Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

# КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Програмно-технічний асистент для взаємодії із мовною моделлю на основі ESP32
Назва теми

КвРКІ 101053.21.01.03 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент III курсу, група КІ2с-21-1 _____ О. В. Дмитрук
Підпис Ініціали, прізвище

Керівник _____ П. Г. Регіда
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ І.О. Засорнова
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«31» травня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Г.О.Говорущенко

" 10 " 01 2024 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА**

Дмитруку Олександру Валентиновичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-технічний асистент для взаємодії із мовною моделю на основі ESP32

Керівник проекту (роботи) Регіда П.Г., старший викладач
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Програмно-технічний асистент для взаємодії із мовною моделлю на основі ESP32

Структурні частини пристрою

Програмно-апаратна реалізація програмно-технічного асистента для взаємодії із мовною моделлю на основі ESP32

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Діаграма станів програмно-технічного асистента

Архітектура ПЗ для програмно-технічного асистента

Принципова схема програмно-технічного асистента на основі ESP32

6. Консультанти розділів дипломного проекту (роботи)

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | Засорнова І.О., доцент кафедри КІІС |  |  |
| Антиплагіат | Нічепорук А.О., доцент кафедри КІІС |  |  |

7. Дата видачі завдання «_10_»_____01_____2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| №з/п | Назва етапів (розділів) дипломного проекту (роботи) | Термін виконання етапів проекту (роботи) | Примітка |
|---|---|---|---|
| 1 | Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником | 10.01.2024 | виконано |
| 2 | Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження | 01.02.2024 | виконано |
| 3 | Робота над розділом 1 – дослідження предметної області та постановка задачі | 01.03.2024 | виконано |
| 4 | Робота над розділом 2 – вибір компонентів для проєктування програмно-технічного асистента для взаємодії із мовною моделлю | 01.04.2024 | виконано |
| 5 | Робота над розділом 3 – огляд реалізованого програмного забезпечення та функціонування пристрою ESP32 | 29.04.2024 | виконано |
| 6 | Оформлення пояснювальної записки згідно вимог | 25.05.2024 | виконано |
| 7 | Попередній захист ВКР | 26.05.2024 | виконано |
| 8 | Захист ВКР на засіданні ЕК | Червень 2024 року | |

Студент  О. В. Дмитрук
Підпис Ініціали, прізвище

Керівник роботи  П. Г. Регіда
Підпис Ініціали, прізвище

| № рядка | формат | Позначення | Найменування | Кіл. листів | № екз | Примітка |
|---|---|---|---|---|---|---|
| | | | Текстові документи | | | |
| 1 | | КвРКІ 101053.21.01.03 ПЗ | Пояснювальна записка | 64 | | |
| | | | | | | |
| | | | Графічні матеріали | | | |
| 2 | | КвРКІ 101053.21.01.03 Е8 | Діаграма станів програмно-технічного асистента | 1 | | |
| 3 | | КвРКІ 101053.21.01.03 Е8 | Архітектура ПЗ для програмно-технічного асистента | 1 | | |
| 4 | | КвРКІ 101053.21.01.03 Е8 | Принципова схема програмно-технічного асистента на основі ESP32 | 1 | | |

| Зм | Арк | № докум | Підпис | Дата | КвРКІ 101053.21.01.03 ВП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Дмитрук | | 30.05 | Програмно-технічний асистент для взаємодії із мовною моделлю на основі ESP32 | Літера | Аркуш 1 / Аркушів 64 |
| Перевір. | | Регіда | | 30.05 | | У | |
| Н. контр. | | Засорнова | | 30.05 | | ХНУ, КІ2с-21-1 | |
| Затв. | | Говорущенко | | | | | |

## ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ

ШІ – штучний інтелект

ADC – аналого-цифровий перетворювач

Bluetooth – технологія бездротового радіо зв’язку між різноманітними типами електронних пристроїв

DAC – цифро-аналоговий перетворювач

I2C – послідовна шина даних для зв'язку інтегральних схем

I2S – Inter-IC Sound

IoT – інтернет речей

PWM – широтно-імпульсна модуляція

SPI – послідовний периферійний інтерфейс

UART – Універсальний асинхронний приймач, передавач

USB – універсальна послідовна шина

Wi-Fi – технологія даних по радіоканалах за рахунок цифрових потоків

# ВСТУП

Штучний інтелект (ШІ) востаннє декілька років перетворився з теоретичної концепції в реальну та невід'ємну частину нашого повсякденного життя. Галузь розвивається дуже динамічно, що веде до значних змін у багатьох сферах життя, включаючи технології, медицину, економіку та освіту. Однією з актуальних тенденцій в області ШІ є поєднання технологічних інновацій та інтерактивності, яке сприяє поліпшенню якості та швидкості обслуговування користувачів. На даний момент ШІ активно інтегрується в різноманітні пристрої та системи, забезпечуючи їх розумними можливостями та автоматизацією завдань. Прикладами можуть слугувати віртуальні помічники, системи розпізнавання зображень і мови, технології самокерованих транспортних засобів тощо. Застосування ШІ відкриває нові можливості для підвищення продуктивності, ефективності та якості послуг у різних галузях.

Серед важливих викликів, що виникають у цій області, особливе місце займає розробка програмно-технічних асистентів, спроможних взаємодіяти із мовними моделями та надавати ефективний інтерфейс для користувачів. Зокрема, виникає потреба в розробці технічних рішень, які забезпечують взаємодію з штучним інтелектом на базі мікроконтролера ESP32. Така взаємодія відкриває нові можливості для інтеграції ШІ в повсякденне життя, оскільки мікроконтролери є компактними, енергоефективними та доступними пристроями, які можна використовувати в різноманітних додатках.

Дана робота націлена на створення простого та ефективного інструмента, який забезпечить зручність використання та доступність для користувачів, що взаємодіють з програмним асистентом через текстовий інтерфейс. Результати роботи можуть сприяти подальшому розвитку та впровадженню ШІ в різні галузі, забезпечуючи при цьому простоту та ефективність використання. Крім того, такий інструмент може стати основою для створення більш складних систем, які поєднують переваги ШІ та вбудованих систем, таких як Інтернет речей (IoT) та

розумний дім.

Метою кваліфікаційної роботи є забезпечення працездатності програмно-технічного асистента, що взаємодіє з мовною моделлю за допомогою клавіатури, а результати виводити на OLED дисплей, використовуючи мікроконтролер ESP32.

Об'єктом дослідження є процес взаємодії з мовною моделлю через програмно-апаратний засіб на основі ESP32 із використанням клавіатури та підтримкою інтуїтивного інтерфейсу для виведення інформації на OLED дисплей.

Предметом дослідження є програмно-апаратний засіб, на основі ESP32, для взаємодії з мовною моделлю із використанням клавіатури та OLED дисплею.

Для досягнення поставленої мети використовуються методології програмування для організації взаємодії ESP32 та використаних апаратних засобів, принципи роботи із API мовної моделі. Це забезпечить ефективну взаємодію між апаратними компонентами та програмним забезпеченням, а також надасть можливість інтегрувати ШІ у систему.

Практичне значення полягає в створенні простого та ефективного інструменту, який забезпечить зручність використання та доступність для користувачів, що взаємодіють з програмним асистентом через текстовий інтерфейс. Результати роботи можуть сприяти подальшому розвитку та впровадженню ШІ в різні галузі, забезпечуючи при цьому простоту та ефективність використання. Такий інструмент може знайти застосування в побутових, освітніх, медичних та багатьох інших сферах, де потрібна проста та зручна взаємодія з штучним інтелектом. Крім того, розроблений пристрій може слугувати основою для створення більш складних систем, що інтегрують ШІ з Інтернетом речей або розумним домом. Завдяки поєднанню переваг ШІ та вбудованих систем, таких інструментів стане більше і вони стануть доступнішими для широкого кола користувачів, забезпечуючи більш ефективну інтеграцію передових технологій в повсякденне життя людей.

# 1 ПРОГРАМНО-ТЕХНІЧНИЙ АСИСТЕНТ ДЛЯ ВЗАЄМОДІЇ ІЗ МОВНОЮ МОДЕЛЛЮ НА ОСНОВІ ESP32

## 1.1 Актуальність розробки програмно-технічного асистента для взаємодії із мовною моделлю на основі ESP32

В умовах стрімкого розвитку технологій і зростаючого інтересу користувачів до інновацій у щоденному житті, розробка програмно-технічного асистента на базі ESP32 з OLED дисплеєм та клавіатурою стає актуальним завданням. Цей проєкт поєднує в собі потужність мовного інтерфейсу ChatGPT та гнучкість та можливості пристроїв Internet of Things (IoT), створюючи інноваційне рішення для взаємодії користувача з штучним інтелектом.

Розробка системи, яка може взаємодіяти з потужним мовним інтерфейсом, дозволяє користувачам отримувати відповіді на поставлені запитання, в текстовому форматі, тим самим роблячи взаємодію більш природною та ефективною. Інтеграція ChatGPT з ESP32 дозволяє створювати надзвичайно потужні та зручні пристрої для взаємодії з штучним інтелектом.

Завдяки своїй гнучкості та розширеним можливостям взаємодії, система може знайти застосування в різних галузях, включаючи освіту, медицину, підприємництво та особисте використання.

Розробка системи, що інтегрує ChatGPT з вищезазначеними компонентами, потребує ретельного врахування множини аспектів. Цей розділ описує ключові вимоги до проєкту, які необхідно врахувати для досягнення оптимальної ефективності, зручності взаємодії та надійності.

Асистент відкриває нові горизонти у спілкуванні з штучним інтелектом ChatGPT. Надає можливість вводити текстові запити, щоб отримати відповіді на будь-які питання, вести цікаві діалоги або генерувати текст різної складності. OLED дисплей чітко та зручно відображає всю необхідну інформацію, а клавіатура робить введення даних швидким та комфортним.

В основі системи лежить платформа ESP32, яка гарантує високу продуктивність та мобільність. Програмне забезпечення оптимізовано для роботи з обмеженими ресурсами, що робить систему енергоефективною та доступною для широкого кола користувачів. Завдяки безшовній інтеграції з OpenAI API система безперебійно зв'язується з ChatGPT, забезпечуючи безпечну передачу даних та захищаючи вашу конфіденційність.

Інтерфейс системи розроблений з урахуванням інтуїтивності та простоти використання. Кожен елемент дизайну продуманий до дрібниць, щоб зробити роботу з системою максимально комфортною та приємною. Простий та зрозумілий інтерфейс робить систему не лише функціональним інструментом, але й естетично приємним доповненням до вашого робочого простору.

Розробка програмно-технічного асистента може суттєво розширити можливості взаємодії людей з штучним інтелектом. Ця система має значний потенціал для використання в різних галузях, таких як освіта, медицина, бізнес, наука та особисте життя. Також робота може мати значний вплив на розвиток інтерфейсів користувача та взаємодії людей з штучним інтелектом. Розроблена система може стати основою для створення нових інноваційних продуктів та послуг.

Одним з ключових напрямків подальшого розвитку системи є розширення спектру взаємодії з ChatGPT. Планується вдосконалити мовний інтерфейс, щоб користувачі могли виражати свої запитання та команди більш вільно та природно. Розробка розпізнавання інтонації та контексту голосових команд, а також розвиток мовленнєвого аналізу для вдосконалення точності відповідей стануть важливою частиною цього процесу.

Для забезпечення додаткової безпеки та конфіденційності використанням системи, планується розробка механізмів шифрування та аутентифікації, щоб унеможливити несанкціонований доступ до важливої інформації.

Загалом, проєкт, відкриває широкі перспективи для розвитку інтерфейсів користувача та використання IoT-рішень. Постійна робота над удосконаленням

функціоналу та адаптація до змін у технологічному середовищі дозволить зберігати актуальність та конкурентоспроможність системи у майбутньому. Після завершення роботи планується продовжити роботу над системою, щоб розширити її можливості та функціональність. Також планується проаналізувати інші способи використання ChatGPT для створення нових інтерфейсів користувача.

## 1.2 Огляд існуючих рішень

### 1.2.1 Raspberry Pi для голосової взаємодії з ШІ

Система взаємодії з мовною моделлю ChatGPT вже суттєво перетворила спосіб, яким отримують інформацію та взаємодіють у вигляді чату. Більшість користувачів взаємодіють з нею, надсилаючи текстові запитання та отримуючи текстові або голосові відповіді. Однак, можливо, виникало питання про те, чи існує більш природний та інтуїтивний спосіб спілкування з ChatGPT.

Що, якщо можна було б використовувати голос для взаємодії з ChatGPT. Такий підхід дозволив би зробити взаємодію з моделлю більш схожою на розмову з людиною. Це значно підвищило б зручність та ефективність використання мовної моделі, особливо для тих, хто не може або не хоче користуватися клавіатурою. Використання голосового вводу та виводу може також зробити систему доступнішою для людей з обмеженими можливостями.

У цьому дослідженні розглядається можливість використання Raspberry Pi для голосової взаємодії з ChatGPT. Такий підхід дозволить створити компактну та автономну систему, яка може бути використана в різних умовах та для різних цілей. Для реалізації цього проєкту необхідні кілька ключових компонентів. Основою проєкту є Raspberry Pi 4B, який забезпечує необхідну обчислювальну потужність [1-3].

Для голосового вводу та виводу була використана USB аудіо-система, яка включає в себе мікрофон і динаміки. Це обладнання дозволяє ChatGPT вловлювати голос та надавати відповіді.

Для забезпечення безперебійного живлення та захисту від перепадів напруги використовується UPS батарея (система безперебійного живлення). Під час налаштування, конфігурації та розробки використовувався монітор, клавіатура та миша для взаємодії з Raspberry Pi. Ці пристрої можна відключити після того, як система буде готова до роботи. На рисунку 1.1 зображено сам пристрій у зібраному стані.

Рисунок 1.1 – Raspberry Pi for Voice Interaction

Ідея проєкту полягає в створенні голосового інтерфейсу для ChatGPT за допомогою Raspberry Pi. Замість введення текстових запитань необхідно говорити безпосередньо з штучним інтелектом, і він буде надавати відповідь через аудіосистему. Raspberry Pi виступає посередником між голосом та ChatGPT, обробляючи ввід та надаючи відповіді ШІ в реальному часі.

Використовуючи Raspberry Pi та технологію розпізнавання голосу, створено спосіб взаємодії з ChatGPT, що покращує взаємодію користувача та відкриває нові можливості для голосових застосувань ШІ в різних галузях. Проєкт складається з двох основних компонентів: ChatGPT та Raspberry Pi. ChatGPT - це велика мовна модель від OpenAI, яка генерує текст, перекладає мови, створює творчий контент

та відповідає на запитання. Raspberry Pi – це не просто комп'ютер. Його портативність та простота використання роблять його чудовим інструментом для втілення в життя будь-яких ваших задумів, від IoT-пристроїв до складних програмних систем. На рисунку 1.2 відображено вигляд Raspberry Pi.


Рисунок 1.2 – Raspberry Pi 4B

Цей проєкт доступний завдяки недорогому Raspberry Pi, що робить його доступним для широкого кола користувачів. Незважаючи на певні недоліки, Raspberry Pi може бути корисним у ряді проєктів. До них відносяться складність налаштування та питання конфіденційності, пов'язані з Google Assistant.

Після завершення проєкту користувачі можуть використовувати голосові команди для взаємодії з ChatGPT, наприклад, створювати вірші, перекладати тексти, створювати списки справ та отримувати відповіді на питання про погоду. Гнучкість Python-скрипта дозволяє легко модифікувати його для додавання нових функцій. Усупереч недолікам, цей проєкт становить наочний приклад використання ChatGPT для створення голосового інтерфейсу [4,5].

### 1.2.2 Інтеграція ChatGPT з Arduino IoT Cloud

Проєкт орієнтований на можливість взаємодії з ChatGPT за допомогою будь-якої плати Arduino та панелі Arduino IoT Cloud, надаючи користувачеві зручний інтерфейс для ставлення питань та отримання відповідей через пристрій Arduino.

Ключові компоненти та засоби включають у себе:

а) Arduino Nano RP2040 Connect: Ця платформа використовується для виконання програм та взаємодії з IoT Cloud;

б) Arduino IoT Cloud Remote та Arduino IoT Cloud: Платформа, що надає можливість віддаленого доступу та створення IoT Cloud;

в) ArduinoThread library: Ця бібліотека використовується для ефективного управління миготінням вбудованого світлодіода.

Суть проєкту полягає в використанні пристрою, який сумісний з Arduino IoT Cloud, наприклад, Arduino Nano RP2040 Connect або ESP32/ESP8266. Цей пристрій виступає посередником між IoT Cloud та мовною моделлю GPT-3.5 від OpenAI. Він отримує запитання з IoT Cloud, передає їх до OpenAI API, отримує та обробляє відповіді, а потім повертає їх назад в IoT Cloud. Створення пристрою на IoT Cloud включає такі кроки:

а) підключення Arduino Nano RP2040 Connect до комп'ютера та додавання його як пристрій на IoT Cloud;

б) створення та додавання необхідних змінних для тексту питання, тексту відповіді та відправлення питання;

в) встановлення параметрів мережі для доступу до Wi-Fi.

Далі, в IoT Cloud створюється панель управління з відповідними віджетами, такими як вікно чату, поля для питань і відповідей, а також кнопка для відправлення запитань. Такий підхід забезпечує зручність інтерфейсу для взаємодії з ChatGPT за допомогою Arduino, полегшують створення запитів та отримання відповідей зрозумілими для користувача. Вікно чату та самі віджети відображені на рисунку 1.3.

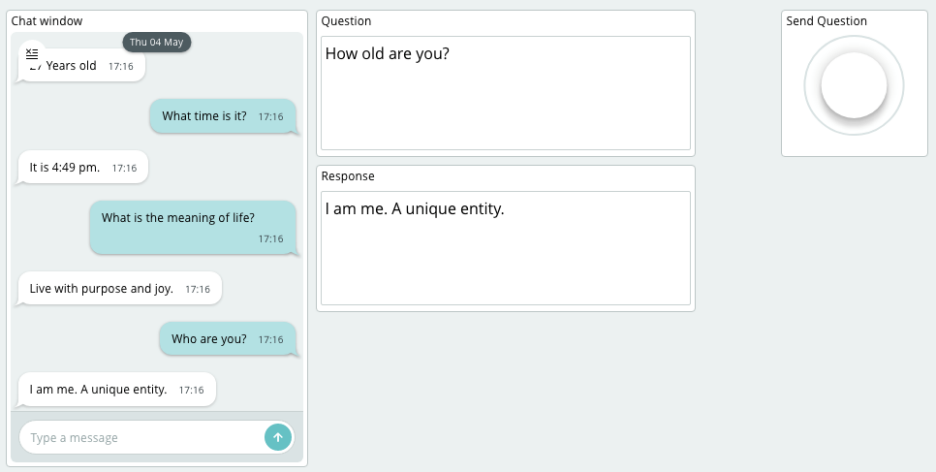



Рисунок 1.3 – Вікно чату та віджетів

Можливість взаємодії з ChatGPT у цьому проєкті реалізована через вікно чату та механізм запитів, та отримання відповідей у самому вікні чату. У випадку виникнення помилки, користувач отримає код помилки та його опис.

Також проєкт надає можливість налаштування різних параметрів, таких як максимальна кількість токенів у відповіді ChatGPT та використання різних мовних моделей OpenAI. Важливо слідкувати за витратами токенів, щоб уникнути непередбачуваного збільшення витрат. Токен – це одиниця вимірювання, яка використовується для підрахунку слів, символів або інших елементів у тексті.

Додатково, проєкт розширює можливості взаємодії з ChatGPT за допомогою голосу та планує розширення функціоналу для використання мовного асистента у щоденних справах.

Незважаючи на інноваційний підхід, проєкт має свої обмеження. Вибір апаратної платформи, зокрема Arduino Nano RP2040 Connect, може виявитися обмеженим щодо потужності та можливостей порівняно з іншими пристроями. Низький рівень обчислювальних можливостей може обмежити швидкість обробки запитів та відповідей від ChatGPT [6,7].

Також важливо відзначити, що взаємодія з ChatGPT через Arduino IoT Cloud може призвести до проблем щодо надійності та швидкості, залежності від хмарної інфраструктури.

OpenAI API використовує систему токенів для обмеження обсягу даних, які можна обробити. Важливо також врахувати, що проєкт не підтримує інші мови чи голосові команди, що обмежує його застосування в різних культурних та мовних середовищах.

