning lights on or off. The voice command is captured by the Android application and converted by the Google Speech-to-Text API into text, which is displayed on the home page. The text is then sent to the server using Socket IO, which forwards it to the Raspberry Pi. The Raspberry Pi performs text processing using the Cosine Similarity method to match the received text against command sentences in the dataset, then sends the appropriate command to the Relay module to control the desired home appliance. After the device changes state, the Raspberry Pi sends the updated device status back to the server, which forwards it to the Android application for display.

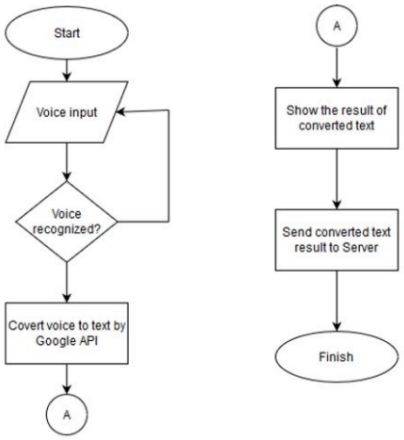

Figure 2. Flowchart of voice command processing

In the voice command input activity, the user says a voice command corresponding to the desired home device. The application checks whether the voice can be recognized. If not, the user is asked to repeat the command. When recognized, the Google Speech-to-

Text API converts it to text, which is displayed on the home page and sent to the server via Socket IO to be forwarded to the Raspberry Pi.

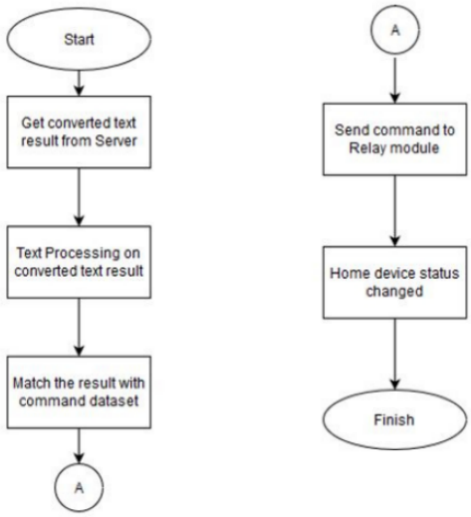

Figure 3. Flowchart activity of Raspberry Pi

The Raspberry Pi program listens for data from the server using Socket IO. After receiving data, it performs text processing using the Cosine Similarity method, and forwards the result as a command to the Relay module to control the desired home device.

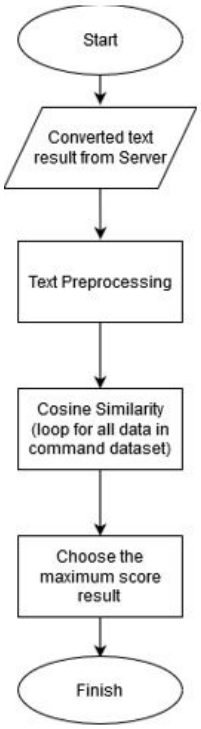

Figure 4. Flowchart activity of text processing

The text conversion results received from the server are first selected through text pre-processing stages to become more structured. The processed data is then matched with the existing command sentence database using the Cosine Similarity method, and the command with the highest similarity score is forwarded to the Relay module to control the desired home device.

