МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное
учреждение высшего образования

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовой проект по курсу

«Операционные системы»

III семестр

Группа: М80 – 207Б-18
Студент: Цапков Александр Максимович
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____

Москва, 2019.

Содержание

Введение

Для данного курсового проекта по курсу операционных сисем я решил взять следующую тему: “Создание и настройка Телеграмм бота. HHTP интерфейсы”. Данную тему я выбрал по 2-м причинам: для начала я хотел написать что-то полезное, чем можно было бы и хотелось пользоваться самому. По причине вовлеченности меня в эту тему я делал работу качественнее, чем если бы тема была для меня скучна. Второй же причиной является то, что мне хотелось бы разобраться хотя бы в общих чертах в том как работают HTTP интерфейсы, так как это с огромной вероятностью пригодится мне в будущем. Я уже сталкивался с незнанием основ в этой теме, что вышло мне боком, поэтому целью для себя в этом курсовом проекте я поставил понять прицепы работы HTTP интерфейсов для дальнейшего мною их применения.

Осталось придумать что же должен делать мой бот. Долго думать мне не пришлось. В свободное от учебы время я часто играю на музыкальных инструментах и мне часть приходится настраивать свою гитару. Для этого хорошо подходят скачиваемые на телефон тюнеры. Но иногда бывает, что своего телефона под рукой нет, а устанавливать тюнер на чужой не хочется. Да и вообще хотелось бы кросплатформенное и легкое решение. Тут то и родилась моя идея — телеграмм бот тюнер. Идея проста: мы просто открываем бота, присылаем ему голосовым сообщением ноты с нашего инструмента, и он нам их называет и говорит куда их подстроить. Кроме того хотелось бы добавит еще немного полезного для настройки функционала, например конвертер, который может сказать частоту любой ноты и назвать ноту по частоте. Именно такого бота я и собираюсь написать.

Подготовка и сбор информации

Для начала нужно разобраться с основами, то есть: на каком я буду писать языке и что мне понадобится (библиотеки, документация, какие либо дополнительные средства).

Начнем с самого начала. Для того чтобы создать телеграмм нужно посетить официальный сайт телеграмма и узнать о том как же происходит создание и управление нашим ботом. А делается это с помощью HTTP интерфейса, то есть API. После регистрации нашего бота у… другого бота, общение с ним происходит через http запросы. То есть для того чтобы , к примеру отправить пользователю сообщение мы должны сделать запрос в виде:

https://api.telegram.org/bot<token>/НАЗВАНИЕ_МЕТОДА

Подробнее об Telegram BotAPI я расскажу позднее.

Теперь определимся с языком программирования. Его нужно выбирать по функциональности, количеству предоставляемых возможностей, и простоте написания кода. Мой выбор пал на высокоуровневый язык программирования Python. Данный выбор я сделал по причине того, что для этого языка написано множество полезных библиотек, но главное, что на нем есть замечательная библиотека pyTelegramBotAPI, которая инкапсулирует HTTP интерфейс для работы с ботом и позволяет сосредоточится на логике программы. Но основным для меня преимуществом является наличие так называемых хендлеров, что заметно упрощает работу с ботом в целом.

Теперь по задаче. Для того чтобы сделать тюнер нам нужно работать со звуком, а в частности конвертировать форматы и находит базовую частоту сэмплов. Для конвертации мы используем библиотеку ffmpeg и ее обертку для python — pydub. Для поиска базовой частоты звуковых фрагментов можно вручную применить, к примеру, трансформацию Фурие, но и для такой задачи есть библиотека для python — aubio.

И теперь, когда мы подготовили базу, можно приступать к работе.

TelegramBotAPI

Давайте поподробнее разберемся с TelegramBotAPI и регистрацией бота. Бот телеграмм по сути своей является лишь интерфейсом для работы с вашим сервисом.Сам бот находится на серверах телеграмма и только принимает запросы от пользователя и передает их ему. Но обрабатывать все эти запросы уже должен ваш сервер, где и будет находиться все логика бота.

