논문 22-47-08-16 The Journal of Korean Institute of Communications and Information Sciences '22-08 Vol.47 No.08
https://doi.org/10.7840/kics.2022.47.8.1185

Implementation of a Reliable VUI System on Edge Device

Jeongin Kim◆, Paul Angelo Oroceo*, Wansu Lim°

ABSTRACT

Voice user interface (VUI) is becoming highlighted as an effective tool for customized user interactions. In this paper, we implement a reliable VUI system using speech-to-text technology. The user's utterance is converted into text using Google STT API and followed by keyword determination. Then, the system provides appropriate response corresponding to this keyword query to the user. We use four response types according to voice characteristics (male vs. female) and amount of information (reference vs. short). Our proposed VUI shows a real-time performance on the edge device and has been evaluated on two separate experiments.

Key Words : Voice-user Interface, speech recognition, speech-to-text, embedded system, keyword parsing

Ⅰ. Introduction

Voice user interface (VUI) is an interface method that allows a user to interact and control a human-machine interaction through voice commands based on speech recognition technology[1]. Existing research on VUI was carried out when a phone call while driving or giving commands through a speech recognition engine from a smartphone is required[2]. The ultimate goal of VUI is to recognize natural human voice and understand the speech and speech and give a response back to the user. For this purpose VUI uses automatic speech recognition (ASR) technology to convert user speech into a machine language or transcript[3]. In a technical aspect, ASR is the process of converting a speech signal into a sequence of words or sentences for text-based communication purpose and device controlling.

ASR and speech-to-text (STT) have been developed to improve our daily life like personal voice assistants and deeply integrated into many business chains[4]. [5] utilized ASR with Fourier transform, which creates features from the sound file and applies the filtering and aggregation method via windowing. They apply an acoustic model to match the phonemes and a language model that uses probability distribution to predict words from the phonemes. In [6], they utilized deep learning model for improvement in ASR tasks. [6] have applied convolutional neural networks and recurrent neural networks while recently transformer networks have achieved high accuracy. Using this ASR technology, system transcribe the voice into a full sentence to perform keyword parsing.

The prevalent method for detecting keywords is to utilize an acoustic filler model[7]. This acoustic filler model is based on a garbage model which trains the words and sorts out the non-speech or task-irrelevant words using the maximum likelihood method. For instance, they train on noise and non-keywords which are not relevant for the task with a filer model. Then, they compare the likelihood that the word series will match the

※ This work was supported by the National Research Foundation of Korea (2020R1A4A101777511)
◆ First Author : Kumoh National Institute of Technology, Department of Electronic Engineering, 학생회원
° Corresponding Author : Kumoh National Institute of Technology, Department of Aeronautics, Mechanical and Electronic Convergence Engineering, wansu.lim@kumoh.ac.kr, 정회원
* Kumoh National Institute of Technology, Department of Aeronautics, Mechanical and Electronic Convergence Engineering
논문번호 : 202110-294-D-RN, Received October 22, 2021; Revised March 19, 2022; Accepted May 11, 2022

keyword model with the likelihood that the word series will match the filler model, maximizing the contrast between the two models. This process allows the system to spot the main keyword and understand user's command.

In this paper, we propose user-oriented VUI system by preparing diversified voice database that gives variation in gender of speaker and amount of information. In addition, a real-time VUI operation is performed on the edge device (NVIDIA Jetson TX2[8]), which has advanced computing speed, credibility, and accuracy for audio processing. Our proposed system considers memory size to fully run in embedded system and process the audio data in a real-time. VUI extracts voice as input which is processed by speech recognition method and generates human-like response. This audio processing results in conversion from human speech to text.

With regard to the keyword detection, the system parses the transcribed sentences considering as text without training. Then, the system searches for the partial matching points in the sentence by comparing every word with the preselected keyword list. This process determines the main keyword among the incoming word series. Once a keyword is detected on the device, the audio corresponding to the keyword (e.g., "weather") is streamed from the database. Our database consists of the speech data spoken by male and female speakers was recorded and stored for the response. In this paper, we describe a VUI system that fully runs on the edge device by uploading the database locally on Jetson TX2. We uploaded the complete database on the local edge device so that the system is able to retrieve the answers from the local database without the Internet. This can improve the speed of user interaction in comparison with using the Internet-based database. To verify the proposed VUI system, we evaluated the performance parameter as word error rate (WER[9]) with generic numbers, words, sentences and 240 question scripts.

