Міністерство освіти і науки України

Чернівецький національний університету імені Юрія Федьковича»

Відокремлений структурний підрозділ «Фаховий коледж
Чернівецького національного університету імені Юрія Федьковича»

Природниче відділення

Циклова комісія комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітньо-професійного ступеня «фаховий молодший бакалавр»

зі спеціальності *123 Комп'ютерна інженерія*

підготовки за освітньо-професійною програмою «*Комп'ютерна інженерія*»

на тему: «**Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32»**

Виконав (ла):
студент (ка) 4-го курсу *Кирчул Костянтин Костянтинович* .
(прізвище, ім'я та по батькові) (підпис)

Керівник:
Тащук О.Ю. .
(науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент:
Букурос О.В .
(науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

До захисту допущено:

Протокол засідання циклової комісії № _

від „___"_____ 2024 р.

Голова циклової комісії _____ Олександр ТАЩУК

Чернівці, 2024

Завдання та календарний план роботи

Міністерство освіти і науки України
Чернівецький національний університету імені Юрія Федьковича»
Відокремлений структурний підрозділ «Фаховий коледж
Чернівецького національного університету імені Юрія Федьковича»

Затверджую
Голова циклової комісії,
Олександр ТАЩУК.
_____
“____”_____ 2024 р.

Завдання

На кваліфікаційну роботу	Кирчул Костянтин Костянтинович
Тема кваліфікаційної роботи	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32
Постановка завдання в короткій формі	Створення апаратно-програмної реалізації Wi-Fi радіо на основі ESP32.
Вихідні дані (2-3 адреси сайтів, матеріали яких рекомендує керівник магістерської роботи)	https://github.com/karawin/Ka-Radio32 https://www.espressif.com/en/products/sdks/esp-idf.

Календарний план роботи

Дата отримання версії звіту керівником	Етап виконання роботи	Виконання (зазначає керівник)
10.10.23	Отримання завдання до кваліфікаційної роботи.	
16.10.23	Огляд літератури, присвяченої розробці систем Wi-fi радіо.	
01.11.23	Аналіз існуючих конструкторських рішень. Оформлення розділу 1.	
05.11.23	Вибір компонентної бази, узгодження модулів системи.	
28.11.23	Написання теоретичного матеріалу по використовуваних модулях.	
06.12.23	Проектування та розробка апаратної частини, підключення Оформлення розділу 2.	

25.03.24	Проектування та розробка програмної частини, підключення. Оформлення розділу 3.	
20.05.24	Оформлення та редагування кваліфікаційної роботи.	
31.05.24	Подання роботи на перевірку Інтернет-сервісом Unicheck.	
03.06.24	Подання роботи на рецензію	
13.06.24	Представлення роботи на засіданні Циклової комісії	
за графіком	Захист кваліфікаційної роботи	

Дата видачі завдання 10. 10. 2023 _____.

Студент Костянтин КИРЧУЛ

Керівник (П.І.Б) Олександр ТАЩУК

АНОТАЦІЯ

Кирчул Костянтин Костянтинович Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32 Кваліфікаційна робота.

В роботі виконано моделювання та розробку Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32.

На сьогоднішній день тема Wi-Fi радіо залишається дуже актуальною, особливо у зв'язку з поширенням бездротових мереж та їх важливістю для спілкування та передачі даних. Wi-Fi є надзвичайно важливою технологією бездротового з'єднання, що широко використовується в різних сферах: від домашнього використання до комерційних та громадських мереж. Вона надає зручний спосіб доступу до Інтернету та спільного використання мережевих ресурсів, зменшуючи необхідність у фізичних з'єднаннях. Останні роки свідчать про постійний прогрес у розвитку Wi-Fi технологій, що викликано постійним зростанням потреб у бездротовому зв'язку та підвищенням вимог до швидкості передачі даних.

Робота складається з 4 розділів, в яких проаналізовано роботу, описано та виконано розробку Wi-Fi радіо на основі ESP32.

Кваліфікаційна робота містить 72 сторінок, 7 рисунків та 10 посилань на літературні джерела.

Ключові слова: Wi-fi радіо, радіо, ESP32.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Wi-Fi – Wireless Fidelity

МК – мікроконтролер

USB – Universal Serial Bus – універсальна послідовна шина

ISSP – In Circuit Serial Programming – коло послідовного програмування

ДДП – Двостороння друкована плата

ДП – Друкована плата

ЕРЕ – Електрорадіоелемент

РЕС – Радіоелектронні системи

ТЗ – технічне завдання

ПЗ – Програмне забезпечення

ВСТУП

На даний момент тема Wi-Fi радіо залишається актуальною, особливо в контексті бездротових мереж і зв'язку. Wi-Fi є технологією бездротового мережевого з'єднання, яка широко використовується в домашніх, комерційних та громадських мережах. Вона забезпечує зручний спосіб доступу до Інтернету і спільного використання ресурсів мережі без необхідності фізичних з'єднань. Останні роки свідчать про постійний розвиток технологій Wi-Fi, що зумовлено постійним зростанням потреб у бездротовому зв'язку та швидкості передачі даних.

Метою даної роботи є розробка та впровадження пристрою для бездротового відтворення радіопотоку через Wi-Fi. В ході роботи використано мікроконтролер ESP32 та додаткові компоненти для створення функціонального пристрою.

Об’єктом дослідження є розробка апаратно-програмного пристрою для бездротового відтворення радіо потоку через Wi-Fi.

Предметом дослідження є розробка конструкції та програмного забезпечення для Wi-Fi радіо.

Мета роботи: розробка та впровадження бездротового пристрою для відтворення радіопотоку через Wi-Fi на базі мікроконтролера ESP32. Для досягнення цієї мети необхідно виконати наступні задачі:

- Вивчення можливостей та аналіз різних компонентів для побудови пристрою.

- Проектування та збірка пристрою на базі ESP32.

- Розробка програмного забезпечення для бездротового зчитування та відтворення радіопотоку через Wi-Fi.

- Тестування пристрою та аналіз його функціональності.

Методи дослідження. Дослідження ґрунтується на методиках аналізу та проектування електронних пристроїв, програмування мікроконтролерів, а також використанні засобів розробки для мікроконтролерів ESP32. Також

використовуються методи тестування та апробації пристрою для підтвердження його працездатності та ефективності.

Практична цінність отриманих результатів полягає в можливості створення ефективного та функціонального пристрою для бездротового відтворення радіопотоку через Wi-Fi з використанням доступних компонентів та технологій. Такий пристрій може бути використаний у різних сферах, включаючи домашнє аудіо, офісні системи та інші області, де потрібна зручна та ефективна система відтворення аудіопотоку.

1. ПОСТАНОВКА ЗАДАЧІ

Wi-Fi радіо - це пристрій, який використовує бездротову технологію для отримання радіо сигналів через мережу Wi-Fi. Ця технологія дозволяє слухачам насолоджуватися доступом до тисяч радіостанцій з усього світу без необхідності використання традиційних радіоприймачів.

Однією з основних переваг Wi-Fi радіо є його можливість отримувати доступ до різних радіостанцій, включаючи місцеві станції, станції з інших країн та спеціалізовані інтернет-радіостанції, які працюють в різних жанрах та форматах. Користувачі можуть слухати музику, новини, підкасти, радіодрами та інші аудіо контенти, що транслюються через інтернет.

Типовий Wi-Fi радіо складається з антени для прийому Wi-Fi сигналів, процесора для обробки даних, вбудованої пам'яті для зберігання налаштувань та інших даних, а також аудіо-виходу для підключення навушників або аудіо-системи. Крім того, деякі моделі можуть мати інші функції, такі як можливість запису програм, секундоміри, таймери сну та інші.

Однією з переваг Wi-Fi радіо є його зручність. Користувачі можуть вибирати радіостанції та контент безпосередньо на пристрої за допомогою комп’ютера або телефона та використовуючи потенціометр.

Wi-Fi радіо може бути більш економічно ефективним порівняно з традиційними радіоприймачами, оскільки воно не потребує використання антен та інших апаратних засобів для отримання радіосигналу, велика частина контенту доступна через інтернет, користувачам не потрібно купувати або завантажувати музичні файли окремо.

Загалом, Wi-Fi радіо є потужним та зручним засобом отримання доступу до різноманітного аудіо контенту з усього світу, і воно продовжує набирати популярність серед слухачів, які цінують різноманітність та зручність цього способу отримання радіо.

1.1 Огляд існуючих систем бездротового зчитування та відтворення радіопотоку через Wi-Fi.

На даний момент системи Wi-Fi радіо залишаються важливим компонентом сучасного зв'язку та передачі даних. Wi-Fi технології постійно розвиваються, пристосовуючись до потреб користувачів і стандартів безпеки, що змінюються. Auna Connect 100 Wi-Fi BT USB - це компактний інтернет-радіоприймач з бездротовими можливостями, який поєднує в собі багато функцій та високу якість звуку Рис 1.1.

Рисунок. 1.1. Auna Connect 100 Wi-Fi BT USB

Основні можливості:

1. Інтернет-приймач: За допомогою Wi-Fi можна отримати доступ до понад 10 000 радіостанцій і потокових музичних архівів.

2. Bluetooth і USB: Підтримка Bluetooth для відтворення музики з смартфонів, планшетів або комп'ютерів. USB-порт для відтворення музики з USB-накопичувачів та MP3-плеєрів.

3. Розширений будильник: Програмований двоканальний будильник з регульованим періодом пробудження і таймером сну, що дозволяє вибирати сигнал будильника через звук, мелодію або інтернет-радіо.

4. Кольоровий TFT-дисплей: Дисплей з діагоналлю 6,5 см (2,5 дюйма) та яскравим фоновим підсвічуванням, що відображає час та іншу інформацію.

5. Дерев'яний корпус: Компактний дерев'яний корпус з відображенням басів і передня панель з полірованого алюмінію та чорного дерева.

Бездротові функції:

1. Мережа: Підтримка стандарту 802.11 b/g/n для бездротового з'єднання.

2. Потокове відтворення: Підтримка UPnP та DLNA для потокової передачі аудіо.

3. Профілі шифрування: Підтримка різних профілів шифрування, включаючи WEP, WPA, WPA2 (PSK) та WPS.

Характеристики входів/виходів:

- Входи: 1 x USB-порт, 1 x 3,5 мм роз'єм AUX.

- Виходи: 1 x 3,5 мм лінійний вихід.

Мультимедійні можливості:

- Підтримувані формати: MP3, WMA, WAV.

Комплектація та розміри:

- Комплектація: 1 x пристрій, 1 x блок живлення, 1 x пульт дистанційного керування.

- Розміри: 22 x 12 x 13.5 см (ШхВхГ).

- Вага: 1,6 кг.

- Колір: Чорний.

Wifi-радіо Majority Peterhouse Graduate Radio є інноваційним пристроєм для отримання доступу до радіо контенту через Інтернет. Вироблений в Німеччині, цей пристрій відзначається своїми передовими технологіями та розширеними можливостями для користувачів Рис. 1.2.

Рисунок. 1.2. Majority Peterhouse Graduate Radio

Характеристики Majority Peterhouse Graduate Radio

1. Розміри: Приміри 30,5 x 13,8 x 13,6 см, що робить його компактним і зручним для розміщення в будь-якому приміщенні.

2. Дисплей: Кольоровий ЖК-дисплей з можливістю диммірування, який надає зручне і чітке відображення інформації про радіостанції, час, а також інші налаштування.

3. Підтримка аудіоформатів: Програвач підтримує широкий спектр аудіоформатів, включаючи MP3, WMA, WAV, FLAC, AAC+, що забезпечує сумісність з різними типами аудіофайлів.

4. Підтримка інтернет-радіо: Підключення до безлічі радіостанцій з усього світу через Wi-Fi, що дозволяє користувачам насолоджуватися різноманітністю контенту.

5. Функція будильника: Можливість налаштування двох будильників з індивідуальними режимами і налаштуванням днів тижня, коли вони активні.

6. Підтримка зовнішніх пристроїв: AUX-вхід і USB-порт для підключення зовнішніх пристроїв, таких як смартфони або USB-накопичувачі, для програвання музики або заряджання.

7. Bluetooth: Можливість підключення до інших пристроїв за допомогою Bluetooth, що розширює можливості відтворення музики зі смартфонів чи планшетів.

8. Функції повтору і таймера: Можливість встановлення повтору для додаткового сну та таймера для автоматичного вимкнення пристрою.

9. Погодна інформація: Можливість відображення погодної інформації на дисплеї, яка надає додаткову корисну інформацію користувачеві.

10. Живлення: Живлення від мережі 12 В постійного струму зі струмом до 2 А, що забезпечує стабільну роботу пристрою.

Wifi радіо TuneUp OneConcept - це сучасний пристрій, який дозволяє користувачам насолоджуватися музикою та радіостанціями з усього світу. Він пропонує широкі можливості підключення, зручне управління та високоякісний звук, що робить його ідеальним вибором для будь-якого музичного ентузіаста Рис 1.2.

Рисунок. 1.2. TuneUp OneConcept

Основні особливості:

1. Необмежене насолодження музикою: TuneUp дозволяє отримати доступ до безлічі музичних станцій з усього світу через інтернет-радіо.

2. Чіткий дисплей: Високоякісний дисплей HCC з діагоналлю 2,4 дюйма забезпечує чітке відображення інформації з високою контрастністю та яскравістю кольорів.

3. Зручне управління: Управління пристроєм можливе через додаток AirMusic, що забезпечує зручну навігацію та управління вмістом.

5. Потужний звук: Вбудовані динаміки мають потужність 5 Вт, що забезпечує якісне звучання музики.

Характеристики:

- Порти підключення: USB, лінійний вихід (3,5 мм)
- Мережевий адаптер: WLAN 802.11 b/g/n
- Підтримувані формати файлів: WMA, MP3
- Підтримка потокової передачі: UPnP, DLNA
- Вихідна потужність: 5 В / 1 А
- Електроживлення: 100-240 В ~ | 50/60 Гц

Габаритні розміри:

- Розміри: приблизно 19,4 х 11,2 х 9,8 см (ШхВхГ)
- Довжина кабелю живлення: 160 см
- Вага: приблизно 0,5 кг

1.2. Аналіз технічного завдання

Проект має на меті створення компактного, легкого у використанні та стабільного Wi-Fi радіо на базі мікроконтролера ESP32. Функціональні вимоги, такі як реалізація Wi-Fi підключення для потокового відтворення музики з Інтернет-радіостанцій, керування радіоприймачем через потенціометр та можливість відтворення аудіо через динамік. Обмеження включають використання ESP32 для його потужності та можливостей Wi-Fi, використання підсилювача PAM8403 для покращення якості звуку, інтеграцію потенціометр для зручного керування пристроєм та дисплей для показу радіостанцій. Вимоги до продукту включають компактність, простоту використання та стабільність роботи. Аналіз цих аспектів дозволить розробити технічне завдання, яке визначить всі необхідні функції та характеристики для успішної реалізації Wi-Fi радіо на базі ESP32 з урахуванням вимог до продукту та обмежень.

2.ПРОЕКТУВАННЯ АПАРАТНОЇ ЧАСТИНИ

2.1 Структурна та електрична принципова схеми модуля Wi-Fi радіо

На рисунку 2.1 представлена структурна схема модуля Wi-Fi радіо, який побудований на апаратній платформі ESP32.

