

Implementation of Google Assistant on Raspberry Pi

Septimiu Mischie
Faculty of Electronics,
Telecommunications and Information
Technologies
Politehnica University of Timisoara
Timisoara, Romania
septimiu.mischie@upt.ro

Liliana Mățiu-Iovan
Faculty of Electronics,
Telecommunications and Information
Technologies
Politehnica University of Timisoara
Timisoara, Romania
liliana.matiu-iovana@upt.ro

Gabriel Găspăresc
Faculty of Electronics,
Telecommunications and Information
Technologies
Politehnica University of Timisoara
Timisoara, Romania
gabriel.gasparesc@upt.ro

Abstract—This paper investigates the implementation of the voice Google assistant on a Raspberry Pi microcomputer. First, the Voice Kit of Google is presented. This device can be attached to a Raspberry Pi and a voice Google assistant is obtained. The necessary steps to implement the voice Google assistant on a system that contains among the Raspberry Pi only a microphone and a speaker, instead of the Voice Kit, are presented. In this way a voice Google assistant can be made more easily and flexible. This newly created device has several working modes that are analyzed. Finally, a speech recognition system that works in Romanian language is presented and evaluated.

Keywords—google assistant, voice, Raspberry Pi, speech recognition, Python.

I. INTRODUCTION

Spoken dialogue systems or voice-controlled assistants are devices that can respond to multiple voices, regardless of accent, can execute several commands or can provide an answer, thus imitating a natural conversation [1-3]. Such a system, Fig. 1, contains more elements: an automatic speech recognition part (ASR), a spoken language understanding (SLU), a dialogue manager (DM), a knowledge data base (KDB), a natural language generator (NLG) and a text to speech synthesis (TTS) system [7]. One of the most important parts of such an assistant is represented by speech recognition also called speech to text translation because it transforms human voice into a string of data than can be interpreted by the computer. There are some open source software packages that allow speech recognition such as Kaldi [4] or Pocket Sphinx [5]. However, in recent years cloud-based speech recognition systems have been developed a lot. In this way, all elements of a voice controlled assistant are placed in cloud. The most important ones from this category of assistants are Apple Siri, Google Assistant and Amazon Alexa. They are present in most smartphones and are based on artificial intelligence elements such as deep learning and neural networks. Apart from voice digital assistants, other systems that have been developed using this idea are call centers [6], systems for self-management of diabetes patients [7] or shopping assistant robots [8].

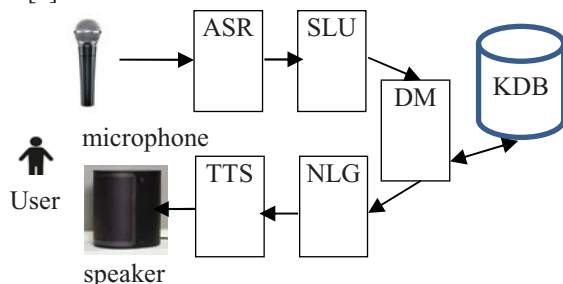


Fig. 1. The structure of the voice controlled assistant

Another field where the speech recognition is widely used is represented by automotive industry [13]. Thus, it is estimated that 55% of the cars will contain a voice recognition system in 2019. An important part of the car's function can be controlled by voice, especially those on entertainment area.

Google has launched in May 2017, a project called AIY (Artificial Intelligence Yourself) Voice Kit [9-10]. This device contains mainly a so-called hardware on top (HAT) that can be connected on the GPIO (General Purpose Input Output) connector of a Raspberry Pi 3 microcomputer, a microphone, a speaker and an Arcade style button. The last three components must be connected to the HAT by suitable wires and connectors. The function of this combo device, AIY Voice Kit and Raspberry Pi 3, correspond to those of a voice assistant. Thus, by using Google's cloud facilities it can answer questions asked by the user. It can be further improved to execute some commands because the GPIO pins of the Raspberry Pi are available on the HAT and thus it can be interfaced with other devices as remote controls, actuators, sensors. The software part of this project contains several Python files and is included on the image of the Raspbian operating system (OS) which is available on the AIY Voice Kit site [10].

