Decision Support System for Controlling Home Automation Appliances with Resource Constraints

Agnieszka Betkowska Cavalcante
Gido Labs sp. z o.o.
Poznań, Poland
email: a.b.cavalcante@gidolabs.eu

Monika Grajzer
Gido Labs sp. z o.o.
Poznań, Poland
email: m.grajzer@gidolabs.eu

Michał Raszewski
LARS Andrzej Szymański
Niepruszewo, Poland
email: michal.raszewski@lars.pl

***Abstract*—We introduce a Decision Support System (DSS) for controlling access to embedded home automation devices working fully offline and with limited resources. To provide a hands-free and safe interface to its users, the system combines the following technologies: keyword spotting, speaker verification, automatic speech recognition and understanding, dialog management, and speech synthesis. The proposed system was implemented on an embedded device and tested in home and office environments for the English and Polish languages. In experiments, end-users gave a score of 8.4 out of 10 with respect to a subjective figure-of-merit we have proposed.**

Keywords–Access Control; voice-controlled home automation; Decision Support System; natural language communication; voice biometrics verification.

I. INTRODUCTION

Speech-based interfaces are recently becoming one of the key components of home automation systems. In this kind of setting, they are used to command many home appliances such as thermostats and lights [1]. One of the core elements of those interfaces is Automatic Speech Recognition (ASR) and the related Conversational Agent (CA) that manages the speech-based interactions with a user. The CA is responsible for identifying user commands and intentions based on which changes are made to the parameters of the devices (such as the temperature of thermostats). The entire dialog system can be, thus, considered as a Decision Support System (DSS) that analyses incoming audio signals, detects speech commands, and performs actions based on the collected input data. The ASR and CA modules perform sophisticated, resource-consuming operations, which may use a significant amount of device resources, especially in home automation systems implemented on custom-built embedded devices. This implies that DSSs need to be equipped with an access control module, which activates the core ASR/CA functionalities based on its acoustic-based knowledge [2] only if necessary. The access control module is composed primarily of (i) Voice Activity Detection (VAD), which switches the device to a listening mode after speech is detected; and (ii) a Keyword Spotting (KWS) module, which detects if the particular pass-phrase (keyword) has been spoken [3]. Other methods of access control for home automation devices may also include Speaker Recognition (SR) technology, which is based on voice biometrics and allows only authorized users to access system resources [2][3].

Owing to the complexity of speech processing, many components of contemporary DSS systems of similar functionality are often performed in the cloud. This, however, requires sending and, potentially, storing audio data at external premises, which may raise privacy concerns [4]. Therefore, there is a strong need for solutions working locally, fully offline.

Developing such systems for embedded devices is challenging because of their hardware limitations, especially those related to a CPU speed and a battery consumption. As a result, DSS components should offer their expected functionality with a possibly small resource aquisition, and with the ability to process audio input in real-time. In addition, DSS systems and, in particular, the access control modules, should be characterized by a low False Positive Rate (FPR) to avoid raising false alarms and granting unauthorised access to the core home automation system. Moreover, limiting the number of unnecessary system activations helps to increase the performance of the ASR/CA module.

Current solutions, proposed for DSS system components, are typically cloud-based, or require a significant amount of resources. Customisation is necessary for both, the ASR engine and the CA system, in order to limit the system's scope and, as a consequence, also diminish the required computational power. Additionally, in case of some components, such as the KWS module, even though the system performance reported in the recent research results is very promising, the reported FPR levels are still not sufficient for commercial application, when applied without any supporting intelligence [3][5]–[7]. Moreover, the commercially available ASR, AC and KWS systems are mostly aimed at recognising English speech/speakers [5]–[7]. Implementing them for a less common language (such as Polish), typically requires gathering a training database of audio samples, which is both time-consuming and financially expensive. Hence, providing a well-performing system with only a small-size database is very challenging [3].

