НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Радіотехнічний факультет

Кафедра прикладної радіоелектроніки

До захисту допущено:
Зав. кафедри
_____Андрій Мовчанюк
«15» червня 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньою програмою «Інтелектуальні технології радіоелектронної техніки», за спеціальністю 172 «Телекомунікації та радіотехніка»

на тему: «Сигналізатор повітряної тривоги»

Виконав:
студент IV курсу, групи РЕ-11

Слинчук Олексій Олексійович

Керівник:

Асистент Стешенко Владлен Дмитрович

Рецензент:
Саратов Євген

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025року

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4	РЕ11.468232.001 ПЗ	Пояснювальна записка	47	
2	А4	РЕ11.468232.001	Специфікація на пристрій	1	
3	А1	РЕ11.468232.001 Е1	Схема структурна	1	
4	А1	РЕ11.468232.001 Е3	Схема електрична принципова	1	
5	А1	РЕ11.758735.001	Друкована плата	1	
6	А1	РЕ11.687131.001	Корпус	1	

				РЕ11.468232.001		
	ПІБ	Підп.	Дата			
Розробн.	Слинчук О. О.			Сигналізатор повітряної тривоги	Лист	Листів
Керівн.	Стешенко В. Д.				1	1
					КПІ ім. Ігоря Сікорського Каф. ПРЕ, Гр. РЕ-11	
Н/контр.						
Зав.каф.	Мовчанюк А.В.					

Київський політехнічний інститут імені Ігоря Сікорського»
Радіотехнічний факультет
Кафедра прикладної радіоелектроніки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність 172 «Телекомунікації та радіотехніка»
Освітня програма «Інтелектуальні технології радіоелектронної техніки»

ЗАТВЕРДЖУЮ
Зав. кафедри
_____Андрій МОВЧАНЮК
«15» червня 2025 р.

ЗАВДАННЯ
на дипломний проєкт студента
Слинчука Олексія Олексійовича

1.Тема проєкту «Сигналізатор повітряної тривоги», керівник проєкту Владлен Дмитрович Стешенко, затверджені наказом по університету від «29» травня 2025 р. №1842-с

2. Строк подання студентом проєкту 10 червня 2025 року

3. Вихідні дані до проєкту: Напруга живлення 5В, струм до 0.1 А, модульність конструкції, автономність

4. Зміст пояснювальної записки 1)Огляд існуючих рішень, 2) Розробка схеми електричної принципової 3) проектування друкованого вузла 4)Конструювання корпусу та пристрою.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) Складальний кресленик плати, складальний кресленик пристрою, презентація 10 сторінок.

6. Дата видачі завдання 14 квітня 2025 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Огляд існуючих рішень	14.04.25-06.05.25	Виконано
2	Розробка та аналіз завдання на проєкт	07.05.25-10.05.25	Виконано
3	Розробка функціональної схеми та вибір компонентів	11.05.25-18.05.25	Виконано
4	Розробка схеми електричної принципової	19.05.25-23.05.25	Виконано
5	Розробка друкованої плати	24.05.25-29.05.25	Виконано
6	Розробка конструкторської документації	30.05.25-05.06.25	Виконано
7	Оформлення текстової та графічної документації	06.06.25-09.06.25	Виконано

Студент  Олексій СЛИНЧУК

Керівник  Владлен СТЕШЕНКО

АНОТАЦІЯ

Дипломна робота складається з пояснювальної записки обсягом 47 сторінок, що містить 33 ілюстрації, 3 таблиць, 5 додатків та 15 посилань.

Метою даного дипломного проєкту є розробка автономного пристрою сповіщення про повітряну тривогу з використанням мікроконтролера ESP32 та бездротової технології обміну даними. Основним призначенням пристрою є оперативне інформування користувачів про початок та відбій тривоги за допомогою звукових сигналів.

У проєкті було проведено аналіз існуючих рішень, визначено їх переваги та недоліки, на основі чого обрано архітектуру системи, протокол обміну, елементну базу і спосіб автономного живлення. Розроблено схему електричну принципову, макет пристрою та прототип друкованої плати. Також реалізовано програмну частину з періодичним опитуванням API та публікацією повідомлень у MQTT-канал.

Пристрій призначений для використання в побутових умовах, офісах та інших приміщеннях з наявністю Wi-Fi-з’єднання. Живлення забезпечується літієвим акумулятором типу 18650 з можливістю зарядки від порту USB. Рішення може бути масштабоване до мережі приймачів та адаптоване під різні сценарії використання.

Ключові слова: тривога, сповіщення, ESP32, MQTT, Wi-Fi, автономність, мікроконтролер, сигналізатор, API.

ANOTATION

This diploma project consists of an explanatory note of 47 pages, which includes 33 illustrations, 3 tables, 5 appendices, and 15 references.

The purpose of this diploma project is to develop an autonomous air raid alert notification device using the ESP32 microcontroller and wireless data exchange technology. The primary purpose of the device is to promptly inform users about the beginning and end of an alert through sound signals.

In the project, an analysis of existing solutions was conducted to identify their advantages and disadvantages. Based on this, the system's architecture, the MQTT exchange protocol, the component base, and methods for autonomous power supply were selected. We developed the electrical schematic diagram, a device prototype on a breadboard, and a printed circuit board (PCB) prototype. Additionally, the software part was implemented, featuring periodic API polling and message publication to an MQTT channel.

The device is designed for use in residential environments, offices, and other premises with Wi-Fi connectivity. Power is supplied by a 18650 type lithium battery with the ability to charge via a USB port. This solution can be scaled to a network of receivers and adapted for various usage scenarios.

Keywords: alarm, notification, ESP32, MQTT, Wi-Fi, autonomy, microcontroller, annunciator, API.

Пояснювальна записка
до дипломного проєкту
на тему: Сигналізатор повітряної тривоги

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

Повітряна тривога — ПТ

Мікроконтролер — МК

Акумуляторна батарея — АКБ

Друкована плата — ДП

Програмне забезпечення — ПЗ

Цифро-аналоговий перетворювач — ЦАП

ВСТУП

Життя українців та українок після 24 лютого 2022 року зазнало кардинальних і незворотних змін. Повномасштабна війна проти України поставила перед суспільством нові виклики, головним з яких стало збереження життя та здоров’я цивільного населення в умовах постійної загрози ракетних, артилерійських, авіаційних та інших атак. У повсякденному мовленні з’явилися нові терміни, такі як «вибухи», «приліт», «збиття», «тривога», які стали буденністю для більшості громадян.

Одним із найважливіших інструментів забезпечення безпеки цивільного населення є система сповіщення про загрози — передусім повітряної тривоги. Оперативне й достовірне інформування дає змогу людям швидко реагувати на небезпеку, перейти в укриття чи вжити інших заходів безпеки.

Наразі в Україні для передавання таких сигналів використовуються різні засоби, серед яких основними залишаються аналогові сирени на базі електродвигунів та відцентрових випромінювачів, а також цифрові системи, які транслюють звукові повідомлення через динаміки. Однак ці рішення, здебільшого, орієнтовані на зовнішній простір і великі відкриті території. Через наявність стін у приміщеннях звуковий сигнал тривоги часто або нечіткий, або взагалі не чути. Це створює додаткові ризики для життя та здоров’я людей, які перебувають усередині будівель.

Крім того, частина населення не завжди має смартфон поруч або не встигає зреагувати на Push-сповіщення, тому виникає потреба у спеціалізованих автономних пристроях, які б могли виконувати функцію локального сигналізатора — просто, надійно та з мінімальними вимогами до інфраструктури.

Метою даного дипломного проєкту є створення компактного, надійного та автономного приладу оповіщення про повітряну тривогу, спеціально адаптованого для використання у внутрішніх приміщеннях. Основним завданням пристрою є забезпечення миттєвого та чіткого інформування користувачів про початок або відбій повітряної тривоги за допомогою звукових і світлових сигналів. Пристрій має бути зручним у використанні, придатним до масштабування в рамках локальних мереж і здатним працювати без зовнішнього живлення протягом тривалого часу.

Розробка такого приладу не лише відповідає технічним викликам сучасності, а й сприяє підвищенню загального рівня безпеки населення та розвитку системи цивільного захисту в Україні.

1 АНАЛІЗ РИНКУ

1.1 Існуючі рішення

Незважаючи на постійні обстріли, важко знайти готовий до використання оповіщувач для офісного або побутового використання. Скоріше, це модулі, з якими необхідно проводити додаткові роботи з налаштування. Через відсутність прямих конкурентів, будемо розглядати найпопулярніші способи інформування про загрозу.