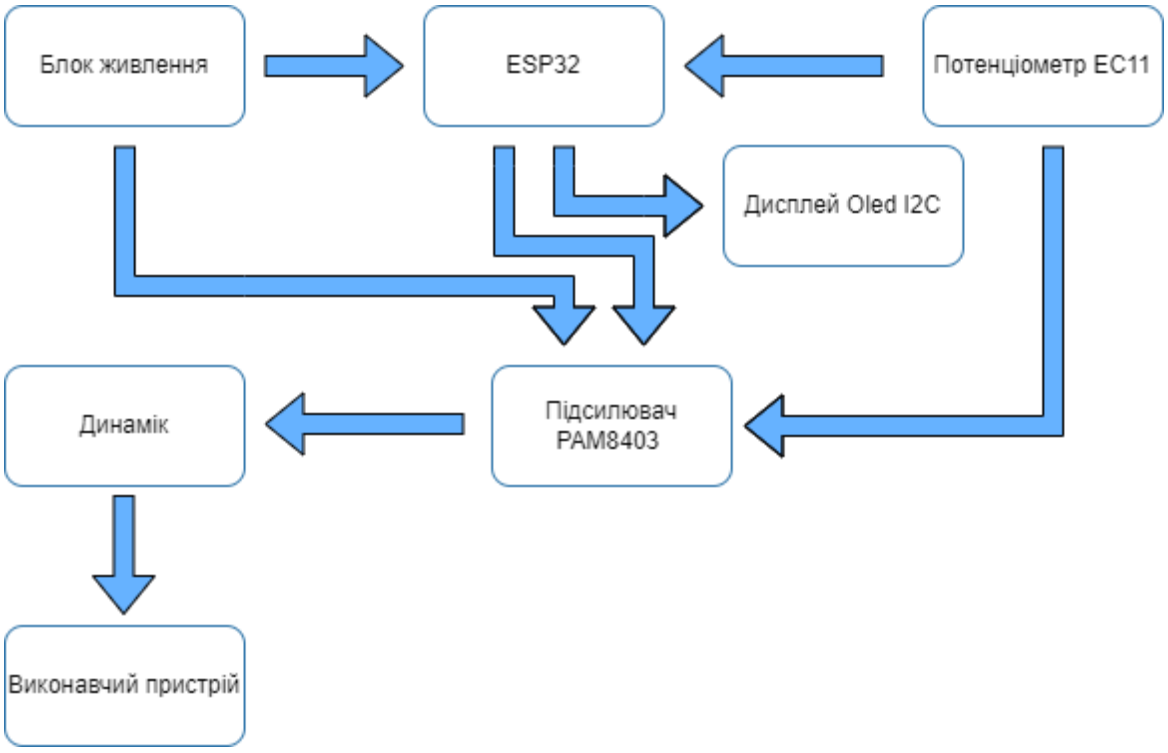


Рисунок. 2.1. Структурна схема модуля Wi-Fi радіо

Основним елементом є апаратна платформа ESP32, побудована на базі мікроконтролера Tensilica Xtensa LX6. Цей мікроконтролер дозволяє програмувати та керувати пристроєм. З блоку живлення подається напруга 5 В на ESP32 та підсилювач PAM8403. ESP32 через Wi-Fi отримує дані з радіопотоку. Мікроконтролер аналізує отримані дані, передає їх на дисплей та на підсилювач, а він, у свою чергу, передає їх на динамік для відтворення звуку.

Отже, після створення структурної схеми пристрою можна переходити до проектування електричної схеми модуля Wi-Fi радіо.

Для створення схеми модуля Wi-Fi радіо було використанро програмне забезпечення KiCad EDA, що є зручним інструментом для розробки схем електричних кол та печатних плат.

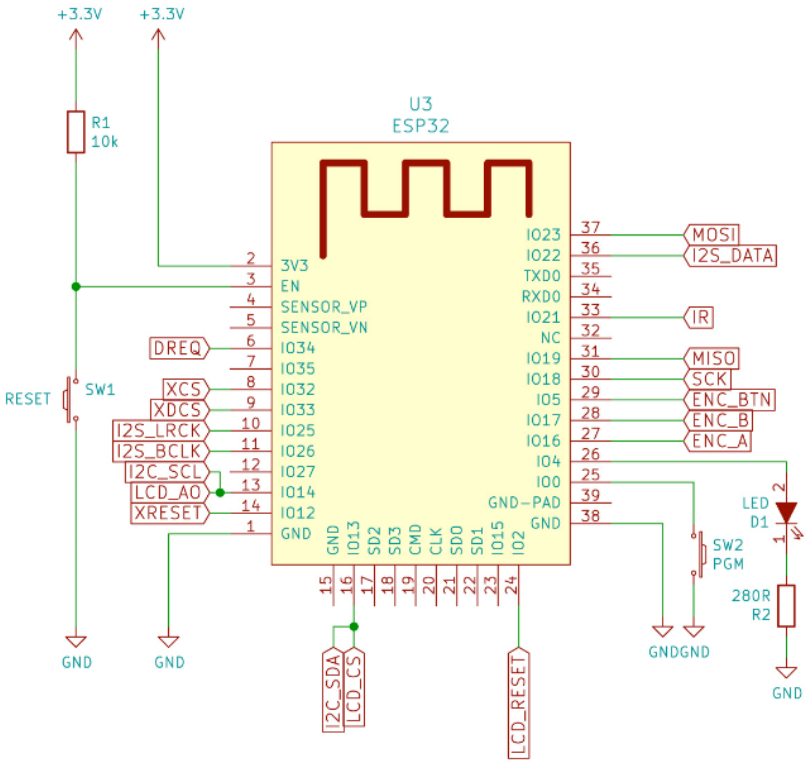


Рисунок. 2.2.

ESP32 (U3) - мікроконтролер, який обробляє всі функції радіо, включаючи підключення до Wi-Fi, декодування аудіо та керування інтерфейсом.

Потенціометр (ENC) - використовується для навігації по меню радіо та регулювання гучності.

Дисплей (LCD) - використовується для відображення інформації про стан радіо, наприклад, назву станції, частоту та гучність.

Інші компоненти - резистори, конденсатори, індуктори та інші компоненти, необхідні для роботи схеми.

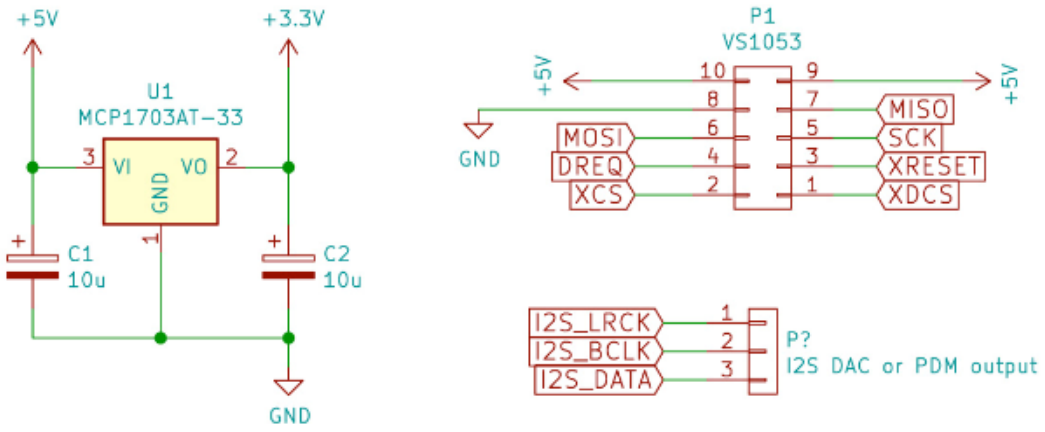


Рисунок. 2.2.

VS1053 (U2) - ЦАП (цифро-аналоговий перетворювач), який перетворює цифрові аудіодані з ESP32 в аналоговий сигнал, який можна подати на динамік.

MCP1703AT-33 (U3) - регулятор напруги, який забезпечує стабільне живлення 3,3 В для ESP32 та VS1053.

C1 та C2 - конденсатори, які використовуються для фільтрації напруги живлення.

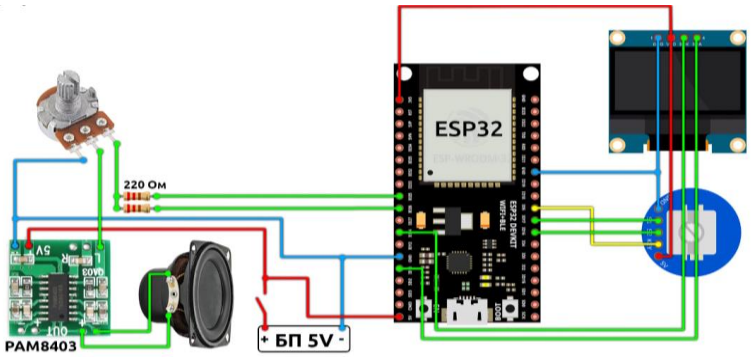
R1 та R2 - резистори, які використовуються для підтягування ліній даних SPI.

X1 та X2 - кварцові резонатори, які забезпечують тактову частоту для ESP32 та VS1053.

Антена - використовується для підключення до Wi-Fi мережі.

Динамик - використовується для відтворення звуку.

Інші компоненти - резистори, конденсатори, індуктори та інші компоненти, необхідні для роботи схеми.



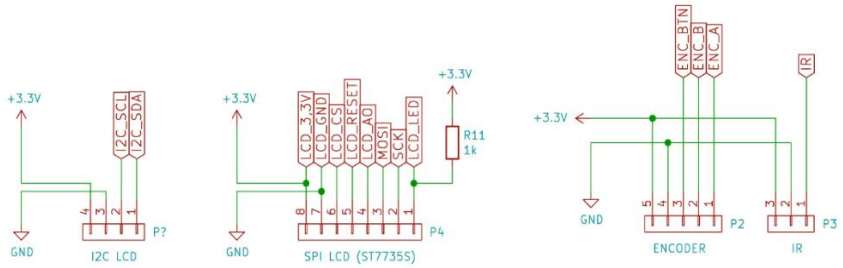


Рисунок. 2.2.

Дисплей: РК-дисплей, який використовується для відображення інформації про радіо, наприклад, назву станції, частоту та гучність він підключений до ESP32 через SPI-інтерфейс.

Потенціометр: використовується для керування радіо, наприклад, для перемикання станцій або регулювання гучності, підключається до ESP32 через аналогові входи.

2.2 Вибір компонент

2.2.1 Апаратна платформа ESP32

Вибір ESP32 для проекту Wi-Fi радіо обумовлений кількома ключовими факторами. По-перше, ESP32 має вбудований Wi-Fi модуль, що робить його ідеальним для проектів, які потребують бездротового з'єднання з Інтернетом. Також ESP32 має достатньо потужний процесор та об'єм оперативної пам'яті, що дозволяє обробляти аудіо дані та виконувати інші завдання, пов'язані з відтворенням музики та роботою з Wi-Fi. ESP32 також підтримує широкий спектр інтегрованих середовищ розробки, таких як Arduino IDE та MicroPython, що робить програмування зручним та доступним. Крім того, ESP32 має багато різноманітних периферійних пристроїв і може взаємодіяти з різними сенсорами, аудіо підсистемами та іншими пристроями, що дозволяє розширити функціональність проекту. Неостаннім фактором є вартість ESP32, яка є досить доступною порівняно з його можливостями, що робить його привабливим вибором для проектів з обмеженим бюджетом. Узагальнюючи, ESP32 відповідає всім вимогам проекту Wi-Fi радіо, надаючи потужність, функціональність та доступність, що робить його оптимальним вибором для реалізації цього проекту Рис. 2.2.

а) б)

Рисунок. 2.2 Зовнішній вигляд ESP32: спереду (а) і ззаду (б)

Таблиця 2.1 Характеристики апаратної платформи ESP32.

Параметр	**Значення**
Мікроконтролер	Tensilica Xtensa LX6
Робоча напруга	3.3 В
Напруга живлення	2.2-3.6 В
Wi-Fi Стандарти:	FCC/CE/IC/TELEC/KCC/SRRC/NCC
Протоколи	802.11 b/g/n/d/e/i/k/r (802.11n до 150 Мбіт/с)
Частотний діапазон:	ГГц 2.4 ~ 2.5
Bluetooth протоколи:	Bluetooth v4.2 BR/EDR та BLE specification
Flash-пам'ять	4МБ
SRAM	520 КБ
EEPROM	448 КБ
Тактова частота	240 МГц
Габаритні розміри	25.2 x 18 мм

ESP32 - це мікроконтролер, що базується на архітектурі Xtensa LX6, розроблений компанією Espressif Systems. Основними характеристиками ESP32 є вбудований двоядерний процесор, підтримка Wi-Fi і Bluetooth, а також різноманітні периферійні пристрої, такі як GPIO, SPI, I2C, UART та інші.

ESP32 має ряд пінів, які можуть використовуватися для підключення різноманітних пристроїв і датчиків:

1. GPIO (General Purpose Input/Output) - Ці піни можна налаштовувати як вхідні або вихідні, що дозволяє зчитувати стан датчиків або керувати зовнішніми пристроями.

2. UART (Universal Asynchronous Receiver/Transmitter) - Дозволяє забезпечити зв'язок з іншими пристроями через UART, такі як серійні порти.

3. SPI (Serial Peripheral Interface) - Ці піни використовуються для обміну даними між ESP32 та іншими пристроями через SPI.

4. I2C (Inter-Integrated Circuit) - Дозволяє підключати пристрої до ESP32 через шину I2C для обміну даними.

5. ADC (Analog-to-Digital Converter) - Ці піни використовуються для зчитування аналогових значень, таких як напруга з датчиків або потенціометрів.

6. DAC (Digital-to-Analog Converter) - Дозволяє генерувати аналогові сигнали з цифрових значень.

7. PWM (Pulse Width Modulation) - Ці піни використовуються для генерування ШШУ сигналів, що дозволяє керувати яскравістю світлодіодів, швидкістю моторів тощо.

Ці піни у ESP32 забезпечують широкі можливості для підключення різних пристроїв і реалізації різноманітних проектів у галузі вбудованих систем, Інтернету речей (IoT) та інших застосувань.

Однією з ключових можливостей ESP32 є його можливість безпровідного зв'язку. Вбудований Wi-Fi дозволяє підключатися до мережі Інтернет, створювати точки доступу, взаємодіяти з веб-службами та виконувати різноманітні завдання, пов'язані з мережевою комунікацією. Більше того, наявність Bluetooth відкриває широкі можливості для взаємодії з іншими Bluetooth-пристроями, такими як смартфони, датчики та інші периферійні пристрої.

ESP32 також підтримує різні інтерфейси вводу-виводу (GPIO), аналогові та цифрові перетворювачі, що дозволяє йому взаємодіяти з різноманітними

сенсорами, актуаторами та іншими пристроями. Це робить його ідеальним вибором для створення різних пристроїв зв'язку та контролю.

Оскільки ESP32 має невеликий розмір, він легко інтегрується в різноманітні пристрої та системи. Його можна використовувати як автономний мікроконтролер або як частину більш складних систем, що забезпечує гнучкість у проектуванні та розробці.

