**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний університет кораблебудування імені адмірала Макарова**

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"
Завідувач кафедри ІУСТ
_____
к.т.н, доц. Михелєв І.Л.
"____" _____ 2021 р.

# *КВАЛІФІКАЦІЙНА РОБОТА*

**на здобуття ступеня вищої освіти "магістр"**

**на тему:** Дослідження методів і алгоритмів розпізнавання мови та розробка системи голосового управління

Виконала: студентка 6145 м групи
_____ ***Заіченко Т.С.***
(підпис)
Керівник роботи:
доцент кафедри ІУСТ, к.т.н.,
(посада, науковий ступінь вчене звання)
_____ ***Книрік Н.Р.***
(підпис)

м. Миколаїв – 2021 р.

**ЗМІСТ**

# ВСТУП

На сьогодення світ стає все більше технічно розвинутий та не стоїть на одному місці. Кожен день винаходять нові технології та розробки. Однією з таких розробок став голосовий помічник, історія створення якого нараховує близько 60 років, та не має наміру зупиняти вивчення цієї технології та її розвитку. Якісних продуктів в цій сфері було створено не багато, однак досить ідеального продукту створено ще не було.

З розвитком цієї системи стає зрозуміло, що використання цих систем буде мати більш ширший спектр використання, якщо людина буде вести діалог з комп'ютером під час роботи, а саме, стане можливим керувати машиною за допомогою голосу в реальному часі, а також введення та виведення інформації у вигляді звичайної людської мови.

Голосові помічники приносять багато користі, адже вони прості в застосуванні та спроможні економити час користувача на пошук інформації в ситуаціях коли потрібна мультизадачність користувача в роботі, тому на даний момент часу вони дуже поширені через ряд їх переваг:

– виконання простих запитів (час, дата, ввімкнення/вимкнення ПК і т.д.);

– можливість в майбутньому замінити людей на небезпечних для їх здоров'я підприємствах використовуючи штучний інтелект голосового помічника в якості робота помічника, котрий буде виконувати не однотипну а роботу різних типів, та переходити з одного підприємства на інше не змінюючи їх основну програму;

– використання голосового помічника на зустрічах для автономності процесу представлення продукту (відкриття/знаходження файлу, презентація, відповіді на запитання);

– можливість використовувати помічника не залежно від типу пристрою (ПК, Android, IOS).

Не дивлячись на всі переваги голосового помічника, основною проблемою є реалізація «живого» діалогу людини та технічних засобів.

На даний момент тема створення та розвитку голосового помічника є актуальною, оскільки все більше технологій можуть працювати с голосовим помічником. Так наприклад більшість браузерів, телефонів, ПК можуть керуватись за допомогою голосу, однак окрім невеличких технологій до яких звикли всі, голосового помічника використовують також і в автомобілях. Тобто під час поїздки водію не потрібно відволікатись від дороги використовуючи лише запити до асистента, а в більш сучасних машинах які використовують штучний інтелект, та не потребують втручання водія в процес, можуть використовувати голосового асистента, як голосове управління, тобто видача запитів які не стосуються процесу водіння, або запити стосовно зміни режиму роботи машини, зміна маршруту, зупинки і т.д.

Тож основною метою дипломної роботи є розробка системи голосового управління ПК за допомогою Python. Для досягнення поставленої мети, необхідно виконати такі задачі:

– проаналізувати існуючі проблеми роботи голосового помічника;

– виконати порівняльний аналіз існуючих рішень голосового управління ПК;

– аналіз структури типової технології голосового управління ПК;

– розробити узагальнену структуру та алгоритмічну базу технології голосового управління ПК;

– реалізувати тестовий застосунок голосового асистента та підтвердити його працездатність.

# РОЗДІЛ 1. ОСОБОЛИВОСТІ ПРИЙМАННЯ ТА ОБРОБКИ АУДІОДАНИХ ЗА ДОПОМОГОЮ ГОЛОСОВОГО АСИСТЕНТА

## 1.1. Особливості обробки даних за допомогою голосового асистента

Голосовий помічник – це робот який працює за допомогою штучного інтелекту та використовує технологію розпізнавання голосу та обробки мови, щоб відповідати на запитання, вести розмови, виконувати прості завдання. З появою такої опції, виконувати більшість задач стало набагато простіше та зручніше. Голосовий асистент побудований на базі штучного інтелекту, технології машинного навчання та розпізнавання голосу.

Ще з 1930 року вчені намагались розпізнати голос за допомогою технологій, тоді це були машині запрограмовані на розпізнавання простих звуків, а зараз наші смартфони та комп'ютери розуміють цілі речення. Цей скачок відбувся в результаті розвитку машинного навчання, яке заставляє нейроні мережі самостійно аналізувати контекст та ефективно знаходити основне джерело звуку. Однак лише на цю технологію вчені витратили 80 років [1].

Принцип роботи голосового асистента такий же простий, як і його використання:

1. пасивне зчитування звуку, за допомогою активації функції кодовим словом;
2. фільтрація сигналу – етап фільтрації звуку від шуму та звукових перешкод, котрі з'являються під час запису голосового запиту;
3. оцифровка звуку – виконується перетворення звукового сигналу в цифровий вид, який буде зрозумілий комп'ютеру;
4. аналіз сигналу – виділяються частини з голосом та виконується оцінка параметрів, таких як частина мовлення, форма слова, зв'язок в один запит;
5. пошук шаблонів в даних – штучний інтелект збирає різні раніше почуті слова, порівнює з шаблонами та видає результат [2].

Функціональність використання голосових асистентів базується на рішенні простих задач. В результаті частого використання голосовий асистент

запам’ятовує функції, котрі користувач використовує частіше та намагається облегшити подальшу роботу програми. Так як всі помічники мають штучний інтелект, то при спілкуванні вони використовують дані зміни місцезнаходження, час доби, дні тижня, історію пошукових запитів, різноманітні попередні запити та інше [3].

Розглянемо узагальнений функціонал голосового помічника. Більшість систем мають широкий набір корисних можливостей., наприклад:

– виконання дзвінків по заданим номерам;
– пошук різноманітної інформації;
– ввімкнення/вимкнення системи, переведення системи в режим сну;
– відкриття/закриття застосунків на телефоні або на ПК;
– ввімкнення музики, фільмів та іншого контенту;
– активація будильника, створення заміток;
– написання та відправка текстових повідомлень;
– ведення розмов з голосовим асистентом;
– відображення прогнозу погоди та ін [4].

Умовно віртуальні помічники можна розділити на декілька видів:

– для смартфонів – перед початком використання, голосового помічника необхідно встановити на пристрій. Голосовий помічник може керувати деякими функціями смартфону або набирати за допомогою деяких систем номера;

– для комп’ютера – в цю категорію входять голосові помічники, за допомогою яких можна керувати елементами введення або відривати різноманітні програми. Також можна використовувати пошук різноманітної інформації в браузері;

– для будинку – цей варіант використовується для взаємодії з екосистемами «Розумний будинок». В більшості випадків віртуальний помічник має вигляд портативної колонки, яка досить компактна та мобільна, адже її можна встановити в будь-якій точці будинку.

Перед вибором голосового помічника необхідно врахувати досить багато нюансів. Для початку варто ознайомитись з популярними системами, а також

проаналізувати переваги та недоліки кожної з них. До ключових критеріїв вибору можна віднести:

– сумісність – для кожної операційної системи є свої віртуальні асистенти. Наприклад, для Windows з великим попитом використовується Cortana. Для гаджетів Apple розроблена система Siri, яка має широкий спектр функцій та високу швидкість пошуку інформації. Однією з самих популярних для Android вважається Аліса.

– функціональні можливості – в цьому випадку багато чого залежить від особистих бажань користувача. Одна система може краще виконувати пошук інформації, інша – більш проста в керуванні ПК або смартфоном. Є навіть помічники, які виступають в якості розвинутої навігаційної системи для полегшення подорожей.

– рівень «людяності» – багато систем мають досить високий рівень штучного інтелекту, що на деякому рівні можуть імітувати розмову з живою людиною. В якості прикладу можна привести помічника Amazon Alexa, який може жартувати та дотепно відповідати на різноманітні питання [5].

## 1.2. Порівняльний аналіз існуючих рішень технології голосового асистента

Голосовий помічник є високотехнологічною системою управління, яка може реагувати на голосові команди. Тому все більше компаній намагаються створити свого асистента, який приверне увагу їх клієнтів та зможе полегшити їм життя. Основною особливістю помічника є його розвинутий штучний інтелект та самонавчання, що дозволяє йому розуміти голос користувача та виконувати указані дії. Тобто головною задачею голосового помічника є максимально швидке виконання запитів та їх обробка. Але перед створенням голосового помічника, будь-яка компанія проаналізує вже існуючі технології, тому розглянемо самі поширені технології голосового управління та проаналізуємо їх переваги та недоліки.

Аліса, Alexa, Siri, Cortana – всі ці жіночі імена були вигадані та дані голосовим помічникам від найбільших компаній. З цього списку відрізняється лише Google, який дали своєму помічнику назву «Асистент», що показує їх серйозних та одночасно простий підхід до створення помічника, адже цю назву користувачам легко запам’ятати. А тепер проаналізуємо можливості кожного з цих представників, якій щороку займають топи кращих голосових асистентів [6].

Почнемо з Аліси, котру створила компанія «Яндекс» в 2017 році. Головна перевага цього голосового користувача в порівнянні з іншими конкурентами – робота на Android, IOS і навіть десктопах. Аліса спроможна вести діалог, знаходити інформацію по розпізнаному запиту та цитувати результат пошуку. Більш того, це ще й штучний інтелект, який розвивається за рахунок розширення мази знань. З кожним днем ця програма спроможна знаходити все більше і більше актуальної інформації та все частіше може озвучувати все своїм голосом.

Аліса, як сервіс від Яндекс, має ряд переваг:

– можна виконувати пошук без допомоги рук. Достатньо просто надиктувати запит голосом та прослухати відповідь;

– програма постійно розвивається, поповнюючи власну базу даних;

– може підтримувати живу бесіду, нехай і з обмеженим набором речень, але це лише питання часу, а також подальшого розвитку;

– для розпізнання голосу використовується технологія SpeechKit, яка дозволяє розділяти голоси за акцентом, діалогом та тембром;

– також використовується технологія Turning. Завдяки їй виконується «осмислення» розпізнаного запиту та пошук відповідей. При цьому відповідь формується в залежності від геолокації. Саме тому різні користувачі отримують унікальні відповіді;

– може перевірити ціни в магазині, замовити їжу та таксі;

– може ввімкнути музику, відео або навіть прочитати казку;

– в технології вибору методу відповіді на базі отриманих даних, використовуються фільтри, які виключають злість та зухвалість, а також розмови на небажані теми. Аліса говорить тільки по справі, а коли тема доходить до небажаного рівня, плавно уходить від відповіді, починаючі нову тему.

Окрім стандартного функціоналу, Аліса може також замінити вам будь яку людину в розмовах, адже вона добре розуміє людську мову та може відповісти на будь яке звернення до неї, або навіть смішно пожартувати.

Однак Аліса має один недолік, який сильно вплинув на її роботу. Нажаль сервіси Яндекс, в тому числі й Аліса, не працюють на території України через указ президента України №133 від 28 квітня 2017 року «Про застосування персональних спеціальних економічних та інших обмежувальних заходів (санкцій)», тому використання Аліси на території України заборонене та неможливе.

Що ж стосується збору даних, то отримати їх, при чому про кожного користувача, не так й складно, тим паче люди самі розповсюджують інформацію про себе реєструючись на сумнівних інтернет-сервісах.

Тепер розглянемо Siri, котра була створена компанією Apple в серпні 2011 року. У голосового помічника Siri є деякі особливості, які більшість користувачів ігнорують через їх незнання:

– Siri дозволяє налаштувати демонстрацію повідомлень від різноманітних додатків та сайтів;

– за допомогою цього голосового помічника можна відфільтрувати файли по даті. Наприклад, можна попросити Siri показати замітки, які були створені у вказаний день;

– якщо ви не можете обрати між двома варіантами, попросіть Siri кинути монетку або обрати карту;

– в голосового помічника Siri влаштована функція розпізнання музики (як у застосунку Shazam). Щоб дізнатись яка пісня зараз грає, просто задайте Siri це питання;

– в більшості випадків телефоном можна керувати через голосові команди;

– з нею можна завести бесіду, адже вона добре розуміє людську мову та кожен рік розширює свій словниковий запас;

– новини, погода, кіно, телевізійна програма – все це вона може, крім того в неї є можливість керувати деякими елементами розумного будинку [7].

Однак деякі користувачі відмічають, що робота Siri іноді не дуже швидка, якщо порівнювати її з іншими голосовими помічниками. Також погано працює інтеграція з іншими сервісами, можливо розробники вважають, що для Siri цього достатньо. Також на початку свого шляху Siri важко давалось розуміння людського мовлення, тому в мережі дуже багато прикладів дивакуватих відповідей від Siri. Також серед недоліків можна відмітити те, що Siri не вміє відрізняти користувачів одне від одного та іноді її ставлять в тупик найпростіші питання. Вона не може поставити одразу декілька таймерів, що в деяких випадках користувачам вважається необхідним.

Перейдемо до Amazon, який створив Alexa в листопаді 2014-го року. Головна відмінність Alexa від інших подібних проектів – це те, що вона заточена під управління системами розумного будинку. Вона може керувати роботами-пилососами, освітленням та багатьма іншими функціями.

Раніше Alexa була зосереджена лише на керуванні розумним будинком, але тепер додаток з помічником працює й на смартфонах. Серед переваг також можна виділити:

– швидкий розвиток системи. За декілька років функціонал виріс с 5 до 15 тисяч можливих сценаріїв роботи;

– широка можливість ітерації з розумними приборами та віртуальними сервісами завдяки відкритому API;

– можливість будь-якому новачку-програмісту створювати власні сценарії для голосового управління Alexa;

– моніторинг погоди, новин та затор;

– можливість замовляти товар або їжу онлайн;

– добре розумію людську мову та може підтримати бесіду.

Але серед недоліків також відмічають відсутність можливості керувати голосовим асистентом використовуючи російську або українську мови. Єдине, що можливе на сьогодні – це прослуховувати новини на рідній мові. Також основною проблемою Alexa є реклама під час розмов, не перериваючи діалогу, вона може вам порадити купити щось, на Amazon або магазинах-партнерах, декілька разів на день [8].

Cortana є технологією від Microsoft, вона була створена в квітні 2014 року. Сьогодні вона себе відмінно відчуває не лише на платформі Windows, а також й на Android та IOS. Сама компанія Microsoft позиціонує Cortana, як голосового помічника, який допоможе вам виконувати більше справ, затрачаючи на них менше часу. Наприклад, вона може організувати роботу вашого ПК з Windows 10 або буде керувати приладами розумного будинку.

Розглянемо її переваги, як голосового помічника:

– інтегрована під всі види пристроїв, навіть під консоль Xbox One;

– дозволяє задавати запити у вигляді текстового повідомлення, наприклад якщо ви на важливій зустрічі або ваш голосовий запит не був розпізнаним, то текстові запити будуть зручніші;

– у Cortana є унікальна функція запам’ятовування «Записна книжка», котру Microsoft додали до програми після розмови з реальними особистими помічниками. Як було помічено більшість голосових помічників можуть надати вам потрібну інформацію, наприклад результати матчу вашої улюбленої команди, але про наступні матчі ні один з голосових асистентів не згадає, й не нагадає вам;

– відслідковує ваші посилки та поїздки. Вона може сканувати електронні листи для отримання інформації про рейси, а далі вона буде постійно тримати вас в курсі всіх ваших планів.

Серед недоліків я би виділила саме використання нею власну пошукову систему Bing. Таким чином Microsoft вирішила продемонструвати свою незалежність від інших компаній. Однак, я, як користувач, надаю перевагу саме тій пошуковій системі, якою я користуюсь довше та частіше, адже це питання зручності та естетичності. Також 31 грудня 2020 року Microsoft вирішив припинити роботу Cortana як мінімум в трьох країнах: Австралії, Канаді та Великобританії. З чим пов’язане це рішення поки що невідомо, так як й з іншими країнами.

Також недоліком Cortana є проблема з налаштуванням її роботи в деяких регіонах. Для того щоб ввімкнути її на своєму ПК необхідно зробити велику кількість маніпуляцій, щоб вона почала працювати. В деяких країнах немає

підтримки Cortana, через що необхідно підключати сусідні регіони які мають підтримку голосового помічника [9].

Google Assistant як проект з'явився в травні 2016 року. Цей голосовий помічник не має свого власного імені, він просто є Асистентом для користувача й нічим більше.

Серед переваг Google Assistant можна виділити такі особливості:

– можна виконати швидкий пошук інформації використовуючи голосовий запит після чого можна прослухати коротку інформацію по результатам пошуку;

– знаходить місця поблизу за критеріями, після чого асистент покаже декілька варіантів з яких можна обрати необхідне місце;

– функція запам'ятовування інформації наданої користувачем, під час розмови з користувачем асистент запам'ятовує необхідні факти, та при подальших запитах враховаю надану вами інформацію;

– влаштований перекладач, якщо під час запиту вказати слово та мову на яку необхідно перекласти, асистент автоматично виконає цю команду не переходячи на інші сервіси або сайти, що дуже зручно, оскільки займає мало часу;

– Google Assistant може прочитати вірш, розповісти жарт або зіграти з користувачем, якщо він попросить про цю послугу [10].