***DOI:** https://doi.org/10.52158/javict.v1i2.1545*
***ISSN:** 3124-0437 (media online)*

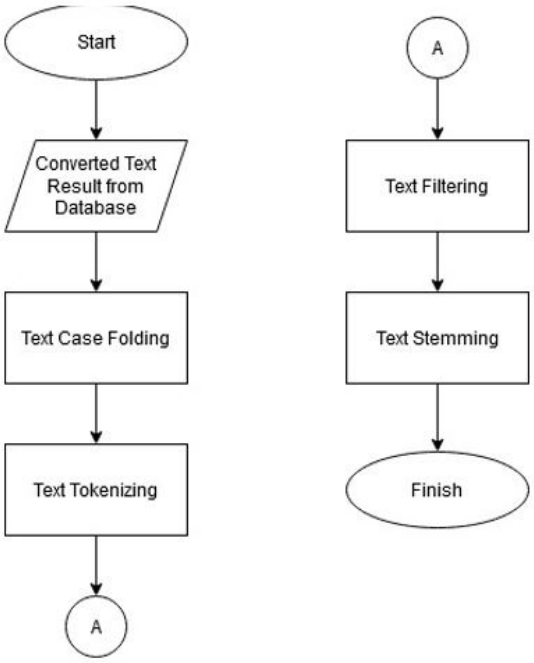

Figure 5. Flowchart activity of text pre-processing

The text pre-processing stage processes the input data through four sequential steps: Case Folding, Tokenizing, Filtering, and Stemming, resulting in a clean and structured sentence ready to be matched against the command dataset.

2.2. Text Pre-processing

The text pre-processing stage consists of four steps: Case Folding, Tokenizing, Filtering, and Stemming. Case Folding converts all text into lowercase as shown in Table 2:

Table 2. Illustration of Case Folding

Data Review	Case Folding Result
Turn off garage lamp *(Matikan lampu Garasi)*	turn off garage lamp *(matikan lampu garasi)*
Turn on garage lamp please *(Nyalain lampu Garasi dong)*	turn on garage lamp please *(nyalain lampu garasi dong)*
Open door lock *(Bukain kunci pintu)*	open door lock *(bukain kunci pintu)*

Text Tokenizing cuts the input string based on each word as shown in Table 3.

Table 3. Illustration of Tokenizing

Data Review	Tokenizing Result
turn off garage lamp *(matikan lampu garasi)*	*'matikan'* (turn off), *'lampu'* (lamp), *'garasi'* (garage)
turn on garage lamp please *(nyalain lampu garasi dong)*	*'nyalain'* (turn on), *'lampu'* (lamp), *'garasi'* (garage), *'dong'* (please)
open door lock *(bukain kunci pintu)*	*'bukain'* (open), *'kunci'* (lock), *'pintu'* (door)

Text Filtering removes stopwords (less important words) as shown in Table 4.

Table 4. Illustration of Filtering

Data Review	Filtering Result
turn off garage lamp *(matikan lampu garasi)*	turn off garage lamp *(matikan lampu garasi)*
turn on garage lamp please *(nyalain lampu garasi dong)*	turn on garage lamp *(nyalain lampu garasi)*
open door lock *(bukain kunci pintu)*	open door lock *(bukain kunci pintu)*

Text Stemming groups words to their base form as shown in Table 5.

Table 5. Illustration of Stemming

Data Review	Stemming Result
turn off garage lamp *(matikan lampu garasi)*	turn off garage lamp *(mati lampu garasi)*
turn on garage lamp *(nyalain lampu garasi)*	turn on garage lamp *(nyala lampu garasi)*
open door lock *(bukain kunci pintu)*	open door lock *(buka kunci pintu)*

2.3. Cosine Similarity Method

Cosine Similarity is a method used to measure the degree of similarity between two text vectors. It calculates the cosine of the angle between two non-zero vectors in a multi-dimensional space. The formula is defined as follows:

$$cos\theta = \frac{(A \cdot B)}{(\| A \| \times \| B \|)} \quad (1)$$

Where A and B are the vector representations of two text sentences, A · B is the dot product of the two vectors, and ||A|| and ||B|| are the Euclidean norms (magnitudes) of the vectors, respectively. The resulting value ranges from 0 to 1, where 1 indicates identical sentences and 0 indicates no similarity.

In this system, both the input text (after pre-processing) and each command sentence in the dataset are represented as term-frequency vectors. The Cosine Similarity score is computed between the input vector and every command vector in the dataset. The command sentence with the highest similarity score is selected as the matching command and forwarded to the Relay module. A threshold is applied to prevent false command execution when similarity scores are too low

3. Results and Discussion

3.1. Testing Environment

All experiments were conducted in a controlled indoor laboratory environment at the Wireless Sensor Network Laboratory, Politeknik Elektronika Negeri Surabaya (PENS), Surabaya, Indonesia. The test room measured approximately 4 × 5 meters. A Samsung Android smartphone was used to run the Android application and issue voice commands. The smartphone microphone was positioned approximately 20–30 cm from the user's mouth during all tests. A sound level meter was used to measure ambient noise levels in

***DOI:** https://doi.org/10.52158/javict.v1i2.1545*
***ISSN:** 3124-0437 (media online)*

decibels (dB). Noise was introduced using a speaker playing crowd noise at varying volumes to simulate real-world conditions. The Raspberry Pi 3 Model B+ served as the Single Board Computer (SBC) for text processing, connected to the server via a local Wi-Fi network.

