ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»
ПО ПРОФИЛЮ «ИНЖЕНЕРНОЕ ДЕЛО»

44168

регистрационный номер

Секция: Радиоэлектроника и лазерная медицинская техника
название секции

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО ГОЛОСОВОГО АССИСТЕНТА ДЛЯ ПРОГРАММИСТОВ С НАРУШЕНИЕМ ЗРЕНИЯ

название работы

Автор: Шелагинов Александр Вадимович
фамилия, имя, отчество
ГБОУ Школа № 1367, 10А
наименование учебного заведения, класс

Научный руководитель: Кручинина Елена Викторовна
фамилия, имя, отчество
МГТУ им. Н.Э. Баумана
место работы
к.т.н., доцент кафедры БМТ-1
звание, должность

Подпись научного руководителя

Москва – 2024

РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНОГО ГОЛОСОВОГО АССИСТЕНТА ДЛЯ ПРОГРАММИСТОВ С НАРУШЕНИЕМ ЗРЕНИЯ

Аннотация

Работа посвящена созданию программного обеспечения для работы программиста с ограничениями по зрению, в среде разработки PyCharm.

При работе за компьютером человек большую часть информации получает визуально, потеря или существенное ухудшение зрения программиста может привести к потере возможности дальнейшей работы.

Цель проекта: создание голосового ассистента, предназначенного для помощи слабовидящим и слепым при написании и отладке программного кода.

Задачи проекта:

1. Обзор инструментов и методов работы с компьютером для людей с нарушением зрения.
2. Разработка требований к голосовому ассистенту для программистов с нарушением зрения.
3. Изучение инструментов, необходимых для создания голосового ассистента.
4. Разработка прототипа голосового ассистента.
5. Тестирование и проведение исследований разработанного голосового ассистента для программистов с нарушением зрения.

Проведено исследование устройств для взаимодействия слабовидящих с компьютером, показана актуальность создания голосового помощника для программирования. В процессе работы были изучены возможности модулей языка программирования Python: PyTorch с Silero Models, Kaldi и vosk, picovoice.

Создан голосовой помощник для программистов, реализованы функции голосового ввода команд: инициализации рабочей среды, перехода между строками, запуска и прерывания программы, озвучивание результата выполнения программы, озвучивания текста программы (целиком и диапазонами строк). Проведена апробация разработанного продукта.

Содержание

Введение

Глаза – самый важный из органов чувств человека, с помощью зрения мы получаем 90% информации из окружающего нас мира. Однако, люди иногда рождаются слепыми или теряют зрение в процессе жизни.

По данным ООН почти 3,5% населения Земли является полностью слепыми или имеют инвалидность по зрению. В России же по данным Всероссийского общества слепых за 2016 год людей с инвалидностью по зрению около 206 тысяч человек.

Также имеются данные по округам от тифло центра «Вертикаль», представленные в таблице 1, которые говорят о 9,67 % населения всей страны [1].

Таблица 1 – Количество инвалидов по зрению на территории России

Территория	**Всего**	**% инвалидизации**	**Мужчины**		**Женщины**	
			Чел.	**Доля, %**	**Чел.**	**Доля, %**
Российская Федерация	11302920	9,67	4806447	42,52	6496473	57,48
Центральный федеральный округ	3304401	10,23	1332739	40,33	1971662	59,67
Северо-Западный федеральный округ	1250742	11,01	460604	36,83	790138	63,17
Северо-Кавказский федеральный округ	883858	12,31	382983	43,33	500875	56,67
Южный федеральный округ	1197379	9,09	554258	46,29	643121	53,71
Приволжский федеральный округ	2222226	9,45	963249	43,35	1258977	56,65
Уральский федеральный округ	722036	7,52	329174	45,59	392862	54,41
Сибирский федеральный округ	1372939	9,18	623831	45,44	749108	54,56
Дальневосточный федеральный округ	349034	7,24	159457	45,69	189577	54,31

Информационные технологии очень плотно вошли в жизнь современного человека и для людей с инвалидностью по зрению бывает очень трудно отказаться от привычного быта.

Именно для тех, кто хочет заниматься программированием, необходимо создавать инструментарий, потому что все люди должны иметь возможность обучаться и работать на компьютере в сфере программирования.

Мониторов, которые выводили бы функциональную информацию с помощью шифра Брайля не существует, также в мире пока нет необходимых инструментов для решения данной задачи в полном объеме.

На данный момент многие приложения добавляют экранного диктора, но этого недостаточно для деятельности в сфере разработки и отладки программного обеспечения. Экранный диктор не позволяет быстро и удобно пользоваться основным функционалом программы, также в большинстве случаев они не реагируют на действия пользователя (результат выполнения программы, написанной пользователем, почти никогда не озвучивается).

Поэтому работа посвящена разработке интеллектуального голосового ассистента для программистов с нарушением зрения, которое позволит решать базовые задачи при создании программных продуктов.

