

**МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»  
(НГТУ)**

Институт радиоэлектроники и информационных технологий  
Направление подготовки (специальность) 09.03.01 Информатика и вычислительная техника  
Направленность (профиль) образовательной программы Вычислительные машины, комплексы, системы, сети  
Кафедра Вычислительные системы и технологии

## ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

бакалавра

Студента Стеклова Сергея Андреевича группы 16-В-2

на тему «Программная система голосового управления мобильного робота»

**СТУДЕНТ:**



(подпись)

Стеков С.А.

(фамилия, и., о.)

«3» июля 2020г.

(дата)

**РУКОВОДИТЕЛЬ:**



(подпись)

Гай В.Е.

(фамилия, и., о.)

«3» июля 2020г.

(дата)

**РЕЦЕНЗЕНТ:**

\_\_\_\_\_

(подпись)

\_\_\_\_\_

(фамилия, и., о.)

\_\_\_\_\_

(дата)

**ЗАВЕДУЮЩИЙ КАФЕДРОЙ**



(подпись)

Жевнерчук Д.В.

(фамилия, и.о.)

«3» июля 2020г.

(дата)

**КОНСУЛЬТАНТЫ:**

1. По \_\_\_\_\_

\_\_\_\_\_

(подпись)

\_\_\_\_\_

(фамилия, и., о.)

\_\_\_\_\_

(дата)

2. По \_\_\_\_\_

\_\_\_\_\_

(подпись)

\_\_\_\_\_

(фамилия, и., о.)

\_\_\_\_\_

(дата)

3. По \_\_\_\_\_

\_\_\_\_\_

(подпись)

\_\_\_\_\_

(фамилия, и., о.)

\_\_\_\_\_

(дата)

ВКР защищена \_\_\_\_\_

(дата)


протокол № \_\_\_\_\_

с оценкой \_\_\_\_\_

**МИНОБРНАУКИ РОССИИ**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**  
**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  
**«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**  
**УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»**  
**(НГТУ)**

Кафедра «Вычислительные системы и технологии»

УТВЕРЖДАЮ

 Зав. кафедрой ВСТ  
Жевнерчук Д.В.,  
«12» мая 2020 г.

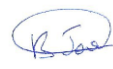
**ЗАДАНИЕ**  
**на выполнение выпускной квалификационной работы**

**по направлению подготовки (специальности) 09.03.01 Информатика и вычислительная техника студенту Стеклову С.А. группы 16-В-2**  
(Ф.И.О.)

1. Тема «Программная система голосового управления мобильного робота»  
(утверждена приказом по вузу от № 867/5 от 15.04.2020)
2. Срок сдачи студентом законченной работы 3 июля 2020г
3. Исходные данные к работе описание робота ELCBot
4. Содержание расчетно-пояснительной записки  
Введение  
1. Техническое задание  
2. Анализ технического задания  
3. Разработка системы голосового управления мобильного робота  
4. Разработка программных средств  
5. Тестирование системы  
Заключение  
Список литературы
5. Перечень графического материала (с точным указанием обязательных чертежей)  
Презентация (Цель и задачи исследования, Принцип работы голосовых интерфейсов, Обзор систем распознавания речи, Работа с Google Cloud Speech, Варианты распознавания речи, Протокол взаимодействия с роботом, Схема работы программы, Тестирование работы системы)
6. Консультанты по ВКР (с указанием относящихся к ним разделов)  
Нормоконтроль Жевнерчук Д.В.

7. Дата выдачи задания «13» апреля 2020г.

Код и содержание Компетенции	Задание	Проектируемый результат	Отметка о выполнении
ПК-1 Способность разрабатывать модели компонентов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно вычислительная машина"	Разработка системы взаимодействия человека с мобильным роботом посредством системы голосового управления при помощи микрофона	Возможность голосового управления мобильным роботом	Выполнено
ПК-2 способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	Разработка системы голосового управления мобильного робота	Система голосового управления мобильного робота	Выполнено
ПК-3 способностью обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности	Реализация системы голосового управления мобильного робота. Тестирование готовой системы	Система голосового управления реализована и протестирована.	Выполнено

Руководитель  Гай В.Е.  
(подпись)

Задание принял к исполнению «13» апреля 2020г.  
(дата)

Студент  Стеклов С.А.  
(подпись)

**МИНОБРНАУКИ РОССИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»  
(НГТУ)**

**АННОТАЦИЯ**

**к выпускной квалификационной работе**

**по направлению подготовки (специальности) 09.03.01 Информатика и  
вычислительная техника студента Стеклова С.А. группы 16-В-2  
по теме «Программная система голосового управления мобильного робота»**

Выпускная квалификационная работа выполнена на 30 страницах, 12 рисунков,  
библиографический список из 9 источников, 1 приложение (дополнительно 5 страниц).

Актуальность: Выбранная задача актуальна и предоставляет возможности для безопасного  
и более простого внедрения информационных технологий в массы.

Объект исследования: робот ELCBot.

Предмет исследования: голосовое управление мобильного робота.

Цель исследования: разработка программной реализации система голосового управления  
мобильного робота.

Задачи исследования: выбрать систему распознавания речи и систему синтеза речи,  
разработать схему работы системы голосового управления мобильного робота, написать  
программный алгоритм системы голосового управления мобильного робота.

Методы исследования: Теоретический анализ, системный подход, эксперимент .

Структура работы:

Во введении рассказывается о цели работы и ее актуальности.

В 1 разделе рассмотрены требования к ЭВМ, системе голосового управления.

Во 2 разделе анализируется техническое задание, происходит выбор систем распознавания  
речи и языка программирования.

В 3 разделе разрабатывается система голосового управления мобильного робота.

В 4 разделе реализуется система голосового управления мобильного робота.

В 5 разделе производится тестирование системы.

В заключении проанализирована проделанная работа и сделаны выводы.

Выводы:

1. Разработана система голосового управления мобильного робота.
2. Реализована система голосового управления мобильного робота

.

Рекомендации:

1. Дальнейшее развитие проекта.
2. Модернизирование существующего алгоритма системы голосового управления.







/ Стеклов С.А.

подпись студента /расшифровка подписи

«3» июля 2020 г.

## Оглавление

Введение.....	6
1 Техническое задание .....	7
1.1 Назначение разработки и область приенения .....	7
1.2 Технические требования .....	7
2 Анализ технического задания.....	8
2.1 Выбор операционной системы.....	8
2.2 Выбор языка программирования .....	10
2.3 Основные элементы системы.....	11
2.3.1 Система распознавания речи .....	11
2.3.2 Событийная система обработки данных. ....	11
2.3.3 Система синтеза речи.....	11
2.4 Обзор систем распознавания речи.....	12
2.4.1 CMU Sphinx .....	12
2.4.2 Kaldi .....	12
2.4.3 Google Cloud Speech API.....	13
2.4.4 Yandex SpeechKit.....	13
2.4.5 Финальный выбор системы.....	13
3 Разработка системы голосового управления мобильного робота .....	14
3.1 Разработка общей структуры системы .....	14
3.2 Протокол взаимодействия с роботом .....	15
4 Разработка программных средств .....	18
4.1 Работа с Google Cloud Speech .....	18
4.1.1 Регистрация .....	18
4.1.2 Установка.....	19
4.1.3 Варианты взаимодействия .....	19
4.2 Разработка программной структуры системы.....	20
4.2.1 Элементы системы .....	20
4.2.2 Схема работы программы .....	21
4.3 Программная реализация модулей системы .....	23
4.3.1 Реализация MicrophoneStream .....	23

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)				
Изм.	Лист	№ докум.	Подпись	Дата	Разработка системы голосового управления мобильного робота	Лит.	Лист	Листов	
Провер.		Гай В.Е.		03.07					
Разраб.		Стеклов С.А.		03.07			4	33	
Н. Контр.		.		03.07		НГТУ им. Р.Е. Алексеева			
Утверд.		Жевнурчук Д.В.		03.07					

4.3.2	Реализация main .....	25
4.3.3	Реализация listen_loop .....	26
4.3.4	Реализация process_command.....	27
4.3.5	Реализация text_to_speech .....	27
5	Тестирование системы.....	29
	Заключение.....	32
	Список литературы .....	33

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

## Введение

Голосовое управление — способ взаимодействия с устройством при помощи голоса. Считается, что это следующая стадия управления техникой, после сенсорного ввода информации.