Що стосується програмування, ESP32 підтримує різні середовища розробки, включаючи Arduino IDE, MicroPython, PlatformIO та інші. Це дає можливість вибирати оптимальне середовище в залежності від ваших потреб та вподобань. Крім того, наявність великої спільноти користувачів та обширної документації сприяє швидкому вивченню та розробці з використанням ESP32.

ESP32 - це потужний та гнучкий мікроконтролер з великим набором можливостей для взаємодії з комп'ютером, іншими пристроями та Інтернетом що робить його ідеальним вибором для широкого спектру проектів, від простих датчиків до складних систем автоматизації та IoT.

2.2.2 Підсилювач PAM4803

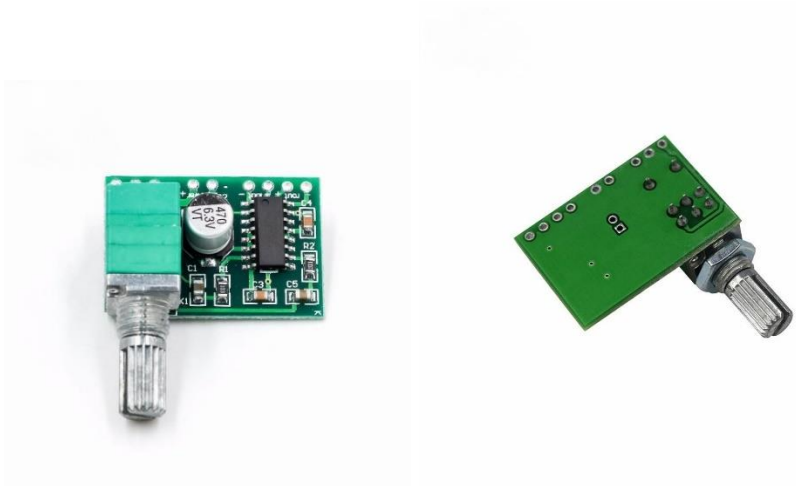
Підсилювач PAM8403 — це дуже популярний вибір для аудіо підсилення у різноманітних електронних системах, особливо з обмеженим простором і обмеженими ресурсами, таких як портативні аудіо пристрої, маленькі радіоприймачі та інші.

Підсилювач PAM8403 має дуже маленькі розміри, що дозволяє легко інтегрувати його в пристрої з обмеженим простором, такі як навушники, портативні колонки тощо. Незважаючи на це, він має високий коефіцієнт підсилення і ефективно відтворює звук, що робить його популярним вибором для багатьох застосувань.

PAM8403 відомий своєю низькою споживаною потужністю, що робить його ідеальним вибором для пристроїв, які працюють від батареї або вимагають економії енергії.

Підсилювач має досить простий у використанні інтерфейс, що дозволяє легко інтегрувати його в будь-який проект без значного досвіду у роботі з електронікою.

PAM8403 є популярним вибором завдяки своїм низьким вартості, компактності, ефективності, енергоефективності та простоті використання, що робить його ідеальним рішенням для багатьох аудіо-пристроїв Рис 2.3.



а) б)

Рисунок. 2.3 Зовнішній вигляд PAM8403: спереду (а) і ззаду (б)

Таблиця 2.2 Характеристики підсилювача PAM8403

Параметр	Значення
Клас підсилювача:	D
Вихідна потужність:	2 x 3Вт
Частотний діапазон:	20 Гц – 20 кГц
Опір навантаження:	від 4Ом до 8Ом
Напруга живлення:	від 2.5В до 5В
Гранична напруга живлення:	5.5В
Розміри:	18.5 x 21.1 мм

PAM8403 - це потужний підсилювач, що робить його ідеальним для обмеженого простору. Він відноситься до класу D підсилювачів, які відомі своєю високою ефективністю (до 90%) та мінімальним тепловиділенням. Це означає, що він може видавати чистий звук без перегріву, навіть за тривалої роботи.

PAM8403 - це стереопідсилювач, тож він може керувати двома окремими каналами, забезпечуючи справжнє стереозвучання. Кожен канал може

видавати до 3 Вт потужності, чого цілком достатньо для більшості невеликих приміщень. Варто зазначити, що ця потужність вимірюється за рівня спотворень 10% THD+N. THD+N (Total Harmonic Distortion + Noise) - це співвідношення бажаного сигналу до небажаного шуму та спотворень. Чим нижче цей показник, тим чистіший звук.

Підсилювач може працювати з широким діапазоном джерел живлення, оскільки він працює від постійного струму з напругою від 2.5 до 5.5 В. Це робить його придатним для використання з батареями або USB-джерелами живлення, що ідеально підходить для портативних проектів. Він також сумісний з більшістю динаміків, оскільки може працювати з навантаженням від 4 до 8 Ом.

Ще однією перевагою PAM8403 є те, що він оснащений захистом від короткого замикання та перегріву. Це захищає підсилювач від пошкоджень у разі несправностей або перевантажень. Крім того, він має вражаюче співвідношення сигнал/шум 80 дБ, що означає, що він видає чистий звук із мінімальним рівнем шуму на виході.

Підсумовуючи, PAM8403 - це універсальний підсилювач, який поєднує в собі компактний розмір, високу ефективність, потужність та захисні функції. Він ідеально підходить для використання в портативних колонках, DIY-проектах з підсиленням звуку та багатьох інших застосунках, де потрібен якісний звук за мінімальних розмірів.

2.2.3 Дисплей Oled I2C display 128x64 0.96”

Дисплей OLED (Organic Light Emitting Diode) з інтерфейсом I2C розміром 128x64 пікселів та діагоналлю 0.96 дюйма - це популярний компонент, що широко використовується в проектах електроніки та робототехніки. Він дозволяє відображати текст, графіку та інші візуальні елементи з високою якістю.

Основні переваги цього дисплея включають високу якість зображення завдяки можливості кожного пікселя світитися незалежно, низьке споживання енергії, що особливо важливо для пристроїв на батарейках, широкий кут огляду, легку інтеграцію завдяки інтерфейсу I2C та компактні розміри, що роблять його ідеальним для вбудованих систем з обмеженим простором.

Дисплей OLED I2C 128x64 0.96” може знайти застосування в різних проектах, включаючи мобільні пристрої, пристрої з відстеженням даних, IoT пристрої та багато інших, надаючи можливість виводу різноманітної інформації для користувачів Рис. 2.4.

а) б)

Рисунок. 2.4 Зовнішній вигляд дисплею Oled I2C display 128x64 0.96: спереду (а) і ззаду (б)

Таблиця 2.3 Характеристики дисплею Oled I2C display 128x64 0.96”

Розмір екрану:	0.96 "
Тип екрану:	OLED
Драйвер OLED модуля:	SSD1306
Роз'єм:	4-pin
Напруга живлення:	+3.3~ +6 В
Роздільна здатність дисплея:	128 * 64 пк
Розмір:	4 x 27 x 28 мм
Вага :	5 г

OLED дисплей з розміром екрану 0.96 дюйма представляє собою компактний і економічний засіб візуалізації інформації для різноманітних пристроїв та систем. Він оснащений технологією OLED, що забезпечує

високий контраст зображення та яскравість, що робить його ідеальним вибором для використання навіть у яскравому освітленні.

Завдяки роздільній здатності дисплея 128x64 пікселів, користувач може чітко відображати текст, графіку та символи. Кут огляду понад 160 градусів дозволяє зручно спостерігати за вмістом екрану з будь-якого кута.

Для зручного підключення до мікроконтролера або міні-комп'ютера використовується поширений інтерфейс зв'язку I2C, що дозволяє легко і швидко інтегрувати дисплей в будь-яку систему. Зазначена адреса 0x3c спрощує процес взаємодії з дисплеєм через шину I2C.

Цей OLED дисплей оснащений драйвером SSD1306, що забезпечує стабільну та ефективну роботу пристрою. Живлення дисплея здійснюється в діапазоні від +3.3V до +6V, що робить його сумісним з різними джерелами живлення.

Загальні розміри дисплея (товщина 4 мм, ширина 27 мм, висота 28 мм) роблять його досить компактним і легким для використання в різних пристроях та системах. Роз'єм з 4 контактами (VCC, GND, SCL, SDA) спрощує підключення дисплея до джерела живлення та мікроконтролера.

Загалом, цей OLED дисплей є відмінним вибором для тих, хто шукає компактний, ефективний і контрастний дисплей з широким спектром застосувань у різних проектах та пристроях.

2.2.4 Потенціометр EC11

Потенціометр EC11 – це один із типів обертового потенціометра, який використовується для виміру кутового зміщення або кількості обертів. Він зазвичай складається з двох основних компонентів: внутрішнього датчика з гвинтовим робочим колесом та зовнішнього корпусу, що містить контактні елементи. Коли вал потенціометра обертається, внутрішній датчик реєструє

зміни положення та передає відповідний сигнал на контактні елементи, що виводяться на зовнішній корпус.

Потенціометр EC11 зазвичай мають два типи виходів: імпульсний (або так званий "квадратурний") вихід, який дозволяє вимірювати напрямок руху та кількість обертів, і тактильний вихід, який генерує сигнал при натисканні на Рухому частину потенціометра (якщо така функція підтримується). Ці потенціометри широко використовуються в електронних пристроях для забезпечення точного та зручного управління, таких як аудіо та відео пристрої, робототехніка, 3D-принтери, а також в промислових пристроях і автомобілях Рис. 2.5.

b) б)

Рисунок. 2.5 Зовнішній вигляд потенціометра EC11: спереду (а) і ззаду (б)

Таблиця 2.4 Характеристики потенціометра EC11

Довжина обертової частини:	12мм
Діаметр обертової частини:	6мм
Тип:	інкрементальний, оптичний
Кількість імпульсів на оборот:	20
Функція натискання:	Да
Тип вихідного сигналу:	2-бітний код Грея
Напруга живлення:	5 В
Максимальний струм:	10 мА
Розмір:	13 x 11 x 25 мм

Потенціометр EC11 є невід'ємною частиною багатьох електронних пристроїв. Цей інкрементальний оптичний енкодер працює на постійному струмі напругою 5 В та має низьке споживання енергії - лише 10 мА. Він генерує 20 імпульсів за один повний оберт рухомої частини, дозволяючи точно виміряти рух.

Його оптичний принцип роботи забезпечує стійкість до шуму та вібрацій, що робить його надійним у різних умовах експлуатації. Потенціометр може обертатися як за годинниковою стрілкою, так і проти, забезпечуючи більшу універсальність у використанні. Крім того, він оснащений кнопкою, яка дозволяє виконувати додаткові функції, такі як введення даних або підтвердження вибору.

Вихідний сигнал потенціометра - це 2-бітний код Грея, що робить передачу даних більш стійкою до помилок. Це важливо, оскільки навіть невеликі помилки можуть призвести до неправильного тлумачення значень.

Щодо фізичних характеристик, потенціометр EC11 має компактні розміри - всього 13 x 11 x 25 мм, з невеликою вагою 5 грам. Це робить його ідеальним для використання в портативних пристроях. Крім того, він має стандартний діаметр вала 6 мм і довжину 12 мм, що робить його сумісним з багатьма іншими компонентами.

Загалом, потенціометр EC11 є надійним, ефективним та універсальним пристроєм з доступною ціною, який може задовольнити потреби вимірювання обертання або швидкості великої кількості проектів.

2.2.5 Вибір резисторів

Резистор - це елемент електричного кола, який зазвичай виглядає як маленький пристрій із двома виводами. Основне призначення резистора полягає в обмеженні або контролі потоку електричного струму в колі. Він володіє певним значенням електричного опору, що визначається його конструкцією та матеріалом, з якого він виготовлений.

У різних ситуаціях резистор може використовуватися для різних цілей. Наприклад, в електричних колах його можуть використовувати для зниження напруги, регулювання струму, поглинання електричної енергії або навіть як частина фільтрів для зменшення шуму чи випрямлення сигналів.

При виборі резисторів для конкретного проекту важливо враховувати ряд факторів. Наприклад, габаритні розміри резистора можуть впливати на його можливе розміщення в схемі. Електричний опір визначає, наскільки ефективно резистор буде обмежувати струм у колі. Надійність і стійкість до зовнішніх чинників також мають велике значення, особливо якщо проект передбачає роботу в екстремальних умовах або на довгій відстані від обслуговування.

Крім того, вартість резистора також є важливим фактором, особливо для великих проектів з великою кількістю компонентів. Тому вибір оптимальних резисторів для будь-якого проекту вимагає уважного аналізу всіх цих характеристик.

Резистор на 220 ом може бути використаний для забезпечення правильного рівня опору в певних частинах кола Wi-Fi радіо, що дозволить контролювати потік електричного струму та напруги.

3. НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ АПАРАТНО-ПРОГРАМНОГО МОДУЛЯ WI-FI РАДІО НА ОСНОВІ ESP32

Проєкт був зроблений на базі «Ka-radio 32», тому треба встановити програмне забезпечення на мікросхему.

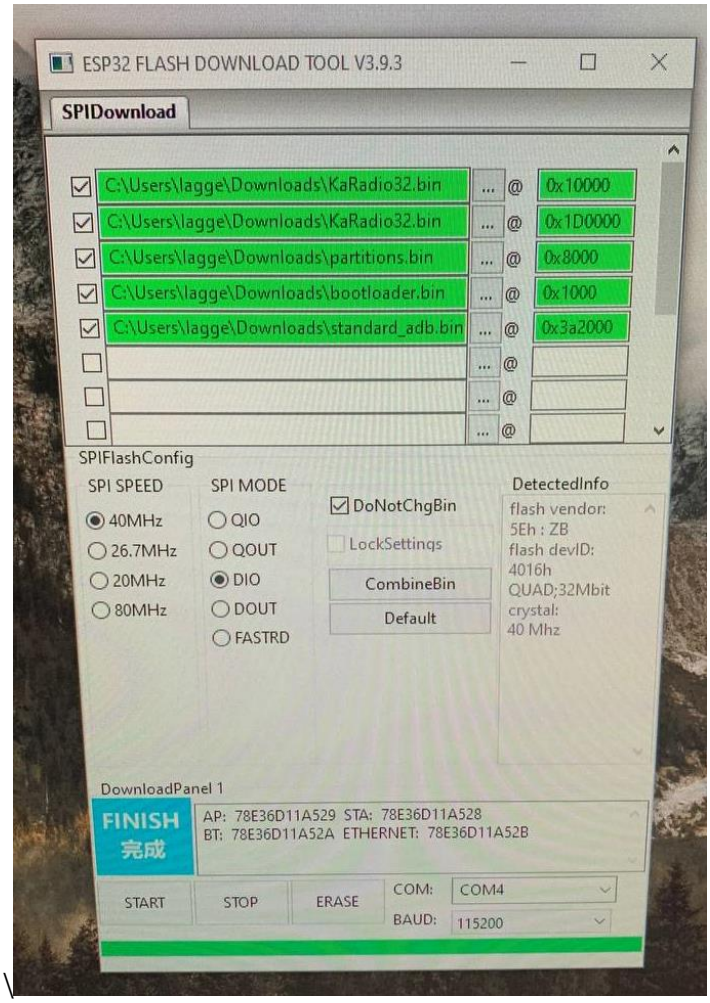
Посилання: Github «karadio32» - https://github.com/karawin/Ka-Radio32

Для встановлення програмного забезпечення на мікросхему потрібно встановити «Flash Download Tool».