Як недолік, можна виділити те, що асистент на даний момент має повний функціонал лише на смартфоні, тоді коли на ПК він виконує лише голосовий пошук.

Підсумовуючи всю отриману інформацію в таблиці 1.1 ми можемо спостерігати, що розглянуті голосові асистенти можуть керувати розумними пристроями та будинком, взаємодіяти з іншими додатками, а також всі асистенти працюють на платформі iOS. Однак є й недоліки, які варто згадати, а саме:

– Siri може працювати лише з продукцією від компанії Apple, через що стає недоступною для користувачів Android та іншої комп'ютерної техніки не від компанії Apple. Окрім цього, більшість користувачів скаржаться на погане розуміння мовлення, через що Siri може надати некоректну інформацію.

– Google Assistant містить в собі багато функцій та може приймати запити навіть без підключення до інтернету, але повний функціонал він має лише на смартфонах. Тобто використати асистента на комп'ютері буде неможливо, адже це не передбачено, але можна виконати пошук інформації за допомогою голосового запиту, перед цим зайшовши на сайт компанії.

– Cortana також може працювати без підключення до мережі Internet, а також вона працює, як на комп'ютері так й на смартфонах. Однак великим недоліком є те, щоб використати Cortana як помічника, спочатку необхідно буде виконати деякі маніпуляції, щоб вона почала працювати на комп'ютері користувача.

– Alexa не використовується для роботи на комп'ютері, але працює на смартфонах та на розумних пристроях. Та вона має два вагомих недоліки – це недоступність на території України та постійна реклама. Окрім цього Alexa не може працювати без інтернету, оскільки всі запити користувачів оброблюються за допомогою серверів [11].

– Аліса навіть по розглянутим перевагам та недолікам є фаворитом для більшості людей, яким пощастило користуватися нею. Вона має великий спектр функцій, працює на комп'ютерах та смартфонах, а сервіси Яндекс створюють можливість виконати будь який запит, але Аліса не може працювати без підключення до інтернету та користувачі не мають доступу до даного помічника на території України.

Таблиця 1.1 Функціонал відомих голосових асистентів

| Голосовий асистент | Siri | Google Assistant | Cortana | Alexa | Аліса |
|---|---|---|---|---|---|
| Компанія представник | Apple | Google | Windows | Amazon | Яндекс |
| Управління розумним будинком/пристроями | Так | Так | Так | Так | Так |
| Добре розуміння людського мовлення | Ні | Так | Так | Так | Так |
| Доступність в Україні | Так | Так | Ні | Ні | Ні |

| | | | | | |
|---|---|---|---|---|---|
| Можливість взаємодіяти з іншими додатками | Так | Так | Так | Так | Так |
| Можливість працювати без підключення до мережі Internet | Так | Так | Так | Ні | Ні |
| Платформи | iOS | Android, iOS | Android, iOS, PC | Android, iOS | Android, iOS, PC |

За даними компанії Futuresource Consulting, котра проаналізувала ринок голосових помічників, була складена діаграма популярності голосових асистентів по світу (рис.1.1)

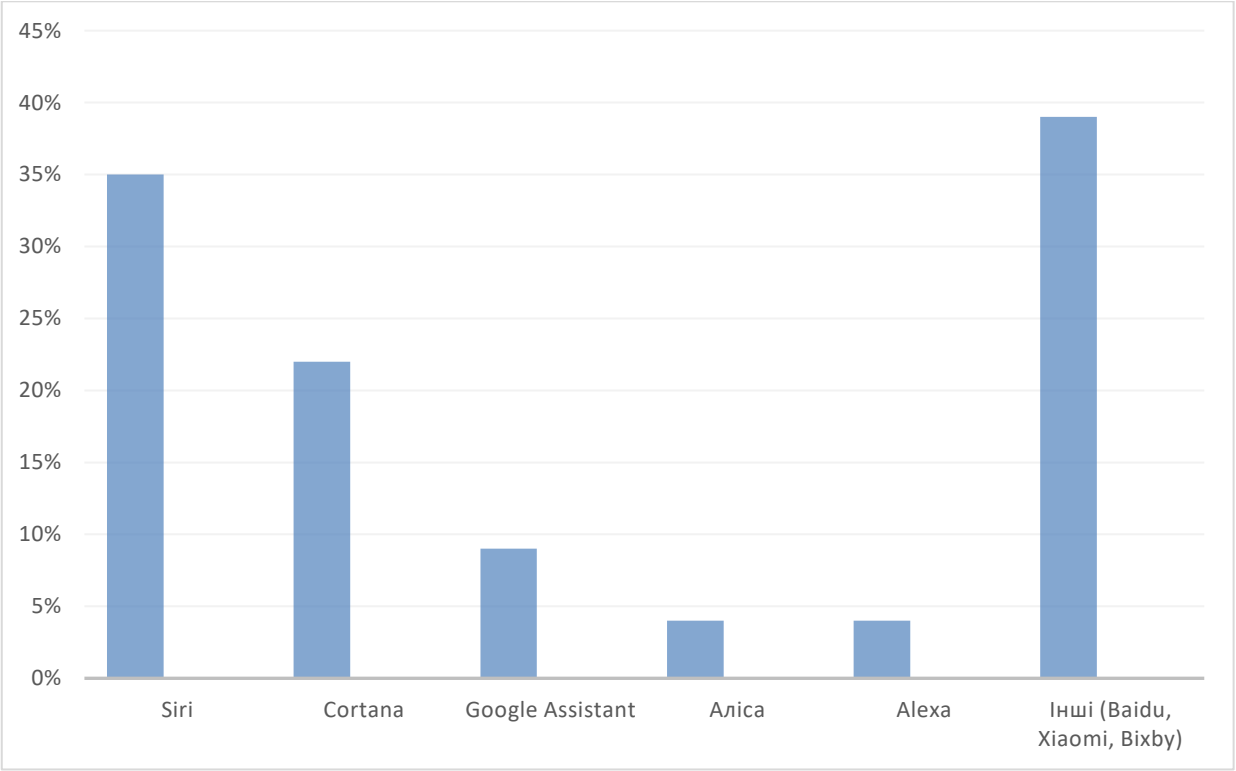


Рисунок 1.1 Діаграма популярності голосових асистентів в світі

Тож найпопулярнішими є Siri (35%) та Cortana (22%). Вони отримали таку популярність через те, що Siri встановлена на всі операційні системи Apple, хоча її успіх пов’язаний в основному через популярність iPhone. Cortana встановлена на всі комп’ютери з Windows 10, яка являється самою популярною операційною системою в світі.

Google Assistant (9%), Аліса (4%) та Amazon Alexa (4%) мають такі низькі показники через те, що вони встановлюються на відносно невеличку кількість пристроїв, але й починають набирати свою популярність.

Що стосується інших голосових асистентів, таких як Baidu, Xiaomi, Bixby та ін., то вони всі разом ділять між собою загальних 39%. В цілому більшість цих голосових асистентів використовують люди з Китаю та Азії.

Також необхідно звернути увагу на причини використання таких цифрових помічників, адже це важливо розуміти для того щоб створити асистента націленого в більшій мірі на саме необхідне завдання. Тому на рисунку 1.2 можна спостерігати діаграму самих розповсюджених запитів від користувачів різних голосових асистентів.

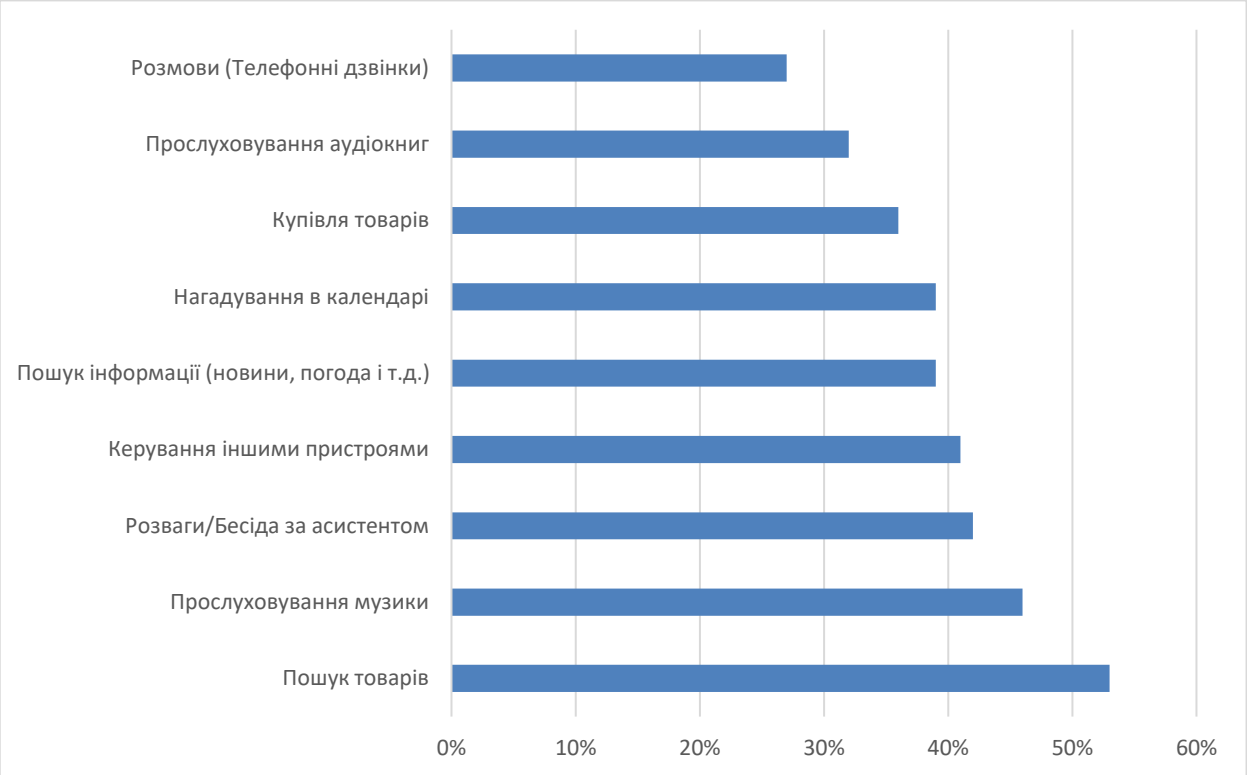


Рисунок 1.2 Причини використання голосових асистентів

## 1.3. Постановка задач дослідження

Виходячи з розглянутого матеріалу в попередніх підрозділах, необхідно розуміти які завдання потрібно визначити. Основним завданням є розробка

системи голосового асистента за допомогою Python. Проаналізувавши існуючі аналоги таких технологій можна виділити такі причини створення нової технології голосового асистенту:

– більшість голосових помічників націлені на роботу лише на смартфонах (Android, iOS), мають обмежений функціонал при роботі на комп'ютері або працює лише на пристроях від компанії виробника технології голосового помічника;

– не може використовуватись на території України по різним причинам (не орієнтована на країну, накладені санкції проти даного продукту, потребує додаткових маніпуляцій, котрі не завжди успішні, для роботи з продуктом або інше);

– проблеми з розумінням голосових запитів та надання невірних даних по запиту;

– мають влаштовану рекламу.

Більшість всіх вищевказаних недоліків можна виправити дуже просто, чим інші компанії й користуються. Тому аналізуючи також й ринок голосових помічників, можна сказати що їх з'являється все більше й більше, але вони не мають такого великого розповсюдження через не популярність бренду або великої кількості помилок при роботі з запитами. Також однією із проблем більшості таких «маленьких компаній» є те, що вони продають своїх голосових асистентів або пропонують підписку на деякий період за певну кількість коштів, але як вже було сказано продукт або недороблений і має багато помилок в роботі, або надто дорогий, як для помічника, котрого можна

Окрім цього, також існує й багато відомих брендів, які вирішили брати приклад з раніше розглянутих мною компаній ті не змінювати нічого. Тому все більше компаній, котрі займаються виготовленням гаджетів, намагається бути максимально клієнт орієнтованими та випускають ексклюзивні моделі пристроїв (смартфони, планшети, розумні пристрої для будинку та ін.), на які вже встановлений власний голосовий помічник.

Також необхідно буде розглянути задачу по використанню кодового слова для активації голосового помічника, адже якщо помічник буде записувати

кожне ваше слово, алгоритм заплутається та буде видавати неправильні відповіді на запити, яких ви навіть не потребували. Помічники повинні записувати звук тільки після того, як почують від користувача кодове слово. Однак вони можуть відреагувати на схожі слова та мову по телевізору або іншим пристроям, або навіть спрацювати без причини. Така проблема спостерігалась у багатьох голосових асистентів, оскільки їх кодові слова були або простими або були додані варіанти спрацьовування на слова котрі лише нагадують їх команди, але не означають їх насправді.

Якщо голосовий помічник буде створений, як окремий додаток на пристрій та зможе працювати без підключення до мережі Internet (окрім сервісів (запитів) які будуть потребувати підключення до мережі) то такий додаток буде працювати не залежачи від серверу виробника, як в більшості компаній. Працівники компаній-розробників можуть отримати доступ до персональної інформації. Оскільки перевіряють якість роботи голосових помічників люди, вони можуть дізнатись особисті данні. За даними кореспондентської сторінки The Guardian, Apple внесли зміни в програму контролю якості голосового помічника Siri. За новими правилами, без згоди користувачів, працівники та розробники не матимуть змоги прослуховувати голосові команди, які відправляють користувачі Siri.

Злочинці можуть використати данні надані користувачами. Як й будь-яка інша компанія, що збирає персональні данні про своїх користувачів та голосові записи знаходяться під загрозою кібератак. Їх можуть використати для імітації голосу користувача та захвату його акаунтів, захищений біометрією. В деяких випадках атаки можуть не потребуватися наприклад, зафіксовано випадок коли користувачу Amazon випадково було відправлено 1700 аудіозаписів іншої людини, з якою він не був знайомий, після чого, як він запросив файл зі своїми власними даними.

Може виникнути конфлікт інтересів. Компанії збирають особисті данні для більш ефективного рішення задач клієнтів, але можуть використовувати їх в свою користь або на користь партнерів. Так за даними кореспондентського порталу Bloomberg, деякі працівники компанії Amazon можуть дізнатись, де

були виконані запити зі зверненням до Alexa, та вияснити домашню адресу користувача.

Для вирішення зазначених проблем, котрі зустрічаються при роботі з голосовими помічниками на сьогодення необхідно перейти до комплексного підходу вирішення цих проблем. Основними етапами вирішення раніше розглянутих проблем, повинні стати наступні положення:

– розробка стандартних механізмів роботи голосового помічника для забезпечення швидкої обробки запитів – стандартизація підходів до роботи голосового помічника дозволить створити більш зручне середовище як користувачу, так й розробнику. Так можна опиратися на шаблонні запити від користувачів та додати їх в програму чим й забезпечити помічнику можливість працювати без підключення до мережі Internet;

– розробка додатку голосового помічника з високим рівнем розпізнавання мовлення людини – забезпечення якісного розпізнання мовлення дозволить виконувати запити без помилок, розуміти декілька команд в одному запиті та здійснювати пошук інформації в інтернеті;

– забезпечити можливість майбутнього навчання асистента для збільшення можливостей – використання бібліотек, які не обмежують систему голосового управління декількома вказаними командами, а дозволить в майбутньому навчатись та запам’ятовувати розмови з користувачем для виявлення його потреб та сподобань;

# РОЗДІЛ 2. РОЗРОБКА СИСТЕМИ ГОЛОСОВОГО УПРАВЛІННЯ ЗА ДОПОМОГОЮ PYTHON

## 2.1. Розробка структури голосового асистента

Голосовий асистент виконує різноманітні задачі користувачів, такі як пошук інформації в інтернеті, пошук місць на карті, інформування щодо прогнозу погоди, може підтримати розмову та інше. Для виконання цих задач більшість асистентів використовують сервери компанії виробника (рис. 2.1), команда почута голосовим асистентом направляється до серверів для аналізу команди її розпізнання та розбиття її на окремі компоненти, після чого сервер повертає команду назад до голосового асистента та перевіряє чи збігаються компоненти з вказаними шаблонами команди та використовує вказівки вказані в програмі для виконання команди.


Рисунок 2.1 Структура системи голосового управління в онлайн режимі

Якщо голосовий асистент створений як незалежний додаток, встановлюються необхідні голосові пакети, які виконують роль «сервера», всі дії по розпізнаванню слів користувача та розбиття його на складові виконуються завдяки цим пакетам та програмі, виконання самої команди виконується так, як

й в попередньому прикладі (рис. 2.2). Однак у голосових пакетів є свій недолік, а саме, при використанні методу онлайн перевірки команди витрачається менше часу ніж при перевірці голосовим пакетом, окрім цього голосові пакети займають додаткове місце в оперативній пам'яті, що може сповільнити роботу ПК користувача. Якщо все ж використовувати голосові пакети, то є декілька варіантів, малий та великий голосові пакети, малий пакет займає менше місця в оперативній пам'яті комп'ютера ніж великий, але має меншу кількість шаблонів.


