# VOICE ASSISTANT MODULE

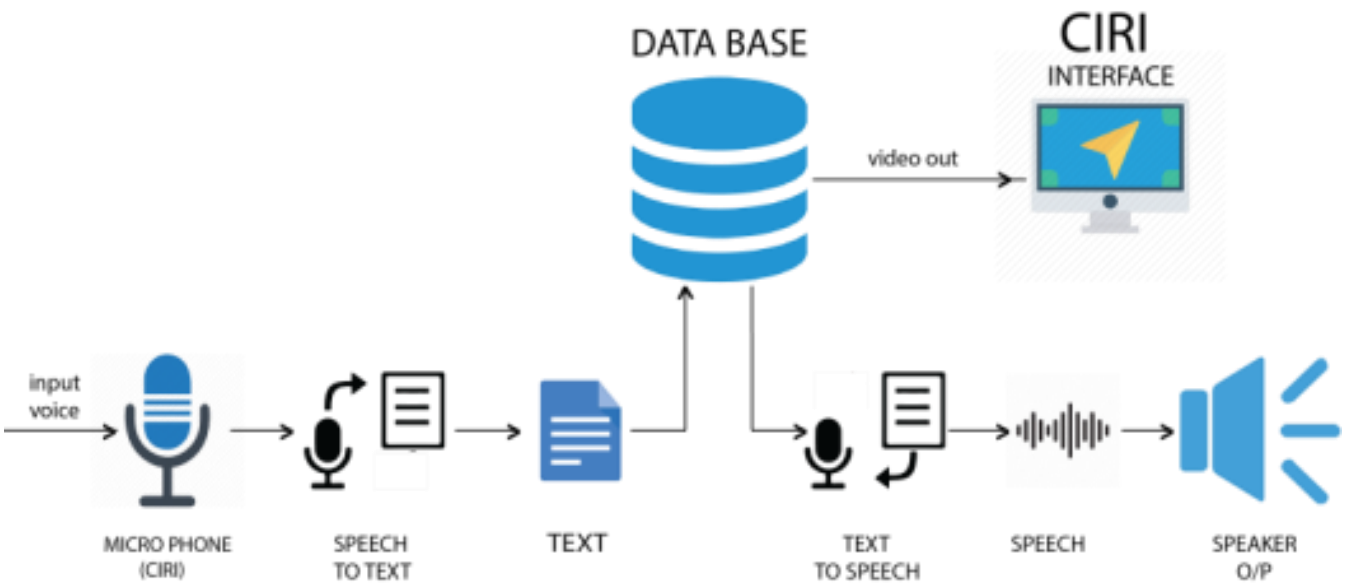


Fig 1. Block Diagram For Voice Assistant

## 1. TEXT TO SPEECH :

Here, for converting the text to voice form we use pyttsx3. pyttsx is a good text to speech conversion library in python but it was written only in python2 until now ! even some fair amount of googling didn't help much to get a tts library compatible with python3. There is however, one library gtts which works perfectly in python3 but it needs internet connection to work since it relies on google to get the audio data. but pyttsx is completely offline and works seemlesly and has multiple tts-engine support.

Pyttsx for python3 [ Offline text to speech for python3]

```
import pyttsx3
```

- ***How to install :***

```
$ pip install pyttsx3
```

➢ ***Fixes for possible errors :***

- No module named win32com.client
- No module named win32
- No module named win32api

```
$ pip install pypiwin32
```

➢ ***Usage :***

```
>> import pyttsx3;
>> engine = pyttsx3.init();
>> engine.say( "I will speak this text");
>> engine.runAndWait() ;
```

➢ ***Included TTS engines :***

- sapi5
- nsss
- espeak

**2. SPEECH TO TEXT :**

Here, Google Speech Recognition API is use by importing the library called speech_recognition.

***How to install :***

The easiest way to install this is using pip install Speech Recognition. Otherwise, download the source distribution from PyPI, and extract the archive. In the folder, run python setup.py install.

Or `$ pip install speech_recognition` in terminal

```
>> import speech_recognition as sr
```

***AudioSource:-***

Base class representing audio sources. Do not instantiate. Instances of subclasses of this class, such as Microphone and WavFile, can be passed to things like recognizer_instance. record and recognizer_instance. listen.

***AudioData:-***

Storage class for audio data. Contains the fields rate and data, which represent the framerate and raw audio samples of the audio data, respectively.

***Library requirement :***

* The first software requirement is Python 2.6, 2.7, or Python 3.3+. This is required to use the library.

* If you want to use the Microphone class (necessary for recording from microphone input), PyAudio is also necessary. If not installed, the library will still work, but Microphone will be undefined.

```
import pyaudio
```

* A FLAC encoder is required to encode the audio data to send to the API. If using Windows or Linux on an i385- compatible architecture, the encoder is already bundled with this library. Otherwise, ensure that you have the flac command line tool, which is often available through one's system package manager.