To address the above challenges, we propose a DSS that would work on an embedded device and allow to efficiently make decisions on granting access to the voice-controlled home automation system and on user intentions, regarding the device set-up. The custom-designed decision-making engine exploits the knowledge extracted from the audio signals – collected by the device microphone with the support of KWS and SR modules for the access control system and ASR together with CA for further information processing and retrieval. In this paper, we present the research results related to design and implementation of all necessary components of such a DSS system, which was targeting the small-scale embedded device based on the Yocto Linux distribution [8] and the STM core hardware platform [9]. The proposed solution has been tested with a number of end-users in real-world scenarios. The research results presented in this paper constitute a follow-up to our previous work on the access control system design which was presented at the eKNOW2020 [3]. In this paper,

 ISBN: 978-1-61208-986-7

we focus on the performance results obtained during the final field trials and describe the DSS system as a whole, focusing on the performance of making the final decision about the identified user intent and the executed home automation device functionality.

The rest of the paper is organised as follows: in Section II, we present a short overview of related work, followed by the description of the proposed DSS system and the methodology taken to implement its components for embedded devices in Section III. In Section IV, we discuss the test environment and the obtained test results and follow with conclusion in Section V.

II. Related Work

Voice-controlled home automation systems are equipped with voice interfaces that identify user's commands and provide an access to the core system functionality [10]–[12]. They can use a dialog system that exploits elements of Natural Language Processing (NLP) techniques to extract user intent from potentially incomplete input information [13][14]. Although technical solutions exist, which implement the necessary components of a DSS system for the voice-based interfaces, their usage is often envisioned for the devices with higher computational power (such as, e.g., mobile phones). Hence, they use complex NLP techniques, which cannot be handled by a small, resource-constrained devices that we were targeting [14]. In particular, most of the best-performing ASR and AC components are based on the complex neural networks [12][14][15]. In addition, the top-performing solutions available today, including the ones that are capable of controlling home automation devices, such as, e.g., Amazon Alexa [16], Apple's Siri [17] and Google Home [18], are available only in the cloud-based set-up. This way, resource-consuming techniques are implemented in the cloud and made available to mobile devices via API calls over Internet connection. However, for systems targeting strictly offline environments, similar to the one presented in this paper, such solutions are inaccessable and must be replaced by small-size, robust, custom-made local implementation. Moreover, the popular cloud-based solutions [16]–[18], offer full NLP-based functionality only to a set of languages, typically limited to the most common ones.

Referring to the Access Control system, the most straightforward solution is using VAD/KWS module [6]. The KWS system, which has been trained on silence or background noise samples, can also take on a role of a VAD solution. The recent research on implementing KWS systems focuses on applying neural networks, particularly Residual Neural Networks architectures (ResNet) [6][19]. ResNets are characterised by a lower complexity and faster training phase and were proved to obtain very good performance even for relatively small-sized networks – reaching the accuracy of 95% [6]. While such results are impressive, they are far from being industrially applicable: assuming that FPR is 2% [6] and the system makes a prediction every second, there will be 72 false alarms in one hour — a number inacceptable from the point of view of an end-user [3]. However, since their complexity is not that high as for the NNs used in, e.g., the ASR systems, they are a good candidate technology for the target devices investigated in this paper.

Similarly, the recent approaches to the Speaker Recognition also use lightweight neural networks, which offer good performance with limited resource requirements [19]. These are also promising candidate solutions to support a biometric-based DSS, which can identify speakers and grant them proper permissions [20]. However, additional actions should be usually taken to improve FPR of such systems in practical set-ups. Hence, a challenge in the design of a targeted DSS still remains in improving the FPR of a combined KWS and SR technologies for the Access Control module. Some approaches to address this problem introduce pushing a button [11], detecting the audio louder then a certain threshold [10] or rely on more advanced features – such as, e.g., in [12], where KWS is followed by additional reasoning [3]. The latter kind of solutions are often very complex and, likely, too resource-consuming for small embedded devices.

Although computational complexity of possible approaches to construct the DSS system for the voice-controlled devices remains a challenge, the access control approaches and their FPR in practical set-ups also calls for new solutions, while many related works on voice interfaces neglect this aspect, regardless of the fact that it has significant impact on the practical implementation of such solutions [21][22]. This calls for new approaches that would offer required effectiveness on embedded devices.