3.2. System Development

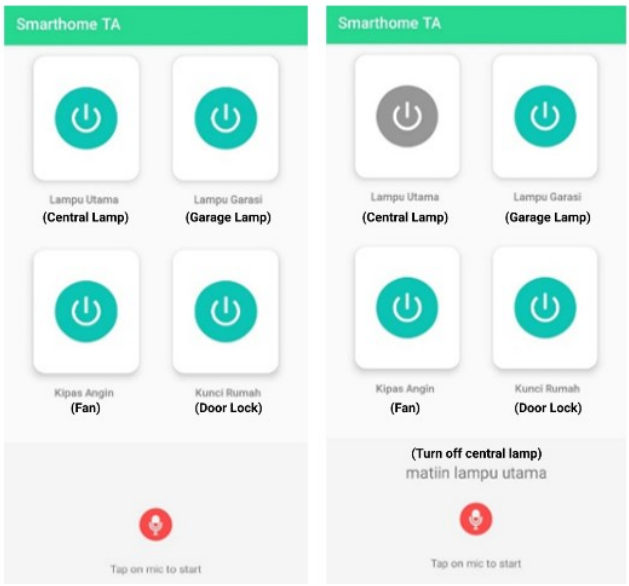

Figure 6. Android application before and after given voice commands

This system requires an Android application that will be used by users to give commands either through buttons or voice to control the desired home device. The application was developed using Android Studio and has two options: button control or voice command. For voice commands, the Google Speech API is implemented to convert voice commands from the user into text. The text is sent to the server via Socket IO and displayed on the main page.

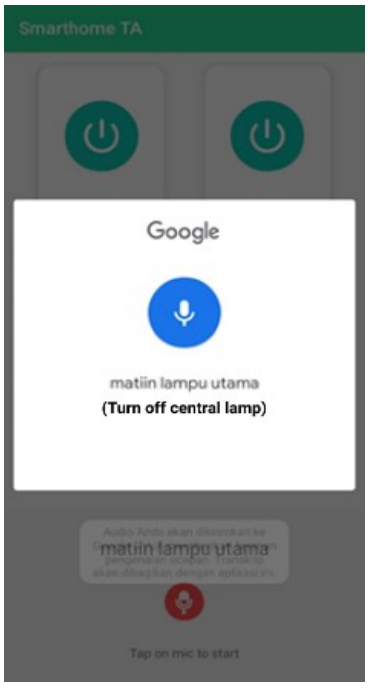

Figure 7. Android application after given voice commands

To enable communication between the Android application and the Raspberry Pi, the authors created a server implementing Socket IO as a communicator and MongoDB as the database. Socket.IO is a library that allows real-time, two-way, event-based communication between browser and server. MongoDB is a document-based NoSQL database with JSON format.

Figure 8. JSON Data Home Device Status stored on MongoDB

Socket.IO sends the results of text conversion from the Android application to Raspberry Pi for further processing. After the Relay processes the command and changes the home device state, the Raspberry Pi sends the device condition back to the MongoDB server, which the Android application then retrieves to display the latest status.

