Example of Google Cloud Speech API usage and interpretation of recognized content on the E-ink display

Igor Medenica, Nikola Radulović, Vladimir Milovanović, Jelena Vasiljević and Miloš Jovanović

***Abstract*— Accelerated lifestyle and increasing technological development requires an ever faster way to establish communication between two people. Virtual communication has become a common and frequent way of contact between people in the 21st century. It is performed via the Internet, voice or SMS messages. This paper presents one of the custom types of accelerated communication. Using the Cloud Speech API, speech recognition and conversion to text format is performed. The recognized content from the Android device is transmitted via Bluetooth communication to the Arduino UNO to which the E-ink display is connected, on which the received content is displayed. The use of this system speeds up communication with people who are unable to communicate in any other way, while maintaining privacy. The advantages of use are portability and small dimensions, which enable easy viewing of content without the use of a mobile device.**

***Index Terms*— Google Cloud Speech to Text, Waveshare e-Paper display, Arduino UNO, HC-05**

I. Introduction

The development of information and communication technologies contributes to the introduction of numerous innovations that facilitate and accelerate the establishment of contact between two or more persons. Smartphones contribute to the reduction of direct communication and increase the text communication that is realized via SMS messages or via the Internet. Voice over text (Automatic Speech Recognition, ASR) is becoming a very popular method that is increasingly used in practice.

Voice recognition has been brought to a level of perfection with a small percentage of error. The development of artificial intelligence enables a large number of ASR systems that provide high quality transcripts. Quite a large number of successful technology companies have created such systems for both research and improvement of technological services. Some of the famous ones are Google Cloud, IBM Watson, Microsoft Azure.

Research has shown that the quality of transcripts reaches the same level as manual transcripts [1]. This type of technology supports the daily life of people with hearing impairments. In people with this disability, textual communication via smart devices is present in a large percentage [2]. Feedback on whether a message has arrived or not, whether a person is able to respond and much more information that can replace a direct form of communication plays an important role [3]. Many of these services strive to customize tools to use as many spoken languages as possible. The distinction between ASR systems can be divided into systems that recognize isolated words and those that recognize related isolated words, and a large number of systems are distinguished by the size of the dictionary (the amount of words which can be recognized) [4]. The Google Cloud Speech-to-Text API is used to make the problem solving more successful.

In addition to all the advantages of this type of system, one of its main disadvantages is portability, storage space and battery life [5]. This system tries to show that with the selected hardware and the use of free Google services, these problems can be avoided.

II. Design of the System

To implement the presented system, we used an Android phone, Arduino UNO programming board, HC-05 Bluetooth module and Waveshare e-Paper display. The system can be grouped into two entities (Fig. 1):

- speech recognition - consists of an Android application that communicates with the Google Cloud platform
- data presentation - consists of an Arduino Uno board with a Bluetooth module and an E-ink display

The user starts a conversation session via an application on a smartphone with the Android operating system. User speech is recognized using the Google Cloud API. The mentioned platform converts the recognized audio sample into text format in real time. Text content is forwarded via Bluetooth communication to an Arduino Uno board that prints the transcript on an e-Paper display.

Igor Medenica is with the School of Computing, University UNION, 6 Knez Mihajlova, 11020 Belgrade, Serbia (e-mail: medenicaigor@gmail.com).

Nikola Radulović is with the Faculty of Science, University of Novi Sad, Trg Dositeja Obradovica 3, 21000 Novi Sad, Serbia (e-mail: nikolardlvc@gmail.com).

Vladimir Milovanvić is with the School of Computing, University UNION, 6 Knez Mihajlova, 11000 Belgrade, Serbia (e-mail: vmilovanovic@raf.rs).

Jelena Vasiljević is with the School of Computing, University UNION, 6 Knez Mihajlova, 11000 Belgrade, Serbia (e-mail: jvasiljevic@raf.rs).

Miloš Jovanović is with the School of Computing, University UNION, 6 Knez Mihajlova, 11000 Belgrade, Serbia (e-mail: mjovanovic@raf.rs).