Посилання: «Flash Download Tool»:

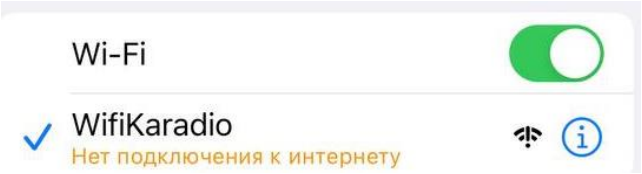
Підключаємо мікросхему до комп'ютера через кабель micro USB.

Відкриваємо «Flash Download Tool» і встановлюємо такі параметри як на фото.



Нажимаємо спочатку «Erase» і потім «Start». Очікуємо поки не з'явиться «Finish».

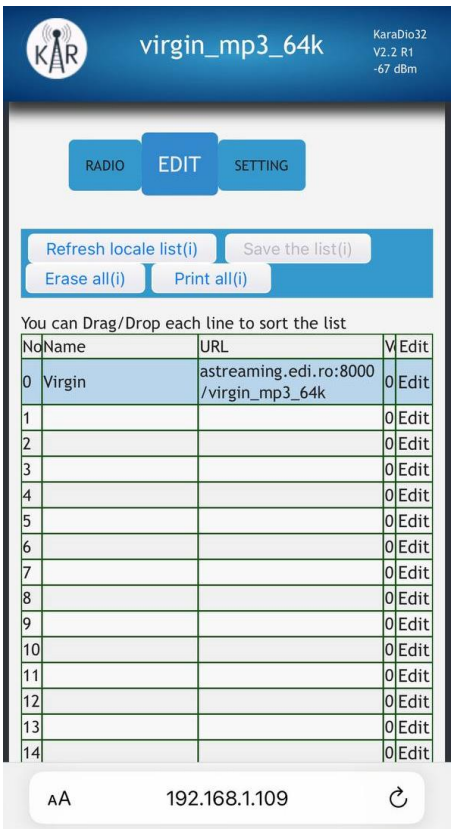
Після цього знаходимо на телефоні або комп’ютері мережу під назвою «WifiKaradio», підключаємось до неї.



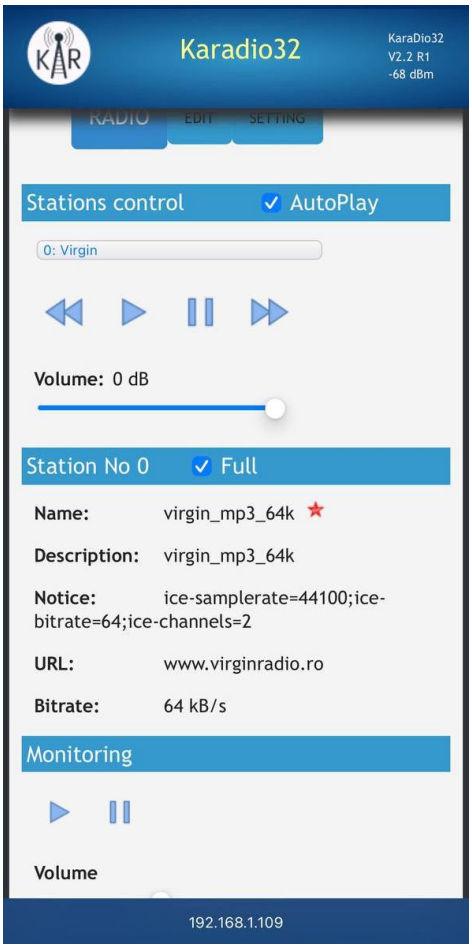
Через браузер вводимо 192.168.4.1, заходимо в Setting, вводимо в AP SSID назву Wifi до якого він має підключитись і AP PASSWORD пароль від Wifi.

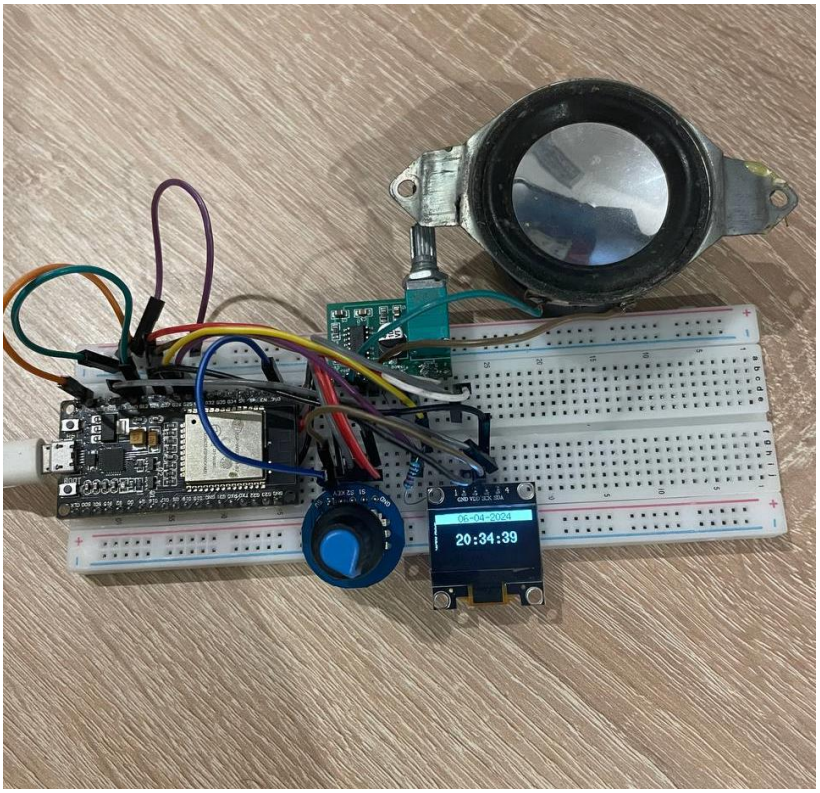
Після перезавантаження мікросхеми, підключаємось з телефону до Wifi яку підключили мікросхему, і вводимо а браузері 192.168.1.109. Заходимо в edit і добавляємо радіо станцію.

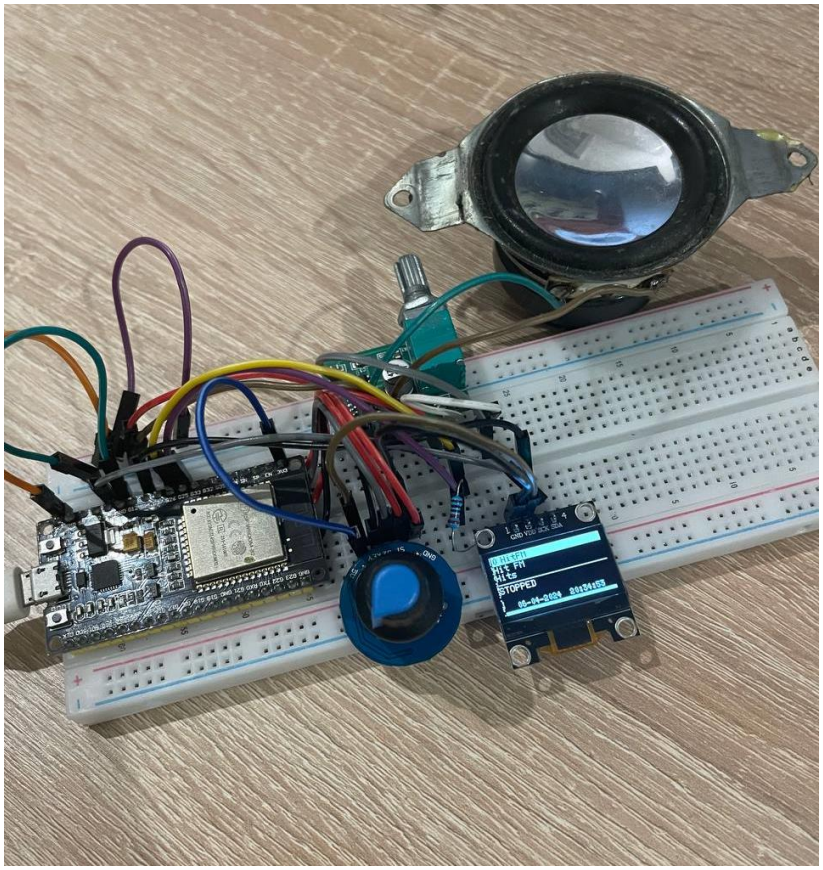
Посилання на список станцій: https://espradio.ru/stream_list/



Після цього треба зайти в Radio і запустити за допомогою кнопки Play.




06-04-2024
20:34:39



ВИСНОВКИ

У результаті виконання роботи та практичної реалізації було успішно створено Wi-Fi радіо на базі мікроконтролера ESP32.

Аналіз сучасних можливостей мікроконтролерів показав, що ESP32 відповідає вимогам щодо швидкості передачі даних та має вбудований Wi-Fi модуль, що робить його ідеальним вибором для реалізації радіо-пристроїв.

Під час дослідження було виявлено, що реалізація Wi-Fi радіо на базі ESP32 дозволяє забезпечити стабільний та надійний зв'язок з мережею Інтернет у різних умовах експлуатації. Використання даного радіо модуля відкриває широкі можливості для реалізації різноманітних проектів у сфері Інтернету речей, мобільних додатків та інших сучасних технологій.

Отже, апаратно-програмна реалізація Wi-Fi радіо на базі ESP32 є успішним кроком у напрямку розвитку бездротових комунікаційних технологій та має великий потенціал для використання у різних сферах життя та промисловості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Espressif Systems. "Огляд ESP32." Espressif Systems, https://www.espressif.com/en/products/socs/esp32/overview.

2. Espressif Systems. "Технічний довідник ESP32." Espressif Systems, https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

3. Espressif Systems. "Wi-Fi та Bluetooth для ESP32." Espressif Systems, https://www.espressif.com/en/products/sdks/esp-idf.

4. Random Nerd Tutorials. "Початок роботи з ESP32." Random Nerd Tutorials, https://randomnerdtutorials.com/getting-started-with-esp32/.

5. Adafruit. "Adafruit HUZZAH32 – плата ESP32 Feather." Adafruit, https://www.adafruit.com/product/3405.

6. GitHub. "espressif/arduino-esp32." GitHub, https://github.com/espressif/arduino-esp32.

7. Instructables. "WiFi радіо на ESP32." Instructables, https://www.instructables.com/ESP32-WiFi-Radio/.

8. YouTube. "Навчальний відео по WiFi радіо на ESP32." YouTube, https://www.youtube.com/watch?v=O0tc5BFj9vA.

9. Hackster.io. "Створення інтернет-радіо на ESP32." Hackster.io, https://www.hackster.io/makers/esp32-internet-radio.

10. GitHub. "ESP32 Audio: бібліотека для відтворення звуку та музики на ESP32." GitHub, https://github.com/earlephilhower/ESP8266Audio.

ДОДАТКИ

Додаток А

Лістинг програми

```
#define LOG_LOCAL_LEVEL ESP_LOG_VERBOSE
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <nvs.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "freertos/event_groups.h"
#include "freertos/queue.h"
#include "esp_system.h"
#include "esp_wifi.h"
#include "esp_event_loop.h"
#include "esp_log.h"
#include "esp_ota_ops.h"
//#include "esp_heap_trace.h"
#include "nvs_flash.h"
#include "driver/i2s.h"
#include "driver/uart.h"
#include "lwip/sys.h"
#include "lwip/netdb.h"
#include "lwip/api.h"
#include "lwip/tcp.h"
#include "lwip/dns.h"
#include "mdns.h"
#include "app_main.h"
//#include "spiram_fifo.h"
#include "audio_renderer.h"
//#include "bt_speaker.h"
#include "bt_config.h"
//#include "mdns_task.h"
#include "audio_player.h"
#include <u8g2.h>
#include "u8g2_esp32_hal.h"
#include "addon.h"
#include "addonu8g2.h"
```

```
#include "ClickButtons.h"
#include "eeprom.h"
#include "bt_config.h"
#include "driver/gpio.h"
#include "driver/i2c.h"
//#include "esp_wifi.h"
//#include "xi2c.h"
//#include "fonts.h"
//#include "ssd1306.h"
#include "nvs_flash.h"
#include "gpio.h"
#include "servers.h"
#include "webclient.h"
#include "webserver.h"
#include "interface.h"
#include "vs1053.h"
#include "ClickEncoder.h"
#include "addon.h"

const int CONNECTED_BIT = 0x00000001;
const int CONNECTED_AP  = 0x00000010;
#define TAG "main"

//Priorities of the reader and the decoder thread. bigger number
= higher prio
#define PRIO_READER configMAX_PRIORITIES -3
#define PRIO_MQTT configMAX_PRIORITIES - 3
#define PRIO_CONNECT configMAX_PRIORITIES -1
#define striWATERMARK  "watermark: %d  heap: %d"

void start_network();
void autoPlay();

static bool wifiInitDone = false;
static EventGroupHandle_t wifi_event_group ;
xQueueHandle event_queue;

static uint16_t FlashOn = 5,FlashOff = 5;
bool ledStatus; // true: normal blink, false: led on when playing
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів*	1
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

```
bool ledPolarity; // true: normal false: reverse
bool logTel; // true = log also on telnet
player_t *player_config;
static output_mode_t audio_output_mode;
static uint8_t clientIvol = 0;
static char localIp[20];
static bool bigRam = false;
static uint32_t ctimeMs = 0;
static bool divide = false;
IRAM_ATTR                  void                  noInterrupt1Ms()
{timer_disable_intr(TIMERGROUP1MS, msTimer);}
IRAM_ATTR  void  interrupt1Ms()  {timer_enable_intr(TIMERGROUP1MS,
msTimer);}
IRAM_ATTR char* getIp() {return (localIp);}
IRAM_ATTR uint8_t getIvol() {return clientIvol;}
IRAM_ATTR  void  setIvol(  uint8_t  vol)  {clientIvol  =  vol;};
//ctimeVol = 0;}
IRAM_ATTR   output_mode_t   get_audio_output_mode()   {   return
audio_output_mode;}

IRAM_ATTR bool bigSram() { return bigRam;}
IRAM_ATTR void   msCallback(void *pArg) {
     int timer_idx = (int) pArg;