Цель проекта

Создание голосового ассистента, предназначенного для помощи слабовидящим и слепым при написании и отладке программного кода.

Задачи проекта

6. Обзор инструментов и методов работы с компьютером для людей с нарушением зрения.
7. Разработка требований к голосовому ассистенту для программистов с нарушением зрения.
8. Изучение инструментов, необходимых для создания голосового ассистента.
9. Разработка прототипа голосового ассистента.

10. Тестирование и проведение апробации разработанного голосового ассистента для программистов с нарушением зрения.

Оснащение и оборудование, использованное при создании работы

Персональный компьютер с установленным ПО Python и PyCharm.

Глава 1 Исследование взаимодействия программиста и компьютера

1.1 Описание взаимодействия программиста и компьютера

Программист взаимодействует с компьютером в основном посредством зрения, осязания и частично слуха. В свою очередь процесс программирования, без учета непосредственно интеллектуальной деятельности, заключается в совершении следующих базовых операций:

- запуск среды разработки и её настройка;
- нажатия кнопок на клавиатуре и мышке;
- чтение кода целиком и отдельными строками;
- запуск и прерывание выполнения программы;
- чтение результата выполнения программы;
- переход к определенной строке.

При нарушении зрения выполнение практически всех перечисленных функций становится невозможным: программист не видит текст на мониторе и не может считывать информацию, а также взаимодействовать с элементами управления графических интерфейсов множества программ. Поэтому требуется создания альтернативного канала взаимодействия с компьютером. Таким каналом может выступать слух, а именно голосовое управление компьютером и воспроизведение текста программ и результата их выполнения.

В настоящее время существуют следующие способы взаимодействия программиста с нарушением зрения и компьютера: экранные дикторы, голосовые помощники, специализированное программное обеспечение и модифицированные брайлевские дисплеи.

1.2 Обзор систем взаимодействия программиста с нарушением зрения и компьютера

В данный момент на рынке представлено множество систем замены зрительного канала в комплексе человек-компьютер. Среди них в основном представлены экранные дикторы. Наиболее популярными являются программы:

NVDA, COBRA и JAWS [2]. Отдельно стоит отметить программу Grid Editor [3], которая была разработана совместно со слабовидящими и позволяет им редактировать уже написанный код, а также брайлевский дисплей, который выводит только текстовую информацию и изначально предназначен для чтения электронных книг. Каждый из представленных продуктов имеет ряд преимуществ и недостатков, рассмотрим каждый более подробно:

COBRA и JAWS – это однотипные экранные дикторы, которые работают только на Windows 7 и подходят для работы в офисных программах и обывательского взаимодействия. Также каждое обновление этих программ имеет отдельную стоимость.

NVDA – бесплатный экранный диктор с открытым исходным кодом. Поддерживает работу с большинством современных программ и подлежит модификации.

Grid Editor – специализированное программное обеспечение, которое было создано группой ученых совместно со слабовидящими программистами. Данное приложение позволяет только редактировать уже существующий код только слабовидящим.

Брайлевский дисплей – специализированное устройство, позволяющее выводить текст с помощью шифра Брайля. Изначально был разработан для чтения электронных книг, но после модификаций способен выводить текст и из сторонних редакторов. С его помощью можно только читать текстовую информацию и ничего больше.

Результаты сравнения вышеописанных продуктов приведены в таблице 2.

Стоит отметить, что NVDA умеет озвучивать кнопки на экране, что позволяет человеку определять положения курсора на экране и запускать код, но это не является отдельной функцией данного продукта. COBRA и JAWS вовсе не поддерживают работу в средах разработки и предназначены для стандартных (обывательских) задач.

Таблица 2 – Результаты сравнения систем взаимодействия программиста с нарушением зрения и компьютера

Устройство	**Чтение кода целиком**	**Чтение кода построчно**	**Чтение результата выполнения**	**Перевод курсора на определенную строку**	**Инициали-зация рабочей среды**	**Запуск и прерывание программы**
Брайлевский дисплей (Модифицированный)	да	нет	да	нет	нет	нет
Grid Editor	нет	нет	нет	да	нет	нет
NVDA	да	нет	да	нет	нет	нет
COBRA и JAWS	нет	нет	нет	нет	нет	нет
Голосовой ассистент	да	да	да	да	да	да

В результате изучения доступных приложений был сделан вывод, что аналогов голосового ассистента для программистов с нарушениями зрения на данный момент не существует. Таким образом, доминирующие на рынке экранные дикторы не в полном объеме решают поставленную задачу. Поэтому работа была посвящена задаче создания интеллектуального голосового ассистента для программистов с нарушениями зрения. При разработке которого были бы учтены все недостатки экранных дикторов и усовершенствованы механизмы взаимодействия программиста с нарушением зрения и компьютера.