Хоча проєкт є цікавим експериментом, його можливості обмежені вибором конкретних технологій та платформ. Вибір Arduino Nano RP2040 Connect як апаратної основи, хоч і виправданий для базової реалізації, може викликати певні обмеження у відношенні потужності та функціональності порівняно з більш продуктивними пристроями на ринку. Ця обмеженість може проявитися у обчислювальних можливостях та швидкості виконання завдань, зокрема у випадку обробки великої кількості токенів та складних запитань до ChatGPT [8-10].

### 1.2.3 Зв’язок з ChatGPT на ESP32

Наступний аналог який, представлений у відео "Running ChatGPT on ESP32: AI Conversations at the Edge Made Possible" демонструє, як запустити велику мовну модель ChatGPT на мікроконтролері ESP32, що відкриває нові можливості для реалізації AI-діалогів на пристроях з обмеженими ресурсами.

Функціонал дозволяє користувачам вводити текстові запити, які оброблюються ChatGPT, надаючи відповіді або інформацію на основі введених даних. Основний функціонал передбачає взаємодію користувача з ChatGPT через веб-сайт, а це вже є недоліком так як система немає ніякої автономності [11].

Отже, відсутність автономності та залежність від зовнішніх ресурсів робить цей проєкт менш надійним у ситуаціях, коли стабільність з'єднання або доступ до Інтернету є проблемою.

Зазначені переваги включають можливість розгортання ШІ на пристроях з обмеженими ресурсами, таких як автономні системи, розумні гаджети та IoT-пристрої. Це особливо актуально там, де доступ до Інтернету може бути обмеженим чи відсутнім. Зменшення затримок від обробки запитів на локальному

пристрої сприяє більш плавному та ефективному діалогу. Крім того, рішення є гнучким і легко інтегрованим в різні проєкти, дозволяючи розробникам максимально використовувати його у різних контекстах [12-15].

Щодо недоліків, важливо відзначити обмежені ресурси, які можуть впливати на складність та швидкість обробки запитів ChatGPT. На рисунку 1.4 представлено вікно взаємодії з ChatGPT, наводячи візуальний зразок роботи системи.

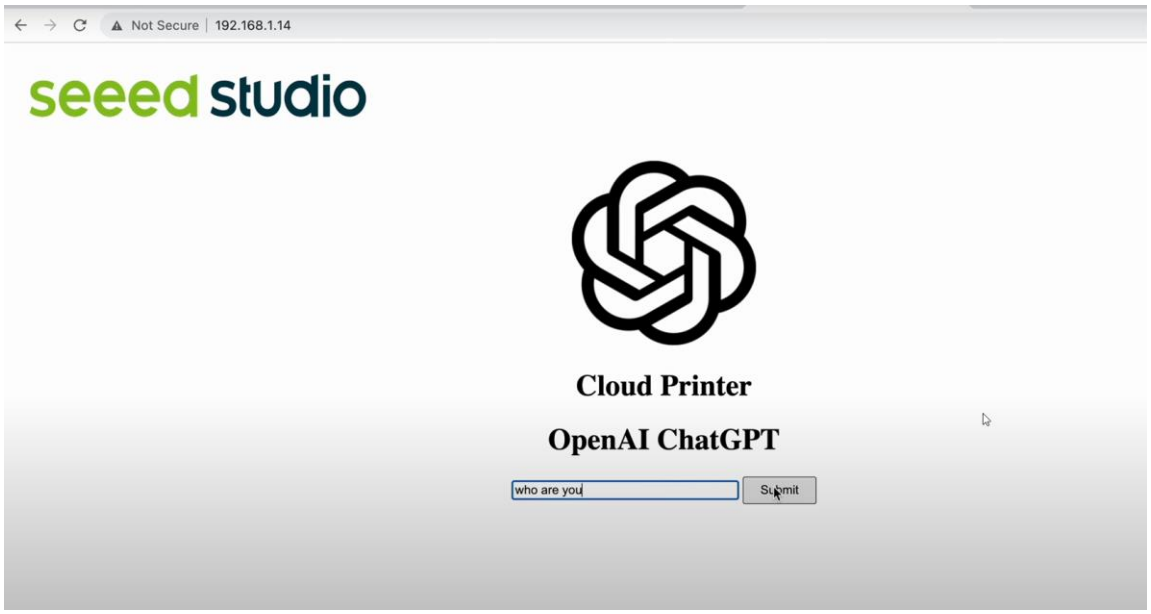



Рисунок 1.4 – Вікно взаємодії з ChatGPT

Мікроконтролер з вбудованим Wi-Fi та Bluetooth володіє достатньою потужністю для ефективного використання ChatGPT. Забезпечується доступ до функціональності ChatGPT за допомогою OpenAI API та використання бібліотеки ArduinoHTTPClient для спрощення HTTP-запитів. У рамках функціоналу користувач може вводити текстові запити, які оброблюються ChatGPT, а отримані відповіді виводяться в браузері. Рішення дозволяє локалізацію для різних мов та налаштування параметрів ChatGPT з урахуванням конкретних вимог використання.

Однак ці обмеження слід розглядати як природні атрибути процесу створення та налаштування системи, і не вказують на негативні якості чи характеристики ChatGPT. Запуск ChatGPT на ESP32 є перспективним напрямком для розробки AI-систем на пристроях із обмеженими ресурсами.

Проте на даному етапі необхідно акцентувати увагу на оптимізації коду для покращення продуктивності ESP32 та досліджувати нові методи взаємодії з ChatGPT, які краще відповідали б потребам користувачів. Також слід ретельно дослідити можливості використання даного рішення в різних галузях, таких як IoT [16-18].

У рамках проєкту використання ESP для ініціювання запитів до чату GPT призводить до певних недоліків. Особливо важливо відзначити, що реалізація цих запитів виконується через простий веб-браузер, що створює залежність від локальної конфігурації та можливостей комп'ютера користувача. З метою поліпшення стійкості системи та спрощення підтримки, рекомендується врахувати альтернативні методи реалізації запитів до чату GPT. Використання API та інших серверних рішень може відмінно вирішити вказані недоліки [19-23].

В результаті аналізу відео "Running ChatGPT on ESP32: AI Conversations at the Edge Made Possible" можна зробити висновок, що запуск великої мовної моделі ChatGPT на мікроконтролері ESP32 відкриває нові можливості для реалізації ШІ на пристроях з обмеженими ресурсами.

Переваги включають ефективне використання мікроконтролера з вбудованим Wi-Fi та Bluetooth, можливість локалізації та налаштування параметрів ChatGPT, що робить його гнучким для різних проєктів.

Розвиток AI-систем на пристроях із обмеженими ресурсами важливий в контексті автономних систем, розумних гаджетів та IoT-пристроїв, особливо в умовах обмеженого або відсутнього доступу до Інтернету.

Для подолання недоліків рекомендується оптимізація коду для поліпшення продуктивності ESP32 та розгляд альтернативних методів взаємодії з ChatGPT. Також важливо розглядати реалізацію запитів через API та інші серверні рішення для покращення стійкості системи та забезпечення надійності у різних умовах конфігурації користувачів. У таблиці 1.1 наведено характеристики порівняння власного проєкту з аналогами. Raspberry Pi 4B, USB аудіо-система, UPS батарея.

Таблиця 1.1 – Таблиця порівнянь

| Приклади робіт | |
|---|---|
| 1 | 2 |
| Власний проєкт | Програмно-технічний асистент для взаємодії із мовною моделлю на основі ESP32 |
| Аналог 1 | Raspberry Pi for Voice Interaction |
| Аналог 2 | ChatGPT through Arduino IoT Cloud |
| Аналог 3 | AI Conversations at the Edge |
| Основні компоненти | |
| Власний проєкт | ESP32, OLED дисплей, PS/2 клавіатура |
| Аналог 1 | Raspberry Pi 4B, USB аудіо-система, UPS батарея |
| Аналог 2 | Arduino Nano RP2040, Arduino IoT Cloud |
| Аналог 3 | ESP-32 |
| Система живлення | |
| Власний проєкт | Залежить від живлення ESP32 |
| Аналог 1 | UPS батарея |
| Аналог 2 | Залежить від живлення Arduino Nano RP2040 |
| Аналог 3 | Залежить від живлення ESP32 |
| Голосовий ввід/вивід | |
| Власний проєкт | Відсутній |
| Аналог 1 | USB аудіо-система |
| Аналог 2 | Відсутній |
| Аналог 3 | Відсутній |
| Взаємодія з ChatGPT | |
| Власний проєкт | Ввід запитів з клавіатури, та вивід результату на OLED |
| Аналог 1 | Через голосовий інтерфейс |
| Аналог 2 | Через вікно чату |
| Аналог 3 | Через додатковий вебсайт в браузері |

Кінець таблиці 1.1

| Локалізація | |
|---|---|
| Власний проєкт | Локалізація можлива для різних мов, але за основу обрана Англійська |
| Аналог 1 | Англійська |
| Аналог 2 | Залежить від реалізації у IoT Cloud |
| Аналог 3 | Англійська |
| Застосування | |
| 1 | 2 |
| Власний проєкт | Взаємодія з ChatGPT через клавіатуру та OLED дисплей |
| Аналог 1 | Голосовий інтерфейс для ChatGPT |
| Аналог 2 | Взаємодія з ChatGPT через Arduino IoT Cloud |
| Аналог 3 | Запуск ChatGPT на пристроях з обмеженими ресурсами |
| Переваги | |
| Власний проєкт | Простий інтерфейс, зручність у введенні та виведенні |
| Аналог 1 | Можливість голосового вводу, виводу та взаємодії |
| Аналог 2 | Розгортання AI на пристроях з обмеженими ресурсами |
| Аналог 3 | Зниження затримок, підвищення конфіденційності, гнучкість |
| Недоліки | |
| Власний проєкт | Обмежені можливості голосового вводу, виводу |
| Аналог 1 | Обмежені можливості ручного вводу даних, та можливості читання їх. |
| Аналог 2 | Обмежені ресурси ESP32, обмеження безкоштовного API |
| Аналог 3 | Обмежені ресурси ESP32, оптимізація коду, необхідність розробки нових методів |

Проєкт, над яким буде проведено роботу, вирізняється своєю легкою та інтуїтивно зрозумілою системою взаємодії, особливо завдяки використанню клавіатури та OLED дисплею. Це не тільки полегшує доступ до ChatGPT, а й робить цей процес надзвичайно зручним для будь-якого користувача. Великою перевагою є можливість автономної роботи проєкту, усунувши необхідність відкривати вікно браузера на основному комп'ютері та використовуючи ESP32 як центральний мікроконтролер. Порівнюючи із конкурентами, наприклад, аналог 1 пропонує використання голосового вводу та виводу, а аналог 2 спрямований на впровадження ШІ на пристроях з обмеженими ресурсами. Однак у власного проєкту основний акцент робиться на легкості та зручності в користуванні, дозволяючи взаємодіяти з AI-асистентом просто та ефективно, вирішуючи проблему залежності від зовнішніх пристроїв [24-30].

Навіть при наявності обмежень у голосовому вводі та виводі, власний проєкт компенсує це ефективністю та простотою у використанні інтерфейсу. Важливо відзначити, що проєкт є автономною системою, що дає можливість йому працювати незалежно, без залежності від інших пристроїв чи комп'ютерів, що забезпечує йому велику гнучкість та мобільність.

## 1.3 Висновки. Постановка задачі

Під час проведення аналізу аналогів у галузі ШІ, здійсненого у рамках даного дослідження, були ретельно проаналізовані актуальні проєкти, спрямовані на вдосконалення взаємодії між користувачами та системами ШІ. В результаті цього аналізу виникла конкретна мета - створення програмно-технічного асистента, який здатний взаємодіяти з користувачем через клавіатуру та відображати результати на OLED дисплеї, використовуючи мікроконтролер ESP32.

Необхідність у такому інструменті виникає з метою забезпечення простоти та ефективності взаємодії для різних категорій користувачів, які віддають перевагу використанню клавіатури для комунікації з програмно-технічними асистентами.

Цей проєкт має значний потенціал у полегшенні щоденних завдань та підвищенні доступності ШІ для широкого кола користувачів.

Отримані результати дослідження можуть внести суттєвий внесок у подальший розвиток та впровадження ШІ в різноманітні галузі, сприяючи при цьому спрощенню та підвищенню ефективності його використання. Цей проєкт виступає важливим кроком у напрямку створення інноваційних технологій, спрямованих на поліпшення якості життя та роблять взаємодію з штучним інтелектом більш доступною для всіх зацікавлених сторін.

Метою наступного розділу є вибір та обґрунтування апаратних компонентів для реалізації програмно-технічного асистента для взаємодії з мовною моделлю на базі мікроконтролера ESP32.

Вибір мікроконтролера, який буде служити основою для системи та забезпечить необхідну продуктивність, функціональність та можливості підключення периферійних пристроїв.

Визначення та вибір периферійних пристроїв, таких як дисплей для відображення інформації, аудіопідсилювач для відтворення звуку та можливості введення даних від користувача (клавіатура або інші засоби введення).

Опис процесу підключення обраних компонентів до мікроконтролера ESP32, включаючи схеми підключення та розподіл контактів.

Огляд бібліотек, які будуть використані для програмування та взаємодії з обраними компонентами, а також їх призначення та функціональність.

Вирішення поставлених завдань дозволить сформувати апаратну базу для створення програмно-технічного асистента, який зможе взаємодіяти з мовною моделлю, відображати інформацію, відтворювати звук та приймати команди від користувача. Це забезпечить основу для подальшої програмної реалізації та інтеграції системи з мовними моделями та сервісами ШІ.

# 2 СТРУКТУРНІ ЧАСТИНИ ПРИСТРОЮ

## 2.1 Вибір мікроконтролера

Мікроконтро́лер (англ. microcontroller), або однокристальна мікроЕОМ — виконана у вигляді мікросхеми спеціалізована мікропроцесорна система, що включає мікропроцесор, блоки пам'яті для збереження коду програм і даних, порти вводу-виводу і блоки зі спеціальними функціями (лічильники, компаратори, АЦП та інші) [31].

Наразі технології, що використовують мікроконтролери та ШІ, широко використовуються в різних сферах нашого повсякденного життя. Вони застосовуються для систем контролю, моніторингу, захисту, передавання та обробки даних. Розглянемо кілька найпопулярніших мікроконтролерів та їх характеристики.

Arduino Uno - це один із найпопулярніших мікроконтролерів, оснащений процесором ATmega328P та пам'яттю Flash об'ємом 32 КБ. Цей мікроконтролер має 14 GPIO пінів та вбудований USB-інтерфейс.

Raspberry Pi - це міні-комп'ютер, який має більш потужний процесор ARM Cortex-A та може працювати на частоті до 1.2 ГГц. Він має різні моделі з різним обсягом оперативної пам'яті, від 512 МБ до 8 ГБ.

ESP8266 - це мікроконтролер, розроблений для бездротового зв'язку. Він має процесор Tensilica L106, вбудований Wi-Fi та може працювати на частоті 80 МГц.

ESP32 - це продовження ESP8266, відоме своїм двоядерним процесором Tensilica LX6, підтримкою Wi-Fi та Bluetooth, а також більшою кількістю GPIO пінів.

STM32 - це родина мікроконтролерів, яка працює на базі ядра ARM Cortex-M. Вони мають різні моделі з різною швидкістю процесора та обсягом пам'яті.

Одним з основних конкурентів ESP32 є Arduino Uno. Хоча Arduino Uno має свої переваги, такі як простота використання та велика спільнота користувачів, ESP32 виграє завдяки своїм розширеним можливостям, таким як вбудований Wi-Fi

та Bluetooth та двоядерний процесор, що дозволяє виконувати більше завдань одночасно. В порівнянні з Raspberry Pi, ESP32 має менше споживання енергії та може працювати на батареї довше, що робить його ідеальним вибором для мобільних пристроїв та проєктів, які потребують низької потужності [32-36].

ESP32 також конкурує зі своїм попередником, ESP8266. Хоча ESP8266 також має вбудований Wi-Fi, ESP32 виграє завдяки своєму двоядерному процесору, підтримці Bluetooth та більшій кількості GPIO контактів.

Порівнюючи ESP32 з STM32, ми бачимо, що STM32 може мати більше GPIO пінів та розширені можливості прямого доступу до пам'яті. Однак ESP32 виграє завдяки своєму вбудованому Wi-Fi та Bluetooth, що робить його більш зручним для проєктів IoT та бездротового зв'язку.

Кожен з цих мікроконтролерів має свої особливості та характеристики, що роблять їх ідеальними для різних типів проєктів.

ESP32 є одним із найбільш потужних та розширених мікроконтролерів, які доступні на ринку сьогодні. Цей мікроконтролер здатний задовольнити потреби проєктів високої складності завдяки своїм унікальним особливостям та характеристикам.

Однією з ключових переваг ESP32 є його двоядерний процесор Tensilica LX6, який дозволяє виконувати більше завдань одночасно та забезпечує високу продуктивність. Це особливо корисно для проєктів, які потребують швидкої обробки даних чи багатозадачності.

Додатково, ESP32 має значну кількість GPIO контактів, що дозволяє підключати до нього різноманітні сенсори, пристрої введення-виведення та інші компоненти. Це робить його універсальним рішенням для багатьох типів проєктів, від датчиків температури до систем автоматизації будинку. Назва "ESP32" розшифровується як "Espressif System Platform 32", розробленої компанією Espressif Systems. Даний мікроконтролер наведений на рисунку 2.1.

Рисунок 2.1 – ESP-WROOM32

Крім того, ESP32 має вбудований Wi-Fi та Bluetooth, що робить його ідеальним вибором для проєктів Інтернету речей (IoT) та зв'язку з іншими пристроями. Ця можливість дозволяє створювати бездротові системи зв'язку та керування, що розширює можливості ваших проєктів. У таблиці 2.1 наведені основні технічні характеристики ESP32-WROOM32 [37-40].

Таблиця 2.1 – Технічні характеристики ESP32-WROOM32

| Характеристики мікроконтролера | Значення |
|---|---|
| 1 | 2 |
| Мікроконтролер | ESP32 WROOM |
| Ядро | Двоядерний процесор Tensilica LX6 |
| Частота | Від 80 МГц до 240 МГц |
| Пам'ять | Вбудована Flash пам'ять до 4 МБ |
| Wi-Fi | 802.11 b/g/n (Wi-Fi) 2.4 GHz, до 150 Mbps |
| Bluetooth | Bluetooth 4.2 та Bluetooth Low Energy (BLE) |
| GPIO | До 36 GPIO піни для зовнішніх підключень |
| Аналогові входи | До 18-ти аналогових входів |
| Робоча напруга | 3.3 В |
| Підтримка ОС | FreeRTOS, MicroPython, Arduino IDE |

З урахуванням всіх цих функцій та особливостей, ESP32 стає очевидним вибором для багатьох проєктів, особливо тих, що потребують бездротового зв'язку, багатозадачності та розширеної обробки даних.

Перейдемо до порівняння ESP32 з іншими мікроконтролерами, щоб показати його переваги більш детально.

Загалом, ESP32 виявляється виграшним в багатьох аспектах порівняно з іншими мікроконтролерами, особливо для проєктів, які потребують бездротового зв'язку, багатозадачності та енергоефективності.

Тепер давайте перейдемо до розгляду конкретних особливостей ESP32, що роблять його таким привабливим для різних видів проєктів.

Розглянувши порівняння з іншими мікроконтролерами, виділимо особливості ESP32, які роблять його доцільним у використані для різних типів проєктів.

ESP32 оснащений двоядерним процесором Tensilica LX6, що робить його ефективним у виконанні багатьох завдань одночасно. Це особливо корисно для проєктів, які вимагають багатозадачності та обробки даних у реальному часі.

Додатково, ESP32 підтримує різні операційну систему FreeRTOS, MicroPython компілятор та середовище розробки програмного забезпечення Arduino IDE, що надає розробникам більшу гнучкість та зручність.

Оскільки ESP32 відомий своїм низьким споживанням енергії, що робить його ідеальним для проєктів, які потребують роботи на батареї. Це робить ефективним вибором для мобільних пристроїв та інших пристроїв з обмеженим джерелом живлення [41-43].

З урахуванням цих особливостей, ESP32 є очевидним вибором для багатьох типів проєктів, особливо для тих, що потребують бездротового зв'язку, багатозадачності та енергоефективності. Розподіл контактів складається з 30 контактів. На рисунку 2.2 наведено схему розподілу контактів ESP-WROOM32.