III. System Architecture

We have designed a DSS system to control devices, such as air conditioners, thermostats, and heating furnace, among others. To increase the security of the system, it works locally and with access to neither the Internet nor cloud-based resources. All communication takes place via speech interfaces where users' utterances are decoded into technical commands and where users are verified by a biometric voice system. The proposed system architecture is depicted in Figure 1.

The DSS system operates on embedded devices, such as intercoms in the continuous listening mode. It consists of many modules, including keyword spotting, speaker recognition, automatic speech recognition, dialog system / conversational agent, and speech synthesis (Text To Speech – TTS). The decision taken by the DSS is sent to a middleware and later to a backend, which executes the commands on a desired device.

The access control DSS constantly analyses the signal from the microphone and searches for a specific keyword with a access control system that has been proposed in [3]. Once that keyword is spotted, the core ASR-based dialog system is activated. The command spoken by the user (e.g., "set the temperature in the living room to 5 degrees") is converted to text by the ASR system. The transcribed utterance is processed by the CA, which tries to understand the user's intent ("set the temperature") and assesses whether the input contains enough information related to that intent. If so, the DSS decides the type of command, its parameters, and its recipient (one of the devices of the backend). The constructed technical command is sent to backend via a dedicated middleware. If the provided user's input is not sufficient to define such command, the Dialog System will continue the conversation and ask the user for the missing information.

In parallel, the DSS authorizes a user with the SR module and the stored voice biometric patterns. The system executes commands only if the user voice is successfully recognized. The speaker recognition processing can take a few seconds because of the limited computation capabilities of the targeted devices. Therefore, this step is performed in the background

 ISBN: 978-1-61208-986-7

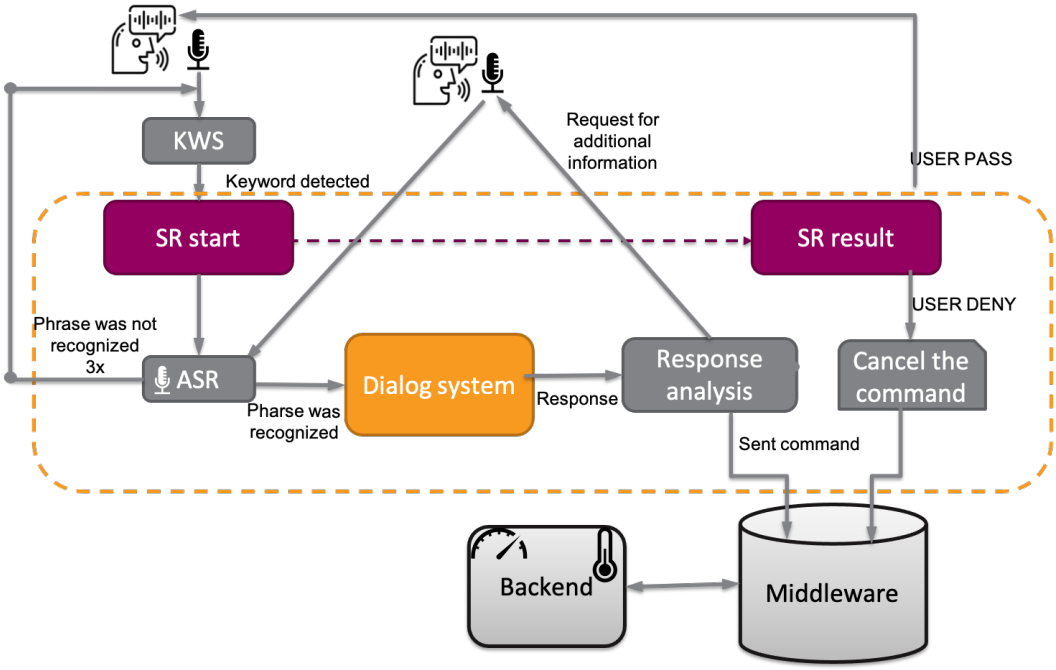

Figure 1. The Access Control DSS System design.

while the user speaks with the dialog system. By doing so, the user has the impression that the DSS operates in real-time.