Создание систем автоматического распознавания речи достаточно актуальное направление в сфере развития информационных технологий. Мало кто задумывается о том, на сколько эти технологии уже распространены в нашей жизни. Но многие разработчики считают, что эра помощников с голосовым управлением уже не за горами.

В связи с этим возникла идея создания программной системы голосового управления. Такая система могла бы найти себе множество применений, как в робототехнике, так и вне ее. Распознавание речи найдет применение как способ управления роботами в виртуальной реальности (VR), приборах, играх, инструментах и компьютерах. Данная технология обладает в долгосрочной перспективе очень хорошим потенциалом, поэтому компании развивают методы распознавания речи. Возможность управлять и отдавать команды компьютеру (или прибору) непосредственно голосом сделает процесс управления таким устройством гораздо более простым, эффективным и удобным. Такой тип управления голосом в своей основе позволит пользователю осуществлять параллельно и другие операции (т. е. при голосовой работе с компьютером или прибором глаза и руки остаются «свободными» для другой работы).

Таким образом, целью данной работы является создание программной системы голосового управления мобильного робота.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		



## **1 Техническое задание**

### **1.1 Назначение разработки и область применения**

Разрабатываемая программная система предназначена для осуществления голосового управления мобильного робота.

Область применения разрабатываемой системы:

- Интеграция с мобильного робота ELC Bot
- Интеграция с любой системой, подразумевающей командное управление

### **1.2 Технические требования**

Требования, предъявляемые системой к ЭВМ:

- Операционная система Windows, начиная с версии 7 и выше, Linux или Mac OS X
- Доступ к сети интернет
- Устройства ввода: клавиатура, микрофон
- Устройства вывода: монитор, динамики

Рассмотрим, каким функционалом должна обладать разрабатываемая система голосового управления:

- В системе должна быть реализация системы распознавания речи [1], что подразумевает преобразование аудио-данных, поступающих из микрофона в текст.
- В системе должна быть реализация системы синтеза речи [2], что подразумевает преобразование текста в аудио-данные, понятные человеку.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

## 2 Анализ технического задания

### 2.1 Выбор операционной системы

Одной из важнейших задач при подготовке к разработке программной системы является выбор операционной системы, под управлением которой будет происходить сама разработка.

Для решения этой задачи необходимо произвести анализ существующих и доступных на сегодняшний день операционных систем для персональных компьютеров и ноутбуков, каждая из которых обладает своими уникальными свойствами, которые могут оказать влияние на процесс разработки, отладки и тестирования. Результатом этого анализа должно быть определение операционной системы, в наибольшей степени удовлетворяющей требованиям, предъявляемым к разрабатываемой системе.

Рассмотрим наиболее популярные на данный момент ОС:

- Windows - семейство коммерческих операционных систем (ОС) корпорации Microsoft, ориентированных на управление с помощью графического интерфейса. Изначально Windows была всего лишь графической программой-надстройкой для распространенной в 1980-х и 1990-х годах операционной системы MS-DOS.
- Mac OS - семейство проприетарных операционных систем производства корпорации Apple. Разработана для линейки персональных компьютеров Macintosh. Популяризация графического интерфейса пользователя в современных операционных системах часто считается заслугой Mac OS.
- Linux - семейство Unix-подобных операционных систем на базе ядра Linux, включающих тот или иной набор утилит и программ проекта GNU, и, возможно, другие компоненты. Как и ядро Linux, системы на его основе как правило создаются и распространяются в соответствии с моделью разработки свободного и открытого программного обеспечения. Linux-системы распространяются в основном бесплатно в виде различных дистрибутивов — в форме, готовой для установки и удобной для сопровождения и обновлений, — и имеющих свой набор системных и прикладных компонентов, как свободных, так, возможно, и собственных.

Теперь сравним предоставленные ОС:

- Удобность использования. Дело привычки, однако если начинать работать с Windows, то переход на Linux подобные ОС будут вызывать дискомфорт. Linux - система для продвинутых пользователей. Настройки доступа, навигация для неопытного пользователя будут непривычны. Большим плюсом считаю установку приложений в Linux через консоль. Очень быстро и удобно.
- Визуальный интерфейс. В Windows простой, понятный интерфейс, за счет этого большинство неопытных пользователей начинают знакомство с компьютером с этой ОС. В Linux можно почти полностью поменять внешний вид ОС. В Mac, также как и в Windows возможность кастомизация небольшая.
- Производительность. Производительность рассматриваемых систем отличается не сильно, хотя Windows тратит больше ресурсов, чем остальные.
- Безопасность. У всех трех ОС реализованы системы безопасности (firewall, антивирус). Т.к. Windows самая популярная, то для неё существует больше вредоносного ПО.
- Стоимость. Linux распространяется на бесплатной основе, Mac ОС устанавливается только на оф. машины от Apple (Hackintosh не рассматривается), Windows от \$199.00.

Для разработки программной системы голосового управления мобильного робота была выбрана ОС Linux. Это обусловлено тем, что эта операционная система обладает наибольшим выбором и простотой установки программного обеспечения для разработки, она легковесна и бесплатна, а для разработки данной системы это очень важно.

В целом ОС Linux полностью соответствует необходимым технологическим и функциональным требованиям для разработки данного проекта.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

## 2.2 Выбор языка программирования

Ключевой задачей при разработке приложения является выбор языка программирования, на котором и будет разрабатываться система.

Для целей данной работы лучше всего подходит Python [3] – это универсальный современный язык программирования высокого уровня, к преимуществам которого относят высокую производительность программных решений и структурированный, хорошо читаемый код. Синтаксис Питона максимально облегчен, что позволяет выучить его за сравнительно короткое время. Ядро имеет очень удобную структуру, а широкий перечень встроенных библиотек позволяет применять внушительный набор полезных функций и возможностей. ЯП может использоваться для написания прикладных приложений, а также разработки WEB-сервисов.

Python может поддерживать широкий перечень стилей разработки приложений, в том числе, очень удобен для работы с ООП и функционального программирования.

Питон активно развивается. Примерно раз в 2 года выходят обновления. Важной особенностью языка является отсутствие таких стандартов кодировки как ANSI, ISO и некоторых других, они работают благодаря интерпретатору.

ЯП имеет четко структурированное семантическое ядро и достаточно простой синтаксис. Все, что пишется на этом языке, всегда легко читаемо. В случае необходимости передать аргументы язык использует функцию call-by-sharing.

Питон поддерживает практически все распространенные операционные системы. Он может прекрасно работать на карманных компьютерах, так и на больших серверах. В случае, если платформа значительно устаревает, она исключается из поддержки ядра. К примеру, версии языка, начиная от 2.6, уже не работают с платформами Windows 95, 98 и ME. В случае необходимости можно воспользоваться более старыми версиями, отказавшись от применения современных инструментов языка. И тогда приложение будет работать в том числе с этими ОС. Для старых версий периодически выходят патчи. Язык также может поддерживать работу с виртуальной машиной Java.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

## **2.3 Основные элементы системы**

Можно рассматривать общую структуру голосовых интерфейсов через три основные составляющие:

### **2.3.1 Система распознавания речи**

Компьютеры не способны понять человеческую речь в той форме, в которой ее понимают люди. Для этого существуют системы [1], которые позволяют преобразовывать слитную проблемно-ориентированную человеческую речь в компьютерную форму (ввод звуковой информации в систему и преобразование ее в текст).