Ⅱ. Implementation of VUI System

2.1. Process of VUI system

Fig. 1 is the overall process of the proposed VUI system in the following steps: 1) STT, 2) keyword parsing, 3) answering.

Step 1 (STT): Step 1 starts with receiving the user's voice input. When the voice enters the system through the microphone, the STT system prepares to

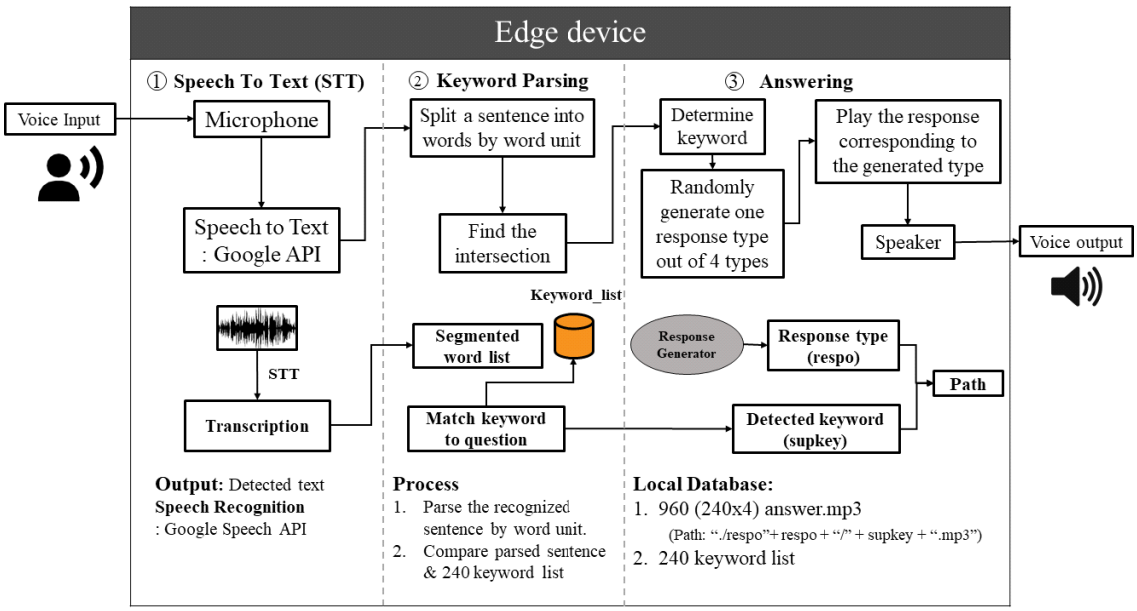

Fig. 1. Overall process of proposed VUI system.

convert this speech into text. Among the various methods of STT, our proposed model uses Google API in python. Once the system receives input voice from a microphone source, it puts an audio file as an argument and listens to the voice by initializing the recognizer class for speech recognition by setting the API and language. The parameters used in the function call were audio='microphone' and the default language_code='ko' In our proposed model, it starts to recognize the audio object using Google web speech API and Korean language.

Step 2 (Keyword parsing): This step splits the transcribed question sentence into word units. We use 'split' function which automatically divides the string by recognizing spaces. This enables to separate sentences by word unit and save it as a new list. The predetermined 240 keywords are saved as a txt file in the local database, and the open function returns this file object. Then, the parsed sentence is compared with all 240 keyword list to find the intersection.

Fig. 2 is an example performance of keyword parsing for actual user's utterance input. When user asks "Can you recommend me a good food for diabetes?" to our VUI system, the system performs STT process and transcribe the user's speech in a full sentence. Then, the system separates this full question sentence into word units and compares them with the entire keyword list from the given local database. Thus, the system finds the intersection between parsed words and keyword list which brings the determined keyword. This intersection indicates the keyword which is used for the final input for step 3.

Step 3 (Answering): In our local database, there are four types of answers, which were given variation in voice gender, and amount of information as follows: MR (man, reference), MS (man, short), WR (woman, reference), and WS (woman, short). The reference type allows the VUI to provide abundant information to the user so that they can interact for a longer period of time. On the other hand, the short type is providing the user with only the necessary information by delivering as simple as possible. Table. 1. shows an example of response according to 4 different response types.

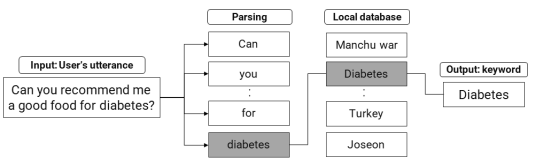

Fig. 2. Example performance of keyword parsing.