1.1.1 Elgato

Контролер повітряних тривог GSM/WEB зображений на рис. 1.1

Рисунок 1.1 — Elgato

Сумісний проєкт українських компаній «Elgato» та «Polymetrica», які розробили апаратну та програмну частину відповідно. Можливе підключення до сервера за допомогою Wi-Fi або GSM (необхідна SIM-картка). Створений для використання у приміщеннях, напруга живлення 10-14 В, містить вбудоване реле для сирени. Містить акумулятор, що забезпечує до 8 годин роботи при втраті зовнішнього живлення. Під час тривоги контролер надсилає команду на контролер, що вимикає електромагніти на магнітних замках дверей бомбосховища на час небезпеки. Досить багатофункціональний пристрій, але першочергово створювався саме як GSM-сигналізація для охорони приміщень. Варто зазначити, що це не готова система, тому для повноцінної роботи додаються сирени та, за бажанням, електромагнітні замки і датчики.

1.1.2 Око

Контролер тривог GSM «Око» зображено на рис. 1.2

Рисунок 1.2 — Око

Розробка українського виробника «Око» GSC-3.1, чимось схожа на минулий прилад, але коштує дешевше і має дещо менший функціонал. Пристрій побудовано на базі модуля GSM SIM800L. Використовується у приміщення, напруга живлення 10-15 В, має реле для керування сиреною та ще декілька цифрових входів і виходів. Не має вбудованого джерела живлення, проте має менший струм споживання, ніж «Elgato». Аналогічно потребує додавання сторонніх сирен і, за бажанням, датчиків.

Керування відбувається або через WEB-інтерфейс, або USSD-запитами через мобільний телефон.

1.1.3 Сповіщувач у формі мапи України

Дерев’яна карта зображена на рис. 1.3

Рисунок 1.3 — прилад від бренду _air.box_

Приклад світлового сигналізатора у формі мапи України, де кожна окрема область підсвічується певним кольором в залежності від наявності небезпеки. Керується через телеграм-канал, живиться блоком живлення. Існує декілька моделей мапи від цього виробника, проте відрізняються вони лише зовнішнім виглядом.

1.1.4 Додатки на смартфоні

Користувацький інтерфейс додатку «Повітряна тривога» зображений на рис. 1.4

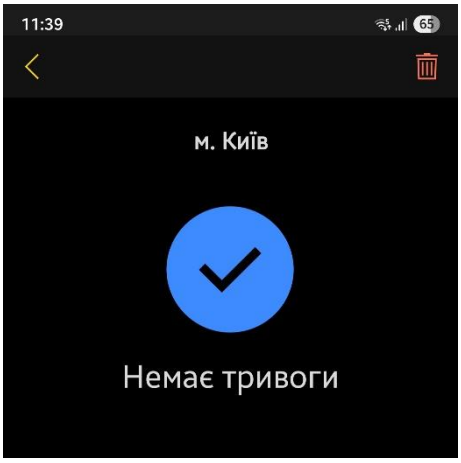


Рисунок 1.4 — скріншот Android-додатку «Повітряна тривога»

Існує для операційних систем Android та iOS, швидко та просто налаштовується під необхідний регіон і сповіщає власника пристрою за допомогою повідомлення, вібрації та звукового сигналу. Потребує підключення до Інтернету.

За час повномасштабного вторгнення було створено чимало подібних застосунків з різним додатковим функціоналом (наприклад, додавання свого звуку під час сповіщення), але одним сенсом. Серед мінусів можна зазначити необхідність завжди мати смартфон чи планшет під рукою, що не завжди можливо під час робочого процесу.

1.2 Порівняння аналогів і підсумки

Готових рішень для інформування населення існує чимала кількість. Направлені вони здебільшого на побутове використання. Для офісів, адміністративних будівель та інших місць масового скупчення людей розроблені тільки окремі модулі-контролери, які спочатку позиціонувалися як сигналізації для охорони. Після деяких модифікацій від виробника такий контролер дійсно може використовуватися для відстежування повітряних тривог, але вимагає додавання звукових або світлових сигналізаторів.

У продажі можна знайти і саморобні мапи зі світлодіодами, що підсвічують необхідні області певним кольором під час небезпеки, однак це скоріше функціональна побутова декорація, ніж серйозний сповіщувач.

Ключові характеристики оглянутих сигналізаторів зображено у табл. 1.1

Таблиця 1.1 — порівняння характеристик розглянутих сповіщувачів

Параметр	Наявність АКБ	Спосіб оповіщення	Потреба в Інтернет-з'єднанні	Сфера застосування
Elgato	Так	Звуковий чи світловий	Так	Багатоквартирні будинки, офіси, адмінбудівлі
Око	Ні	Звуковий чи світловий	Так	Багатоквартирні будинки, офіси, адмінбудівлі
Додаток ПТ	Так	Push-повідомлення, звуковий	Так	Особисте користування
Дерев'яна мапа	Ні	Світловий	Так	Побутове користування
ТБ / Радіо	Ні	Звуковий, світловий	Ні	Офісне користування

Таблиця з порівнянням основних характеристик дає чітке розуміння, що побутові пристрої або програмні засоби не пристосовані до офісного користування і навпаки. Наприклад, складність встановлення і налаштування систем на основі контролерів унеможливлює їх застосування вдома для пересічного користувача без технічної підготовки. Водночас мобільні додатки та прості світлові індикатори не забезпечують достатнього рівня помітності й надійності для використання у великих приміщеннях з підвищеним рівнем шуму або значною кількістю людей.

Також деякі існуючі рішення не мають повноцінної автономності — при відсутності електроенергії чи інтернету вони втрачають функціональність. Крім того, відсутність звукових повідомлень з чітким інформуванням суттєво знижує ефективність таких приладів у критичних ситуаціях.

Отже, на ринку бракує універсального, простого у використанні та автономного пристрою, спеціально адаптованого для приміщень, що одночасно забезпечує звукове й візуальне інформування та не потребує додаткового технічного обслуговування. Саме розробка такого рішення і стала предметом даної дипломної роботи.

2 СИНТЕЗ СХЕМИ ПРИСТРОЮ

2.1 Створення структурної схеми

Структурну схему пристрою зображено на рис. 2.1

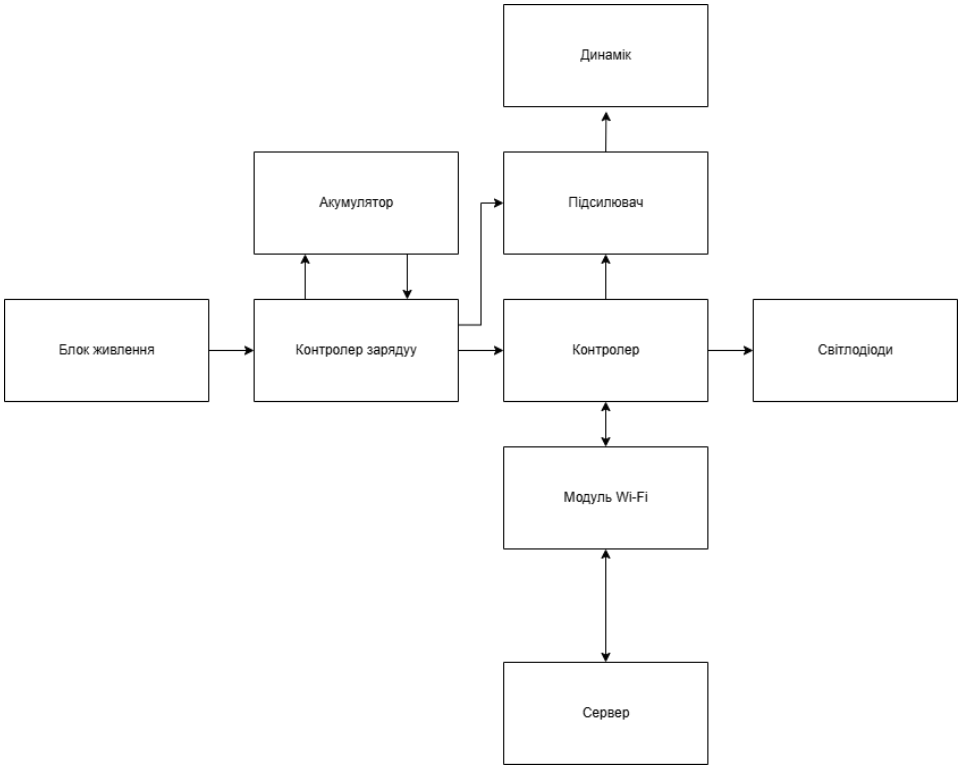


Рисунок 2.1 — структурна схема

Основою є мікроконтролер (МК), який за допомогою вбудованого Wi-Fi модуля підключається до сервера і отримує від нього інформацію. Приймач періодично отримує повідомлення про небезпеку або її відсутність. У разі тривоги чи відбою відбувається світлове та звукове інформування.

