**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

**(Университет ИТМО)**

Факультет СУ и Р

**Образовательная программа** Робототехника и искусственный интеллект

О Т Ч Е Т

о производственной практике, проектно-технологической

Тема задания: Разработка человеко-машинного интерфейса для сервисного робота

Обучающийся: Лещенко Наталья Андреевна, гр. R3235

Руководитель практики от университета: Бжихатлов Ислам Асланович, доцент факультета систем управления и робототехники

Санкт-Петербург

2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc170252315)

[1. Подготовка к практике, подбор библиотек 4](#_Toc170252316)

[2. Прототип программы и последующая реализация 5](#_Toc170252317)

[ВЫВОДЫ 6](#_Toc170252318)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 7](#_Toc170252319)

# ВВЕДЕНИЕ

Человеко-машинный интерфейс – это активно развивающаяся область, которая включает в себя программирование, промышленный дизайн, теорию автоматического управления, *usability engineering* и многие другие науки[1]. ЧМИ позволяет роботам эффективно взаимодействовать с людьми, упрощает процесс управления, помогает предотвратить ошибки операторов, минимизировать риски при работе с роботами. Для меня, как для робототехника, данная тема является особенно актуальной, так как ЧМИ играет ключевую роль в интеграции роботов в рабочие процессы.

Целью моей практической деятельности было создание человеко-машинного интерфейса. В ходе своей работы я разработала код на языке *Python*, который имитирует функциональность умного голосового помощника, что является одним из аспектов ЧМИ. Разработанный код умеет распознавать голосовые команды пользователя, обрабатывать результат, строить с пользователем диалог, а также выполнять определенный набор команд, который воспроизводится голосом, но в дальнейшем может быть передан как команда для управления роботом.

Главными задачами практики стали: изучение существующих решений, подбор необходимых библиотек на языке *Python*, создание прототипа человека-машинного интерфейса и его реализация, проведение пользовательского тестирования и исправление выявленных ошибок и недоработок.

# Подготовка к практике, подбор библиотек

Для подготовки к практике мною были просмотрены видео-лекции на *YouTube*[2], а также прочитана соответствующая литература. Наиболее интересным и практичным оказалось пособие на сайте *vc.ru*[3], а также подробное руководство на *EasyDot*[4].

Для реализации голосового ассистента использовался язык *Python* и установленные в нем библиотеки *vosk*, *time*, *pyttsx3*, *nltk*, *pyaudio*, *json*. Библиотека *vosk* использовалась для распознавания произносимых слов и преобразования их в речь. Для использования *vosk* была установлена языковая модель, весом около 2 Гб, содержащая в себе несколько тысяч русских слов для наилучшего распознавания речи[5]. Библиотека *pyttsx3* послужила для преобразования распознанного текста в речь для реализации функции *text\_to\_speech*:

def text\_to\_speech(text):

engine = pyttsx3.init()

engine.say(text)

engine.runAndWait()

*ntlk* использовалась для токенизации текста (разбиения его на отдельные слова) и лемматизации (приведения слов к словарной форме). Библиотека *pyaudio* использовалась для обработки аудио с микрофона и воспроизведению голосовых реплик ассистента.

# Прототип программы и последующая реализация

Прототипом готового голосового ассистента стала разработанная мной программа, работающая только через интернет из-за использования *API Google Speech Recognition* для распознавания речи и преобразования ее в текст. Программа требовала интернет-соединение для отправки аудиоданных на сервера *Google* для последующей обработки. Модель стала хорошим прототипом и отличным началом для конечного продукта, однако имела ряд недостатков: зависимость от интернет-соединения, проблемы с приватностью, зависимость от третьей стороны (если *API Google* изменится или станет недоступным, программа не сможет работать), долгое время работы – отправка аудиоданных на сервер занимает время. С готовым кодом вы можете ознакомиться по ссылке на *GitHub*[6].