### **2.3.2 Событийная система обработки данных.**

Система обрабатывает полученные от системы распознавания речи входные данные. Именно эта часть отвечает за все действия, совершаемые устройством. Подобные системы могут быть абсолютно разными, от простых чат-ботов до полноценных голосовых помощников (обработка текстовой информации, выполнение каких-либо действий, формирование ответа).

### **2.3.3 Система синтеза речи**

Синтез речи [2] — это технология, которая дает возможность воспроизвести текст, как можно более похожим естественным человеческим голосом (преобразование текстовой информации в звуковую с последующим выводом на устройства воспроизведения звука).

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

## 2.4 Обзор систем распознавания речи

Пожалуй, самым важным элементом данной работы является система распознавания речи, так как без нее невозможно создание голосового интерфейса в принципе. Проанализируем и сравним четыре самых доступных на данный момент реализации системы:

### 2.4.1 CMU Sphinx

CMU Sphinx [4] – это система распознавания речи, созданная разработчиками из университета Карнеги-Меллон и состоящая из различных модулей для извлечения речевых признаков, распознавания речи и обучения такому распознаванию. CMU Sphinx использует скрытые марковские модели на акустико-фонетическом уровне распознавания и статистические N-граммные модели на лингвистическом уровне распознавания.

Система подвержена глубокой и гибкой настройке, можно создавать и обучать акустические модели для разных языков, создавать кастомные словари, которые отвечают за непосредственные слова, которые система будет в состоянии распознать.

Система работает локально и не требует подключения к интернету, а при небольших словарях не потребляет большого количества ресурсов.

Однако, практические опыты показали низкую скорость и точность распознавания, а также плохую интеграцию с русским языком.

При использовании системы только как голосовую активацию, было зафиксировано большое количество ложных срабатываний.

### 2.4.2 Kaldi

Kaldi [5] - это набор инструментов для распознавания речи с открытым исходным кодом, написанный на языке C++ для распознавания речи и обработки сигналов, свободно доступный под лицензией Apache v2.0. Это программное обеспечение, которое является гибким и расширяемым, предназначено для использования исследователями автоматического распознавания речи (ASR для построения системы распознавания).

Система бесплатна и не требует подключения к интернету, а так же довольно неплохо распознает как слитную речь, так и отдельные слова.

Из минусов можно выделить то, что система довольно требовательна к ресурсам компьютера и работает только с аудиофайлами, что будет создавать существенные задержки при работе системы в реальном времени.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

### 2.4.3 Google Cloud Speech API

Google Cloud Speech API [6] [7] – облачная технология распознавания речи и синтеза речи. Сервис позволяет распознать, озвучить или перевести любой текст на многих языках.

Из минусов можно выделить необходимость в постоянном интернет соединении с сервером, задержки в получении ответа, а также наличие тарифного плана на работу сервиса.

Преимуществом системы является то, что все вычисления выполняются на удаленном сервере, что позволяет снизить нагрузку на локальное устройство, точное распознавание русского языка, наличие бесплатного ключа для тестирования системы, а также гибкость настройки системы.

### 2.4.4 Yandex SpeechKit

Yandex SpeechKit [8] — облачная технология распознавания речи и синтеза речи от российской компании Яндекс. Практически все то же самое, что и у Google.

Минусы: необходимость в постоянном интернет соединении с сервером, задержки в получении ответа, а также наличие тарифного плана на работу сервиса, меньшее количество распознаваемых языков – всего 3 (русский, английский и турецкий).

Плюсы: все вычисления выполняются на удаленном сервере, что позволяет снизить нагрузку на локальное устройство, более приятный на слух синтез речи, а также менее накладная ценовая политика.

### 2.4.5 Финальный выбор системы

Для наших целей был выбран Google Cloud Speech API для распознавания и синтеза речи.

У системы очень хорошая интеграция с языком Python, подробно описанная документация, низкие затраты ресурсов, высокая гибкость как в настройке системы, так и в способах обработки данных.

Также у Google Cloud Speech API, несмотря на то, что вычисления проводятся в облаке, одна из самых низких задержек распознавания и самая высокая точность распознавания, что в совокупности с бесплатным ключом для тестов и высокой гибкостью системы делает систему лучшей для данной работы.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
						13
Изм.	Лист	№ докум.	Подпись	Дата		

### 3 Разработка системы голосового управления мобильного робота

#### 3.1 Разработка общей структуры системы

Теперь рассмотрим общую структуру работы системы:

- Данные голосовой команды, записанной микрофоном, подаются на вход системы распознавания речи.
- Система распознавания речи преобразует полученный аудио-сигнал в текст, который может быть обработан роботом.
- Робот обрабатывает полученный текст, выполняет команду, генерирует ответ для пользователя в текстовом виде и отправляет его на вход системы синтеза речи.
- Система синтеза речи получает текст ответа и преобразует ее в аудио дорожку, которая впоследствии воспроизводится из динамиков.

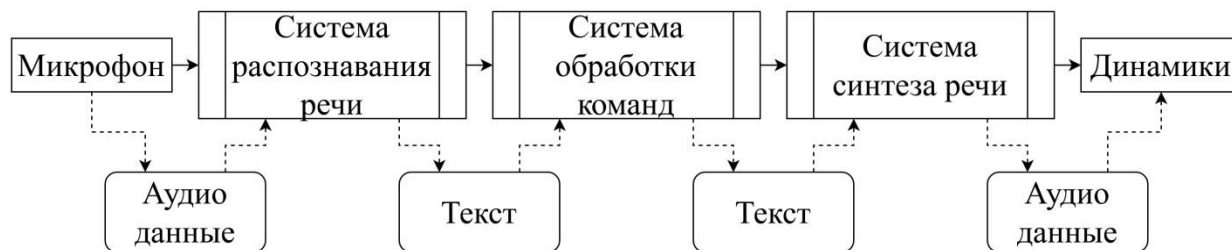


Рис. 1 Схема общей структуры системы



### 3.2 Протокол взаимодействия с роботом

Для того, чтобы робот мог адекватно воспринимать отдаваемые ему команды, необходимо продумать структуру голосового запроса. Чтобы робот мог отличать голосовые данные, которые его не касаются, от направленных к нему запросов необходимо **кодовое слово** – ключевое слово для активации работы робота, вроде “Ok Google” или “Alexa”, в нашем примере – “Ёлка” (ELC Bot).

**Команда** – непосредственное поручение роботу, которое подается одним словом, простым предложением или вопросом.

Первоначальная схема общения с роботом при использовании CMU Sphinx для распознавания кодового слова и Google Cloud Speech в режиме синхронного распознавания для распознавания команды:

- Ожидание кодового слова. Система прослушивает аудио с микрофона, распознает его и ищет в тексте кодовое слово.
- Запись команды. После обнаружения кодового слова начинается запись команды.
- Отправка записи на сервер. Запись команды отправляется на сервер Google для обработки и преобразования ее в текст.
- Обработка команды на сервере.
- Принятие текста команды.
- Обработка команды. Парсинг полученного текста модулем обработки естественного языка на наличие известной команды.
- Исполнение команды. В случае распознавания системой команды, она ее выполняет, генерирует ответ, который подается на систему синтеза речи. Система синтеза речи преобразует текст ответа в аудио-файл, который воспроизводится с динамиков. Если система не смогла распознать команду, генерируется ответ, сообщающий об этом пользователю. Далее система возвращается к ожиданию кодового слова.



Рис. 2 Первоначальная схема общения с роботом

Такой формат работы создавал значительные временные задержки, т.к. система CMU Sphinx распознавала текст довольно долго и аудио команды отправлялось только после полной записи, поэтому схема подверглась доработке.

В итоговой схеме общения с роботом для распознавания речи используется Google Cloud Speech в режиме потокового распознавания.