Рисунок 2.2 Структура системи голосового управління в офлайн режимі

На першому етапі виконується активація, наприклад, при використанні користувачем ключової фрази. Асистент постійно прослуховує оточуючі звуки, аналізує наявність ключової фрази та, якщо фраза була розпізнана, асистент переходить в активний режим.

Далі користувач говорить текст, який може пояснити асистенту, що користувач хоче зробити. Система розпізнавання (Automatic Speech Recognition) перетворює текст в N-кількість кращих гіпотез того, що мав на увазі користувач. Далі система розпізнання мовлення (Natural Language Understanding) перетворює текст в N-кількість кращих варіантів для розуміння фрази користувача, далі фрази діалогу інтерпретуються та класифікуються та визначає, що необхідно

зробити на основі отриманої інформації. Наприклад, звернутись в різноманітні сервіси для отримання інформації.

Після отримання необхідних даних, система виконує ще один процес повернення інформації користувачеві, тобто система генерації мовлення (Natural Language Generation) генерує текст для відповіді користувачу, далі система генерації голосу (Text-To-Speech) на основі отриманих моделей генерує звукову інформацію, яка й оголошується користувачеві в якості реакції-відповіді. Окрім відповіді, може також виконуватись будь-яка дія на комп'ютері, наприклад запуск іншого додатку або пошук інформації в пошуковій системі.

## 2.2. Основи цифрової фільтрації сигналу

Система голосового управління приймає аналоговий сигнал з мікрофону користувача, після чого сигнал перетворюється на цифровий для розбиття на складові та їх порівняння з існуючими шаблонами запитів в програмі, тому необхідно розглянути основи цифрової фільтрації сигналу, для того щод зрозуміти роботу таких систем краще.

В загальному випадку аналого-цифровим перетворенням називається процес перекладу фізичної величини в цифрове значення. Цифровим значенням є набір одиниць та нулів зрозумілих цифровому пристрою. Таке перетворення потрібне для взаємодії цифрового середовища з оточуючим середовищем.

Так як аналоговий електронний сигнал повторює своєю формою вхідний сигнал, але він не може бути записаний в цифровому вигляді, оскільки він має нескінчену кількість значень. Прикладом можна привести процес запису звуку (рис. 2.3).

Він представляє собою суму хвиль з різноманітними частотами, які при розкладанні по частотам, так або інакше виглядають як хвиля синусу. [12]

Рисунок 2.3 Вигляд типового аналогового сигналу

Фактично перетворення сигналу в цифровий це складний процес, який складається з двох основних етапів, а саме дискретизація та квантування по рівню.

Дискретизація сигналу це визначення проміжків часу, на яких виміряється сигнал. Чим коротші ці проміжки – тим точніші виміри. Періодом дискретизації (T) називається відрізок часу від початку зчитування даних до його кінця. Частота дискретизації (f) – це зворотна величина (2.1).

$$f_{\text{д}} = \frac{1}{\text{T}} \tag{2.1}$$

Нехай x(t) – деякий неперервний сигнал. Тут t – неперервний аргумент, під яким мається на увазі час, хоча в загальному випадку аргументом може бути будь-яка величина, а n порядковий номер відліку. Тоді у формулі 2.2 представлений результат рівномірної дискретизації неперервного сигналу x(t).

$$x(n) = x(nT) \tag{2.2}$$

На рисунку 2.4 приведений приклад процесу рівномірної дискретизації.

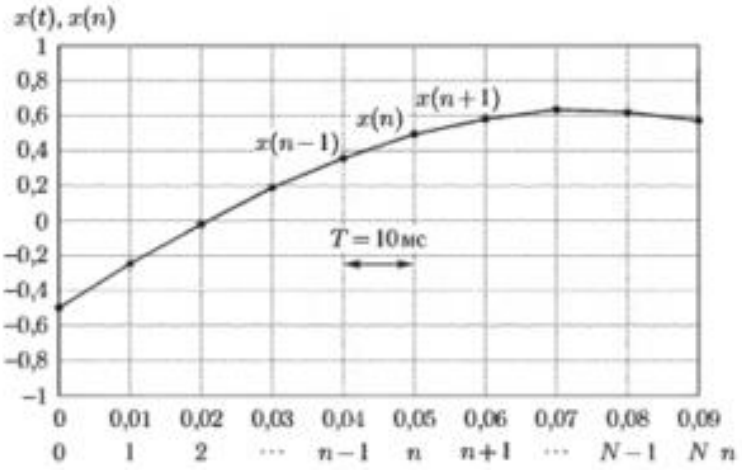


Рисунок 2.4 процес рівномірної дискретизації сигналу

Очевидно, що в дискретному сигналі x(n), отриманому від неперервного сигналу x(t), втрачена інформація про значення, які лежать на проміжках між точками взяття відліку. Якщо функція, яка залежить від часу, має обмежений частотою F спектр, то вона повністю визначається дискретними відліками її значення, послідуюча з частотою вказаною в формулі 2.3 та може бути точно реконструйована за допомогою виразу (2.4).

$$f_n = 2 \cdot F \tag{2.3}$$

$$x(t) = \sum_{n=-\infty}^{\infty} x(n) \frac{Sin[\frac{\pi(t-nT)}{T}]}{\pi(t-nT)/T} \tag{2.4}$$

Твердження щодо можливості повної реконструкції функції є теоретичним, так як засноване на припущені про нескінчену кількість відліків

дискретного сигналу x(n), що на практиці не здійсненим. Тим не менш, теорема відліків є однією із основ всієї теорії дискретної обробки сигналів.

Порушення умов теореми відліків проявляється у появі ефекту накладення частот, який полягає в тому, що при виборі недостатньо високої частоти диксретизації деякі частотні складові стають нерозбірливими.

Природу виникнення ефекту накладення частот можна продемонструвати за допомогою нескладних тригонометричних викладок.

Нехай задано три сигнали вказаних у формулах 2.5, 2.6 та 2.7.

$$x(t) = Cos(2\pi Ft) \tag{2.5}$$

$$x_1(t) = Cos[2\pi(F_д + F)t] \tag{2.6}$$

$$x_2(t) = Cos[2\pi(F_д - F)t] \tag{2.7}$$

Де F та $F_д$ – деякі частоти, при чому відповідають виразу зазначеному у формулі 2.8.

$$F < \frac{1}{2}F_д \tag{2.8}$$

Тоді в результаті процедури рівномірної дискретизації з частотою $F_д$ будуть відповідно отримані наступні три послідовності відліків (2.9, 2.10, 2.11)

$$x(n) = Cos(2\pi FnT) \tag{2.9}$$

$$x_1(n) = Cos[2\pi(F_д + F)nT] \tag{2.10}$$

$$x_2(n) = Cos[2\pi(F_д - F)nT] \quad (2.11)$$

Виконаємо перетворення (2.12, 2.13)

$$x_1(n) = Cos[2\pi(F_д + F)nT] = Cos\left[2\pi\left(\frac{1}{T} + F\right)nT\right] = \\ = Cos(2\pi n + 2\pi FnT) = Cos(2\pi FnT) = x(n) \quad (2.12)$$

$$x_2(n) = Cos[2\pi(F_д - F)nT] = Cos\left[2\pi\left(\frac{1}{T} - F\right)nT\right] = \\ = Cos(2\pi n - 2\pi FnT) = Cos(2\pi FnT) = x(n) \quad (2.13)$$

В приведених перетвореннях використовувались властивості періодичності та парності функції косинус. Ми отримали рівність зазначену у формулі 2.14. Це означає, що заздалегідь різні гармонічні сигнали x(t), $x_1(t)$, $x_2(t)$ після їх дискретизації з частотою $F_д$ неможливо відрізнити один від одного.

$$x_1(n) = x_2(n) = x(n) \quad (2.14)$$

На рисунку 2.5 показаний приклад, який зображує приведені викладки. На всіх трьох графіках цільними лініями показані сигнали, дискретизовані з частотою $F_{д1}$=1000Гц, а пунктирними – з частотою $F_{д2}$=100Гц.

З графіків видно, що порушення умов теореми відліків приводить до того, що по відлікам, взятими з частотою дискретизації $F_{д2}$=100Гц, задані три синусоїди неможливо відрізнити одне від одного.

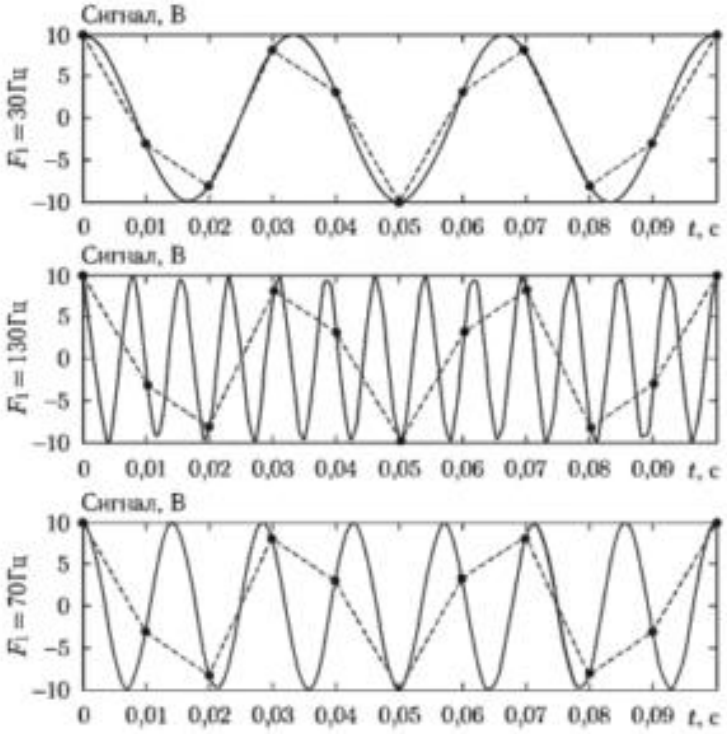


Рисунок 2.5 Ілюстрація ефекту накладення частот

На рисунку 2.6 показаний один і той же фрагмент ЕКГ, який містить в собі короткі імпульси, які мають тривалість близько 2мс. На верхньому графіку відліки взяті з частотою 1000Гц, а на нижньому – з частотою 125Гц, яка явно не достатня для правильного представлення даного сигналу. Із графіків видно, що в даному випадку видно втрату важливої інтерпретації сигналу, що є слідством порушення умов теореми відліків.

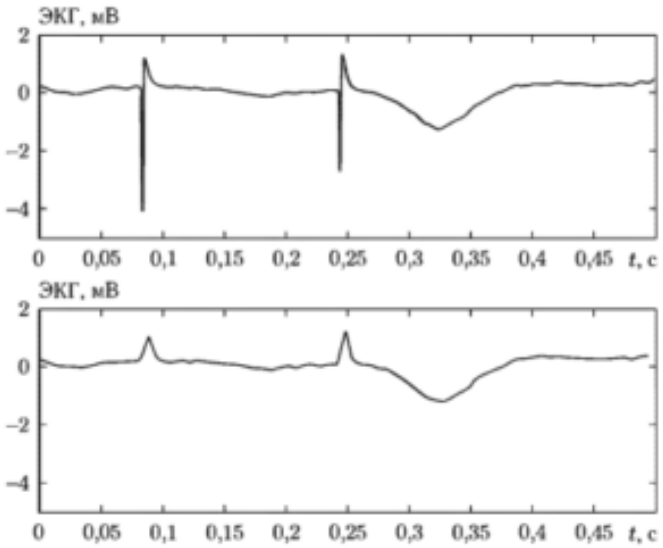


Рисунок 2.6 Приклад втрати високочастотних складових сигналу

Тож частота дискретизації повинна обиратись так, щоб вона перевищувала верхню частоту інформативних складових сигналу, перед аналого-цифровим перетворенням полоса частот початкового аналогового сигналу повинна бути обмежена з використанням аналогового фільтра нижніх частот з частотою зрізу набагато меншої ніж половина обраної частоти дискретизації. [13]

Наступним кроком для перетворення аналогового сигналу в цифровий є квантування, як відомо це присвоєння відліку цифрового значення, відповідного деякому фіксованому рівню сигналу.

Точність квантування сигналу з використанням аналого-цифрового перетворювача визначається двома параметрами, це динамічний діапазон, який задає межі вимірювання вхідного сигналу та розрядністю, яка визначає число двійкових розрядів вихідного сигналу.

Відзначу розмах динамічного діапазону в формулі 2.15.

$$A = A_{max} - A_{min} \tag{2.15}$$

Де $A_{max}$ та $A_{min}$ – найменші та найбільші вимірювані значення неперервного сигналу на вході АЦП відповідно. Якщо далі позначити розрядність АЦП як k, то найбільшим можливим числом рівнів квантування буде $K=2^k$, а величина a=A/K відповідає дистанції між сусідніми рівнями квантування та називається кроком квантування.

Крок квантування визначає точність представлення вхідного аналогового сигналу у вигляді цифрових відліків (рис. 2.7). Максимальна помилка квантування залежить від принципу роботи пристрою та дорівнює ±a/2.

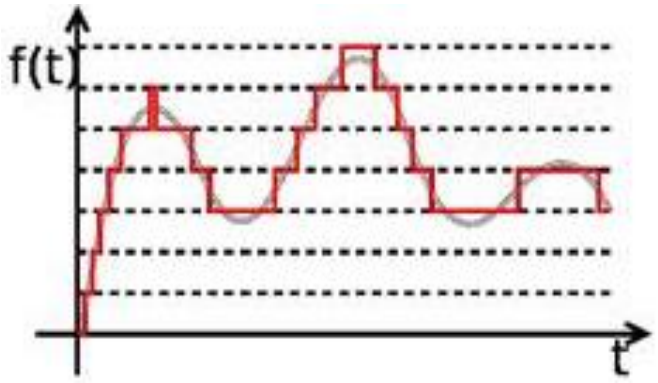


Рисунок 2.7 Вигляд квантованого сигналу

Для оптимального вибору динамічного діапазону та його розрядності необхідна інформація про діапазон можливого вимірювання сигналу на вході та необхідної точності представлення цифрового сигналу. Чим підвищення точності за рахунок збільшення розрядності обмежене точністю представлення самого аналогового сигналу.

На рисунку 2.8 представлений приклад цифрового сигналу після квантування.

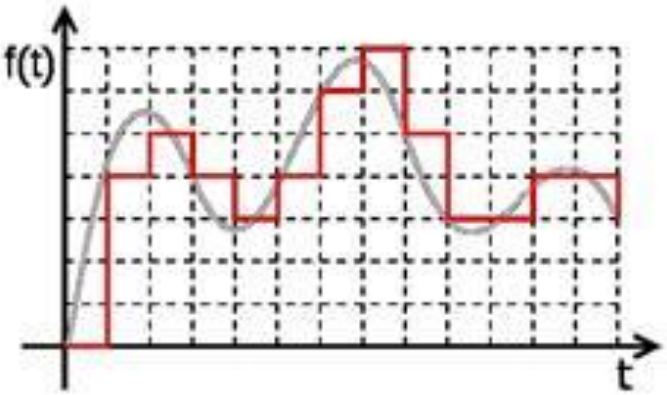


Рисунок 2.8 Вигляд цифрового сигналу після дискретизації та квантування

При перетворенні аналогового сигналу в цифровий рівень квантування називають також глибиною дискретизації або бітністю. Глибина дискретизації вимірюється в бітах та означає кількість бітів, які виражають амплітуду сигналу. Чим більше глибина дискретизації, тим більш точно цифровий сигнал відповідає аналоговому. [14]

## 2.3. Розробка алгоритмічного забезпечення роботи голосового асистента

Оскільки структура вже створена, розглянемо більш детально принцип роботи системи голосового управління який наведений нижче (рис. 2.1):

1. формується запит від користувача з використанням певних команд (ключових слів);
2. запит відправляється на попередню обробку запиту, а саме перетворення отриманого запиту в цифровий сигнал;
3. запис голосового запиту фільтрується від сторонніх шумів;
4. оброблений запит направляється на сервери для розпізнання команди та розбиття її на складові, отриманий результат відправляється від серверу назад до програми;
5. після отримання результату обробки команди від серверу, програма перевіряє наявність ключових слів в команді відсіюючи зайве
6. на цьому етапі програма перевіряє виконання умови розпізнання команди:

6.1 якщо команда не була розпізнана або не було знайдено ключових слів зазначених в програмі:

6.1.1 програма формує повідомлення про нечітку команду та потребує від користувача повторити його запит або дати нову команду та повертається в активний режим (режим прослуховування)

6.2 якщо був збіг ключових слів та команда була розпізнана без помилок:

6.2.1 виконуються дії зазначені в програмі, тобто команда відправляється на виконання;


Рисунок 2.9 Алгоритм роботи системи голосового управління

# РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ ГОЛОСОВОГО УПРАВЛІННЯ

## 3.1 Порівняльний аналіз інструментів для розробки системи голосового управління

Для виконання поставлених задач перед розробкою системи, необхідно зробити порівняльний аналіз вже існуючих рішень та обрати більш зручні. Розглянемо кожну задачу та визначимо, як реалізувати кожну з них, тож на первинному етапі необхідні такі функції:

– прослуховувати мову користувача;
– розпізнавати мову;
– включати в себе метод нечіткого порівняння;
– синтез мови;
– отримувати данні щодо виконуваних процесів та перевіряти стан оперативної пам’яті;
– виконувати пошук інформації за допомогою пошукових систем.