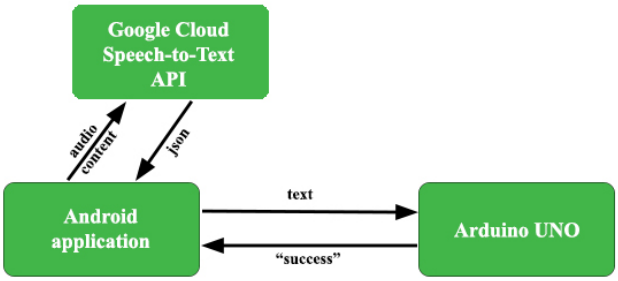

Fig. 1. System structure and communication between entities

A. Hardware Infrastructure

The paper used a smartphone with Android operating system version 9.0. The features of the phone are 2 GB of RAM and it has a CPU with four cores.

The Arduino Uno is an ATmega328P based microcontroller board. It has 14 digital input / output pins (6 of which can be used as PVM outputs), 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB port, a power port, an ICSP header and a reset button.

The HC-05 is one of the types of Bluetooth modules that has two-way wireless functionality. This module can be used to communicate between two microcontrollers such as an Arduino or to communicate with any device with a Bluetooth function, such as a mobile phone or laptop. It has two modes of operation, one is the data mode in which it can send and receive data from other Bluetooth devices, and the other is the AT Command mode where the default device settings can be changed. Schematic representation of the module in the figure is shown in the figure 2. (Fig. 2).

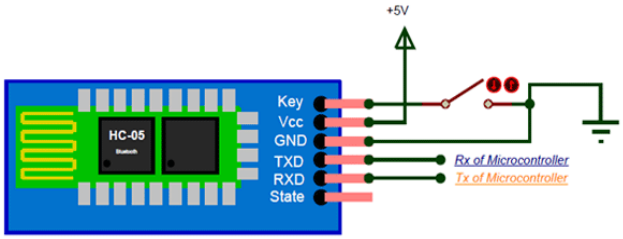

Fig. 2. Schematic representation of the Bluetooth module HC-05

E-ink or e-Paper indicates electronic use and mimics the appearance of normal ink on paper. It is useful when displaying text due to its high visibility and contrast, wide viewing angle and low power requirements. Once programmed, it remains unchanged until reprogrammed, even if there is no power. We used a 1.54 inch display. The advantage of this display is that under sunlight it has great visibility with a wide viewing angle of 180 degrees [6].

B. Speech Recognition

The speech recognition part of this solution is an Android application that communicates with the Google Cloud Speech API. The Google Cloud platform enables the development of applications that have the functionality to convert audio to text using neural network models.

This platform is a part of the system that recognizes spoken content in real time. Using the Google platform, a user's voice message is converted into textual content or transcript and displayed in the Android application. The user checks the authenticity of the converted content. Upon confirmation of accuracy, the transcript is forwarded and presented to another user. The implementation of frontend component is achieved in a few steps:

- Configuring Google Cloud API
- User Interface Design
- implementation of communication between Android application and Google Cloud API
- implementation of communication between Android application and Arduino UNO board

C. Data Presentation

The data presentation part of the system contains three components (Arduino Uno, Bluetooth module, E-ink display). This part of the system is based on the Arduino Uno board on which the HC-05 Bluetooth module is attached, which communicates with the Android phone. The presentation of the received content is done on the Waveshare e-Paper display connected to the Arduino board. The implementation of this part is divided into two parts based on the use of two modules:

- implementation of communication
- implementation of text printing on display

III. Description of Speech Recognition

A. Configuring Google Cloud API

Machine learning is part of the Google Cloud platform in creating applications that can hear, see and understand the world around them. In pretrained machine learning models, the Google Translate API and the Cloud Vision API are integrated into the Google Cloud Speech API.

By applying the already mentioned model of neural networks that converts an audio sample into a text format, it gives us the possibility of conversion in several ways. By recognizing sound in real time through a microphone through the application, sending an already recorded file from local memory or uploading to Google Storage.

This platform currently supports more than 125 languages [7]. The following steps were used to configure this platform:

- Speech API Approval
- Creating access credentials
- Creating a private key

B. User Interface Design

A simple design of the Android speech recognition application is described in the following two phases of work.

During the first phase, a screen is displayed waiting for the user to record a voice message.

Through the second phase, a screen is displayed after the user records a voice message the application will send the data to Google Cloud. After processing the data on the platform, the input voice is converted to text and displayed in TextView.

C. Implementation of communication between Android and Google Cloud API

As mentioned above, the API allows audio sample conversion in a number of ways. Realtime sound recognition was chosen in this study. By sending a request from the application, it is possible to get temporary and final recognition results in response. Provisional results are represented by current recognized patterns, while the final result is the best hit.