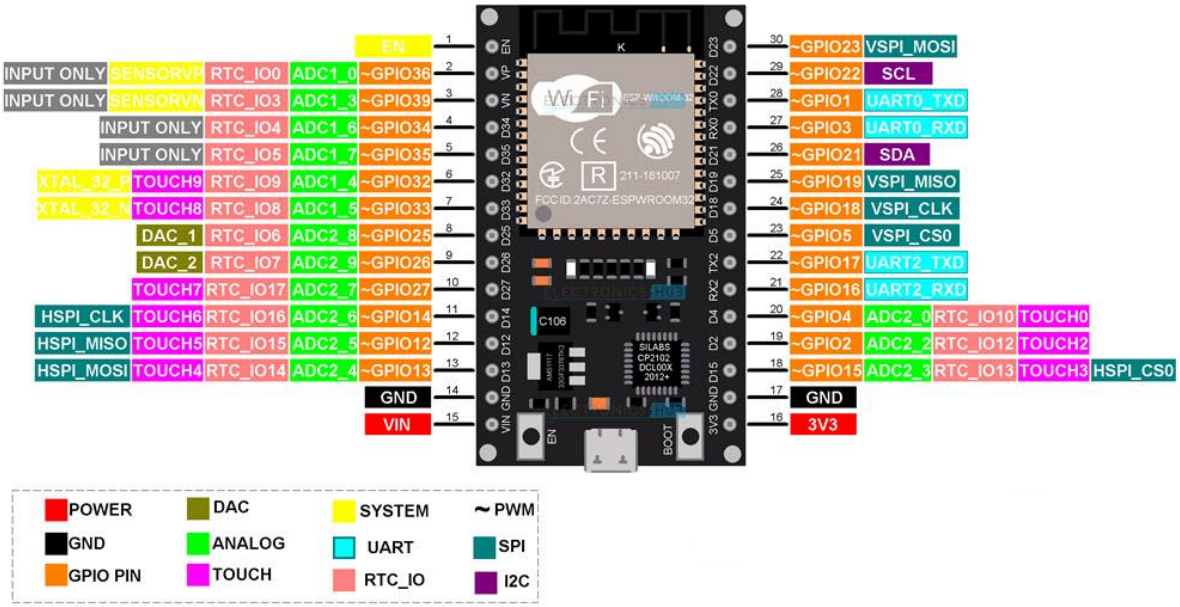



Рисунок 2.2 – Схема розподілу контактів ESP-WROOM32

Як видно з зображення, кожен контакт має більше однієї можливої функції, розглянемо наявні контакти даного мікроконтролера. У таблиці 2.2 приведено наявні периферійні інтерфейси.

Таблиця 2.2 – Контакти вводу-виводу даних

| Периферійні інтерфейси | Кількість |
|---|---|
| 1 | 2 |
| Програмовані GPIOs | 34 |
| 12-бітні канали ADC | 18 |
| 8-бітні канали DAC | 2 |
| Канали PWM | 16 |
| Інтерфейси UART | 3 |
| Інтерфейси SPI | 3 |
| Інтерфейси I2C | 2 |
| Інтерфейси I2S | 2 |
| CTS GPIOs | 10 |
| RTC GPIOs | 16 |

Програмовані GPIOs: ESP32 має 34 програмовані GPIO, що дозволяють вам налаштовувати їх для виконання різних функцій в програмі. Кожен GPIO може виконувати більше однієї функції, проте тільки одна з них може бути активною в певний момент часу.

12-бітні канали ADC: ESP32 має 18 каналів ADC з роздільною здатністю 12 біт, що дозволяє зчитувати аналогові значення з досягненням високої точності у широкому діапазоні.

8-бітні канали DAC: цей мікроконтролер має два незалежних канали DAC з роздільною здатністю 8 біт, що дозволяють конвертувати цифрові значення в аналогові сигнали з точністю і контролем.

Канали PWM: мікроконтролер має 16 незалежних каналів PWM з налаштовуваною частотою і відсотком заповнення, що дозволяє створювати ШІМ сигнали для керування різними пристроями, такими як мотори та світлодіоди.

Інтерфейси UART: цей мікроконтролер має три UART інтерфейси, що дозволяють здійснювати передачу та отримання даних з іншими пристроями через серійний інтерфейс.

Інтерфейси SPI: ESP32 має три SPI інтерфейси, які можуть працювати як майстер або раб у режимі спілкування, що дозволяє забезпечувати зв'язок зі зовнішніми пристроями, такими як флеш-пам'ять.

Інтерфейси I2C: мікроконтролер має два I2C інтерфейси з повною гнучкістю в призначенні контактів, що дозволяє забезпечувати зв'язок з різними пристроями, такими як сенсори та інші периферійні пристрої.

Інтерфейси I2S: ESP32 має два I2S інтерфейси для звукового введення та виведення, що дозволяє підключати аудіопристрої та обробляти аудіосигнали.

CTS GPIOs: ESP32 налічує 10 CTS GPIOs з можливістю ємнісного дотику, які можуть використовуватися для виявлення дотику або наближення пальця до певного контакту, що дозволяє реалізувати ємнісні тачпади без додаткового обладнання.

RTC GPIOs: ESP32 має 16 RTC GPIO, які можуть використовуватися для управління глибоким сном та пробудження мікроконтролера з низького споживання енергії.

Даний проєкт передбачає створення програмно-технічного асистента для взаємодії з мовною моделю, яка потребує бездротового зв'язку та багатозадачності. ESP32 ідеально підходить для цієї задачі, оскільки він має вбудовані модулі Wi-Fi та Bluetooth, що дозволяють легко забезпечити зв'язок з іншими пристроями та серверами.

З урахуванням всіх наведених вище характеристик, ESP32 є найбільш ефективним вибором для даного проєкту. Його здатність до бездротового зв'язку, багатозадачності та енергоефективності робить його ідеальним рішенням для реалізації наших задач [44-49].

## 2.2 Вибір периферійних пристроїв

Дисплей (від англ. display — «показувати») — електронний засіб для відтворювання графічної й текстової інформації. Дисплеї комп'ютерів зазвичай називають моніторами. Варіантів дисплеїв є декілька, але основні ми роглянемо далі.

LCD дисплеї: LCD дисплеї широко використовуються в електроніці та мобільних пристроях. Вони мають яскраве підсвічування та високу роздільну здатність, але вони вимагають більше енергії і не так прості у використанні з мікроконтролерами.

OLED дисплеї: OLED дисплеї володіють високою контрастністю та яскравістю, а також можуть бути дуже енергоефективними. Вони підходять для проєктів з обмеженим джерелом живлення.

E-ink дисплеї: E-ink дисплеї використовуються для відображення тексту та зображень зниженого розділення, але вони мають дуже низьке споживання енергії

та можуть залишатися включеними протягом тривалого часу без додаткового живлення.

TFT дисплеї: TFT дисплеї забезпечують яскраве та насичене зображення, але вони можуть бути більш вимогливими до енергії і складнішими у використанні з мікроконтролерами.

З наданих типів дисплеї було прийнято рішення зупинитись на OLED SSD1306. Модуль OLED дисплея з діагоналлю 0.96" і роздільною здатністю 128x64 з керуванням SSD1306 - це компактний та потужний засіб візуалізації для різних електронних пристроїв. OLED (Organic Light Emitting Diode) технологія надає високу якість зображення і відмінну контрастність порівняно з іншими типами дисплеїв. Використання такого дисплею дозволяє створювати чіткі та яскраві візуальні інтерфейси для користувачів електронних пристроїв, що є надзвичайно важливим у сучасному світі технологій. Характеристики дисплею відображені у таблиці 2.3.

Таблиця 2.3 – Характеристики OLED Display SSD1306

| Характеристики мікроконтролера | Значення |
|---|---|
| 1 | 2 |
| Тип дисплея | OLED |
| Діагональ | 0.96" |
| Роздільна здатність | 128x64 |
| Керувальний чіпсет | SSD1306 |
| Колір відображення | Білий |
| Інтерфейс | I2C, SPI |
| Робоча напруга | 3.3 В |
| Контрастність | Висока |
| Швидкість оновлення | Швидка |

OLED SD1306 забезпечує простоту підключення та керування дисплеєм через інтерфейси I2C або SPI, що робить його ідеальним вибором для широкого спектру мікроконтролерних пристроїв. Крім того, завдяки своїй компактності, цей дисплей може бути легко інтегрований в різноманітні проєкти з обмеженим простором.

Узагальнюючи, OLED Display SSD1306 є ефективним, компактним та легким у використанні рішенням для візуалізації даних у різних проєктах, надаючи високу якість зображення та велику гнучкість у використанні.

Модуль аудіопідсилювача - це пристрій, який використовується для підсилення аудіосигналу з метою створення вихідного звуку на акустичних системах або інших аудіо пристроях. Він може бути вбудованим в аудіоапаратуру або працювати як окремий компонент [50-52].

Модуль аудіопідсилювача приймає вхідний аудіосигнал з джерела, такого як музичний програвач або мікрофон, та збільшує його амплітуду за допомогою підсилювача. Це дозволяє створити потужний вихідний сигнал, який може бути поданий на акустичні системи для відтворення звуку.

Модулі можуть мати різні характеристики, такі як потужність, опір, спотворення та інші, які впливають на їхню ефективність і якість звуку. Вони широко використовуються в домашній аудіо, професійних аудіосистемах, а також у автомобільному аудіо.

Регулювання рівня гучності, тембри, еквалайзер та інші параметри звуку зазвичай здійснюється за допомогою регуляторів, вбудованих у модуль або підключених до нього.

При виборі модуля аудіопідсилювача для будь-якого проєкту важливо враховувати ряд факторів, що включають у себе якість звуку, сумісність з іншими компонентами, легкість використання та можливості налаштування. У цьому розділі ми розглянемо процес вибору аудіопідсилювача та аргументи, що підтримують вибір модуля MAX98357 I2S Amplifier.

Почнемо з аналізу потреб проєкту та вимог до аудіосистеми. Визначимо, які саме функції має виконувати аудіопідсилювач у контексті нашого проєкту, а також які технічні характеристики є ключовими для успішної реалізації завдання.

MAX98357 I2S Amplifier - це інтегрований аудіопідсилювач, розроблений компанією Maxim Integrated, яка відома своєю високоякісною продукцією та надійністю. Цей модуль відзначається своєю простотою використання та рядом переваг. На рисунку 2.3 відображений модуль MAX98357 I2S Amplifier.

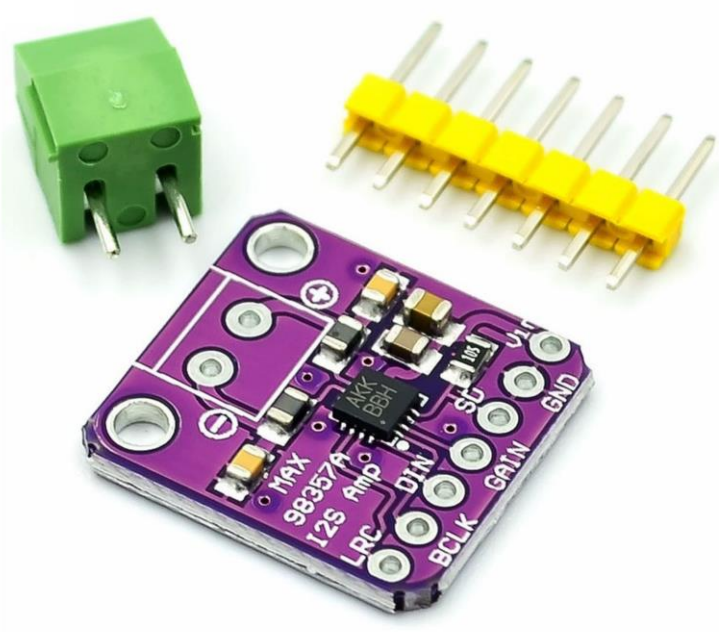



Рисунок 2.3 – Аудіопідсилювач MAX98357 I2S Amplifier

Однією з ключових особливостей MAX98357 є його підтримка інтерфейсу I2S. Це означає, що він може безпосередньо працювати з цифровим аудіосигналом, що надає перевагу в якості звуку та зменшує шумове спотворення порівняно з аналоговими рішеннями. Інтерфейс I2S дозволяє передавати аудіодані разом з клоковим сигналом та сигналом управління, що спрощує процес комунікації між аудіопідсилювачем та іншими пристроями, такими як мікроконтролери або DSP.

Також MAX98357 I2S Amplifier, існують інші аудіопідсилювачі, які можуть бути варіантами для розгляду.

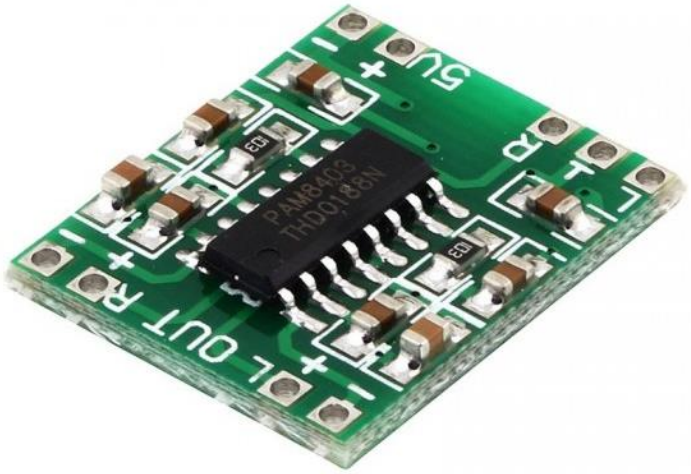



Рисунок 2.4 – Аудіопідсилювач PAM8403

PAM8403 є невеликим та економічним цифровим потужним аудіопідсилювачем. Він часто використовується у простих аудіосистемах, де важлива ефективність використання простору та низька вартість. Зазвичай цей модуль має обмежений функціонал, але його невеликі розміри та низька вартість роблять його привабливим вибором для проєктів з обмеженими бюджетами або простими вимогами до аудіосистеми. Однак, якість звуку та можливості налаштування PAM8403 можуть бути обмеженими у порівнянні з більш продвинутими модулями, такими як MAX98357.

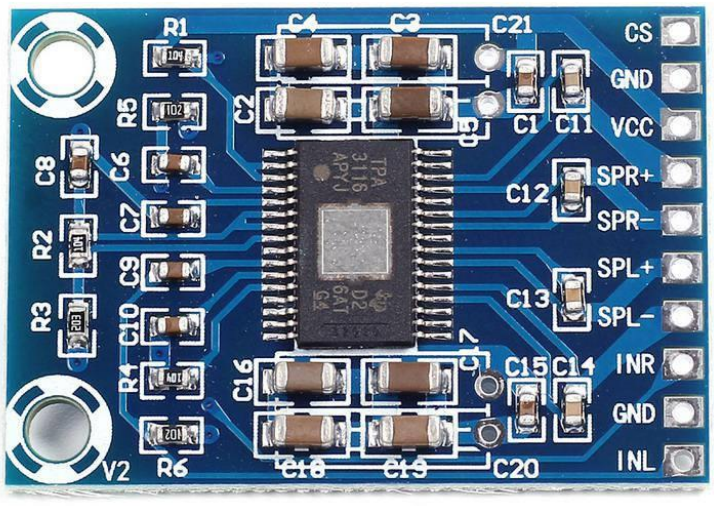



Рисунок 2.5 – Аудіопідсилювач TPA3116D2

TPA3116D2 - це аудіопідсилювач класу D з високою потужністю та хорошою якістю звуку. Він часто використовується в пристроях, де важлива велика потужність аудіосигналу, таких як домашні кінотеатри або професійні аудіосистеми. Однак, цей аудіопідсилювач може вимагати більше простору для встановлення через свої розміри та потребу в додаткових компонентах для оптимальної роботи [53-56].

Одна з ключових переваг MAX98357 полягає в широкому діапазоні вхідних напруг, який становить від 2,5 до 5,5 В. Це робить його досить гнучким у використанні, оскільки він може легко інтегруватися з різними джерелами живлення. Крім того, його висока якість звуку і низький рівень спотворення роблять його ідеальним вибором для проєктів, де важлива висока якість аудіо. Ці характеристики дозволяють максимально використовувати потенціал звукових даних без втрати якості. Більш детальні технічні специфікації можна знайти в таблиці 2.4.

Таблиця 2.4 – Характеристики MAX98357 I2S Amplifier

| Показник | Значення |
|---|---|
| 1 | 2 |
| Вихідна потужність | До 3.2 Вт на 4 Ом, до 1.8 Вт на 8 Ом |
| Діапазон напруг входу | 2.5V - 5.5V |
| Діапазон напруг виходу | 2.6V - VDD |
| Тип підсилювача | Цифровий (Class D) |
| Інтерфейс | I2S (Inter-IC Sound) |
| Споживана потужність | 3.2 мВт в режимі очікування (standby) |
| Коефіцієнт підсилення | 3.5, 9, 15, 23, 30, 38, 47, 60, 75, 90 dB |
| Інші особливості | Вбудований DC затискавший конденсатор, захист від перевантаження та короткого замикання |

Можна зазначити, що MAX98357 I2S Amplifier вирізняється своєю високою якістю звуку, легкістю використання та підтримкою інтерфейсу I2S, що робить його привабливим вибором для багатьох проєктів в яких необхідне звукове оформлення, особливо в проєктах, де потрібно забезпечити якісне аудіо відтворення з мінімальними зусиллями щодо налаштування і підключення. Його сумісність з інтерфейсом I2S спрощує інтеграцію з різноманітними пристроями та системи, забезпечуючи стабільну та ефективну роботу.

## 2.3 Опис підключення

У світі постійного технологічного прогресу та інновацій стає все більш важливим розробка інтелектуальних систем, спрямованих на полегшення нашого повсякденного життя. Одним із таких інноваційних проєктів є створення програмно-технічного асистента для взаємодії із мовною моделлю на основі ESP32, що включає в себе OLED дисплей, клавіатуру та аудіопідсилювач. Ця робота націлена на створення зручного та ефективного інтерфейсу, що дозволить користувачеві взаємодіяти з пристроєм у найбільш зручний та інтуїтивно зрозумілий спосіб.

Початковий етап розробки включає в себе підключення ESP32 до OLED дисплею, клавіатури та аудіопідсилювача та динаміка. Цей процес є ключовим для подальшої роботи над програмною складовою проєкту. Підключення та налаштування всіх компонентів дозволить створити стабільну та функціональну апаратну базу для реалізації програмних можливостей асистента [57].

Попередні дослідження та аналіз сучасних технологій дозволяють зрозуміти значення подальшої розробки подібних систем у сфері інформаційних технологій. Інноваційні проєкти, такі як цей, мають потенціал значно покращити якість життя та забезпечити нові можливості для спілкування та взаємодії з технологіями.

Для підключення ESP32 до OLED SSD1306 дисплею, спочатку необхідно правильно з’єднати контакти: SDA дисплею повинен бути підключений до контакту GPIO D4 ESP32, а SCL дисплею - до контакту GPIO D5. Також, для

належної роботи, необхідно з'єднати контакт GND дисплею з контактом ESP32, щоб забезпечити належне заземлення. Щодо живлення VCC дисплею, його потрібно підключити до джерела живлення 3.3 вольт на ESP32.

Цей крок є критичним для успішної роботи OLED дисплею з ESP32. Після завершення цього етапу підключень можна переходити до наступних кроків у створенні програмно-технічного асистента для взаємодії із мовною моделлю на базі ESP32.

В рамках даного проєкту, для підключення клавіатури необхідно використати USB, зокрема USB 2.0, до ESP32. Для цього проведемо наступні кроки: спочатку з'єднаємо D+ USB з контактом GPIO D23 на ESP32, а D- USB - з контактом GPIO D21 на ESP32. Далі, підключимо GND USB до заземлення GND на ESP32, а VCC USB - до живлення 3.3 вольт на ESP32 [58].

Наступним етапом є підключення аудіопідсилювача MAX98357 I2S Amplifier до ESP32 для відтворення звукових відповідей асистента. Для цього необхідно виконати наступні кроки: спочатку підключити контакт LRC аудіопідсилювача до контакту D26 на ESP32, BCLK - до контакту D27, а контакт DIN - до контакту D25 на ESP32. Далі, з'єднати контакти GAIN та GND аудіопідсилювача з заземленням на ESP32, а контакт VCC - з живленням 3.3 вольт на ESP32.

Наступне підключення забезпечить можливість відтворення звукових відповідей асистента через аудіопідсилювач.

Додатково необхідно підключити динамік до аудіопідсилювача. Для цього з'єднаємо плюсовий (+) полюс динаміка з плюсовим (+) виходом аудіопідсилювача, а мінусовий (-) полюс динаміка з мінусовим (-) виходом аудіопідсилювача.

Завдяки правильно виконаному підключенню та налаштуванню, ваш асистент на базі ESP32 та MAX98357 I2S Amplifier буде здатен надавати якісні звукові відповіді, роблячи взаємодію більш природною та ефективною. Підключення відображені на таблиці 2.5.