//   queue_event_t evt;
     TIMERG1.hw_timer[timer_idx].update = 1;
     TIMERG1.int_clr_timers.t0 = 1; //isr ack
     if (divide)
     {
     ctimeMs++;     // for led
//   ctimeVol++; // to save volume
     }
     divide = !divide;
     if (serviceAddon != NULL) serviceAddon(); // for the encoders
and buttons
     TIMERG1.hw_timer[timer_idx].config.alarm_en = 1;
}
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

```
IRAM_ATTR void   sleepCallback(void *pArg) {
     int timer_idx = (int) pArg;
     queue_event_t evt;
     TIMERG0.int_clr_timers.t0 = 1; //isr ack
     evt.type = TIMER_SLEEP;
        evt.i1 = TIMERGROUP;
        evt.i2 = timer_idx;
     xQueueSendFromISR(event_queue, &evt, NULL);
     TIMERG0.hw_timer[timer_idx].config.alarm_en = 0;
}
IRAM_ATTR void   wakeCallback(void *pArg) {

     int timer_idx = (int) pArg;
     queue_event_t evt;
     TIMERG0.int_clr_timers.t1 = 1;
        evt.i1 = TIMERGROUP;
        evt.i2 = timer_idx;
     evt.type = TIMER_WAKE;
     xQueueSendFromISR(event_queue, &evt, NULL);
     TIMERG0.hw_timer[timer_idx].config.alarm_en = 0;
}

uint64_t getSleep()
{
     uint64_t ret=0;
     uint64_t tot=0;
     timer_get_alarm_value(TIMERGROUP, sleepTimer,&tot);
     timer_get_counter_value(TIMERGROUP, sleepTimer,&ret);
     ESP_LOGD(TAG,"getSleep:   ret:   %lld,   tot:   %lld,   return
%lld",ret,tot,tot-ret);
     return ((tot)-ret)/5000000;
}
uint64_t getWake()
{
     uint64_t ret=0;
     uint64_t tot=0;
     timer_get_alarm_value(TIMERGROUP, wakeTimer,&tot);
     timer_get_counter_value(TIMERGROUP, wakeTimer,&ret);
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

```
    ESP_LOGD(TAG,"getWake:  ret:  %lld,  tot:  %lld    return
%lld",ret,(tot),tot-ret);
    return ((tot)-ret)/5000000;
}

void tsocket(const char* lab, uint32_t cnt)
{
    char* title = malloc(strlen(lab)+50);
    sprintf(title,"{\"%s\":\"%d\"}",lab,cnt*60);
    websocketbroadcast(title, strlen(title));
    free(title);
}

void stopSleep(){
    ESP_LOGD(TAG,"stopSleep");
    ESP_ERROR_CHECK(timer_pause(TIMERGROUP, sleepTimer));
    ESP_ERROR_CHECK(timer_set_alarm_value(TIMERGROUP,
sleepTimer, 0x00000000ULL));
    ESP_ERROR_CHECK(timer_set_counter_value(TIMERGROUP,
sleepTimer, 0x00000000ULL));

    tsocket("lsleep",0);
}
void startSleep(uint32_t delay)
{
    ESP_LOGD(TAG,"startSleep: %d min.",delay );
    if (delay == 0) return;
    stopSleep();
    ESP_ERROR_CHECK(timer_set_counter_value(TIMERGROUP,
sleepTimer, 0x00000000ULL));
    ESP_ERROR_CHECK(timer_set_alarm_value(TIMERGROUP,
sleepTimer,TIMERVALUE(delay*60)));
    ESP_ERROR_CHECK(timer_enable_intr(TIMERGROUP, sleepTimer));
    ESP_ERROR_CHECK(timer_set_alarm(TIMERGROUP,
sleepTimer,TIMER_ALARM_EN));
    ESP_ERROR_CHECK(timer_start(TIMERGROUP, sleepTimer));
    tsocket("lsleep",delay);
}
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.						*Циклова комісія Комп'ютерної інженерії*		
Н. Контр.								
Затверд.		*Тащук О.Ю.*						

```
void stopWake(){
     ESP_LOGD(TAG,"stopWake");
     ESP_ERROR_CHECK(timer_pause(TIMERGROUP, wakeTimer));
     ESP_ERROR_CHECK(timer_set_counter_value(TIMERGROUP,
wakeTimer, 0x00000000ULL));
     ESP_ERROR_CHECK(timer_set_alarm_value(TIMERGROUP, wakeTimer,
0x00000000ULL));
     tsocket("lwake",0);
}
void startWake(uint32_t delay)
{
     ESP_LOGD(TAG,"startWake: %d min.",delay);
     if (delay == 0) return;
     stopWake();
     ESP_ERROR_CHECK(timer_set_counter_value(TIMERGROUP,
wakeTimer, 0x00000000ULL));
     ESP_ERROR_CHECK(timer_set_alarm_value(TIMERGROUP,
wakeTimer,TIMERVALUE(delay*60)));
     ESP_ERROR_CHECK(timer_enable_intr(TIMERGROUP, wakeTimer));
     ESP_ERROR_CHECK(timer_set_alarm(TIMERGROUP,
wakeTimer,TIMER_ALARM_EN));
     ESP_ERROR_CHECK(timer_start(TIMERGROUP, wakeTimer));
     tsocket("lwake",delay);
}

void initTimers()
{
timer_config_t config;
     config.alarm_en = 1;
    config.auto_reload = TIMER_AUTORELOAD_DIS;
    config.counter_dir = TIMER_COUNT_UP;
    config.divider = TIMER_DIVIDER;
    config.intr_type = TIMER_INTR_LEVEL;
    config.counter_en = TIMER_PAUSE;

    ESP_ERROR_CHECK(timer_init(TIMERGROUP, sleepTimer, &config));
     ESP_ERROR_CHECK(timer_pause(TIMERGROUP, sleepTimer));
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

```
    ESP_ERROR_CHECK(timer_isr_register(TIMERGROUP,   sleepTimer,
sleepCallback, (void*) sleepTimer, 0, NULL));
    /*Configure timer wake*/
   ESP_ERROR_CHECK(timer_init(TIMERGROUP, wakeTimer, &config));
    ESP_ERROR_CHECK(timer_pause(TIMERGROUP, wakeTimer));
    ESP_ERROR_CHECK(timer_isr_register(TIMERGROUP,    wakeTimer,
wakeCallback, (void*) wakeTimer, 0, NULL));
    /*Configure timer 1MS*/
    config.auto_reload = TIMER_AUTORELOAD_EN;
    config.divider = TIMER_DIVIDER1MS ;
    ESP_ERROR_CHECK(timer_init(TIMERGROUP1MS,          msTimer,
&config));
    ESP_ERROR_CHECK(timer_pause(TIMERGROUP1MS, msTimer));
    ESP_ERROR_CHECK(timer_isr_register(TIMERGROUP1MS,   msTimer,
msCallback, (void*) msTimer, 0, NULL));
    /* start 1MS timer*/
    ESP_ERROR_CHECK(timer_set_counter_value(TIMERGROUP1MS,
msTimer, 0x00000000ULL));
    ESP_ERROR_CHECK(timer_set_alarm_value(TIMERGROUP1MS,
msTimer,TIMERVALUE1MS(1)));
    ESP_ERROR_CHECK(timer_enable_intr(TIMERGROUP1MS, msTimer));
    ESP_ERROR_CHECK(timer_set_alarm(TIMERGROUP1MS,
msTimer,TIMER_ALARM_EN));
    ESP_ERROR_CHECK(timer_start(TIMERGROUP1MS, msTimer));

// Renderer config creation
static renderer_config_t *create_renderer_config()
{
    renderer_config_t      *renderer_config      =      calloc(1,
sizeof(renderer_config_t));

    if(renderer_config->output_mode == I2S_MERUS) {
        renderer_config->bit_depth = I2S_BITS_PER_SAMPLE_32BIT;
    }

    if(renderer_config->output_mode == DAC_BUILT_IN) {
        renderer_config->bit_depth = I2S_BITS_PER_SAMPLE_16BIT;
    }
```

Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

```
    return renderer_config;
}

uint32_t checkUart(uint32_t speed)
{
     uint32_t                        valid[]                        =
{1200,2400,4800,9600,14400,19200,28800,38400,57600,76880,115200,
230400};
     int i ;
     for (i=0;i<12;i++){
     if (speed == valid[i]) return speed;
     }
     return 115200; // default
}

static void init_hardware()
{
     if (VS1053_HW_init()) // init spi
     VS1053_Start();

    ESP_LOGI(TAG, "hardware initialized");
}

/* event handler for pre-defined wifi events */
static esp_err_t event_handler(void *ctx, system_event_t *event)
{
    EventGroupHandle_t wifi_event = ctx;

    switch (event->event_id)
    {
    case SYSTEM_EVENT_STA_START:
     FlashOn = FlashOff = 100;
        esp_wifi_connect();
        break;

     case SYSTEM_EVENT_STA_CONNECTED:
     xEventGroupSetBits(wifi_event, CONNECTED_AP);
     ESP_LOGI(TAG, "Wifi connected");
```

Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

```
        if (wifiInitDone)
        {
             clientSaveOneHeader("Wifi     Connected.",18,METANAME);

             vTaskDelay(1000);
             autoPlay();
             } // retry
        else wifiInitDone = true;          break;

       case SYSTEM_EVENT_STA_GOT_IP:
        FlashOn = 5;FlashOff = 395;
           xEventGroupSetBits(wifi_event, CONNECTED_BIT);
           break;

       case SYSTEM_EVENT_STA_DISCONNECTED:
           /* This is a workaround as ESP32 WiFi libs don't currently
    auto-reassociate. */
        FlashOn = FlashOff = 100;
        xEventGroupClearBits(wifi_event, CONNECTED_AP);
           xEventGroupClearBits(wifi_event, CONNECTED_BIT);
        ESP_LOGE(TAG, "Wifi Disconnected.");
        vTaskDelay(100);
           if (!getAutoWifi()&&(wifiInitDone))
        {
             ESP_LOGE(TAG, "reboot");
             vTaskDelay(100);
             esp_restart();
        } else
        {
             if (wifiInitDone) // a completed init done
             {
                  ESP_LOGE(TAG, "Connection tried again");
    //            clientDisconnect("Wifi Disconnected.");
                  clientSilentDisconnect();
                  vTaskDelay(100);
                  clientSaveOneHeader("Wifi
    Disconnected.",18,METANAME);
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

```
                vTaskDelay(100);
                while  (esp_wifi_connect()  ==  ESP_ERR_WIFI_SSID)
vTaskDelay(10);
          } else
          {
               ESP_LOGE(TAG, "Try next AP");
               vTaskDelay(100);
          } // init failed?
     }
        break;

     case SYSTEM_EVENT_AP_START:
     FlashOn = 5;FlashOff = 395;
     xEventGroupSetBits(wifi_event, CONNECTED_AP);
     xEventGroupSetBits(wifi_event, CONNECTED_BIT);
     wifiInitDone = true;
     break;

     case SYSTEM_EVENT_AP_STADISCONNECTED:
     break;

    default:
        break;
    }
    return ESP_OK;
}

static void unParse(char* str)
{
     int i ;
     if (str == NULL) return;
     for (i=0; i< strlen(str);i++)
     {
     if (str[i] == '\\')
     {
          str[i] = str[i+1];
          str[i+1]=0;
          if (str[i+2] !=0)strcat(str, str+i+2);
```

Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

```
     }
     }
}

static void start_wifi()
{
    ESP_LOGI(TAG, "starting wifi");
     setAutoWifi();
//   wifi_mode_t mode;
     char ssid[SSIDLEN];
     char pass[PASSLEN];

     wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
    ESP_ERROR_CHECK( esp_wifi_init(&cfg) );
//    ESP_ERROR_CHECK( esp_wifi_set_storage(WIFI_STORAGE_RAM) );

     tcpip_adapter_init();
     tcpip_adapter_dhcpc_stop(TCPIP_ADAPTER_IF_STA); // Don't run
a DHCP client
    /* FreeRTOS event group to signal when we are connected & ready
to make a request */
     wifi_event_group = xEventGroupCreate();
    ESP_ERROR_CHECK(           esp_event_loop_init(event_handler,
wifi_event_group) );

     if (g_device->current_ap == APMODE)
     {
     if (strlen(g_device->ssid1) !=0)
     {g_device->current_ap = STA1;}
     else
     {    if (strlen(g_device->ssid2) !=0)
          {g_device->current_ap = STA2;}
          else g_device->current_ap = APMODE;
     }
     saveDeviceSettings(g_device);
     }
```

Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

```
        while (1)
        {
        ESP_ERROR_CHECK( esp_wifi_stop() );
        vTaskDelay(10);

        switch (g_device->current_ap)
        {
             case STA1: //ssid1 used
                  strcpy(ssid,g_device->ssid1);
                  strcpy(pass,g_device->pass1);
                  esp_wifi_set_mode(WIFI_MODE_STA) ;
                  break;
             case STA2: //ssid2 used
                  strcpy(ssid,g_device->ssid2);
                  strcpy(pass,g_device->pass2);
                  esp_wifi_set_mode(WIFI_MODE_STA) ;
                  break;

             default: // other: AP mode
                  g_device->current_ap = APMODE;
                  ESP_ERROR_CHECK(esp_wifi_set_mode(WIFI_MODE_AP)) ;
        }

        if (g_device->current_ap == APMODE)
        {
             printf("WIFI GO TO AP MODE\n");
             wifi_config_t ap_config = {
                  .ap = {
                       .ssid = "WifiKaradio",
                       .authmode = WIFI_AUTH_OPEN,
                       .max_connection = 2,
                       .beacon_interval = 200
                  },
             };
             ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_AP,
&ap_config));
             ESP_LOGE(TAG, "The default AP is  WifiKaRadio. Connect
your wifi to it.\nThen connect a webbrowser to 192.168.4.1 and go
to Setting\nMay be long to load the first time.Be patient.");
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів*	1
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

```
            vTaskDelay(1);
            ESP_ERROR_CHECK( esp_wifi_start() );

//          audio_output_mode = I2S;
            option_get_audio_output(&audio_output_mode);
      }
      else
      {
            printf("WIFI TRYING TO CONNECT TO SSID %d\n",g_device-
>current_ap);
            wifi_config_t wifi_config = {
                  .sta = {
                        .bssid_set = 0,
                        .scan_method = WIFI_ALL_CHANNEL_SCAN,
                        .sort_method = WIFI_CONNECT_AP_BY_SIGNAL,
            },
            };
            strcpy((char*)wifi_config.sta.ssid,ssid);
            strcpy((char*)wifi_config.sta.password,pass);
            unParse((char*)(wifi_config.sta.ssid));
            unParse((char*)(wifi_config.sta.password));
            if  (strlen(ssid)/*&&strlen(pass)*/)
            {
                  esp_wifi_disconnect();
                  ESP_ERROR_CHECK(esp_wifi_set_config(WIFI_IF_STA,
&wifi_config));
//                ESP_LOGI(TAG,    "connecting    %s,    %d,    %s,
%d",ssid,strlen((char*)(wifi_config.sta.ssid)),pass,strlen((char
*)(wifi_config.sta.password)));
                  ESP_LOGI(TAG, "connecting %s",ssid);
                  ESP_ERROR_CHECK( esp_wifi_start() );
            } else
            {
                  g_device->current_ap++;
                  g_device->current_ap %=3;
```

Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