За живлення приймача відповідає під'єднаний до контролера заряду акумулятор. Заряджання пристрою забезпечується зовнішнім блоком живлення. До схеми додані світлодіоди для відображення наявності чи відсутності повітряної небезпеки та індикації про стан підключення до сервера. Звукове оповіщення організовано динаміком, що через підсилювач підключений до мікроконтролера. Звуки тривоги та відбою зберігаються у вбудованій постійній пам’яті мікроконтролера.

2.2 Створення схеми електричної принципової

Електричну схему принципову зображено на рис. 2.2.

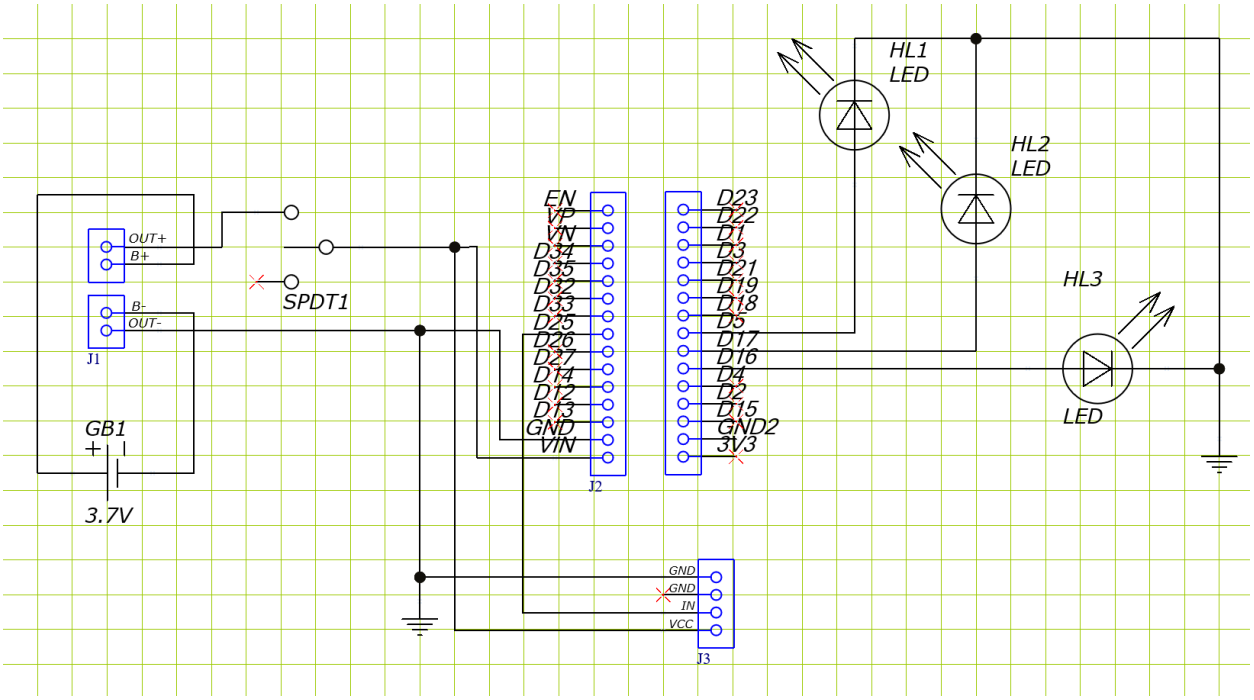


Рисунок 2.2 — схема електрична принципова приймача повітряної тривоги

Схема містить основні функціональні елементи для живлення, керування індикацією та взаємодію з МК.

Джерелом живлення є літій-іонний акумулятор напругою 3,7 В (GB1), що підключений до контролера заряду через роз'єм J1. Для вмикання та вимикання пристрою встановлений перемикач SPDT1.

До роз'єму J2 підключений мікроконтролер, що отримує, оброблює та виводить сигнал про тривогу та стан підключення до сервера. Світлова індикація забезпечується світлодіодами HL1, HL2, HL3.

Роз'єм J3 використовується підсилювачем, що також заживлений акумулятором, для виводу звука на динамік.

Схема адаптована для виконання на односторонній друкованій платі, що має зменшити собівартість виробництва.

2.3 Підбір елементної бази та обґрунтування

2.3.1 Контактні роз’єми

Контактні роз’єми або пін хедери (англ. pin header) використовуються для швидкого, простого та надійного підключення модулів до друкованої плати. Усі необхідні однорядні гнізда з відстанню між контактами 2,54 мм зображені на рис. 2.3

Рисунок 2.3 — однорядні гнізда на 2, 4 та 15 контактів відповідно

2.3.2 Перемикач

Перемикач зображено на рис. 2.4

Рисунок 2.4 — повзунковий перемикач SS12D07VG4

Підходить для комутації напруги у різних пристроях, має невеликий розмір та великий запас по струму та напрузі.

2.3.3 Тримач для акумулятора

Холдер (тримач) для АКБ формату 18650 зображено на рис. 2.5

Рисунок 2.5 — холдер для акумуляторної батареї 18650

Звичайна деталь для зручності заміни АКБ.

2.3.4 Мікроконтролер

Мікроконтролер ESP32 (зображений на рис. 2.6) від компанії Espressif Systems, що широко використовується у сучасних IoT-приладах і не тільки. Основними перевагами є низьке енергоспоживання, підтримує протокол I^2S, який знадобиться для виводу звуку, та має вбудований модуль Wi-Fi і Bluetooth. Сюди ж можна додати і низьку вартість, що добре вплине на загальну собівартість оповіщувача.

Рисунок 2.6 — плата розробника ESP-WROOM-32 30 pin

2.3.5 Підсилювач

Підсилювач LM386, розпаяний на платі, зображено на рис. 2.7

Рисунок 2.7 — підсилювач LM386

До основних переваг можна віднести низьку вартість і можливість підлаштовувати підсилення за допомогою змінного резистора. Робоча напруга у діапазоні 5…12 В, опір навантаження 8…32 Ом, посилення вхідного сигналу у 200 разів.

2.3.6 Динамік

Динамік зображений на рис. 2.8

Рисунок 2.8 — круглий динамік

Діаметр 40мм, товщина 5мм, потужність 1 Вт і опір 8 Ом. Своїми характеристиками повністю підходить обраному підсилювачу. Невисока ціна, нормальна якість звучання.

2.3.7 Контролер заряду

Розпаяний на платі контролер заряду TP4056 зображено на рис. 2.9

Рисунок 2.9 — контролер заряду

Цей контролер призначений для заряду акумуляторів формату 18650. Напруга і струм живлення — 5 В і 1 А відповідно. Має захист від перезаряду і перерозряду і короткого замкнення. Містить два світлодіоди для індикації стану заряджання АКБ.

2.3.8 Акумулятор

АКБ зображена на рис. 2.10

Рисунок 2.10 — акумулятор формату 18650

Один з найзручніших форматів акумуляторів, який, за наявності холдера, можна легко замінити на інший. Ємність 2400 мА*год, що забезпечить приблизно до 8 годин роботи за відсутності зовнішнього джерела живлення.

2.3.9 Світлодіоди

Для індикації стану підключення до сервера, тривоги і відбою до схеми додано три прямокутні світлодіоди синього, зеленого, та червоного кольорів відповідно. Приклад зображено на рис. 2.11

Рисунок 2.11 — прямокутний зелений світлодіод

Напруга живлення 2.5 В, максимальний струм споживання 20 мА, керуються за допомогою мікроконтролера.

2.3.10 Блок живлення

Для заряджання приймача може використовуватися будь-який блок живлення з напругою 5 В і струмом 1 А, роз’єм USB-C.

2.3.11 Сервер

У якості сервера було обрано Raspberry Pi 5, що зображено на рис. 2.12

Рисунок 2.12 — Raspberry Pi 5

Багатофункціональний мікрокомп’ютер з низьким енергоспоживанням та доволі простим налаштування, що відмінно підходить під поставлене завдання. Має безкоштовну та загальнодоступну операційну систему та декілька вбудованих у неї середовищ програмування.

3 РОЗРОБКА ДРУКОВАНОГО ВУЗЛА ТА КОНСТРУКЦІЇ ПРИСТРОЮ

3.1 Створення друкованої плати

3.1.1 Обґрунтування створення друкованої плати

Після складання макету (рис. 3.1) було вирішено створити друковану плату, оскільки використання макетної плати не може забезпечити достатньої надійності з’єднань між елементами та сильно збільшує розмір виробу, що унеможливлює використання компактного корпусу та масштабування виробництва.