Итоговая схема общения с роботом:

- Ожидание кодового слова. Система прослушивает аудио с микрофона и отправляет его на сервер, постоянно принимая текст.
- Вычленение команды из текста. Как только в тексте будет обнаружено кодовое слово, весь текст, полученный после него будет использован в качестве команды.
- Проверка команды на соответствие. Парсинг полученного текста на наличие известной команды.
- Исполнение команды. В случае распознавания системой команды, она ее выполняет, генерирует ответ, который подается на систему синтеза речи.
- Выдача ответа. Система синтеза речи преобразует текст ответа в аудио-файл, который воспроизводится с динамиков. Если система не смогла распознать команду, генерируется ответ, сообщающий об этом пользователю. Далее система возвращается к ожиданию кодового слова.

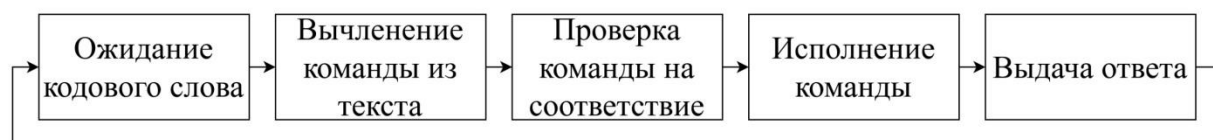


Рис. 3 Итоговая схема общения с роботом

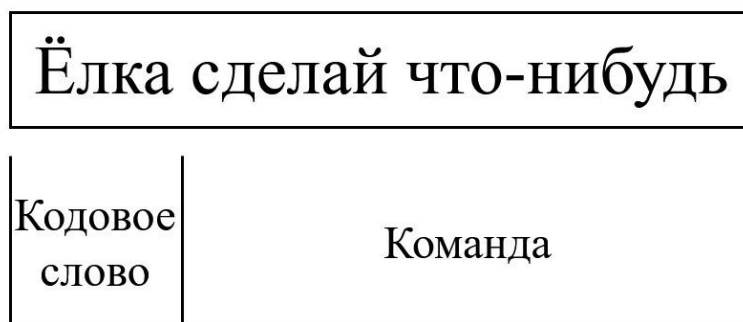


Рис. 4 Формат команды

Так как команда произносится сразу за кодовым словом и используется потоковое распознавание, текст команды принимается клиентом во время произнесения самой команды, и к моменту завершения произнесения уже начинает ее обрабатывать. Таким образом минимизируется временная задержка между началом обращения пользователя к системе и началом выполнения отданной команды.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

## 4 Разработка программных средств

### 4.1 Работа с Google Cloud Speech

#### 4.1.1 Регистрация

Для начала работы с Google Cloud Speech [6] [7] необходимо:

- зарегистрировать аккаунт как сервис, получив таким образом доступ к различным системам вычисления в облаке
- создать проект, в котором будут храниться все данные о использовании облака
- получить личный ключ авторизации в виде файла json.

При регистрации пробного периода для тестирования проекта пользователю выдается объем доступных вычислений на сумму 600 американских долларов на срок в 365 дней, по истечении которого необходимо будет приобрести полноценный аккаунт.

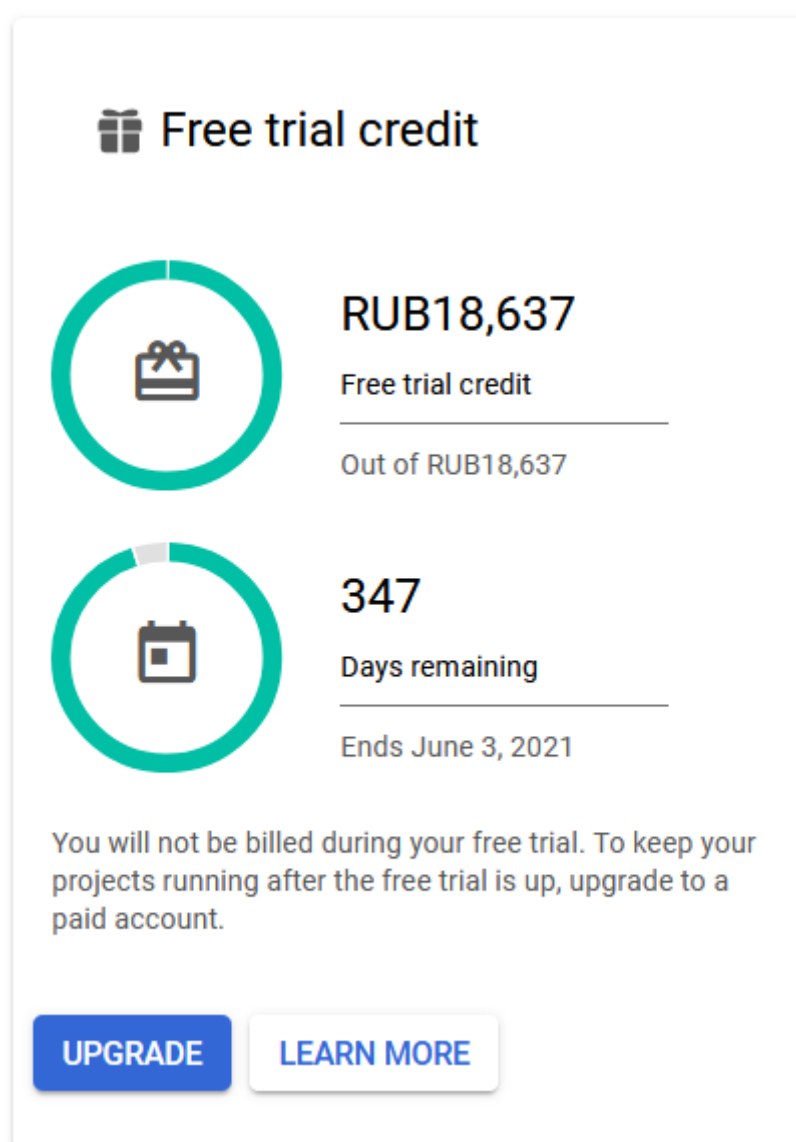


Рис. 5 Оставшаяся сумма и срок использования пробного периода

#### 4.1.2 Установка

Далее нужно прописать переменную окружения личного ключа для автоматической авторизации запросов во время работы системы. На операционной системе Linux команда выглядит следующим образом:

```
export GOOGLE_APPLICATION_CREDENTIALS="[PATH]"
```

где PATH – полный путь до файла ключа.

При вводе команды переменная будет определена только для текущей сессии, поэтому ее целесообразно прописать в файле конфигурации командной оболочки.

Для установки библиотек необходимо ввести следующую команду:

```
pip3 install google-cloud-speech
```

Установщик пакетов сам загрузит и установит все необходимые библиотеки и зависимости.

#### 4.1.3 Варианты взаимодействия

Система имеет три основных метода для выполнения распознавания речи:

- Синхронное распознавание - клиент отправляет звуковые данные на сервер, который выполняет распознавание и возвращает результат в виде текста после обработки всего аудио. Присутствует ограничение по длительности аудио-дорожки в 1 минуту. Синхронные запросы являются блокирующими, это значит, что сервер должен вернуть ответ перед тем, как снова приступить к обработке запроса. Система обычно обрабатывает аудио быстрее реального времени, в среднем обрабатывая 30 секунд аудио-данных за 15 секунд, однако при ухудшении качества аудио-данных время обработки может существенно возрасти.
- Асинхронное распознавание — клиент отправляет звуковые данные на сервер и инициирует длительную работу по распознаванию. Во время распознавания клиент может периодически запрашивать частичные результаты. Ограничение по длительности аудио-дорожки равно 480 минутам.
- Потокосное распознавание - распознавание выполняется на звуковых данных, предоставленных в двунаправленном потоке. Потокосные запросы предназначены для распознавания в реальном времени, таком как захват звука с микрофона. Сервер возвращает промежуточные результаты во время захвата звука, позволяя получить текст когда пользователь все еще говорит. Окончательный результат распознавания представляет собой последнее лучшее предположение системы.