Calling the Streaming Speech API requires multiple requests, while configuration and audio are sent in a single request. The StreamingRecognizeRequest function imported within the com.google.cloud.speech package is used to forward platform requests [8].

Parts of the audio recognition are sent in sequential StreamRecognizeRequest messages. The first message must not contain audio content, while all subsequent StreamingRecognizeRequest messages must contain audio content. Audio bytes must be encoded as specified in RecognitionConfig. StreamingRecognizeConfig consists of the fields shown in the table (Table I).

TABLE I
DESCRIPTION OF STREAMINGRECOGNIZECONFIG PARAMETER FIELDS

Parameter fields	Description
config	contains information for audio, type RecognitionConfig
single_utterance	(default value is false) indicates whether the request automatically terminates if there is no more speech detection
interim_results	(default value false) indicates that this stream request should return temporary results that can be refined later

For StreamingRecognize requests, the sound must be sent at a speed that roughly corresponds to real time. Attempting to process content that exceeds the defined limits will result in an error.

A SpeechContext resource, in which one word counts as a phrase, contains a list of request-specific phrases which can also be provided with any request. The limit on phrases per request is a maximum of 5,000, 100 characters per phrase, and a maximum number of 100,000 characters [9]. A maximum of 900 requests per 60 seconds is possible, while each StreamingRecognize session is considered a single request even though it includes multiple StreamingRecognizeRequest sound frames in progress.

Through the RecognitionConfig, the parameters of the encoding type, the language code and the speed pattern in Hz type integer. Sample speed in Hz audio samples sent in all RecognitionAudio messages. Valid values are: 8000-48000. For best results, set the sampling frequency of the audio source to 16000 Hz. If this is not possible, the original sample rate of the audio source is used. This field is not required for FLAC and WAV audio files, but is required for all other audio formats. The Google Cloud platform does not support stereo audio files. LINEAR16 is used for AudioEncoding and Serbian is selected as the recognition language under LanguageCode, sr-RS is forwarded.

To respond, you need to create a StreamObserver array of StreamingRecognitionResponse type. The answer consists of the fields shown in the table below (Table II).

TABLE II
DESCRIPTION OF STREAMINGRECOGNITIONRESPONSE PARAMETER FIELDS

Parameter fields	Description
error	retuns Rpc status message
results	contains zero or one is_final = true (newly formed part), followed by zero or is_final = false (provisional result)
speech_event_type	type of speech event

StreamingRecognizeResponse is the only message returned to the client by StreamingRecognize. A series of zero or more StreamingRecognizeResponse messages is returned to the client. If there is no recognizable sound, and single_utterance is set to false, then no message is returned to the client. Based on the stability parameter, the one with the highest value is displayed by comparing the results (Table III).

TABLE III
DESCRIPTION OF SPEECHEVENTTYPE PARAMETER FIELDS

Parameter fields	Description
config	contains information for audio, type RecognitionConfig
single_utterance	(default value is false) indicates whether the request automatically terminates if there is no more speech detection

D. Implementing of communication between Android and Arduino UNO board

A transcript is obtained in response to the audio sample. Invoking the actionListener by clicking the "Confirm" button, the user makes a choice if the returned sample has a valid context, otherwise it is "Wrong". All successfully recognized transcripts are forwarded via Bluetooth connection to the Arduino UNO.

The connection for the HC-05 Bluetooth module used in

operation was implemented. The required parameters for communication are its MAC address and UUID. BLUETOOTH and BLUETOOTH_ADMIN permissions need to be imported to AndroidManifest in order to enable Bluetooth communication.

During initialization, an attempt is made to establish communication with the Bluetooth module. First, it is checked whether the Bluetooth communication on the phone is switched on, if not, an Intent of the BluetoothAdapter.ACTION_REQUEST_ENABLE type is created, which starts the confirmation dialog.

Through the initBluetoothProcess() method, it establishes a connection with the module and creates a handler that receives information via ConnectedThread. This thread is in charge of communication with the module via BluetoothSocket. Each successfully sent message is answered with the value "success".

IV. Description of data presentation

The following steps show how to start sending a transcript of a recognized audio sample for printing on an e-Paper display:

1. By clicking the button, the user starts sending from the smartphone via Bluetooth communication
2. The Bluetooth module accepts the message and sends it to the Arduino
3. The Arduino processes the received message, which is printed on the Waveshare e-Paper display
4. The Arduino sends a message to the Bluetooth module, which returns this message to the smartphone

A. Implementation of communication

By importing the SoftwareSerial library, communication with the Bluetooth module is enabled.

