

Prototype Development of a Smart Voice-Controlled Audio System Based on the Raspberry Pi Platform

(Pembangunan Prototaip Sistem Audio Pintar Kawalan-Suara Berasaskan Platform Raspberry Pi)

Weng Thong Kuan^a & Muhammad Faiz Bukhori^{a,b,*}

^aElectrical and Electronic Engineering Programme,

^bCentre for Integrated Systems Engineering and Advanced Technologies (INTEGRA),
Faculty of Engineering & Built Environment, Universiti Kebangsaan Malaysia, Malaysia

Abdul Halim Ismail

CITRA Centre, Universiti Kebangsaan Malaysia, Malaysia

*Corresponding author: mfaiz_b@ukm.edu.my

Received 26 August 2019, Received in revised form 12 July 2019

Accepted 30 August 2019, Available online 31 October 2019

ABSTRACT

We report a prototype audio system that can be controlled by means of voice commands. The operations of this audio player – such as song playback and volume control – can be effectively controlled by verbal commands alone without requiring any button-press or dial-turn, enabling a user to operate it with full hands. This feature could be particularly useful in situations when the user's view and attention are pre-occupied, as in the case of driving or cycling. A voice-command device also has the advantage of a smaller form factor and simpler product casing, due to fewer dials and buttons. While there are a wide variety of commercial audio systems, only very limited few allow voice-control operations. This voice-controlled audio system is low-cost, open-source, and capable of "smart" features that include notifying weather forecasts, automatic volume control, and automatic shutdown. Its platform is the single-board computer Raspberry Pi, which runs speech-to-text processes that record and transcribe a given voice command, such as "Play music" and "Volume up." The Raspberry Pi also connects to cloud-based services – and upon voice requests – can check and tell real-time information such as date, time, and weather. The system's average response time to a given voice command was measured to be about 5 seconds. A simple circuit of LED indicators was also designed to prompt users to give voice commands at the right window of time, therefore improving system responsiveness and overall user experience. This prototype can be extended to voice-control other appliances in an integrated smart home environment.

Keywords: Audio system; Voice command; Voice recognition; Consumer electronics; Raspberry Pi; Prototyping

ABSTRAK

Kami laporkan sebuah prototaip sistem audio yang boleh dikawal melalui arahan suara. Operasi sistem audio ini – seperti main balik lagu dan kawalan pembesar suara – boleh dikawal dengan efektif menggunakan arahan vokal sahaja tanpa memerlukan sebarang tekanan-punat atau putaran-dial, lantas membolehkan pengguna mengawalinya walau dengan tangan yang penuh. Fungsi ini amat berguna dalam situasi di mana pengelihatan dan perhatian pengguna sedang digunakan untuk tugas yang lain, seperti ketika memandu atau berbasikal. Sebuah peranti arahan-suara juga mempunyai kelebihan saiz yang lebih kecil dan bingkai produk yang lebih ringkas, kerana kurangnya dial dan butang. Walaupun terdapat pelbagai jenis sistem audio komersil, hanya segelintir yang membolehkan operasi arahan-suara. Sistem audio kawalan-suara ini berkos rendah, bersumber-terbuka, dan berkebolehan fungsi "pintar" termasuk maklumat ramalan cuaca, kawalan automatik pembesar suara, dan penutupan secara automatik. Platformnya adalah komputer papan-tunggal Raspberry Pi, yang menjalankan proses-proses pertuturan-kepada-teks yang merekod dan menerjemah arahan suara seperti "mainkan muzik" dan "kuatkan bunyi." Raspberry Pi juga berhubung dengan perkhidmatan-awan – dan apabila menerima arahan suara – boleh menyemak dan memberitahu maklumat masa-nyata seperti tarikh, masa, dan cuaca. Purata masa tindak balas sistem ini terhadap arahan suara telah diukur dan didapati adalah selama 5 saat. Sebuah litar ringkas LED juga telah direka untuk membantu pengguna memberikan arahan suara pada tempoh waktu yang betul, bagi meningkatkan tindak balas sistem dan pengalaman pengguna. Prototaip ini boleh dikembangkan untuk mengawal peralatan-peralatan lain di dalam sebuah rumah pintar melalui kawalan-suara.

Kata kunci: Sistem audio; Arahan suara; Pengesanan suara; Elektronik pengguna; Raspberry Pi; Prototaip

INTRODUCTION

This is a prototype development work that demonstrates a proof-of-concept for a simple audio player that can be controlled by voice. The operations of this voice-command device (VCD) – such as song selection, playback functions, and volume control – can be controlled by verbal commands spoken in a natural human language. Not only is this feature desirable from the standpoint of convenience, it could also be particularly needed in situations where the user’s attention, view, and hands are pre-occupied, as in the case of driving or cycling.