Розглянемо кожну з цих задач та методів їх виконання по порядку.

Python має безліч інструментів та бібліотек, за допомогою яких програма зможе отримувати звук з мікрофона та програвати його з динаміків, але звернемо увагу на дві бібліотеки: PYO та PyAudio. Оскільки ці бібліотеки більш популярні та гарно справляються за своїми функціями.

PYO – це модуль Python для створення сценарію цифрової обробки сигналів. Цей модуль Python містить в собі класи для обробки самих різних типів аудіосигналів. Завдяки цьому користувачі можуть маніпулювати звуковими сигналами в реальному часі за допомогою інтерпретатора.

Інструмент модулів PYO в Python має примітиви, такі як математичні операції, базову обробку сигналів: затримки, фільтри та інше. Але він також об’єднує алгоритми для створення звукової грануляції та багатьох інших художніх звукових операцій.

PyAudio – це бібліотека Python, яка представляє собою кросплатформенний аудіовхід-вивід. Він має широкий спектр функцій, пов’язаних зі звуком та в основному орієнтований на сегментацію, класифікацію та візуалізацію. За допомогою PyAudio, користувачі можуть виявляти звукові події та фільтрувати період тиші, застосовувати зменшення розмірності для візуалізації аудіоданих та подібності контенту.

Бібліотека PyAudio працює таким чином:

1. отримання даних щодо доступних аудіопристроїв – через додаткові налаштування можна отримати індекси всіх аудіопристроїв та обрати той, через який буде виконуватись запис даних, або додати налаштування щодо автоматичного пошуку та підключення аудіопристрою, який прийматиме звук;
2. налаштування необхідних параметрів запису – дана бібліотека дозволяє налаштувати всі тонкощі запису звуку: формат; розмір; кількість каналів; вивід звуку; частоту;
3. запис аудіодоріжки – через цикл програма буде записувати дані запису в масив;
4. збереження запису – після завершення запису бібліотека дозволяє зберегти запис на пристрій, як окремий аудіофайл, або зберегти його лише на відведений час для його обробки, після чого аудіофайл видаляється самостійно, що дозволяє економити місце на пристрої.

PyAudio є дуже поширеною бібліотекою для запису звуку, окрім цього він може виділяти моменти коли користувач говорить, а коли мовчить, що не змушує програму слухати весь час, а лише почути кодове слово для початку роботи, що є перевагою перед PYO, яка працює весь час, тим паче, бібліотека PYO більше підходить для програвання звука ніж для запису та прослуховування програмою для виконання окремих команд, тому зазвичай цю бібліотеку використовують для роботи з музикою ніж з голосом. Вагомою перевагою PyAudio перед PYO є гнучке налаштування збереження файлів, бібліотека PyAudio може зберігати файли, як на завжди так й на декілька секунд, в той час, як PYO не має таких гнучких налаштувань та зберігає аудіофайли назавжди, чим займає місце на пристрої. Також бібліотека PyAudio дозволяє побачити всі підключені звукові

пристрої для правильного налаштування, та окрім цього якщо дана бібліотека працює разом з SpeechRecognition, вона може налаштувати автоматично з якого аудіо пристрою прослуховувати та на який виводити звук.

Для розпізнавання мови Python має декілька бібліотек, таких як apai, wit, google-cloud-speech, але такі бібліотеки, як wit та apai доволі складні у використанні та більше підходять для самонавчання програми, вони доволі добре справляються з розпізнанням мови та виявляють потреби користувача, які виходять за рамки базового розпізнання мови. Google cloud speech може розпізнавати мову також на високому рівні, але вона заточена під перетворення сказаного користувачем в текст, тому на даному етапі вона не підходить для використання її як основного розпізнавача.

У той час як бібліотека SpeechRecognition спрощує отримання даних щодо аудіопристрою, з якого й буде отримувати інформацію, окрім цього вона діє як оболонка для декількох популярних API, таким чином є доволі гнучкою. Один з них API Google Web Speech, він підтримує ключ API за умовчанням, який вже запрограмований в бібліотеці. Це означає що можна розпізнавати мову не підписуючись на послугу, окрім цього вона також містить в собі й функції бібліотеки wit, що також дозволяє використовувати її для самонавчання програми. Тож серед переваг бібліотеки SpeechRecognition основними є відкритий код, доступність більшості влаштованих API, простота у використанні та можливість розпізнавати мову користувача в офлайн режимі.

Окрім розглянутих бібліотек щодо запису мови користувача та його розпізнавання необхідно гарантувати правильне розуміння користувача програмою, але в Python майже немає бібліотек які мають нечітке порівняння для використання її в системі голосового управління. Як приклад ми маємо доволі багато модулів для порівняння текстових даних, вони зазвичай використовуються для перевірки на правильність тексту ніж на виконання команд та запитів.

Regex – це модуль роботи з текстом. Він дозволяє виконувати пошук, заміну або інші операції над текстом. Зазвичай дану бібліотеку використовують

для перевірки користувацького введення , пошук на сторінці або у файлі, аналіз текстової інформації та інше.

Що стосується нечіткого порівняння для системи голосового управління, однозначним вибором є бібліотека fuzzywuzzy оскільки вона є найефективнішим методом для порівняння двох рядків, які відносяться до одного й того ж об’єкту, але написаних трішки по різному. Це процес пошуку рядків, який відповідає заданому шаблону. Він використовує відстань Лівенштейна для розрахунку різниці між послідовностями.

Ця бібліотека може допомогти порівнювати бази даних, в яких відсутній загальний ключ, наприклад об’єднати дві таблиці по найменуванню компаній, та котрі по різному відображаються в обох таблицях, але в нашому випадку, бібліотека також може порівняти сказане користувачем з тим, що вказано в програмі. Наприклад користувач вказав свій запит таким чином – «Знайди в пошуковій системі інформацію …», але в програмі ця команда вказана іншим чином – «Знайди інформацію в пошуковій системі…». В такому випадку програма за допомогою цієї бібліотеки порівняє обидва варіанта та зробить команди користувача зрозумілішими для програми, якщо бібліотеку не використовувати, при заданні «неправильних» команд, програма не зрозуміє користувача та буде говорити про помилку не виконуючи жодної команди.

Відстань Лівенштейна використовується для розрахунку відстані між двома послідовностями слів. Воно розраховує мінімальну кількість корегувань, які нам необхідно замінити в цьому рядку. Ці зміни можуть бути вставкою, видаленням чи заміною.

Відстань Лівенштейна між двома рядками a та b визначається функцією зазначеною у формулі 3.1 та 3.2.

$$d_{a,b}(|a|,|b|) \tag{3.1}$$

$$d_{a,b}(i,j) = \begin{cases} \max(i,j) & if\ \min(i,j) = 0 \\ min \begin{cases} d_{a,b}(i-1,j)+1 \\ d_{a,b}(i,j-1)+1 \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \\ d_{a,b}(i-2,j-2)+1 \end{cases} & if\ i,j > 1\ and\ a_i = b_{j-1} \\ & and\ a_{i-1} = b_j \\ min \begin{cases} d_{a,b}(i-1,j)+1 \\ d_{a,b}(i,j-1)+1 \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & otherwise \end{cases} \quad (3.2)$$

Де $1_{(a_i \neq b_j)}$ це індикаторна функція, яка рівна нулю при $a_i$=$b_j$ та 1 в іншому випадку.

Кожен рекурсивний виклик відповідає одному з випадків відповідно до наведених нижче формул:

– формула 3.3 відповідає за видалення символу (з a в b);

– формула 3.4 відповідає за вставку (з a в b);

– формула 3.5 перевіряє відповідність або невідповідність, в залежності від співпадіння символів;

– формула 3.6 використовується у випадку перестановки двох послідовних символів.

$$d_{a,b}(i-1,j)+1 \quad (3.3)$$

$$d_{a,b}(i,j-1)+1 \quad (3.4)$$

$$d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \quad (3.5)$$

$$d_{a,b}(i-2,j-2)+1 \quad (3.6)$$

Після всіх переглянутих функцій варто звернути увагу й на синтез мови для створення відповідей для користувача від системи. Для Python існує доволі невелика кількість бібліотек для синтезу мови, розглянемо найвідоміші з них.

Бібліотека Google Text to Speech API (далі gTTS) є доволі простою та зручною у використанні, вона має підтримку великої кількості мов, такі як українська, російська, англійська, німецька та інші. Зазвичай вона зберігає та програє файл у форматі .mp3, однак в останніх версіях цієї бібліотеки голосовий

файл неможливо редагувати, він генерується системою та не підлягає змінам. Так, бібліотека має вагому перевагу в якості та кількості перекладів на різні мови, але займає місце в пам’яті, що є доволі великим недоліком.

Інша бібліотека це pyttsx3, вона є автономною та може використовувати будь-які голосові пакети для створення мовлення, вона не використовує збережені аудіофайли для створення відповіді, а генерує його одразу виходячи з команди користувача та вказаної відповіді в програмі, це економить місце що й є перевагою перед gTTS. Окрім цього бібліотека дозволяє переводити текст на мови, які підтримуються голосовим пакетом, тому зазвичай її використовують разом із голосовими пакетами від операційної системи, для кожної системи він різний. Також при зміні відповіді або при виборі з декількох відповідей ця бібліотека виграє у gTTS та може використовуватись в системах з машинним навчанням.

Ще однією задачею системи голосового управління є перевірка процесів та стану оперативної пам’яті, оскільки деякі запити потребують відкриття або виконання сторонній програм по запиту користувача, тому щоб не навантажувати пристрій користувача, необхідного перевіряти стан оперативної пам’яті та попереджувати користувача про неможливість виконання цього запиту або дозвіл виконувати запит на розсуд користувача. Окрім цього однією із задач є закриття програми в той час коли вона користувачу вже не потрібна. Таким чином при поданні команди користувачем про завершення роботи системи голосового управління, програма виконуючи цей запит видалить власний процес з виконання. Для таких цілей є декілька бібліотек, наприклад PSI, pystatgrab та psutil. Але в даному випадку бібліотеки PSI та pystatgrab не варто використовувати, раніше вони були доволі популярні, але вони мають й свої недоліки, ці бібліотеки більше не мають оновлень для більш нової версії Python та мають свої обмеження по платформам, наприклад pystatgrab не підтримується на платформі Windows.

Бібліотека psutil досі оновлюється та підтримує всі відомі платформи, окрім цього проста у використанні. Окрім цього бібліотека дозволяє отримувати дані та доступ до файлів та папок, тобто можна за допомогою команди повністю

керувати своїм ПК: вимкнення системи, блокування системи, відкриття файлів та папок, завершення процесів, отримати дані щодо часу/дати/дня тижня та багато іншого. Тоді коли вище вказані бібліотеки мають доступ до меншої кількості функцій. Зазвичай бібліотека psutil встановлюється разом з pywin32, яка в свою чергу є не тільки допоміжною бібліотекою, але й надає доступ до ширшого спектру функцій Windows для розробника, в моєму випадку це надання доступу до голосових пакетів для синтезу мовлення, що дуже зручно, адже немає необхідності завантажувати та встановлювати сторонні додаткові голосові пакети.

Останньою на даному етапі основна задача це пошук інформації в інтернет просторі, на жаль Python ще не отримав необхідну кількість бібліотек тому серед них не складно обрати одну яка буде влаштовувати результатами своєї роботи та кількістю функцій. Зараз популярними є BeautifulSoup та google. BeautifulSoup добре справляється з пошуком інформації, може зберігати в окремий файл текст сторінки та відокремити необхідні сторінки від непотрібних. Дана бібліотека має дуже тонкі налаштування та потребує встановлення додаткових бібліотек, таких як lxml та requests. BeautifulSoup дозволяє отримувати інформацію з окремих сторінок, які необхідні користувачеві, але в системі голосового управління, у якої користувачі можуть цікавитись різними речами, вона не підходить, саме через її тонкощі в налаштуванні, не дивлячись на важкість цих налаштувань лише для однієї сторінки. Якщо необхідно знайти інформацію на сторінці, яка не була вказана в програмі то запит не виконається, оскільки програма не буде розуміти, що від неї потребують.

Бібліотека google виграє у цьому випадку, адже не дивлячись на запит необхідної користувачу інформації, вона знайде все що він потребує, навіть якщо це не було вказано в програмі. Якщо в попередніх версіях бібліотека надавала інформацію лише від конкретної кількості сторінок вказаної розробником, то наразі можна використати модуль web для відкриття браузера та автоматичного знаходження великої кількості різних результатів. Окрім цього використовуючи допоміжний модуль web ми також можемо зайти на інші сторінки, які зацікавлять користувача, або ввімкнути відео. Так, з цим модулем буде складно

змусити систему прочитати новини або отриману інформацію, але основну свою функцію пошуку модуль виконує, цього й достатньо на цьому етапі роботи системи голосового управління.

## 3.2 Програмна реалізація системи голосового управління

Для того, щоб програмно реалізувати систему голосового управління необхідно поставити задачі, яким вона повинна відповідати, в таблиці 3.1 були описані основні задачі системи та необхідні для їх виконання ресурси.

Таблиця 3.1

| Опис задач | Необхідні залежності |
|---|---|
| Розпізнавати та синтезувати мову | pip install PyAudio (використання мікрофону)<br>pip install puttsx3 (синтез мови)<br>для розпізнання мови:<br>pip install SpeechRecognition (висока якість online-розпізнавання, підтримка великої кількості мов) |
| Виконувати прості завдання (вимкнути комп’ютер, сказати час/дату/день тижня і т.д.) | pip install pywin32<br>pip install psutil (бібліотека для отримання даних про оперативну пам'ять, основні процеси комп'ютера та ін.) |
| Ввімкнути відео | - |
| Знаходити інформацію в пошуковій системі Google | pip install google (бібліотека для пошуку інформації через пошукову систему та відкриття результатів пошуку) |
| Вітатися та прощатися (після прощання робота програми завершується) | - |

Окрім вказаних бібліотек та можливостей в таблиці, також були додані бібліотеки для нечіткого порівняння (fuzzywuzzy.fuzz) та додаткова бібліотека

для прискорення цього порівняння (python-Levenshtein). Дані білбіотека необхідні для покращення рівня розуміння системою людської мови. Таким чином, система має команду «Знайди інформацію в пошуковій системі…», але користувач вказав системі запит з іншим порядком слів, наприклад «Знайди в пошуковій системі інформацію…», тоді в даному випадку, система проаналізує свою команду з командою вказаною користувачем, та відповість правильно незалежно від порядку слів.

Вказавши всі необхідні мені бібліотеки, я отримала результат зображений на рисунку 3.1

```
1  import time, ctypes, webbrowser as web
2  from fuzzywuzzy.fuzz import ratio as rat
3  from random import choice
4  from colorama import *
5  from os import system, getpid
6  from datetime import datetime
7  from psutil import virtual_memory as ozu
```

Рисунок 3.1 Підключення основних та додаткових бібліотек

Розберемо детальніше деякі з них:

– import time, ctypes, webbrowser as web – використовуємо бібліотеку зовнішніх функцій та модуль який дозволяє виконувати пошук інформації в браузері, використовуючи Python (виконання пошуку інформації використовуючи систему голосового управління);

– from random import choice – дозволяє обрати випадкову відповідь користувачу на його запит, зазвичай це привітання, розмови, прощання або інформація про помилку;

– from os import system, getpid – бібліотека надає доступ до основних можливостей операційної системи, в даному випадку system – виконує деякі системні команди, а getpid – надає інформацію щодо процесів системи;

– from datetime import datetime – імпортуємо дані з системи, котрі надають доступ до інформації щодо системного часу та дати;

– from psutil import virtual_memory as ozu – дозволяє програмі отримати дані про стан комп'ютера, а саме про його зайняту оперативну пам'ять, це зроблено для того щоб не навантажувати систему через додаткові запити;

Основними діями програми є розпізнавання та синтез мови. Для розпізнавання мови програмі необхідно чути коли до неї звертаються, та що саме їй говорять (рис 3.2). Я реалізувала в системі функцію прослуховування, тобто коли користувач активує програму асистент переходить автоматично в активний режим, подаючи команду «Я вас слухаю», після чого очікує відповіді, як тільки вона отримує будь-яку відповідь, система одразу фільтрує звук від сторонніх шумів та відправляє на ідентифікацію до серверів, в цьому випадку я використала онлайн розпізнавач Google оскільки він підтримує велику кількість мов та дуже швидко повертає відповідь системі. Після отримання ідентифікованого запиту, система перевіряє наявність ключових слів вказаних в програмі (рис. 3.3), після чого повертає відповідь користувачу у вигляді виконаного запиту, або повідомленням про помилку, якщо запит не було розпізнано або при порушенні процесів роботи програми.