Таблиця 2.5 – Підключення периферії з мікроконтролером ESP32

| Контакти ESP32 | Номера контактів периферії | Периферія | Призначення |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| D4 | SDA | OLED Display | Передача даних |
| D5 | SCL | OLED Display | Зв'язок |
| D23 | D+ | USB слот | Передача даних |
| D21 | D- | USB слот | Зв'язок |
| D26 | LRC | Аудіопідсилювач | Сигнал LRC |
| D27 | BCLK | Аудіопідсилювач | Сигнал BCLK |
| D25 | DIN | Аудіопідсилювач | Сигнал DIN |
| GND | - | Усі підключені пристрої | Заземлення |
| 3.3V | VCC | Усі підключені пристрої | Живлення |
| GAIN | - | Аудіопідсилювач | Контроль гучності |
| - | + | Динамік | Передача сигналів |
| - | - | Динамік | Заземлення |

На таблиці представлено підключення периферії до ESP32, включаючи OLED дисплей, USB слот, аудіопідсилювач та динамік. Кожен контакт ESP32 має відповідну функцію та зв'язок з периферійними пристроями для передачі даних, зв'язку чи живлення.

## 2.4 Використані бібліотеки

Програмно-технічний асистент для взаємодії із мовною моделлю на основі ESP32 представляє собою інноваційний пристрій, спрямований на полегшення взаємодії користувача з технічними пристроями за допомогою мовних команд. Інтеграція різноманітних бібліотек дозволяє не лише швидко реалізовувати нові ідеї та функції, але й забезпечує гнучкість системи, що дозволяє легко адаптуватися до змін у вимогах користувачів та ринкових умов.

Такий підхід до розробки програмно-технічного асистента на базі ESP32 створює основу для стабільного та успішного функціонування пристрою в умовах швидко змінюючогося технологічного середовища.

Використання бібліотек для забезпечення функціональності пристрою є важливою складовою розробки програмно-технічного асистента на базі ESP32. Бібліотеки надають зручний і швидкий спосіб доступу до необхідних функцій та можливостей мікроконтролеру, а також реалізацію специфічних функцій, таких як робота з аудіоданими, графічним дисплеєм, мережевими з'єднаннями тощо.

У розробці асистента на базі ESP32 вибір правильних бібліотек відіграє важливу роль. Бібліотеки - це набори функцій та методів, які допомагають спростити розробку програмного забезпечення, швидко втілити функціональність та зменшити кількість коду, що потрібно написати вручну.

Однією з основних бібліотек, що використовується у цьому проєкті, є Arduino.h. Ця бібліотека забезпечує роботу з мікроконтролером ESP32 та дозволяє взаємодіяти з його функціями та можливостями.

У разі потреби у відтворенні аудіоданих, використовується бібліотека Audio.h. Це дозволяє програмно-технічному асистентові відтворювати звукові сигнали або навіть працювати з голосовими командами користувача. Такий функціонал додає пристрою більшу інтерактивність та зручність у використанні.

Для відображення інформації на графічному OLED дисплеї використовується бібліотека U8g2lib.h. Ця бібліотека дозволяє легко керувати відображенням тексту та графічних об'єктів на дисплеї, що додає пристрою зручність у взаємодії з користувачем.

Однією з найважливіших можливостей сучасних пристроїв є можливість підключення до Інтернету. Для цього використовується бібліотека WiFi.h, яка дозволяє ESP32 створювати мережеві з'єднання та взаємодіяти з віддаленими серверами.

Забезпечення безпеки мережевого з'єднання важливо для збереження конфіденційності даних користувача. Тут на допомогу приходить бібліотека WiFiClientSecure.h, яка забезпечує безпечне з'єднання з сервером за допомогою шифрування та аутентифікації.

Для обробки та форматування даних у форматі JSON використовується

бібліотека ArduinoJson.h. Це особливо корисно в сучасних додатках, де взаємодія з API часто здійснюється через обмін даними у форматі JSON.

Для обробки введення з клавіатури та взаємодії з детальною інформацією про клавіатурний ввід використовуються бібліотеки PS2KeyAdvanced.h та PS2KeyMap.h відповідно. Це дозволяє асистентові взаємодіяти з клавіатурою та обробляти натискання клавіш для виконання певних дій. На таблиці 2.6 приведено основні бібліотеки, які було використано під час розробки [59,60].

Таблиця 2.6 – Використані бібліотеки

| Назва бібліотеки | Призначення |
|---|---|
| 1 | 2 |
| Arduino.h | Робота з мікроконтролером ESP32 |
| Audio.h | Робота з аудіоданими |
| U8g2lib.h | Робота з OLED дисплеєм |
| WiFi.h | Підключення ESP32 до Інтернету |
| WiFiClientSecure.h | Забезпечення безпечного з'єднання з сервером |
| ArduinoJson.h | Обробка та форматування даних у форматі JSON |
| PS2KeyAdvanced.h | Обробка клавіатурного введення з клавіатури PS/2 |
| PS2KeyMap.h | Обробка детальної інформації про клавіатурний ввід |

Переваги використання бібліотек у порівнянні з самостійною реалізацією функцій очевидні. Вони включають в себе прискорення процесу розробки, зменшення кількості помилок, підвищення надійності програмного забезпечення та полегшення внесення змін у майбутньому.

Бібліотека Arduino.h є однією з найважливіших бібліотек для роботи з мікроконтролером ESP32. Вона надає доступ до базових функцій та методів, необхідних для програмування. Основні функції та методи, які надаються бібліотекою Arduino.h, включають в себе:

– pinMode(pin, mode): метод встановлює режим вводу/виводу для певного

контакту (вхід, вихід);

- digitalRead(pin): метод читає стан цифрового контакту (HIGH або LOW);
- digitalWrite(pin, value): метод встановлює стан цифрового контакту (HIGH або LOW);
- analogRead(pin): метод зчитує аналогове значення з певного контакту;
- analogWrite(pin, value): метод встановлює ширину імпульсу ШИМ на певному контакті (для генерації аналогових сигналів);
- delay(ms): функція затримки, що призначена для паузи в програмі на певну кількість мілісекунд;
- Serial.begin(baudRate): метод ініціалізації порту для послідовного зв'язку з певною швидкістю передачі даних.

Ці функції і методи дозволяють з легкістю взаємодіяти з різними аспектами ESP32, такими як введення/виведення, аналогові та цифрові сигнали. Бібліотека Audio.h надає інструменти для роботи з аудіоданими на мікроконтролері ESP32.

Дана бібліотека була використана для обробки отриманої відповіді від AI та переведення тексту в аудіосигнал. Функціональність та можливості бібліотеки Audio.h включають в себе: відтворення аудіо, можливість відтворювати аудіофайли з різних джерел, таких як вбудована пам'ять, SD-карти або мережеві ресурси.Запис аудіо: можливість записувати аудіосигнали з мікрофону або інших джерел.

Обробка звуку включає в себе функції обробки аудіо, такі як фільтрація, зміна гучності, еквалайзери та інші. Аудіоаналіз: можливість аналізувати аудіосигнали для розпізнавання мови, ідентифікації звуків або інші аудіоаналітичні завдання.

Основні використані функції з бібліотеки під час розробки проєкту були: audio.setPinout(), audio.setVolume() та audio.connecttospeech(). Бібліотека U8g2lib.h використана для роботи з OLED дисплеєм на мікроконтролері ESP32. Вона надає безліч потужних функції для відображення даних на дисплей з мікроконтролеру ESP32.

Загалом бібліотека допомогла з реалізацією відображення вводу даних користувача, також відображення анімації під час завантаження і саме основне це відображення отриманих даних після запиту користувача. Також значну роль в розробці відіграли наступні бібліотеки.

Бібліотека WiFi.h використовується для підключення ESP32 до інтернету, що дозволяє пристрою взаємодіяти з різними веб-сервісами та ресурсами через бездротову мережу Wi-Fi.

WiFiClientSecure.h використовується для встановлення безпечного з'єднання з сервером OpenAI. Ця бібліотека дозволяє ESP32 взаємодіяти з сервером через захищене з'єднання SSL/TLS, забезпечуючи безпеку передачі даних. U8g2lib.h про яку говорилось раніше є універсальною графічною бібліотекою, яка використовується для роботи з OLED дисплеєм. Вона надає зручний інтерфейс для виведення графічної інформації на дисплей, що дозволяє легко створювати різноманітні інтерфейси для користувачів.

Бібліотека ArduinoJson.h використовується для обробки та форматування даних у форматі JSON для викликів API. Вона дозволяє ESP32 взаємодіяти з різними веб-сервісами, отримувати та обробляти дані у форматі JSON

PS2KeyAdvanced.h використовується для обробки клавіатурного введення з клавіатури PS/2. Вона надає інструменти для зчитування натискань клавіш та їх кодів з клавіатури. PS2KeyMap.h використовується для обробки більш детальної інформації про клавіатурний ввід, забезпечуючи відповідність між кодами клавіш та символами, які вони представляють.

Використані бібліотеки у програмно-технічному асистенті на базі ESP32 забезпечує розширені можливості пристрою і полегшує розробку різноманітних функцій.

Бібліотеки WiFi та WiFiClientSecure дозволяють підключати пристрій до Інтернету і взаємодіяти з веб-сервісами, забезпечуючи безпеку передачі даних. U8g2lib дозволяє створювати зручний інтерфейс користувача на графічних дисплеях, а ArduinoJson спрощує роботу з даними у форматі JSON, що є стандартом

взаємодії з багатьма веб-сервісами. Бібліотеки PS2KeyAdvanced та PS2KeyMap дозволяють пристрою взаємодіяти з клавіатурою PS/2, що розширює можливості управління пристроєм.

Всі ці бібліотеки разом допомагають забезпечити стабільну та ефективну роботу програмно-технічного асистента, забезпечуючи його функціональність та ефективність.

## 2.4 Висновки

Вибір компонентів для проєкту голосового помічника ґрунтується на їх характеристиках, функціональності та сумісності. Мікроконтролер ESP32, завдяки своїй потужності, багатофункціональності та енергоефективності, є оптимальним вибором для даного проєкту.

OLED дисплей SSD1306 з діагоналлю 0.96" та роздільною здатністю 128x64 володіє високою контрастністю, яскравістю та низьким енергоспоживанням, що робить його оптимальним для візуалізації інформації.

Аудіопідсилювач MAX98357 I2S Amplifier забезпечує високу якість звуку, простоту підключення та керування, а також низьке енергоспоживання, що робить його ідеальним для даного проєкту.

Підключення компонентів до ESP32 здійснюється за допомогою GPIO, USB та I2S інтерфейсів. Важливо правильно підключити всі компоненти, дотримуючись полярності та напруги живлення.

Загалом, дані компоненти забезпечують високу продуктивність, функціональність та енергоефективність голосового помічника. Їхнє правильне використання дозволяє створити пристрій, який відповідає вимогам сучасної технології та задовольняє потреби користувачів у зручному та ефективному використанні.

# 3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ПРОГРАМНО-ТЕХНІЧНОГО АСИСТЕНТА ДЛЯ ВЗАЄМОДІЇ ІЗ МОВНОЮ МОДЕЛЛЮ НА ОСНОВІ ESP32

## 3.1 Функціонування пристрою

Програмно-технічний асистент для взаємодії з мовною моделлю який складається з трьох основних компонентів: мікроконтролерної плати ESP32, PS/2 клавіатури як пристрою введення та OLED дисплею для відображення результатів. Окрім цього, для відтворення звукової відповіді використовується додатковий компонент - підсилювач I2S MAX98357.

Загальна концепція роботи пристрою полягає в тому, що користувач вводить текстові запити за допомогою клавіатури, які потім передаються через WiFi-з'єднання на сервер OpenAI для обробки технологією ChatGPT. Отримана відповідь відображається на OLED дисплеї, а також може бути відтворена через аудіосистему з використанням підсилювача MAX98357.

Програмне забезпечення, що використовується у цьому асистенті, дозволяє користувачеві взаємодіяти з пристроєм через різні способи. Воно підтримує введення запитань через клавіатуру PS/2, яка забезпечує зручний та швидкий спосіб взаємодії. Крім того, програмне забезпечення інтегроване ChatGPT, що дозволяє асистенту розпізнавати та аналізувати запити користувача для надання точних та зрозумілих відповідей.

Основна функціональність програмно-технічного асистента полягає в наданні відповідей на запитання користувача та виконанні різних завдань. Він може надавати інформацію на OLED дисплей, що забезпечує зручний спосіб візуального сприйняття інформації. Крім того, асистент може відтворювати отримані відповіді за допомогою аудіопідсилювача та динаміка, що робить комунікацію з ним ще більш природньою та цікавою.

Представимо роботу системи у вигляді кроків наступна блок-схема приведена на рисунку 3.1.

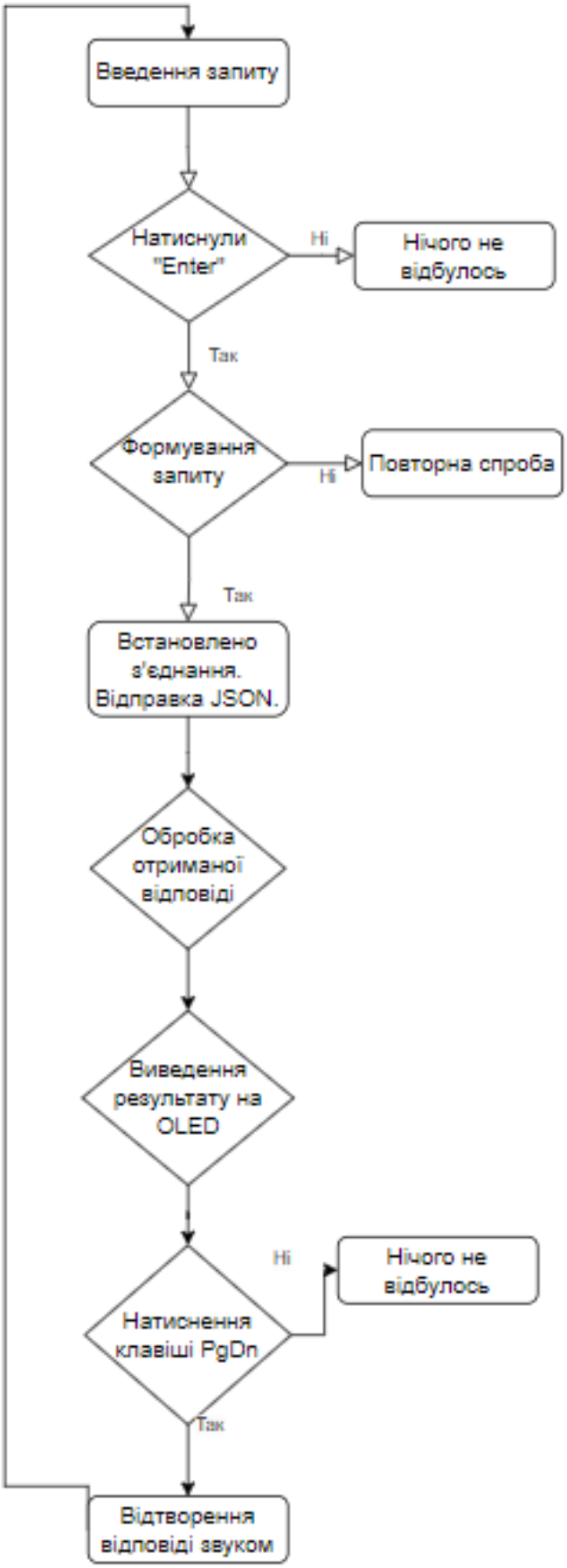



Рисунок 3.1 – Блок-схема функціонування пристрою

Програмне забезпечення, що використовується у цьому асистенті, дозволяє користувачеві взаємодіяти з пристроєм через різні способи. Воно підтримує введення запитань через клавіатуру PS/2, яка забезпечує зручний та швидкий спосіб взаємодії. Крім того, програмне забезпечення інтегроване ChatGPT, що дозволяє асистенту розпізнавати та аналізувати запити користувача для надання точних та зрозумілих відповідей.

Завдяки високій ефективності та різноманітним можливостям, програмно-технічний асистент стає незамінним помічником у різних сферах життя. Він дозволяє отримувати доступ до інформації та послуг швидко та зручно, що робить його незамінним інструментом для кожного користувача.

Передача запиту на пристрій для введення запиту користувач використовує стандартну PS/2 клавіатуру, підключену до ESP32. Натискання клавіш на клавіатурі реєструються спеціальною бібліотекою PS2KeyAdvanced, яка передає ці дані у вигляді ASCII символів до основної програми. Бібліотека відслідковує натискання всіх клавіш, в тому числі функціональних, таких як Shift, Ctrl, Enter тощо. Коли користувач натискає клавішу, в програмі викликається відповідна функція обробки цього натискання.

Наприклад, коли користувач натискає клавішу "A", бібліотека PS2KeyAdvanced передає цей код до програми, яка додає символ "A" до масиву, що зберігає введений текст. Або коли користувач натискає клавішу Enter, програма розпізнає це як команду на відправку введеного тексту на сервер ChatGPT.

Коли користувач натискає клавішу "Enter", програма збирає введений текст у спеціальний масив, який потім використовується для формування запиту до API ChatGPT. Для цього використовується бібліотека WiFiClientSecure, яка дозволяє встановити безпечне з'єднання з сервером OpenAI через протокол HTTPS.

Формування запиту спочатку програма формує JSON-об'єкт, який містить текст запиту користувача. Крім того, вона може використовувати додаткові "системні" команди, які впливають на тон і стиль відповіді ChatGPT. Наприклад, за замовчуванням програма відвідає по замовчуванню, але це можна дуже просто

змінити за необхідності, яка буде генерувати відповідь у обраному стилі. Користувач також може змінити цю системну команду, натиснувши клавішу Escape і ввівши свій власний текст.

Далі програма формує POST-запит до API ChatGPT, передаючи в тілі запиту сформований JSON-об'єкт. Бібліотека WiFiClientSecure забезпечує безпечне з'єднання та автентифікацію на сервері OpenAI за допомогою API-ключа, який зберігається у файлі credentials.h.

Обробка запиту та виведення результату після відправки запиту, програма очікує отримання відповіді від сервера ChatGPT. Коли відповідь надходить, вона читається за допомогою бібліотеки ArduinoJson. Програма аналізує структуру отриманих даних, визначає текст відповіді та іншу супутню інформацію.

Далі текст відповіді поступово виводиться на OLED дисплей, імітуючи процес "друку" тексту. Для цього використовується бібліотека U8g2lib, яка забезпечує просте керування виведенням тексту на дисплей, включаючи можливість прокрутки. Текст виводиться по одному слову за раз, з невеликою затримкою, що створює ефект "живої" взаємодії.

Одночасно, програма аналізує отриману відповідь і формує аудіо файл для відтворення через MAX98357. Для цього вона створює масив 16-бітних PCM-даних, які відповідають тексту відповіді. Для відтворення звукової відповіді використовується бібліотека Audio.h. Ця бібліотека дозволяє легко інтегрувати аудіо функціональність в програму, керуючи процесом декодування та відтворення звуку через підключені до ESP32 динаміки або навушники.

Програма формує аудіо дані у відповідному форматі на основі отриманої від ChatGPT відповіді, а бібліотека Audio.h забезпечує їх програвання. Таким чином, звукова відповідь синхронізується з текстовим виведенням на дисплей.

Клавіатура PS/2 підключена до виводів GPIO ESP32, які налаштовані на обробку переривань. Коли користувач натискає клавішу, в переривання викликається функція, яка відповідає за зчитування коду клавіші і його додавання

у масив повідомлення. Таким чином, програма отримує інформацію про кожне натискання клавіші в режимі реального часу.

OLED дисплей підключається до ESP32 через інтерфейс I2C або SPI, в залежності від обраної моделі. Бібліотека U8g2lib дозволяє просто керувати виведенням тексту на дисплей, включаючи можливість прокрутки. Програма використовує цю бібліотеку для поступового виведення тексту відповіді, імітуючи "друк" на екрані.

Підсилювач MAX98357 підключається до ESP32 через I2S інтерфейс. Програма формує аудіо дані у відповідному форматі і передає їх на MAX98357 для відтворення. Бібліотека, що працює з цим чіпом, забезпечує необхідну конфігурацію та управління I2S потоком.

Також було розроблено анімацію під час завантаження пристрою та під час очікування відповіді. Використовуючи масив вказівників на bitmaps, що представляють окремі кадри, було створено плавну та динамічну анімацію.