1.3 Разработка требований к голосовому ассистенту

Для комфортной работы программиста с нарушениями зрения голосовой ассистент должен обладать следующими функциями:

- функция инициализация среды разработки и её настройка для дальнейшей работы;
- функция чтение кода целиком, отдельными строками, диапазонами строк;
- функция запуска и прерывания выполнения программы;

- функция перехода к заданной строке;

- чтение результата выполнения программы.

Также быть реализована возможность настройки скорости чтения текста программы и результата её выполнения.

Немаловажным является и автономная работа голосового ассистента без подключения к сети Интернет, потому что время ответа системы должна быть минимальна и не зависеть от скорости Интернета.

В качестве среды разработки, которой будет управлять голосовой ассистент был выбран PyCharm [4], потому что данная среда является одним из самых популярных и качественных профессиональных инструментов.

Глава 2 Разработка интеллектуального голосового ассистента для людей с нарушениями зрения

2.1 Изучение технологий создания голосового ассистента

Голосовой помощник – это программное обеспечение, с помощью которого пользователь может взаимодействовать с устройствами с помощью голосовых команд.

Голосовые ассистенты имеют две основные функции:

- распознавание команд;
- синтез речи.

В настоящее время создано множество различных сервисов, библиотек и моделей для распознавания голосовых команд. Среди них доступными для использования являются: pocketsphinx, vosk, SpeechRecognition, Julius, Kaldi.

В таблице 3 приведено сравнение модулей распознавания речи по параметрам: поддержка русского языка, возможность автономной работы и субъективная оценка качества распознавания русского языка.

Таблица 3 – Сравнения модулей распознавания речи

Модуль	**Поддержка русского языка**	**Автономная работа**	**Качество**
pocketsphinx	Частично	Да	Среднее
Vosk	Да	Да	Высокое
SpeechRecognition	Да	Нет	Среднее
Julius	Нет	Да	Среднее
Kaldi	Да	Да	Высокое

В ходе анализа модулей распознавания речи, установлено, что не все из них поддерживают распознавание речи на русском языке, а также достаточно точны для решения поставленной задачи [5]. Изучение документации на перечисленные библиотеки позволили выбрать два наиболее удобных и качественных модуля:

Kaldi и Vosk. Данные модули имеют простой интерфейс подключения, поддерживают распознавание русского языка, поддерживают автономную работу без подключения к сети Интернет, но Kaldi — это исследовательский набор инструментов для распознавания речи, который реализует множество современных алгоритмов. Vosk — это практичная библиотека распознавания речи, которая поставляется с набором точных моделей, сценариев и практик, и обеспечивает готовое к использованию распознавание речи для различных платформ с лучшей производительностью [6]. Поэтому для создания голосового ассистента был выбран модуль Vosk.

Далее была выбрана модель для синтеза речи. Рассмотрены следующие модели синтеза речи: gTTS, модели Silero, pyttsx3, speechkit, в таблице 4 приведена из сравнительная оценка. Одной из наиболее качественных, бесплатных и работающих автономно моделей оказалась модель от Silero, которая была успешно использована с помощью модуля PyTorch.

Таблица 4 – Сравнения модулей синтеза речи

Модуль	**Тип лицензии**	**Поддержка русского языка**	**Качество**	**Автономная работа**
gTTS	MIT License	Частично	Плохое	Нет
Silero	CC BY-NC-SA 4.0	Да (6 голосов)	Высокое	Да
pyttsx3	MPL 2.0	Да (2 голоса)	Среднее	Да
Speechkit	Yandex License	Да (15 голосов)	Высокое	Нет

Механизм Wake Word Engine — это малообъёмный алгоритм, который обнаруживает произнесение заданной фразы пробуждения в аудиопотоке. Обучение готовой к использованию модели Wake Word требует значительных усилий по обработке данных и опыта для моделирования реальных сред во время

обучения. Поэтому для оптимизации процессов анализа входящего потока речи был использован готовый модуль Picovoice. Он необходим для того, чтобы не тратить ресурсы системы на постоянное прослушивание и распознавание всего аудио, которое поступает в систему. Picovoice позволяет тратить меньше ресурсов системы и находить фразу пробуждения (слово или словосочетание, которым вызывается ассистент) в среднем всего за 30 миллисекунд, вместо 550 миллисекунд при стандартном анализе потока (Intel Core i7-10870H, 16 Гб ОЗУ). После его обнаружение запускается полноценный цикл распознавания всего потока речи.

2.2 Разработка голосового ассистента

2.2.1 Схема программного обеспечения голосового ассистента

В процессе проектирования предложена структурно-функциональная схема голосового ассистента для программиста, которая включает выбранные модули для распознавания и синтеза речи, разработанный программный модуль голосового ассистента для управления средой разработки, а также дополнительные модули для взаимодействия с аппаратным и программным обеспечением компьютера программиста. Схема представлена на рисунке 1.