We have acquired, installed and used the AIY Voice Kit. However, in this paper we present the implementation of a Google assistant on a Raspberry Pi without the AIY Voice Kit. That means we used a usual USB microphone and a speaker for the 3.5 mm audio output of this microcomputer. Other applications such as speech recognition in different languages especially in Romanian are explored. All of these imply knowledge of the operating system such as accessing the audio devices or knowledge of Python language to modify the initial form of the software part. In this way, the Google assistant can be embedded in a hardware device. This is an advantage in comparison with [11] where a smartphone is used as an additional element in an assistive device for visually impaired individuals. Thus, the smartphone receives voice commands and it sends them via Bluetooth to an Arduino-based system that accesses a speaker or a motor to implement some tasks depending on the voice commands. The paper is organized as follows. In the second section the necessary changes to implement the Google assistant on the Raspberry Pi without the AIY Voice Kit are presented. The third section presents the applications that can be executed using the initial form of the software among the newly introduced ones and the last section concludes this work.

II. IMPLEMENTATION OF GOOGLE ASSISTANT ON RASPBERRY PI WITHOUT THE AIY VOICE KIT

The software part of AIY project is included in the image of a Raspbian OS that can be downloaded from their site [10]. All of the following explanations regarding the

software are valid for the release called `aiyprojects-2017-09-11.img.xz`. Thus, after installing the Raspbian OS, the Google assistant can be used. It is available on `/home/pi/AIY-voice-kit-python`. Mainly, the entire software of this project is designed to use an input to record or capture speech signal and an output to generate the audio answer. By default, these two elements correspond to the Voice Kit. This device contains two digital microphones (only one is used) that communicate with the Raspberry Pi via a serial interface and a serial controlled DAC (digital to analog converter) that commands the speaker. Due to the HAT driver the two digital devices are seen by the OS as two usual audio devices as it will be seen in the following paragraphs.

In order to find the available audio devices, the Linux command can be executed:

```
cat /proc/asound/modules
```

Thus, if this command is executed during the initial form of the software, the answer is

```
0 snd_soundcard_googleVoice_hat
```

A similar response can be obtained by the commands `aplay -l` and `arecord -l` that present the available Playback, and the recording device, respectively. Furthermore, with a right click on the speaker icon from the status bar of the Raspbian OS, the same information is obtained. From all of these it follows that both recording and playback can be performed with the same device, `soundcard_googleVoice_hat`, that corresponds to the Voice Kit.

In order to implement the Google assistant onto the Raspberry without the Voice Kit we must disable the audio devices from the Voice Kit and validate the onboard audio output of the Raspberry Pi. Then we need to verify if a usual USB microphone device is recognized.

Thus, by considering the file `/boot/config.txt`, the lines `dtoverlay = i2s-map` and `dtoverlay = googlevoicehat-soundcard` must be removed while the line `#dtparam=audio=on` must be activated to enable loading the driver `bcm2835` that corresponds to the onboard audio output. After a reboot, the answer to the command `cat /proc/asound/modules` is `0 bcm_snd2835`, as expected. Then, if an USB microphone (actually we used the Logitech C270 USB webcam that includes a microphone too) is connected, the answer to the same command will be `0 bcm_snd2835` and `1 snd_usb_audio`. The numbers 0 and 1 represent so-called cards. If the commands `arecord -l` and `aplay -l` are executed again it follows that card 0 is used for playing while card 1 is used for recording. On the other hand, a right click on the speaker icon presents now `HDMI` (checked) and `Analog`. In order to allow the sound to be sent to the onboard audio output, `Analog` option must be checked.

Then the file `/etc/asound.conf` must be modified to have the following structure

```
pcm.!default {
    type asym
    playback.pcm "speaker"
    capture.pcm "mic"}
pcm.mic {
    type plug
    slave {
        pcm "hw:1,0"}
}
```

```
pcm.speaker
    type plug
    slave {
        pcm "hw:0,0"
    }
}
```

In this way the default devices are those having the card number 1 (microphone) and 0 (onboard audio output). Thus, recording and playing can be performed using suitable commands. However, there is an icon on the desktop of Debian OS, *Check audio*, than can be used for an interactive test that verifies playing and recording. By double clicking on this icon a python file is executed (`AIY-voice-kit-python/checkpoints/checkaudio.py`) but it must be modified before, namely the line `VOICEHAT_ID = 'googlevoicehat'` must be replaced by `VOICEHAT_ID = 'bcm2835'`.

If everything is all right it can move on to execute the applications such as those from the following section.

There are only two problems before to test the AIY project. They are regarding the arcade style button that also contains a lamp. Thus, in our implementation of the Google assistant that does not contain the Voice Kit, the function of the arcade style button is replaced by a push button that must apply the GND to GPIO23 pin and a LED that must be connected to GPIO25 pin.