Якщо після ввімкнення програми, користувач не давав команди або команда була виконана, але користувач хоче звернутись до системи ще раз, програма залишається активною в очікуванні нового запиту, та при використанні ключового слова для виклику помічника, він відповість користувачеві в очікуванні нового запиту. Більш детально з цією функцією можна ознайомитись в Додатку А.

```
def listen(self):
    system('cls')
    text = ''
    print(choice((Fore.GREEN, Fore.WHITE, Fore.YELLOW)) + 'Я вас слухаю: ')
    with self.m as self.sourse:
        self.r.adjust_for_ambient_noise(self.sourse)
    while text == '':
        with self.m as self.sourse:
            audio = self.r.listen(self.sourse)
            try:
                text = (self.r.recognize_google(audio, language="ru-RU")).lower()
            except:
                pass
    if text in ('эй', 'юми', 'ты здесь', 'ты тут', 'слушай', 'слышишь', 'юми ты здесь', 'юми ты тут'):
        self.first_ans()
        return ''
    else:
        return( text )
```

Рисунок 3.2 Програмна реалізація розпізнавання мови

Створення словника команд є більш зручним рішенням, оскільки так зручніше додавати, редагувати або видаляти команди, так до такого словника буде складно додавати команди на інших мовах, однак він є компактним. Більш детально можна ознайомитись в Додатку Б.

```
def __init__(self):
    self.cmds = {
        ('не мешай', 'уйди', 'я ушел', 'я ушла', 'пока', 'уходи', 'прощай', 'до встречи', 'до скорого', 'увидимся') : self.bye,
            ('привет', 'здравствуй', 'я пришел', 'я пришла', 'доброе утро', 'добрый день', 'добрый вечер', 'приветствую') : self.hello,
            ('поздоровайся', 'поприветствуй всех', 'поздоровайся со всеми', 'поприветствуй') : self.greet,
            ('как дела', 'как ты', None) : self.how_are_you,
            ('что делаешь', 'чем занимаешься', 'чем занята') : self.what_doing,
            ('спасибо', 'благодарю') : self.thanks,
            ('молодец', 'умница', 'какая ты молодец', 'какая умница') : self.well_done,
        ('завершай работу', 'давай спать', 'выключи систему', 'выключи всё') : self.shut_down,
        ('заблокируй компьютер', 'блокировка', 'заблокируй', 'заблокируй экран') : self.lock,
            ('открой youtube', 'зайди на youtube', 'включи youtube', 'зайди в youtube', 'youtube') : self.open_youtube,
            ('день недели', 'какой день недели', 'какой сегодня день недели', 'какой сегодня день') : self.day_of_week,
            ('сколько время', 'время', 'сколько времени', 'какое сейчас время', 'какой сейчас час') : self.time_now,
            ('число', 'какое сегодня число', 'какое число', 'какая сегодня дата') : self.date_is_today,
        ('повтори', 'ещё раз', 'повтори ещё раз', 'ну ка ещё раз', 'ну ка повтори', 'повтори пожалуйста', 'что ты говорила') : self.repeat,
        ('юми', 'уми') : self.first_ans
    }
    self.keywords = [
        'включи',       'уйди',         'включай',      'выключи',      'выключай',     'открой',
        'отключи',      'отключай',     'открывай',     'запусти',      'запускай',     'запустить',
        'умница',       'помоги',       'поприветствуй','открыть',      'расскажи',     'скажи',
        'объясни',      'рассказывай',  'поздоровайся', 'здравствуй',   'привет',       'добавь',
        'удали',        'сложи',        'посчитай',     'вычисли',      'умножь',       'подели',
        'раздели',      'прибавь',      'вычти',        'молодец']
```

Рисунок 3.3 Набір ключових слів та запитів в системі

Якщо в програмі виникла помилка під час роботи, помічник обов’язково сповістить користувача про неї та створить документ з описом помилки (рис. 3.4)

```
def run(self):
    try:
        self.cmd_exe(self.clear_cmd(self.listen()))
    except Exception as e:
        self.error_log(e)
        self.add_phrases(f"Произошла ошибка! {choice(['файл ошибок обновлён', 'проверьте файл ошибок', ''])}")
        self.talk()
```

Рисунок 3.4 Реалізація повідомлення про помилку

Також мною була реалізована друга важлива функція системи, а саме синтез мовлення (рис.3.5). Після отримання запиту, система обирає відповідь на запит та виконує його, відповідь слугує підтвердженням почутої команди та її виконання. В деяких випадках помічник не попереджує про виконання команди, а тихо її виконує, зазвичай це стосується команд які одразу показують результат

та не вимагають попередження про її виконання. Більш детальна інформація в Додатку В.

```
class VirtualAssistent:

    def __init__(self):
        pass

    def talk(self):
        for text in self.phrases:
            self.engine.Speak(text)
        self.phrases = []

    def add_phrases(self, text):
        self.last_phrase = text
        self.phrases.append(text)

    def first_ans(self):
        vals = ('я вас слушаю', 'да-да', 'я здесь')
        self.add_phrases(choice(vals))
```

Рисунок 3.5 Програмна реалізація синтезу мовлення

Після створення основних функцій, я перейшла до етапу виконання команд (рис. 3.6). Я також звернула увагу на можливі проблеми з перевантаженням комп'ютера, для чого й була встановлена раніше оговорена бібліотека psutil.

```
if mode == 'web':
    for var in ['найди пожалуйста', 'найди кто такой', 'найди кто такая', 'найди кто такие',
                'найди что такое', 'найди как сделать', 'найди как', 'найди', 'поищи', 'почему']:
        task = task.replace(var, '').replace('  ', ' ').strip()

    if task:
        self.engine.Speak(choice(['Уже ищу', 'Сейчас найду', 'Сейчас поищем']))
        if float(ozu().percent) <= 95:
            web.open(f'http://google.com/search?btnG=1&q={task}')
        else:
            self.engine.Speak(choice([
                'Компьютер сильно перегружен, не стоит открывать новые вкладки в браузере.',
                'Компьютер сильно нагружен, давайте я лучше почитаю?']))

            answer = ''
            while not(answer):#ждёт ответа
                answer = self.Listen()

            flag = 0
            for var in ('нет', 'не нужно', 'не надо'):
                if rat(var, answer) > 70:
                    self.engine.Speak('Хорошо')
                    flag = 1
                    break
```

Рисунок 3.6 Програмна реалізація виконання команд

У випадку, якщо користувач захоче знайти інформацію через пошукову систему, або знайти відео в YouTube, програма спочатку скористається встановленою бібліотекою, перевірить чи можливо виконати цей запит без перенавантаження оперативної пам’яті комп’ютера та лише потім виконає запит. Якщо оперативної пам’яті не вистачає на виконання цього запиту, програма повідомить про це користувача, та запитає підтвердження виконання цього запиту, після чого перейде в режим очікування відповіді, де користувач може відмовитись від виконання запиту (запит перестане виконуватись) або погодиться виконати запит (запит продовжить виконуватись).

## 3.3 Підтвердження працездатності розробленої системи голосового управління

Реалізувавши систему голосового управління необхідно протестувати її на працездатність, тож перевіримо основні функції:

– виконання простих запитів – привітання, підтримання розмови та інше;
– виконання змішаних команд;
– пошук інформації в пошуковій системі;
– пошук відео за допомогою сервісу YouTube;
– працездатність видалення повторюваних команд.

При завантаженні програми асистент в першу чергу попередить про свою присутність, на екран буде виведено повідомлення від асистента (рис. 3.7).

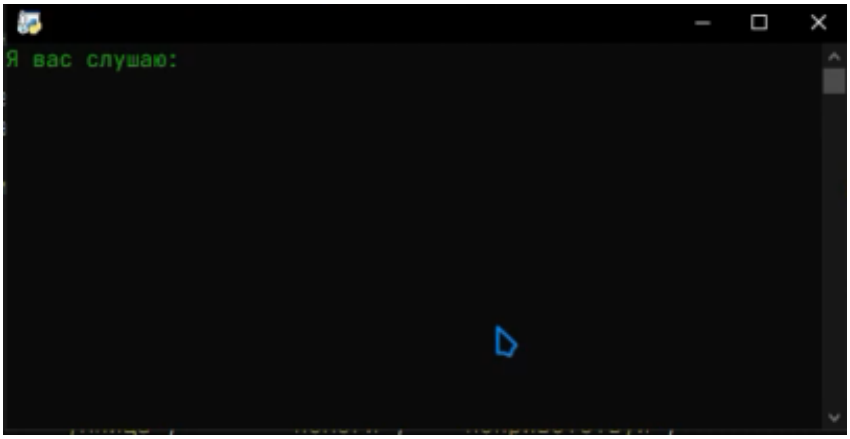


Рисунок 3.7 Запуск програми

Продовжимо виконанням простих команд, які потребують відповіді від голосового асистента (рис.3.8). В програмі вказані варіанти відповіді помічника на команду, ці відповіді можуть обиратись випадково, обираючись з декількох вказаних шаблонів.

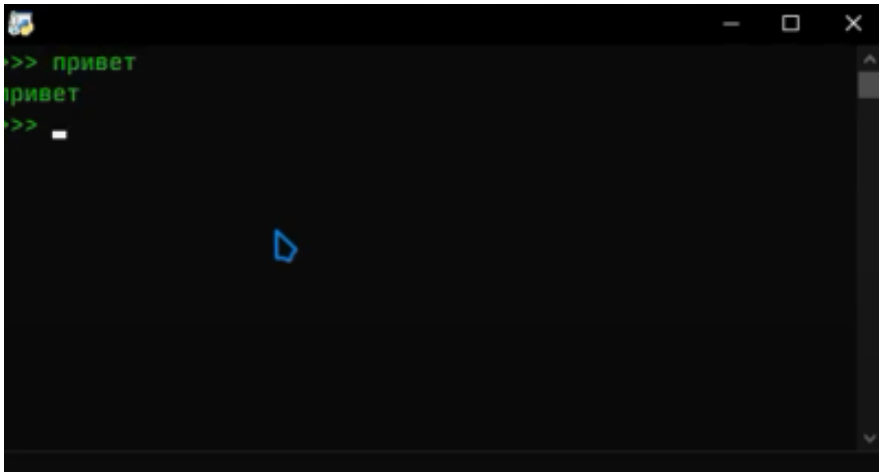


Рисунок 3.8 Результат виконання простого запиту

Простий запит виконується без перешкод, тому перейдемо до виконання змішаного запиту (рис. 3.9), в цьому випадку достатньо додати окреме слово, наприклад ключову команду яка викликає голосового асистента та перевіримо чи відповість вона на обидва ключових запита чи лише на один із них.

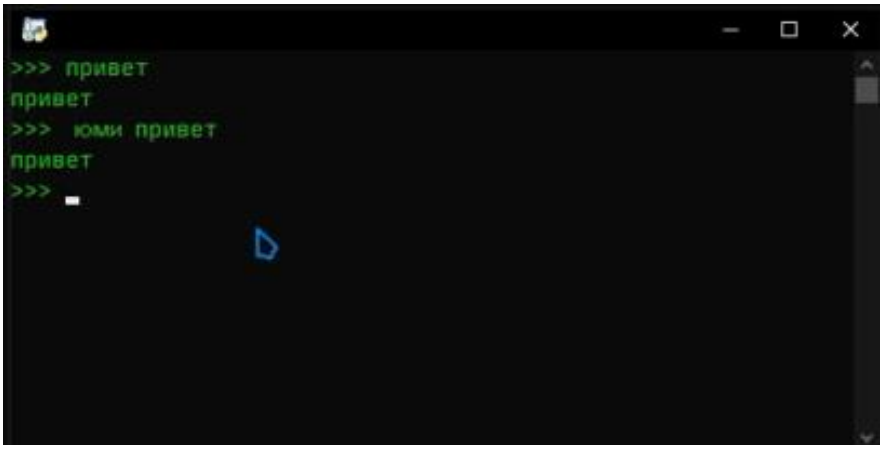


Рисунок 3.9 Результат виконання змішаної команди з використанням ключового слова

Після використання ключового слова виклику системи голосового управління та команди привітання, асистент відповів привітанням ігноруючи ключове слово. Програма в першу чергу перевірила наявність команд, та виконала команду привітання, оскільки команда виклику асистента виконана при ввімкнені програми. Якби ключове слово було використано самостійно без додаткових команд, асистент би попередив, що він на місці та слухає наш наступний запит.

Для подальших підтверджень працездатності я вирішила ввімкнути виведення знайдених програмою команд. Програма буде виводити в якості повідомлення складні команди, такі як пошук інформації або пошук відео в інтернеті, так буде візуально зображено чи розуміє та чи розпізнає команди програма, якщо так, то буде виведено на екран повідомлення з використаною командою та тонкощі виконання команди вказані користувачем.

Тож спробуємо більш складний запит, а саме відкрити в браузері сервіс YouTube (рис. 3.10), якщо цей запит виконується, то в майбутньому можна додавати будь-які сайти та відкривати їх не відволікаючись від роботи.

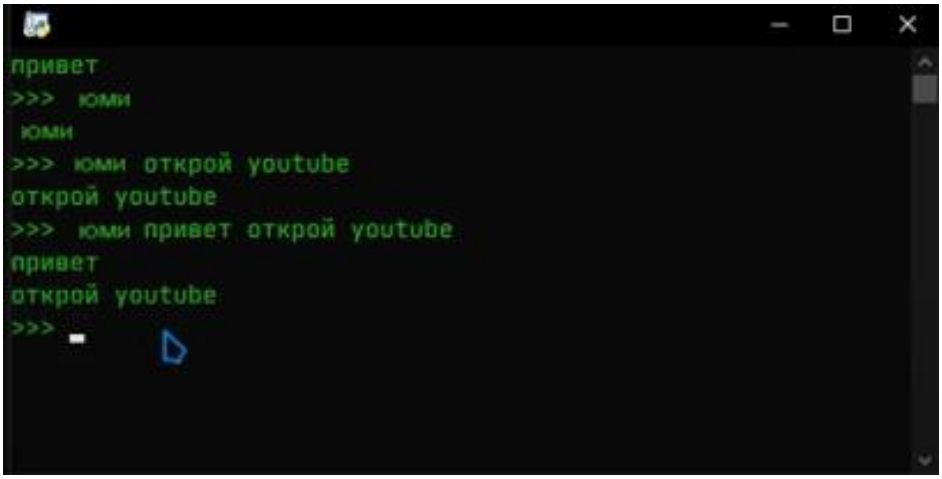


Рисунок 3.10 Виконання запиту з відкриттям сервісу YouTube

Як можна побачити на знімку, програма розпізнала запит та показала повідомлення з розпізнаною командою. Перевіримо випадок, якщо запит буде складатись не з двох команд, а більш складних, наприклад використаємо ключове слово виклику та три команди (рис. 3.11), з попереднього прикладу вже відомо що ключове слово ігнорується при використанні інших команд, та сприймається як звернення, тож необхідно це перевірити в більш складному випадку.

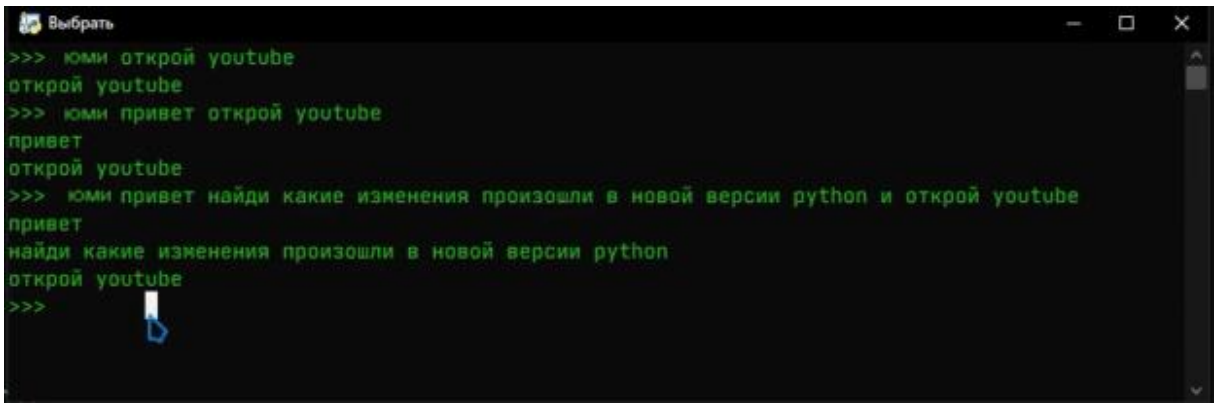


Рисунок 3.11 Використання декількох складних команд

Використавши ключове слово звернення та три команди, а саме привітання, команду «знайди» та команду відкриття сервісу, як і в попередньому прикладі, система розпізнала команди задані їй, а в запиті щодо пошуку інформації чітко почула запит та подала його на виконання.

Далі необхідно перевірити функцію видалення повторень, використовуючи функцію нечіткого порівняння не лише в розумінні команд, але й неможливість їх повторювати декілька разів підряд, можна уникнути проблем з роботою асистента. Тому необхідно перевірити спочатку повторення протої команди (рис. 3.12).