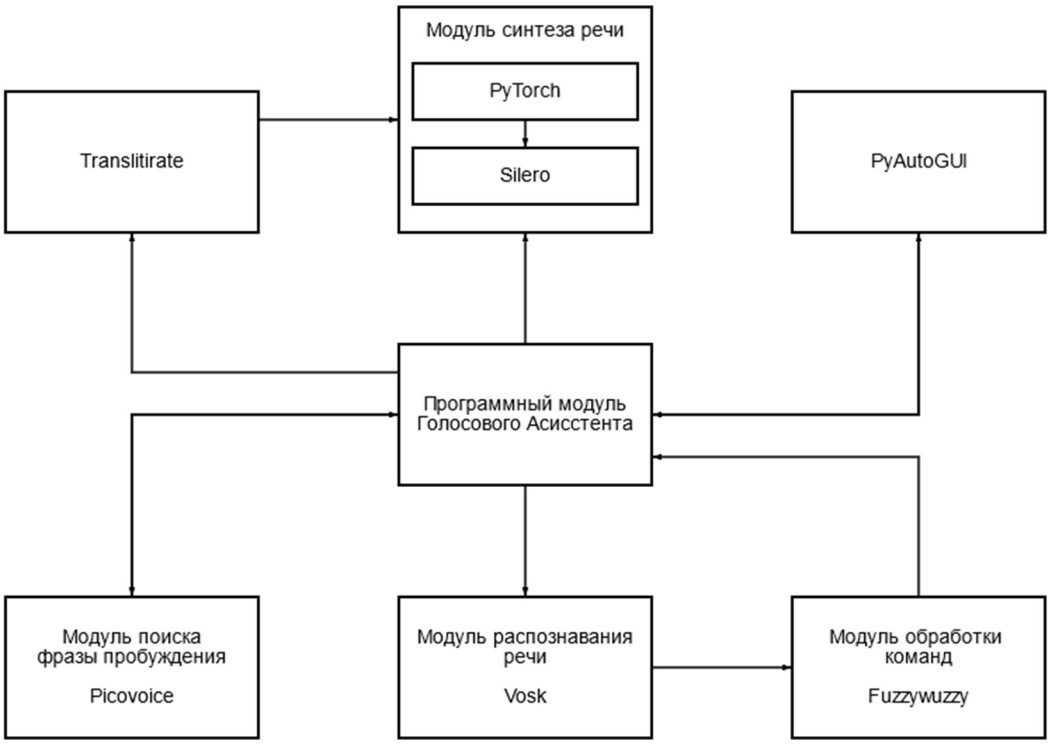

Рисунок 1 – Структурная схема голосового ассистента

Голосовой ассистент был разработан для взаимодействия программиста с нарушением зрения со средой разработки PyCharm. Были спроектированы основные модули и описаны взаимодействия между ними.

Модуль поиска фразы пробуждения получает на вход информацию с аудиовхода системы и при помощи модуля Picovoice пытается обнаружить в потоке слово «компьютер». Если оно было произнесено модуль возвращает истину.

Модуль распознавания речи включается после возвращения модулем поиска фразы пробуждения истины. Он также получает на вход информацию с аудиовхода системы и с помощью Vosk возвращает текстовую расшифровку поступающей речи.

Распознанная речь сразу поступает на вход модулю обработки команд. Последний убирает из нее все лишние слова и пытается извлечь из нее только ключевые слова и с помощью модуля Fuzzywuzzy определяет какую команду нужно выполнить, именно ее имя он и возвращает.

Модуль синтеза речи представляет из себя модуль-исполнитель, который получает на вход текстовую информацию от главного модуля голосового ассистента (или через transliterate). С помощью PyTorch и модели Silero он синтезирует речь и обеспечивает взаимодействие с пользователем.

Программный модуль голосового ассистента – представляет собой основной модуль, разработанного программного обеспечения. Он связывает между собой все остальные модули. Получая информацию от других модулей, он запускает необходимые команды и процессы.

2.2.2 Разработка модулей поиска фразы пробуждения и распознавания речи

Модуль Picovoice используется для оптимизации процессов распознавания и уменьшения задержки ответа голосового ассистента. В работе использовалась бесплатная часть модуля, поэтому в качестве фразы пробуждения использовано слово «компьютер».

Работа с модулем Vosk выполнялась на основе официальной документации [6]. К коду была подключена одна из встроенный моделей Vosk, а именно легковесная модель для распознавания русского языка, потому что при запуске голосового ассистента она быстрее подгружается и в условиях поставленной задачи не отличается по качеству распознавания.