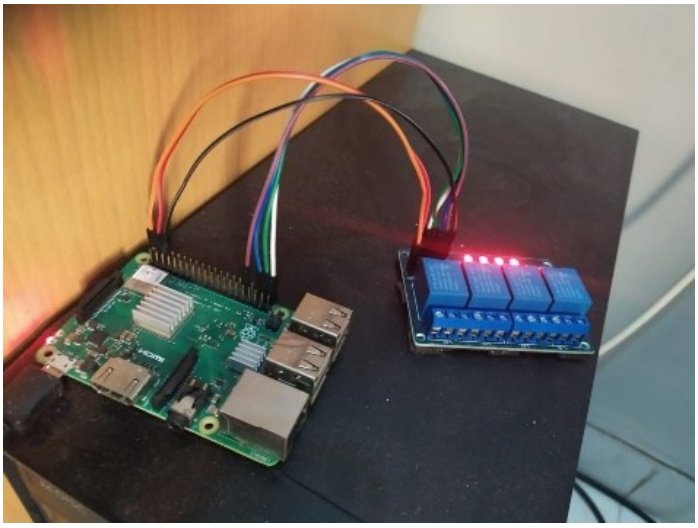

Figure 9. Raspberry Pi and Relay set-up

***DOI:** https://doi.org/10.52158/javict.v1i2.1545*
***ISSN:** 3124-0437 (media online)*

Figure 10. Implementation of the system on a miniature house

3.3. Android Application Functionality Test

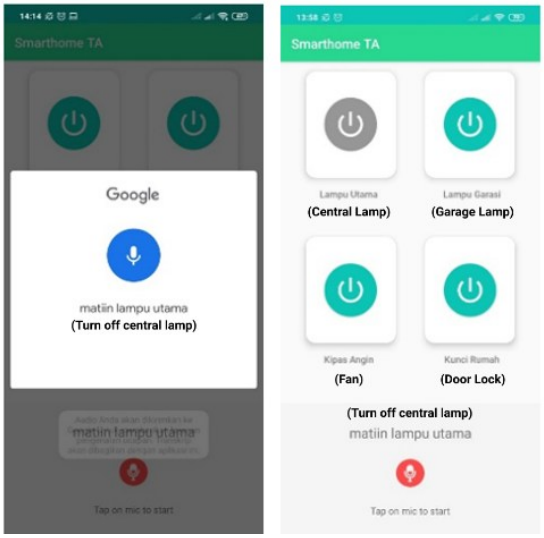

Figure 11. Display of Android applications that have been made

Testing was done to verify the functionality of the Android application in capturing voice commands and converting them to text through Google Speech-to-Text API. Results confirmed that the application can successfully convert voice commands to text and display them on the front page, as well as control desired home devices using the available buttons.

3.4. Testing with Various Voice Commands

In this experiment, all home devices were controlled by giving voice commands with various sentences to test the accuracy of the Text Processing method. Table 6 shows the command dataset recognized by the system.

Table 6. Voice Commands Dataset

Voice Commands Dataset
Turn on/off garage lamp (nyala/mati/hidup/padam lampu garasi), Turn on/off central lamp (nyala/mati/hidup/padam lampu utama), Turn on/off fan (nyala/mati/hidup/padam kipas angin), Open door lock (buka kunci pintu), Close door lock (tutup kunci pintu)

Table 7. Test Results with Various Voice Commands

No	Voice Command	After Text Processing	Initial State	Final State	Match?
1	Turn off garage lamp (Padamkan lampu garasi)	Turn off garage lamp (padam lampu garasi)	On	Off	Yes
2	Turn off central lamp please (Matiin lampu utama dong)	Turn off central lamp (mati lampu utama)	On	Off	Yes
3	Turn on central lamp please (Nyalakan lampu utama dong)	Turn on central lamp (nyala lampu utama)	Off	On	Yes
4	Turn on garage lamp (Hidupin lampu garasi)	Turn on garage lamp (hidup lampu garasi)	Off	On	Yes
5	Turn on fan (Nyalain kipas angin)	Turn on fan (nyala kipas angin)	Off	On	Yes
6	Turn on fan (Hidupkan kipas angin)	Turn on fan (hidup kipas angin)	Off	On	Yes
7	Turn off fan (Matikan kipas angin)	Turn off fan (mati kipas angin)	On	Off	Yes
8	Turn off fan (Padamin kipas angin)	Turn off fan (padam kipas angin)	On	Off	Yes
9	Open door lock (Buka kunci pintu)	Open door lock (buka kunci pintu)	Close	Open	Yes
10	Close door lock (Tutup kunci pintu)	Close door lock (tutup kunci pintu)	On	Off	Yes

From the above results, the Smart Home system was able to control all home devices correctly across all 10 tested command variations. It should be noted that these results represent performance under quiet conditions (below 50dB) with a limited set of 5 command types. The accuracy observed in these controlled tests may not fully reflect real-world performance with a wider range of vocabulary or accent variations.