These decision-making subsystems form a DSS, where at each step decisions are made based on the knowledge collected on the previous steps. A detailed description of each element is presented in the following subsections.

A. *Access control module*

The KWS system – a core component of the access control module – is based on a neural network with the ResNet architecture "res8" [6], and it contains a relatively small number of network parameters (approx. 110K). More specifically, we have based our solution on the system described in [6], which was trained on the Google Speech Commands Dataset (GSCD) – we have then performed a transfer learning procedure and used solution [6] as a feature extractor. For this purpose, a dataset of 698 positive examples of the selected keyword from 36 people has been collected. Together with a larger database of phonetically representative Polish speech (taken as negative examples), it was used for training the KWS classifier layer with 2 outputs (keyword vs. non-keyword). In addition, for the training phase 80% of examples were augmented with background noises of variable loudness, originating from the GSCD dataset. These background noises were also used to create negative examples of silence (i.e., the lack of speech) [3].

The designed system works in real-time, with a sliding window of $1s$ that moves by $0.2s$ [3]. Those parameters were established experimentally, as a compromise between satisfactory spotting of utterances and computational overhead. 40 Mel-Frequency Cepstral Coefficients acoustic features are used, normalised, and fed into the ResNet, which acts as a binary classifier returning a numerical score. Typically, if the score is higher than a certain threshold, it is assumed that the keyword was uttered. In our case, a threshold corresponding to the Equal Error Rate (EER) was taken [3].

The access control module also consists of many other submodules that are described in [3]. The first one – "Loudness Checker" – requires the incoming audio to have a certain minimum level of loudness. The second one, called "Timer", limits the time duration of an audio buffer that is being processed to $1.2s$ – if the triggering word is not observed shortly after that, the audio can be considered as background noise, and further processing of that signal can be skipped. Such a design also allows to lower the resource consumption. The third submodule – "Score Smoothing" – smooths the results obtained from KWS by calculating the mean of the last n predictions. The usage of these submodules allowed to reduce significantly the FPR of the Access Control system [3].

The proposed solution was primary implemented and tested on a Raspberry Pi 3B (CPU: 1,2 GHz quad-core ARM-8 Cortex-A53 (64-bit); 1 GB RAM). The device was equipped with a custom-made microphone matrix with 5 independent microphones. The experiments were conducted as field trials, where real hardware was used. 13 users were asked to utter the correct keyword 30 times. In addition, they spoke two words very similar to the keyword and eight other words three times each. The access control DSS evaluated each sample and the results were averaged. The proposed module, with the Score Smoothing submodule, using the last three predictions achieved an accuracy of 90.77% with TPR of 86.41% and FPR of 4.87% [3].

B. *Speaker recognition module*

The input to the SR module is strongly correlated with the KWS scores. Only a single frame processed by the KWS module, with the strongest trigger, is used to perform user authorisation. This process, hence, uses the utterance containing the system keyword. During authorization, the collected voice sample is compared to the user voice footprints (speaker embeddings), which store biometric information representative

 ISBN: 978-1-61208-986-7

for the person's voice. These were computed earlier based on sample recordings, being the repetition of the custom keyword (10 repetitions have been collected, approx. 1s each). As a result, the created SR system can be considered as text-dependent, which allowed to increase its performance.

The SR system is based on ResNet neural network similar to the one presented in [19]. The related model was further trained by us with a low learning rate using a dataset with voices of 100 polish speakers in a transfer learning procedure that improved the performance for the Polish language. The accuracy of the trained SR module was tested on the database of 36 polish speakers (III-A), which contained the samples of a custom keyword and several sentences recorded by each speaker. The new model, together with the proposed new system design using a text-dependent approach, allowed to improve the performance (expressed as an EER) for the identification of a single speaker from 9.83% to 1.7%. The reference value of 9.83% has been obtained for the basic model from [19] in the test performed on the database of 100 polish speakers, by using random enrollment samples and random test samples of length similar to the ones used in the experiments on the database of 36 speakers (evaluating the new system design). For the experiment comparing the same SR system set-ups (i.e., enrollment on the recordings containing 5 repetitions of a custom keyword and test on the single recording containing this keyword), the newly created model with the EER of 1.7% outperformed the basic model, which resulted in the EER of 2.27%. The SR solution is able to recognize both – English and Polish speakers.

