**Nazwa Projektu:**

Wyrachowana Betty

**Wykonawcy projektu:**

Artur Krepski

Robert Górski

Piotr Zieliński

Wojciech Łosowski

**Zakres Projektu:**

Program służy do wykonywania obliczeń matematycznych, w oparciu wyłącznie o komendy głosowe użytkownika. System ten przeprowadza działania matematyczne objęte zakresem pracy typowego kalkulatora kieszonkowego z wyświetlaczem 8-mio cyfrowym: dodawanie, odejmowanie, mnożenie, dzielenie, pierwiastkowanie stopnia drugiego. Dane do obliczeń podawane są w postaci komend głosowych przez użytkownika. Wynik dokonanych obliczeń, zostaje przekazany użytkkownikowi rónież w formie głosowej.

Program do poprawnego działania wymaga stałego dostępu do internetu.

**Logo projektu:**



**Technologie użyte w projekcie:**

Program zostanie napisany w języku **Python 3+**, ze względu na duży zbiór bibliotek, które są dostępne jako oprogramowanie open source i łatwe w implementacji. oraz na dotychczasową modę programowania w tym języku  (ale pięknie napisane xD)

użyte komponenty oprogramowania:

* SoeechRecognition 3.8.1
* pyAudio
* Google API
* Pysttx
* Tkinter

**Założenia funkcjonalne programu:**

Czyli : co po kolei robi program

Zasady działania programu:

1. przetwarzanie danych wejściowych
2. moduł kalkulatora
3. generowanie danych wyjściowych

coś jak to, tylko trochę więcej…

* opracowanie założeń projektowych
* GUI,
* stworzenie bufora cyklicznego do ustawiania onset-u
* przechwycenie komendy głosowej
* podzielenie nagrania na składowe: liczby i zlecone działania matematyczne
* konwersja składowych
* przeprowadzenie działań matematycznych
* konwertowanie liczby na reprezentacje tekstowe, a potem konwertowanie liczb na ich odpowiedniki słów i tak dalej
* itp

Here is the logic:

1. At first, we will set our microphone device.
2. Accept voice from the user with the mic.
3. Remove noise and distortion from the speech.
4. Convert the speech or voice to text.
5. Now store the text as a string in a variable.
6. Print the string if you wish. ( Not necessary but it will help you determine if the text is all right or not )
7. split the string into three parts:  
   first operand, operator and the second operand
8. Now convert the operands to integers.
9. Finally, do the calculation in your program as you got all the things you need.

**Schemat systemu:**

Na diagramie poniżej, została przedstawiona ogólna budowa programu, jego komponenty i zależności

**Interface aplikacji:**

W programie przewidziano minimalny panel sterowania z poziomu użytkownika, posiadający jedynie podstawowe opcje pracy (Start, Zakończ), oraz ustawień (wielkość okna programu, opcja pracy w tle). Wynika to z podstawowego założenia funkcjonalnego aplikacji, bazującego na sterowaniu pracą programu poprzez wydawanie komend głosowych.

|  |  |  |
| --- | --- | --- |
| **Biblioteka** | **Synteza / przechwytywanie** | **System operacyjny** |
| pyttsx3 | synteza | windows / linux |
| speechSDK | synteza / przechwytywanie | windows / linux |
| RTAudio | Realtime audio i/o C++ classes.  RtAudio provides a common API (Application Programming Interface) for realtime audio input/output across Linux, Macintosh OS X (CoreAudio), and Windows (DirectSound and ASIO) operating systems. | linux/windows |
| PortAudio | It lets you write simple audio programs in 'C' or C++ that will compile and run on many platforms |  |
| Uberi | przechwytywanie |  |
| pyAudio analysis | pyAudioAnalysis can be used to extract audio features, train and apply audio classifiers, segment an audio stream |  |
| Speech Recognition | wykorzystuje google do rozpoznawania mowy - konieczny internet - wszystko zlecane do chmury |  |
| Keras - sieci neuronowe |  |  |
| Wavein |  |  |
| Tkinter | GUI (okienkowe) - nie za trudne | windows/linux |

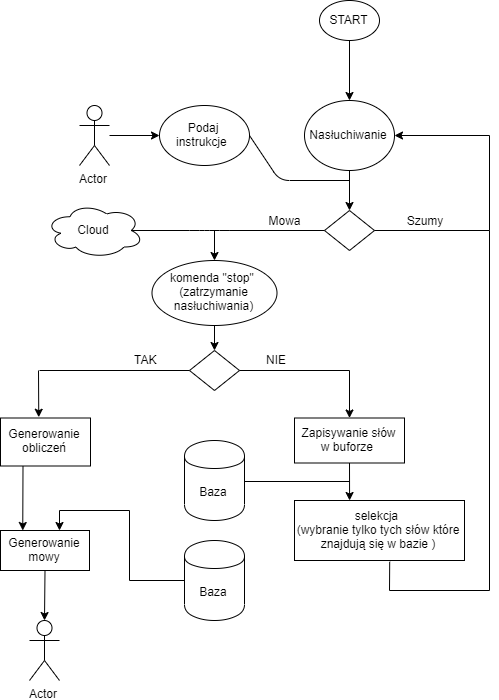
|  |  |  |
| --- | --- | --- |
| pyAudio | Bindings for PortAudio v19, the cross-platform audio input/output stream library. napisane w C  With PyAudio, you can easily use Python to play and record audio streams on a variety of platforms |  |
|  |  |  |