## 4.2 Разработка программной структуры системы

### 4.2.1 Элементы системы

Рассмотрим элементы системы:

- `MicrophoneStream` – класс, предназначенный для реализации получения данных из микрофона, их обработки и нарезки на отдельные чанки данных и выдачу для последующей отправки данных на сервер.
- `main` – основная функция, в которой реализованы запуск получения данных из микрофона, отправка запросов на сервер, принятие результатов вычислений сервера и запуск функции обработки результатов.
- `listen_loop` – функция, предназначенная для обработки результатов, принятых с сервера. На нее поступают куски текста, по которым ведется поиск на наличие кодового слова, и при нахождении которого из текста вырезается команда и подается на вход функции обработки команд.
- `process_comand` – функция, предназначенная для обработки текста команд, которые она получает на вход. В ней ведется парсинг текста на совпадение с известными командами, и в случае обнаружения таковой она выполняется, генерируется ответ и запускается функция озвучки ответа. При отсутствии совпадений генерируется ответ, сообщающий об этом пользователю.
- `text_to_speech` – функция, предназначенная для озвучки текста ответа, который получает на вход. Она отправляет запрос с текстом ответа на сервер, принимает ответ в виде аудио-данных и записывает их в файл, который затем воспроизводится из динамиков. Для минимизации задержек по времени реализован функционал, позволяющий сохранять ответы в отдельные файлы для будущего использования, чтобы не тратить временные и денежные ресурсы для повторной генерации уже использованных ответов.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

#### 4.2.2 Схема работы программы

- Микрофон генерирует аудио-данные частотой дискретизации 16МГц, которые программа захватывает, нарезает на чанки объемом 10% от частоты дискретизации, т.е. 1600 значений, и возвращает их функции main
- Функция формирует запрос на сервер Гугла, в котором отправляет чанк аудио-данных для последующего распознавания.
- После завершения распознавания сервер возвращает ответ с текстом, который отправляется на функцию парсинга текста запроса.
- Функция проверяет ответ по его флагу как финальный, и если результат распознавания не финальный программа возвращается к началу и приступает к обработке следующего чанка данных. Если результат распознавания финальный, функция проверяет текст на наличие в нем кодового слова, и при наличии такового вычленяет весь текст следующий за кодовым словом, который воспринимается как команда.
- Далее текст команды подается на модуль обработки, в котором проводится парсинг полученного текста на наличие известной команды. В случае распознавания системой команды, она ее выполняет. Далее в зависимости от результата генерируется текст ответа, который отправляется на сервер для синтеза речи.
- Сервер в свою очередь после синтеза отправляет назад аудио-данные, которые записываются в файл и воспроизводятся через динамики.

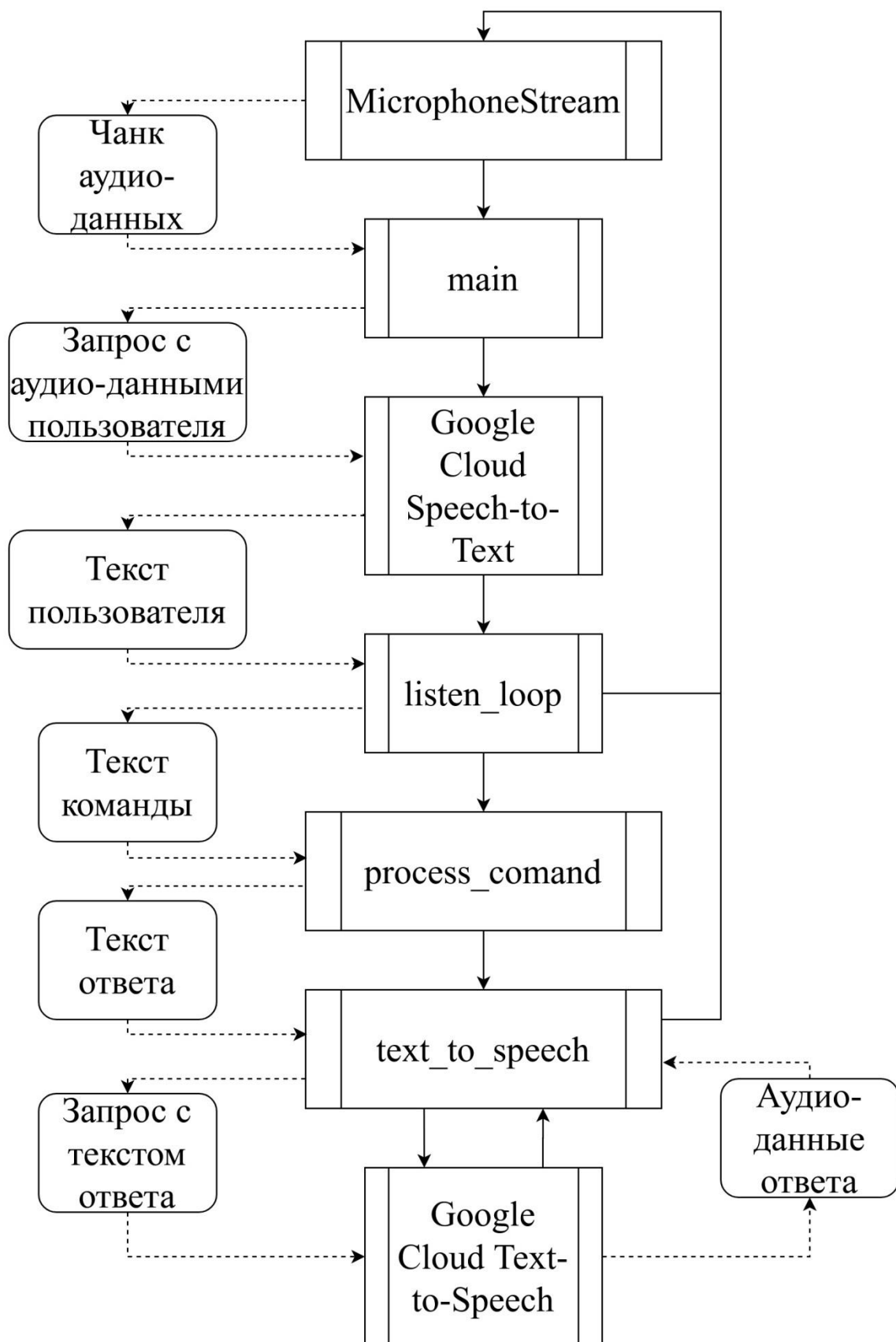


Рис. 6 Итоговая схема работы программы



## 4.3 Программная реализация модулей системы

### 4.3.1 Реализация `MicrophoneStream`

Отметим основные этапы работы класса [9]:

```
def __init__(self, rate, chunk):
    self._rate = rate
    self._chunk = chunk
    self._buff = queue.Queue()
    self.closed = True
def __enter__(self):
    self._audio_interface = pyaudio.PyAudio()
    self._audio_stream = self._audio_interface.open(
        ...
    )
    self.closed = False
    return self
```

При создании объекта класса задаются параметры записи аудио и создается интерфейс к микрофону, данные из которого будут записываться в буфер с помощью функции `_fill_buffer`:

```
def _fill_buffer(self, in_data, frame_count, time_info, status_flags):
    self._buff.put(in_data)
    return None, pyaudio.paContinue
```

Функция `generator` вынимает чанк аудио-данных из буфера, если они там есть, и возвращает его:

```
def generator(self):
    while not self.closed:
```

В первую очередь выполняется проверка на состояние потока. Если в переменной закрытия стоит значение `true`, генерация данных прекращается:

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

```

    chunk = self._buff.get()
    if chunk is None:
        return
    data = [chunk]