Все команды имеют заранее определенное множество слов активаторов (хранятся в yaml файле), при распознавании которых выполняется определенные действия. Важно отметить, что используется алгоритм неточного сравнения, а именно метод расстояния Левенштейна (модуль Fuzzywuzzy), что позволяют повысить процент, верно, распознанных команд и сгладить неточности модели, которая распознает речь. Таким образом, был создан «движок» для голосового помощника, который умеет обрабатывать список команд из yaml файла и, при активации команды голосом, вызывает необходимую функцию. Список реализованных команд представлен в таблице 5.

Таблица 5 – Список команд голосового помощника

Слова активаторы	**Команда (служебное имя)**	**Краткое описание**
-новый - создай - новый проект - создай новый проект	init_new_project	Создает новый проект, настраивая IDE и конфигурируя параметры запуска
- открой	open_project	Открывает существующий проект по индексу
- тест - вперед - запуск	run_project_main_file	Запускает main.py файл в текущем проекте
- стоп - килл - уничтожить	stop_project_main_file	Останавливает main.py файл в текущем проекте
- консоль - что в логах - какая ошибка - какой вывод	read_console	Озвучивает результат выполнения программы
- все - файл - общая проверка - целиком - весь файл	tell_all_file	Озвучивает все содержимое main.py
- строки с по - диапазон с по	tell_string_range	Озвучивает диапазон строк с <число> по <число> из main.py
- строка - строка номер - скажи строку	tell_string	Озвучивает строку номер <число> из main.py
- перейди к строке	go_to	Переводит курсор на строку номер <число> в main.py

2.2.3 Разработка модуля синтеза речи

При реализации модуля синтеза речи были предложены и реализованы следующие решения:

1. Бесплатная модель синтеза речи работала одновременно лишь с одним языком и для того, чтобы голосовой помощник говорил на русском языке и озвучивал код пришлось транслитерировать весь код с помощью модулей transliterate и num2words;

2. Все специальные символы и пробелы пришлось заменять на слова «вручную», для чего был создан словарь с условными обозначениями;

3. Для чтения результата выполнения кода было необходимо дублировать его в файл. В официальном руководстве PyCharm имеются настройки, отвечающие за это, и они были внесены на автоматическое выполнение для команды инициализации [4];

4. Для того, чтобы озвучивалась всегда актуальная версия файла пришлось изменить время бездействия до сохранения. При этом последовательность действий для соответствующей настройки была автоматизирована и добавлена в цикл инициализации.

Для того, чтобы помощник озвучивал как отдельные строки, так и их группы был создан алгоритм получения чисел из речи: с помощью модуля words2numrus все найденные числа в команде переводились в целочисленный тип и становились аргументами функции.

2.2.4 Разработка модуля взаимодействия со средой разработки

В данном модуле реализован цикл-обработчик, в котором при нахождении слова-отклика запускается цикл распознавания речи и последующий обработчик расшифрованных команд. Схема работы приведена на рисунке 2.