3.5. Response Time Testing

Table 8. Calculation of System Response Time

No	Voice Command	Initial State	Final State	Response Time (ms)
1	Turn off garage lamp (Padamkan lampu garasi)	On	Off	1016 ms
2	Turn off central lamp please (Matiin lampu utama dong)	On	Off	991 ms
3	Turn on central lamp please (Nyalakan lampu utama dong)	Off	On	1013 ms
4	Turn on garage lamp (Hidupin lampu garasi)	Off	On	998 ms
5	Turn on fan (Nyalain kipas angin)	Off	On	999 ms
6	Turn on fan (Hidupkan kipas angin)	Off	On	999 ms
7	Turn off fan (Matikan kipas angin)	On	Off	999 ms

DOI: *https://doi.org/10.52158/javict.v1i2.1545*
ISSN: *3124-0437 (media online)*

8	Turn off fan (Padamin kipas angin)	On	Off	1004 ms
9	Open door lock (Buka kunci pintu)	Close	Open	995 ms
10	Close door lock (Tutup kunci pintu)	On	Off	1000 ms

Table 8 shows the system response time results with an average of 1001.4 milliseconds (ms) to control the desired home device.

3.6. Testing with Various Noise Levels (dB)

In this experiment, home devices were controlled by voice commands at various noise levels to test the accuracy of Google Speech-to-Text API under different crowd conditions: below 50dB, 51–65dB, and above 65dB.

Table 9. Test Results with Noise Level Below 50 dB

No	Noise Level (dB)	Voice Command	Sentence Detected	Final State	Match?
1	45.3dB	Turn off garage lamp *(Padamkan lampu garasi)*	Turn off garage lamp *(Padamkan lampu garasi)*	Off	Yes
2	44.5dB	Turn off central lamp please *(Matiin lampu utama dong)*	Turn off central lamp please *(Matiin lampu utama dong)*	On	Yes
3	47.1dB	Turn on central lamp please *(Nyalakan lampu utama dong)*	Turn on central lamp please *(Nyalakan lampu utama dong)*	Off	Yes
4	43.5dB	Turn on garage lamp *(Hidupin lampu garasi)*	Turn on garage lamp *(Hidupin lampu garasi)*	On	Yes
5	45.5dB	Turn on fan *(Nyalain kipas angin)*	Turn on fan *(Nyalain kipas angin)*	On	Yes
6	46.9dB	Turn on fan *(Hidupkan kipas angin)*	Turn on fan *(Hidupkan kipas angin)*	Off	Yes
7	44.8dB	Turn off fan *(Matikan kipas angin)*	Turn off fan *(Matikan kipas angin)*	On	Yes
8	43.2dB	Turn off fan *(Padamin kipas angin)*	Turn off fan *(Padamin kipas angin)*	Off	Yes
9	45.3dB	Open door lock *(Buka kunci pintu)*	Open door lock *(Buka kunci pintu)*	Open	Yes
10	42.4dB	Close door lock *(Tutup kunci pintu)*	Close door lock *(Tutup kunci pintu)*	Close	Yes

Table 10. Test Results with Noise Level 51–65 dB

No	Noise Level (dB)	Voice Command	Sentence Detected	Final State	Match?
1	58.7dB	Turn off garage lamp *(Padamkan lampu garasi)*	Turn off garage lamp *(Padamkan lampu garasi)*	Off	Yes

2	63.4dB	Turn off central lamp please *(Matiin lampu utama dong)*	Turn off central lamp please *(Matiin lampu utama dong)*	On	Yes
3	61.5dB	Turn on central lamp please *(Nyalakan lampu utama dong)*	Turn on central lamp please *(Nyalakan lampu utama dong)*	Off	Yes
4	64.2dB	Turn on garage lamp *(Hidupin lampu garasi)*	Turn on garage lamp *(Hidupin lampu garasi)*	On	Yes
5	55.1dB	Turn on fan *(Nyalain kipas angin)*	Turn on fan *(Nyalain kipas angin)*	On	Yes
6	59.3dB	Turn on fan *(Hidupkan kipas angin)*	Turn on fan *(Hidupkan kipas angin)*	Off	Yes
7	62.3dB	Turn off fan *(Matikan kipas angin)*	Turn off fan *(Matikan kipas angin)*	On	Yes
8	58.6dB	Turn off fan *(Padamin kipas angin)*	Turn off fan *(Padamin kipas angin)*	Off	Yes
9	54.5dB	Open door lock *(Buka kunci pintu)*	Open door lock *(Buka kunci pintu)*	Open	Yes
10	53.dB	Close door lock *(Tutup kunci pintu)*	Close door lock *(Tutup kunci pintu)*	Close	Yes