После тестирования прототипа стало очевидным, что программа должна работать с помощью *off-line* библиотек. С помощью *vosk* была подключена голосовая модель и разработана главная функция распознавания речи *listen\_to\_speech*, которая слушает аудио-вход с микрофона и возвращает текст, в случае успешной расшифровки. Функция открывает поток аудио-входа с заданными параметрами, выводит сообщение «*Listening*»(Слушаю) для пользователя, далее запускает бесконечный цикл, в котором читает поток байт из аудио-файла и возвращает текстовое значение.

Далее в игру вступает функция *text\_to\_speech*, которая преобразует полученный текст в речь с помощью библиотеки *pyttx3*. Следующая по порядку выполнения функция *execute\_voice\_command* анализирует текст на наличие ключевых слов и выполняет действия, прописанные в коде.

Основной цикл программы слушает аудио-вход с микрофона и ожидает слово *wake word* «маруся», если *wake word* обнаружен, программа начинает работу, говорит пользователю «Привет» и начинает слушает команды. Она лемматизирует полученную речь, при обнаружении ключевых слов, которые соответствуют *dialog\_data*, выводит соответствующий ответ с помощью *text\_to\_speech*. Ожидает ответа от пользователя *listen\_to\_speech* и при наличии в нем ключевых слов выполняет соответствующие действия.

Важным качеством созданного ассистента является не просто его умение слушать голосовые команды и отвечать на них голосом, но и строить диалог с пользователем. Так программа, услышав от пользователя фразу «Выпить бы колы», уточняет: «Вы хотите выпить колы?». В случае ответа «Да» она подсказывает ближайший магазин, в случае отрицательного ответа недоумевает: «А зачем вы спрашивали?»

Программа является отличным голосовым помощником, обладающим гибкой архитектурой: она обладает модульностью, конфигурируемостью (*dialog\_data* легко изменяем под нужные параметры и новые функции без дополнительного изменения всей программы), логика программы отделена от данных, может быть легко расширена для поддержки новых модулей и конфигураций.

Готовый разработанный проект находится по ссылке[7].

# ВЫВОДЫ

В результате проделанной работы по созданию кода для голосового ассистента Маруси была разработана программа с гибкой архитектурой, которая позволяет легко изменять, расширять и поддерживать ее.

Данная программа прошла несколько этапов улучшений от прописывания каждой функции отдельно до гибкого удобного кода. Программа является отличным голосовым помощником, работающим без использования интернета, а также может быть использована для голосового управления сервисным роботом с помощью создания интерфейса взаимодействия, модуля управления роботом и интеграции сенсоров. Это может привести к созданию робота, который умеет отвечать на вопросы, выполнять команды и обрабатывать запросы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. [Проектирование человеко-машинного взаимодействия – Учебные курсы – Национальный исследовательский университет «Высшая школа экономики» (hse.ru)](https://www.hse.ru/edu/courses/341840900)

2. [(143) Как сделать голосового помощника на Python - YouTube](https://www.youtube.com/watch?v=YlbAPIijZeI)

3. [Python: как создать простейшего голосового помощника? — Разработка на vc.ru](https://vc.ru/dev/158797-python-kak-sozdat-prosteishego-golosovogo-pomoshnika)

4. [Как создать голосового помощника на языке Python: подробное руководство для новичков (easydoit.ru)](https://www.easydoit.ru/python/kak-sozdat-golosovogo-pomoshhnika-na-yazyke-python-podrobnoe-rukovodstvo-dlya-novichkov/)

5. [VOSK Models (alphacephei.com)](https://alphacephei.com/vosk/models)

6. [Experiment/import speech\_recognition as sr.py at main · NatalyaLeshchenko/Experiment (github.com)](https://github.com/NatalyaLeshchenko/Experiment/blob/main/import%20speech_recognition%20as%20sr.py)

7. [Voice-assistant/import vosk.py at main · NatalyaLeshchenko/Voice-assistant (github.com)](https://github.com/NatalyaLeshchenko/Voice-assistant/blob/main/import%20vosk.py)