Для начала нужно разобраться с самым основным — как получить и отправить какое либо сообщение. С отправкой все относительно просто, мы должны послать серверу телеграмма запрос, чтобы наш бот отправил сообщение, или сделал какое либо иное действие. Запрос этот отправляется по протоколу HTTP (поэтому это и называется HHTP интерфейс) на сервера телеграмм с уникальным идентификатором нашего бота, который конфиденциален, так как с помощью его можно управлять нашим ботом. Вид такого HTTP запроса следующий:

https://api.telegram.org/bot<token>/НАЗВАНИЕ_МЕТОДА

token — это и есть это уникальный идентификатор нашего бота, который мы получает при его регистрации. Регистрация бота и получение этого токена происходит через другого телеграмм бота @BotFather. Через него же происходит и администрирование нашего бота, но вернемся к HTTP интерфейсам. Ответ придёт в виде JSON-объекта, в котором всегда будет булево поле ok и опциональное строковое поле description, содержащее человекочитаемое описание результата.

С посланием запроса разобрались, но как же получить информацию о том что пришел запрос нам (к примеру сообщение)? Для этого есть 2 пути, у каждого из которых есть свои плюсы. Все обновления (действия с ботом со стороны пользователей) на сервере будут храниться 24 часа и нам нужно их получить.

Так вот: первый способ— это getUpdates. Этот метод используется для получения обновлений через long polling, когда мы отправляем запрос

серверу на получение обновления, а сервер, если обновлений не имеется, вместо того чтобы вернуть нам пустой ответ ждет пока обновление не появится, не завершая таким образом запрос ответом. Когда обновление все же происходит сервер таки отвечает на тот запрос, завершая цикл. Это позволяет избавится от задержки периодически запросах об обновлениях и снижает нагрузку на трафик. Это метод очень просто в реализации на допашем компостере, но у него есть один минус. Иногда сервера телеграмма начинают выдавать ошибку на такие getUpdates запросы. Это привалит к полной потери работоспособности вашего бота.

Второй же способ это webHook и метод setWebhook. Веб хук это такой сервер, который будет получать HTTPS POST запросы напрямую от сервером телеграмма при получении ими обновления. Чтобы настроить вебхук, нужно воспользоваться методом setWebhook и передать телеграмму URL вашего webHook сервера. Необязательным параметром также является бубличный ключ подписи, так как телеграмм работает только с подписанными серверами (также и с самоподписанными).

Я же пока выбрал первый способ, просто потому что он легче, но в будущем, если я буду планировать развивать своих ботов, мне придется настроить webHook для большей надежности и отказоустойчивостьи.

pyTelegramBotAPI

Немного о библиотеке для python telebot (pyTelegramBotAPI). Она инкапсулирует саму работу с HTTP интерфейсом, но при этом сохраняет полный функционал TelegramBotAPI. Самое удобное в этой библиотеке это хэндлеры. С их помощью можно создать несколько фильтров сообщений и каждый хэндлер будет вызывать свою функцию при получение сообщения с сервера, и если это сообщение будет удовлетворять неким критериям.

Процесс написания программы

Теперь наконец приступим к написанию программы. Начнем с настройки соединения с серверами телеграмм. Дело в том, что ip адреса телеграмм заблокированы в России. Сам месседжер использует встречную настройку переадрессации и обходит эту блокировку. Но вот программа которую мы пишем просто отправляет HTTP запросы, которые блокируются в России. Для преодоления этой трудности нужно отправлять запросы через proxy server, благо TelegramBotAPI, используя библиотеку requests позволяет это сделать:

```
from telebot import apihelper

ip = 'orbtl.s5.opennetwork.cc'
port = '999'
usrP = '214931371'
passP = 'xxxxxxxx'

proxy = {'https': 'socks5h://{}:{}@{}:{}'.format(usrP, passP,
ip,port)}
apihelper.proxy = proxy
```