III. TESTING AND IMPROVEMENT OF AIY APPLICATIONS

The experimental setup that allows us implementing the Google assistant is presented in Fig. 2. It is based on a Raspberry Pi 3 system that contains a WiFi adapter. This is absolutely necessary because Google assistant needs an Internet connection which must be available all the time. The software part of the AIY project has two categories of applications. The first one is based on Google Assistant SDK and the other one is based on Cloud Speech API. The first one is completely free while the second one requires minimal fee. Both of them require a Google account and several steps that include some information such as name of the project. This registration part is finalized by downloading of two `.json` files. The first one has the name `assistant.json` and is used for Google Assistant SDK applications. The other one has the name `cloud_speech.json` and is used for Cloud Speech API applications.

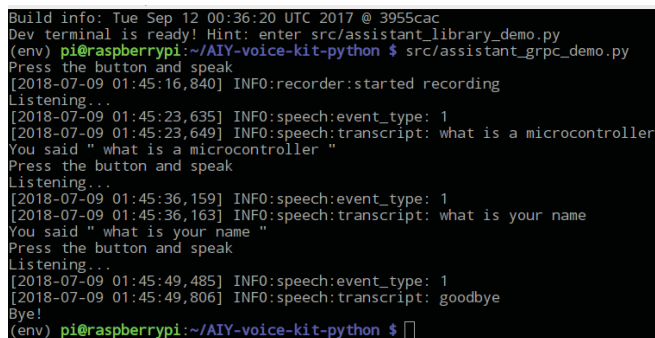


Fig. 2. The experimental setup

The AIY project applications are included in the `AIY-voice-kit-python/src` folder and are written in Python language. The applications `assistant_grpc_demo.py` and `assistant_library_demo.py` need `assistant.json` to be run,

while *cloudspeech_demo.py* needs *cloud_speech.json* to be run.

When any of these applications are run for the first time, the Google website is accessed and the user name and the password are required. On subsequent runnings, no further login information is required, but however the internet access is necessary because all the components of the Google assistant are in cloud servers. First, the assistant *_grpc_demo.py* is presented. It asks the user to press the connected push button and then the *Listening...* message is displayed, Fig. 3. That means the user can say a few words. The spoken text is immediately displayed and the answer of the assistant is then generated to the speaker. This of course could not be identified in Fig.3. The connected LED becomes ON after the user pressed the button and remains in this state until the end of the answer of the assistant. It can be seen from Fig. 3 that the spoken text “Good bye” has the effect to ends the application.



```
Build info: Tue Sep 12 00:36:20 UTC 2017 @ 3955cac
Dev terminal is ready! Hint: enter src/assistant_library_demo.py
(env) pi@raspberrypi:~/AIY-voice-kit-python $ src/assistant_grpc_demo.py
Press the button and speak
[2018-07-09 01:45:16,840] INFO:recorder:started recording
Listening...
[2018-07-09 01:45:23,635] INFO:speech:event_type: 1
[2018-07-09 01:45:23,649] INFO:speech:transcript: what is a microcontroller
You said " what is a microcontroller "
Press the button and speak
Listening...
[2018-07-09 01:45:36,159] INFO:speech:event_type: 1
[2018-07-09 01:45:36,163] INFO:speech:transcript: what is your name
You said " what is your name "
Press the button and speak
Listening...
[2018-07-09 01:45:49,485] INFO:speech:event_type: 1
[2018-07-09 01:45:49,806] INFO:speech:transcript: goodbye
Bye!
(env) pi@raspberrypi:~/AIY-voice-kit-python $
```

Fig. 3. A screenshot from the terminal during running assistant_grpc_demo.py application

By studying the source of the application we have observed that there is a very powerful function

text, audio = assistant.recognize() that waits for an input speech and then returns the recognized text as the variable *text* and the answer as the variable *audio*. This can be played by the function

aiy.audio.play_audio(audio)

According to [10] this application, *assistant_demo_grpc.py*, can be used for a dialogue with the assistant while the application *cloudspeech_demo.py* can be used to implement new commands that do not need answers. However, because the variable *text* is available, some simplest tests of this variable can be used to generate commands.

The application *assistant_library_demo.py* is mainly similar to the previous application. The differences are: the trigger of the application is represented by the words “OK Google” while the decoded speech is not displayed on the screen. Furthermore, the connected LED becomes ON when the trigger is detected and it blinks during the answer of the assistant.

As we previously presented the application *cloudspeech_demo.py* uses Cloud Speech API and is not free. However the cost of using it is negligible. When we made the operations to obtain the *cloud_speech.json* file, we obtained a credit that could be used for 12 months. This application does not return any audio answer. It only returns the spoken text, by function *text = recognizer.recognize()*

Thus, according to the content of the variable *text* some actions can be executed as we previously presented for the application *assistant_grpc_demo.py*. The main advantage of this application is that it can make speech recognition in about 80 languages.