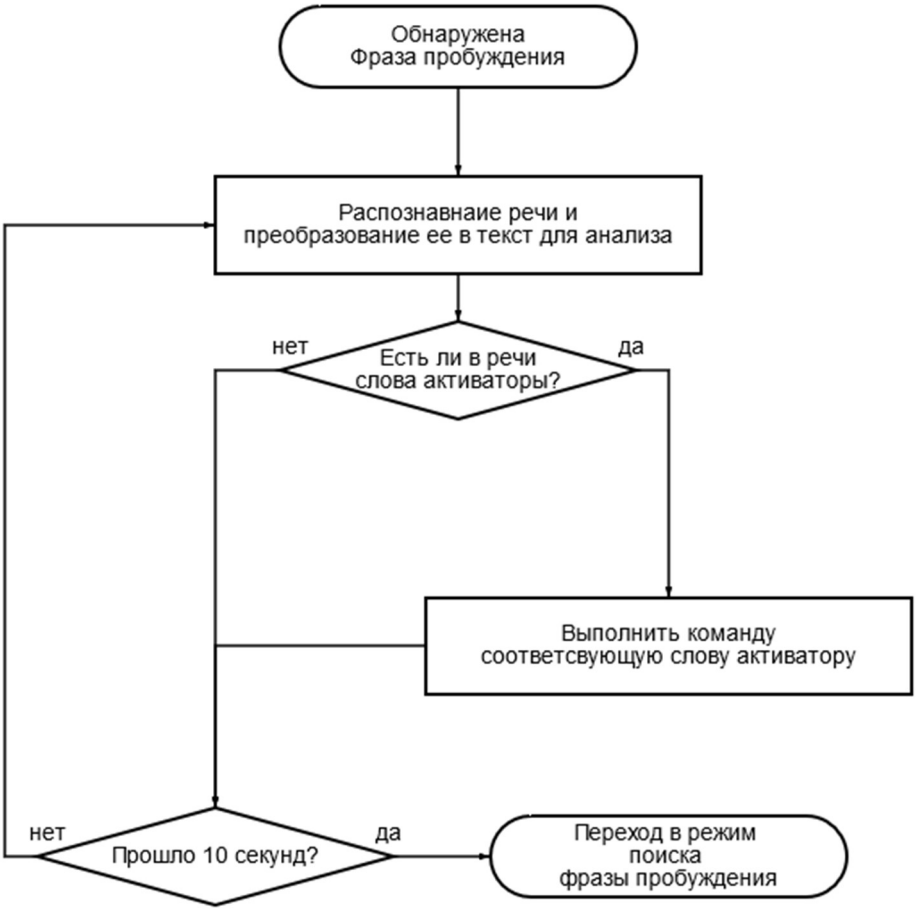

Рисунок 2 – Схема работы основного цикла программы

Одной из важных функций является перевод курсора на заданную строку. Получение из речи целочисленного аргумента осуществлялось также, как и в случае с номером строки для озвучания.

Далее были рассмотрены два возможных решения задачи:

1. Использование горячих клавиш PyCharm вместе с модулем pyautogui, который может по картинке-эталону нажать на необходимые кнопки на экране (ctrl + g далее ввести номер строки и нажать кнопку ok), но он был почти сразу отброшен по причине своей нестабильности.
2. С помощью модуля pyautogui использовать горячие клавиши ctrl + home для перехода на первую строку файла и далее переходить к необходимой строке посредством автоматического нажатия стрелки вниз.

В результате анализа был выбран второй метод, как более стабильный и универсальный метод.

Функции запуска и остановки кода были реализованы через горячие клавиши и модуль pyautogui.

После отладки всех вышеперечисленных функций началась реализация функции инициализации новых файлов. Основной проблемой стала частичная несовместимость PyCharm с работой из командной строки. При попытках запуска PyCharm через консоль IDE теряла связь с интерпретатором и/или выдавала ошибку при загрузке. Поэтому был придуман новый способ настроить IDE для корректной работы ассистента. Так с использованием pyautogui PyCharm запускается из начального экрана и далее с помощью горячих клавиш самой IDE и автоматическому нажатию всех необходимых кнопок настраивается окружение и особые пункты настроек, например, время бездействия до сохранения.

После написания кода и его отладки в ходе комплексного тестирования голосового помощника была настроена пауза между произносимыми им словами. Результаты исследования приведена в разделе 3.2.

Глава 3 Апробация интеллектуального голосового ассистента для программистов с нарушением зрения

3.1 Исследование времени поиска синтаксической и логической ошибок при отладке кода

Для тестирования разработанного голосового ассистента предложено провести исследование времени поиска и исправления синтаксической и логической ошибок в коде.

В тестировании приняли участие 3 добровольца, которые имели опыт в программирования на языке Python. Им было предложено отладить уже написанный код в смоделированной ситуации при помощи ассистента. Для этого яркость экрана компьютера была снижена до минимума, а сам монитор был закрыт листом бумаги, который почти не пропускал свет, тем самым были смоделированы условия близкие к уровню видимости слабовидящих.

В качестве задачи была использована задача «FizzBuzz» [7], в решении которой намерено были совершены ошибки: был нарушен синтаксис условного оператора и поменяны местами строки «Fizz» и «Buzz». Результаты тестирования представлены в таблице 6.

Таблица 6 – Время прохождения теста

	Время прохождения теста, с
Испытуемый 1	1575
Испытуемый 2	1357
Испытуемый 3	1487
Среднее для всех испытуемых	1473

Среднее время, затраченное на успешное прохождение теста, составило 24 минуты 33 секунды, что в условиях отсутствия опыта работы вслепую у тестируемых, можно считать хорошим результатом, учитывая, что время решения такой задачи без голосового ассистента для программистов с нарушением зрения стремится к бесконечности.

3.2 Исследование длительности паузы между словами

В разработанном программном обеспечении реализована возможность настройки скорости воспроизведения текста. Выполнено исследование влияние длительности паузы между словами при синтезе речи на результат отладки программы с ошибкой.

В тестировании приняли участие семь добровольцев - программистов с разным опытом в разработке на языке программирования Python. Им было предложено ответить на пять вопросов, представляющих из себя короткие программы с синтаксическими ошибками и(или) несоблюдением стандартов PEP8.

Каждый вариант был пройден каждым участником на четырех различных скоростях воспроизведения, а именно: без паузы между словами (сплошной поток речи), с паузами 0,25; 0,5 и 0,75 с

Затем был проведен анализ полученных результатов тестирования несколькими способами.

Для анализа был введен коэффициент прохождения, который представляет собой отношение оценки прохождения теста в баллах от 0 до 8 и времени, затраченного на его прохождение, умноженное на коэффициент k=100.