После этого нужно создать экземпляр класса TeleBot по своему токену

```
TOKEN = ‘99404744:ASS678fhhHH8oCRGnxm-hhf8hfu3bf84n81cO'

bot = telebot.TeleBot(TOKEN)
```

После этого осталось сделать хендлеры для разных команд и типов сообщений. Самое главное для нас это хендлер аудиосообщений. При обработки аудиосообщений нам нужно их скачать с сервера. В классе message мы получаем только id файла, который нужно также сказать по HTTP запросу (используя все тот же proxy). Все аудиосообщения в телеграмме хранятся в формате .ogg. Этот формат не поддерживается библиотекой по анализу сэмплов, которая нам нужна, так что мы конвертируем наш файл в .wav:

```
@bot.message_handler(content_types=['voice'])
def analyse_audio(message):
    audio_path = bot.get_file(message.voice.file_id).file_path
    request = requests.get('https://api.telegram.org/file/bot{}/
                {}'.format(TOKEN, audio_path), proxies=proxy)

    my_file = open("UsersAudio/test.ogg", "wb")
    my_file.write(request.content)
    ogg_version = AudioSegment.from_ogg("UsersAudio/test.ogg")
    ogg_version.export("UsersAudio/test.wav", format="wav")
```

Теперь открываем получившийся .wav файл библиотекой aubio. Дальше в цикле разбиваем его на сэмплы и получаем базовую частоту этих сэмплов. После этого берем по 5 сэмплов и находим медиану (это я сделал для уменьшения случайных помех). Все частоты этих чанков по 5 сэмплов добавляются в список для дальнейшей обработки.

```
pDetection = aubio.pitch("yin", 2048, 2048, src.samplerate)
    pDetection.set_unit("Hz")
    pDetection.set_silence(-60)

    buf_count = 0
    buf_freak = [ 0 for i in range(5)]
    all_sl = []
    res = ''

    while True:
        samples, read = src()
        buf_freak[buf_count] = (pDetection(samples)[0])

        if buf_count == 4:
            all_sl.append(median(buf_freak))
            buf_count = 0
        else:
            buf_count += 1

        if read < src.hop_size:
            break
```

В этом списке требуются найти продолжительные участи с одной частотой. Для этого мы смотрим на текущую частоту сэмпла с сравниваем с от части усреднённой частотой предыдущих. При не вхождении за определённый диапазон увеличивается счетчик длинны. Если длина такой последовательности больше 4-х, то мы из полученной частоты вычисляем ближайшую ноту и то насколько частота недотягивает до этой ноты.

```
curr_count = 0
    curr_f = 0
    for slise in all_sl:
        print(slise)
        if abs(curr_f - slise) < (curr_f * (2**(1/12)) -
                                                curr_f):
            if curr_count == 0:
                curr_f = slise
            else:
                curr_f = (curr_f + slise) / 2
            curr_count += 1
        elif curr_count > 4:
            n = n_of_note(curr_f)
            note = n_to_note(n)
            res += note + ': ' + offset(curr_f, n) + '\n'

            curr_count = 0
            curr_f = slise
        else:
            curr_count = 0
            curr_f = slise

    if curr_count > 4:
        n = n_of_note(curr_f)
        note = n_to_note(n)
        res += note + ': ' + offset(curr_f, n) + ‘\n'
```

После получения всех частот и нот мы просто проверяем нашли ли мы хоть что-то и отправляем запрос на отправку сообщения на сервер телеграмма через TelegramBotAPI методом класса, объект которого мы создали в самом начале программы.

Остальной код это вычисление частоты нот и наоборот. Работа же с сообщениям API почти такая же, поэтому я не буду заострять на этом внимание. Единственно скажу, что для того чтобы определить что пользователь отправил мне сообщением ноту, я использовал регулярные выражения.