Table 11. Test Results with Noise Level Above 65 dB

No	Noise Level (dB)	Voice Command	Sentence Detected	Final State	Match?
1	72.4dB	Turn off garage lamp *(Padamkan lampu garasi)*	Turn off garage lamp *(Padamkan lampu garasi)*	Off	Yes
2	78.6dB	Turn off central lamp please *(Matiin lampu utama dong)*	Turn off *(Matiin)*	On	No
3	81.3dB	Turn on central lamp please *(Nyalakan lampu utama dong)*	Turn on central lamp please *(Nyalakan lampu utama dong)*	Off	Yes
4	88.3dB	Turn on garage lamp *(Hidupin lampu garasi)*	Turn on Galaxy lamp *(Hidupin lampu Galaksi)*	Off	No
5	74.5dB	Turn on fan *(Nyalain kipas angin)*	Turn on fan *(Nyalain kipas angin)*	On	Yes
6	77.9dB	Turn on fan *(Hidupkan kipas angin)*	Turn on fan *(Hidupkan kipas angin)*	Off	Yes
7	80.5dB	Turn off fan *(Matikan kipas angin)*	Turn off fan *(Matikan kipas angin)*	On	Yes
8	70.4dB	Turn off fan *(Padamin kipas angin)*	Turn off fan *(Padamin kipas angin)*	Off	Yes
9	75.4dB	Open door lock *(Buka kunci pintu)*	Open door lock *(Buka kunci pintu)*	Open	Yes

DOI: *https://doi.org/10.52158/javict.v1i2.1545*
ISSN: *3124-0437 (media online)*

10	72.3dB	Close door lock *(Tutup kunci pintu)*	Close door lock *(Tutup kunci pintu)*	Close	Yes

Based on the trial results, the Smart Home device operated with 100% accuracy at noise levels below 65dB, where the Google Speech API was able to correctly recognize all voice commands. However, when the noise level exceeded 65dB, the system accuracy dropped to approximately 80% (2 out of 10 commands failed). The two failure cases observed were: (1) at 78.6dB, the system only captured a partial command ("Matiin"), which did not match any complete dataset entry; and (2) at 88.3dB, the word "garasi" (garage) was misrecognized as "Galaksi" (galaxy) due to phonetic similarity under high noise interference. These failures indicate that the Google Speech API is sensitive to high ambient noise and that phonetically similar words are more prone to misrecognition. Figure 12 below illustrates the trend of system accuracy across the three noise level conditions, showing a clear performance drop-off above 65dB.

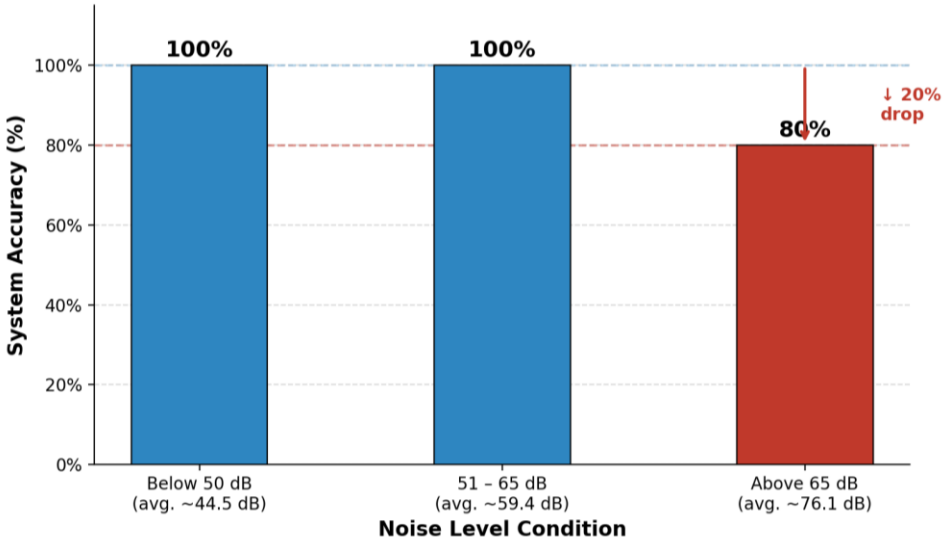

Figure 12. Graph of System Accuracy vs. Noise Level (dB)