The following code shows how the speech is recognized using the speech_recognition library and the command is printed which is stored int command variable.

```
import speech_recognition as sr
import pyaudio
r = sr.Recognizer()
with sr.Microphone() as source:
    print("what can i do for you :")
    r.pause_threshold = 1
    r.adjust_for_ambient_noise(source, duration=1)
    print("Listening...")
    audio = r.listen(source)
```

```
command = r.recognize_google(audio).lower()
print('You said : ' + command + '\n')
```

EXAMPLE 1. a : speech is recognized using the speech_recognition library

**3. TRANSLATOR :**

Here, the translator is used for converting Malayalam language to English language and vice versa. The biggest reason to use translate is make translations in a simple way without the need of much effort and can be used as a translation tool like command line. For that translate library is imported using the below command.

```
>> from translate import Translator
```

***How to install :***

```
$ pip install translate
```

***Features :***

- Translate your outputs in real time
- Do translation in terminal using command line.

***Use it :***

```
>> from translate import Translator
>> translator = Translator(from_lang="english", to_lang="malayalam")
>>  translation = translator.translate("text")
```

In the version 3.5.0 sphinx documentation is also added.

**4. AIML ( Artificial Intelligence Markup Language ) :**

Artificial Intelligence Markup Language, is an xml dialect for creating natural language software agents. aiml implements an interpreter for AIML, the Artificial Intelligence Markup Language

developed by Dr. Richard Wallace of the A.L.I.C.E. Foundation. It can be used to implement a conversational AI program. For that aiml library is imported using the below command.

```
import aiml
```

***How to install (aiml 0.9.2) :***

```
$ pip install aiml
```

***What is the use of Aiml?***

AIML stands for Artificial Intelligence Modelling Language. AIML is an XML based markup language meant to create artificial intelligent applications. AIML makes it possible to create human interfaces while keeping the implementation simple to program, easy to understand and highly maintainable.

***Elements of AIML :***

AIML contains several elements. The most important of these are described in further detail below.

***Categories :***

Categories in AIML form the fundamental unit of knowledge. A category consists of at least two further elements: the pattern and template elements. Here is a simple category:

```
<category>
  <pattern>WHAT IS YOUR NAME</pattern>
  <template>My name is CIRI</template>
</category>
```

(When this category is loaded, an AIML bot will respond to the input "What is your name" with the response "My name is CIRI." )

***Patterns :***

A pattern is a string of characters intended to match one or more user inputs. A literal pattern like

WHAT IS YOUR NAME

***Templates :***

A template specifies the response to a matched pattern. A template may be as simple as some literal text, like

My name is CIRI.

```
<aiml version="1.0.1" encoding="UTF-8">
<!-- basic_chat.aiml -->

   <category>
      <pattern>WHAT IS YOUR NAME</pattern>
      <template>
         My name is CIRI.
      </template>
   </category>

</aiml>
```

EXAMPLE 1. b : aiml pattern matching

**5. WEB INTERFACE :**

Here, eel library is imported for developing the **Graphical User Interface (GUI).** eel is a little python library for making simple electron-like offline html/js GUI apps, with full access to python capabilities and libraries. Eel hosts a local webserver, then lets you annotate functions in python so that they can be called from javascript, and vice versa. Eel is designed to take the hassle out of writing short and simple gui applications.

Eel is built on Bottle and Gevent, which provide an asynchronous event loop similar to Javascript. A lot of Python's standard library implicitly assumes there is a single execution thread - to deal with this, Gevent can "monkey patch" many of the standard modules such as time. This monkey patching is done automatically when you call import eel. If you need monkey patching

you should import gevent.monkey and call gevent.monkey.patch_all() before you import eel. Monkey patching can interfere with things like debuggers so should be avoided unless necessary.

***How to install (Eel 0.10.4) :***

```
$ pip install Eel
>> import eel
```

***Use it :***

```
import eel
eel.init('web', allowed_extensions=['.js', '.html'])

@eel.expose       # Expose this function to Javascript
def say_hello_py(x):
    print('Hello from %s' % x)
    engine = pyttsx3.init()
    sound = engine.getProperty('voices')
    engine.setProperty('voice', sound[1].id)
    print("\n\n\tCHOOSE A LANGUAGE")
    engine.say("helo! i'am ciri, please choose a language")
    engine.say("1 for english ,2 for malayalam")
    engine.runAndWait()
eel.start('index.html', size=(1000, 600)) #it will open the window
```

EXAMPLE 1. c :using eel for gui

HTML, CSS, JavaScript and XML are used for the developing the webapp interface.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="description" content="">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>CIRI voice assistent</title>
    <link href="styl123.css" rel="stylesheet">
    <script type="text/javascript" src="/eel.js"></script>
    <script type="text/javascript">
      eel.say_hello_py("Javascript World!"); // Call a Python function
    </script>
```

```
</head>

<body>
  <section class="wellcome_area clearfix" id="home">
    <div class="container h-100">
      <div class="row h-100 align-items-center">
        <div class="col-12 col-md">
          <div class="wellcome-heading">
            <h2>Choose a Language</h2>
            <h3>CIRI</h3>
          </div>
          <div class="get-start-area">
            <form class="form-inline" >
              <input  type="button"  class="submit1" value="1.ENGLISH"
onclick="location.href='hello1.html';">

              <input type="button" class="submit2" value="2.മലയാളം"
onclick="location.href='hello2.html';">

              <input type="button" class="submit3" value="Exit" onclick="window.close()">

</form>
          </div>
        </div>
      </div>
    </div>
  </section>
</body>
</html>
```

EXAMPLE 1. d : HTML for web interface

---

Created By :

B ANANTHU

Email: ananthu3454@gmail.com
Date :  13-07-2019