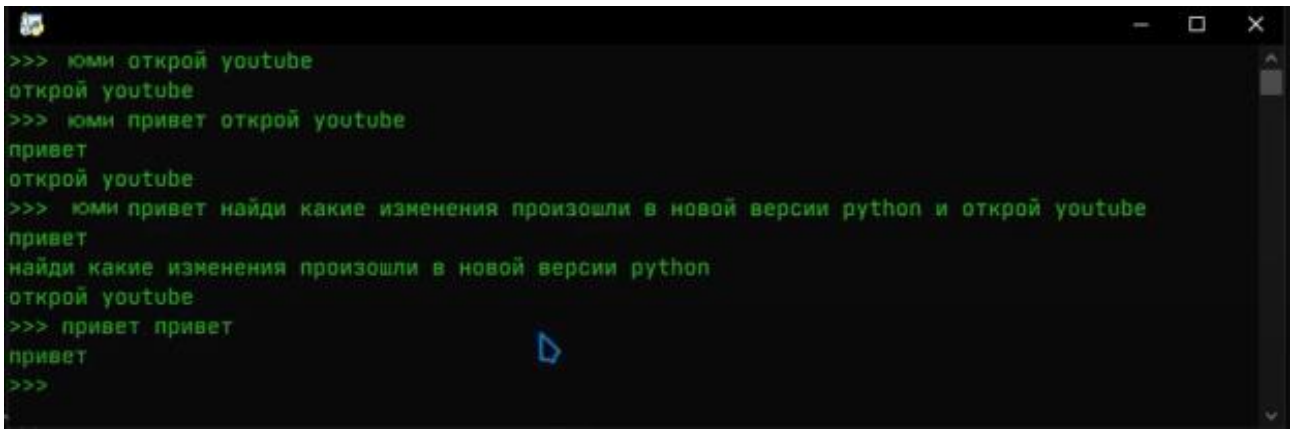


Рисунок 3.12 Перевірка видалення повторюваних команд

Якщо команда повторювалась одразу, видалення спрацьовує та виконується лише одна дія з декількох однакових. Тобто, якщо виникла помилка та команда була вказана декілька разів підряд (незалежно, яка кількість), програма виконає лише одну.

Окрім розглянутих прикладів команд, які використовувались для перевірки працездатності системи, програма також містить в собі декілька простих команд на розмову (як справи, чим займаєшся, привітання з іншими користувачами, відповідь на похвалу/вдячність та інше), програма може завершити роботу комп’ютера або заблокувати систему (режим сну) за командою користувача, відповісти на прості запитання щодо дати, часу, дня тижня та повторити останню сказану нею відповідь на ваш запит, не виконуючи його повторно (якщо запит потребував попереднього виконання).

Перевіривши прості та більш складні команди на їх виконання, програма показала себе з гарної сторони, кожен запит був розпізнаний та відправлений на виконання, навіть при використанні декількох команд, програма зрозуміла та виконала кожний по черзі вказаний користувачем.

При повторюванні команд, програма видалила повторювані команди та виконала лише одну, однак якщо команда переривалась іншою інформацією, наприклад запит пошуку інформації в інтернеті, але мета пошуку різна, обидві команди будуть виконані, але якщо мета була однакова, виконається лише одна команда, це збереже пристрій від навантаження та не змусить користувача закривати додаткові повторювані сторінки.

# РОЗДІЛ 4. РОЗРАХУНОК ЕКОНОМІЧНОЇ ЄФЕКТИВНОСТІ

Економічна ефективність – це позитивна величина, яка виражається у відношенні отриманих результатів та використаних ресурсів для їх досягнення. Основними критеріями являються ступінь задоволеності клієнта, а також ефективність виробництва, соціальної сфери та державного управління.

Кожне підприємство бажає максимізувати дохід, тому використовує різні методи для вдосконалення існуючих продуктів або створення нових. На сьогоднішній час компанії борються за увагу кожного користувача покращуючи свої продукти. З виникненням такої технології, як голосовий асистент, який почав свій розвиток ще до піднятого шуму навколо асистентів від провідних компаній світу, кожен хотів бути першим, однак це було великим ризиком, необхідно було досконало проаналізувати методи його створення та максимально орієнтуватись на вподобання будь-якого користувача. Голосовий асистент повинен був запам’ятовувати всі вподобання користувача та приносити прибуток з рекламних акцій.

Першим хто ризикнув та впровадив рекламу в технологію, яка повинна допомагати людям, був Amazon, за допомогою асистента та вбудованого алгоритму запам’ятовування вподобань користувача, або необережно сказаних слів, Amazon підняли свій дохід за лічені хвилини, адже асистент пропонував саме те, що потрібно користувачеві, й у 80% випадків, користувач купував перший запропонований йому товар, далі Amazon рекламували партнерів, від яких також отримували дохід.

Тому створити систему не складно, складніше її монетизувати та змусити приносити прибуток. Здається, що це не просто, адже на сьогоднішній день існує безліч асистентів, але лише одиниці приносять дохід. Розробники одразу вирішили питання щодо збільшення популярності своєї системи, а саме створення портативної системи розумного будинку, зазвичай вони виглядають, як звичайні колонки, котрі підключені до всіх функцій розумного будинку, та мають змогу керувати не лише ним, а й виконувати необхідні функції на інших підключених пристроях, таких як телефони та комп’ютери. Лише в 2020 році

світовий ринок розумних колонок перейшов відмітку в 150 мільйонів проданих пристроїв, а за прогнозами в 2023 році продажі збільшяться до 640 мільйонів колонок. Тому поки провідні компанії влаштували гонку за допомогою портативних колонок, вони втратили більшу аудиторію, а саме користувачів мобільними пристроями та ПК. На сьогоднішній день близько 40% всього населення світу, використовують голосових помічників для покупок в інтернеті.

В даному випадку доцільно буде створити додаток, який зможе отримувати дохід через рекламу партнерів, така система зможе підняти дохід компаній лише пропонуючи товари необхідні користувачеві, а також система може рекомендувати місця для відпочинку (кафе, ресторани, торговельні центри та інше) або навіть додатки (ігри, корисні додатки для побуту, спорту та інше), звичайно все при використанні запиту.

Тож скільки буде коштувати лише розробка цієї системи, та чи зможе вона приносити дохід вже в першим рік запуску проекту.

Розраховуючи економічну ефективність проекту, необхідно враховувати:

– заробітна плата працівників (розробника);

– витрати на оплату послуг (інтернет, мобільний зв’язок, електроенергія);

– витрати на матеріальні засоби;

– витрати на амортизацію.

Середня заробітна плата розробника програмного забезпечення на базі Python на рівні початківця складає 10 000 гривень на місяць. Для розрахунку повної суми заробітної плати для розробника, використаємо формулу 4.1.

$$З_с = З_м \cdot Т_р \quad (4.1)$$

Де $З_с$ – сумарна заробітна плата за весь період розробки системи, $З_м$ – заробітна плата за місяць роботи, $Т_р$ – період розробки системи.

Для розробки системи голосового управління з наявністю базових функцій необхідно 3 місяці. Використовуючи відомі дані дізнаємося сумарну заробітну плату працівника за цей період (4.2).

$$З_с = 10\,000 \cdot 3 = 30\,000 \text{ (грн)} \quad (4.2)$$

Виходячи з результатів, один розробник початківець, на період розробки проекту, буде коштувати 30 000 гривень.

До собівартості продукту, також необхідно врахувати витрати на оплату послуг, які наведені в таблиці 4.1.

Таблиця 4.1 Вартість використаних послуг

| Найменування послуги | Кількість | Вартість (грн.) |
|---|---|---|
| Інтернет | 80мбіт/сек | 160 |
| Електропостачання | 1кВт/год | 1,44 |
| Мобільний зв’язок | - | 100 |
| Разом | | 261,44 |

Тож судячи по даним наведеним в таблиці, щодо послуг необхідних для розробки проекту, можна розрахувати загальну вартість кожної з послуг. Витрати на мобільний зв’язок за 3 місяці будуть складати 300 гривень, а загальна вартість послуги надання інтернет з’єднання буде складати 480 гривень.

Для того щоб дізнатись загальну вартість електропостачання, котра була використана в період розробки проекту, необхідно зазначити, що проект розроблявся на ноутбуці Lenovo ideapad 320, та споживання електроенергії ноутбуком, складало приблизно 0,45 кВт/год. Враховуючи, що проект розроблявся протягом трьох місяців, тому в робочі дні написання проекту займало близько 4 годин, тобто загальний час розробки проекту зайняв близько 264 години. Для початку необхідно розрахувати вартість години роботи за ноутбуком (4.3).

$$Е_{вг} = Е_{сн} \cdot В_г = 0{,}45 \cdot 1{,}44 = 0{,}648 \text{ грн./год} \quad (4.3)$$

Де $Е_{вг}$ – вартість електроенергії спожитої за годину роботи, $Е_{сн}$ кількість електроенергії спожитої ноутбуком, $В_г$ – вартість електроенергії в розмірі

1кВт/год. З отриманими даними можна дізнатись загальну вартість спожитої електроенергії за час розробки проекту (4.4).

$$В_{ез} = Т_{рн} \cdot Е_{вг} = 264 \cdot 0{,}648 = 171{,}07 \text{ грн.} \quad (4.4)$$

Де $В_{ез}$ – загальна вартість електропостачання, $Т_{рн}$ – період роботи ноутбука.

Витратами на матеріальні засоби вважаються купівля та використання матеріалів, котрі були використані під час розробки проекту, це можуть бути ручки, олівці, папір, флеш накопичувач, всі необхідні дані наведені в таблиці 4.2.

Таблиця 4.2 Вартість витратних матеріалів

| Найменування | Кількість | Вартість (грн.) |
|---|---|---|
| Ручка | 2 | 25 |
| Папір | 7 | 11 |
| Флеш накопичувач | 1 | 150 |
| Разом | | 186 |

Про розробці також варто враховувати витрати на амортизацію, тобто витрати на придбання обладнання, його експлуатацію та ремонт при необхідності. У собівартість проект включається амортизація лише за під час його розробки. Тому за наведеною нижче формулою 4.5 можна дізнатись суму.

$$А = \frac{В_у \cdot Н_А \cdot Т_{ву}}{365 * 100} \quad (4.5)$$

Де $В_у$ – це початкова вартість устаткування, $Н_А$ – норма амортизації у відсотках (прийнято 30%), $Т_{ву}$ – термін використання устаткування (66 днів).

Для розробки був використаний лише ноутбук, вартість якого склала 7 500 грн. В такому випадку всі необхідні дані для розрахунку ми маємо (4.6).

$$A = \frac{7500 \cdot 30 \cdot 66}{365 \cdot 100} = 406{,}85 \text{ (грн)} \quad (4.6)$$

Виходячи з цього можна розрахувати загальну вартість обслуговування обладнання (4.7).

$$B_{оо} = B_{ез} + A = 171{,}07 + 406{,}85 = 577{,}92(\text{грн}) \quad (4.7)$$

Результати всіх проведених розрахунків наведені в таблиці 4.3.

Таблиця 4.3 Результати розрахунків загальної вартості проекту

| Найменування | Вартість (грн) |
|---|---|
| Заробітна плата | 30 000 |
| Оплата послуг | 261,44 |
| Матеріальні витрати | 186 |
| Вартість обслуговування обладнання | 577,92 |
| Всього витрачено: | 31 025,36 |

Тож собівартість створення системи голосового управління з набором базових функцій складає 31 025,36 гривень.

Якщо в систему впровадити рекламні акції партнерів, можна підвищити дохід, середня плата за просту рекламу (банер або виведення в топ місця (якщо це місце для відпочинку)) буде складати приблизно 144 гривні за 1000 показів. У випадку, якщо умова була виконана та за рік було виконано 365 тисяч показів, дохід додатку буде складати 52 560 гривні, але це у випадку рекламної кампанії лише одного партнера, тому чим більше партнерів буде задіяно, тим більше дохід буде від програми.

# РОЗДІЛ 5. ОХОРОНА ПРАЦІ

На людину діють різноманітні фактори виробничого середовища: фізичні, хімічні, біологічні та психофізичні. Інтегральна оцінка комбінованої дії всіх факторів дозволяє об’єктивно класифікувати умови праці за ступенями шкідливості та небезпечності. Крім того, використання в роботі принтерів, наявність вентиляційних установок, а також присутність великої кількості людей у приміщенні, яке обладнане великою кількістю комп’ютерів, можуть створити підвищений рівень шуму на робочому місці.

Одним з небезпечних факторів є небезпека ураження людини електричним струмом. Статичний струм, який виникає на поверхні матеріалів в результаті явища контактної електризації, також є небезпечним фактором, так як при контакті людини з наелектризованим приладдям від дії статичного струму виникають больові відчуття, в результаті яких людина може отримати травму від різких рухів.

Тому користувачі ПК та його керівник повинні знати про шкідливі та небезпечні фактори та про ефективні способи щодо захисту від них, що зменшить вірогідність отримання різноманітних професійних захворювань.

## 4.1 Аналіз небезпечних та шкідливих факторів

При аналізі шкідливих факторів, які впливають на умови праці персоналу, необхідно враховувати, що окрім фізичних факторів, на працівників та користувачів діють й психофізичні фактори. До них відноситься розумове перенапруження, яке виникає внаслідок напруженої творчої роботи в режимі діалогу з комп’ютером, підвищене навантаження на органи зору.

В ході аналізу шкідливих та небезпечних факторів праці, були виявлені найбільш ймовірні:

– недостатня освітленість (недостатня кількість приборів для освітлення, неправильне розташування робочого місця по відношенню до світла);

– підвищений рівень шуму (шум від роботи принтера, системного блоку комп’ютера, вентиляційних установок, розмови на фоні);

– дія шкідливого випромінювання (використання обчислювальної техніки в процесі трудової діяльності;

– відхилення від норми показників, які характеризують мікроклімат (протяг, відсутність кондиціювання повітря в літній період року, недостатнє опалення в зимній період);

– небезпека ураження електричним струмом;

– підвищений рівень статичного струму (накопичення електричного статичного заряду на корпусах комп’ютерів);

– розумова напруженість (робота з великим об’ємом інформації, необхідність концентрації уваги);

– підвищене навантаження на зір (тривала та одноманітна робота за комп’ютером).

Перелічимо основні порушення, які допускаються зі сторони адміністрації:

– майже по всім робочим місцям користувачів та робітників не проводиться атестація робочих місць за вимогами праці, а це означає, що існуючі порушення вимог безпеки не виявляються та не усуваються;

– більшість користувачів та робітників не знають, які шкідливі або небезпечні виробничі фактори діють на них за комп’ютеризованим робочим місцем;

– деякі робітники не знайомі з основами трудового кодексу про охорону праці, зі своїми правами, з обов’язками адміністраторів по забезпеченню нормальних умов праці;

– не завжди проводиться навчання безпечним прийомам та методам праці за ПК, а також працівники та користувачі не знають інструкцій, тоді коли робота

за ПК нерідко відноситься до категорії робіт з небезпечними та шкідливими умовами праці;

– приладдя яке знаходиться в експлуатації або щойно придбані монітори можуть не мати офіційного підтвердження, від компанії виробника, що їх безпечного використання.

Приміщення з комп'ютерами повинні мати природне та штучне освітлення. Освітленість на поверхні столу в зоні розміщення робочого документу повинна бути 300-500 лк. Для забезпечення нормованих значень освітленості в приміщенні варто проводити чистку вікон, віконних рам та світильників не рідше двох разів на рік та проводити своєчасну зміну ламп, які вийшли зі строю. Також необхідно щоб вікна в приміщенні були обладнані завісою, ролетами, зовнішніми козирками та ін. До ряду шкідливих факторів, з якими зштовхується людина, котра працює за монітором, відноситься рентгенівське та електромагнітне випромінювання, а також електростатичне поле. Допустимі норми для цих параметрів наведені в таблиці 4.1.

Таблиця 4.1. Допустимі норми

| Параметри | Допустимі значення |
|---|---|
| Потужність експозиційної дози рентгенівського випромінювання на відстані 0,05м навколо відеомонітора | 100мкР/год |
| Електромагнітне випромінювання на відстані 0,5м навколо відеомонітора за електричною складовою: в діапазоні 5Гц-2кГц | 25В |
| Електромагнітне випромінювання на відстані 0,5м навколо відеомонітора за електричною складовою: в діапазоні 2-400кГц | 2,5В |
| Електромагнітне випромінювання на відстані 0,5м навколо відеомонітора за магнітною складовою: в діапазоні 5Гц-2кГц | 250нТл |
| Електромагнітне випромінювання на відстані 0,5м навколо відеомонітора за магнітною складовою: в діапазоні 2-400кГц | 25нТл |
| Поверхневий електростатичний потенціал | Не більше 500В |

При роботі монітора виникає й електростатичне поле. Рівні його напруженості невеличкі й сильно не впливають на організм людини на відміну

від більш високих рівнів електростатичного поля, які характерні для промислових факторів. До показників мікроклімату відносяться: температура, вологість, рухливість повітря та теплове випромінювання. Оцінку мікроклімату здійснюють окремо для теплої та холодної пори року. Оптимальні та допустимі показники величин мікроклімату для приміщення, повинні відповідати значенням вказаним в таблиці 4.2 для категорії роботи, яка виконується сидячі та не вимагає фізичної напруги, при яких витрата енергії складає до 120ккал/год.