```

Далее в бесконечном цикле выполняется изъятие данных из буфера:

```

while True:
    try:
        chunk = self._buff.get(block=False)

```

Если в буфере ничего нет, функция завершает работу:

```

    if chunk is None:
        return

```

Если данные были изъаты из буфера, они записываются в накапливаемую переменную:

```

        data.append(chunk)
    except queue.Empty:
        break

```

В конце функция выдает данные для отправки на сервер в байтовом формате:

```

    yield b''.join(data)

```

### 4.3.2 Реализация main

В начале создаются необходимые для работы Google Cloud Speech API необходимые элементы и настройки конфигурации:

```
client = speech.SpeechClient()
config = types.RecognitionConfig(encoding =
enums.RecognitionConfig.AudioEncoding.LINEAR16, sample_rate_hertz = RATE,
language_code = language_code)
streaming_config = types.StreamingRecognitionConfig (config = config,
interim_results = True)
```

Затем создается экземпляр класса MicrophoneStream и генератор данных:

```
with MicrophoneStream(RATE, CHUNK) as stream:
    audio_generator = stream.generator()
```

Запросы реализуются через функцию библиотеки Googl'a, в нее подаются данные из генератора:

```
requests = (types.StreamingRecognizeRequest(audio_content=content) for content
in audio_generator)
```

Прием ответов реализуется также через функцию библиотеки Google'a, в нее подаются данные конфигурации и сами запросы:

```
responses = client.streaming_recognize(streaming_config, requests)
```

В конце запускается функция обработки результатов:

```
listen_loop(responses)
```

### 4.3.3 Реализация listen\_loop

Во время разговора в объект responses постоянно записываются новые данные. Функция проверяет его на наличие новых данных, и в случае наличия таковых сохраняет их:

```
for response in responses:
    if not response.results:
        continue
    result = response.results[0]
    if not result.alternatives:
        continue
    transcript = result.alternatives[0].transcript
```

Функция ожидает конечного результата распознавания, который поступает после окончания фразы, которую произнес пользователь:

```
if not result.is_final:
    ...
else:
```

Далее весь текст переводится в нижний регистр для упрощения обработки текста:

```
transcript = transcript.lower()
```

Затем проводится проверка текста на наличие кодового слова, и в случае обнаружения такового из текста ответа выделяется команда и отправляется на обработку вызовом функции process\_command:

```
if WAKE_WORD in transcript:
    command = transcript.split(WAKE_WORD,1)[1]
    process_command(command)
```

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

#### 4.3.4 Реализация process\_command

Функция написана только для теста и презентации работы системы распознавания и не является конечным результатом работы программы.

Вот фрагмент ее функционала, который отвечает за завершение работы программы. Тут происходит базовый поиск ключевых слов в команде, и при их обнаружении переменная HALT выставляется в 0, что впоследствии завершит работу программы, и запускается функция text\_to\_speech, которая воспроизведет слово “выхожу”:

```
global HALT
if ("выключись" in text) or ("выход" in text):
    HALT = 0
    text_to_speech("выхожу")
    return
```

#### 4.3.5 Реализация text\_to\_speech

Функция реализует в себе отправку текста ответа на сервер, принятие аудио-данных, сохранение их в файл и воспроизведение данных через динамики.

На вход функции поступает текст ответа:

```
def text_to_speech(play_text):
```

Обозначается имя файла, в который будут записаны аудио-данные:

```
sound_filename = "output.mp3"
```

Задаются параметры запроса, которые включают в себя текст для синтеза, настройки голоса и языка, которым будет озвучен текст и параметры аудио-данных:

```
client = texttospeech.TextToSpeechClient()
synthesis_input = texttospeech.SynthesisInput(text=play_text)
voice      =      texttospeech.VoiceSelectionParams(language_code="ru-RU",
ssml_gender=texttospeech.SsmlVoiceGender.NEUTRAL)
audio_config      =      texttospeech.AudioConfig
(audio_encoding=texttospeech.AudioEncoding.MP3)
```

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
						27
Изм.	Лист	№ докум.	Подпись	Дата		

Выполняется запрос с заданными параметрами:

```
response = client.synthesize_speech(input=synthesis_input, voice=voice,  
audio_config=audio_config)
```

Далее синтезированные аудио-данные записываются в файл:

```
with open(sound_filename, "wb") as out:  
    out.write(response.audio_content)  
    print('Audio content written to file "output.mp3"')
```

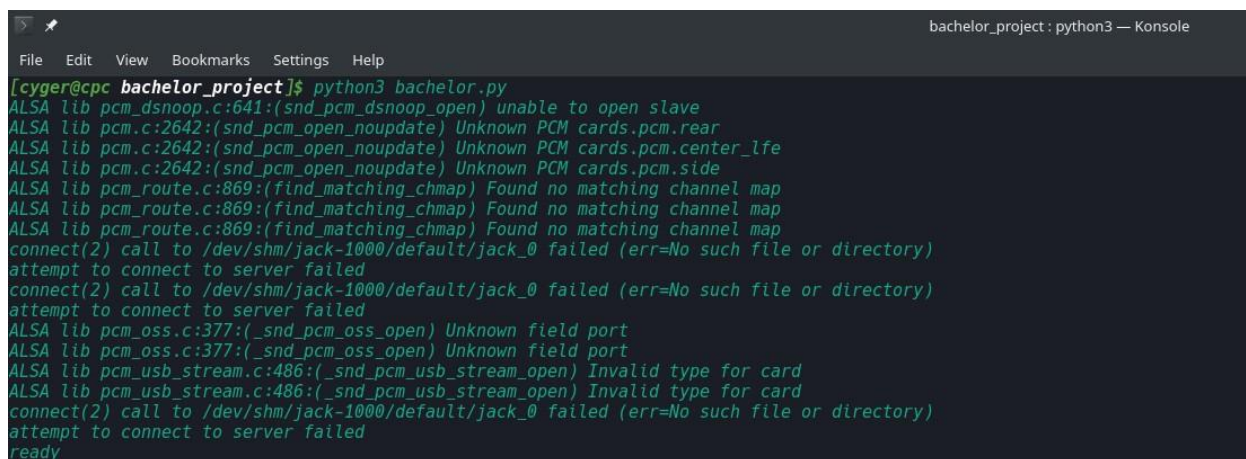
В конце записанный файл воспроизводится:

```
playsound.playsound(sound_filename)
```

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		28

## 5 Тестирование системы

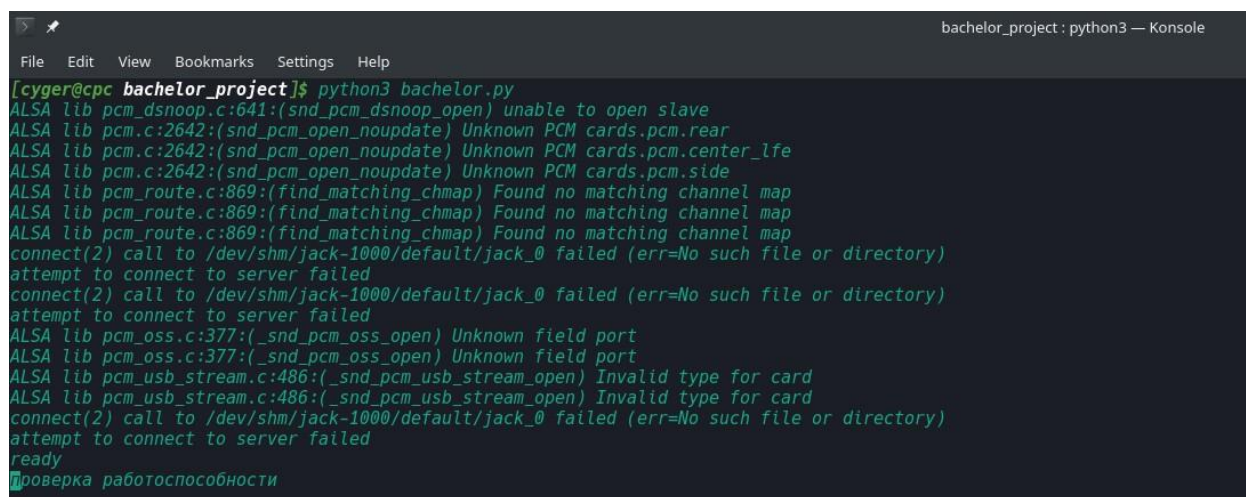
На рисунке 7 показан процесс запуска программы. В начале системы выполняет поиск системы всех доступных слотов для микрофона. Затем выводится сообщение “ready”, сообщающее о том, что система готова к работе.



```
bachelor_project : python3 — Konsole
File Edit View Bookmarks Settings Help
[cyger@cpc bachelor_project]$ python3 bachelor.py
ALSA lib pcm_dsnoop.c:641:(snd_pcm_dsnoop_open) unable to open slave
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ready
```