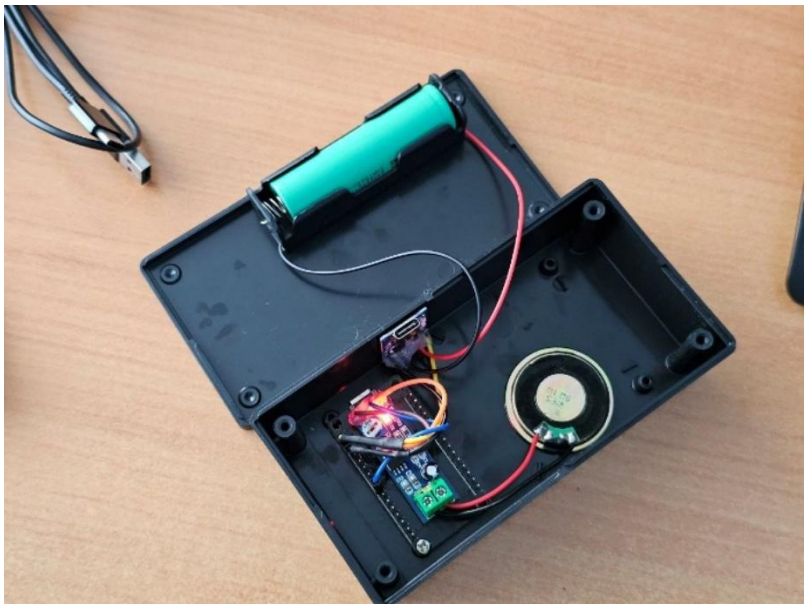


Рисунок 3.1 — один із перших макетів сигналізатора

Плата має бути основою, куди будуть підключатися всі модулі та, у випадку несправності якогось із них, швидко та без пайки замінюватися. Така можливість забезпечена наявністю мама-тато хедерів на платі та модулях відповідною.

3.1.2 Обґрунтування класу точності

Враховуючи можливості сучасних українських виробників друкованих плат, було обрано 4 клас точності. Щільний монтаж елементів відсутній, доріжки широкі, що спрощує виробництво та позитивно впливає на ціну.

3.1.3 Обґрунтування матеріалів виконання ДП

Основою є фольгований склотекстоліт FR-4, як сучасний композитний матеріал з гарними механічними, тепловими та діелектричними властивостями.

У якості припою використовується паяльна паста «НАЗВА», що має низьку температуру плавлення та невисоку ціну.

3.1.4 Обчислення розмірів друкованої плати

Мінімальна площа плати обчислюється виразом, у якому площа футпринтів (посадкових місць елементів) множиться на їх кількість та певний коефіцієнт. Формула зображена у виразі 3.1.

$$\sum_{k=1}^{n} S \cdot n \cdot k, \quad (3.1)$$

де S — загальна площа посадкових місць, n — кількість елементів, а k — коефіцієнт, що залежить від кількості виводів. Для 8 виводів $k = 2$; для діапазону 2 — 8 $k = 1{,}5$; а при 2 виводах $k = 1$. Усі значення та результати розрахунків занесені до табл. 3.1.

Таблиця 3.1 — список елементів, їх площі та загальна площа

Найменування	Площа	Коефіцієнт	Кількість	Сума
Виводи ESP32	113,801	2	1	227,602
Виводи TP4056	19,313	1,5	1	28,9695
Виводи LM386	35,061	1,5	1	52,5915
Перемикач	58,789	1	1	58,789
Світлодіоди	41,165	1	3	123,495
Акумулятор	1760,049	1	1	1760,049
Кріплення	23,04	1	4	92,16
S	—			2343,65

Отже, мінімальна необхідна площа плати складає 2343мм, однак враховуючи відстані між виводами ESP32 та TP4056, площу буде дещо збільшено.

3.1.5 Обчислення ширини друкованих провідників

Враховуючи специфікації модулів, окремих елементів і технічні можливості виробника ДП, максимальний струм для заряджання акумулятора буде I_{max} = 1 А, а для сигнальних виводів не буде перебільшувати 50 мА, тому I_{min} = 0,1 А. Результати розрахунків внесено до табл. 3.2.

Таблиця 3.2 — обчислення ширини друкованих провідників

Вид лінії	Напруга (В)	Струм (А)	Ширина провідника (мм)	Зазор (мм)
Живлення	5	1	1,5	0,25
Сигнал	3,3	0,1	0,5	0,25

3.1.6 Обчислення ширини друкованих провідників

Трасування провідників, як і весь процес створення плати, відбувалося у програмному середовищі “Altium Designer”. Одностороння плата, 104 мм у довжину та 55 мм у ширину. Земля виконана у вигляді полігона на одній стороні ДП. Результати трасування верхнього та нижнього шарів наведено у рис. 3.2 та рис. 3.3.

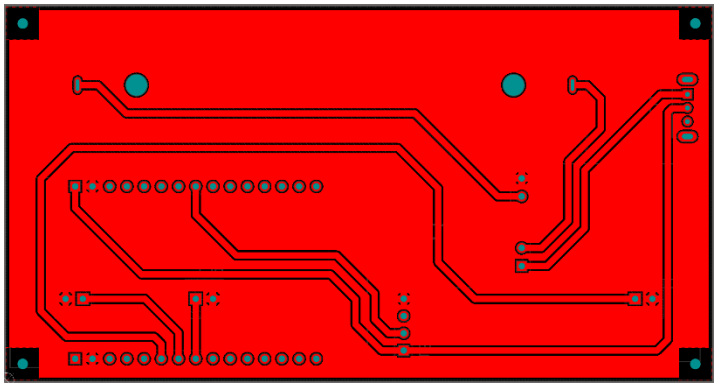


Рисунок 3.2 — трасування верхнього шару

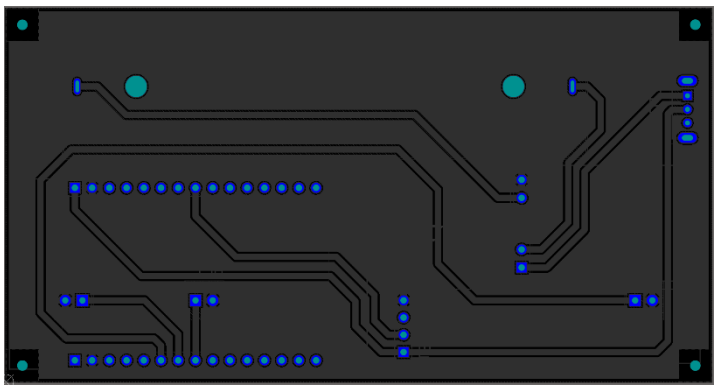


Рисунок 3.3 — трасування нижнього шару

4 РОЗРОБКА ТРИВИМІРНОЇ МОДЕЛІ КОРПУСУ ПРИЙМАЧА

За основу було взято тривимірну модель друкованої плати, що зображено на рис. 4.1.

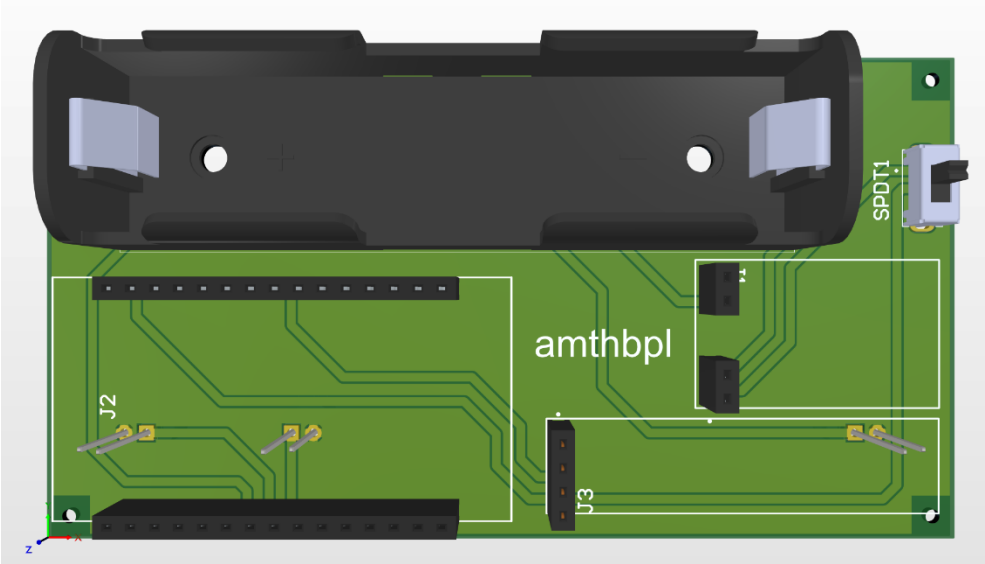


Рисунок 4.1 — 3D-модель ДП з розпаяним тримачем АКБ, роз’ємами, перемикачем та світлодіодами з іншого боку

Корпус має задовольняти наступні потреби: решітка для динаміка, отвір для порту USB-C для заряджання, три отвори під світлодіоди індикації, отвір для перемикача і наявність кришки, що можна знімати для заміни окремих модулів та АКБ.