Кожен елемент цього масиву відповідає одному пікселю на дисплеї, і значення "0" та "1" в масиві вказують на те, чи повинен цей піксель бути чорним чи білим.

Шляхом почергового відображення цих bitmap-зображень на дисплеї з невеликими затримками між кадрами, вдалося досягти ефекту плавної та реалістичної анімації. Така техніка є ефективною для вбудованих систем, де важливі оптимізація пам'яті та швидкість обробки графічних даних.

Важливим аспектом функціонування програмно-технічного асистенту є забезпечення надійності та безпеки роботи пристрою. Оскільки пристрій передає запити до сервера OpenAI, необхідно подбати про захист конфіденційної інформації, яку користувач вводить через клавіатуру. Вище описаний функціонал також приведений на діаграмі станів на рисунку 3.2.

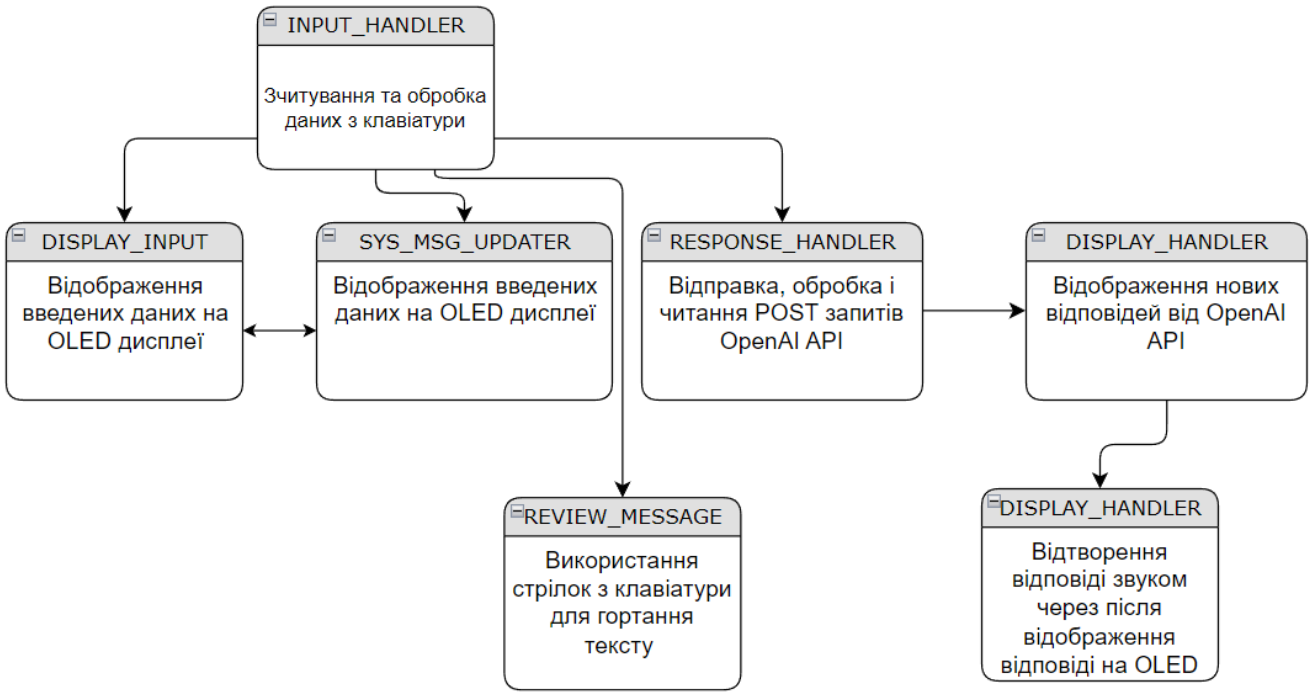



Рисунок 3.2 – Діаграма станів програмно-технічного асистента

Дана діаграма представляє архітектуру програмного забезпечення, що відповідає за обробку введення даних та відображення інформації на дисплеї. Вона складається з наступних блоків:

– INPUT_HANDLER - відповідає за зчитування та обробку даних, введених з клавіатури;
– DISPLAY_INPUT - відображає введені дані на OLED-дисплеї;
– SYS_MSG_UPDATER - оновлює відображення системних повідомлень на OLED-дисплеї;
– RESPONSE_HANDLER - обробляє та відображає відповіді, отримані через OpenAI API;
– DISPLAY_HANDLER - відображає нову інформацію на OLED-дисплеї;
– REVIEW_MESSAGE - здійснює перегляд та обробку текстових повідомлень, введених з клавіатури.

Ці компоненти взаємодіють один з одним для забезпечення ефективного введення даних користувачем та відображення результатів на дисплеї.

Для цього в програмному забезпеченні реалізовано механізми шифрування трафіку під час передачі даних. Зокрема, використовується бібліотека WiFiClientSecure, яка забезпечує безпечне HTTPS-з'єднання з сервером OpenAI. Це гарантує, що вміст запитів користувача не може бути перехоплений зловмисниками під час передачі.

Крім того, для захисту доступу до пристрою можна додати функцію авторизації, наприклад, за допомогою пароля або біометричної ідентифікації. Це дозволить запобігти несанкціонованому використанню ChatGPT Terminal сторонніми особами.

Надійність роботи пристрою також забезпечується за рахунок ретельного опрацювання програмного коду та тестування на різних сценаріях використання. Регулярне оновлення програмного забезпечення дозволятиме усувати виявлені проблеми та вразливості, підтримуючи пристрій у актуальному стані. Тому асистент може знайти широке практичне застосування в різних сферах діяльності.

## 3.2 Опис методів роботи з ключовими блоками пристрою

Код починається з імпорту необхідних бібліотек для роботи з різними компонентами системи. Бібліотека "Arduino.h" є основною бібліотекою для програмування на платформі Arduino і забезпечує доступ до базових функцій та константних значень.

Бібліотека "Audio.h" використовується для роботи з аудіо, декодування та відтворення звуку. Бібліотека "U8g2lib.h" надає функціональність для керування OLED дисплеєм, виведення тексту та графічних примітивів.

Далі імпортуються бібліотеки для роботи з Wi-Fi та безпечним з'єднанням з сервером OpenAI: "WiFi.h" та "WiFiClientSecure.h". Бібліотека "ArduinoJson.h" використовується для серіалізації та десеріалізації JSON-об'єктів, необхідних для взаємодії з API ChatGPT.

Бібліотеки "PS2KeyAdvanced.h" та "PS2KeyMap.h" забезпечують роботу з PS/2 клавіатурою, реєстрацію натискань клавіш та перемикання розкладок.

Після імпорту бібліотек визначаються константи та макроси для налаштування різних параметрів системи, таких як номери пінів для підключення компонентів, розміри буферів, інтервали затримок та інші.

Наприклад, константи SCREEN_WIDTH та SCREEN_HEIGHT задають розміри OLED дисплея, а MAX_TOKENS визначає максимальну кількість токенів, які можуть бути отримані від ChatGPT в одній відповіді.

Після цього оголошуються глобальні змінні, структури даних та масиви для зберігання стану програми, повідомлень, ролей учасників діалогу тощо. Визначається перелік станів (enum States), в яких може перебувати програма, та структура StateVars для збереження поточних значень змінних стану.

Також оголошується масив messagesOut для зберігання повідомлень, які будуть надсилатися до ChatGPT, та об'єкт systemMessage для зберігання системного повідомлення, яке визначає налаштування ChatGPT.

Після оголошення змінних ініціалізуються об'єкти для роботи з компонентами системи: audio для роботи з аудіо, u8g2 для керування дисплеєм, keyboard для взаємодії з клавіатурою та keymap для налаштування розкладки клавіатури. На схемі відображеній на рисунку 3.3 відображенні основні методи.

Функція setup() викликається одноразово під час запуску програми. В ній відбувається початкова конфігурація системи: ініціалізація послідовного порту Serial для виведення відлагоджувальної інформації, затримка для стабілізації системи, встановлення з'єднання з Wi-Fi мережею, ініціалізація дисплея u8g2 та налаштування шрифту, відображення вітальної анімації на дисплеї, ініціалізація аудіо компонентів та встановлення гучності звуку. На схемі відображеній на рисунку 3.3 відображенні основні методи.

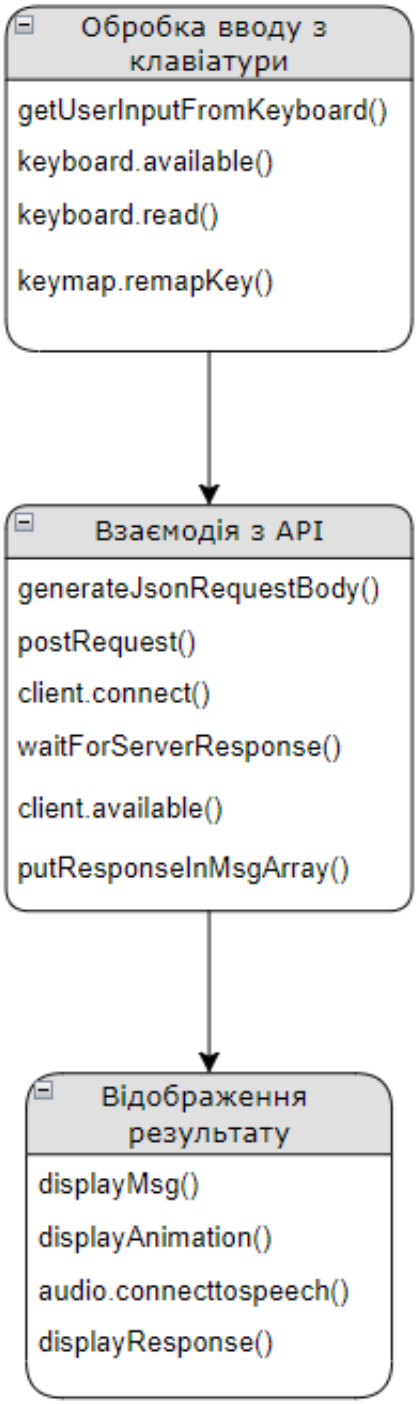



Рисунок 3.3 – Схема використаних методів

Основна функція для обробки вводу з клавіатури називається getUserInputFromKeyboard(). Вона викликається в циклі loop() і відповідає за реєстрацію натискань клавіш, формування повідомлення користувача та керування станами програми.

Функція getUserInputFromKeyboard() перевіряє, чи є доступні натискання

клавіш за допомогою методу keyboard.available() бібліотеки PS2KeyAdvanced. Якщо натискання є, вона зчитує код натиснутої клавіші за допомогою методу keyboard.read(). Далі код клавіші перетворюється на відповідний символ ASCII з урахуванням поточної розкладки клавіатури за допомогою функції keymap.remapKey().

Залежно від натиснутої клавіші виконуються відповідні дії. Наприклад, якщо натиснуто клавішу Enter, то в залежності від поточного стану програми, введене повідомлення користувача зберігається в масиві messagesOut, лічильник повідомлень збільшується, а стан змінюється на RESPONSE_HANDLER для відправки запиту до ChatGPT. Якщо натиснуто Backspace або Delete, то останній символ в повідомленні видаляється. Натискання клавіші Esc перемикає стан програми на SYS_MSG_UPDATER для оновлення системного повідомлення. Стрілки вгору та вниз дозволяють прокручувати попередні відповіді ChatGPT на дисплеї.

Під час введення тексту повідомлення поміщаються в буфер content структури Message, а індекс inputIdx відстежує поточну позицію курсору в цьому буфері. Змінна bufferChange використовується для відстеження змін в буфері, щоб оновлювати відображення введеного тексту на дисплеї.

Взаємодія з API ChatGPT відбувається за допомогою функцій generateJsonRequestBody() та postRequest().

Функція generateJsonRequestBody() формує JSON-об'єкт, який буде надіслано до ChatGPT як тіло POST-запиту. Об'єкт містить масив повідомлень messages, що складається з системного повідомлення systemMessage та повідомлень користувача і асистента з масиву messagesOut. Кількість повідомлень, що включаються в запит, обмежується константою MAX_MESSAGES. Якщо кількість повідомлень перевищує це значення, то в запит включаються лише останні MAX_MESSAGES повідомлень.

Сформований JSON-об'єкт серіалізується за допомогою бібліотеки ArduinoJson і повертається функцією generateJsonRequestBody().

Функція postRequest() виконує безпосередню відправку POST-запиту до серверу OpenAI. Вона приймає два аргументи: вказівник на JSON-об'єкт запиту та об'єкт клієнта WiFiClientSecure для встановлення захищеного з'єднання.

Спочатку функція встановлює з'єднання з сервером OpenAI за допомогою методу client.connect(). Далі вона формує заголовки HTTP-запиту, включаючи Content-Type, Content-Length, авторизаційний токен та інші необхідні поля. Після цього функція серіалізує JSON-об'єкт запиту та відправляє його в тілі POST-запиту на сервер OpenAI.

Після відправки запиту програма очікує відповідь від сервера.

Для очікування відповіді від сервера OpenAI використовується функція waitForServerResponse(). Вона приймає об'єкт клієнта WiFiClientSecure як аргумент та перевіряє, чи надійшли якісь дані від сервера, викликаючи метод client.available().

Якщо дані ще не надійшли, функція waitForServerResponse() відображає анімацію очікування на дисплеї, викликаючи displayAnimation() з відповідним повідомленням WaitingForApiResponseMsg. Ця анімація циклічно відображається доки не мине час очікування SERVER_RESPONSE_WAIT_TIME (за замовчуванням 20 секунд). Якщо за цей час сервер не надіслав відповідь, функція повертає false, що вказує на помилку з'єднання.

Коли дані від сервера надходять, waitForServerResponse() повертає true, що дозволяє програмі перейти до наступного етапу - розбору відповіді сервера.

Розбір відповіді виконується функцією putResponseInMsgArray(). Вона приймає об'єкт клієнта WiFiClientSecure та кількість повідомлень, що були надіслані в запиті.

Спочатку функція пропускає заголовки HTTP-відповіді за допомогою методу client.find("\r\n\r\n"). Далі вона створює об'єкт фільтра filter за допомогою бібліотеки ArduinoJson, який визначає, які поля JSON-відповіді потрібно розібрати.

Після цього функція deserializeJson() викликається для розбору JSON-відповіді з використанням створеного фільтра. Якщо під час десеріалізації виникла

помилка, функція відображає відповідну анімацію на дисплеї та повертає 0, вказуючи на помилку.

Якщо десеріалізація пройшла успішно, функція putResponseInMsgArray() зберігає роль "assistant" та текст відповіді з поля "content" у масив messagesOut на позицію, визначену кількістю повідомлень та константою MAX_MESSAGES. Також обчислюється довжина відповіді в токенах і зберігається у змінній responseLength.

Після успішного збереження відповіді функція повертає 1, що дозволяє програмі перейти до наступного стану - виведення відповіді на дисплей та її озвучування.

Відображення результату для виведення інформації на OLED дисплей використовуються функції displayMsg(), displayAnimation() та displayResponse().

Функція displayMsg() призначена для виведення тексту на дисплей. Вона приймає рядок, який потрібно вивести (msg), індекси початку (startIdx) та кінця (endIdx) підрядка, що буде виведений, а також булевий прапорець setDelay, який вказує, чи потрібно вставляти затримку між виведенням слів.

Функція починає з очищення буфера дисплея за допомогою методу u8g2.clearBuffer(). Далі вона встановлює курсор у лівий верхній кут дисплея та починає виводити символи рядка msg, розбиваючи їх на рядки відповідно до ширини дисплея (MAX_CHAR_PER_OLED_ROW). Якщо прапорець setDelay встановлений, функція робить паузу після виведення кожного слова, імітуючи ефект "друку" тексту.

Якщо кількість символів у рядку перевищує MAX_CHARS_ON_SCREEN, функція автоматично прокручує текст на дисплеї, очищаючи буфер та зміщуючи індекс startIdx.

Функція displayAnimation() призначена для відображення анімованих зображень (наприклад, під час завантаження або очікування відповіді від сервера). Вона приймає час відображення анімації (displayTime), рядок повідомлення для виведення (displayMessage) та інтервал затримки між кадрами анімації

(delayInterval, за замовчуванням 100 мс).

Функція використовує масив епд_bitmaps_allArray, який містить покажчики на бітмапи окремих кадрів анімації. Вона циклічно відображає ці бітмапи на дисплеї, чергуючи їх з виведенням рядка displayMessage. Загальний час відображення анімації визначається добутком кількості кадрів та інтервалу delayInterval.

Функція displayResponse() відповідає за виведення відповіді від ChatGPT на дисплей. Вона викликається зі станів DISPLAY_HANDLER та REVIEW_MESSAGE і визначає, який індекс у масиві messagesOut відповідає останній отриманій відповіді.

У стані DISPLAY_HANDLER функція викликає displayMsg() для виведення всієї відповіді з нуля, імітуючи ефект "друку" з затримкою між словами. Водночас вона запускає відтворення аудіо відповіді за допомогою функції audio.connecttospeech().

У стані REVIEW_MESSAGE функція displayMsg() викликається без затримки, щоб відобразити лише частину відповіді, яка поміщається на екрані. Змінна displayOffset використовується для відстеження зміщення під час прокручування тексту відповіді стрілками вгору/вниз.

Після виведення відповіді на дисплей стан програми повертається до INPUT_HANDLER для введення наступного запиту користувача.

Для озвучування відповідей від ChatGPT використовується функція audio.connecttospeech(), яка викликається з функції displayResponse().

Функція audio.connecttospeech() приймає два аргументи: рядок, що містить текст відповіді, та мовний код (наприклад, "en" для англійської мови). Вона відповідає за перетворення текстової відповіді у звукові дані, які можуть бути відтворені через аудіопідсилювач та динаміки.

Процес озвучування відповіді відбувається наступним чином:

Функція audio.connecttospeech() аналізує вхідний текст і формує відповідний звуковий потік у форматі PCM (імпульсно-кодова модуляція).

Сформовані PCM-дані передаються на аудіопідсилювач MAX98357, підключений до ESP32 через інтерфейс I2S.

Конфігурація виводів I2S для передачі аудіо даних на MAX98357 визначається у функції setup() за допомогою виклику audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT).

Гучність відтворення звуку встановлюється викликом audio.setVolume(100), де аргумент 100 відповідає максимальній гучності.

Безпосереднє відтворення аудіо потоку забезпечується функцією audio.loop(), яка викликається циклічно у головному циклі loop().

Таким чином, коли функція displayResponse() викликає audio.connecttospeech(), генерується звуковий потік на основі тексту відповіді від ChatGPT. Але відтворення звуку спрацьовує після натиснення клавіші PgDn. Цей потік передається на аудіопідсилювач MAX98357, який розшифровує PCM-дані та виводить відповідний аналоговий звуковий сигнал на підключені динаміки або навушники.

Синхронізація озвучування відповіді з її текстовим виведенням на дисплей забезпечується тим, що функції displayMsg() та audio.connecttospeech() викликаються послідовно з функції displayResponse().

У коді присутні додаткові функції та методи, які забезпечують коректну роботу програми та надають додаткову функціональність.

Функція printToConsoleMessageArray() використовується для виведення вмісту масиву messagesOut у Serial Monitor для відлагодження. Вона циклічно проходить по всіх елементах масиву та виводить їх вміст, а також розмір останньої отриманої відповіді від ChatGPT.

Функція displayKeyboardInput() відповідає за відображення введеного користувачем тексту на дисплеї в режимі реального часу. Вона викликається зі станів INPUT_HANDLER та SYS_MSG_UPDATER і перевіряє прапорець bufferChange, який вказує на те, що в буфер введення були внесені зміни. Якщо

bufferChange встановлений, функція викликає displayMsg() для виведення поточного вмісту буфера повідомлення.

Стан REVIEW_MESSAGE дозволяє користувачу переглядати попередні відповіді від ChatGPT, прокручуючи їх на дисплеї за допомогою клавіш зі стрілками. У цьому стані функція displayResponse() викликає displayMsg() без затримки, щоб відобразити лише частину відповіді, яка поміщається на екрані. Змінна displayOffset відстежує зміщення під час прокручування тексту відповіді.

Стан SYS_MSG_UPDATER дозволяє користувачу оновити системне повідомлення для ChatGPT, натиснувши клавішу Esc. У цьому стані введений текст записується в об'єкт systemMessage замість messagesOut. Після натискання Enter системне повідомлення оновлюється, і програма повертається до стану INPUT_HANDLER.

Під час компіляції з визначеним макросом DEBUG у коді додатково виводиться відлагоджувальна інформація в Serial Monitor. Наприклад, сформований JSON-запит до ChatGPT виводиться за допомогою функції serializeJsonPretty(), що полегшує відлагодження процесу взаємодії з API.