C. Automatic speech recognition

The choice of a suitable speech recognition platform for the communication between the DDS system and users (speech to text) is influenced by two requirements: (1) the need to operate with limited resources; and (2) the need to train acoustic models with a relatively small amount of training data. Taking that into account, a CMU open source vocabulary, speaker-independent continuous speech recognition engine [23] for embedded devices was chosen.

For English, an acoustic model and phonetic dictionary provided with pocketsphinx library [23] were selected. The grammar in Java Speech Grammar Format (JSGF) were custom-prepared to enable voice control of home automation systems.

For the Polish language, a dedicated acoustic model was trained with recordings collected with the mPass platform [24]. Each of 100 people, 49 males and 51 females, recorded the following: (i) a phonetically balanced text consisting of 322 sentences, and (ii) approximately 300 commands for controlling home devices. Each phoneme was represented by a three-state HMM model with eight mixtures per state – the number of mixtures were selected experimentally based on the size of the training material. Mel Frequency Cepstral Coefficients (MFCC) with delta-delta were chosen as a signal representation. A phonetic dictionary with 234 words and the JSGF Grammar were prepared in a similar manner for the English language.

The evaluation of the Polish speech recognition system was done with the cross-validation method. The data of one user were removed from the training set, and the set of commands for that user was recognised by the ASR. The results were averaged, and the system achieved the accuracy of 96.2%.

In addition, the experiments were conducted as field trials with a Raspberry Pi platform (with the configuration described in section III-A). In office and home environments, 10 Polish speakers spoke 41 sentences/commands in Polish to a ASR installed on an embedded device. Four native English speakers, five native Polish speakers, and one Brazilian did the same in English. The results were averaged. The word accuracy of the system was 97.98% for Polish and 94.77% for English.

D. Dialog system / conversational agent

The proposed dialog system uses openDial [25] – a lightweight, Java based, domain-independent toolkit that was originally designed to perform dialog management tasks. It combines logical and statistical approaches to dialogue modeling. More specifically, it consists of many modules with a shared memory represented by a Bayesian network. The domain models are specified via probabilistic rules encoded in XML.

We have created a natural language understanding model, which recognizes the user's intended actions from the text provided by the ASR system (such as 'increase the temperature', 'set the time', 'decrease the temperature', 'turn on device' ,'get the date', 'get battery level', among others). In addition, the model needs to update slots (required parameters) to the chosen action. For instance, in case of 'set the holiday mode', the required information consists of: (i) the number of days and hours, (ii) the temperature value, and (iii) the room identifier. If any of the slots are not filled, the system asks the user to provide the missing pieces of information.

In addition, the dialog management module was designed to make decisions regarding dialog states and flow – such as asking user to repeat the sentence, ask the user about missing information, finish the dialog, among others.

The final module of the dialog system is the natural language generation module, which specifies the mapping between the current dialog state and the system response. The output has a twofold form: (i) the text addressed to the user via speech synthesis, and (ii) a technical command for the device.

For instance, a dialog could have the following form:

- **user**: Increase the temperature in the living room
- **system**: By how many degrees?
- **user**: By 3 degrees.
- **system**: Temperature in the living room has been increased by 3 degrees.

E. Middleware

The software implementation on an embedded device contains also the middleware, which is responsible for handling communication between the voice-controlled DSS and the hardware of the embedded device (i.e., sensors/actuators). This way, the commands identified by the logic of the DSS system can be aquired and stored in a dedicated database and communicated to the actuators with a proper timing and order. The RabbitMQ broker is used for handling communication between the DSS system and a middleware. After middleware receives a RabbitMQ command, it sends an appropriate request on the internal REST (HTTPS) server. The REST server then converts that request to a message in format understandable by the Aura module on the hardware-related backend and

 ISBN: 978-1-61208-986-7

then sends it to the selected actuator module via a serial port. The Aura module itself is an STM32 microcontrolller with a 868MHz transceiver (Spirit1), it communicates with the main board using UART. The microcontroller's firmware allows it to communicate with other home automation devices wirelessly through the Aura Radio Protocol, which may be usefull for more complex set-ups. The Aura protocol is a proprietary solution developed specifically for the Auraton Smart devices.