First, it is checked whether there are incoming messages from the Bluetooth module, when receiving the message, byte by byte is read (reading the "char" type of each iteration). In order for messages to arrive successfully, it is necessary to pass 10 delay functions. Iteratively, byte by byte of the message is read and placed in the buffer. After the message is successfully placed, a flag is placed for the end and '\0' is added, which marks the end of the buffer. Finally, it is checked once more to see if there are any more incoming messages and all counters and buffer are set to zero.

B. Implementation of text printing on display

The epd.h library provided by Waveshare is used to present the text on the e-Paper display, to allow easy communication of the Arduino board with the screen. After initializing the screen, wake-up is activated using epd_wakeup function and the memory it uses during operation is defined. This display has the option of using its built-in NAND memory or external memory via SD card.

Receiving a message calls the printOnDisplay function. In order to clear the previous content from the display, the epd_clear function is used and the received message is printed with a call to the epd_disp_string function, which is forwarded with a text message, x and y positions. The screen content needs to be refreshed by calling epd_update function.

The screen used is low refresh rate and therefore it cannot be used to display animations, while in this research it proved to be effective.

V. System testing

The solution was implemented through four phases of testing divided into two parts. We used two languages, Serbian and English.

Each phase had its own peculiarities. In the first phase, random numbers from one to one hundred were spoken. From the second to the fourth phase, the complexity increases. In the second phase, two-syllable words were used, and with each phase the number of syllables was increased. Each of the phases used fifty words and the success of their detection was tested (Table IV).

TABLE IV
TABLE SHOWING MEASURED HITS IN FOUR PHASES DIVIDED INTO TWO CYCLES

	I	II	III	IV
Serbian	41/50 82%	32/50 64%	26/50 52%	15/50 30%
English	47/50 94%	44/50 88%	39/50 78%	31/50 62%

This testing has shown that the success in recognizing the Serbian language decreases significantly with increasing number of phrases. Thus, we conclude that currently the better choice is the English language for a more authoritative functioning of our system.

VI. Conclusion

This paper presents one of the ways to use the Google Cloud Speech-to-Text API with the aim of speeding up and facilitating communication between two people.

Based on this research, for better functioning of the system, it is necessary to move the communication between people through the server. With this improvement, the use can be extended to more people without any distance restrictions. Adding the Google Translate API to the system, allows us to select the language for the presentation of recognized audio content.

References

[1] J L. Deng and X. Huang, "Challenges in adopting speech recognition," Commun. ACM, vol. 47, no. 1, pp. 69–75, 2004.

[2] Power, M.R. and D. Power, Everyone Here Speaks TXT, "Deaf People Using SMS in Australia and the Rest of the World," Journal of Deaf Studies and Deaf Education, 9(3): p. 333-343, 2004.

[3] Tucker, W., E. Blake, and M. Glaser, Building Bridges for Deaf Telephony in South Africa A Community-Centred Approach. 'Information Technology in Developing Countries', 13(2), 2003.

[4] M Stenman. "Automatic speech recognition An evaluation of Google Speech". UMEA UNIVERSITY. 2015.

[5] A. Glasser, K. Kushalnagar, and R. Kushalnagar, "Deaf, Hard of Hearing, and Hearing perspectives on using Automatic Speech

Recognition in Conversation," Rochester Institute of Technology., Rochester, NY 14623, 2019.
[6] Waveshare, "1.54 inch e-Paper Module", 2020. [Online]. Available: https://www.waveshare.com/wiki/1.54inch_e-Paper_Module#Arduino_UNO [Accessed 06.09.2020.]
[7] Google Cloud Speech-to-Text, "Language support", 2020. [Online]. Available: https://cloud.google.com/speech-to-text/docs/languages, [Accessed 06.09.2020.]
[8] Google Cloud Speech-to-Text, "API", 2020. [Online]. Available: https://cloud.google.com/speech-to-text/docs/streaming-recognize, [Accessed 06.09.2020.]
[9] Google Cloud Speech-to-Text, "Quotas & limits", 2020. [Online]. Available: https://cloud.google.com/text-to-speech/quotas, [Accessed 06.09.2020.]