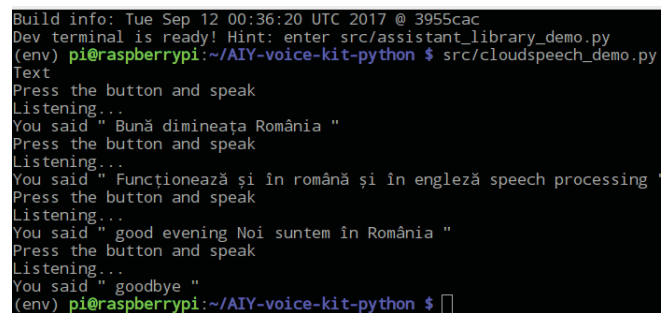
To change the language used for speech recognition, the following program line from the file *AIY-voice-kit-python/src/i18n.py* must be used:

DEFAULT_LANGUAGE_CODE = ‘ro-RO’

where ro-RO is the code for Romanian language. All the codes are available on [12]. In Fig.4 a capture obtained during running of *cloudspeech_demo.py* application is presented. It can be seen that there is the same trigger (button) as in *assistant_grpc_demo.py* application. On the other hand the application recognizes at the same time the selected language (Romanian language in this example; we selected German language too) but also English.

In order to create an application we connected three LEDs, having the colours blue, red and yellow, respectively, to the appropriate GPIO pins. This application allows the control of the status of the three LEDs through Romanian words as “Led albastru aprins”, “Led albastru stins” for the Blue LED and in a similar way for the other two LEDs. As it was described in the previous example, this function is built by accessing the variable *text*. In the following, a short part of the program is presented:

```
if text == 'led albastru aprins':
    print('albastru ON')
    GPIO.output(24,1)
if text == 'led albastru stins ':
    print ('albastru OFF')
    GPIO.output(24,0)
```



```
Build info: Tue Sep 12 00:36:20 UTC 2017 @ 3955cac
Dev terminal is ready! Hint: enter src/assistant_library_demo.py
(env) pi@raspberrypi:~/AIY-voice-kit-python $ src/cloudspeech_demo.py
Text
Press the button and speak
Listening...
You said " Bună dimineața România "
Press the button and speak
Listening...
You said " Funcționează și în română și în engleză speech processing "
Press the button and speak
Listening...
You said " good evening Noi suntem în România "
Press the button and speak
Listening...
You said " goodbye "
(env) pi@raspberrypi:~/AIY-voice-kit-python $
```

Fig. 4. A screenshot from the terminal during running of the *cloudspeech_demo.py* application

Here we present an experiment that examines the response time of the implemented Google assistant. The previous application was modified so that it contains a counter that is incremented for each spoken text and which also displays the current time stamp when the string of text is received. In addition, this application no longer requires the user to press the button to be able to speak, as of Fig.4. Instead, the user can speak when the message “Listening...” is displayed. While this application is running, the user does the following:

- 1.Says the text ‘albastru aprins’
- 2.Waits until the blue LED becomes ON
- 3.Says the text ‘albastru stins’

4. Waits until the blue LED becomes OFF

5. Goes to step 1.

We executed this experiment of few times, until the counter got different values, for instance 21. Fig. 5 presents a screenshot from the terminal during this experiment, while Fig. 6 presents the differences between successive time stamps. It can be seen that the minimum value is about 2 sec and the maximum one is 2.6 sec. We computed the average of these time difference and obtained 2.3 sec. This value represents the period of time necessary to execute voice commands. Of course this value contains the time to say the text (about 1 sec), to send the speech samples in cloud, to find and receive the spoken text. This also depends on the user concentration to observe the changes of the LED's state and to make a correct pronunciation. However this value is acceptable for a practical application.