```
                if  (getAutoWifi()  &&  (g_device->current_ap  ==
APMODE))
                {
                     if (fgetc(stdin)==0xFF) // if a char read,
stop the autowifi
                     g_device->current_ap = STA1; // if autoWifi
then wait for a reconnection to an AP
                     ESP_LOGI(TAG,"Wait for the AP");
                }
                else
                     ESP_LOGI(TAG,"Empty AP. Try next one");

                saveDeviceSettings(g_device);
                continue;
           }
     }

     /* Wait for the callback to set the CONNECTED_BIT in the event
group. */
     if          (          (xEventGroupWaitBits(wifi_event_group,
CONNECTED_AP,false, true, 1500) & CONNECTED_AP) ==0)
     //timeout . Try the next AP
     {
          g_device->current_ap++;
          g_device->current_ap %=3;
          if (getAutoWifi() && (g_device->current_ap == APMODE))
          {
               char inp = fgetc(stdin);
               printf("\nfgetc : %x\n",inp);
               if (inp==0xFF) //
                    g_device->current_ap = STA1;//if a char read,
stop the autowifi
          }
          saveDeviceSettings(g_device);
          ESP_LOGI(TAG,"device->current_ap:         %d",g_device-
>current_ap);
     }    else break;    //
     }
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

```
}

void start_network(){
//   struct device_settings *g_device;
     tcpip_adapter_ip_info_t info;
     wifi_mode_t mode;
     ip4_addr_t ipAddr;
     ip4_addr_t mask;
     ip4_addr_t gate;
     uint8_t dhcpEn = 0;

     switch (g_device->current_ap)
     {
     case STA1: //ssid1 used
          IP4_ADDR(&ipAddr,    g_device->ipAddr1[0],    g_device-
>ipAddr1[1],g_device->ipAddr1[2], g_device->ipAddr1[3]);
          IP4_ADDR(&gate,           g_device->gate1[0],g_device-
>gate1[1],g_device->gate1[2], g_device->gate1[3]);
          IP4_ADDR(&mask,      g_device->mask1[0],      g_device-
>mask1[1],g_device->mask1[2], g_device->mask1[3]);
          dhcpEn = g_device->dhcpEn1;
     break;
     case STA2: //ssid2 used
          IP4_ADDR(&ipAddr,    g_device->ipAddr2[0],    g_device-
>ipAddr2[1],g_device->ipAddr2[2], g_device->ipAddr2[3]);
          IP4_ADDR(&gate,           g_device->gate2[0],g_device-
>gate2[1],g_device->gate2[2], g_device->gate2[3]);
          IP4_ADDR(&mask,      g_device->mask2[0],      g_device-
>mask2[1],g_device->mask2[2], g_device->mask2[3]);
          dhcpEn = g_device->dhcpEn2;
     break;

     default: // other: AP mode
          IP4_ADDR(&ipAddr, 192,168,4,1);
          IPADDR2_COPY(&gate,&ipAddr);
          IP4_ADDR(&mask,255,255,255,0);
     }

     IPADDR2_COPY(&info.ip,&ipAddr);
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів*	1
Реценз.						*Циклова комісія Комп'ютерної інженерії*		
Н. Контр.								
Затверд.		*Тащук О.Ю.*						

```
     IPADDR2_COPY(&info.gw,&gate);
     IPADDR2_COPY(&info.netmask,&mask);

     ESP_ERROR_CHECK(esp_wifi_get_mode(&mode));
     if (mode == WIFI_MODE_AP)
     {
          xEventGroupWaitBits(wifi_event_group,
CONNECTED_BIT,false, true, 3000);
          IPADDR2_COPY(&info.ip,&ipAddr);
          tcpip_adapter_set_ip_info(TCPIP_ADAPTER_IF_AP, &info);
          strcpy(localIp , ip4addr_ntoa(&info.ip));
          printf("IP: %s\n\n",ip4addr_ntoa(&info.ip));

     }
     else // mode STA
     {
     if (dhcpEn ) // check if ip is valid without dhcp
          tcpip_adapter_dhcpc_start(TCPIP_ADAPTER_IF_STA);     //
run a DHCP client
     else
     {

     ESP_ERROR_CHECK(tcpip_adapter_set_ip_info(TCPIP_ADAPTER_IF_S
TA, &info));
          dns_clear_servers(false);
          IP_SET_TYPE(( ip_addr_t* )&info.gw, IPADDR_TYPE_V4); //
mandatory
          (( ip_addr_t* )&info.gw)->type = IPADDR_TYPE_V4;
          dns_setserver(0,( ip_addr_t* ) &info.gw);
          dns_setserver(1,( ip_addr_t* ) &info.gw);
     // if static ip     check dns
     }

     // wait for ip
     if         (         (xEventGroupWaitBits(wifi_event_group,
CONNECTED_BIT,false, true, 3000) & CONNECTED_BIT) ==0) //timeout
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32		Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Кирчул К.К.							
Перевір.		Тащук О.Ю.							
Т. Контр.							Арк.	Аркушів	1
Реценз.									
Н. Контр.							Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.							

```
        { // enable dhcp and restart
             if (g_device->current_ap ==1)
                  g_device->dhcpEn1 = 1;
             else
                  g_device->dhcpEn2 = 1;
             saveDeviceSettings(g_device);
             esp_restart();
        }

        vTaskDelay(1);
        // retrieve the current ip
        tcpip_adapter_ip_info_t ip_info;
        ip_info.ip.addr =0;
        while (ip_info.ip.addr ==0)
        {
             if (mode == WIFI_MODE_AP)
                  tcpip_adapter_get_ip_info(TCPIP_ADAPTER_IF_AP,
   &ip_info);
             else
                  tcpip_adapter_get_ip_info(TCPIP_ADAPTER_IF_STA,
   &ip_info);
        }
        ip_addr_t *ipdns0 = (ip_addr_t *)dns_getserver(0);
   //   ip_addr_t ipdns1 = dns_getserver(1);
        printf("\nDNS:  %s    \n",ip4addr_ntoa((  struct  ip4_addr*  )
   ipdns0));
        strcpy(localIp , ip4addr_ntoa(&ip_info.ip));
        printf("IP: %s\n\n",ip4addr_ntoa(&ip_info.ip));


        if (dhcpEn) // if dhcp enabled update fields
        {
             tcpip_adapter_get_ip_info(TCPIP_ADAPTER_IF_STA, &info);
             switch (g_device->current_ap)
             {
             case STA1: //ssid1 used
                  IPADDR2_COPY(&g_device->ipAddr1, &info.ip);
                  IPADDR2_COPY(&g_device->mask1, &info.netmask);
                  IPADDR2_COPY(&g_device->gate1, &info.gw);
```

```
                break;
                case STA2: //ssid2 used
                     IPADDR2_COPY(&g_device->ipAddr2, &info.ip);
                     IPADDR2_COPY(&g_device->mask2, &info.netmask);
                     IPADDR2_COPY(&g_device->gate2, &info.gw);
                break;
                }
           }
           saveDeviceSettings(g_device);
           tcpip_adapter_set_hostname(TCPIP_ADAPTER_IF_STA,
      "karadio32");
           }

           lcd_welcome(localIp,"IP found");
           vTaskDelay(10);

      }

      //blinking led and timer isr
      void timerTask(void* p) {
      //   struct device_settings *device;
           uint32_t cCur;
           bool stateLed = false;
           bool isEsplay;
           gpio_num_t gpioLed;
      //   int uxHighWaterMark;

           initTimers();
           isEsplay = option_get_esplay();
           gpio_get_ledgpio(&gpioLed);
           setLedGpio(gpioLed);

           if (gpioLed != GPIO_NONE)
           {
           gpio_output_conf(gpioLed);
           gpio_set_level(gpioLed, ledPolarity ? 1 : 0);
           }
           cCur = FlashOff*10;
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

```
     queue_event_t evt;

     while(1) {
     // read and treat the timer queue events
//   int nb = uxQueueMessagesWaiting(event_queue);
//   if (nb >29) printf(" %d\n",nb);
     while (xQueueReceive(event_queue, &evt, 0))
     {
          switch (evt.type){
                    case TIMER_SLEEP:
                         clientDisconnect("Timer");  //  stop  the
player
                    break;
                    case TIMER_WAKE:
                    clientConnect(); // start the player
                    break;
                    default:
                    break;
          }
     }
     if (ledStatus)
     {
          if (ctimeMs >= cCur)
          {
               gpioLed = getLedGpio();

               if (stateLed)
               {
                    if        (gpioLed        !=        GPIO_NONE)
gpio_set_level(gpioLed,ledPolarity?1:0);
                    stateLed = false;
                    cCur = FlashOff*10;
               } else
               {
                    if        (gpioLed        !=        GPIO_NONE)
gpio_set_level(gpioLed,ledPolarity?0:1);
                    stateLed = true;
                    cCur = FlashOn*10;
```

Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.				Арк.	Аркушів	1
Т. Контр.								
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

```
                    }
                    ctimeMs = 0;
               }
          }

          vTaskDelay(10);

          if (isEsplay) // esplay board only
               rexp = i2c_keypad_read(); // read the expansion
          }
//   printf("t0 end\n");
          vTaskDelete( NULL ); // stop the task (never reached)
}

void uartInterfaceTask(void *pvParameters) {
          char tmp[255];
          int d;
          uint8_t c;
          int t ;
          esp_err_t err;
//   struct device_settings *device;
          uint32_t uspeed;
          int uxHighWaterMark;

          uspeed = g_device->uartspeed;
     uart_config_t uart_config0 = {
               .baud_rate = uspeed,
               .data_bits = UART_DATA_8_BITS,
               .parity = UART_PARITY_DISABLE,
               .stop_bits = UART_STOP_BITS_1,
               .flow_ctrl          =          UART_HW_FLOWCTRL_DISABLE,
//UART_HW_FLOWCTRL_CTS_RTS,
               .rx_flow_ctrl_thresh = 0,
          };
          err = uart_param_config(UART_NUM_0, &uart_config0);
          if                                                  (err!=ESP_OK)
ESP_LOGE("uartInterfaceTask","uart_param_config err: %d",err);

          err = uart_driver_install(UART_NUM_0, 1024 , 0, 0, NULL, 0);
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

```
        if (err!=ESP_OK)
        {
        ESP_LOGE("uartInterfaceTask","uart_driver_install      err:
    %d",err);
        vTaskDelete(NULL);
        }

        for(t = 0; t<sizeof(tmp); t++) tmp[t] = 0;
        t = 0;

        while(1) {
        while(1) {
             d= uart_read_bytes(UART_NUM_0, &c, 1, 100);
             if (d>0)
             {
                  if((char)c == '\r') break;
                  if((char)c == '\n') break;
                  tmp[t] = (char)c;
                  t++;
                  if(t == sizeof(tmp)-1) t = 0;
             }
             //else printf("uart d: %d, T= %d\n",d,t);
             //switchCommand() ;  // hardware panel of command
        }
        checkCommand(t, tmp);
        uxHighWaterMark = uxTaskGetStackHighWaterMark( NULL );

        ESP_LOGD("uartInterfaceTask",striWATERMARK,uxHighWaterMark,x
    PortGetFreeHeapSize( ));

        for(t = 0; t<sizeof(tmp); t++) tmp[t] = 0;
        t = 0;
        }
    }

    // In STA mode start a station or start in pause mode.
    // Show ip on AP mode.
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

```
void autoPlay()
{
     char apmode[50];
     sprintf(apmode,"at IP %s",localIp);
     if (g_device->current_ap == APMODE)
     {
     clientSaveOneHeader("Configure the AP with the web
page",34,METANAME);
     clientSaveOneHeader(apmode,strlen(apmode),METAGENRE);
     } else
     {
     clientSaveOneHeader(apmode,strlen(apmode),METANAME);
     if ((audio_output_mode == VS1053) && (getVsVersion() < 3))
     {
          clientSaveOneHeader("Invalid audio output. VS1053 not
found",38,METAGENRE);
          ESP_LOGE(TAG,"Invalid audio output. VS1053 not found");
          vTaskDelay(200);
     }

     setCurrentStation( g_device->currentstation);
     if ((g_device->autostart ==1)&&(g_device->currentstation !=
0xFFFF))
     {
          kprintf("autostart:                          playing:%d,
currentstation:%d\n",g_device->autostart,g_device-
>currentstation);
          vTaskDelay(10); // wait a bit
          playStationInt(g_device->currentstation);
     } else clientSaveOneHeader("Ready",5,METANAME);

     }
}

void app_main()
{
     uint32_t uspeed;
     xTaskHandle pxCreatedTask;
     esp_err_t err;
```

Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.				Арк.	Аркушів	1
Т. Контр.								
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

```
     ESP_LOGI(TAG, "starting app_main()");
    ESP_LOGI(TAG, "RAM left: %u", esp_get_free_heap_size());

     const          esp_partition_t          *running          =
esp_ota_get_running_partition();
     ESP_LOGE(TAG, "Running partition type %d subtype %d (offset
0x%08x)",
              running->type, running->subtype, running->address);
    // Initialize NVS.
    err = nvs_flash_init();
    if (err == ESP_ERR_NVS_NO_FREE_PAGES) {
        // OTA app partition table has a smaller NVS partition
size than the non-OTA
        // partition table. This size mismatch may cause NVS
initialization to fail.
        // If this happens, we erase NVS partition and initialize
NVS again.
        ESP_ERROR_CHECK(nvs_flash_erase());
        err = nvs_flash_init();
    }
    ESP_ERROR_CHECK( err );

     // Check if we are in large Sram config
     if (xPortGetFreeHeapSize() > 0x80000) bigRam = true;
     //init hardware
     partitions_init();
     ESP_LOGI(TAG, "Partition init done...");

     if (g_device->cleared != 0xAABB)
     {
     ESP_LOGE(TAG,"Device config not ok. Try to restore");
     free(g_device);
     restoreDeviceSettings(); // try to restore the config from
the saved one
     g_device = getDeviceSettings();
     if (g_device->cleared != 0xAABB)
     {
          ESP_LOGE(TAG,"Device config not cleared. Clear it.");
```

Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.				Арк.	Аркушів	1
Т. Контр.								
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

```
            free(g_device);
            eeEraseAll();
            g_device = getDeviceSettings();
            g_device->cleared = 0xAABB; //marker init done
            g_device->uartspeed = 115200; // default
//          g_device->audio_output_mode = I2S; // default
            option_get_audio_output(&(g_device-
>audio_output_mode));
            g_device->trace_level = ESP_LOG_ERROR; //default
            g_device->vol = 100; //default
            g_device->led_gpio = GPIO_NONE;
            saveDeviceSettings(g_device);
        } else
            ESP_LOGE(TAG,"Device config restored");
        }

        copyDeviceSettings(); // copy in the safe partion

        // Configure Deep Sleep start and wakeup options
        deepSleepConf(); // also called in addon.c
        // Enter ESP32 Deep Sleep (but not powerdown uninitialized
peripherals) when P_SLEEP GPIO is P_LEVEL_SLEEP
        if (checkDeepSleepInput())
        esp_deep_sleep_start();

        // led mode
        if (g_device->options & T_LED)
        ledStatus = false; // play mode
        else
        ledStatus = true; // blink mode

        if (g_device->options & T_LEDPOL)
        ledPolarity = true;
        else
        ledPolarity = false;

        // log on telnet
        if (g_device->options & T_LOGTEL)
        logTel = true; //
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів*	1
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

```
        else
        logTel = false; //

        // init softwares
        telnetinit();
        websocketinit();

        // log level
        setLogLevel(g_device->trace_level);

        //time display
        uint8_t ddmm;
        option_get_ddmm(&ddmm);
        setDdmm(ddmm?1:0);

        //SPI init for the vs1053 and lcd if spi.
        VS1053_spi_init();

       init_hardware();

        //ESP_LOGE(TAG,"Corrupt1
    %d",heap_caps_check_integrity(MALLOC_CAP_DMA,1));

        // the esplay board needs I2C for gpio extension
        if (option_get_esplay())
        {
        gpio_num_t scl;
        gpio_num_t sda;
        gpio_num_t rsti2c;
        gpio_get_i2c(&scl,&sda,&rsti2c);
        ESP_LOGD(TAG,"I2C GPIO SDA: %d, SCL: %d",sda,scl);
        i2c_config_t conf;
        conf.mode = I2C_MODE_MASTER;
        conf.sda_io_num = sda;
        conf.sda_pullup_en = GPIO_PULLUP_ENABLE;
        conf.scl_io_num = scl;
        conf.scl_pullup_en = GPIO_PULLUP_ENABLE;
        conf.master.clk_speed = I2C_MASTER_FREQ_HZ;
        esp_err_t res = i2c_param_config(I2C_MASTER_NUM, &conf);
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