From the design perspective, a VCD has the advantage of not requiring dedicated dials and buttons vis-à-vis a non-VCD, allowing simpler product casing, smaller form factor, and better aesthetics. In addition, there are potentially countless custom operations that can be programmatically embedded in a VCD just via software updates, unlike a non-VCD where the operations are hard-wired into the circuitries and buttons. Voice command is a speech recognition technology that enables computers to recognize and translate spoken languages into text (Anusuya & Katti 2009; Hansen & Hasan 2015; Sarma & Prasanna 2018). Speech recognition algorithms apply wide-ranging models and methods from multi-disciplinary fields of computer science, linguistics, and electronic engineering (Huang et al. 2014; Saksamudre et al. 2015). Most modern systems today utilize a deep learning method called long short-term memory (Sundermeyer et al. 2012), which is a specific form of recurrent artificial neural network (ANN) that is well-suited for the purpose of classifying and processing time-series data (Lipton et al. 2015), such as a speech. Various techniques involving variants of ANNs have also been reported to achieve remarkable capabilities of feature learning (Bengio et al. 2013; Szegedy et al. 2015), leading to wide-ranging applications that include image recognition (Egmont-Petersen et al. 2002; Kim 2010; Abdullah 2007) and machine translation (Ling et al. 2015; Norris et al. 2016).

Recent advances in deep learning (LeCun et al. 2015; Schmidhuber 2015) have led major innovations and commercial deployments (Borzo 2007; Lemley et al. 2017; Tang et al. 2017) of voice-control systems in Internet search engines (Marr 2017), virtual assistants (Google Assistant), healthcare (Johnson et al. 2014), and in-car systems (Krishnan 2018).

In this work, a simple audio player has been chosen as a target deployment of a voice-control system. This is motivated by our cursory finding that typical commercial audio systems are not capable of voice-control i.e., their operations are controlled by manual input such as buttons and dials. The limited few that do allow voice commands (Kabir 2018; “When voice meets sound,” 2017), however, are high-end variants and priced at a relative premium. It seems only natural that an audio player, which is designed for audio processing, should also be capable of being controlled by means of audio. Furthermore, most conventional audio systems particularly of the entry-level models do not have

“smart” functions such as automatic loudspeaker volume adjustment, which are deemed as high-end features (Silva 2018). Therefore, this work seeks to develop and demonstrate a “smart” voice-controlled audio system that is low-cost and open-source.

METHODOLOGY

A specific list of system specifications was initially defined, with the primary objective of demonstrating a proof-of-concept, as well as to embed “smart” features into the prototype. This was then followed by a design of the system operational flow that ensures meeting of the system specifications.

SYSTEM SPECIFICATIONS

Firstly, the audio system operations should be able to be controlled by voice commands. These are standard operations typically expected from an audio player system, which are music selection, playback functions (play, pause, stop-play), and loudspeaker volume control. The user should be able to control these operations by way of giving verbal commands in a natural language, without having to use any button, dial, or keypad. This would require the audio system to recognize the moment when a voice command is given, and then to accurately transcribe the spoken words into text for later processing.

Secondly, the audio system must be capable of the following “smart” features:

1. Automatic boot-up and shutdown at user-defined times, without requiring any user intervention.
2. Automatic control of loudspeaker volume in response to dynamic surrounding noise levels. If surrounding noise level increases, the system should automatically and proportionately increase the volume of music, and vice-versa, without any user intervention.
3. Ability to tell real-time information such as local weather, date, and current time upon getting voice commands from the user requesting for such information.

Thirdly, the audio system should have a small physical form factor, low power consumption, and low total component costs when benchmarked against commercial audio systems of comparable features.

Based on the above specifications, the single-board computer Raspberry Pi 3 Model B+ (Raspberry Pi 3 Model B+, n. d.) was chosen as the hardware platform on which this “smart” audio system would be developed. This is considering its adequate computing and on-board networking capabilities (1.4 GHz CPU, 1 GB memory, dual-band 2.5/5.0 GHz WiFi), credit-card sized form factor, low power consumption, relatively affordable cost, open-source operating system (Raspbian n. d.), and vast technical resources from an active community of hobbyists and developers (Sachdeva & Katchii 2014). For this project, the Raspberry Pi would be installed

with Jasper (Jasper n. d.), which is an open-source software platform for developing voice-controlled applications.