Листинг программы

```
import requests
from pydub import AudioSegment
import telebot
from telebot import apihelper
import numpy as np
import aubio
import re
import math as m

BASE_NOTE = 440 / ((2**(1/12))**48)

def note_to_n(n, octave):
    if n == 'A':
        in_oc = 12
    elif n == 'A#' or n == 'Bb':
        in_oc = 13
    elif n == 'B':
        in_oc = 14
    elif n == 'C':
        in_oc = 3
    elif n == 'C#' or n == 'Db':
        in_oc = 4
    elif n == 'D':
        in_oc = 5
    elif n == 'D#' or n == 'Eb':
        in_oc = 6
    elif n == 'E':
        in_oc = 7
    elif n == 'F':
        in_oc = 8
    elif n == 'F#' or n == 'Gb':
        in_oc = 9
    elif n == 'G':
        in_oc = 10
    elif n == 'G#' or n == 'Ab':
        in_oc = 11
    else:
        return 0
    return BASE_NOTE * ((2**(1/12))**(in_oc + 12 * (octave)))


def oct_note(n):
    if n == 0:
        return 'A'
    elif n == 1:
        return 'A#'
    elif n == 2:
        return 'B'
    elif n == 3:
        return 'C'
    elif n == 4:
        return 'C#'
    elif n == 5:
        return 'D'
    elif n == 6:
        return 'D#'
    elif n == 7:
        return 'E'
    elif n == 8:
```

```
        return 'F'
    elif n == 9:
        return 'F#'
    elif n == 10:
        return 'G'
    elif n == 11:
        return 'G#'

def n_of_note(freq):
    return m.ceil(12 * m.log2(freq / BASE_NOTE) - 0.5)
def n_to_note(n):
    return oct_note(n % 12) + str((n + 9) // 12 - 1)

def median(lst):
    quotient, remainder = divmod(len(lst), 2)
    if remainder:
        return sorted(lst)[quotient]
    return sum(sorted(lst)[quotient - 1:quotient + 1]) / 2.

def offset(freq, note):
    ideale_freq = BASE_NOTE * ((2**(1/12))**note)
    if abs(freq - ideale_freq) < ((ideale_freq * (2**(1/12))) -
ideale_freq) * 0.1:
        return 'OK'
    elif freq - ideale_freq < 0 :
        return '-' + str(abs(m.floor(freq - ideale_freq)))
    else:
        return '+' + str(abs(m.floor(freq - ideale_freq)))


def help():
    res = "\t-I can tune your instument, just play me via voice
message\n"
    res += "\t-Tell of a different tunes, simply type your
instrument\n"
    res += "\t-Convert frequency to note and vice versa, just type
some!\n"
    return res




ip = 'orbtl.s5.opennetwork.cc'
port = '999'
usrP = '214931371'
passP = 'xxxxxxxx'

proxy = {'https': 'socks5h://{}:{}@{}:{}'.format(usrP, passP,
ip,port)}
apihelper.proxy = proxy

TOKEN = '99404744:ASS678fhhHH8oCRGnxm-hhf8hfu3bf84n81cO'

bot = telebot.TeleBot(TOKEN)

@bot.message_handler(commands=['start'])
def send_welcome(message):
    res = "Hello and welcome. If you are a musician your on the right
way!\n"
    res += help()
    bot.send_message(message.chat.id, res)
```

```
@bot.message_handler(commands=['help'])
def send_help(message):
    res = help()
    bot.send_message(message.chat.id, res)

@bot.message_handler(content_types=['voice'])
def analyse_audio(message):
    audio_path = bot.get_file(message.voice.file_id).file_path
    request = requests.get('https://api.telegram.org/file/bot{}/
{}'.format(TOKEN, audio_path), proxies=proxy)

    my_file = open("UsersAudio/test.ogg", "wb")
    my_file.write(request.content)

    ogg_version = AudioSegment.from_ogg("UsersAudio/test.ogg")
    ogg_version.export("UsersAudio/test.wav", format="wav")

    src = aubio.source('UsersAudio/test.wav', hop_size=2048)

    pDetection = aubio.pitch("yin", 2048, 2048, src.samplerate)
    pDetection.set_unit("Hz")
    pDetection.set_silence(-60)

    buf_count = 0
    buf_freak = [ 0 for i in range(5)]
    all_sl = []
    res = ''

    while True:
        samples, read = src()
        buf_freak[buf_count] = (pDetection(samples)[0])

        if buf_count == 4:
            all_sl.append(median(buf_freak))
            buf_count = 0
        else:
            buf_count += 1

        if read < src.hop_size:
            break


    curr_count = 0
    curr_f = 0
    for slise in all_sl:
        print(slise)
        if abs(curr_f - slise) < (curr_f * (2**(1/12)) - curr_f):
            if curr_count == 0:
                curr_f = slise
            else:
                curr_f = (curr_f + slise) / 2
            curr_count += 1
        elif curr_count > 4:
            n = n_of_note(curr_f)
            note = n_to_note(n)
            res += note + ': ' + offset(curr_f, n) + '\n'

            curr_count = 0
            curr_f = slise
        else:
            curr_count = 0
```

```
            curr_f = slise

    if curr_count > 4:
        n = n_of_note(curr_f)
        note = n_to_note(n)
        res += note + ': ' + offset(curr_f, n) + '\n'

    if res == '':
        answer = "I can't quite hear you."
    else:
        answer = "You playd:\n" + res

    #print(total_read)
    bot.reply_to(message, answer)

@bot.message_handler(regexp=r'^Guitar$')
def guitar_tune(message):
    res = "Here's the standart tune for a guitar:\n"
    res += "1-E3 2-B2 3-G2\n4-D2 5-A1 6-E1\n"
    bot.reply_to(message, res)

@bot.message_handler(regexp=r'^Piano$')
def piano_tune(message):
    res = "There's no other tune for piano.\nBut here's the tip: you
can start your tuning from A3:\n"
    bot.reply_to(message, res)

@bot.message_handler(regexp=r'^\d+(\.)?\d* ?(Hz|HZ|hz)?$')
def freq_to_note(message):
    freq = float(re.search(r'^\d+(\.)?\d*', message.text)[0])
    n = n_of_note(freq)
    note = n_to_note(n)
    off = offset(freq, n)
    if off == 'OK':
        off = ''
    else:
        off += " Hz"
    res = note + ' ' + off + '\n'
    bot.reply_to(message, res)

@bot.message_handler(regexp=r'^[ABCDEFG][#b]?\d$')
def note_to_freq(message):
    note = re.search(r'[ABCDEFG][#b]?', message.text)
    octave = re.search(r'\d$', message.text)
    if (not note) or (not octave):
        bot.reply_to(message, "Write in capitale!")
        return
    n = note_to_n(note[0], int(octave[0]))
    answer = str(m.floor(n)) + " Hz"
    bot.reply_to(message, answer)

@bot.message_handler(func=lambda message: True)
def echo_all(message):
    bot.reply_to(message, "I can't undestend you... Мяу:3")

bot.polling()
```

Вывод

Завершив данный курсовой проект я получил хорошо работающего и достаточно полезного телеграмм бота, которым я бы мог и сам пользоваться. Это не отняло у меня очень много времени, и оставило положительные впечатления. Хорошо написанная API не доставила мне никаких неудобств, а все проблемы решались 5-и минутным просмотром документации. И вся эта мощь доступна через обычные HTTP запросы. У этого подхода есть один огромный плюс. И это огромная кроссплатформенность и многоязычность того, на чем вы хотите написать бота. Все что нудно для работы с API это отправлять HTTP запросы, а интернет в наше время есть даже у калькуляторов. Я пользовался python’ом, потому что мне так было удобнее, но если бы я гнался за производительностью, то я бы мог сделать тоже самое на C++ и не сильно потерять в удобстве.

Список литературы

1. Справочник по Bot API [Электронный ресурс]. URL: https://core.telegram.org/bots/api
2. Документауия pyTelegramBotAPI [Электронный ресурс]. URL: https://github.com/eternnoir/pyTelegramBotAPI
3. Python aubio documentation [Электронный ресурс]. URL: https://aubio.org/manual/latest/python.html#python