In order to derive an appropriate response from the database, we need two input parameters: keyword and response type. After the above detailed steps, the determined keyword was stored as a parameter and linked with the response type parameter randomly selected by the response generator. Then we combine the keyword with the response type together in the path. Path is an important variable to find the location of the matching audio file in our local database. This answer is streamed from the output speaker device to the user.

Furthermore, we assigned auxiliary condition setup for our VUI system in step 3. In Algorithm 1, from line 1 to line 7 explains the variable used in this pseudo code. The main function starts from line 8 represents the overall process including finding keyword by searching intersection and counting each response type until it becomes 15 times each. The counting methods used nested loop. Each response type is streamed 15 times equally during one experiment. The response is randomly generated in

Table. 1. Example response by 4 types.

keyword / type	South Korea
MR	According to the Ministry of Land, Infrastructure and Transport, the area of South Korea is 1004128.518 hectares. (Male voice)
MS	It is 1004128.518 hectares. (Male voice)
WR	According to the Ministry of Land, Infrastructure and Transport, the area of South Korea is 1004128.518 hectares. (Female voice)
WS	It is 1004128.518 hectares. (Female voice)

Algorithm 1 Counting and limitating the number of response type

```
1  S_i: every word existing in full command
2  C_i: keyword list in the local database
3  A_i: Response type array
4  len: The number of specific response types
5  in the 'rem' array
6  rem: The result of counting for each
7  response type
8  function response()
9  begin
10 rem = [ ]:
11 Initialize the counting result of response
12 type to an empty state
13 respo = random.choice(A_i)
14 supkey := set(S_i) ∩ set(C_i)
15 for supkey in C_i do
16     for x in A_i do
17         if respo == x then
18             len = rem.count(respo)
19             if len != 15 & len < 15 then
20                 remain.append(y)
21                 path = 'respo/' + respo +
22                   '/' + supkey[0] + '.mp3'
23                 playsound(path)
24             elif len == 15 then
25                 A_i.remove(y)
26 end
```

every iteration based on the remaining response array as shown in line 13. The initial state of counter list which represents the number of each remaining type exists empty. Line 20 indicates appending the corresponding type in the 'rem' array when the value of 'len' satisfies the condition of 15 or less. On the other hand, line 25 shows that when the number of specific types reaches 15. The corresponding response type is removed from the response type array so that selection of the type is no longer possible. Thus, the system is keep checking the number of remaining response type and updates the response array in every iteration.

Ⅲ. Performance Evaluation

3.1 Environmental setting

Hardware configuration: In Fig. 3, we visualized the actual prototype for our VUI system by presenting three hardware components: 1) an embedded computer 2) audio input device 3) audio output device. For the embedded computer, we employed NVIDIA Jetson TX2 with Ubuntu 18.04.6 LTS equipped with an ARMv8 (64-bit) Multi-Processor CPU. Jetson TX2 contains audio

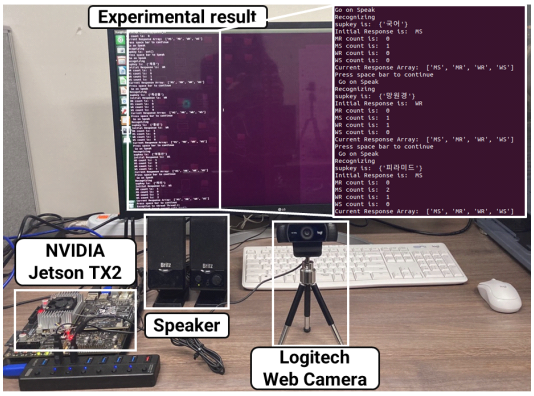

Fig. 3. Experimental setup.

processing engine which enables full hardware support for multi-channel audio over multiple interfaces. For the voice input, the microphone device is Logitech webcam C922 which enables recording visual and auditory modalities. Lastly, for the audio output, we used Britz coupe speaker.

Software configuration: For software configuration, all pre-requisite dependencies has been installed in edge device. On the edge device, essential python libraries were installed to access microphone, keyboard, random response type generator, STT functionality using python 'pip' function in linux operation system. Followings are main python libraries we utilized in our proposed VUI system.

"Speech recognition" is a python library that includes multiple engines and APIs for STT perfomance in online and offline. Among several APIs, we employed Google web speech for this prototype which provides a default API key. This enables the code to access the microphone by creating an audio source and recognizing the user's speech by transcribing spoken words using the Internet.