Таблиця 4.2. Оптимальні норми мікроклімату для приміщення

| Пора року | Температура, °С | Відносна вологість, % | Швидкість руху, м/с | | | |
|---|---|---|---|---|---|---|
| | | | Допустима | Оптимальна | Допустима | Оптимальна |
| Холодна | 22-24 | 21-25 | 40-60 | 75 | 0,1 | 0,1 |
| Тепла | 23-25 | 22-28 | 40-60 | 55 при 28°С | 0,1 | 0,1-0,2 |

### 4.2 Заходи щодо техніки безпеки

Перед початком роботи з ПК, потрібно знати основні пункти охорони праці, техніки безпеки та загальні правила роботи з ПК:

– якщо ви близько сидите за комп'ютером, то йде сильне навантаження на очі від монітора і може погіршитися зір;

– якщо ви довго сидите за комп'ютером, то виникає навантаження на хребет;

– підвищений рівень шуму при роботі комп'ютера і принтера створює навантаження на мозок;

– електромагнітне випромінювання від монітора може також привести до захворювань;

– пил, що виникає при роботі з комп'ютером, призводить до зниження імунітету;

– комп'ютер працює під високою напругою, а при несправній проводці може виникнути загоряння монітора і поразки користувача електрострумом.

З оглядом на ці небезпеки, слід знати правила роботи з комп'ютером:

– правильно встановіть комп'ютер: задня частина монітора не повинна бути спрямована на людей і місце їх відпочинку; дивани і ліжка повинні знаходитися на відстані не менше 2,5 - 3 метрів від комп'ютера; на екрані повинен бути захисний фільтр;

– комп'ютер необхідно підключити через розетку і пілот;

– обов'язково в приміщенні повинно бути увімкнене світло і повинна горіти настільна лампа;

– при роботі з комп'ютером треба дотримуватися правильної постави;

– відстань від очей монітора повинна бути не менше півметра;

При користуванні засобами обчислювальної техніки і периферійним обладнанням кожен працівник повинен уважно і обережно поводитися з електропроводкою, приладами і апаратами і завжди пам'ятати, що нехтування правилами безпеки загрожує і здоров'ю, і життя людини

Щоб уникнути ураження електричним струмом необхідно знати і виконувати наступні правила безпечного користування електроенергією:

– необхідно постійно стежити за справним станом електропроводки, вимикачів, штепсельних розеток, за допомогою яких обладнання включається в мережу, і заземлення;

– щоб уникнути пошкодження ізоляції проводів і виникнення коротких замикань забороняється:

а) вішати що-небудь на дроти;

б) зафарбовувати і білити шнури і дроти;

в) закладати дроти і шнури за газові і водопровідні труби, за батареї опалювальної системи;

г) висмикувати вилку з розетки за шнур, зусилля повинно бути додано лише до корпусу вилки;

– для виключення ураження електричним струмом забороняється:

а) часто вмикати і вимикати комп'ютер без необхідності;

б) торкатися до екрану і до тильної сторони блоків комп'ютера;

в) працювати на засобах обчислювальної техніки і периферійному обладнанні мокрими руками;

г) класти на засоби обчислювальної техніки і периферійне обладнання сторонні предмети;

– забороняється перевіряти працездатність електроустаткування в непристосованих для експлуатації приміщеннях із струмопровідними підлогами, сирих, що не дозволяють заземлити доступні металеві частини;

– ремонт електроапаратури проводиться тільки фахівцями-техніками з дотриманням необхідних технічних вимог;

– неприпустимо під напругою проводити ремонт засобів обчислювальної техніки і периферійного обладнання;

– щоб уникнути ураження електричним струмом, при користуванні електроприладами не можна торкатися одночасно будь-яких трубопроводів, батареї опалення, металевих конструкцій, з'єднаних з землею;

– при користуванні електроенергією в сирих приміщеннях дотримуватися особливої обережності;

– при виявленні обірваного проводу необхідно негайно повідомити про це адміністрацію, вжити заходів по виключенню контакту з ним людей;

– порятунок потерпілого при ураженні електричним струмом головним чином залежить від швидкості звільнення його від дії струмом.

У всіх випадках ураження людини електричним струмом негайно викликають лікаря. До прибуття лікаря потрібно, не гаючи часу, приступити до надання першої допомоги потерпілому.

Необхідно негайно почати робити штучне дихання, найбільш ефективним з яких є метод «рот в рот» або «рот в ніс», а також зовнішній масаж серця.

Штучне дихання ураженому електричним струмом проводиться аж до прибуття лікаря.

Кожен працівник навчального закладу, який виявив пожежу або її ознаки (задимлення, запах горіння або тління різних матеріалів, підвищення температури в приміщенні тощо), зобов'язаний:

– негайно повідомити про це за телефоном до пожежної частини (при цьому слід чітко назвати адресу об'єкта, місце виникнення пожежі, а також свою посаду та прізвище);

– задіяти систему оповіщення людей про пожежу, розпочати самому і залучити інших осіб до евакуації людей з будівлі до безпечного місця згідно з планом евакуації;

– сповістити про пожежу керівника закладу, установи і організації або працівника, що його заміщує;

– організовувати зустріч пожежних підрозділів, вжити заходів до гасіння пожежі наявними в установі засобами пожежогасіння.

Керівник закладу, установи чи організації або працівник, що його заміщує, який прибув на місце пожежі, зобов'язаний:

– перевірити чи повідомлено до пожежної охорони про виникнення пожежі;

– здійснювати керівництво евакуацією людей та гасінням пожежі до прибуття пожежних підрозділів. У разі загрози для життя людей негайно організувати їх рятування, використовуючи для цього всі наявні сили і засоби;

– організувати перевірку наявності всіх учасників навчально-виховного процесу, евакуйованих з будівлі, за списками і журналами обліку навчальних занять;

– виділити для зустрічі пожежних підрозділів особу, яка добре знає розміщення під'їзних шляхів та джерел води;

– перевірити включення в роботу автоматичної (стаціонарної) системи пожежогасіння;

– вилучити з небезпечної зони всіх працівників та інших осіб, не зайнятих евакуацією людей та ліквідацією пожежі;

– у разі потреби викликати до місця пожежі медичну та інші служби;

– припинити всі роботи, не пов'язані з заходами щодо ліквідації пожежі;

– організувати відключення мереж електро- і газопостачання, зупинку систем вентиляції та кондиціонування повітря і здійснення інших заходів, які сприяють запобіганню поширенню пожежі;

– забезпечити безпеку людей, які беруть участь в евакуації та гасінні пожежі, від можливих обвалів конструкції, дії токсичних продуктів горіння і підвищеної температури, ураження електрострумом тощо;

– організувати евакуацію матеріальних цінностей із небезпечної зони, визначити місця їх складування і забезпечити, при потребі, їх охорону;

– інформувати керівника пожежного підрозділу про наявність людей у будівлі.

Під час проведення евакуації та гасінні пожежі необхідно:

– з урахуванням умови, що склалася, визначити найбезпечніші евакуаційні шляхи і виходи до безпечної зони в найкоротший термін;

– ліквідувати умови, які сприяють виникненню паніки;

– евакуацію людей слід починати з приміщення, у якому виникла пожежа, і суміжних з ним приміщень, яким загрожує небезпека поширення вогню і продуктів горіння;

– у разі гасіння слід намагатися у першу чергу забезпечити сприятливі умови для безпечної евакуації людей;

– утримуватися від відчинення вікон і дверей, а також від розбивання скла, в протилежному разі вогонь і дим поширяться до суміжних приміщень.

# РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

На відміну від будь-якого технологічного процесу та виробничого обладнання, сама по собі інформаційна система не може нести пряму екологічну загрозу навколишньому середовищу. Однак робота всіх електроприладів, до яких відносяться й цифрові пристрої та комп'ютери використовують електроенергію, отримання якої вже важливе питання щодо екологічності процесу. Розглянемо, які негативні наслідки з'являються після роботи будь-якого електроприладу та основні заходи безпеки, які приймаються для виключення їх негативного впливу.

Енергетика входить, як підсистема в глобальну систему життєдіяльності. Розвиток та життя суспільства на теперішній час неможливі без енергетики, яка рухає прогрес всього народного господарства. Однак при ознайомленні з перевагами енергетики, необхідно враховувати також й негативні сторони впливу енергетики на навколишнє середовище.

На сьогоднішній день, шляхом спалювання палива (включаючи дрова та інші біоресурси) виробляють близько 90% енергії. Не дивлячись на те, що спалювання палива є основним джерелом енергії, цей метод також є найважливішим постачальником забруднюючих речовин в навколишнє середовище. Теплові електростанції являються фактором, який посилює парниковий ефект.

У викидах, які потрапляють в атмосферу при роботі ТЕС, міститься достатньо велика кількість металів та їх з'єднань. Ці речовини потрапляють до організмів людей та тварин через повітря, воду та ґрунт. Окрім цього найбільш частими видами палива вважаються природний газ, нафта, кам'яне вугілля, сланець, торф.

Одне з найважливіших впливів гідроенергетики пов'язано з відведенням значної площі під водосховища. На цих місцях знищенні природні екосистеми. Величезні площі земель поряд з побудованими водосховищами відчувають підтоплення через підвищений рівень ґрунтових вод. Ці землі часто переходять в категорію заболочених. Таким чином при будівництві водосховищ

відбувається різке порушення гідрологічного режиму річок, знищення екосистем та водного складу.

Основні екологічні проблеми АЕС пов’язані з утилізацією відходів, а також з ліквідацією самих АЕС після завершення строків їх експлуатації. Не дивлячись на те, що це достатньо екологічний вид енергетики можна назвати наступні негативні наслідки атомних електростанцій на оточуюче середовище:

– знищення екосистем та їх елементів в місцях видобування копалин;

– використання великих площ земель для побудови самих АЕС;

– використання значних об’ємів вод з різноманітних джерел та їх підігрівання;

– радіоактивне забруднення атмосфери, гідросфери та ґрунту

Сьогодні висока вірогідність збільшення долі вуглю та інших менш екологічно чистих видів палива в отриманні енергії. У зв’язку з цим розглядаються різноманітні шляхи та способи їх використання, які дозволяють значно зменшити негативний вплив на середовище. Ці способи базуються в основному на вдосконаленні технологій підготовки палива та ліквідації шкідливих відходів. Такими способами являються наступні:

1. використання та удосконалення очисних пристроїв. Сьогодні на багатьох ТЕС за допомогою фільтрів різного виду усуваються в основному шкідливі викиди;

2. хімічні або фізичні методи очищають вугілля та інші види палива від сірки зменшуючи потрапляння в атмосферу з’єднань сірки. Схожими методами можливо вилучити з палива 50-70% сірки до моменту спалювання;

3. економія електроенергії. Зменшити постачання забруднень в оточуюче середовище можливо лише за допомогою економії електроенергії. Також реально здійснити економію енергії за рахунок зменшення металомісткості в продукції, підвищуючи її якість та збільшуючи тривалість життєвого циклу виробів. В перспективі енергозбереження у зв’язку з переходом на наукоємні технології, пов’язані з використанням комп’ютерних технологій та інших пристроїв;

4. використання альтернативних видів отримання енергії, що значно знизить негативний вплив на навколишнє середовище. При використанні відновлюваних нетрадиційних джерел енергії зменшуються викиди різноманітних шкідливих речовин, в тому числі парникових газів, що значно збільшить якість повітря в містах та зонах відпочинку.

Раціональне використання енергії, зменшення використання енергоносіїв, використання технологій, які не причиняють значної шкоди навколишньому середовищу, являються важливими складовими в сфері охорони навколишнього середовища.

Варто приділити увагу також підприємствам по роботі з сировиною. Адже в світі з'являється все більше техніки, яка з часом замінюється новою, а підприємства працюють без зупинки виготовляючи нові пристрої.

На даний момент на багатьох підприємствах створені системи впровадження безпечних в екологічному відношенні технологічних процесів.

Для ефективної реалізації цих задач необхідно, щоб кожен спеціаліст та керівник будь-якого профіля мав глибокі знання щодо охорони навколишнього середовища, знали природоохоронне законодавство, вміло користуватись ними в повсякденній діяльності. На сьогоднішній день розроблені методи по захисту навколишнього середовища:

– зменшення робіт на другорядному виробництві, яке забруднює атмосферу;

– безперебійна робота очисних споруд;

– посилення контролю за якістю робіт пилоочисних систем;

– зменшення руху транспорту по території підприємства.

Для зменшення засміченості виробничої площі та навколишнього середовища, відходи виробництва кожен рік впроваджуються нові методи по їх утилізації:

– сміття після прибирання території підприємства вивозити на звалище;

– відходи металу пресувати та відправляти на переплавлення;

– відроблені рідкі речовини збираються в спеціальний контейнер та підлягають подальшій переробці;

– стокові води підприємства проходять фільтрацію чи фізико-хімічну обробку;

– для очистки пилу використовуються пилоочисні системи.

Одним з найважливіших шляхів раціонального використання матеріальних ресурсів являється їх комплексна обробка. При цьому головне місце відводиться питанням збору, зберігання та переробки виробничих відходів.

# ВИСНОВКИ

В ході роботи було виконано дослідження типового голосового асистента, також були приведені основні принципи роботи актуальних систем голосового управління, приведені їх приклади та проаналізовані їх функції та особливості, виявлені їх головні недоліки та запропоновані методи їх вирішення.

Також були розглянуті способи досягнення поставлених задач, проаналізовані методи роботи допоміжних інструментів для створення системи голосового управління, побудована та розглянута структура проекту, а також побудовано алгоритм роботи голосового асистента.

В рамках роботи була розроблена система голосового управління , проведено тестування та оцінка можливостей роботи алгоритмів програми, перевірка якості розпізнання мовлення.

Основною задачею було створення додатку з високим рівнем розпізнання мовлення та швидким виконанням команд. Програма показала високі результати під час тестувань, розпізнавання та синтез мови працюють без помилок, всі команди виконуються без проблем.

Виконано аналіз можливих шляхів отримання доходів від системи, а також було зроблено розрахунок економічної ефективності. На початковому етапі програма не буде приносити дохід, але робота з рекламними партнерами збільшить дохід від системи за короткий час.

Приділена увага охороні праці, досліджено основні небезпечні та шкідливі фактори при роботі з електронною технікою, а також розглянуті заходи щодо безпечної роботи з електроприладами, а також розглянуті проблеми охорони навколишнього середовища та шляхи їх рішення.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Поначугін О. В., Пічужкіна Д. Ю., Смєкалова К. С. Наука без меж. Голосовий помічник як технологія обробки даних. 2020. 96 с.
2. Поляков Є. В., Мажанов М. С., Качалова М. В., Поляков С. В. Розробка інтелектуального голосового асистента та дослідження можливості алгоритмів розпізнавати мовлення. 2017.
3. Крогер Д., Люц О., Рашки Ф. Наслідки аналізу голосу та мовлення для конфіденційності – розкриття інформації шляхом спостереження. Конфіденційність та ідентифікація. 2020. 242-244.
4. Смірнов І. В., Панов О. І., Скриннік О. О., Чистова К. В. Персональний когнітивний асистент: концепція та принципи роботи. Інформатика та її застосування. 2019. 105-107.
5. Восков Л. С., Соціальні мережі Web речей. Нові інформаційні технології. 2013. 53-58.
6. Поляков Є. В., Дослідження методів машинного навчання для аналізу та прийняття рішень на основі даних інтернету. 2017. 66-68.
7. Ассефі М., Гуанчі Л., Вітті М. П., Ізурієта К., Експериментальна оцінка розпізнавання мовлення Apple та Google. 2015. 3-4.
8. Чунь Х., Чи можна довіряти Alexa. Комп’ютерні технології. 2017. 100-104.
9. Арріані А. А., Мушбах М. С., Використання технологій розпізнавання голосу для мережі розумного будинку. 2016. 1-6.
10. Демпсі П., Персональний асистент Google Home Інженерія та технології. 2017. 80-81.
11. Лопез Г., Куессада Л., Гуереро Л. А., Alexa проти Siri проти Cortana проти Google Assistant: порівняння користувацьких інтерфейсів на основі мовлення. 2017. 241-250.
12. Цапенко М. П., Вимірювальні інформаційні системи. 2005. 440 с.
13. Лідовський В. І., Теорія інформації. 2002. 120 с.
14. Солоніна А. І., Основи цифрової обробки сигналів. 2012. 299 с.