В первом случае были посчитаны все коэффициенты для каждого прохождения, а также подсчитаны средний балл и время прохождения для каждого значения паузы, используя их был подсчитан соответствующий паузе средний коэффициент. Мы стремимся максимизировать коэффициент прохождения, потому что соответствующая ему пауза будет оптимальной и при ней программист будет совершать малое количество ошибок при минимальных затратах времени. Максимальный коэффициент при таком подходе будет соответствовать паузе в 0,5 с. Результаты анализа представлены таблицах 7 – 10, а основываясь на них был простроен график зависимости (см. рисунок 3).

Таблица 7 – Результат тестирования при отсутствии пауз

Пауза 0,00 с			
Испытуемый №	**Балл**	**Время**	**Коэффициент**
1	2	81,21	2,46
2	1	114,63	0,87
3	0	145,27	0,00
4	3	93,25	3,22
5	1	98,41	1,02
6	0	93,96	0,00
7	3	87,47	3,43
Средний	1	102,03	0,98

Таблица 8 – Результат тестирования при паузах в 0,25 с

Пауза 0,25 с			
Испытуемый №	**Балл**	**Время**	**Коэффициент**
1	6	158,94	3,78
2	5	158,88	3,15
3	2	159,07	1,26
4	7	157,42	4,45
5	6	157	3,82
6	4	166,38	2,40
7	6	152,01	3,95
Средний	5	158,53	3,15

Таблица 9 – Результат тестирования при паузах в 0,50 с

Пауза 0,50 с			
Испытуемый №	**Балл**	**Время**	**Коэффициент**
1	7	169,41	4,13
2	6	179,86	3,34
3	4	175,11	2,28
4	8	170,05	4,70
5	5	181,02	2,76
6	6	171,51	3,50
7	8	172,56	4,64
Средний	6	174,22	3,44

Таблица 10 – Результат тестирования при паузах в 0,75 с

Пауза 0,75 с			
Испытуемый №	**Балл**	**Время**	**Коэффициент**
1	8	183,84	4,35
2	6	182,68	3,28
3	5	185,56	2,69
4	8	189,19	4,23
5	6	190,27	3,15
6	6	193,44	3,10
7	8	183,81	4,35
Средний	6	186,97	3,21

Рисунок 3 – График зависимости среднего коэффициента от паузы.

Второй способ представляет из себя оценку распределения баллов. Построив диаграмму (см. рисунок 4) и линии тренда (линейные аппроксимации методом наименьших квадратов [8]), удалось установить, что наиболее стабильной оказалась пауза в 0,5 с, потому что графически данная прямая имеет наименьший уклон и почти параллельна оси абсцисс, что говорит о независимости результата

от опыта испытуемого, при данной паузе испытуемые проходят тест наиболее стабильно.

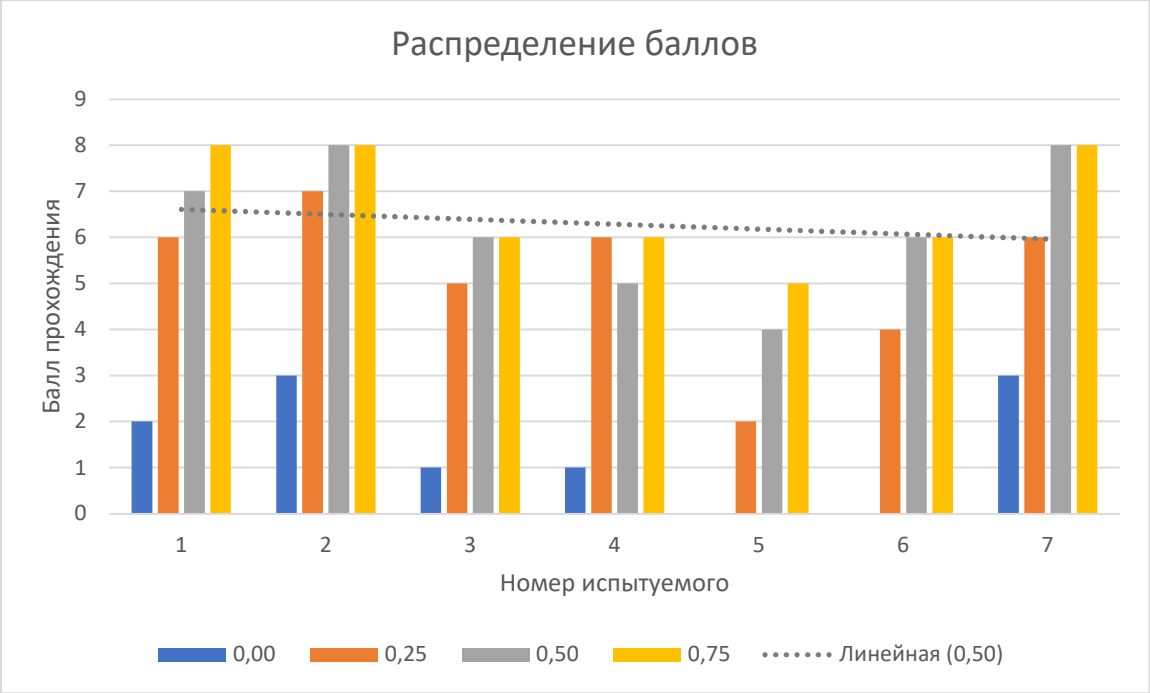

Рисунок 4 – Диаграмма распределения баллов

Полученные в результате исследования данные позволяют оптимально настроить голосового ассистента для программиста с нарушением зрения, а именно при паузе 0,5 с программист будет совершать малое количество ошибок при минимальных затратах времени на прослушивание синтезированного текста.