SYSTEM OPERATIONAL FLOW

A macro-view of the audio player’s operational flow is presented in Figure 1. Also shown in the same figure are the system modules that the operations invoke in order to deliver intended features. These modules, summarized in Table 1, are responsible for music playback, converting speech-to-text and text-to-speech, as well as providing time and weather information. The following sub-sections will explain how each of the modules operates to deliver the “smart” features.

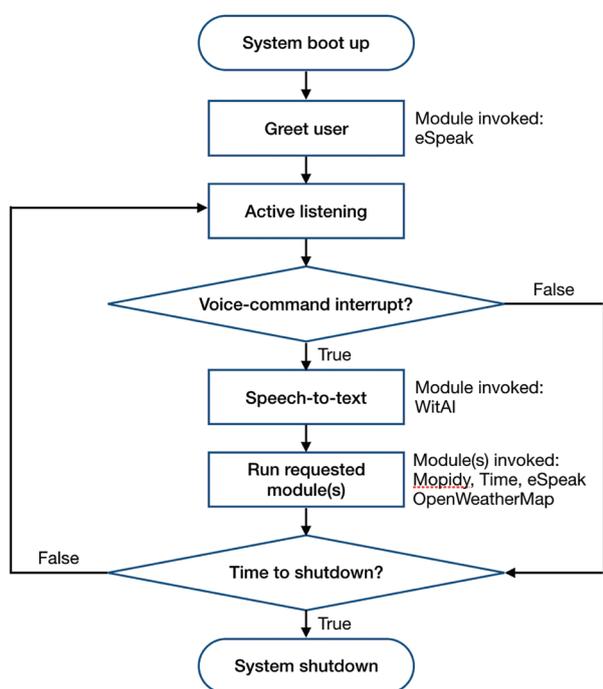


FIGURE 1. Operational flow and system modules of the “smart” audio system

TABLE 1. System modules and their functions

No.	Module	Function
1	Mopidy	A music server that can play offline and online music.
2	Date	Returns the current date and time when queried.
3	Open WeatherMap	Provides weather information for a requested location.
4	WitAI	A cloud-based service that transcribes a given audio recording.
5	eSpeak	A speech synthesizer that converts a text input into computer speech.

USER GREETING

After booting-up, the audio system will first greet the user using its system voice. This greeting serves to welcome the user, and to indicate the system is ready to receive voice commands. The system voice is synthesized using the eSpeak (eSpeak) module, which converts a text input into computer speech. The eSpeak module is also invoked anytime the system is required to “speak” back to the user in response to voice commands, as in the case of telling time or weather.

ACTIVE LISTENING

After greeting the user, the system will enter into a 4-second period of active listening; during which it waits if the user calls out its system name “John.” If the call out is not detected, the program will loop back into the active listening mode. If it is detected, an interrupt event is triggered in anticipation of a voice command. The system will then beep once to acknowledge the user call, and then initiate a speech-to-text process to execute the voice command.

SPEECH-TO-TEXT PROCESS

The overall speech-to-text process is shown in Figure 2. To retrieve the content of a user speech, the audio system relies on the cloud-based speech-recognition service by WitAI (WitAI n. d.). This would require the audio system to record the user speech, establish connection to WitAI, and then upload the speech for transcription. By using WitAI’s application programming interface (API), the audio system can send a POST request that is attached with an audio recording of the voice command, which WitAI will then respond with a transcript of the audio.

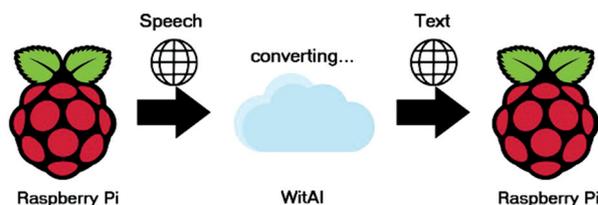


FIGURE 2. The speech-to-text process making use of speech-recognition service by WitAI to transcribe a voice command (WitAI)

RUN REQUESTED MODULE

Upon receiving the transcribed speech from WitAI, the audio system will look for specific keywords, which, if found, will then initiate the specific module associated with those keywords. For example, if the keyword “play” or “music” were found in the transcribed speech, the Mopidy (Mopidy,) music server would be initiated. If “time” or “day” were found, the Date module would be queried for the current time or day. These can be configured via Jasper’s API that links a list of user-defined keywords to specific modules.

