ISSN (Online): 2455-7838
SJIF Impact Factor : 6.093

EPRA International Journal of

Research & Development

(IJRD)

Monthly Peer Reviewed & Indexed International Online Journal

Volume: 4, Issue:4, April 2019

Published By
EPRA Publishing

SJIF Impact Factor: 6.093 Volume: 4 | Issue: 4 | April | 2019 ISSN: 2455-7838(Online)
EPRA International Journal of Research and Development (IJRD)
Peer Reviewed Journal

SOFTWARE ASSISTANT USING RASPBERRY PI

Mrs.V.S.Kulkarni[1]
Asst.Prof.,Electronics And Telecommunication, WIT, Solapur, India

Akash Dyawarkonda[2]
U.G.Student,Electronics And Telecommunication, WIT, Solapur, India

Bharat Kongari[3]
U.*G.Student,Electronics And Telecommunication, WIT, Solapur, India*

Shweta Girgel[4]
U.G.Student,Electronics And Telecommunication, WIT, Solapur, India

ABSTRACT

This project idea is based on building the Software Assistant using Raspberry pi 3 model. This software assistant is just prototype of the role "Jarvis" in Iron man movie. With this software assistant user can operates it by your voice command. It can perform all user's soft work like checking mail, massages, playing music, etc

I. INTRODUCTION

The advent of software assistants has been an important event in the history of computing. Software assistants are useful for helping the users of a computer system automate tasks and accomplish tasks with minimum human interaction with a machine. The interaction that takes place between a user and a software assistant seems natural; the user communicates using their voice, and the software responds in the same way.If you have seen the movie *Iron Man*, you can perhaps imagine having a software assistant like Tony Stark's Jarvis. Does that idea excite you? The movie inspired me to build my own software assistant software, Melissa. Such a software assistant can serve in the Internet of things as well as run a voice-controlled coffee machine or a voice-controlled drone.

Commercial Software Assistants:
Software assistants are useful for carrying out tasks such as saving notes, telling you the weather, playing music, retrieving information, and much more. Following are some software assistants that are already available in the market:
*Google Now:*Developed by Google for Android and iOS mobile operating systems. It also runs on computer systems with the Google Chrome web browser. The best thing about this software is its voice-recognition ability.
*Siri:*Developed by Apple and runs only on iOS, watchOS, and tvOS. Siri is a very advanced personal assistant with lots of features and capabilities.

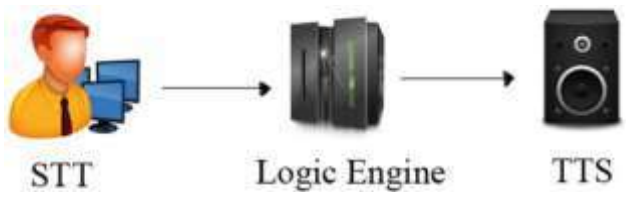
II.REQUIRED COMPNENTS

1) Raspberry pi 3 B+
2) Microphone
3) Speakers
4) 3.5” TFT display

III. BASIC CONCEPT OF ANY SOFTWARE ASSISTANT

For any software assistant three components are required :

1) Speech to Text conversion(STT)
2) Logic Engine
3) Text to Speech conversion(TTS)



A. Speech to Text (STT)

The STT engine converts the user’s speech into a text string that can be processed by the logic engine. This involves recording the user’s voice, capturing the words from the recording (cancelling any noise and fixing distortion in the process), and then using natural language processing (NLP) to convert the recording to a text string.sample that performs the STT conversion. This section discusses that example.

GOOGLE STT

Take a look at this new code snippet:

```
import speech_recognition as
 sr# obtain audio from the microphone
 = sr.Recognizer()with sr.Microphone() as source:
  print("Say something!")
 audio = r.listen(source)
# recognize speech using Google Speech
Recognition
try:
   # for testing purposes, you're just using the
default API key
   # to use another API key, use
`r.recognize_google(audio,
key="GOOGLE_SPEECH_RECOGNITION_API_
KEY")`
   # instead of `r.recognize_google(audio)`
   print("Google Speech Recognition thinks you
said " + r.recognize_google(audio))
except sr.UnknownValueError:
   print("Google Speech Recognition could not
understand audio")
except sr.RequestError as e:
   print("Could not request results from Google
Speech Recognition
```

WIT.AI STT

If you wish to use Wit.ai STT , use this snippet in place of the try/except clause used in the previous code:

```
# recognize speech using Wit.ai
WIT_AI_KEY = "INSERT WIT.AI API
KEY HERE"
try:
  print("Wit.ai thinks you said " +
r.recognize_wit(audio, key=WIT_AI_KEY))
except sr.UnknownValueError:
  print("Wit.ai could not understand audio")
except sr.RequestError as e:
  print("Could not request results from
Wit.ai service; {0}".format(e))
```

While using the Wit.ai service, you have to obtain the Wit.ai key stored in the WIT_AI_KEY constant. You use the r.recognize_wit() function to pass the audio and the key as arguments.