F. The voice control hardware

Figure 2. Voice control embedded device – exterior view.

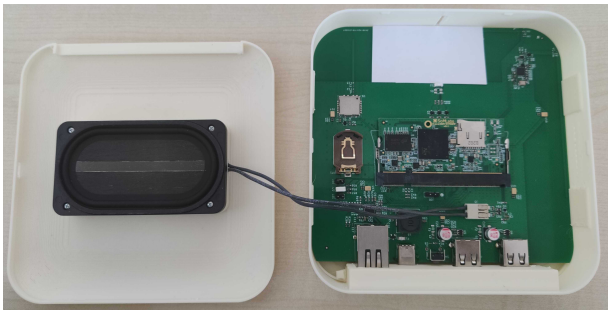

Figure 3. Voice control hardware platform – interior.

The voice control application worked on an STM32MP157 microprocessor, which is based on an ARM-A7 architecture, running two cores at 650 MHz. The developed platform is equipped with 1GB RAM, a Wi-Fi module, Ethernet, and a 868 MHz radio to communicate with peripheral devices. The application runs on an Embedded Linux based on OpenSTLinux and customised with the Yocto framework. Voice is recorded and streamed using a 5 digital MEMS microphones placed on the PCB in a semicircle. The user gets feedback after each command via a mono loudspeaker. Audio samples from each microphone are formed into one stream using the ALSA framework, and they are passed to the DSS module that runs in a Docker container. The identified user command (from the DSS output) is send to a message broker(RabbitMQ). Next, the message is received by a middleware, and it is passed to the REST API server, which saves each request in a dedicated SQLite database.

In addition, the device can also be controlled with a mobile application. It communicates with the device using the same RabbitMQ server, which is also responsible for sending data to the RF 868 MHz module and further on to the peripheral devices. An overview of the embedded device is presented in Figure 2 and Figure 3.

IV. EVALUATION

The evaluation of the proposed DSS system was performed in real-life scenarios. The testers had to perform 27 assignments using the targeted device. More specifically, each of the users has been given a task to set a desired configuration to the chosen home automation system or to collect data from it. The exact information regarding, for example, the choice of the rooms, the temperature, the dates, and the time was not provided, so that the creation of a final command was left to the user. All testers were only informed that the system understands voice communication and what are its general capabilities, but no further details were provided.

The accuracy of the whole DSS was evaluated based on the number of positively completed tasks. In particular, the following items were taken into account:

1) the number of successfully completed tasks in the first attempt (i.e., the user's command was understood correctly and voice biometric verification was positive)
2) the number of successfully completed tasks in the second or third attempt (for instance, in the first or second attempt, the DDS did not understand the user and asked to repeat the command, the user was understood wrongly and canceled the execution of the command, or the user did not pass the voice authorization).
3) the number of unsuccessful task executions in three attempts.

The results are shown in Table I. In this test case, 8 users, male and female, were providing commands to the system in Polish. The average accuracy of task completion in the first attempt was 82.2% – that includes keyword spotting, the successful understanding of the dialog with the user, and correct user's voice verification. The percentage of correctly performed tasks in at most three attempts increased to 97.1%.