The list of keywords and their associated modules that was configured for this prototype is shown in Table 2.

Based on the keywords in Table 2, examples of valid voice commands to start the Mopidy music server are “Play some music” and “Volume up,” because both commands contain the specific keywords linked to the music server. Mopidy can play offline playlists, as well as connect to music streaming services such as Spotify. While playing music, the audio system also periodically samples the background noise level. If successive noise levels are found to increase, the audio volume is increased proportionately as well, simulating a “smart” speaker that automatically adapts its volume based on the ambient noise level.

TABLE 2. Keywords and their associated modules configured for this prototype audio system

No.	Keywords	Associated module
1	“play,” “stop,” “music,” “volume,” “up,” “down.”	Mopidy music server
2	“time,” “day,” “date.”	Date
3	“weather”	OpenWeatherMap

Similarly, “What is the weather today?” and “Tell me the weather” are both valid voice commands to query the OpenWeatherMap module, because both have the same keyword “weather.” Upon query, the module will then obtain weather information for a given location from the World Meteorological Organization.

SHUTDOWN

After completing an interrupt event, the audio system will check for the current time and compare it against its user-defined shutdown time. It will enter into a power-standby mode if it is already shutdown time; otherwise it will loop back into the active listening mode waiting for the next command.

RESULTS AND DISCUSSION

Figure 3 shows the completed prototype of the audio system; and its schematic diagram is shown in Figure 4. A computer can be connected to the Raspberry Pi via an Ethernet cable, and can be used to monitor running processes and system messages. This connection is not required when the system is run in stand-alone mode.

BOOTING-UP AND USER GREETING

Figures 5 shows a process snapshot and system messages as the audio system boots up. It first greets the user by reading out the message, “How can I be of service, Sir?.” It then turns on the green LED shown in Figure 10, indicating to the user that it is in active listening mode and ready to receive voice commands.

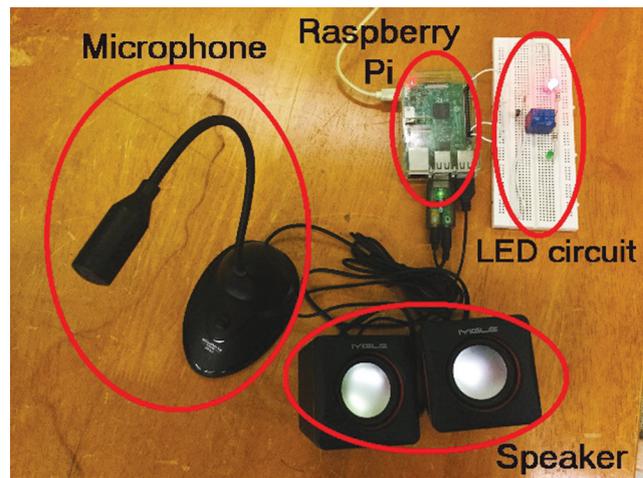


FIGURE 3. Completed prototype of the “smart” voice-controlled audio system

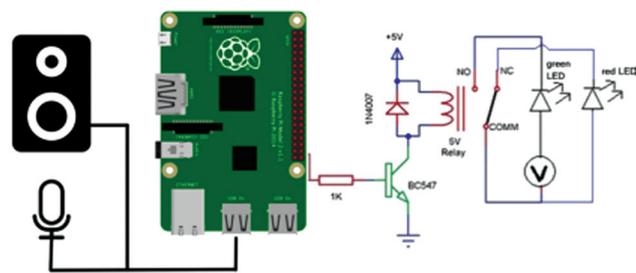


FIGURE 4. Schematic diagram of the “smart” voice-controlled audio system

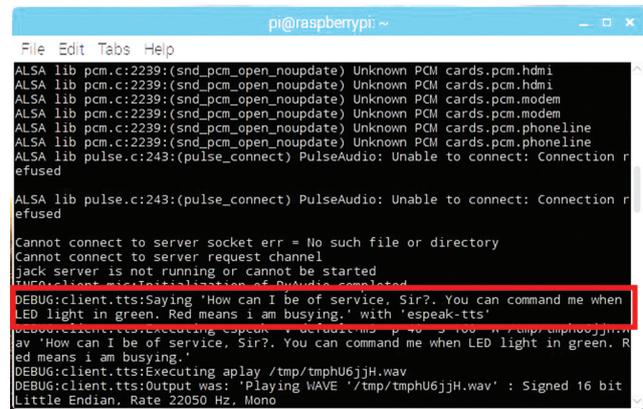


FIGURE 5. Greeting message read out by the audio system using its computer voice when booting-up