У програмному середовищі Autodesk Fusion 360 розроблено тривимірну модель (рис. 4.2, 4.3, 4.4)

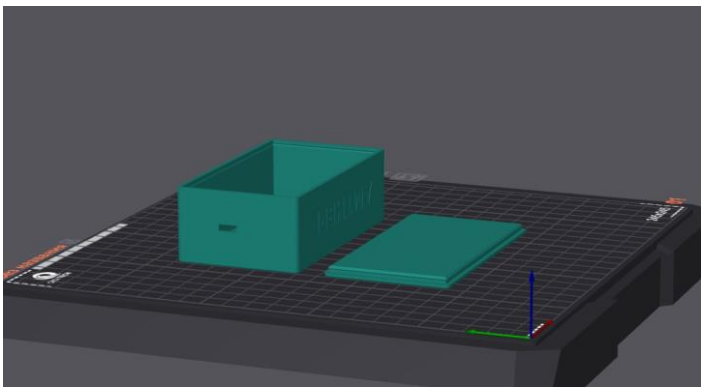


Рисунок 4.2 — модель корпусу для друку на 3D-принтері

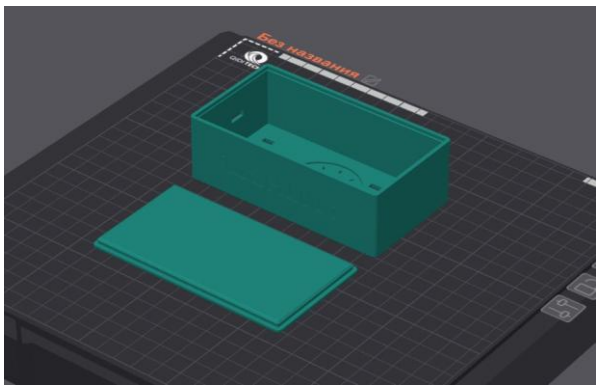


Рисунок 4.3 — вид з іншого ракурсу

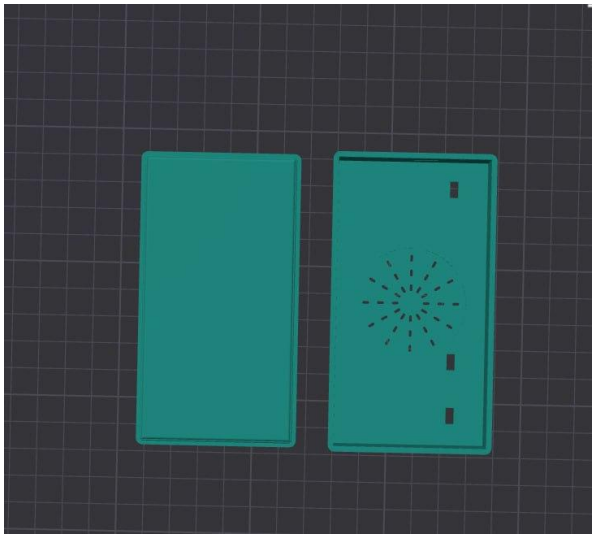


Рисунок 4.4 — вид на внутрішню частину

Товщина стінок 2 мм, довжина 109 мм, ширина 59 мм, висота 35 мм. Матеріал PETG — модифікований поліетилентерефталат з додаванням гліколю, що відрізняє його від звичайного PET більшою міцністю та ударостійкістю. Має менші тенденції до деформації під час друку, ніж ABS.

Отвори для динаміка виконано у вигляді овальних отворів з кругової симетрією. Присутнє заглиблення для розташування динаміка та отвори під світлодіоди стану роботи пристрою. Усі відстані отворів та їх розміри розраховано заздалегідь. Кришка має засувки для створення збірно-розбірної конструкції. На бічну стінку додано декоративний підпис.

Корпус надруковано на 3D-принтері, результат зображено на рис. 4.5

Рисунок 4.5 — готовий корпус з увімкненими світлодіодами

Розміщену плату зображено на рис. 4.6

Рисунок 4.6 — плата всередині кейса

5 СТВОРЕННЯ ПРОГРАМНОЇ ЧАСТИНИ СИГНАЛІЗАТОРА

Програмну частину можна умовно розділити на код для сервера і для приймача, кожен з них розглянемо окремо.

5.1 Організація роботи сервера

Надавачем інформація для приймача є мікрокомп'ютер Raspberry Pi 5 зі встановленою операційною системою Raspberry Pi OS, який цілодобово працює. Весь код розміщено у Додатку Б.

5.1.1 Вибір протоколу передачі даних

Серед множини протоколів було обрано саме MQTT — Message Queuing Telemetry Transport, схема принципу роботи зображена на рис. 5.1.

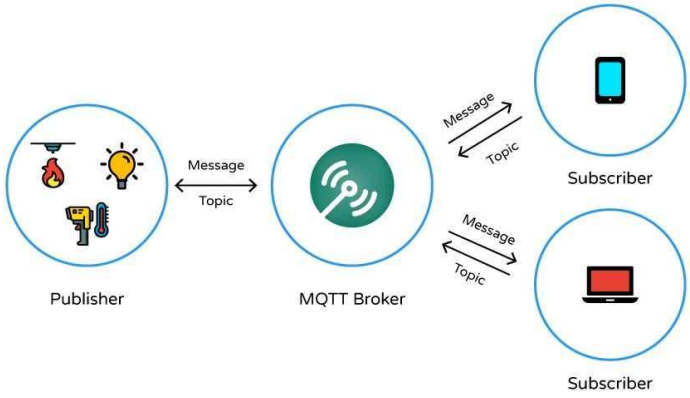


Рисунок 5.1 — схема роботи протоколу MQTT

Він створений для обміну повідомленнями і має переваги над своїми конкурентами: повна підтримка на МК ESP32, низьке енергоспоживання через постійне з’єднання, використовує мало трафіку та має функцію retain, що відіграє важливу роль у роботі приймача. Працює протокол по принципу брокер-клієнт (тобто надавач інформації та її приймач). Це дозволяє централізовано та швидко передавати повідомлення багатьом клієнтам.

5.1.2 Вибір API

API (інтерфейс прикладного програмування) є посередником для роботи однієї програмі з іншою. Для сигналізатора тривоги використовується безкоштовний та загальнодоступний український сервіс від ubilling.net.ua.

Брокер звертається до нього та отримує відповідь у форматі JSON (текстовий формат обміну даними), яку потом обробляє та надсилає до приймача.

5.1.3 Використані бібліотеки

Бібліотеки — це набори функцій та інструментів, що загалом спрощують написання коду. Для брокера використовуються наступні бібліотеки: requests (для надсилання запитів до API, JSON для обробки відповіді від API, paho.mqtt.client для підключення до MQTT-брокера для надсилання повідомлень, time для реалізації циклів та періодичного опитування сервера.

5.1.4 Виконання коду та налагодження

Код виконується у терміналі операційної системи. При наявності змін по стану тривоги відображається відповідне повідомлення у вікні терміналу та надсилається повідомлення до приймача. Знімок екрану зі станом небезпеки зображено на рис. 5.1.

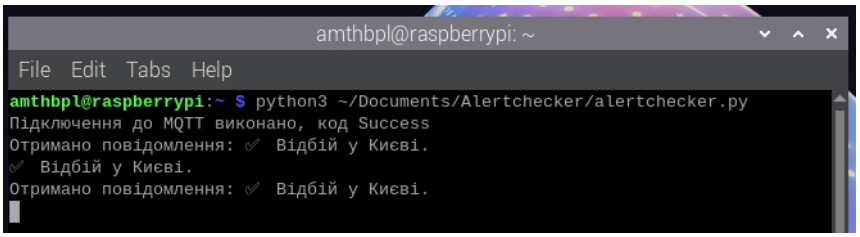


Рисунок 5.1 — отримані повідомлення у процесі виконання коду

Повідомлення з мікрокомп’ютера надсилаються і на самого себе у тому числі, аби розуміти, чи працює брокер взагалі. У процесі створення коду також виводилась додаткова інформація з розшифрованим повідомленням від API, проте у фінальному варіанті було прийнято рішення від цього відмовитись

5.1.5 Загальний принцип роботи та особливості

Брокер раз на 10 секунд звертається до API, отримує JSON-файл, оброблює його та надсилає повідомлення про тривогу, відбій тривоги та помилку підключення до API (за необхідності) у певний топік (канал), на який підписуються клієнти під час успішного з’єднання.

Після довгого тестування були виявлені та виправлені деякі особливості, як от проблема з приймачем, який при підключенні у проміжку між

надсиланнями не отримував і не виводив нічого. Тут знадобилася одна із функцій MQTT — retain. Вона дозволяє клієнту при підключенні або перепідключенні одразу отримувати останню актуальну інформацію від брокера.

Загалом, програма зациклено виконується без участі людини і за час тестування недоліки виявлені не були.