Even if we did not make an intensive test regarding the accuracy of this assistant to recognize the spoken words, we can say that during our tests an accuracy of at least 90 percent has been obtained.

```
You said" albastru stins " 6
Listening...
2018-07-05 01:13:51.731675 end
You said" albastru aprins " 7
Listening...
2018-07-05 01:13:54.176298 end
You said" albastru stins " 8
Listening...
2018-07-05 01:13:56.350310 end
You said" albastru aprins " 9
Listening...
2018-07-05 01:13:58.491273 end
You said" albastru stins " 10
Listening...
2018-07-05 01:14:00.860581 end
You said" albastru aprins " 11
Listening...
```

Fig. 5. A screenshot of the terminal during the experiment for measuring the time response

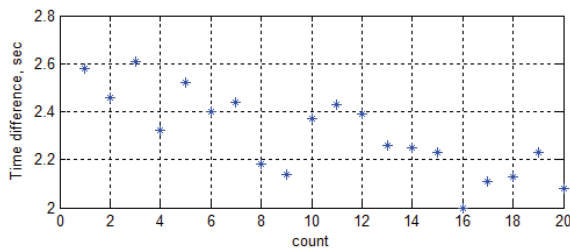


Fig. 6. The differences between time stamps as a function of the counter value

IV. CONCLUSIONS

This paper introduced the possibility of using the Google assistant on a Raspberry Pi microcomputer. Starting from scratch, all the details, software and hardware are presented. Google assistant is available on smartphones, but using it on Raspberry Pi has the advantage that this microcomputer can be interfaced with other hardware devices. We presented a simple example that turns on/off some LEDs by voice commands. However, this application can be extended to any smart home or assistive devices for impaired individuals. Most importantly, this application can work in practice in any language, however, only one at a time. We conducted a large number of tests in Romanian using text strings of

several words and the system responded in real time and with a good accuracy. This system requires internet access, however, this is no longer a problem these days.

In terms of future work, we would like to implement an application that will use the voice command to interact with a DC motor that can be integrated in a smart home device. Also we want to conduct a speech recognizer experiment that does not require internet access. In addition, we would like to incorporate a text to speech function on Raspberry Pi that uses Romanian as its default language.

ACKNOWLEDGMENT

The work of the first author was supported by a grant of the Romanian Ministry of Research and Innovation, CCDCI-UEFISCDI, project number PN-III-P1-1.2-PCCDCI-2017-0917 / contract number 21PCCDCI/2018, within PNCDCI III.

REFERENCES

- [1] P. Milhorat, S. Schogl, G. Chollet et al "Building the Next Generation of Personal Digital Assistants" 1st International Conference on Advanced Technologies for Signal and Image Processing – ATSIP'2014, March 17-19, 2014, Sousse, Tunisia, pp.458–463.
- [2] V. Kepuska and G. Bohouta, "Next Generation of Virtual Personal Assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)", 2018 IEEE 8th Annual Computing and Communication Workshop and Conference 8-10 Jan. 2018 Las Vegas, USA, pp.99–103.
- [3] P. J. Young, J. H. Jin, S. Woo and D. H. Lee, "Bad Voice: Soundless Voice-control Replay Attack on Modern Smartphones", 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna, Austria, pp. 882–887.
- [4] Kaldi Toolkit for Speech Recognition, <http://kaldi-asr.org/index.html>, accessed July 7, 2018.
- [5] Open Source Speech Recognition Toolkit, <https://cmusphinx.github.io/>, accessed July 7, 2018.
- [6] M. Ceaparu, S.A. Toma, S. Segărcănu and I. Gavăt, "Voice-based User Interaction System for Call-Centers, Using a Small Vocabulary for Romanian", The 12th International Conference on Communications, COMM 2018, 14-16 June, 2018, Bucharest, Romania.
- [7] A. Cheng, V. Raghavaraju, J. Janugo et al, "Development and Evaluation of a Healthy Coping Voice Interface Application Using the Google Home for Elderly Patients with Type 2 Diabetes", 2018 15th IEEE Annual Consumer Communication & Networking Conference (CCNC), Las Vegas, USA.
- [8] Chi Zhao, "Text Labeling Applied in Shopping Assistant Robot using Long Short-Term Memory", 2018 International Conference on Intelligent Transportation, Big Data & Smart City, Xiamen, China.
- [9] The MagPi, The official Raspberry Pi magazine, issue 57, May 2017, pp.14-33, raspberrypi.org/magpi, accesed July 7, 2018.
- [10] Google official website of artificial intelligence projects, <https://aiyprojects.withgoogle.com/voice/>, accessed July 7, 2018.
- [11] D. Munteanu and R. Ionel, Voice-Controlled Smart Assitive Device for Visually Impaired Individuals, 2016 12th International Symposium on Electronics and Telecommunications, IESTC 2016, pp. 186-189.
- [12] Google cloud official site, <https://cloud.google.com/speech-to-text/docs/languages>, accessed July 7, 2018.
- [13] F.Weng, P. Angkititrakul, E. Shirberg et al, Conversational In-Vehicle Dialog Systems: The past, present, and future, IEEE Signal Processing Magazine, vol. 33, issue 6, pp. 49-60, 2016.