MUSIC PLAYBACK OPERATIONS

In Figure 6, a speech-to-text process that successfully transcribed and executed the voice command “Play some music” is shown. The audio system initially acknowledged its system name “John” was called. It then beeped once, and then uploaded the user voice command to WitAI to be transcribed. The audio system then invoked the Mopidy music server shown in Figure 7, after recognizing that “Play

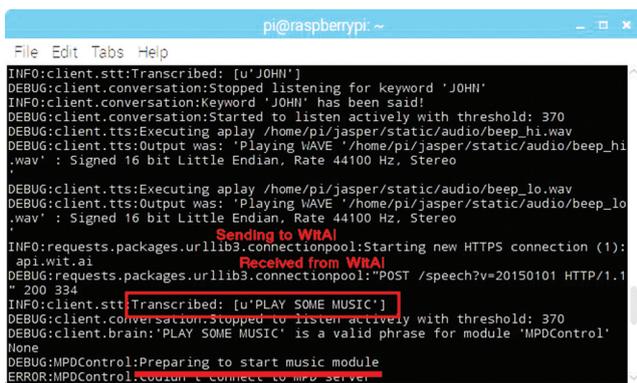


FIGURE 6. Snapshot of a successful speech-to-text process that correctly transcribed and executed the voice command “Play some music”

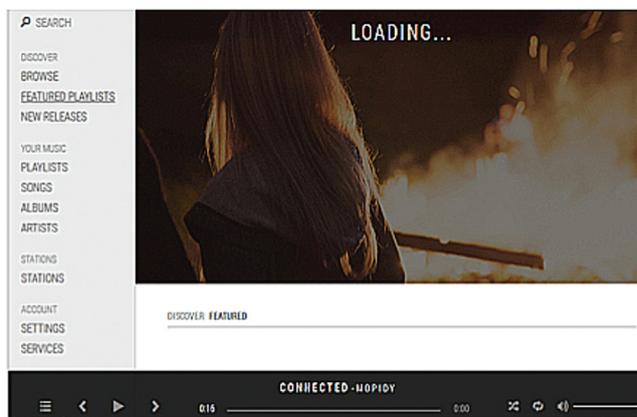


FIGURE 7. The Mopidy music server’s graphical user interface

some music” is a valid phrase to start the music player. The following voice commands were also successfully recognized and executed: “Stop music,” “Volume up,” and “Volume down.” For the case for “Volume up” and “Volume down,” the audio system responded by increasing/decreasing the speaker volume noticeably.

BACKGROUND NOISE SAMPLING AND AUTOMATIC VOLUME CONTROL

While in active listening mode, the audio system samples background noise level for an initial threshold value. This is shown in Figure 6 by the system message in line 4: “... started to listen actively with threshold: 370.” When a user speaks to the system and if this threshold value is exceeded, the system is triggered to record the speech in anticipation of a valid voice command.

When the audio system is playing music, this same process is also used to automatically adjust the audio volume if the background noise level increases. This is demonstrated in Figure 8, where the system detected an increasingly loud background noise. It then responded by updating the noise threshold value to 680, and then increasing the music volume by 10 increments.

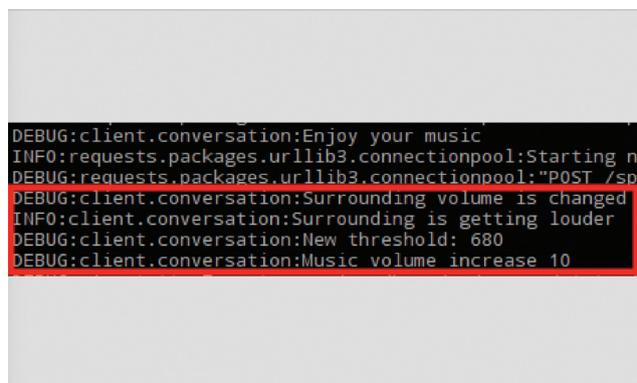


FIGURE 8. The system detected an increased background noise level and responded by increasing the music audio volume

WEATHER INFORMATION

In Figure 9, the audio system successfully invoked the OpenWeatherMap module to search for weather information, in response to the voice command “Tell me the weather.” The speech synthesizer eSpeak then converted the weather information into computer speech. The highlighted box indicates the weather forecast that the system read out using the system voice.