Рис. 7 Запуск программы

На рисунках 8 и 9 можно наблюдать процесс распознавания речи в реальном времени. Система проигнорировала этот текст, так как не было названо кодовое слово.



```
bachelor_project : python3 — Konsole
File Edit View Bookmarks Settings Help
[cyger@cpc bachelor_project]$ python3 bachelor.py
ALSA lib pcm_dsnoop.c:641:(snd_pcm_dsnoop_open) unable to open slave
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ready
Проверка работоспособности
```

Рис. 8 Распознавание речи в процессе

```

bachelor_project : python3 — Konsole
File Edit View Bookmarks Settings Help
[cyger@cpc bachelor_project]$ python3 bachelor.py
ALSA lib pcm_dsnoop.c:641:(snd_pcm_dsnoop_open) unable to open slave
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
ALSA lib pcm_route.c:869:(find_matching_chmap) Found no matching channel map
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ready
проверка работоспособности программы

```

Рис. 9 Окончание распознавания высказывания

```

bachelor_project : bash — Konsole
File Edit View Bookmarks Settings Help
[cyger@cpc bachelor_project]$ python3 bachelor.py
ALSA lib pcm_dsnoop.c:641:(snd_pcm_dsnoop_open) unable to open slave
ALSA lib pcm_dmix.c:1089:(snd_pcm_dmix_open) unable to open slave
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_dmix.c:1089:(snd_pcm_dmix_open) unable to open slave
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ready
ёлка ты кто
я робот телеприсутствия ёлка

```

Рис. 10 Пример команды

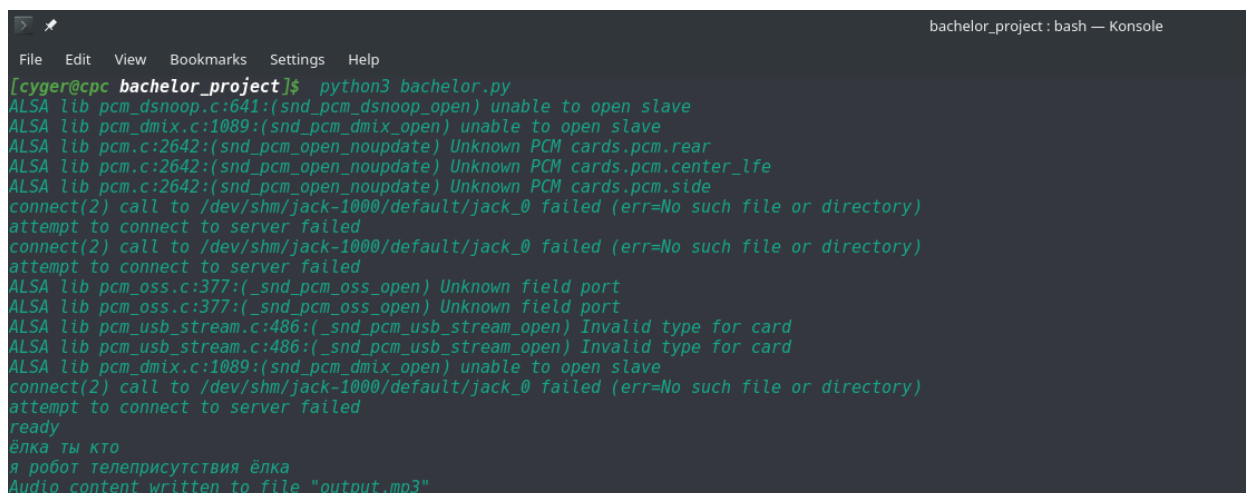
На рисунке 10 показан пример процесса выполнения команды. Было озвучено кодовое слово “ёлка”, после чего была отдана команда “ты кто”.

Система вывела распознанный текст в консоль, после чего рандомно был выбран ответ на эту команду (“я робот телеприсутствия ёлка”), который был выведен в консоль, синтезирован в аудио-данные, сохранен в файл и озвучен.



На рисунке 11 показана реакция системы на неизвестную команду.

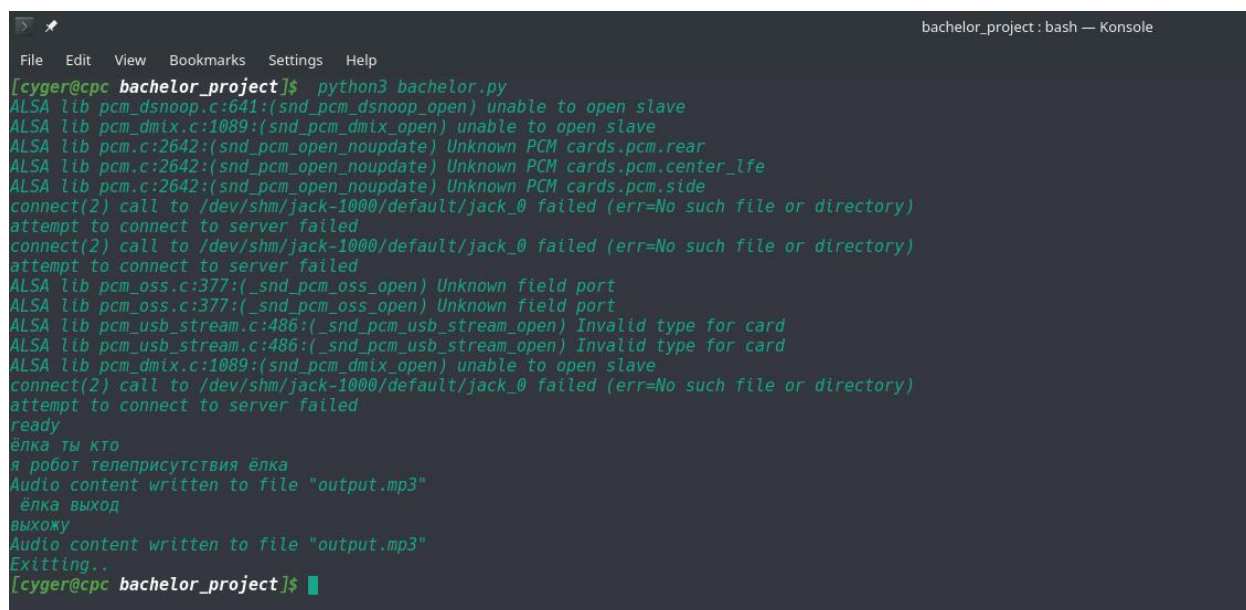
Было озвучено кодовое слово, но система не смогла распознать команду и вывела результат, сообщающий об этом пользователю.



```
batchelor_project : bash — Konsole
File Edit View Bookmarks Settings Help
[cyger@cpc batchelor_project]$ python3 batchelor.py
ALSA lib pcm_dsnoop.c:641:(snd_pcm_dsnoop_open) unable to open slave
ALSA lib pcm_dmix.c:1089:(snd_pcm_dmix_open) unable to open slave
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_dmix.c:1089:(snd_pcm_dmix_open) unable to open slave
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ready
ёлка ты кто
я робот телеприсутствия ёлка
Audio content written to file "output.mp3"
```