"Keyboard" is a python library to access hardware such as an input device such as a keyboard. This library was used for event handling. We set the trigger as users press the specific keyboard from the keyboard to start the KWS system and continuously resume the user's utterance. By setting this controllability user can control when

to start and resume his/her utterance. This allows the user to speech at the desired timing.

"Random": is a built-in python module that can be used to make a random choice. We create a response type array and randomly selected it from the given array for every iteration under our experimental condition.

3.2 Test scenario

Step 1 (STT): When the user initiates a continuous utterance through a microphone connected to the Jetson TX2, the system automatically transcribes the utterance using Google speech API. The Google STT API transcribes the user's spoken sentences into a list of words, including spaces. The system prints the user's speech as shown in Fig. 4. "Tell me about the PCR test" indicates the user's speech converted from speech to text. The transcribed speech exists in python as a string.

Step 2 (Keyword parsing): After the STT process, the system parses the sentence in a word unit to spot the keyword. Since all the words in the sentence are already separated by spaces, we could parse the sentence by word unit in Python. These separated word units are compared with keyword lists stored in the database to find intersections. Among the 6 words in the user's speech, the intersected word was the "PCR" so the system recognizes "PCR" as a final keyword. The final detected keyword was printed as "supkey" variable.

Step 3 (Answering): In step 3, the system generates the path using the response type and detected keyword to derive the appropriate response for the user. The response type is randomly generated from the response type array based on the remaining response type array. The system accesses the subfolder from the current directory. For example, answer path for "WR" type and keyword "PCR" was "respo/WR/PCR.mp3" as shown in Fig. 4. In addition, the number of current remaining types were also printed as an array. This shows a result of limiting the number of response types up to 15.

Table. 2 shows the examples of detected keyword among the series of word transcribed from user's utterance. This 240 detected keywords are the output of comparing process described in step 2. 240 keywords are selected words that convey the most essential meaning in the each question. It can be a

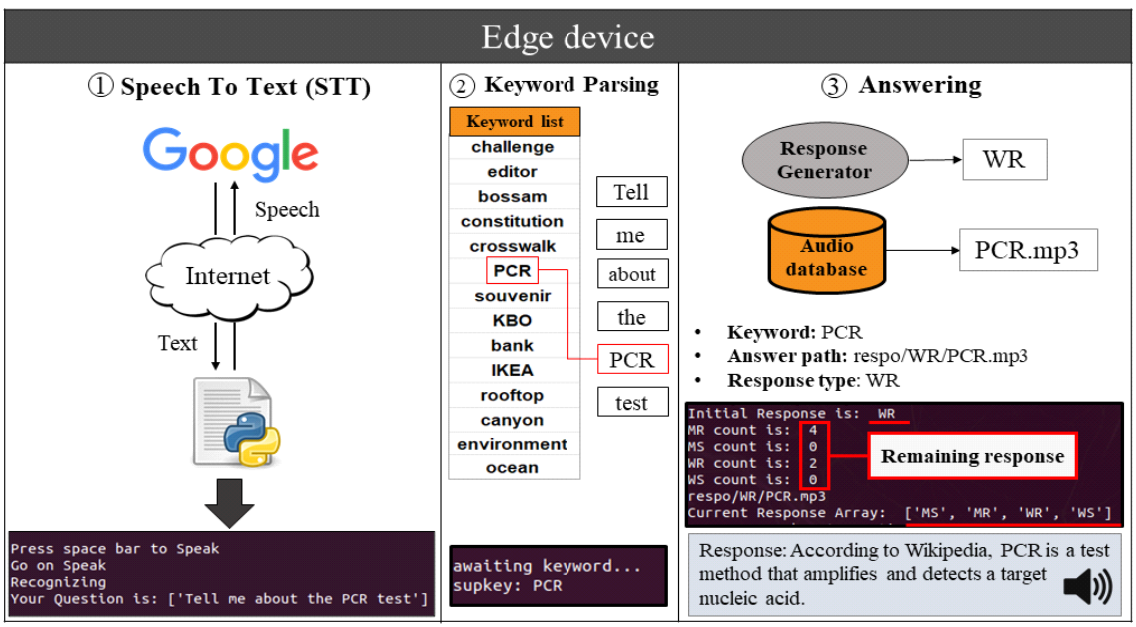

Fig. 4. Example of VUI procedure with question of "Tell me about the PCR test".

Table. 2. Examples of detected keyword in user's utterance.