Результаты и выводы

В работе исследован процесс взаимодействия программиста и компьютера, выявлены базовые операции, которые необходимо выполнять программисту для написания и отладки кода. Изучены инструменты и программные продукты, которые используются программистами с нарушением зрения при работе на компьютере, выявлены их недостатки и ограничения применения при программировании. Обоснована актуальность создания и сформулированы требования к интеллектуальному голосовому ассистенту для программистов с нарушениями зрения.

В работе изучены технологии и методы создания голосовых ассистентов, проведено сравнение наиболее популярных модулей, обосновано применение модели Silero и модуля PyTorch для синтеза речи, модуль Vosk для распознавания речи, интегрирована технология Wake Word Engine посредством модуля Picovoice, для оптимизации процессов анализа входящего потока речи

Спроектировано и разработано программное обеспечение интеллектуального голосового ассистента для программистов с нарушением зрения.

Апробация интеллектуального голосового ассистента для программистов с нарушением зрения на примере с поиском логической и синтаксической ошибки при отладке кода в смоделированных условиях (близкие к видимости слабовидящих) показала, что на решение данной задачи у программиста в среднем уходит 24 минуты. Исследования влияния паузы между словами позволило установить, что при паузе 0,5 с программист будет совершать малое количество ошибок при минимальных затратах времени на прослушивание синтезированного текста.

Развитие проекта предполагается вести в следующих направлениях:

1. Разработать установщик с полным озвучиванием для людей с нарушением зрения.

2. Реализовать возможность работы с несколькими файлами параллельно для обеспечения поддержки работы с приложениями со сложной разветвленной структурой.

Список литературы

1) Статистика слепых и инвалидов по зрению в России, СНГ и мире // тифлоцентр "Вертикаль" URL: https://tiflocentre.ru/stati/kolichestvo-slepyh-i-invalidov-po-zreniju-v-Rossii.php (дата обращения: 28.12.2023).
2) Слепые для слепых: история программ для чтения с экрана // Хабр URL: https://habr.com/ru/companies/cloud4y/articles/678162/ (дата обращения: 29.12.2023).
3) Ученые разработали новый инструмент для чтения и редактирования кода. Он поможет слепым и слабовидящим программистам // сайт виси ру URL: https://vc.ru/future/536409-uchenye-razrabotali-novyy-instrument-dlya-chteniya-i-redaktirovaniya-koda-on-pomozhet-slepym-i-slabovidyashchim-programmistam (дата обращения: 18.10.2023).
4) Getting started // сайт PyCharm от JetBrains URL: https://www.jetbrains.com/help/pycharm/getting-started.html (дата обращения: 24.06.2023).
5) Беленко М.В. СРАВНИТЕЛЬНЫЙ АНАЛИЗ СИСТЕМ РАСПОЗНАВАНИЯ РЕЧИ С ОТКРЫТЫМ КОДОМ / М.В. Беленко, П.В. Балакшин // Международный научно-исследовательский журнал. - 2017. - №4 (58). - URL: https://research-journal.org/archive/4-58-2017-april/sravnitelnyj-analiz-sistem-raspoznavaniya-rechi-s-otkrytym-kodom (дата обращения: 05.01.2024). - doi: 10.23670/IRJ.2017.58.141
6) VOSK documentation // сайт VOSK URL: https://alphacephei.com/vosk/ (дата обращения: 20.06.2023).
7) Fizz buzz // Википедия URL: https://ru.wikipedia.org/wiki/Fizz_buzz (дата обращения: 02.01.2024).
8) А. И. Карасев Теория вероятностей и математическая статистика. - 5-е изд. - М.: Статистика, 2015. - 279 с.
9) Исследование по доступности сайтов и мобильных приложений для людей с проблемами зрения // Информационный портал «Особый взгляд» URL: https://specialviewportal.ru/study (дата обращения: 28.12.2023).

10) PyAutoGUI documentation // Документация модуля PyAutoGUI URL: https://pyautogui.readthedocs.io/en/latest/ (дата обращения: 17.06.2023).
11) Не опускать руки: советы людей, потерявших зрение в зрелом возрасте // Информационный портал «Особый взгляд» URL: https://specialviewportal.ru/articles/articles1511 (дата обращения: 28.12.2023).