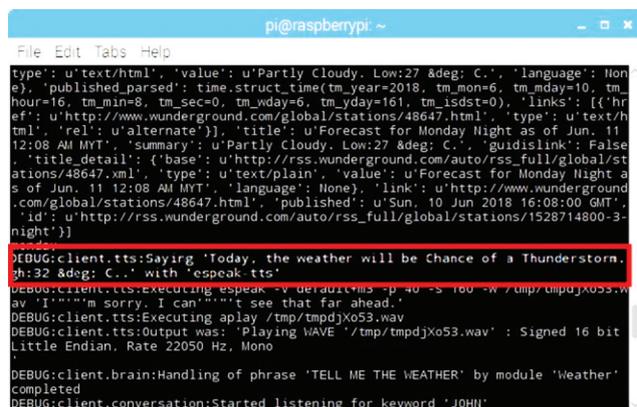


FIGURE 9. Weather information search and read-out by a text-to-speech process

SYSTEM RESPONSIVENESS

It is particularly crucial the user gives voice commands at the right window of time i.e., when the system is not busy processing a (prior) command. If not, the system might be incorrectly perceived as unresponsive. Therefore, a simple circuit of LED indicators shown in Figure 10 was designed and interfaced to the Raspberry Pi. When the system is ready to receive voice commands, it turns on the green LED. When system is busy, the red LED turns on. This signaling serves as a simple visual guide for the user to give voice commands at the right time, hence improving user experience and system responsiveness.

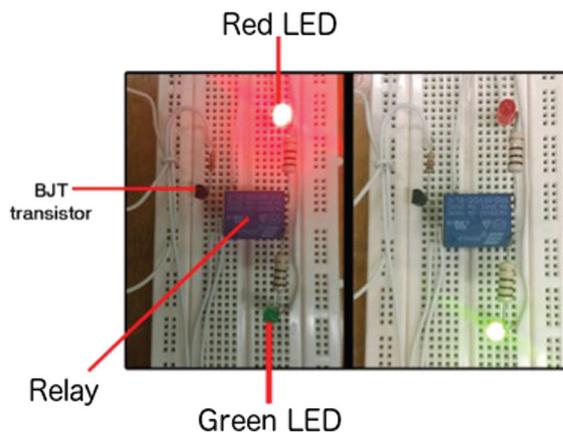


FIGURE 10. LED indicators showing two different system statuses. The red LED indicates the system is busy, and the green LED means the system is ready and waiting for voice commands

An important aspect for an enjoyable user experience is the response time of the audio system. From the user's perspective, the system will be perceived as highly responsive if it is quick to respond to a command i.e., having a short response time. It is during this period of response time that the audio system temporarily disables the microphone (hence not receiving anymore command), uploads the user speech to WitAI, and analyses the returned transcript for specific keywords. Ideally, the response time should be as short as possible so that the user does not have to wait long in between giving successive voice commands.

In Table 3, measurements of response time from 3 identical tests are listed. The response time is measured from the moment a valid voice command interrupt is detected, to the moment the command is executed. In all the measurements, a same user person spoke the same voice command "Play some music" to the audio system. On average, it took about 5 seconds for this prototype audio system to start the music module in response to the voice command.

TABLE 3. Measurements of response time taken by the prototype to respond to a valid voice command

Measurement	Test 1	Test 2	Test 3	Average
Response time, (s)	4.82	5.62	5.57	5.34

For comparison, Amazon describes the response of their market-leading smart speaker, Echo, to voice commands as "instant" (Amazon n. d.). It should be noted that Amazon Echo employs better hardware consisting of a seven-piece microphone array and a high performance microprocessor, and also trains their voice recognition systems using user voice recordings (Wikipedia n.d.). Nevertheless, there have been user reports of response times taking up to 10 seconds (Amazon forum 2017) and 20 seconds (Amazon forum 2018).

This simple test is not meant to be an exhaustive assessment, but rather as a qualitative indication of the prototype's responsiveness. This is because the response time

varies and may be affected by many factors, which include the type of job requested (a command asking for the time could take a shorter response time than playing a music); length and clarity of user speech; surrounding noise levels; bandwidth and speed of connections; and the responsiveness the cloud services.

PROTOTYPE COMPONENT COST

The bill of materials for this prototype is given in Table 4, which shows a total cost of RM304. The major cost constituent is the single-board computer Raspberry Pi 3 Model B+, which costs RM189. For comparison, the smart speaker Amazon Echo (Amazon n. d.) that offers comparable functions retails for USD99.99 (RM410), which is a premium over this prototype. Without compromising any of the pre-defined system specifications and "smart" features, this total cost could be further reduced by using the cheaper single-board computer Raspberry Pi Zero (Raspberry Pi Zero n. d.), which costs only RM40.