## ДОДАТОК А. ФУНКЦІЯ РОЗПІЗНАВАННЯ МОВИ

```
from fuzzywuzzy.fuzz import ratio as rat


class cmd_reccognizer:
    def __init__(self):
        pass


    def clear_cmd(self, global_task):
        def clear_cmd3(k):
            tasks = []
            task = ''
            if len(k.split()) < 2:
                return [k]
            elif k != '':
                k = k.split()
                for i in range(-1, len(k)-2):
                    if (i+1 == len(k)-2):
                        if k[i+2]:
                            task = f'{task} {k[i+1]} {k[i+2]}'
                        else:
                            task = f'{task} {k[i+1]}'
                        tasks.append(task.strip())
                    else:
                        for j in range(len(self.keywords)):
```

```
            if (rat(k[i+1], self.keywords[j]) > 80):
                tasks.append(task.strip())
                task = ''
                break
            if k[i+2]:
                task = f'{task} {k[i+1]} {k[i+2]}'
            else:
                task = f'{task} {k[i+1]}'
    for i in range(len(tasks)):
        task = tasks[i].split()
        for dl in ('уйди', 'уходи'):
            if task:
                if rat(task[-1], dl) > 70:
                    tasks[i] = (tasks[i].replace(task[-1], '')).strip()
                    tasks.append('не мешай')
        if task:
            if rat(task[-1], 'мешай') > 80:
                if rat(task[-2], 'не') > 80:
                    tasks[i] = (tasks[i].replace(task[-1], '')).strip()
                    tasks[i] = (tasks[i].replace(task[-2], '')).strip()
                    tasks.append('не мешай')

    for i in range(len(tasks)):
        if tasks[i].endswith(' и'):
            ntask = tasks[i].split()
```

```
                del ntask[len(ntask)-1]
                tasks[i]=(' '.join(ntask))
        for i in range (len(tasks)):
            if tasks[i].split():
                if fuzz.ratio(tasks[i].split()[-1], 'потом') > 80:
                    ntask = tasks[i].split()
                    del ntask[-1]
                    tasks[i] = ' ',join(ntask)
    if (tasks and k):
        tasks[-1]=f'{tasks[-1]} {k[-1]}'
    return list(filter(None, tasks))


def main(k):
    new_list = clear_cmd3(k)
    taskss = []
    for task in new_list:
        var = clear_cmd3(task)
        for i in var:
            taskss.append(i)


    new_list = []
    for task in taskss:
        var = clear_cmd3(task)
        for i in var:
            new_list.append(i)
```

```
    taskss = []
    for task in new_list:
        task = task.split()
        if len(task) == 1:
            taskss.append(" ".join(task))
        else:
            n_task = ' '
            for var in range(len(task)):
                if (task[var] in self.keywords):
                    taskss.append(n_task.strip())
                    n_task = task[var]
                elif (var == len(task)-1):
                    n_task = f'{n_task} {task[var]}'
                    taskss.append(n_task.strip())
                else:
                    n_task = f'{n_task} {task[var]}'
    return [value for value in taskss if value]


def delete_doubles(ans):
    for task in range(len(ans)):
        wars = []
        last = ''
        var = ans[task].split()
        for word in var:
            if word != last:
```

```
                wars.append(word)
            last = word
        ans[task] = ' '.join(wars)
    return ans


def delete_doubles2(ans):
    last = ''
    tasks = ans
    ans = []
    for task in tasks:
        if task != last:
            ans.append(task)
            last = task
        else:
            pass
    return ans


def task_interpreter(ans):
    def get_pairs(talk):
        double_keys = [['как', 'дела'],
                       ['что', 'делаешь']]
        ans = []
        flag = 0
        for double in double_keys:
            pairs = [' '.join(double).strip(), ' '.join(double[::-1]).strip()]
```

```
        for pair in pairs:
            if pair in task:#если пара есть в запросе
                flag = 1
                var = task.replace(pair, '').strip()

                if task.startswith(pair):
                    ans.append([pair,var])
                elif task.endswith(pair):
                    ans.append([var, pair])

    if flag == 0:
        ans.append(task)

    return ans

def rearkh(mas):
    def rearkh_cycle(ans):
        nans = ans
        ans = []
        for val in nans:
            if type(val) == list:
                for i in val:
                    ans.append(i)
            else:
                ans.append(val)
```

```
        return ans


    def check(mas):
        flag = 0
        for val in mas:
            if type(val) == list:
                flag = 1
        return flag


    while check(mas):
        mas = rearkh_cycle(mas)


        return mas


    for i in range(len(ans)):
        ans[i] = get_pairs(ans[i])
    return rearkh(ans)


ans = delete_doubles(main(global_task))
ans = delete_doubles2(task_interpreter(ans))
n_ans = [x for x in ans if not(x in self.keywords)]


ans = []
for task in n_ans:
```

```
            if task.endswith(' и'):
                task = (task[::-1].replace('и ', '', 1))[::-1]
            if task.startswith('и '):
                task = task.replace('и ', '', 1)

            for i in [' сначала ', ' и ', ' потом ', ' давай ', 'сделай', 'старая', 'ладно']:
                if ((i in task) and (not (task.split()[0] in ['найди', 'расскажи', 'добавь',
'удали']))):
                    task = task.replace(i, ' ').strip().replace(' ', '')
                if fuzz.ratio(task, 'юми') < 70:
                    task = task.replace('юми', ' ').strip()

            ans.append(task.strip().replace(' ', ''))

        if 'юми' in ans:
            if len(ans) > 1:
                del ans[ans.index('юми')]
        return ans
```

## ДОДАТОК Б. МАСИВ КЛЮЧОВИХ СЛІВ

```
import speech_recognition as sr
from win32com.client import Dispatch as d
from fuzzywuzzy.fuzz import ratio as rat
from random import choice

from command_recognizer import *
from virtual_class import VirtualAssistent

class Assistent(VirtualAssistent, cmd_reccognizer):

    def __init__(self):
        self.cmds = {
            ('не мешай', 'уйди', 'я ушел', 'я ушла', 'пока', 'уходи', 'прощай', 'до встречи', 'до скорого', 'увидимся') : self.bye,
            ('привет', 'здравствуй', 'я пришел', 'я пришла', 'доброе утро', 'добрый день', 'добрый вечер', 'приветствую') : self.hello,
            ('поздоровайся', 'поприветствуй всех', 'поздоровайся со всеми', 'поприветствуй') : self.greet,
            ('как дела', 'как ты', None) : self.how_are_you,
            ('что делаешь', 'чем занимаешься', 'чем занята') : self.what_doing,
            ('спасибо', 'благодарю') : self.thanks,
            ('молодец', 'умница', 'какая ты молодец', 'какая умница') : self.well_done,
        ('завершай работу', 'давай спать', 'выключи систему', 'выключи всё') : self.shut_down,
```

```
('заблокируй компьютер', 'блокировка', 'заблокируй', 'заблокируй экран') : self.lock,

('открой youtube', 'зайди на youtube', 'включи youtube', 'зайди в youtube', 'youtube') : self.open_youtube,

('день недели', 'какой день недели', 'какой сегодня день недели', 'какой сегодня день') : self.day_of_week,

('сколько время', 'время', 'сколько времени', 'какое сейчас время', 'какой сейчас час') : self.time_now,

('число', 'какое сегодня число', 'какое число', 'какая сегодня дата') : self.date_is_today,

('повтори', 'ещё раз', 'повтори ещё раз', 'ну ка ещё раз', 'ну ка повтори', 'повтори пожалуйста', 'что ты говорила') : self.repeat,

('юми', 'уми') : self.first_ans

}

self.keywords = [

'включи', 'уйди', 'включай', 'выключи', 'выключай', 'открой',

'отключи', 'отключай', 'открывай', 'запусти', 'запускай', 'запустить',

'умница', 'помоги', 'поприветствуй', 'открыть', 'расскажи', 'скажи',

'объясни', 'рассказывай', 'поздоровайся', 'здравствуй', 'привет', 'добавь',

'удали', 'сложи', 'посчитай', 'вычисли', 'умножь', 'подели',

'раздели', 'прибавь', 'вычти', 'молодец']


self.strs = {

1 : 'первое', 2 : 'второе', 3 : 'третье', 4 : 'четвёртое', 5 : 'пятое', 6 : 'шестое', 7 : 'седьмое', 8 : 'восьмое',

9 : 'девятое', 10 : 'десятое', 11 : 'одинадцвтое', 12 : 'двенадцатое', 13 : 'тринадцатое', 14 : 'четырнадцатое',
```

15 : 'пятнадцатое', 16 : 'шестнадцатое', 17 : 'семнадцатое', 18 : 'восемнадцатое', 19 : 'девятнадцатое', 20 : 'двадцатое',

21 : 'двадцать первое', 22 : 'двадцать второе', 23 : 'двадцать третье', 24 : 'двадцать четвёртое',

25 : 'двадцать пятое', 26 : 'двадцать шестое', 27 : 'двадцать седьмое', 28 : 'двадцать восьмое', 29 : 'двадцать девятое',

30 : 'тридцатое', 31 : 'тридцать первое'

}

self.one_time_executable = [

('не мешай', 'уйди', 'я ушел', 'я ушла', 'пока', 'уходи', 'прощай', 'до встречи', 'до скорого', 'увидимся'),

('привет', 'здравствуй', 'я пришел', 'я пришла', 'доброе утро', 'добрый день', 'добрый вечер', 'приветствую'),

('поздоровайся', 'поприветствуй всех', 'поздоровайся со всеми', 'поприветствуй'), ('как дела', 'как ты', None),

('что делаешь', 'чем занимаешься', 'чем занята'), ('спасибо', 'благодарю'),

('молодец', 'умница', 'какая ты молодец', 'какая умница'), ('юми', 'уми'),

('завершай работу', 'давай спать', 'выключи систему', 'выключи всё'),

('заблокируй компьютер', 'блокировка', 'заблокируй', 'заблокируй экран'),

('день недели', 'какой день недели', 'какой сегодня день недели', 'какой сегодня день'),

('число', 'какое сегодня число', 'какое число', 'какая сегодня дата')

]

self.double_keys = [['как', 'дела'],

```
                ['что', 'делаешь'],
                ['чем', 'занимаешься']]

    self.phrases = []
    self.last_phrase = ''
    self.r = sr.Recognizer()
    self.r.pause_threshold = 0.5
    self.m = sr.Microphone()
    with self.m as self.sourse:
        self.r.adjust_for_ambient_noise(self.sourse)
    self.engine = d("SAPI.SpVoice")

def cmd_exe(self, tasklist):

    tasklist = self.delete_doubles2(tasklist)
    chance = randint(0,100) in range(5)
    is_ans = []
    for ans in tasklist:
        if ans in self.is_answer:
            is_ans.append(ans)

    if chance:
        self.not_answer = True
        if is_ans:
            for task in is_ans:
```

```
                self.cmds[task]()
            else:
                if tasklist:
                    self.add_phrases(choice(['Извините я вас прослушала', 'Простите не услышала']))
        else:
            for task in tasklist:
                self.cmds[task]() #простое выполнение распознанных команд
        self.talk()


    def run(self):
        try:
            self.cmd_exe(self.clear_cmd(self.listen()))
        except Exception as e:
            self.error_log(e)
            self.add_phrases(f"Произошла ошибка! {choice(['файл ошибок обновлён', 'проверьте файл ошибок', ''])}")
            self.talk()
```

## ДОДАТОК В. ІНІЦІАЛІЗАЦІЯ КОМАНД

```
import time, ctypes, webbrowser as web
from fuzzywuzzy.fuzz import ratio as rat
from random import choice
from colorama import *
from os import system, getpid
from datetime import datetime
from psutil import virtual_memory as ozu

class VirtualAssistent:

    def __init__(self):
        pass

    def talk(self):
        for text in self.phrases:
            self.engine.Speak(text)
        self.phrases = []

    def add_phrases(self, text):
        self.last_phrase = text
        self.phrases.append(text)

    def first_ans(self):
```

```
        vals = ('я вас слушаю', 'да-да', 'я здесь')
        self.add_phrases(choice(vals))

    def well_done(self):
        vals = [
            "Спасибо, так приятно!", "Вы тоже ничего...", "Спасибо Вам большое!"
        ]
        self.add_phrases(choice(vals))

    def bye(self):#выход из программы
        self.engine.Speak(choice(['До свидания']))
        system(f'taskkill /IM {getpid()} /F') #принудительно завершаем процесс по его id через cmd

    def hello(self):#приветствие
        time = int(datetime.now().hour)
        if time in (6,7,8,9,10,11,12):
            self.add_phrases(choice(["Доброе утро","Здравствуйте"]))
        elif time in (13,14,15,16,17,18):
            self.add_phrases(choice(["Добрый день"]))
        elif time in (19,20,21,22,23):
            self.add_phrases(choice(["Добрый вечер"]))
        else:
            self.add_phrases(choice(["Доброй ночи"]))
```

```
def greet(self):
    vals = [
        'Здравствуйте, рада вас слышать', 'Всем привет, я Юми',
        'Доброго времени суток'
    ]
    self.add_phrases(choice(vals))

def how_are_you(self): #как дела
    vals = [
        "Всё отлично!", "Всё хорошо, спасибо что спросили"
    ]
    self.add_phrases(choice(vals))

def what_doing(self): #Что делаешь
    vals = ["Именно сейчас? Отвечаю Вам на поставленный вопрос",
        "Жду когда Вы освободитесь", "Выполняю задания в геншине"]
    self.add_phrases(choice(vals))

def thanks(self): #Спасибо
    vals = ['Рада была вам помочь', 'Обращайтесь в любое время',
    'Всегда к вашим услугам', 'Не за что Всегда пожалуйста']
    self.add_phrases(choice(vals))

def shut_down(self): #выключение компьютера
    self.engine.Speak('Вы уверенны, что хотите выключить компьютер?')
```

```
    answer = ''

    while answer == '':
        answer = self.listen()

        if answer:
            for val in ('да', 'да выключай', 'выключай', 'да уверенна'):
                if rat(val, answer) > 90:
                    self.off_light()
                    system('shutdown /s /f /t 10')
                    break

def lock(self): #блокировка
    ctypes.windll.user32.LockWorkStation()

def open_youtube(self):#открытие ютуб
    web.open(f'https://www.youtube.com/')

def create_news(self):#рассказать новости
    pass

def create_news(self):#парсинг новостей
    pass

def day_of_week(self):#день недели
```

```
    weekdays = {
        0 : 'понедельник', 1 : 'вторник', 2 : 'среда', 3 : 'четверг',
        4 : 'пятница', 5 : 'суббота', 6 : 'воскресенье'
    }
    self.add_phrases(f'Сегодня {weekdays[datetime.today().weekday()]}')

def time_now(self):#текущее время
    now = datetime.now()
    self.add_phrases(f'Сейчас {now.hour} {now.minute}')

def date_is_today(self):#текущая дата
    self.add_phrases(f'Сегодня {self.strs[datetime.now().day]} число')

def repeat(self):#повторяет последнее что говорила
    self.add_phrases(self.last_phrase)

def web_talk(self, task):#рассказывает, то что найдёт по запросу
    #task=запрос
    pass

def web_search(self, task):#открывает поисковик с заданным запросом
    mode = 'web'

    for i in range(len(task.split())-1):
        if (rat(task.split()[i], 'найди')> 80) and (rat(task.split()[i+1], 'видео') > 80):
```

```
            task = task.split()
            del task[i]
            del task[i+1]
            task = ' '.join(task)
            mode = 'tube'
            break

    for var in ('найди на youtube', 'найди в youtube', 'найди видео', 'найди видео на канале'):
        if var in task:
            mode = 'tube'
            task = task.replace(var, '').replace('  ',' ').strip()

        if mode == 'web':
            for var in ['найди пожалуйста', 'найди кто такой', 'найди кто такая', 'найди кто такие',
                        'найди что такое', 'найди как сделать', 'найди как', 'найди', 'поищи', 'почему']:
                task = task.replace(var, '').replace('  ', ' ').strip()

            if task:
                self.engine.Speak(choice(['Уже ищу', 'Сейчас найду', 'Сейчас поищем']))
                if float(ozu().percent) <= 95:
                    web.open(f'http://google.com/search?btnG=1&q={task}')
                else:
```

```
                self.engine.Speak(choice([
                    'Компьютер сильно перегружен, не стоит открывать новые вкладки в браузере.',
                    'Компьютер сильно нагружен, давайте я лучше почитаю?']))

                answer = ''
                while not(answer):#ждёт ответа
                    answer = self.Listen()

                flag = 0
                for var in ('нет', 'не нужно', 'не надо'):
                    if rat(var, answer) > 70:
                        self.engine.Speak('Хорошо')
                        flag = 1
                        break

            if flag == 0:
                for var in ('да', 'давай', 'можно', 'рассказывай', 'расскажи', 'читай', 'прочитай'):
                    if rat(var, answer) > 70:
                        self.web_talk(task)
                        break
        else:
            if float(ozu().percent) <= 95:
                for var in ('видео про ', 'видео о ', 'на канале', 'видео'):
```

```
                if var in task:
                    task = task.replace(var, '').replace('  ', ' ').strip()
            self.engine.Speak(choice(['Уже ищу', 'Сейчас найду', 'Сейчас поищем']))
            web.open(f'https://www.youtube.cpm/results?search_query={task}')
        else:
            self.engine.Speak(choice(['Не стоит открывать новые вкладки, компьютер сильно перегружен',
            'Если я открою youtube, компьютер может зависнуть']))


    def listen(self):
        system('cls')
        text = ''
        print(choice((Fore.GREEN, Fore.WHITE, Fore.YELLOW)) + 'Я вас слушаю: ')
        with self.m as self.sourse:
            self.r.adjust_for_ambient_noise(self.sourse)
        while text == '':
            with self.m as self.sourse:
                audio = self.r.listen(self.sourse)
                try:
                    text = (self.r.recognize_google(audio, language="ru-RU")).lower()
                except:
                    pass
        if text in ('эй', 'юми', 'ты здесь', 'ты тут', 'слушай', 'слышишь', 'юми ты здесь', 'юми ты тут'):
            self.first_ans()
```

```
        return ''
    else:
        return( text )


def delete_doubles2(self, ans):
    #удаление повторяющихся запросов
    last = ''
    tasks = ans
    ans = []
    for task in tasks:
        if task != last:
            ans.append(task)
            last = task
        else:
            pass
    return ans
```