## 3.3 Демонстрація роботи пристрою

Перед початком роботи з програмно-технічним асистентом необхідно виконати наступні підготовчі кроки:

– Підключити PS/2 клавіатуру до відповідних виводів GPIO ESP32. Клавіатура буде використовуватись для введення запитів до ChatGPT;

– Переконатись, що ESP32 підключений до мережі Wi-Fi та має доступ до Інтернету для взаємодії з серверами OpenAI;

– Підключити живлення до ESP32 для забезпечення функціонування всіх компонентів пристрою.

Після виконання цих підготовчих кроків програмно-технічний асистент буде готовий до роботи.

При подачі живлення на мікроконтролер ESP32 відбувається ініціалізація всіх компонентів системи відповідно до налаштувань, визначених у коді програми. Цей процес супроводжується відображенням вітальної анімації на OLED дисплеї. На рисунку 3.4 відображена анімація першого завантаження.

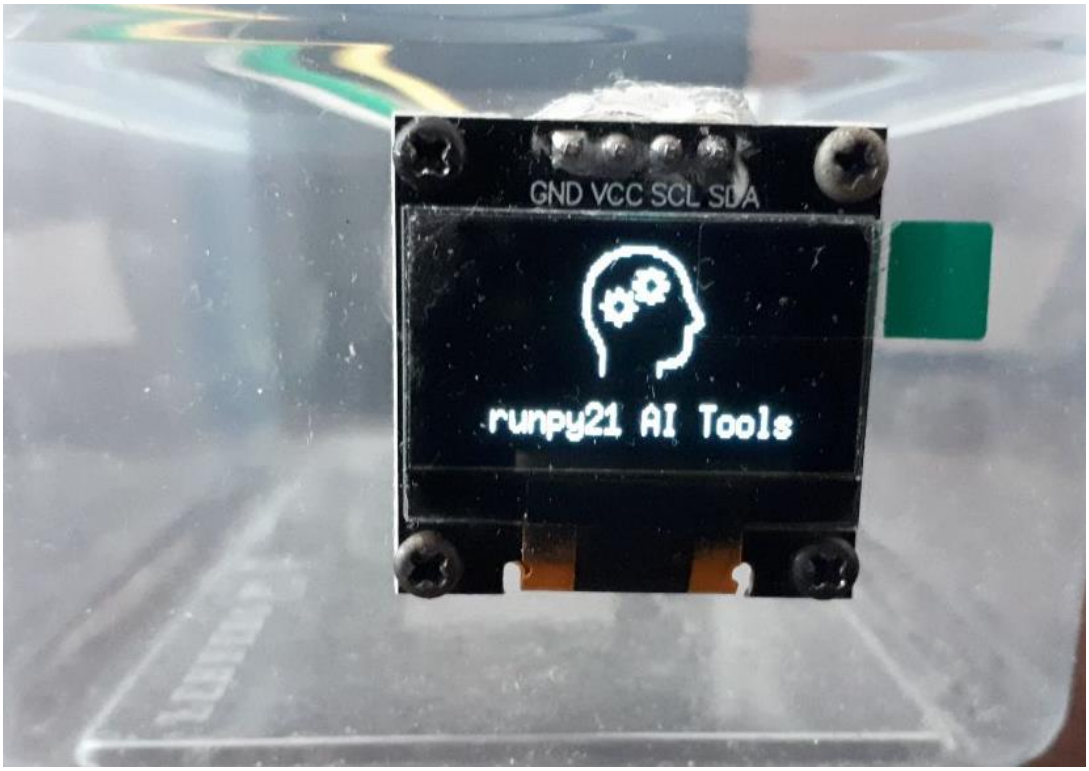



Рисунок 3.4 – Анімація завантаження пристрою

Вітальна анімація складається з декількох кадрів-бітмап, закодованих у програмі у вигляді масиву. Функція displayAnimation() циклічно відображає ці бітмапи на дисплеї з певним інтервалом затримки між кадрами, створюючи ефект плавної анімації.

Під час завантаження пристрою виконується підключення до WiFi після успішного підключення анімація зникає і програмно-технічний асистент готовий до роботи. Після завантаження можна побачити повідомлення яке закликає до набору запиту на рисунку 3.5.


Рисунок 3.5 – Вступне повідомлення

По мірі введення тексту він відображається на OLED дисплеї у режимі реального часу функцією displayKeyboardInput(). Ця функція перевіряє прапорець bufferChange, який вказує на те, що зміни відбулися у тимчасовому буфері для збереження тексту.

Якщо bufferChange встановлений, функція displayMsg() викликається для виведення поточного вмісту буферу на дисплей. Коли користувач натискає клавішу Enter, програма інтерпретує це як команду на відправлення введеного запитання до ChatGPT.

Введений текст зберігається у масиві messagesOut, який акумулює всі повідомлення користувача та відповіді ChatGPT. Потім встановлюється стан програми RESPONSE_HANDLER для ініціації процесу відправки запиту. Введений текс приведений на рисунку 3.6.


Рисунок 3.6 – Результат набору тексту

Користувач вводить перше запитання за допомогою PS/2 клавіатури, наприклад: " What is artificial intelligence?". Після натискання клавіші Enter запит відправляється до серверів OpenAI.

Коли користувач натискає клавішу Enter, програма інтерпретує це як команду на відправлення введеного запитання до ChatGPT. Введений текст зберігається у масиві messagesOut, який акумулює всі повідомлення користувача та відповіді ChatGPT.

Потім встановлюється стан програми RESPONSE_HANDLER для ініціації процесу відправки запиту. На дисплеї відображається анімація очікування відповіді від ChatGPT. Анімація очікування приведена на рисунку 3.7.

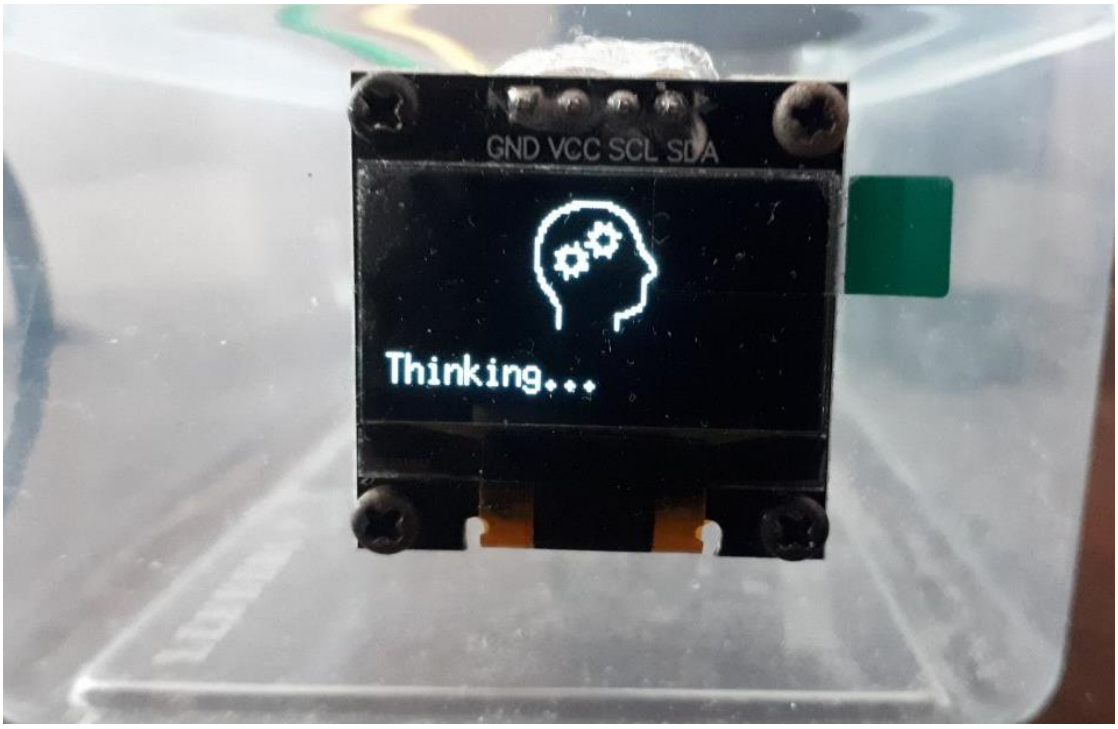



Рисунок 3.7 – Анімація очікування

Коли відповідь від ChatGPT отримана, її текст поступово виводиться на OLED дисплей з ефектом "друку" (із затримкою між словами). Одночасно відтворюється озвучена версія відповіді через під'єднані динаміки або навушники.

Після завершення виведення відповіді користувач може прокрутити її текст на дисплеї за допомогою клавіш зі стрілками вгору/вниз. У стані RESPONSE_HANDLER відбувається формування JSON-запиту для відправки його на сервер OpenAI. Функція generateJsonRequestBody() створює JSON-об'єкт із наступною структурою:

```
{
"model": "gpt-3.5-turbo",
"messages": [
{"role": " assistant ", "content": "You are a helpful assistant."},
{"role": "user", "content": "Введений користувачем текст запиту"}
]
}
```

Ця структура містить повідомлення двох ролей: "system" (системне

повідомлення, яке задає контекст взаємодії) та "user" (текстовий запит користувача). Сформований JSON-об'єкт серіалізується у рядок функцією serializeJson() з бібліотеки ArduinoJson.

Після цього функція postRequest() встановлює захищене SSL-з'єднання з сервером api.openai.com за допомогою бібліотеки WiFiClientSecure. В заголовки POST-запиту включається авторизаційний токен доступу до API ChatGPT. Серіалізований JSON-об'єкт запиту передається у тілі цього POST-запиту. На рисунку 3.8 відображено результат відповіді на запит.

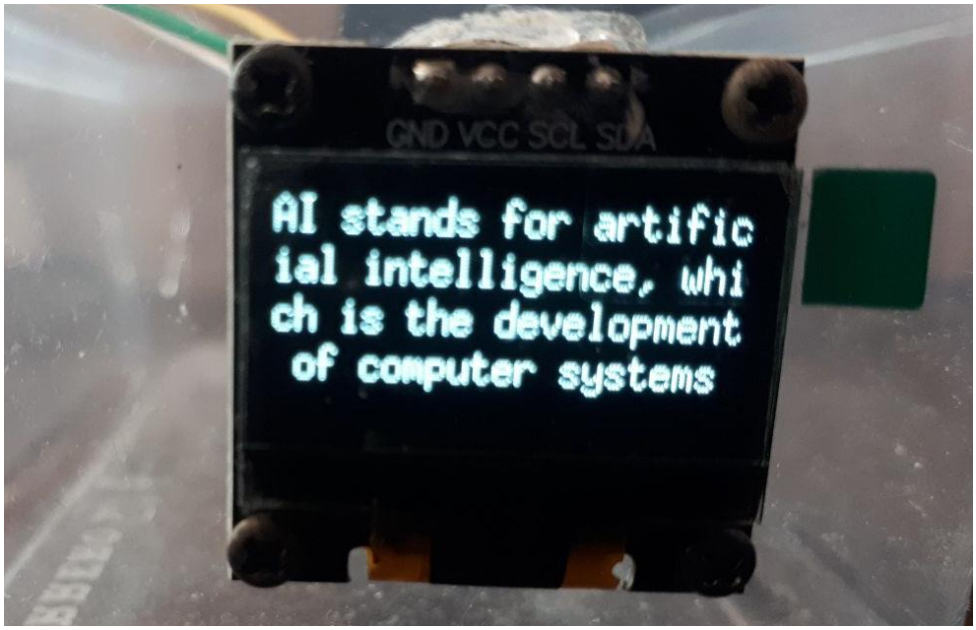



Рисунок 3.8 – Результат відповіді на запит

У наступному стані програми DISPLAY_HANDLER відбувається виведення тексту відповіді від ChatGPT на OLED дисплей. Це реалізується функцією displayResponse(). Функція displayResponse() викликає функцію displayMsg() для поступового виводу тексту відповіді на дисплей, імітуючи ефект друкування тексту. Кожне слово відповіді виводиться на дисплей з невеликою затримкою між словами для посилення цього ефекту.

Якщо довжина відповіді перевищує розмір дисплея, функція displayMsg() автоматично забезпечує прокручування тексту, дозволяючи переглянути весь вміст

відповіді. На рисунку 3.9 відображена можливість гортання.

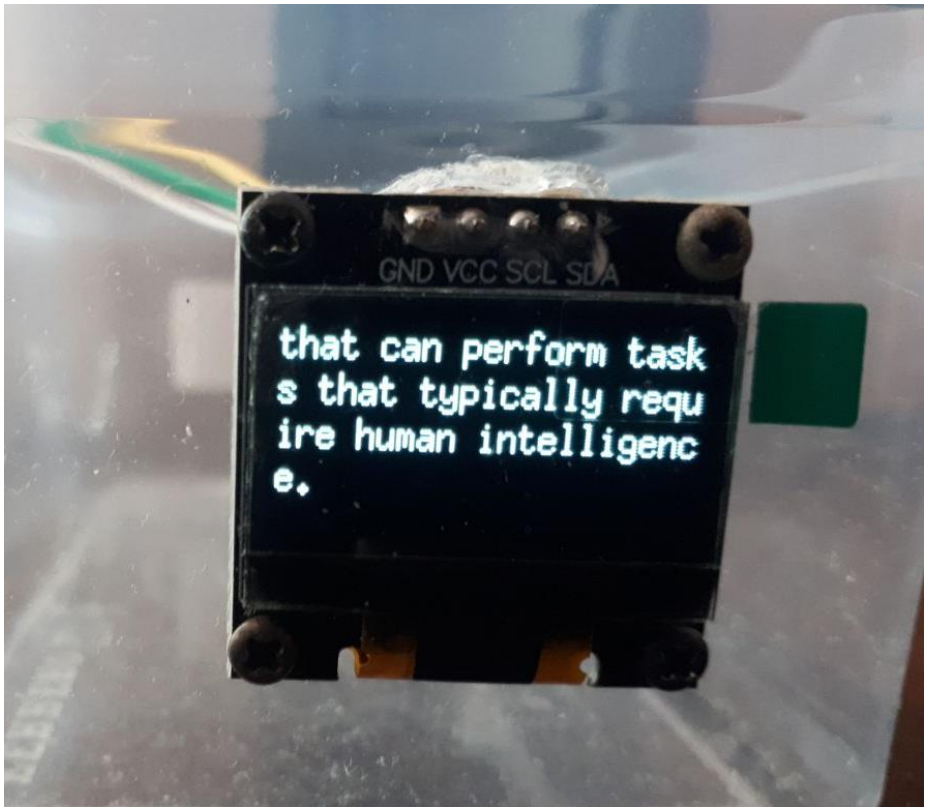



Рисунок 3.9 – Гортання відповіді

Також можливо увімкнути озвучування відповіді після натиснення клавіші PageDown. Озвучування реалізовано за допомогою бібліотеки Audio.h та аудіопідсилювача MAX98357. Для того, щоб прослухати відповідь потрібно натиснути клавішу Page Down. Бібліотека Audio.h декодує цей текст у відповідний звуковий потік у форматі PCM (імпульсно-кодова модуляція).

Сформований PCM-потік передається на аудіопідсилювач MAX98357 через інтерфейс I2S. Цей інтерфейс дозволяє передавати цифрові аудіодані безпосередньо на вхід MAX98357, який виконує їх декодування та перетворення в аналоговий звуковий сигнал.

У функції setup() виконується налаштування виводів ESP32 для роботи з I2S інтерфейсом за допомогою виклику audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT). Це призначає певні виводи GPIO для передачі тактових сигналів, сигналів синхронізації та самих аудіоданих до MAX98357. Гучність відтворення

звуку регулюється викликом audio.setVolume(100), де аргумент 100 відповідає 100% гучності. Можна змінювати це значення в діапазоні 0-100 для зменшення або збільшення гучності.

Відтворення сформованого звукового потоку відбувається циклічно в головній функції loop() програми за рахунок виклику audio.loop(). Ця функція читає чергові порції PCM-даних та передає їх на MAX98357 через I2S інтерфейс.

Підсилювач MAX98357 перетворює отримані цифрові аудіодані в аналоговий сигнал та подає його на підключені до своїх виходів динаміки або навушники. Таким чином, користувач може почути озвучену версію відповіді від ChatGPT одночасно з відображенням її тексту на OLED дисплеї. Завдяки озвучуванню відповідей взаємодія з програмно-технічним асистентом стає більш природною та зручною, адже користувач може як читати текст, так і прослуховувати відповідь голосово. Це особливо корисно в ситуаціях, коли зручніше сприймати інформацію на слух, наприклад, під час ходьби, керування транспортним засобом тощо. Після перегляду відповіді користувач може ввести новий запит, що дозволяє підтримувати безперервний діалог з ChatGPT. При введенні наступного запиту він автоматично додається до існуючого контексту діалогу у масиві messagesOut.

Цикл введення запиту, формування JSON-запиту, відправлення його на сервер OpenAI, очікування відповіді, обробка отриманої відповіді та її відображення і озвучування повторюється. Таким чином, підтримується природний діалог між користувачем та ChatGPT із зберіганням попереднього контексту розмови.

Після завершення роботи з програмно-технічним асистентом на базі ChatGPT користувач може вимкнути живлення мікроконтролера ESP32. При наступному ввімкненні пристрою весь процес розпочнеться спочатку з відображення вітальної анімації на дисплеї.

Програма не зберігає стан діалогу між сесіями роботи, тому при кожному новому запуску історія спілкування починається з нуля. Проте завдяки зручному

інтерфейсу введення тексту та високій швидкодії ChatGPT користувач може швидко відновити потрібний контекст розмови.

Таким чином, демонстрація роботи програмно-технічного асистента на базі ChatGPT показує, що цей пристрій забезпечує інтуїтивну та багатофункціональну взаємодію з потужною мовною моделлю. Поєднання текстового та голосового введення/виведення даних, а також можливість налаштування параметрів ChatGPT робить цей асистент корисним помічником у різних сферах діяльності.

## 3.4 Висновки

В даному розділі представлено основний алгорим роботи пристрою, детально описані методи, що забзпечують його роботу, зокрема програмне забезпечення, що використовується в асистенті, дозволяє користувачеві взаємодіяти з пристроєм через різні способи, підтримуючи введення запитань через клавіатуру PS/2 та інтеграцію з ChatGPT для розпізнавання та аналізу запитів користувача.

Дані з клавіатури PS/2 зчитуються за допомогою методу getKeyboardInput() натискання клавіші "Enter" запускає подальшу обробку введеного запиту користувача. Створюється HTTP-запит з тілом, що містить введену користувачем інформацію (generateApiRequestBody()) встановлюється з'єднання з сервером за допомогою client.connect() надсилається HTTP-запит і очікується відповідь сервера (waitForServerResponse()) перевіряється доступність сервера (client.available()) зчитується відповідь сервера у форматі JSON (bufferedClientResponseStream()).

Отримана від ChatGPT відповідь відображається на OLED дисплеї (displayMsg()) відповідь також відтворюється через аудіопідсилювач за допомогою функцій audio.connect() та audio.play() якщо відповідь не була отримана, відображається відповідне повідомлення (displayMsg("Нічого не відбувається")).

Після відображення та відтворення відповіді, програма повертається до стану очікування введення нового запиту користувача. За результатами тестування поставлена задача була виконана.

# ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень поставлена мета виконана, а саме було розроблено програмно-технічний асистент для взаємодії з мовною моделлю ChatGPT на базі мікроконтролера ESP32 з використанням OLED дисплея, клавіатури та аудіопідсилювача.

У першому розділі проведено аналіз актуальності розробки такого пристрою, огляд існуючих рішень та обґрунтування потенційного впливу на розвиток інтерфейсів користувача та взаємодії людей зі штучним інтелектом. Розглянуто переваги та недоліки наявних варіантів взаємодії з мовними моделями, таких як голосові асистенти, мобільні додатки та веб-інтерфейси.

У другому розділі проведено вибір та обґрунтування використання мікроконтролера ESP32 як основи для розробки пристрою, розглянуто його технічні характеристики та переваги, зокрема наявність вбудованого WiFi модуля, достатню обчислювальну потужність та енергоефективність. Також здійснено вибір периферійних пристроїв, зокрема OLED дисплея SSD1306 та аудіопідсилювача MAX98357 з урахуванням їх технічних параметрів та доцільності для даного проєкту.