TABLE 4. Bill of materials of the prototype

No.	Components	Cost (RM)
1	Raspberry Pi 3 Model B+. Specifications: 1.4 GHz 64-bit quad-core, 1 GB RAM, 2.5/5 GHz WiFi, 2.3 W (idle).	189.00
2	32 GB micro-SD flash-memory card.	60.00
3	USB microphone.	25.00
4	Mini-speaker with 3.5 mm audio jack.	30.00
5	1 K Ω resistors, green LED, red LED, diode, 5V relay, NPN transistor.	N/A
Total cost		304.00

CONCLUSION

This work demonstrates a voice-controlled audio system that can be economically developed based on an open-source platform, affordable single-board-computer, and free cloud services. The developed prototype can be voice-controlled in a natural language, allowing convenience of operations, custom programmable functions, and novel user experience. "Smart" features such as weather notification and automatic volume control have also been successfully implemented in the prototype. To improve user experience, the prototype was interfaced with LED status indicators to guide the user to give voice commands at the right window of time. The prototype can be extended to enable voice-control over other smart-home appliances such as lighting and air-conditioning, without requiring those appliances to be tied to any particular vendor or product-ecosystem. However, the audio system has areas for improvement, chief of which is its response time that should be further reduced so that the user gets an almost instantaneous response after giving a voice-command. Figure 11 shows the SWOT analysis of this prototype as a conclusion.