Рис. 11 Реакция системы на неизвестную команду

На рисунке 12 показано выход из программы с помощью команды “выход”.



```
batchelor_project : bash — Konsole
File Edit View Bookmarks Settings Help
[cyger@cpc batchelor_project]$ python3 batchelor.py
ALSA lib pcm_dsnoop.c:641:(snd_pcm_dsnoop_open) unable to open slave
ALSA lib pcm_dmix.c:1089:(snd_pcm_dmix_open) unable to open slave
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2642:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_oss.c:377:(snd_pcm_oss_open) Unknown field port
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_usb_stream.c:486:(snd_pcm_usb_stream_open) Invalid type for card
ALSA lib pcm_dmix.c:1089:(snd_pcm_dmix_open) unable to open slave
connect(2) call to /dev/shm/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed
ready
ёлка ты кто
я робот телеприсутствия ёлка
Audio content written to file "output.mp3"
ёлка выход
выхожу
Audio content written to file "output.mp3"
Exiting..
[cyger@cpc batchelor_project]$
```

Рис. 12 Окончание работы программы с помощью команды

## Заключение

В результате выполнения выпускной квалификационной работы была разработана и реализована программная система голосового управления роботом. Для этой цели был проведен обзор доступных систем распознавания и синтеза речи, а также изучены различные подходы к разработке систем голосового управления.

Созданный программный продукт отвечает всем поставленным требованиям и решает все необходимые задачи. Тестирование данного продукта подтвердило его работоспособность и возможность дальнейшего использования для решения подобных задач.

					ВКР-НГТУ-09.03.01-(16-В-2)-011-2020(ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		32

## Список литературы

- [1 «VOICE RECOGNITION SYSTEM: SPEECH-TO-TEXT,» [В Интернете]. Available:  
] [https://www.researchgate.net/publication/304651244\\_VOICE\\_RECOGNITION\\_SYSTEM\\_SPEECH-TO-TEXT](https://www.researchgate.net/publication/304651244_VOICE_RECOGNITION_SYSTEM_SPEECH-TO-TEXT).
- [2 «Review on Text-To-Speech Synthesizer,» [В Интернете]. Available:  
] [https://www.researchgate.net/publication/304600612\\_Review\\_on\\_Text-To-Speech\\_Synthesizer](https://www.researchgate.net/publication/304600612_Review_on_Text-To-Speech_Synthesizer).
- [3 «Python 3,» [В Интернете]. Available: <https://docs.python.org/3/index.html>.  
]
- [4 «CMU Sphinx,» [В Интернете]. Available: <https://cmusphinx.github.io/>.  
]
- [5 «Kaldi,» [В Интернете]. Available: <https://kaldi-asr.org/>.  
]
- [6 «Google Cloud Speech-to-Text,» [В Интернете]. Available:  
] <https://cloud.google.com/speech-to-text/>.
- [7 «Google Cloud Text-to-Speech,» [В Интернете]. Available: <https://cloud.google.com/text-to-speech/>.  
]
- [8 «Yandex SpeechKit,» [В Интернете]. Available: <https://cloud.yandex.ru/docs/speechkit/>.  
]
- [9 L. Ramalho, Fluent Python: Clear, Concise, and Effective Programming, O'Reilly Media; 1  
] edition, 2015.

## Приложения

### Приложение 1. Код программы.

```
from __future__ import division

import re
import sys
import random

from google.cloud import speech
from google.cloud.speech import enums
from google.cloud.speech import types

from google.cloud import texttospeech

import pyaudio
from six.moves import queue
import playsound

RATE = 16000
CHUNK = int(RATE / 10)

WAKE_WORD = "ёлка"

HALT = 1

class MicrophoneStream(object):

    def __init__(self, rate, chunk):
        self._rate = rate
        self._chunk = chunk
```

```

        self._buff = queue.Queue()
        self.closed = True

    def __enter__(self):
        self._audio_interface = pyaudio.PyAudio()
        self._audio_stream = self._audio_interface.open(
            format=pyaudio.paInt16,
            channels=1, rate=self._rate,
            input=True, frames_per_buffer=self._chunk,
            stream_callback=self._fill_buffer,
        )

        self.closed = False

        return self

    def __exit__(self, type, value, traceback):
        self._audio_stream.stop_stream()
        self._audio_stream.close()
        self.closed = True

        self._buff.put(None)
        self._audio_interface.terminate()

    def _fill_buffer(self, in_data, frame_count, time_info,
status_flags):

        self._buff.put(in_data)
        return None, pyaudio.paContinue

    def generator(self):
        while not self.closed:

            chunk = self._buff.get()

```

```

    if chunk is None:
        return
    data = [chunk]

    while True:
        try:
            chunk = self._buff.get(block=False)
            if chunk is None:
                return
            data.append(chunk)
        except queue.Empty:
            break

    yield b''.join(data)

```

```

def listen_loop(responses):

```

```

    num_chars_printed = 0
    for response in responses:
        if not response.results:
            continue

        result = response.results[0]
        if not result.alternatives:
            continue

        transcript = result.alternatives[0].transcript

        overwrite_chars = ' ' * (num_chars_printed -
len(transcript))

        if not result.is_final:
            sys.stdout.write(transcript + overwrite_chars + '\r')

```

```

        sys.stdout.flush()

        num_chars_printed = len(transcript)

    else:
        transcript = transcript.lower()
        print(transcript + overwrite_chars)

        if WAKE_WORD in transcript:
            command = transcript.split(WAKE_WORD,1)[1]
            process_command(command)

        num_chars_printed = 0

    if (HALT==0):
        print("Exitting..")
        break

def process_command(text):
    global HALT
    if ("выключись" in text) or ("выход" in text):
        HALT = 0
        text_to_speech("выхожу")
        return

    if ("повтори" in text):
        text_to_speech(text)
        return

    if ("расскажи" in text) and (("что-нибудь" in text) or
("историю" in text)):
        random_respond = ["рандомный текст один", "рандомный
текст два", "рандомный текст три"]

```

```

        text_to_speech(random.choice(random_respond))
    return

if("расскажи о себе" in text) or ("ты кто" in text):
    respond = ["я ёлка", "я робот телеприсутствия"]
    text_to_speech(random.choice(respond))
    return

def text_to_speech(play_text):
    print(play_text)

    sound_filename = "output.mp3"

    client = texttospeech.TextToSpeechClient()
    synthesis_input = texttospeech.SynthesisInput(text=play_text)
    voice = texttospeech.VoiceSelectionParams(language_code="ru-
RU", ssml_gender=texttospeech.SsmlVoiceGender.NEUTRAL)
    audio_config =
texttospeech.AudioConfig(audio_encoding=texttospeech.AudioEncoding.MP3
)
    response = client.synthesize_speech(input=synthesis_input,
voice=voice, audio_config=audio_config)

    with open(sound_filename, "wb") as out:
        out.write(response.audio_content)
        print('Audio content written to file "output.mp3"')

def main():

```



```

language_code = 'ru-RU'

client = speech.SpeechClient()
config =
types.RecognitionConfig(encoding=enums.RecognitionConfig.AudioEncoding
.LINEAR16, sample_rate_hertz=RATE, language_code=language_code)
streaming_config =
types.StreamingRecognitionConfig(config=config, interim_results=True)

with MicrophoneStream(RATE, CHUNK) as stream:
    audio_generator = stream.generator()
    requests =
(types.StreamingRecognizeRequest(audio_content=content) for content in
audio_generator)

responses = client.streaming_recognize(streaming_config,
requests)

print("ready")

listen_print_loop(responses)

if __name__ == '__main__':
    main()

```