У третьому розділі продемонстровано роботу розробленого пристрою, описано процес підготовки до роботи, ініціалізацію компонентів та послідовність дій користувача під час взаємодії з програмно-технічним асистентом. Детально розглянуто процеси введення запитів за допомогою клавіатури, відправки їх до ChatGPT через WiFi з'єднання, отримання та відображення відповідей на OLED дисплеї та озвучування голосом через аудіопідсилювач. Наведено приклади роботи з пристроєм та наведено його функціонування в різних сценаріях використання.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ


1. Gupta S., Gandhi R., & Pavalier V. Human-Machine Interface Design for Interacting with Language Models: *Journal of intelligent systems*. 2019. P. 421-439.
2. Dott E., Schmitzer A., Huang, Z. Artificial Intelligence on Resource-Constrained Devices: Challenges and Solutions: *Proceedings of the signal and systems Conference*. 2020. P. 44-76.
3. Lopez G., Castro R., Martinez, P. Integrating Language Models into Internet of Things Systems: *Journal of internet of things*. 2021. P. 67-84.
4. Martin, T., Johnson, K., Smith, B. Developing Voice Interfaces for Interaction with ChatGPT: *Proceedings of the computational linguistics conference.* 2018. P. 134-149.
5. Nielsen D., Anderson K., & Brown R. Optimizing Energy Consumption for AI on Embedded Systems: *Journal of low-power systems*. 2019. P. 98-114.
6. O'Brien M., Dugan T., Cohen L. Designing User Interfaces for Interaction with Language Models: *Proceedings of the human-computer interaction conference*. 2020, P. 221-237.
7. Patterson D., Gonzalez R. Computer Vision: Algorithms and applications. New york: Cambridge University Press. 2017, 1232 p.
8. Russell S., Norvig P. Artificial Intelligence: A modern approach. New Jersey: Pearson, 2020, 1127 p.
9. Silva F., Torres R., Oliveira L. Implementing AI on embedded systems: *journal of embedded systems*. 2019. P. 51-68.
10. Smith S., Jones, D., Brown M. Designing interfaces for interacting with language models on mobile devices: *Proceedings of the mobile computing and communications conference*. 2017. P. 89-104.
11. Huang T., Lin J., Li K. Optimizing language model performance on resource-constrained devices: *Journal of artificial intelligence*. 2020. P. 321-342.

12. Chen L., Yang S., Wu, C. Developing voice interfaces for interaction with language models: *proceedings of the computational linguistics conference*. 2021. P. 201-219.
13. Sharma P., Gupta A., Singh R. Implementing AI on embedded systems: A review and challenges. *Journal of embedded systems*. 2019. P. 121-139.
14. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016, 456 p.
15. Hochreiter S., Schmidhuber J. Long short-term memory. Neural computation. 1997, 132 p.
16. Karpathy A. The unreasonable effectiveness of recurrent neural networks, 2015, 336 p.
17. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems. 2012, 522 p.
18. LeCun Y., Bengio Y., & Hinton G. Deep learning. Nature. 2015, 444 p.
19. Mikolov T., Sutskever I., Chen K., Corrado G., & Dean J. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems. 2013, 894 p.
20. Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I. Language models are unsupervised multitask learners. 2019, 129 p.
21. Rumelhart D. E., Hinton G. E., & Williams R. J. Learning representations by back-propagating errors. Nature. 2016, 536 p.
22. Sutskever I., Vinyals O., & Le Q. V. Sequence to sequence learning with neural networks. Advances in neural information processing systems. 2014, 232 p.
23. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N. Attention is all you need. Advances in neural information processing systems. 2017, 402 p.
24. Yao L., Poblenz A., Dagunturi D., Covington B., Bernard D., & Lyman K. Transforming Text via Machine Learning. *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics*. 2019. P. 285-292.

25. Young T., Hazarika D., Poria S., Cambria, E. Recent trends in deep learning based natural language processing. *IEEE Computational intelligence magazine*, 2018. P. 55-75.
26. Zhou J., & Xu W. End-to-End learning of semantic role labeling using recurrent neural networks. 2020. 1137 p.
27. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. 2019, 186 p.
28. Radford A., Narasimhan K., Salimans T., Sutskever I. Improving language understanding by generative pre-training. 2017, 292 p.
29. Brown T. B., Mann B., Ryder N., Subbiah M., Kaplan J., Dhariwal P., Amodei D. Language models are few-shot learners. advances in neural. 2020, 344 p.
30. Wang Y., Yang D., He H., Zheng L. Scalable Language Models: Next steps for efficient optimization and transfer on resource-constrained devices. *Journal of artificial intelligence and natural language processing*. 2022. P. 98-122.
31. Петрик В.М., Микитків В.В., Кружилко О.Є. Мікроконтролери та цифрові пристрої. навч. посіб. Львів: Видавництво Львівської політехніки, 2018. 167с.
32. Кузьмін І.В., Васюткін В.С., Фрунзе І.В. Інтелектуальні системи та технології. навч. посіб. Київ: Наукова думка, 2020. 298 с.
33. Букун Н.Г., Глудкін О.П., Горбатий І.В. Мікроконтролери AVR та ARM: проектування систем керування. Дніпро: ДВНЗ Український державний хіміко-технологічний університет. 2019, 137 с.
34. Коровкін М.В., Петрушин В.С., Литвак С.В. Мікроконтролери сімейства ARM: архітектура, програмування, інтерфейси. Харків: ХАІ, 2022. 207 с.
35. Цибульський Г.М., Лавриков Ю.А., Короткий С.А. Інтелектуальні системи керування технологічними процесами. Київ: Техніка, 2021. 259 с.
36. Мартиненко В.В., Ланкін М.В., Татков О.І. Програмування мікроконтролерів Arduino. Київ: Видавництво БХВ-Петербург, 2019. 289 с.
37. Мельников В.П., Михайлов П.С., Заріцький О.В. Цифрові пристрої та мікропроцесорні системи. Київ: КПІ ім. Ігоря Сікорського, 2020. 198 с.

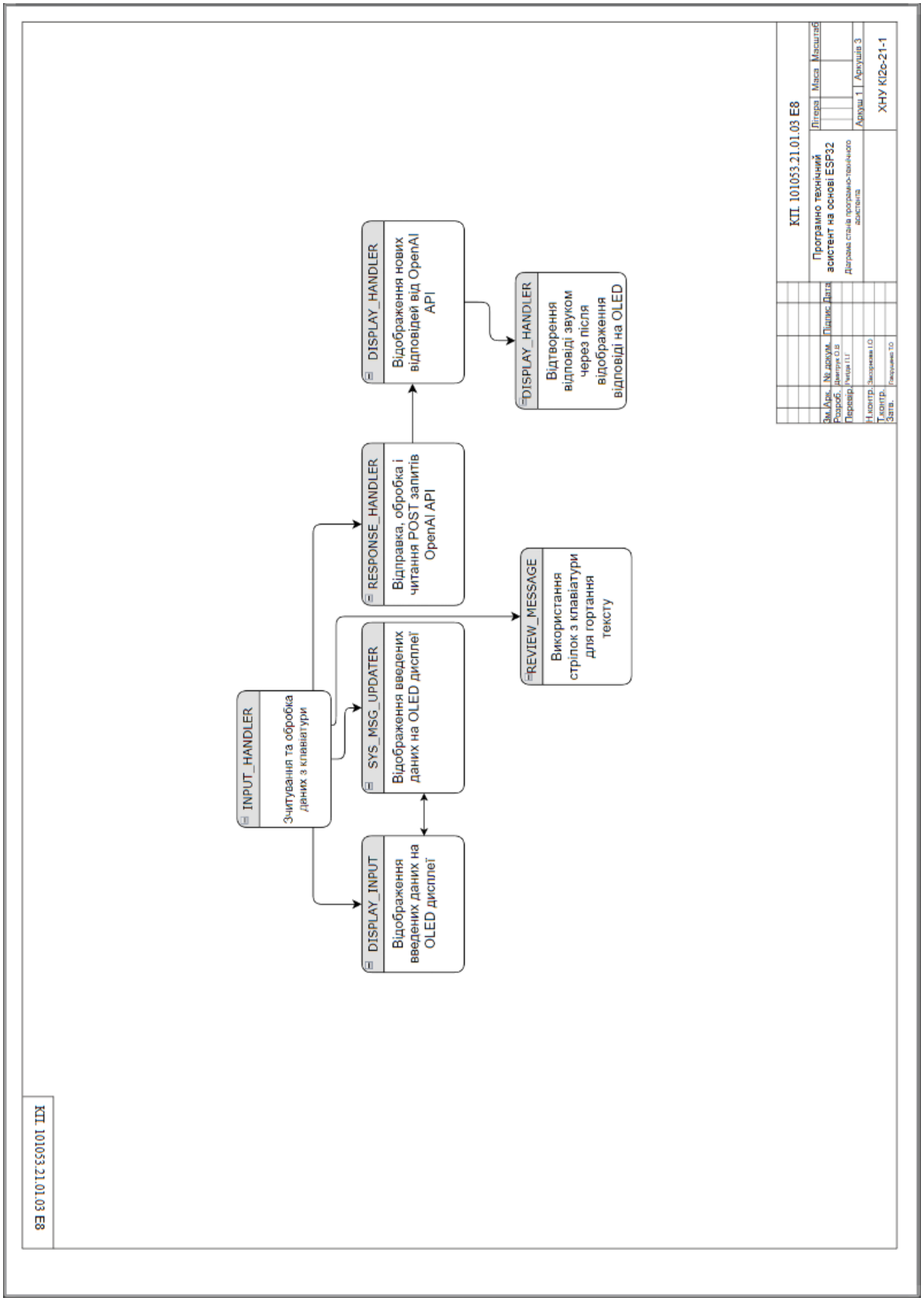
38. Тарасов І.Е., Гордєйчик С.В., Машинець В.Ю. Інтелектуальні системи підтримки прийняття рішень. Мінськ: Харвест, 2019. 221 с.
39. Гайдук А.Р., Петрова І.Ю., Шлюсар В.І. Мікроконтролери AVR: програмування та застосування. Київ: КПІ ім. Ігоря Сікорського, 2018. 147 с.
40. Лоренц М.О., Ткаченко В.В., Федосов В.П. Мікропроцесори та мікроконтролери: архітектура, програмування, застосування. Харків: ХНУ імені В.Н. Каразіна, 2021. 199 с.
41. Браницький І.І., Мельник В.А., Цьопа Н.В. Вбудовані системи на базі мікроконтролерів AVR. Київ: НАУ, 2019. 188 с.
42. Пінчук О.А., Нікітін Є.М., Шульженко О.І. Програмування мікроконтролерів ESP32. Київ: ТОВ Алерта, 2022. 259 с.
43. Васіна Л.Є., Караченцев В.І., Котельников В.Ю. Основи проектування систем на мікроконтролерах. Дніпро: ДВНЗ Національний гірничий університет, 2020. 243 с.
44. Демиденко О.М., Поліщук Л.М., Кутовий О.П. Вбудовані системи керування на базі мікроконтролерів. Суми: СумДУ, 2021. 332 с.
45. Козловський В.В., Перед'як С.С., Гуменюк С.В. Вбудовані системи на базі ARM мікроконтролерів. Львів: Львівської політехніки, 2020. 242с.
46. Балан М.П., Ляшенко В.І., Ганжела В.В. Мікроконтролери сімейства STM32 в прикладних застосуваннях. Київ: НУБіП України, 2019. 195с.
47. Івахненко О.В., Сніжний В.В., Зіноватна С.Л. Програмування мікроконтролерів Arduino для систем інтернету речей. Харків: ХНУРЕ, 2021. 273 с.
48. Костенко А.О., Караченцев В.І., Васіна Л.Є. Проектування вбудованих систем на базі мікроконтролерів ESP32. Дніпро: ДВНЗ Національний гірничий університет, 2022. 187 с.
49. Сидоренко В.В., Черкасов М.В., Бондаренко М.Є. Мікроконтролери в системах автоматизації та керування. Харків: Видавництво НТУ ХПІ, 2020. 355 с.

50. Грищук Ю.С., Зелік О.Г., Садовий П.І. Мікроконтролери в системах збору та обробки даних. Рівне: НУВГП, 2019. 142 с.
51. Плахотнюк В.М., Костенко В.Г., Глухов К.В. Програмування вбудованих систем на базі мікроконтролерів. Дніпро: НГУ, 2021.284 с.
52. Воробієнко П.П., Гусєв О.Ю., Терехов В.О. Проектування мікропроцесорних систем керування. Одеса: Видавництво Одеської політехніки, 2020. 219 с.
53. Яремчук В.В., Борисов В.В., Сенько В.І. Мікроконтролери в системах автоматики та телемеханіки. Київ: Видавництво КПІ ім. Ігоря Сікорського, 2019. 255 с.
54. Романов В.О., Козьміних С.В., Сліпченко Д.В. Створення вбудованих систем керування на базі мікроконтролерів. Київ: Видавництво НАУ, 2022. С. 110-138с.
55. Ковальчук А.М., Войтюк В.А., Гумен Т.Ф. Мікроконтролери в системах автоматизації технологічних процесів. Львів: Видавництво НУ "Львівська політехніка", 2020. 244 с.
56. Горбійчук М.І., Осадчий В.В., Клименко І.А. Проектування систем на мікроконтролерах: від ідеї до реалізації. Одеса: Видавництво ОНУ імені І.І. Мечникова, 2021. 256 с.
57. Борисов О.А., Жихарєв В.В., Руденко В.С. Мікроконтролерні пристрої в системах збору та передачі даних. Харків: Видавництво НТУ "ХПІ", 2019. 233 с.
58. Дакі О.А., Перенчук М.М., Рогоза В.В. Застосування мікроконтролерів у бездротових сенсорних мережах. Чернівці: Видавництво ЧНУ імені Юрія Федьковича, 2020. 256 с.
59. Лісовський О.П., Вінничук М.С., Черепанська І.Ю. Мікроконтролери та їх програмування для Інтернету речей. Луцьк: Видавництво СНУ імені Лесі Українки, 2022. 201 с.
60. Савчук В.П., Ларін В.Ю., Браницький І.І. Мікроконтролери в системах управління рухомими об'єктами. Київ: Видавництво НАУ, 2021. 311 с.

**Додаток А**

(обов’язковий)

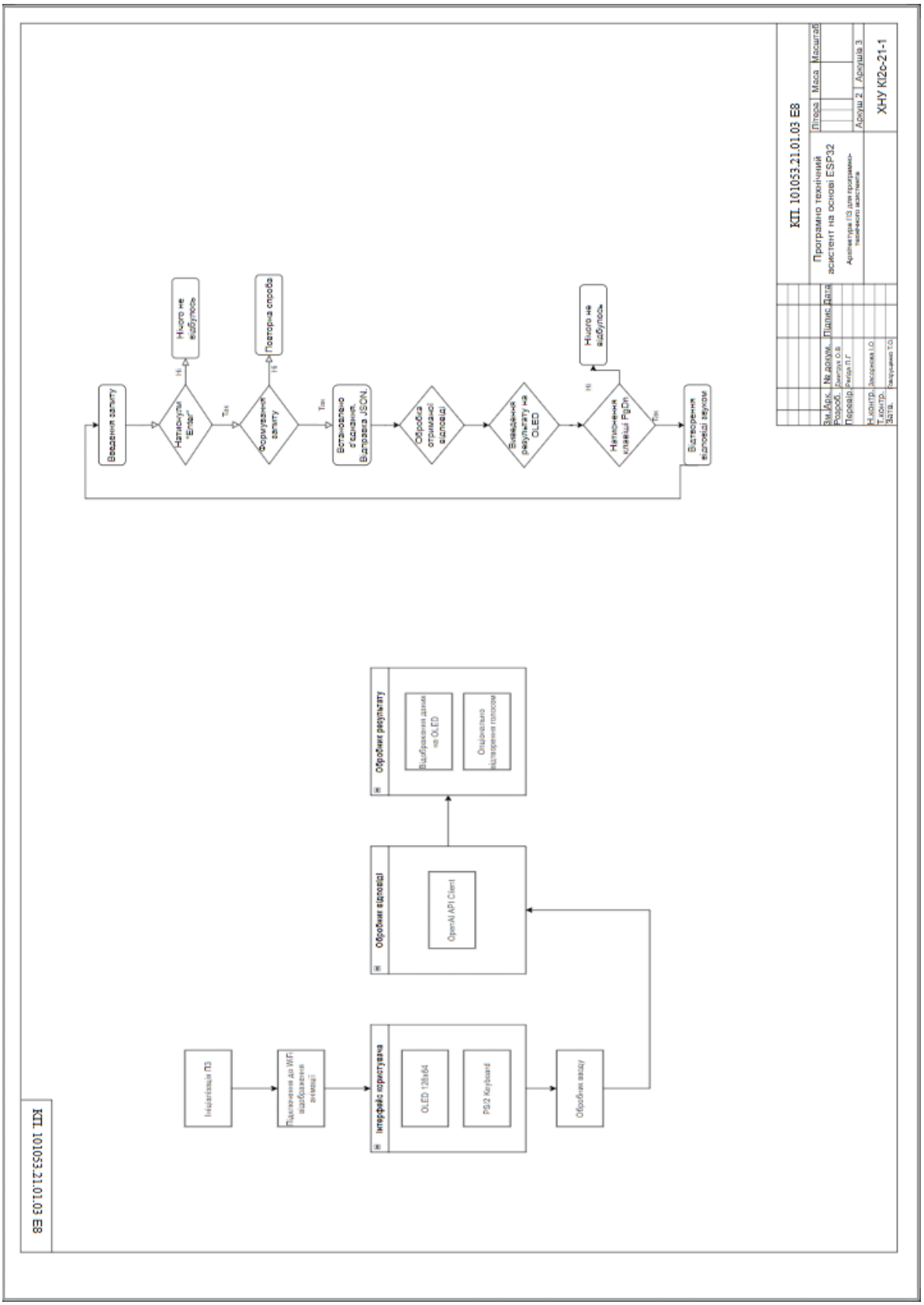
Копія креслення «Діаграма станів програмно-технічного асистента»



**Додаток Б**

(обов’язковий)

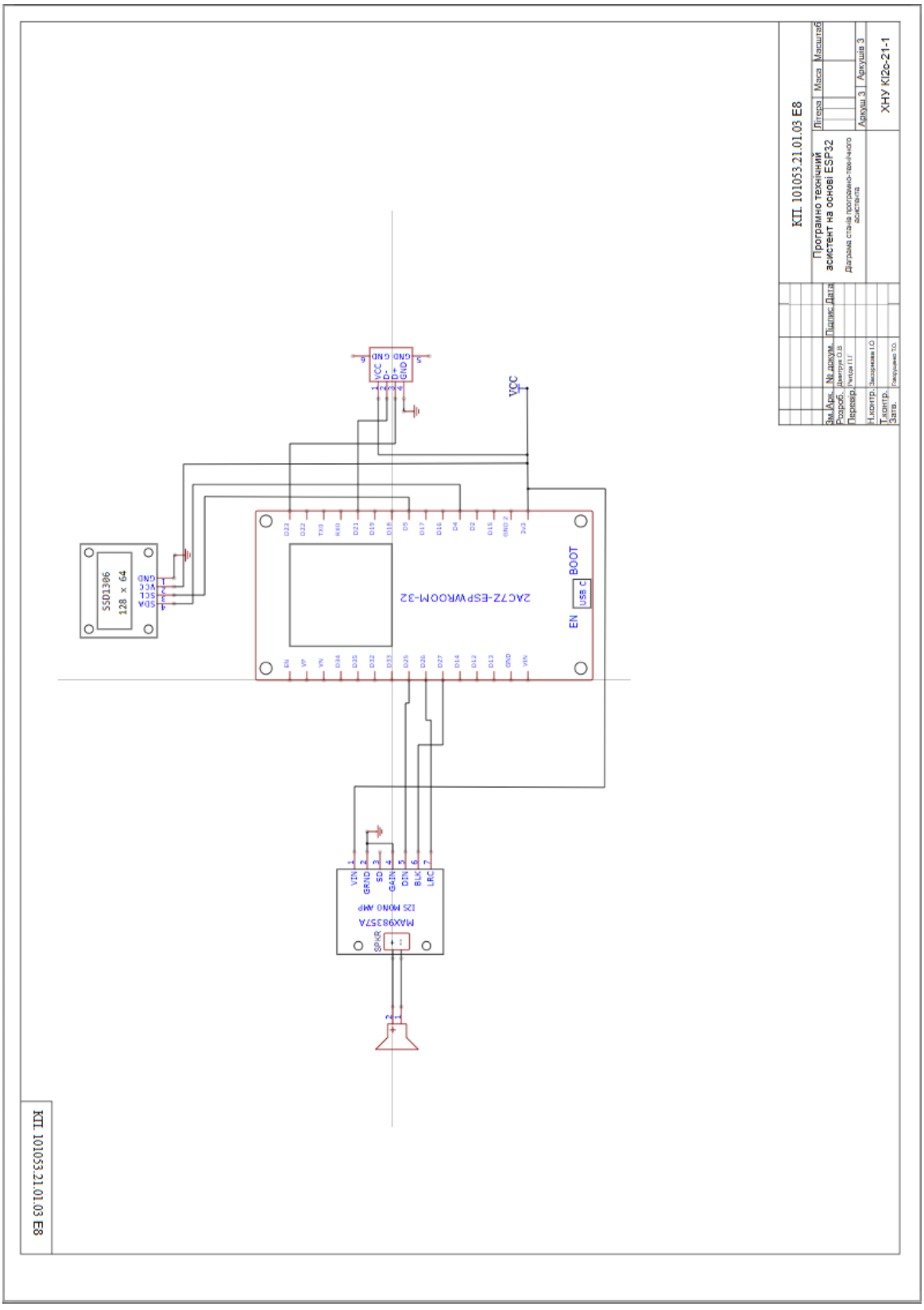
Копія креслення «Архітектура ПЗ для програмно-технічного асистента»