TABLE I. TASK COMPLETION ACCURACY FOR ACCESS CONTROL DEVICE – POLISH NATIVE SPEAKERS

User	Acc. of successfully completed tasks		Acc. of [%]	
	1st attempt	*2nd/3rd attempt*	*SR verification*	*failure*
user1	77.8%	14.8%	100%	7.4%
user2	88.9%	7.4%	90.5%	3.7%
user3	70.4%	29.6%	100%	0.0%
user4	85.2%	14.8%	81.8%	0.0%
user5	85.2%	14.8%	68.2%	0.0%
user6	96.3%	3.7%	100%	0.0%
user7	74.1%	22.2%	95.2%	3.7%
user8	80.0%	12.0%	84.2%	8.0%

In addition, a survey of the level of user satisfaction has been performed. The user was evaluating intuitiveness of the system and its subjective effectiveness. The results are shown in Table II. On average, the testers evaluated the intuitiveness as 8.8, and the effectiveness as 8.4 out of 10. Similar experiments were carried out for the second group of users with the commands spoken in English. Due to the situation with COVID-19, we had only six testers and none of them was a native speaker. The average accuracy of task completion in a first attempt was 78.7%, and the performance of 94.3% was obtained in at most three attempts. On average, the testers evaluated intuitiveness of a DSS system as 8.5, and the effectiveness as 7.8 out of 10.

TABLE II. SURVEY OF THE LEVEL OF USER SATISFACTION WITH ACCESS CONTROL DSS

user	Effectiveness	Intuitiveness
user1	9.5	9.0
user2	8.0	9.0
user3	8.0	10
user4	9.0	9.0
user5	7.0	6.0
user6	10	10
user7	9.0	8.0
user8	7.0	9.0

There were mainly three types of errors that lowered the prototype performance:

- the testers were using grammatically incorrect commands (as it happens in colloquial speech), or they were making mistakes and correcting themselves
- the sequence of words spoken by the testers was not predicted and provided in the dialog system
- the ASR system had problems with correctly recognizing numbers when they were not spoken clearly. This is due to the fact that numbers are recognized mostly based on acoustic models. For instance, in the sentence 'set the temperature to X degrees', the grammar does not provide additional information and occurrence of any number has the same probability.
- the command was understood correctly by the DSS, but the user verification was not successful

In addition, if the system did not understand the command in the beginning of the test, the users frequently got frustrated, and it adversely affected further performance of the DSS. If the error occurred in the middle of the test, such effect was not observed. To counteract this phenomenon, problems related to user experience should be further studied.

The above-mentioned errors cannot be completely removed, if a voice command format is not imposed. However, the ASR and dialog system can be adapted to a specific user. When using a system of this type, the user also tends to learn how to speak to increase the probability of being understood. As a result, the overall system performance should increase with time.

Although one cannot expect flawless operation of any voice interface, we have shown that with a tailored design it can achieve performance levels, which are sufficient to properly control the home automation device. Moreover, it could be very useful when combined with other multi-modal interfaces.

V. CONCLUSION AND FUTURE WORK

We have described the DSS system for voice-controlled home automation devices running on embedded platforms with limited resources. The system grants access to the home automation devices and enables to collect information necessary for their control. The proposed solution combines many technologies, such as keyword spotting, speaker verification, automatic speech recognition and understanding, dialog management, and speech synthesis to provide a safe and natural interaction with the user.

The voice control prototype device that was presented in this article, was tested in real scenarios of home and office environments. The testers freely used natural language to convey their commands. The experiments proved the usefulness of this solution.

As a part of future work, we plan to focus on improving the user experience. This can be done by developing more flexible dialog schemes, and by combining the existing voice-based DSS system with multi-modal interfaces to further reduce the probability of a task completion failure.

ACKNOWLEDGMENT

The research was supported by the National Centre for Research and Development in Poland under the grant no. POIR.01.01.01-00-0044/17.

REFERENCES

[1] C. J. Baby, F. Khan, and J. N. Swathi, "Home automation using web application and speech recognition," in 2017 International conference on Microelectronic Devices, Circuits and Systems (ICMDCS), 2017, pp. 1–6.

[2] A. Kaklauskas, Intelligent Decision Support Systems. Springer International Publishing, Cham, 2015, ch. 2, pp. 31–85, in Biometric and Intelligent Decision Making Support, ISBN: 978-3-319-13659-2.

[3] M. Pabiszkak, M. Grajzer, and L. Sawicki, "Access control method for the offline home automation system," in eKNOW:2020: The Twelfth International Conference on Information, Process, and Knowledge Management. IARIA, 2020, pp. 97–101.