Повний алгоритм роботи зображено на рис. 5.2

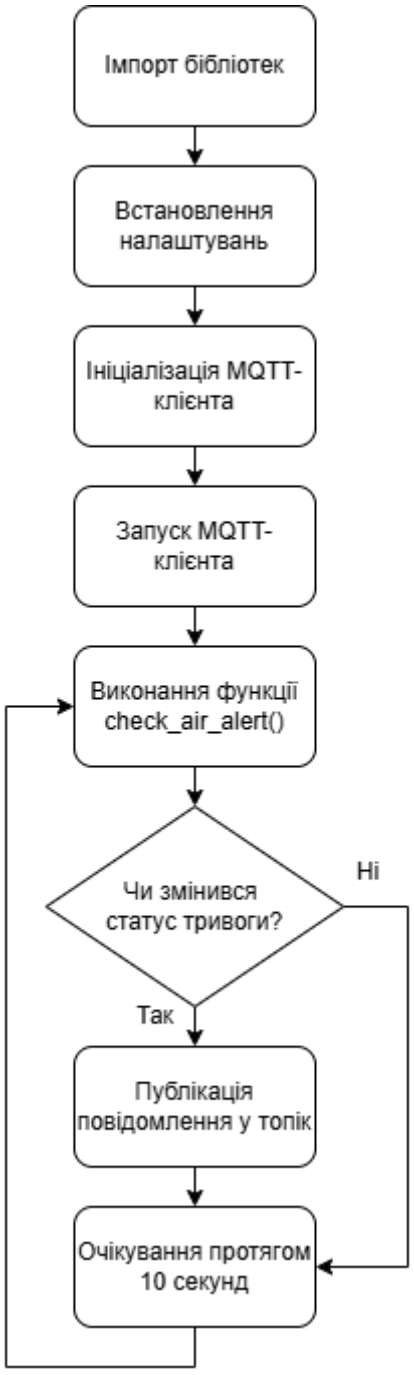


Рисунок 5.2 — алгоритм роботи коду на сервері

5.2 Організація роботи приймача

5.2.1 Середовище програмування

Існує велика кількість ПЗ для програмування мікроконтролерів ESP32, але вибір зупинився на Arduino IDE (рис. 5.3) через простий і зрозумілий інтерфейс, можливість швидко встановлювати різні версії бібліотек, повністю підтримує обраний МК та доступний на Raspberry Pi OS, що дозволяє не використовувати сторонні комп'ютери для прошивання.

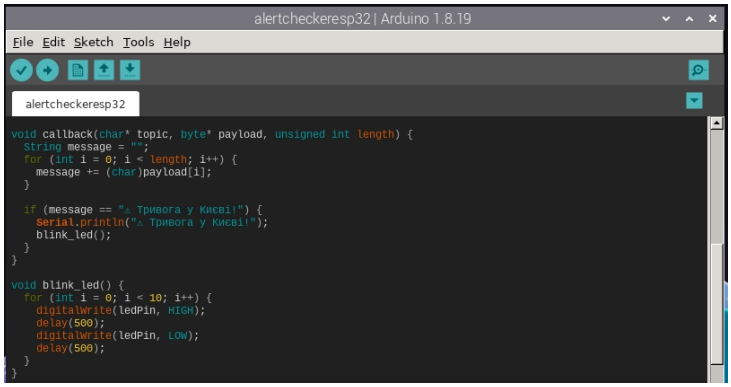


Рисунок 5.3 — користувацький інтерфейс Arduino IDE

5.2.2 Мова програмування

Основною мовою програмування для обраного типу МК є C++. Серед переваг можна зазначити компіляцію у машинний код, що позитивно впливає на продуктивність, доступ низького рівня до апаратури для керування периферійними пристроями та широку спільноту, що постійно генерує безліч нових бібліотек та відкритих проєктів.

5.2.3 Використані бібліотеки

Для коректної роботи написаної програми використовуються дві бібліотеки: WiFi.h — стандартний засіб для підключення до Wi-Fi мереж, PubSubClient.h для роботи з MQTT-протоколом (дозволяє підписуватися на топік та отримувати повідомлення), XT_DAC_Audio.h — бібліотека для відтворення WAV-файлів через вбудований цифро-аналоговий перетворювач у ESP32.

5.2.4 Виведення звука

Обраний МК містить вбудований цифро-аналоговий перетворювач та 4 МБ постійної пам’яті. Це можна використати для виведення аудіозапису без використання стороннього ЦАП та SD-картки, що робить конструкцію простішою, а собівартість меншою.

Після додавання бібліотеки XT_DAC_Audio.h необхідно встановити додаток Audacity (рис. 5.4).

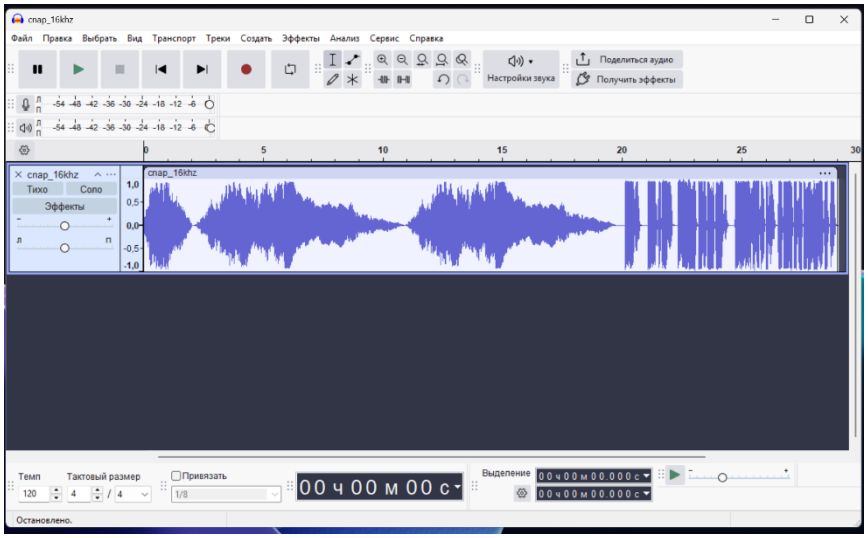


Рисунок 5.4 — користувацький інтерфейс Audacity

Звукозапис треба стиснути через невеликий об’єм пам’яті та апаратні обмеження МК. За необхідності вирізаємо необхідну частину запису. При експорті обираємо тип кодування Unsigned 8-bit PCM (формат WAV-файлу, що займає менше пам’яті), частота дискретизації 16 кГц як компроміс між якістю та розміром, один канал (моно).

Тепер переходимо до додатку HxD — гекс-редактору для подальшого редагування отриманого WAV-файлу (рис. 5.5). Відкриваємо попередньо стиснутий запис, обираємо весь отриманий текст комбінацією клавіш Ctrl+A та копіюємо як C, тобто масив байтів. Це необхідно для прямого вставлення в код на C, або, як у даному випадку, C++. Отриманий код звукозапису вставляємо в Arduino IDE, що зображено на рис. 5.6.