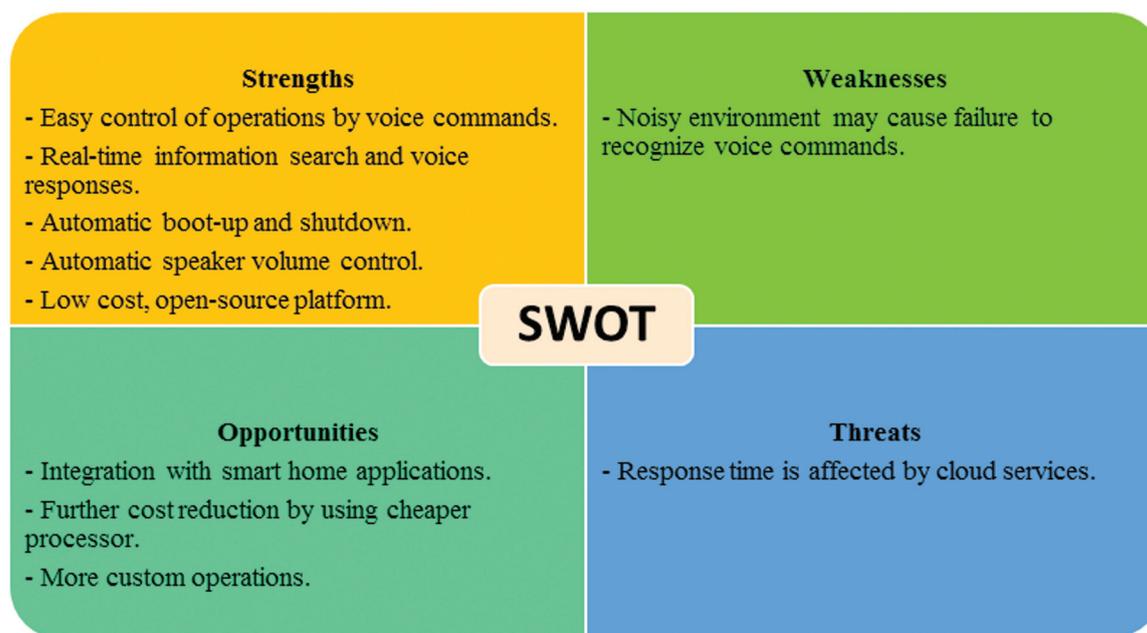


FIGURE 11. SWOT analysis of the completed prototype

REFERENCES

- Abdullah, S. N. H. S., Khalid, M., Yusof, R. & Omar, K. 2007. Pengecaman nombor plat kenderaan menggunakan rangkaian neural dan pengelompokan berbilang aras. *Jurnal Kejuruteraan* 19: 113-126.
- Amazon. <https://www.amazon.com/all-new-amazon-echo-speaker-with-wifi-alexa-dark-charcoal/dp/B06XCM9LJ4>
- Amazon forum. 2017. 10 second delayed response time. <https://www.amazonforum.com/forums/devices/echo-alexa/465884-10-second-delayed-response-time>
- Amazon forum. 2018. Why does Alexa on my Dot delay in responsind to my voice commands? <https://www.amazonforum.com/forums/devices/echo-alexa/473762-why-does-alexa-on-my-dot-delay-in-responding-to-my>
- Anusuya, M. A. & Katti, S. K. 2009. Speech recognition by machine: A review. *International Journal of Computer Science and Information Security* 6(3): 181-205.
- Bengio, Y., Courville, A. & Vincent, P. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8): 1798-1828.
- Borzo, J. 2007. Now you're talking. <http://espeak.sourceforge.net>
- Egmont-Petersen, M., de Ridder, D. & Handels, H. 2002. Image processing with neural networks – A review. *Pattern Recognition* 35(10): 2279-2301.
- Google Assistant. <https://assistant.google.com/>
- Hansen, J. H. & Hasan, T. 2015. Speaker recognition by machines and humans: A tutorial review. *IEEE Signal Processing Magazine* 32(6): 74-99.
- Huang, X., Baker, J. & Reddy, R. 2014. A historical perspective of speech recognition. *Communications of the ACM* 57(1): 94-103.
- Jasper. <https://jasperproject.github.io/>
- Johnson, M. et al. 2014. A systematic review of speech recognition technology in health care. *BMC Medical Informatics and Decision Making* 14(1):94.
- Kabir, K. 2018. Smart speakers – everything you need to know. <https://www.whathifi.com/advice/smart-speakers-everything-you-need-to-know>
- Kim, T. H. 2010. Pattern recognition using artificial neural network: a review. In *International Conference on Information Security and Assurance*, 138-148.
- Krishnan, R. 2018. Bosch develops natural language in-car voice assist system. <https://newatlas.com/bosch-voice-assistant/52817/>
- LeCun, Y., Bengio, Y. & Hinton, G. 2015. Deep learning. *Nature* 521: 436-444.
- Lemley, J., Bazrafkan, S. & Corcoran, P. 2017. Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision. *IEEE Consumer Electronics Magazine* 6(2): 48-56.
- Ling, Z. H., Kang, S. Y., Zen, H., Senior, A., Schuster, M., Qian, X. J. & Deng, L. 2015. Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends. *IEEE Signal Processin Magazine* 32(3): 35-52.
- Lipton, Z. C., Berkowitz, J. & Elkan, C. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

- Marr, B. 2017. The amazing ways Google uses deep learning. <https://www.forbes.com/sites/bernardmarr/>
- Mopidy. <https://www.mopidy.com/>
- Norris, D., McQueen, J. M. & Cutler, A. 2016. Prediction, Bayesian inference and feedback in speech recognition. *Language, Cognition and Neuroscience* 31(1): 4-18.
- Raspbia. <https://www.raspbian.org/>
- Raspberry Pi Zero. <https://www.raspberrypi.org/products/raspberry-pi-zero/>
- Raspberry Pi 3 Model B+ <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>
- Sachdeva, P., & Katchii, S. 2014. A review paper on Raspberry Pi. *International Journal of Current Engineering and Technology* 4(6): 3818-3819.
- Saksamudre, S. K., Shrishrimal, P. P. & Deshmukh, R. R. 2015. A review on different approaches for speech recognition system. *International Journal of Computer Applications* 115(22).
- Sarma, B. D. & Prasanna, S. M. 2018. Acoustic-phonetic analysis for speech recognition: A review. *IETE Technical Review* 35(3): 305-327.
- Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61: 85-117.
- Silva, R. 2018. What is a smart speaker? <https://www.lifewire.com/smart-speaker-4145037>
- Sundermeyer, M., Schlueter, R. & Ney, H. 2012. LSTM Neural Networks for Language Modeling. *INTERSPEECH 13th Annual Conference*, Portland, OR, USA September 9-13.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D. & Rabinovich, A. 2015. Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1-9.
- Tang, J., Sun, D., Liu, S. & Gaudiot, J. L. 2017. Enabling deep learning on iot devices. *Computer* 50(10): 92-96.
- When voice meets sound. 2017. Retrieved from <https://news.harman.com/blog/when-voice-meets-sound>
- Wikipedia. Amazon Echo. https://en.wikipedia.org/wiki/Amazon_Echo
- WitAI. Retrieved from <https://wit.ai/>