```
    ESP_LOGD(TAG,"I2C setup : %d\n",res);
    res    =    i2c_driver_install(I2C_MASTER_NUM,    conf.mode,
I2C_MASTER_RX_BUF_DISABLE, I2C_MASTER_TX_BUF_DISABLE, 0);
    if  (res  !=  0)  ESP_LOGD(TAG,"I2C  already  installed.  No
problem");
    else ESP_LOGD(TAG,"I2C installed: %d",res);

    //init the amp shutdown gpio4 as output level 1
    gpio_output_conf(PIN_AUDIO_SHDN);

    }

    // output mode
    //I2S, I2S_MERUS, DAC_BUILT_IN, PDM, VS1053
    audio_output_mode = g_device->audio_output_mode;
    ESP_LOGI(TAG, "audio_output_mode %d\nOne of I2S=0, I2S_MERUS,
DAC_BUILT_IN, PDM, VS1053, SPDIF",audio_output_mode);

    //uart speed
    uspeed = g_device->uartspeed;
    uspeed = checkUart(uspeed);
    uart_set_baudrate(UART_NUM_0, uspeed);
    ESP_LOGI(TAG, "Set baudrate at %d",uspeed);
    if (g_device->uartspeed != uspeed)
    {
    g_device->uartspeed = uspeed;
    saveDeviceSettings(g_device);
    }

    // Version infos
    ESP_LOGI(TAG, "Release %s, Revision %s",RELEASE,REVISION);
    ESP_LOGI(TAG, "SDK %s",esp_get_idf_version());
    ESP_LOGI(TAG, "Heap size: %d",xPortGetFreeHeapSize());

    // lcd init
    uint8_t rt;
    option_get_lcd_info(&g_device->lcd_type,&rt);
    ESP_LOGI(TAG,"LCD Type %d",g_device->lcd_type);
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів*	1
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

```
        //lcd rotation
        setRotat(rt) ;
        lcd_init(g_device->lcd_type);
        ESP_LOGI(TAG, "Hardware init done...");

        lcd_welcome("","");
        lcd_welcome("","STARTING");

        // volume
        setIvol( g_device->vol);
        ESP_LOGI(TAG, "Volume set to %d",g_device->vol);
        // queue for events of the sleep / wake and Ms timers
        event_queue = xQueueCreate(30, sizeof(queue_event_t));
        // led blinks
        xTaskCreatePinnedToCore(timerTask,  "timerTask",2100,  NULL,
    PRIO_TIMER, &pxCreatedTask,CPU_TIMER);
        ESP_LOGI(TAG,      "%s      task:      %x","t0",(unsigned
    int)pxCreatedTask);

        xTaskCreatePinnedToCore(uartInterfaceTask,
    "uartInterfaceTask",        2500,        NULL,        PRIO_UART,
    &pxCreatedTask,CPU_UART);
        ESP_LOGI(TAG,  "%s  task:  %x","uartInterfaceTask",(unsigned
    int)pxCreatedTask);

       start_wifi();
        start_network();

        clientInit();
        //initialize mDNS service
       err = mdns_init();
       if (err)
           ESP_LOGE(TAG,"mDNS Init failed: %d", err);
        else
        ESP_LOGI(TAG,"mDNS Init ok");

        //set hostname and instance name
        if  ((strlen(g_device->hostname)  ==  0)||(strlen(g_device-
    >hostname) > HOSTLEN))
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

```
     {
     strcpy(g_device->hostname,"karadio32");
     }
     ESP_LOGI(TAG,"mDNS Hostname: %s",g_device->hostname );
     err = mdns_hostname_set(g_device->hostname);
     if (err)
        ESP_LOGE(TAG,"Hostname Init failed: %d", err);

     ESP_ERROR_CHECK(mdns_instance_name_set(g_device->hostname));
     ESP_ERROR_CHECK(mdns_service_add(NULL, "_http", "_tcp", 80,
NULL, 0));
     ESP_ERROR_CHECK(mdns_service_add(NULL,  "_telnet",  "_tcp",
23, NULL, 0));

    // init player config
    player_config = (player_t*)calloc(1, sizeof(player_t));
    player_config->command = CMD_NONE;
    player_config->decoder_status = UNINITIALIZED;
    player_config->decoder_command = CMD_NONE;
    player_config->buffer_pref = BUF_PREF_SAFE;
    player_config->media_stream           =           calloc(1,
sizeof(media_stream_t));
     audio_player_init(player_config);
     renderer_init(create_renderer_config());

     // LCD Display infos
    lcd_welcome(localIp,"STARTED");
     vTaskDelay(10);
    ESP_LOGI(TAG, "RAM left %d", esp_get_free_heap_size());

     //start tasks of KaRadio32
     vTaskDelay(1);
     xTaskCreatePinnedToCore(clientTask,   "clientTask",   3800,
NULL, PRIO_CLIENT, &pxCreatedTask,CPU_CLIENT);
     ESP_LOGI(TAG,    "%s    task:    %x","clientTask",(unsigned
int)pxCreatedTask);
     vTaskDelay(1);
    xTaskCreatePinnedToCore(serversTask,   "serversTask",   3100,
NULL, PRIO_SERVER, &pxCreatedTask,CPU_SERVER);
```

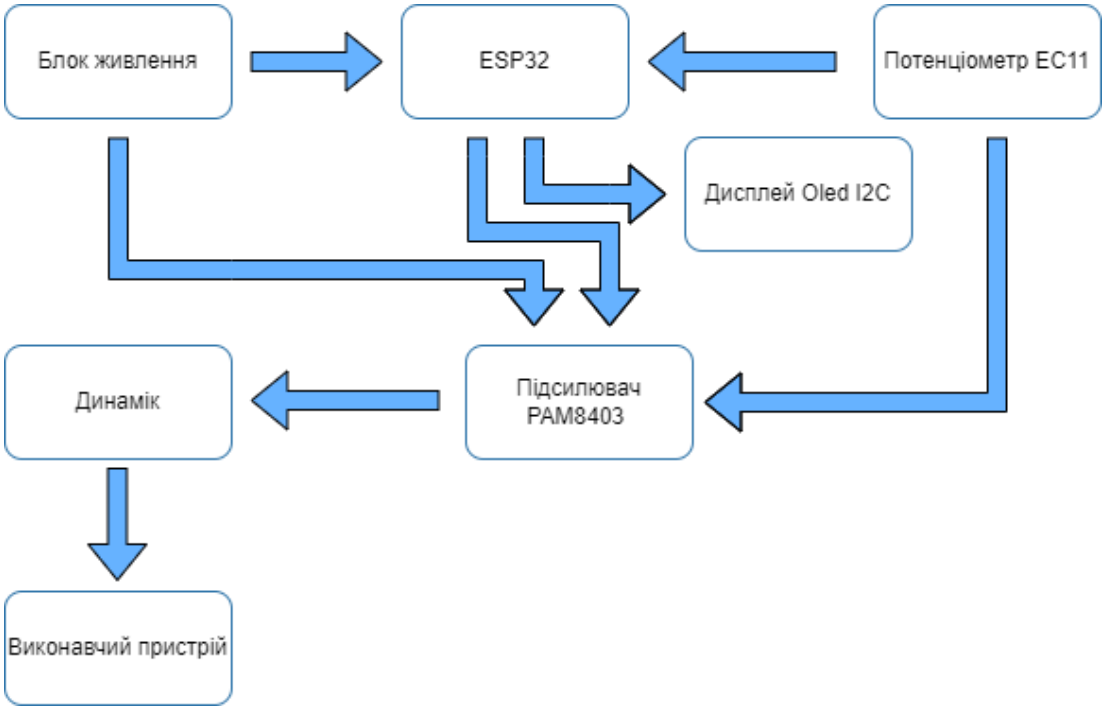
					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів*	1
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

```
    ESP_LOGI(TAG,    "%s    task:    %x","serversTask",(unsigned
int)pxCreatedTask);
    vTaskDelay(1);
    xTaskCreatePinnedToCore  (task_addon,  "task_addon",  2200,
NULL, PRIO_ADDON, &pxCreatedTask,CPU_ADDON);
    ESP_LOGI(TAG,    "%s    task:    %x","task_addon",(unsigned
int)pxCreatedTask);

    vTaskDelay(60);// wait tasks init
    ESP_LOGI(TAG," Init Done");