4. Conclusion

From the results of testing and analysis carried out during this research, it can be concluded that the use of the Google Speech API as a technology to convert voice into text gives excellent results under low-to-moderate noise conditions. However, the system has limitations in high-noise environments above 65dB, where the Google Speech API sometimes provides incorrect conversion results, resulting in approximately 80% accuracy. This limitation should be acknowledged rather than overlooked. The Cosine Similarity method for text processing shows very good results in converting text into relay commands. The Android application functions properly, allowing users to control home devices using voice commands or buttons, and to monitor the current status of home devices. The system achieves up to 100% accuracy in device control under quiet conditions with an average response time of 1001.4 ms (1 second).

For future research, it is suggested to: expand the command dataset with more varied synonyms and sentence structures to improve robustness; increase the number of test trials per condition to at least 30–50 for stronger statistical validity; add sensors such as temperature, humidity, gas, or smoke sensors for more detailed home monitoring; implement offline features using Bluetooth or Wi-Fi 802.11 modules so users can control devices without an internet connection; apply noise cancellation or a more noise-robust

ASR method to address the performance degradation observed above 65dB; and implement voice commands to open applications without touching the smartphone.

5. Acknowledgment

We would like to thank the EEPIS Wireless Sensor Network Laboratory of the Electronic Engineering Polytechnic of Surabaya (PENS), which provided complete facilities to finish this research.

6. Author's Note

The authors hereby declare that there is no conflict of interest related to the publication of this article. Furthermore, the authors confirm that the manuscript is original and free from any form of plagiarism.

7. References

[1] T. Erić, S. Ivanović, S. Milivojsa, and M. Matić, "Voice Control for Smart Home Automation," in IEEE 7th Int. Conf. Consumer Electronics (ICCE-Berlin), 2017.

[2] C. Z. Yue and S. Ping, "Voice Activated Smart Home Design and Implementation," in 2nd Int. Conf. Frontiers of Sensors Technologies (ICFST), 2017.

[3] C.-Y. Peng and R.-C. Chen, "Voice Recognition by Google Home and Raspberry Pi for Smart Socket Control," in 10th Int. Conf. Advanced Computational Intelligence (ICACI), 2018.

[4] G. Alexakis, S. Panagiotakis, A. Fragkakis, E. Markakis, and K. Vassilakis, "Control of Smart Home Operations Using Natural Language Processing, Voice Recognition and IoT Technologies," Designs, vol. 3, no. 3, 2019.

[5] H. Isyanto, A. S. Arifin, and M. Suryanegara, "Design and Implementation of IoT-Based Smart Home Voice Commands for Disabled People Using Google Assistant," in Int. Conf. Smart Technology and Applications (ICoSTA), 2020.

[6] L. Salman et al., "Energy Efficient IoT-Based Smart Home," in IEEE 3rd World Forum on Internet of Things (WF-IoT), 2016.

[7] M. G. M. and C. Vyjayanthi, "Implementation of Cost-Effective Smart Home Controller with Android Application using Node MCU and IoT," in 2nd Int. Conf. Power, Energy and Environment (ICEPE), 2018.

[8] M. E. Abidi et al., "Development of Voice Control and Home Security for Smart Home Automation," in 7th Int. Conf. Computer and Communication Engineering (ICCCE), 2018.

[9] M. F. Riza and N. S. Salahuddin, "Control Home Devices with Voice Commands via a Smartphone," in 4th Int. Conf. Informatics and Computing (ICIC), 2019.

[10] M. KM et al., "An Interactive Voice Controlled Humanoid Smart Home Prototype Using NLP and Machine Learning," in 3rd IEEE Int. Conf. RTEICT, 2018.