Q.	User's utterance	Detected keyword
1	When did the Manchu war happened?	Manchu
22	When was Joseon founded?	Joseon
150	Where is the capital of Turkey?	Turkey
240	What foods are good for diabetes?	Diabetes

specific word or it can be a word combined with a Korean proposition. Korean postpositions are suffixes or short words in Korean grammar that immediately follow a noun or pronoun. Every word in the question sentence are separated by spaces, and keyword is determined by comparing them with the keyword list. Therefore, words including korean postpositions can be the keyword if necessary. When we select the keyword, the entire sentence was transcribed first using Google STT engine. The sentences were read multiple times, and a word that was stably recognized each time was designated as a keyword. The easier a keyword is to pronounce, the lower the error rate. In addition, Our proposed system is using an isolated word as a keyword. It's because the isolated word speech recognition system is lighter weighted than recognizing connected words or continuous speech. Thus, it is more appropriate to apply the isolated word recognition system in an embedded system that has relatively little computing power. In addition, the isolated word recognizer shows a higher recognition rate compared to other types of recognizers, which is effective for accurate speech recognition. All utterances and keywords in Table. 2. have been translated from Korean to English.

3.3 ASR performance evaluation

In this section, we evaluate the Google ASR method, which is pivotal function of our VUI system, as an error rate. The following set of experiments evaluates the variation in error rate that results from the shifting of numbers, words, and sentences. These 3 cases of the experiment were repeated 10 times in the noiseless space. Regarding the speech material, we tested with generic numbers from 0 to 20 in the first set. 14 syllables listed in the Korean dictionary were used as input in the second set. In the third one, we tested sentence recognition by selecting 2 sentences from the pre-registered question list in our database. Each sentence consists of 7 and 4-word segments respectively.

Fig. 5 shows the error rate for three cases (number, word, sentence). Results show a lowest average value of error rate in case of numbers, amounting to 5.09%, while showing the largest value when tested with independent word segments, amounting to 11.09%. In case of full sentences, the average error rate was 7.174%. This difference occurs because the Google STT API tends to predict and modify word pronunciation based on neighbor context. We tested on a series of consecutive numbers, such as 1 to 20, and in the sentence test, a sequence of words that could be followed by context was listed. Recognition rate showed higher value when Google speech recognizer to recognize a sequence of related words, such as numbers or sentences than recognizing a series of completely unrelated words. Various factors affect the recognition rate. For instance, background noise, instability of Internet connection, specification limitations of hardware can cause degradation of system performance. In case of numbers or sentences, Google speech recognizer transcribes them immediately after the user's utterance is finished. So it is difficult to see the problem as a

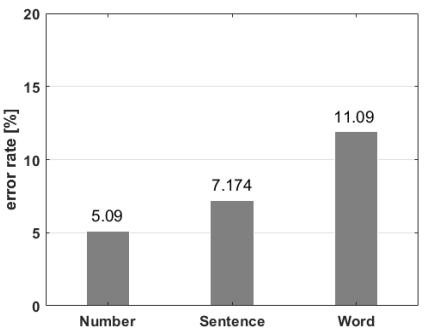

Fig. 5. Average error rate result per number, word, sentence.

limited specification of an embedded system. Also, internet speed used for the Google STT API have been found to be stable. Thus, this error rate has to do with Google ASR system in the STT stage.

Existing automatic speech recognition systems are generally powerful, but not perfect. Transcription errors in speech recognizers occur for several reasons. Variations in accent and speeds that differ from person to person, the substitution of homophones, audio quality and background noise, and the use of abbreviations and industry-specific jargon can cause transcription errors. In the Korean language, each region has a different accent. The absence of a contextual training or natural language processing engine for a particular training will cause the word error. And ASR system can replace the words with homophones which are words that sound the same but have different spellings of meaning. Regarding the background noise, the interaction was performed in a noiseless environment in our experimental setup. But if the sensitivity of the microphone is set too high, even a very small sound can be amplified and recognized as the user's voice. It is necessary to adjust the volume appropriately. In addition, substitution and deletion of other words may occur in the case of neologisms, abbreviations, and industrial jargon, which is not previously trained in the ASR system.