    setIvol( g_device->vol);
    kprintf("READY. Type help for a list of commands\n");
    // error log on telnet
    esp_log_set_vprintf( (vprintf_like_t)lkprintf);
    //autostart
    autoPlay();
// All done.
}
```

					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

Додаток Б

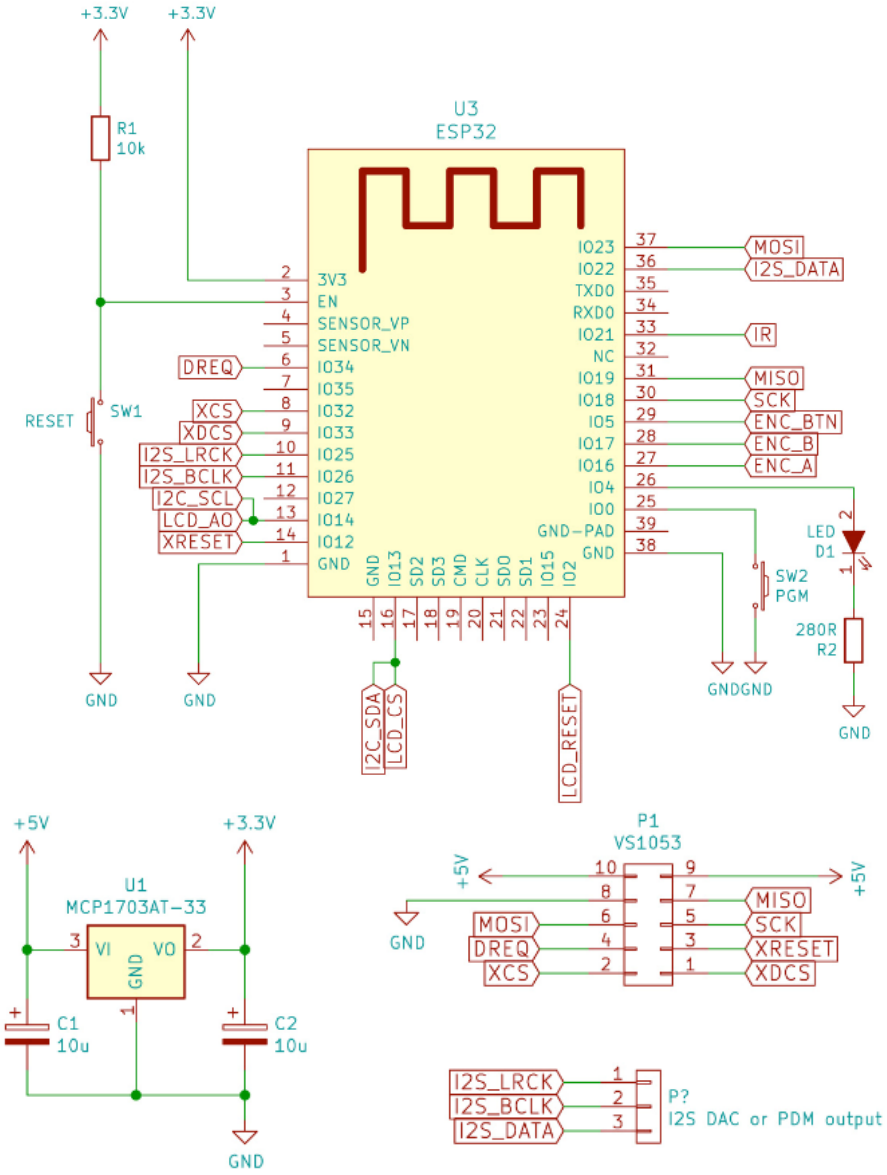
Структурна принципова схема модуля Wi-Fi радіо



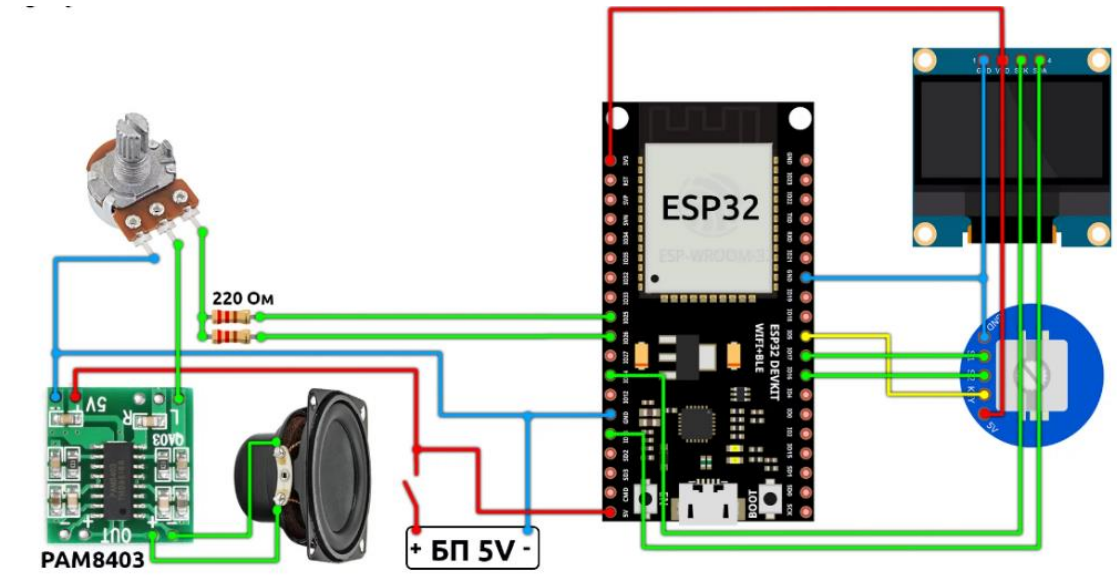
					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

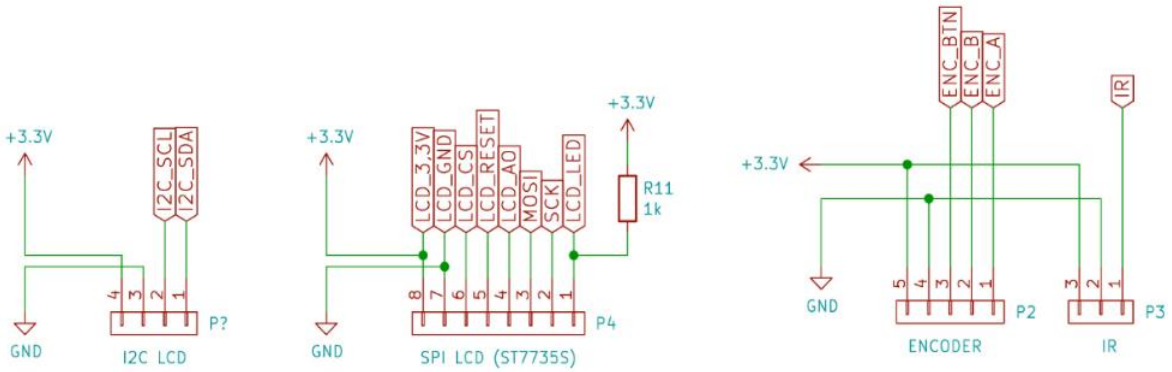
Додаток В

Електрична принципова схема модуля Wi-Fi радіо



Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.				Арк.	Аркушів	1
Т. Контр.								
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						

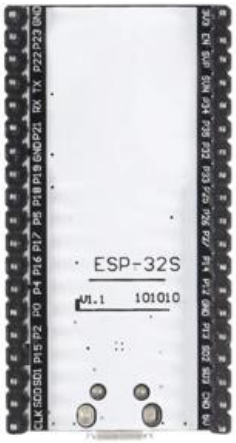




					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.						Циклова комісія Комп'ютерної інженерії		
Затверд.		Тащук О.Ю.						

Додаток Г

Компоненти для апаратно-програмного модуля Wi-fi радіо на основі ESP32

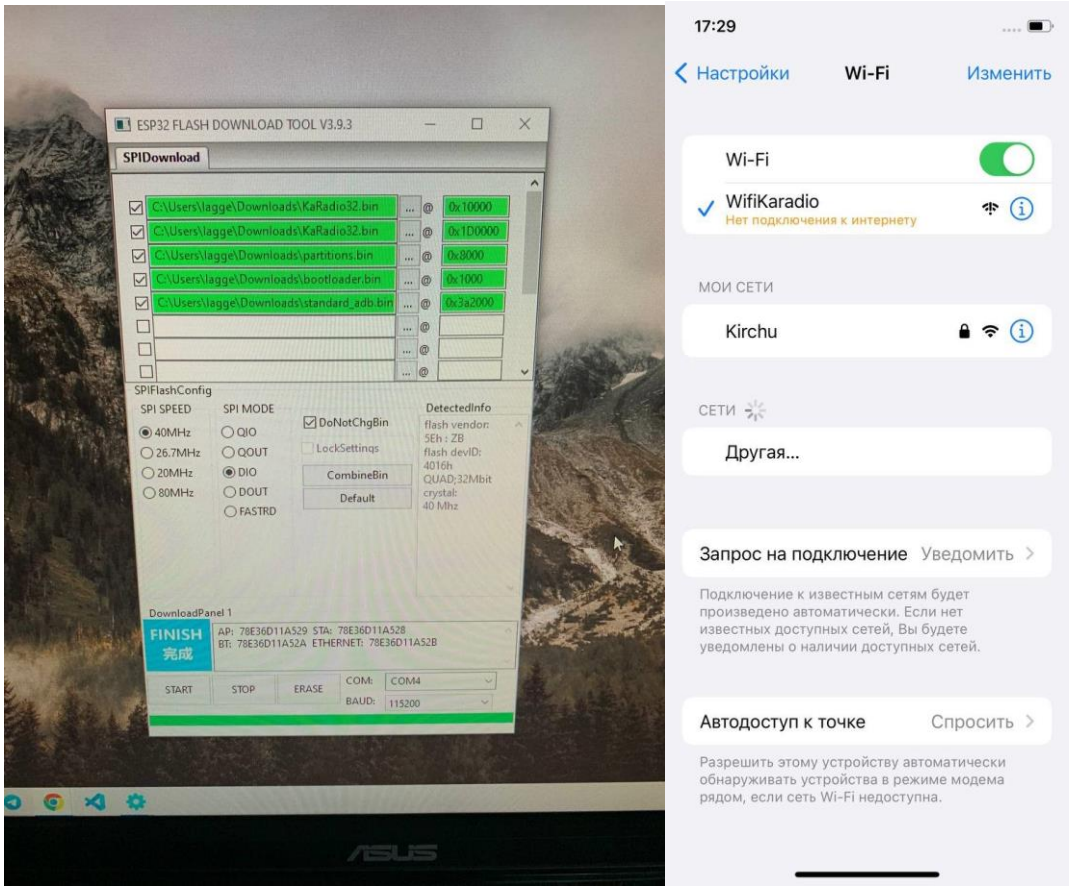


					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

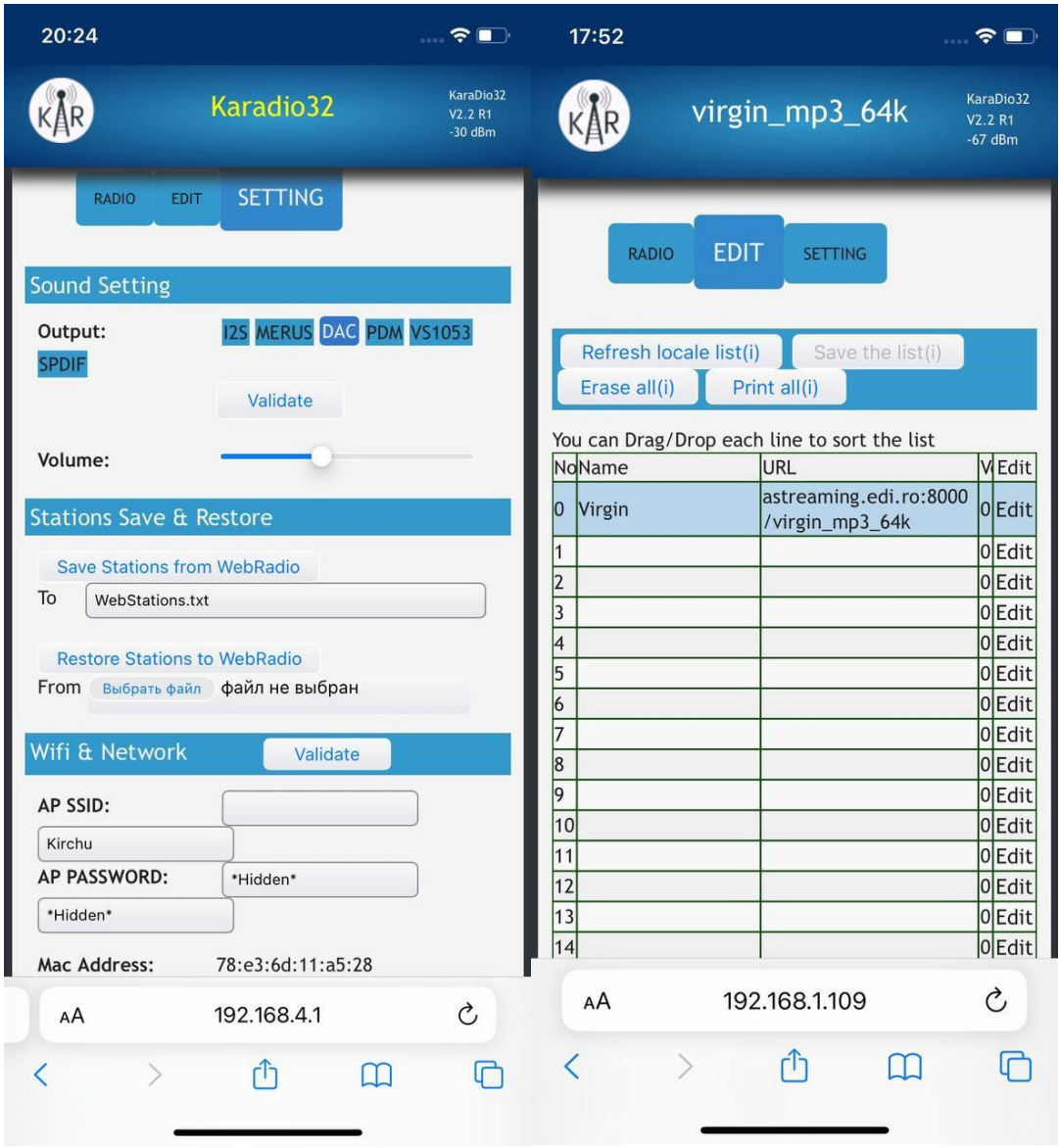
					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	*Літ.*	*Маса*	*Масштаб*
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Кирчул К.К.*						
Перевір.		*Тащук О.Ю.*						
Т. Контр.						*Арк.*	*Аркушів* 1	
Реценз.								
Н. Контр.						*Циклова комісія Комп'ютерної інженерії*		
Затверд.		*Тащук О.Ю.*						

Додаток Д

Налаштування та тестування апаратно-програмного модуля Wi-fi радіо на основі ESP32

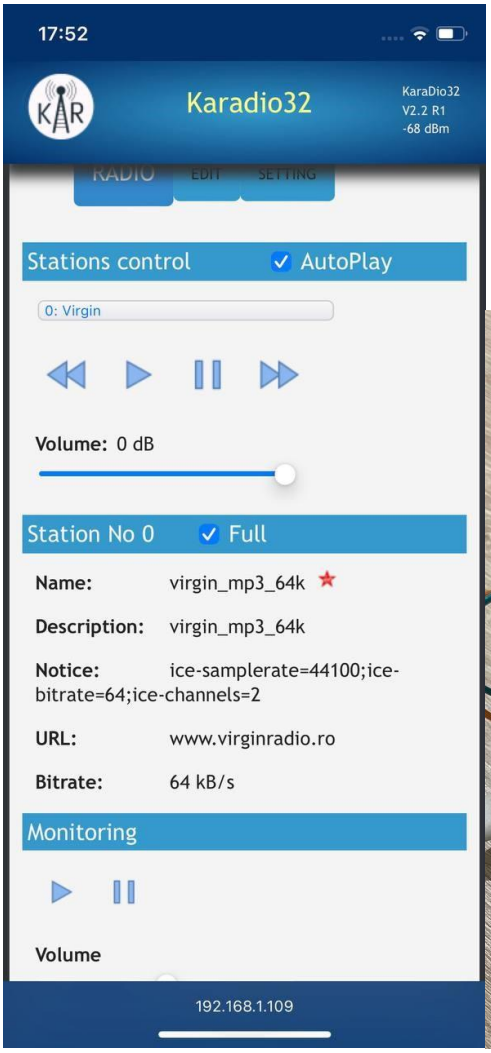


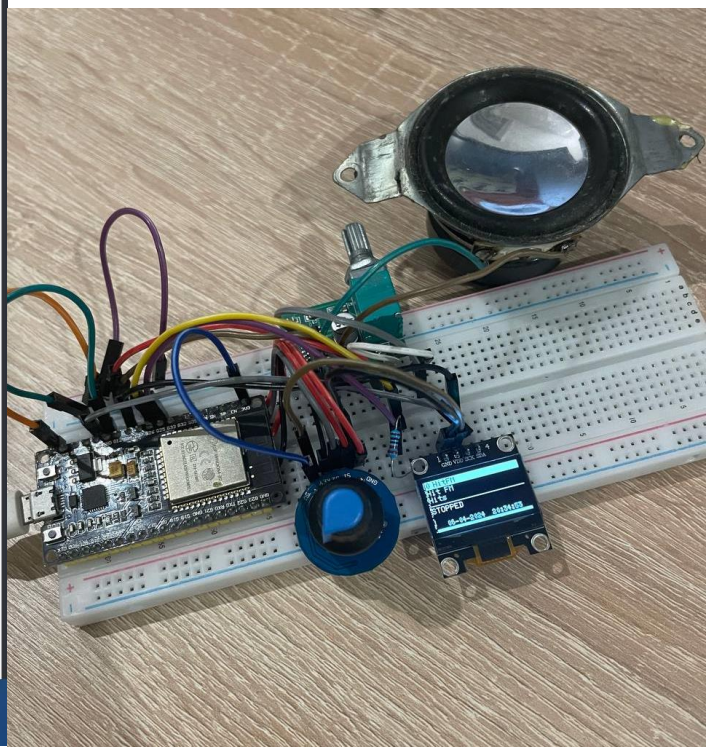
Змн.	Арк.	№ докум.	Підпис	Дата	Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.								
Затверд.		Тащук О.Ю.				Циклова комісія Комп'ютерної інженерії		



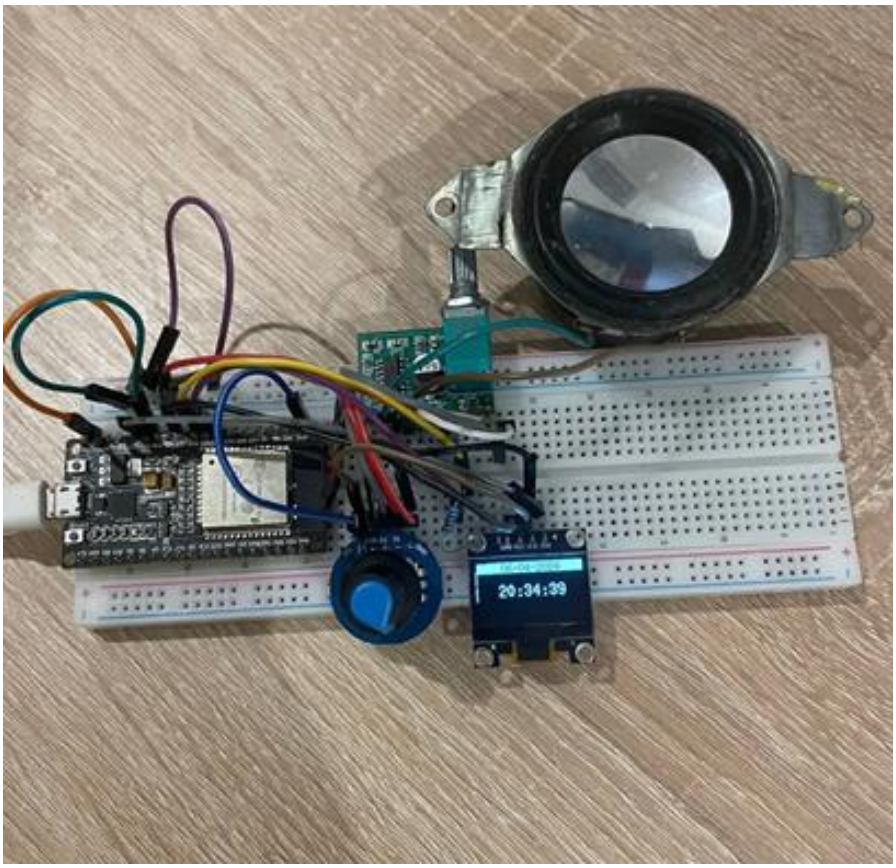
					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.								
Н. Контр.								
Затверд.		Тащук О.Ю.						

Циклова комісія
Комп'ютерної інженерії





					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.				Арк.	Аркушів	1
Т. Контр.								
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						



					Апаратно-програмна реалізація Wi-Fi радіо на основі ESP32	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Кирчул К.К.						
Перевір.		Тащук О.Ю.						
Т. Контр.						Арк.	Аркушів	1
Реценз.						Циклова комісія Комп'ютерної інженерії		
Н. Контр.								
Затверд.		Тащук О.Ю.						