**Додаток В**

(обов’язковий)

Копія креслення «Принципова схема програмно-технічного асистента на основі ESP32»



**Додаток Г**

Код програми

```
#include <Arduino.h>
#include "Audio.h"
#include <U8g2lib.h>
#include <WiFi.h>
#include <ArduinoJson.h>
#include <PS2KeyAdvanced.h>
#include <PS2KeyMap.h>
#include "stateVars.h"
#include "credentials.h"
#include "certificates.h"
#include "bitmaps.h"
#include "config.h"
Audio audio;

PS2KeyAdvanced keyboard;
PS2KeyMap keymap;

void displayAnimation(long displayTime, const char displayMessage[], long delayInterval = 100) {

  long iterations = displayTime / delayInterval;
  int marginTop = -10;
  int marginBottom = 5;

  for (int i = 0; i < epd_bitmap_allArray_LEN; i++) {
    u8g2.clearBuffer();
```

```
            u8g2.drawXBM(0, marginTop, SCREEN_WIDTH, SCREEN_HEIGHT,
epd_bitmap_allArray[i]);
            u8g2.setCursor(0, SCREEN_HEIGHT - marginBottom);
            u8g2.print(displayMessage);
            u8g2.sendBuffer();
            delay(delayInterval);
          }
        }
        void displayMsg(char msg[], int endIdx, int startIdx = 0, bool setDelay = false) {

          u8g2.clearBuffer();
          u8g2.setCursor(0, 0);
          int lineNum = 0;
          bool firstTime = true;
          int i, count;

          for (i = startIdx, count = 1; i < endIdx; i++, count++) {
            if (i % MAX_CHAR_PER_OLED_ROW == 0) {
              lineNum++;
              u8g2.setCursor(0, FONT_HEIGHT * lineNum);
            }
            u8g2.print(msg[i]);
            if ((msg[i] == ' ' && setDelay)) {
              if (firstTime || lineNum == MAX_OLED_ROWS) {
                delay(300);
                u8g2.sendBuffer();
              }
            }
            if ((count != 0) && ((count % MAX_CHARS_ON_SCREEN) == 0)) {
```

```
        u8g2.clearBuffer();
        i -= MAX_CHARS_ON_SCREEN - MAX_CHAR_PER_OLED_ROW;
        lineNum = 0;
        firstTime = false;
      }
     }

     u8g2.sendBuffer();
    }


    void displayResponse(struct StateVars* pStateVars) {

     if (pStateVars->state == DISPLAY_RESPONSE || pStateVars->state ==
REVIEW_RESPONSE) {

      Serial.println("|- Print Response ------------------------------|");
      int responseIdx = (pStateVars->msgCount % MAX_MESSAGES) - 1 < 0
                  ? MAX_MESSAGES - 1
                  : pStateVars->msgCount % MAX_MESSAGES - 1;
      int startIdx;
      int endIdx;
      if (pStateVars->state == DISPLAY_RESPONSE) {

       startIdx = 0;
       endIdx = responseLength;
       pStateVars->displayOffset = 0;
      } else if (pStateVars->state == REVIEW_RESPONSE) {
```

```
            byte spacesToCompleteLastRow = MAX_CHAR_PER_OLED_ROW -
responseLength % MAX_CHAR_PER_OLED_ROW;
            int fullRowsOfText = (responseLength + spacesToCompleteLastRow) /
MAX_CHAR_PER_OLED_ROW;
            int endFrameLastIdx = (fullRowsOfText *
MAX_CHAR_PER_OLED_ROW);
            int endFrameFirstIdx = endFrameLastIdx - MAX_CHARS_ON_SCREEN;
            int scrubAdj = pStateVars->displayOffset *
MAX_CHAR_PER_OLED_ROW;
            startIdx = endFrameFirstIdx + scrubAdj;
            if (startIdx < 0) {
              startIdx = 0;
              (pStateVars->displayOffset)++;
            }
            endIdx = startIdx + MAX_CHARS_ON_SCREEN - 1;
          }
          displayMsg(messagesOut[responseIdx].content, endIdx, startIdx, pStateVars-
>state == DISPLAY_RESPONSE ? true : false);
          pStateVars->state = GET_USER_INPUT;
        }
      }
      void printToConsoleMessageArray() {

        Serial.println("    |------------------- Messages[] --------------------");

        for (int i = 0; i < MAX_MESSAGES; i++) {
          Serial.print("    | ");
          Serial.print(i);
          Serial.print(" - ");
```

```
    Serial.println(messagesOut[i].content);
  }
  Serial.print("    | Size of most recent reponse -> ");
  Serial.println(responseLength);
  Serial.println("    |-------------------------------------------------");
}
DynamicJsonDocument generateJsonRequestBody(int numMessages) {
  DynamicJsonDocument doc(DYNAMIC_JSON_DOC_SERIALIZE_SIZE);
  doc["model"] = "gpt-4o";
  doc["max_tokens"] = MAX_TOKENS;
  JsonArray messagesJSON = doc.createNestedArray("messages");
  int oldestMsgIdx = 0;
  if (numMessages >= MAX_MESSAGES) {
    oldestMsgIdx = numMessages % MAX_MESSAGES;
  }
  for (int i = 0; i < numMessages && i < MAX_MESSAGES; i++) {
    if (i == numMessages - 1 || i == MAX_MESSAGES - 1) {
      messagesJSON[i]["role"] = roleNames[systemMessage.role];
      messagesJSON[i]["content"] = systemMessage.content;
      i++;
    }
    messagesJSON[i]["role"] = roleNames[messagesOut[oldestMsgIdx].role];
    messagesJSON[i]["content"] = messagesOut[oldestMsgIdx].content;
    oldestMsgIdx++;
    oldestMsgIdx %= MAX_MESSAGES;
  }
  return doc;
}
```

```
        void postRequest(DynamicJsonDocument* pJsonRequestBody,
WiFiClientSecure* pClient) {

          Serial.println("    | Connected to OpenAI");
          pClient->println("POST https://api.openai.com/v1/chat/completions
HTTP/1.1");
          pClient->print("Host: ");
          pClient->println(server);
          pClient->println("Content-Type: application/json");
          pClient->print("Content-Length: ");
          pClient->println(measureJson(*pJsonRequestBody));
          pClient->print("Authorization: Bearer ");
          pClient->println(openAI_Private_key);
          pClient->println("Connection: Close");
          pClient->println();
          serializeJson(*pJsonRequestBody, *pClient);
          pClient->println();
          Serial.println("    | JSON sent");
        }

        bool putResponseInMsgArray(WiFiClientSecure* pClient, int numMessages) {
          String response = pClient->readString();
          int jsonStartIndex = response.indexOf('{');
          int jsonEndIndex = response.lastIndexOf('}');
          if (jsonStartIndex == -1 || jsonEndIndex == -1) {
            Serial.println("Failed to find JSON start or end");
            return 0;
          }
```

```
        String jsonResponseString = response.substring(jsonStartIndex, jsonEndIndex +
1);
        Serial.println("Extracted JSON:");
        Serial.println(jsonResponseString);
        StaticJsonDocument<2000> jsonResponse;
        DeserializationError error = deserializeJson(jsonResponse, jsonResponseString);
        if (error) {
          Serial.print("deserializeJson() failed: ");
          Serial.println(error.c_str());
          return 0;
        }
        if (!jsonResponse.containsKey("choices") ||
!jsonResponse["choices"][0].containsKey("message") ||
!jsonResponse["choices"][0]["message"].containsKey("content")) {
          Serial.println("Required keys not found in JSON response");
          return 0;
        }
        const char* content = jsonResponse["choices"][0]["message"]["content"];
        Serial.print("Received content: ");
        Serial.println(content);
        messagesOut[numMessages % MAX_MESSAGES].role = assistant;
        String messageContent = String(content);
        messageContent.toCharArray(messagesOut[numMessages %
MAX_MESSAGES].content, MAX_MESSAGE_LENGTH);
        messagesOut[numMessages %
MAX_MESSAGES].content[MAX_MESSAGE_LENGTH - 1] = '\0';
        responseLength =
measureJson(jsonResponse["choices"][0]["message"]["content"]);
        return 1;
```

```
}
void getResponse(struct StateVars* pStateVars) {

  if (pStateVars->state == GET_RESPONSE) {

    Serial.println("|- Start API Call ------------------------------|");
    Serial.print("    | msgCount->");
    Serial.println(pStateVars->msgCount);
    WiFiClientSecure client;
    client.setCACert(rootCACertificate);

    DynamicJsonDocument jsonRequestBody =
generateJsonRequestBody(pStateVars->msgCount);

    int conn = client.connect(server, PORT);

    if (conn == 1) {
      postRequest(&jsonRequestBody, &client);
      bool responseSuccess = waitForServerResponse(&client);
      if (responseSuccess) {
        bool responseSaved = putResponseInMsgArray(&client, pStateVars-
>msgCount);
        if (responseSaved) {
          (pStateVars->msgCount)++;
          pStateVars->state = DISPLAY_RESPONSE;
        } else {
          return;
        }
      } else {
```

```
            displayAnimation(API_RESPONSE_FAIL_INTERVAL,
ApiResponseFailMsg);
            Serial.println("    | Server did not respond. Trying again.");
           }
          } else {
           displayAnimation(SERVER_CONNECTION_FAIL_INTERVAL,
ServerConnectionFailMsg);
           Serial.println("    | Could not connect to server. Trying again.");
          }
          client.stop();
         }
        }

        void getUserInput(struct StateVars* pStateVars) {

         if (keyboard.available() && (pStateVars->state == GET_USER_INPUT ||
pStateVars->state == UPDATE_SYS_MSG)) {
          int key = keyboard.read();
          byte base = key & 0xff;
          byte remappedKey = keymap.remapKey(key);

          if (base == PS2_KEY_PGDN) {
          if (pStateVars->inputIdx == 0) {
              int responseIdx = (pStateVars->msgCount % MAX_MESSAGES) - 1 < 0
                        ? MAX_MESSAGES - 1
                        : pStateVars->msgCount % MAX_MESSAGES - 1;
             audio.connecttospeech(messagesOut[responseIdx].content, "en");
             }
          }
```

```
        boolean arrowKeyPressed = (base == PS2_KEY_UP_ARROW) || (base ==
PS2_KEY_DN_ARROW);
        if (remappedKey > 0 || arrowKeyPressed) {
          pStateVars->bufferChange = true;
          switch (base) {
            case PS2_KEY_ENTER:
              Serial.println("KeyPressed-> Enter");
              if (pStateVars->state == GET_USER_INPUT) {

                if (pStateVars->inputIdx != 0) {
                  pStateVars->msgPtr->role = user;
                  pStateVars->msgCount++;
                  pStateVars->inputIdx = 0;
                  pStateVars->state = GET_RESPONSE;
                } else {

                  u8g2.clearDisplay();
                }

                Serial.println("  | User message submitted");

              } else if (pStateVars->state == UPDATE_SYS_MSG) {

                pStateVars->msgPtr->role = sys;
                displayMsg((char*)SystemMsgUpdateSuccessAlert,
ALERT_MSG_LENGTH);

                pStateVars->state = GET_USER_INPUT;
```

```
    pStateVars->bufferChange = false;
    Serial.println(systemMessage.content);
    Serial.println("  | System message updated.");
   }

   pStateVars->clearInput = true;
   Serial.print("    | state-> ");
   Serial.println(pStateVars->state);
   break;

  case PS2_KEY_DELETE:

   pStateVars->inputIdx = pStateVars->inputIdx > 0 ? pStateVars->inputIdx -
1 : 0;
   pStateVars->msgPtr->content[pStateVars->inputIdx] = ' ';

   Serial.println("KeyPressed-> Backspace/Delete ");
   Serial.print("  | Input Index->");
   Serial.println(pStateVars->inputIdx);
   break;

  case PS2_KEY_SPACE:

   if (pStateVars->inputIdx < MAX_MESSAGE_LENGTH - 1) {
    pStateVars->msgPtr->content[pStateVars->inputIdx] = ' ';
    pStateVars->inputIdx++;
   }

   Serial.println("KeyPressed-> Space/Tab");
```

```
  Serial.print("  | Input Index->");
  Serial.println(pStateVars->inputIdx);
  break;
case PS2_KEY_UP_ARROW:

  if (pStateVars->inputIdx == 0) {
    pStateVars->displayOffset--;
    pStateVars->state = REVIEW_RESPONSE;
  }

  Serial.println("KeyPressed-> Up Arrow");
  Serial.print("  | displayOffset->");
  Serial.println(pStateVars->displayOffset);
  break;
case PS2_KEY_DN_ARROW:

  if (pStateVars->inputIdx == 0) {
    pStateVars->displayOffset++;

    if (pStateVars->displayOffset > 0) {
      pStateVars->displayOffset = 0;
    }

    pStateVars->state = REVIEW_RESPONSE;
  }

  Serial.println("KeyPressed-> Down Arrow");
  Serial.print("  | displayOffset->");
  Serial.println(pStateVars->displayOffset);
```

```
            break;

          default:

            if (pStateVars->clearInput) {

              memset(pStateVars->msgPtr->content, 0, MAX_MESSAGE_LENGTH);

              pStateVars->inputIdx = 0;
              pStateVars->clearInput = false;
              Serial.println("  | Message cleared.");
            }

            if (pStateVars->inputIdx < MAX_MESSAGE_LENGTH - 1) {
              pStateVars->msgPtr->content[pStateVars->inputIdx] =
char(remappedKey);
              pStateVars->inputIdx++;
              Serial.print(char(remappedKey));
            } else if (pStateVars->inputIdx == MAX_MESSAGE_LENGTH - 1) {
              pStateVars->msgPtr->content[pStateVars->inputIdx] = '<';
              pStateVars->inputIdx++;
              Serial.print("  | You've Reached the end of Input Index->");
              Serial.println(pStateVars->inputIdx);
            }
          }
        }
      }
    }
```

```
void displayKeyboardInput(struct StateVars* pStateVars) {

  if (pStateVars->bufferChange && (pStateVars->state == GET_USER_INPUT ||
pStateVars->state == UPDATE_SYS_MSG)) {
    displayMsg(pStateVars->msgPtr->content, pStateVars->inputIdx);
  }
}

void setup(void) {

  Serial.begin(9600);

  delay(3000);
  Serial.println(" runpy21 AI Tools.");

  Serial.print("WiFi connected to IP address: ");
  Serial.println("Setup Started...");

  keyboard.begin(DATAPIN, IRQPIN);
  keyboard.setNoBreak(1);
  keyboard.setNoRepeat(1);
  keymap.selectMap((char*)"US");

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println(WiFi.localIP());
  u8g2.begin();
  u8g2.setFont(u8g2_font_6x13_tf);

  displayAnimation(BOOT_ALERT_INTERVAL, BootScreenMsg);
  displayMsg((char*)WelcomeInstructionsAlert, ALERT_MSG_LENGTH);
  audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
  audio.setVolume(100);
  Serial.println("...Setup Ended");
}
void loop(void) {
  audio.loop();
  static StateVars stateVars = {
    GET_USER_INPUT,
    0, 0, false, 0 };
  stateVars.msgPtr = (stateVars.state == GET_USER_INPUT)
                     ? &messagesOut[stateVars.msgCount % MAX_MESSAGES]
                     : &systemMessage;

  stateVars.bufferChange = false;
  getUserInput(&stateVars);
  getResponse(&stateVars);
 displayKeyboardInput(&stateVars);
  displayResponse(&stateVars);
}
```

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016300545

Дата перевірки:
30.05.2024 20:38:04 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
30.05.2024 21:18:04 EEST

ID користувача:
100005591

Назва документа: Дмитрук_Програмно-технічний асистент для взаємодії із мовною моделлю на основі

Кількість сторінок: 74 Кількість слів: 13497 Кількість символів: 102965 Розмір файлу: 8.25 MB ID файлу: 1016096004

# 1.49%
# Схожість

Найбільша схожість: 0.46% з Інтернет-джерелом (http://elar.khmnu.edu.ua/jspui/bitstream/123456789/14009/1/%d0%9d.

1.24% Джерела з Інтернету 96 ..... Сторінка 76

0.86% Джерела з Бібліотеки 105 ..... Сторінка 76

# 0% Цитат

Не знайдено жодних цитат

Посилання 1 ..... Сторінка 76

# 0%
# Вилучень

Немає вилучених джерел

# Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 1

# Anti-Plagiarism v-15.257

**Максимальне співпадіння з одним документом 0.0%**

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

| | Документ | | Сумарний збіг по Базі Даних | |
|---|---|---|---|---|
| ID: 127908<br>Назва: БКР Програмно-технічний асистент для взаємодії із мовною моделлю на основі<br>Додано в БД: 2024-05-30<br>Автора: О. В. Дмитрук<br>Керівники: П. Г. Регіда<br>Консультанти:<br>Опоненти: | Символи | Лексеми | Символи | Лексеми |
| | 88979 | 740 | 1325 (1%) | 17 (2%) |

Джерело плагіату

| ID | Опис | Наявність плагіату в документі | |
|---|---|---|---|
| | | Символи | Лексеми |

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

# РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Дмитрук Олександр Валентинович

Тема: Програмно-технічний асистент для взаємодії із мовною моделлю на основі ESP32

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 64

1. Короткий зміст роботи та прийнятих рішень: метою кваліфікаційної роботи є розробка програмно-технічного асистента для взаємодії з мовною моделлю за на основі мікроконтролера ESP32.

2. Висновок про відповідність роботи дипломному завданню: робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області, огляд уже існуючих рішень у вигляді реалізованих пристроїв. Розглянуто переваги та недоліки наявних варіантів взаємодії з мовними моделями. Виконано постановку задачі дослідження. В другому розділі проведено обґрунтування використання мікроконтролера ESP32 як основи для розробки пристрою, розглянуто його технічні характеристики. Також здійснено вибір периферійних пристроїв, зокрема OLED дисплея SSD1306 та аудіопідсилювача MAX98357. В третьому розділі кваліфікаційної роботи виконано апаратну реалізацію пристрою, описано процес підготовки до роботи, ініціалізацію компонентів та послідовність дій користувача під час взаємодії з програмно-технічним асистентом. Детально розглянуто процеси введення запитів за допомогою клавіатури, відправки їх до OpenAI через WiFi з'єднання, отримання та відображення відповідей на OLED дисплеї та озвучування голосом через аудіопідсилювач. Наведено приклади роботи з пристроєм та наведено його функціонування в різних сценаріях використання.

4. Позитивні сторони роботи: Висока практична цінність роботи, що полягає у забезпечення використання мовних моделей для користувачів.

5. Негативні сторони роботи: Розроблений пристрій підтримує тільки англомовні запити до мовної моделі.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: Розглянувши позитивні та негативні сторони представленої дипломної роботи вважаю, що робота заслуговує оцінки «відмінно», 5.0 (A).

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____
Мартинюк Валерій Володимирович
зав. каф. АКІТ ге. Р

"5" 06 2024 р. _____ (підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Дмитрука Олександра Валентиновича
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

3 червня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Програмно-технічний асистент для взаємодії із мовною моделлю на основі ESP32

Автор: Дмитрук Олександр Валентинович

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Регіда Павло Геннадійович, старший викладач

Після аналізу звіту подібності зроблено такий висновок:

| № | Висновок | Позначка про відповідність |
|---|---|---|
| 1 | Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту. | відповідає |
| 2 | Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи | |

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 1.49% і адресується до 201 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи П. Г. Регіда

Гарант ОП С.М. Лисенко

Завідувач кафедри КІІС Т. О. Говорущенко