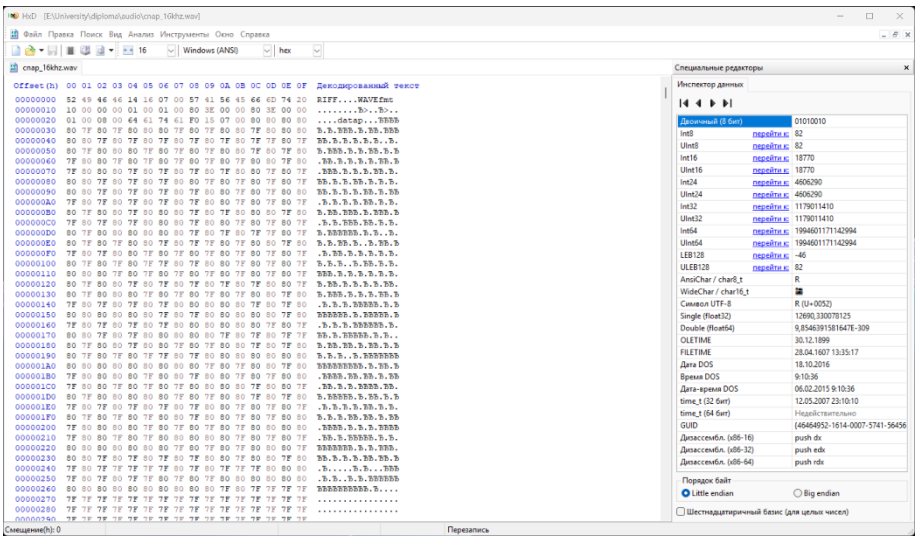


Рисунок 5.5 — користувацький інтерфейс HxD

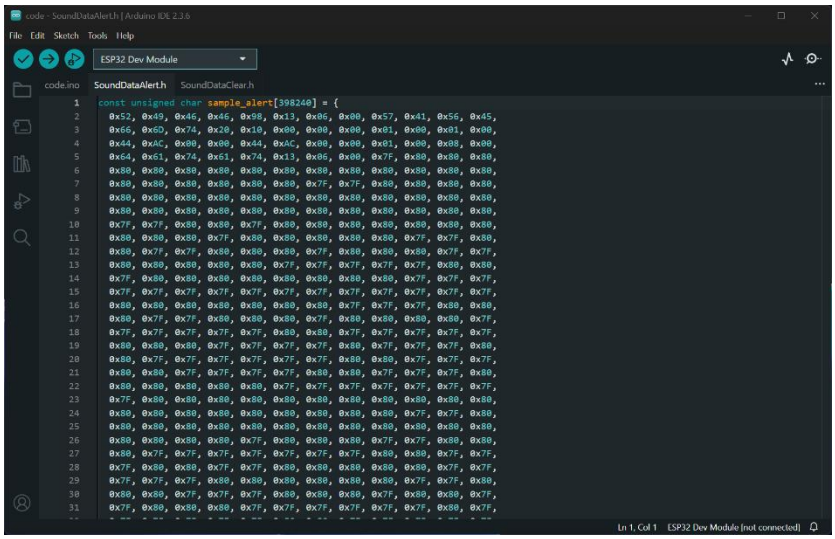


Рисунок 5.6 — код звукозапису в Arduino IDE

5.2.5 Загальний принцип роботи та особливості

Програма на приймачі, як і на сервері, зациклена. Клієнт підключається до бездротової мережі, а потім до сервера, від якого одразу отримує повідомлення про наявність чи відсутність небезпеки. Залежно від отриманого повідомлення, пристрій або сигналізує червоним світлодіодом та звукозаписом «Увага, у м. Києві оголошено повітряну тривогу, просимо вас зберігати спокій та прямувати до тимчасового укриття», або зеленим кольором та мелодією відбою, або синім мерехтінням у випадку проблем з підключенням до брокера.

Приймач раз на 10 секунд буде перевіряти, чи доступний брокер, а у випадку з перебоями бездротового підключення буде завжди намагатися перепід’єднатися.

Повний код програми розміщено у Додатку В, конвертовані звукозаписи у Додатках Г і Д.

Алгоритм роботи зображено на рис. 5.7

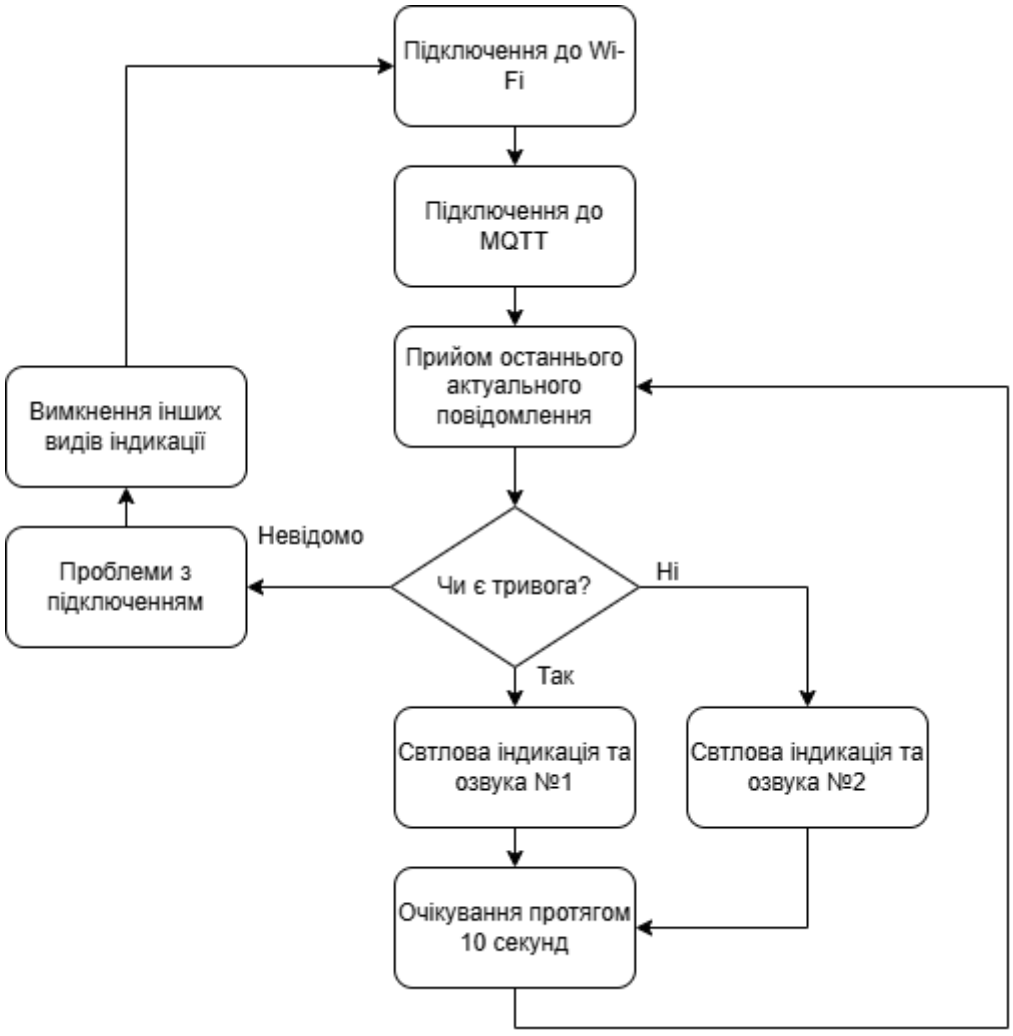


Рисунок 5.7 — алгоритм роботи коду приймача

ВИСНОВКИ

У межах виконання дипломного проєкту було виконано повний цикл розробки пристрою оповіщення про повітряну тривогу.

Було проведено аналіз існуючих рішень, виявлено слабкі та сильні сторони конкурентів. Зроблено висновок, що аналоги мають складне встановлення та налаштування, або орієнтовані виключно під побутове використання.

Розроблено архітектуру приладу на основі мікроконтролера ESP32 з підтримкою Wi-Fi, контролера заряду TP4056, підсилювача LM386 та інших деталей. Створено структурну та електричну принципову схеми, підібрано необхідну елементну базу з урахуванням ціни, енергоспоживання, компактності та надійності.

Спроєктовано та виготовлено друковану плату, враховуючи допуски та електричне навантаження, після чого створено тривимірну модель. Уся конструкція залишається модульною, що покращує загальну ремонтопридатність.

Написано код для приймача та сервера з урахуванням можливих нюансів з перебоями підключення приймача до Wi-Fi. Вдалося зберігати та виводити звук за допомогою МК без використання сторонніх модулів задля спрощення схеми та зниження ціни. Загалом, схема не потребує зовнішнього втручання для роботи.

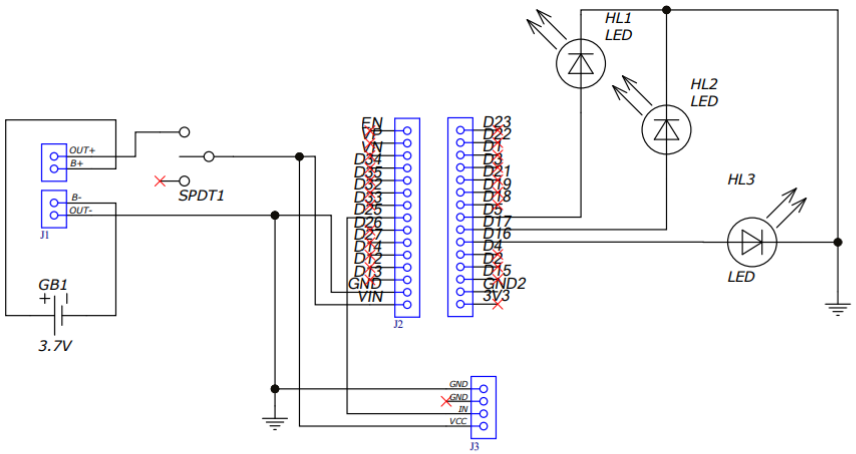
Виконано тестування прототипу пристрою, виконано налагодження програмної та апаратної частини.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Raspberry Pi. Getting Started [Електронний ресурс]. – Режим доступу: https://www.raspberrypi.com/documentation/computers/getting-started.html
2. ESP-32 Based Audio Player [Електронний ресурс]. – Режим доступу: https://www.instructables.com/ESP-32-Based-Audio-Player/
3. Samacsys. Altium Designer Library Instructions [Електронний ресурс]. – Режим доступу: https://www.samacsys.com/altium-designer-library-instructions/
4. Würth Elektronik. PHD 2.54 Socket Header 6130XX11821 [Електронний ресурс]. – Режим доступу: https://www.we-online.com/en/components/products/PHD_2_54_SOCKET_HEADER_6130XX11821?sq=61301011821
5. Espressif Systems. ESP32 Technical Reference Manual [Електронний ресурс]. – Режим доступу: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
6. MQTT Version 3.1.1 Plus Errata 01. OASIS Standard [Електронний ресурс]. – Режим доступу: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
7. Altium LLC. Altium Designer Documentation [Електронний ресурс]. – Режим доступу: https://www.altium.com/documentation/altium-designer
8. Autodesk. Fusion 360 Documentation [Електронний ресурс]. – Режим доступу: https://help.autodesk.com/view/fusion360/ENU/
9. Python Software Foundation. Requests: HTTP for Humans [Електронний ресурс]. – Режим доступу: https://docs.python-requests.org/en/latest/
10. Python Software Foundation. The JSON module documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/json.html
11. Eclipse Foundation. Eclipse Paho MQTT Client for Python [Електронний ресурс]. – Режим доступу: https://www.eclipse.org/paho/clients/python/