[4] A. Cuthbertson. Google defends listening to private conversations on google home: But what intimate moments are recorded? [Online]. Available: https://www.independent.co.uk/life-style/gadgets-and-tech/news/google-home-recordings-listen-privacy-amazon-alexa-hack-a9002096.html [retrieved: 05, 2022]

[5] T. N. Sainath and C. Parada, "Convolutional Neural Networks for Small-footprint Keyword Spotting," Proc. 16th Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH 2015, 2015, pp. 1478–1482.

[6] R. Tang and J. Lin, "Deep residual learning for small-footprint keyword spotting," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 5484–5488.

[7] C. Lengerich and A. Hannun, "An end-to-end architecture for keyword spotting and voice activity detection," arXiv preprint arXiv:1611.09405, 2016.

[8] Yocto Project. Getting started: The yocto project® overview. [Online]. Available: https://www.yoctoproject.org/software-overview [retrieved: 05, 2022]

[9] STMicroelectronics. Secure Hardware Platforms. [Online]. Available: https://www.st.com/en/secure-mcus/secure-hardware-platforms.html [retrieved: 05, 2022]

[10] C. Sidhartha, S. Siddharth, S. S. Narayanan, and J. H. Prasath, "Voice activated home automation system," in National Conference on Man Machine Interaction 2014, vol. 1. ASDF, India, 2014, pp. 69–72.

[11] K. A. Lee, A. Larcher, B. Thai, B. Ma, and H. Li, "Joint application of speech and speaker recognition for automation and security in smart home." in INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, 01 2011, pp. 3317–3318.

[12] Siri Team. Hey siri: An on-device dnn-powered voice trigger for apple's personal assistant. [Online]. Available: https://machinelearning.apple.com/research/hey-siri [retrieved: 05, 2022]

[13] R. Justo et al., "Improving dialogue systems in a home automation environment." in AMBI-SYS, 02 2008, p. 2.

[14] A. M. Bashir, A. Hassan, B. Rosman, D. Duma, and M. Ahmed, "Implementation of a neural natural language understanding component for arabic dialogue systems," Procedia Computer Science, vol. 142, 2018, pp. 222–229, arabic Computational Linguistics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918321835

[15] A. S. Tulshan and S. N. Dhage, "Survey on virtual assistant: Google assistant, siri, cortana, alexa," in International symposium on signal processing and intelligent recognition systems. Springer, 2018, pp. 190–201.

 ISBN: 978-1-61208-986-7

[16] Amazon Alexa. [Online]. Available: https://developer.amazon.com/en-US/alexa [retrieved: 05, 2022]

[17] Apple. Control your home with Siri. [Online]. Available: https://support.apple.com/en-us/HT208280 [retrieved: 05, 2022]

[18] Google Home. [Online]. Available: https://home.google.com/welcome/ [retrieved: 05, 2022]

[19] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 5791–5795.

[20] M. Vacher et al., "Speech and speaker recognition for home automation: Preliminary results," in 2015 International Conference on Speech Technology and Human-Computer Dialogue (SpeD). IEEE, 2015, pp. 1–10.

[21] Y. Mittal, P. Toshniwal, S. Sharma, and D. Singhal, "A voice-controlled multi-functional smart home automation system," in 2015 Annual IEEE India Conference (INDICON), 10 2015, pp. 1–6.

[22] G. López, V. Peláez, R. González, and V. Lobato, "Voice control in smart homes using distant microphones: A voicexml-based approach," in Ambient Intelligence, D. V. Keyson et al., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 172–181.

[23] D. Huggins-Daines et al., "Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices," in 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, vol. 1. IEEE, 2006, pp. I–I.

[24] A. Cavalcante and M. Grajzer, "Proof-of-concept evaluation of the mobile and personal speech assistant for the recognition of disordered speech," International Journal on Advances in Intelligent Systems, vol. 9, 12 2016, p. 589.

[25] P. Lison and C. Kennington, "Opendial: A toolkit for developing spoken dialogue systems with probabilistic rules," in Proceedings of ACL-2016 system demonstrations, 2016, pp. 67–72.

 ISBN: 978-1-61208-986-7