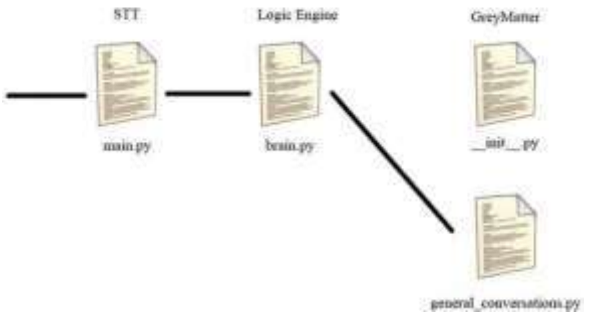
B. Logic Engine :

This is the brain of the system. The Logic engine is based on Artificial Intelligence and Internet of things, using this both of the systems logic engine’s intelligence is depend.

To develop logic engine we used python programming in raspberry pi. . So, you need to have the Python interpreter installed to run the Python code files. *nix systems generally have Python preinstalled.

It handles user queries via a series of if-then-else clauses in the Python programming language. It decides what the output should be in response to specific inputs main.py is the STT engine of your software, and it is also the entry point to your program. You need main.py to direct user queries to its logic engine , which you code in the brain.py file. The brain.py file will contain a ladder of if/else clauses to determine what the user wants to say. If there is a pattern match with one of the statements, brain.py call the corresponding module.

Figure shows the control flow of the program. This will be similar for all the modules you develop for Melissa in future chapters. The difference will be that some other module is called by brain.py instead of general_conversations.py.



C. Text To Speech (TTS) :

This component receives the output from Melissa's logic engine and converts the string to speech to complete the interaction with the user.

For this software assistant we are using the linux inbuilt TTS engine which is espeak

Here is python coding for developing the Text to speech engine,

```
def tts(message):

  """
  This function takes a message as an argument and converts it to speech depending on the OS.
  """
  if sys.platform == 'darwin':

    tts_engine = 'say'
    return os.system(tts_engine + ' ' + message)
  elif sys.platform == 'linux2' or sys.platform == 'linux':
    tts_engine = 'espeak'
    return os.system(tts_engine + ' "' + message + '"')
```

Let's go through the code. The first two import statements import the os and sys modules. Then you define a function called tts that takes a message as an argument. The if statement determines whether the platform is OS X; then it assigns the say value to the tts_engine variable and returns os.system(tts_engine + ' ' + message). This executes the say command with the message on the terminal. Similarly, if the platform is Linux based, it assigns espeak to the tts_engine variable

To test the program, you can add the following additional line at the bottom of the code:

```
tts("Hello world")
```

Save the code, and run the Python file. It should execute successfully.

IV. General Conversation using software assistant

What is time ?

Let's write a new module for telling the time. It will allow you to ask your assistant the time whenever you want.

```
from datetime import datetime
from tts import tts
def what_is_time():
    tts("The time is " +
  datetime.strftime(datetime.now(),
  '%H:%M:%S'))
```

In this code, you first import the datetime module and the datetime function and call datetime.now() in the tts function, in the what_is_time() function. Also note that you format the time using datetime.strftime(), using the format "x hours, y minutes, z seconds."

Repeat what I say

This program performs the same function as the "Repeat What I Say"

```
def main():
  r = sr.Recognizer()
  with sr.Microphone() as source:
    print("Say something!")
    audio = r.listen(source)
  try:
    speech_text =
r.recognize_google(audio).lower().replace("'", "")
    print("Melissa thinks you said '" + speech_text +
"'")
  except sr.UnknownValueError:
    print("Melissa could not understand audio")
  except sr.RequestError as e:
    print("Could not request results from Google
Speech Recognition service; {0}".format(e))
  tts(speech_text)
```

Like this we can built general conversation module using python program. You can build your own module with new ideas like for asking weather, notes taking, Home automation, Music player, etc.

V. CONCLUSION

Software Personal Assistance, which can perform task in offline condition as we given the local modules to Software assistant. In online condition it gets more resources to work with. Also, any peripheral which is connected with the raspberry pi is can be control with the Software Personal Assistance, just by giving the command. The local modules can be added or removed by user as he sees fit. Also, there is simple option for conversation with Software assistant, where it learns further.

This software assistant as product helps to people in their day to day life to live smarter, faster, and enjoyable. Here is the wide future in software assistant as can build your own software assistant and develop it as you want.

REFERENCES

1. *Bor-shen Lin, Hsin-ming Wang, and Lin-shan Lee, "A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history," in Proceeding of the 2nd IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). Citeseer, 1999, vol. 99, p. 4..*

2. *Olivier Lemon and Olivier Pietquin, "Machine learning for spoken dialogue systems," in Proceedings of the 7th Annual Conference of the International Speech Communication Association (Interspeech), 2007, pp. 2685– 2688.*
3. *Reference Book Building a Software Assistant for Raspberry Pi by Tany Pant*
4. *Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefevre, and Olivier Pietquin, "Co-adaptation in spoken ` dialogue systems," in Natural Interaction with Robots,*