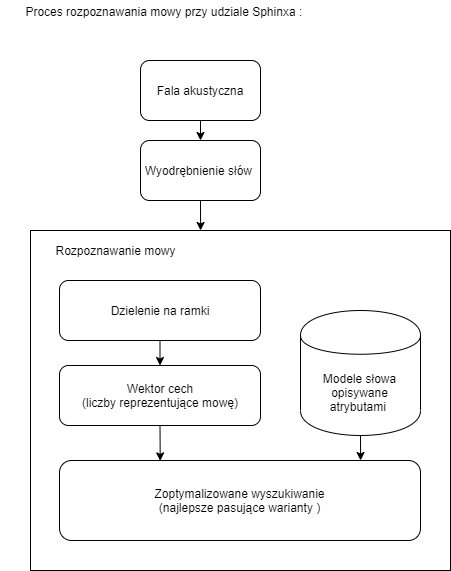
No właśnie jak zacząłem czytać o tym sphinxie to chyba nie jesteśmy w stanie napisać programu który by ogarniał sam tą mowę. A jak mamy go jeszcze samego uczyć to już jest jakaś tragedia. Chyba że to tak opisane po prostu jest bo w kod się nie zagłębiałem

Prace magisterskie na ten temat znalazłem xD

Nie wiem, dodatkowy przedmiot na wypełnienie dziury a może się okazać że to niezłe g….

Piotr





Przechwytywanie i Speech to text - offline

Zastosowana technologia:

Synteza off-line

Biblioteka Text-to-speech **pyttsx3**, python wersja 3+

*Istnieje także starsza wersja biblioteki dla python2*

- zawiera język polski

- czyta poprawnie liczby, np. 10275 jako sto dwa tysiące siedemdziesiąt pięć

- czytanie znaków specjalnych

* , +

Przecinek jest czytany jako „przecinek” nawet w przypadku liczb takich jak 123,52. Znak + jest czytany jako „plus”

* : - .

Znaki te są w całości pomijane

* / \* ^

Odpowiadają one kolejno „slash” „gwiazdka” „zasięg”

W przypadku kiedy synteza następuje po podaniu wyniku występuje tylko znak przecinka. Reszta znaków operacyjnych nie podlega syntezie.

- inicjalizacja

*engine = pyttsx3.init()*

- wybranie głosu polskiego

*voices = engine.getProperty(‘voices’)*

*for voice in voices:*

*if voice.languages[0] == b’\x05pl’:*

*engine.setProperty(‘voice’, voice.id)*

*break*

*b’\x05pl’* jest „numerem” identyfikacyjnym głosu polskiego w bibliotece – w pętli for można dodać komendę „print” wraz z parametrem „voice”. W ten sposób wyświetlimy całą listę głosów wraz z ich parametrami, m.in. ID.

- istnieje możliwość dostosowywania głośności, tempa jak i przerwania czytania. Ostatnie może być szczególnie przydatne w przypadku, gdy podczas czytania wyniku użytkownik zacznie kontynuować interakcję głosową.

- sama funkcja czytania sprowadza się do wpisania w miejsce znaku *X* stringa, który chcemy poddać syntezie:

*engine.say(‘X’)*

- biblioteka posiada dodatkowe funkcjonalności, które także mogą okazać się przydatne. Pełna dokumentacja znajduje się na poniższej stronie:

[*https://pyttsx3.readthedocs.io/en/latest/index.html*](https://pyttsx3.readthedocs.io/en/latest/index.html)

Modele w Sphinx

Zgodnie ze strukturą mowy do dopasowania pasują trzy modele:

* **Model akustyczny** -zawiera właściwości akustyczne
* **Fonetyczny -** słownik zawiera mapowanie od słów
* **Modelu językowy -** służy do ograniczenia wyszukiwania słów. Określa, które słowo może następować po wcześniej rozpoznanych słowach i pomaga znacznie ograniczyć proces dopasowywania, usuwając słowa, które nie są prawdopodobne.

Wykorzystane CMU:

* Sphinx4 - modyfikowany program napisany w Javie
* Sphinxtaint - narzędzie do szkolenia modeli akustycznych