3.4 VUI performance evaluation

We measures the performance of the proposed VUI based on WER. WER is the most widely used metric for ASR evaluation[9]. The 240 questions used in the VUI simulation were transcribed as a sentence and WER was calculated to check for any missing or misspelled words. The lower is WER, the better the performance is. We also verified whether the system is providing the appropriate answer to the user based on the detected keyword. This metric measures the percentage of incorrect words (substitutions (S), insertion (I), deletions (D)) regarding the total number of words processed. It is defined as

$$WER = \frac{S+D+I}{N} \times 100(\%) \quad (1)$$

where I is total number of insertions, D is total number of deletions, S is total number of substitutions, N is total number of words in groud truth text. Based on this, we measured WER with a total of 240 Korean inquiry sentences. As a result, the average WER was 9.28%. This WER occurred by the factors which were discussed in the previous section, such as accent, speed, and substitution of homophone. According to data on [10], ASR systems developed by Google have a WER of 14.29% for the clean speech. Compared to this result, our measured WER showed improved recognition rate. Although this WER occurred, since this recognition error has occurred for words other than the keyword part, there was no difficulty in providing an appropriate answer to the users in our VUI system.

Ⅳ. Conclusion

In this paper, we have implemented a reliable embedded VUI system which can interact with the user in a real time. Our proposed VUI was performed based on keyword detection method via speech recognition. In this reaseach, we built a Korean voice database and developed a small-scale speech recognition engine uploading it on the embedded Jetson TX2 board. The database consists of response audio corresponding to 240 keyword which is divided by 4 speech types. This 4 speech types shows variation in the voice gender and amount of information so that the user can interacts with diverse speakers on VUI system. Since the size of the system memory is limited when applying application to embedded system, weight reductions should be considered. In this study, the total memory requirement of the system is 150Mbytes, and response time is 500ms, which is optimized to implement real-time embedded VUI system.

References

[1] Y. J. Choi, et al., "Comparison of the performance of uav remote control based on online and offline voice recognition," *J. KICS*, vol. 46, no. 4, pp. 688-695, Apr. 2021. (https://doi.org/10.7840/kics.2021.46.4.688)

[2] H.-J. Yoo, et al., "Comparative analysis of Korean continuous speech recognition accuracy by application field of cloud-based speech recognition open API," *J. KICS*, vol. 45, no. 10, pp. 1793-1803, Oct. 2020. (https://doi.org/10.7840/kics.2020.45.10.1793)

[3] J. Ha, et al., "Individual differences in perception toward English pronunciation development with ASR technology," *Multimedia Assisted Lang. Learn.*, vol. 24, no. 3, pp. 10-34, Mar. 2021. (https://doi.org/10.15702/mall.2021.24.3.10)

[4] H. Yang, et al.,"Hyperparameter experiments on end-to-end automatic speech recognition," *Phonetics and Speech Sci.*, vol. 13, no. 1, pp. 45-51, Jan. 2021. (https://doi.org/10.13064/KSSS.2021.13.1.045)

[5] P. Iswarya, et al., "Speech query recognition in Tamil language using wavelet and wavelet packets," *JIPS*, vol. 13, no. 5, pp. 1135-1148, May 2017. (https://doi.org/10.3745/JIPS.02.0033)

[6] Y. Oh, et al., "Automatic proficiency assessment of Korean speech read aloud by non-natives using bidirectional LSTM-based speech recognition," *ETRI J.*, vol. 42, no. 5, pp. 761-772, May 2020. (https://doi.org/10.4218/etrij.2019-0400)

[7] S. Leem, et al., "Multitask learning of deep neural network-based keyword spotting for IoT devices," *IEEE Trans. Consum. Electron.,* vol. 65, no 2, pp. 188-194, Feb. 2019 (https://doi.org/10.1109/TCE.2019.289967)

[8] *Nvidia jetson tx2*, https://developer.nvidia.com

[9] R. Errattahi, et al., "Automatic speech recognition errors detection and correction: A Review," *Procedia Comput. Sci.*, vol. 128, pp. 32-37, 2018. (https://doi.org/10.1016/j.procs.2018.03.005)

[10] B. Xu, et al., "A benchmarking on cloud based speech-to-text services for french speech and background noise effect," *ArXiv abs/2105.03409*, 2021. (https://doi.org/10.48550/arXiv.2105.03409)

김 정 인 (Jeongin Kim)
2019년 3월~현재 : 금오공대, 전자공학부
<관심분야> 임베디드 시스템, 인공지능
[ORCID:0000-0002-3816-6300]

폴 (Paul Angelo Oroceo)
2021년 3월~현재 : 금오공대, 항공기계융합전공
<관심분야> 임베디드 시스템, 지능형 제어
[ORCID:0000-0002-9305-6278]

임 완 수 (Wansu Lim)
2014년 9월~현재 : 금오공대 전자공학부 부교수
<관심분야> 임베디드 시스템, 지능형 제어
[ORCID:0000-0003-2533-3496]