***DOI:** https://doi.org/10.52158/javict.v1i2.1545*
***ISSN:** 3124-0437 (media online)*

[11] N. Anggraini, L. K. Wardhani, A. Kurniawan, and N. Hakim, "Speech Recognition Application for the Speech Impaired using Android-based Google Cloud Speech API," TELKOMNIKA, vol. 16, no. 6, 2018.

[12] S. Mahmud, S. Ahmed, and K. Shikder, "A Smart Home Automation and Metering System using IoT," in Int. Conf. Robotics, Electrical and Signal Processing Techniques (ICREST), 2019.

[13] S. Bajpai and D. Radha, "Smart Phone as a Controlling Device for Smart Home using Speech Recognition," in Int. Conf. Communication and Signal Processing (ICCSP), 2019.

[14] T. Malche and P. Maheshwary, "Internet of Things (IoT) for Building Smart Home System," in I-SMAC 2017.

[15] V. Govindraj, M. Sathiyanarayanan, and B. Abubakar, "Customary Homes to Smart Homes Using IoT and Mobile Application," in Int. Conf. Smart Technology for Smart Nation, 2017.

[16] V. Raj, A. C. BS, and A. P. RS, "IoT Based Smart Home Using Multiple Language Voice Commands," in 2nd ICICICT, 2019.

[17] A. Prasetiyo, "I-On Smart Controller Solusi Smart Home Portable Berbasis Arduino dan Raspberry PI," PENS, Surabaya, 2017, Unpublished.

[18] A. K. Azzam, W. Kurniawan, and M. H. H. Ichsan, "Implementasi Pengontrolan Smart Home Menggunakan Voice Command Pada Facebook Messenger," J. Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 3, no. 5, 2019.

[19] D. I. S. Saputra, S. W. Handani, and G. A. Diniary, "Pemanfaatan Cloud Speech API Untuk Pengembangan Media Pembelajaran Bahasa Inggris," J. Telematika, vol. 10, no. 2, 2017.

[20] M. I. Reza, S. R. Akbar, and H. Fitriyah, "Kontrol Lampu Berbasis Voice Command Pada Raspberry PI," J. Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 2, no. 9, 2018.

[21] Y. Iliev and G. Ilieva, "A Framework for Smart Home System with Voice Control Using NLP Methods," Electronics, vol. 12, no. 1, p. 116, 2023. https://doi.org/10.3390/electronics12010116

[22] C. Irugalbandara, A. S. Naseem, S. Perera, S. Kiruthikan, and V. Logeeshan, "A Secure and Smart Home Automation System with Speech Recognition and Power Measurement Capabilities," Sensors, vol. 23, no. 13, p. 5784, 2023. https://doi.org/10.3390/s23135784

[23] G. H. Alshammri, "IoT-Based Voice-Controlled Smart Homes with Source Separation Based on Deep Learning," Journal of Sensors, vol. 2023, Article ID 1911385, 2023. https://doi.org/10.1155/2023/1911385

Authors Biographies

Mochammad Ilham Maulana is received his degree in Informatics Engineering from Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. During his studies, he was an active member of the MSECT (Multimedia, Security, and Computer Network) research team at PENS. His research interests include Cloud Computing, Internet of Things (IoT), and Smart Home Systems.
Email: ilhammaulana@it.student.pens.ac.id
ORCID: -

Reksa Prastama Putra received his bachelor's degree in Informatics Engineering from Universitas Dian Nuswantoro, Indonesia. He is currently pursuing his master's degree in the Department of Informatics and Computer Engineering at Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. He is also a member of the MSECT (Multimedia, Security, and Computer Network) research team at PENS. His research interests include Embedded Systems, Edge Computing, and Smart Plantation Systems
Email: reksaprastama1905@gmail.com
ORCID: -

Zaky Wahyu Oktavianto received his bachelor's degree in Informatics Engineering from Universitas 17 Agustus 1945 Surabaya, Indonesia. He is currently pursuing his master's degree in the Department of Informatics and Computer Engineering at Politeknik Elektronika Negeri Surabaya (PENS), Indonesia. He is also a member of the MSECT (Multimedia, Security, and Computer Network) research team at PENS. His research interests include Smart Home Systems, Embedded Systems, Edge Computing, and Image Processing.
Email: zayuto33@gmail.com
ORCID: -

***DOI:** https://doi.org/10.52158/javict.v1i2.1545*
***ISSN:** 3124-0437 (media online)*