13. Espressif Systems. Arduino-ESP32 Documentation [Електронний ресурс]. – Режим доступу: https://docs.espressif.com/projects/arduino-esp32/en/latest/
14. Polymetrica. Elgato Air Alert [Електронний ресурс]. – Режим доступу: https://polymetrica.com/
15. Око GSM контролери [Електронний ресурс]. – Режим доступу: https://oko.ua/

ДОДАТОК А. СХЕМА ЕЛЕКТРИЧНА ПРИНЦИПОВА



Зм.	Арк.	№ докум.	Підп.	Дата	РЕ11.468232.001 Е3			
Розробив		Слинчук			Сигналізатор повітряної тривоги	Літ.	Маса	Масштаб
Перевірив					Схема електрична принципова			
Т.контр.						Аркуш	Аркушів	
Н.контр.		Стешенко						
Затв.								

Перв. застосування | Довідк. № | Підп. і дата | Інв. № дубл. | Зам. інв. № | Підп. і дата | Інв. № оріг.

ДОДАТОК Б

```
import requests
import json
import paho.mqtt.client as mqtt
import time

API_URL = "https://ubilling.net.ua/aerialalerts/?json=true"

MQTT_BROKER = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC = "air_alert"

last_alert_state = None
mqtt_client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
def connect_mqtt():
    while True:
        try:
            mqtt_client.connect(MQTT_BROKER, MQTT_PORT, 60)
            print("MQTT Connected!")
            break
        except Exception as e:
            print(f"MQTT Connection failed: {e}")
            time.sleep(5)

connect_mqtt()
mqtt_client.loop_start()

def check_air_alert():
    global last_alert_state
```

```
        try:
            response = requests.get(API_URL, timeout=5)
            response.raise_for_status()

            data = response.json()
            states = data.get('states', {})
            kyiv_state = states.get('м. Київ', {})

            if not kyiv_state:
                print("Дані по Києву не знайдено.")
                return

            kyiv_alert = kyiv_state.get('alertnow', False)

            if kyiv_alert != last_alert_state:
                message = "⚠ Тривога у Києві!" if kyiv_alert else "☑ Відбій у Києві!"
                print(f"Нове повідомлення: {message}")

                mqtt_client.publish(MQTT_TOPIC, message, retain=True)
                last_alert_state = kyiv_alert

        except requests.exceptions.RequestException as e:
            print(f"Помилка при з'єднанні з API: {e}")
        except json.JSONDecodeError:
            print("Помилка парсингу JSON.")

    while True:
        check_air_alert()
        time.sleep(10)
```

ДОДАТОК В

```
#include "SoundDataAlert.h"
#include "SoundDataClear.h"
#include "XT_DAC_Audio.h"
#include <WiFi.h>
#include <PubSubClient.h>

const char* ssid        = "BOOS";
const char* password    = "misha0666417051";
const char* mqtt_server = "176.36.203.130";
const int   mqtt_port   = 1883;
const char* mqtt_topic  = "air_alert";

WiFiClient espClient;
PubSubClient client(espClient);
XT_Wav_Class SoundAlert(sample_alert);
XT_Wav_Class SoundClear(sample_clear);
XT_DAC_Audio_Class DacAudio(25, 0);

const int LED_ALERT = 17;
const int LED_CLEAR = 5;
const int LED_ERROR = 16;

enum EventState {
  STATE_IDLE,
  STATE_ALARM_LED,
  STATE_ALARM_SOUND,
  STATE_ALARM_STEADY,
  STATE_CLEAR_LED,
  STATE_CLEAR_SOUND,
```

```
  STATE_CLEAR_STEADY
};
EventState state = STATE_IDLE;

unsigned long event_start_time = 0;
unsigned long last_blink_time = 0;
unsigned long last_reconnect_time = 0;
bool blink_state = false;

bool isPlaying = false;

void setup_wifi() {
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" Connected");
}

void callback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (unsigned int i = 0; i < length; i++) {
    message += (char)payload[i];
  }
  Serial.print("Received: ");
  Serial.println(message);

  if (message == "⚠ Тривога у Києві!") {
```

```
    state = STATE_ALARM_LED;
    event_start_time = millis();
    digitalWrite(LED_CLEAR, LOW);
  }
  else if (message == "☑ Відбій у Києві!") {
    state = STATE_CLEAR_LED;
    event_start_time = millis();
    digitalWrite(LED_ALERT, LOW);
  }
}

void reconnect() {
  if (!client.connected()) {
    if (WiFi.status() != WL_CONNECTED) return;

    if (client.connect("ESP32Client")) {
      Serial.println("MQTT connected");
      client.subscribe(mqtt_topic);
    } else {
      Serial.print("MQTT failed: ");
      Serial.println(client.state());
    }
  }
}

void start_play(XT_Wav_Class* sound) {
  DacAudio.Play(sound);
  isPlaying = true;
}
```

```
void handle_play() {
  if (isPlaying) {
    DacAudio.FillBuffer();
    if (!SoundAlert.Playing && !SoundClear.Playing) {
      isPlaying = false;
    }
  }
}

void setup() {
  Serial.begin(115200);
  pinMode(LED_ALERT, OUTPUT);
  pinMode(LED_CLEAR, OUTPUT);
  pinMode(LED_ERROR, OUTPUT);
  digitalWrite(LED_ALERT, LOW);
  digitalWrite(LED_CLEAR, LOW);
  digitalWrite(LED_ERROR, LOW);

  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
  client.setCallback(callback);
}

void loop() {
  unsigned long now = millis();

  if (WiFi.status() != WL_CONNECTED || !client.connected()) {
    digitalWrite(LED_ALERT, LOW);
    digitalWrite(LED_CLEAR, LOW);
```

```
  if (now - last_blink_time >= 500) {
    blink_state = !blink_state;
    digitalWrite(LED_ERROR, blink_state ? HIGH : LOW);
    last_blink_time = now;
  }
  if (now - last_reconnect_time >= 5000) {
    reconnect();
    last_reconnect_time = now;
  }
  return;
}

digitalWrite(LED_ERROR, LOW);
client.loop();
handle_play();

switch (state) {
  case STATE_IDLE:
    digitalWrite(LED_ALERT, LOW);
    digitalWrite(LED_CLEAR, LOW);
    break;

  case STATE_ALARM_LED:
    if (now - event_start_time < 5000) {
      if (now - last_blink_time >= 500) {
        blink_state = !blink_state;
        digitalWrite(LED_ALERT, blink_state ? HIGH : LOW);
        last_blink_time = now;
      }
    } else {
```

```
    state = STATE_ALARM_SOUND;
    start_play(&SoundAlert);
  }
  break;

case STATE_ALARM_SOUND:
  if (!isPlaying) {
    state = STATE_ALARM_STEADY;
  }
  break;

case STATE_ALARM_STEADY:
  digitalWrite(LED_ALERT, HIGH);
  digitalWrite(LED_CLEAR, LOW);
  break;

case STATE_CLEAR_LED:
  if (now - event_start_time < 5000) {
    if (now - last_blink_time >= 500) {
      blink_state = !blink_state;
      digitalWrite(LED_CLEAR, blink_state ? HIGH : LOW);
      last_blink_time = now;
    }
  } else {
    state = STATE_CLEAR_SOUND;
    start_play(&SoundClear);
  }
  break;

case STATE_CLEAR_SOUND:
```

```
      if (!isPlaying) {
        state = STATE_CLEAR_STEADY;
      }
      break;

    case STATE_CLEAR_STEADY:
      digitalWrite(LED_CLEAR, HIGH);
      digitalWrite(LED_ALERT, LOW);
      break;
  }
}
```

ДОДАТОК Г

```
const unsigned char sample_alert[398240] = {
  0x52, 0x49, 0x46, 0x46, 0x98, 0x13, 0x06, 0x00, 0x57, 0x41, 0x56, 0x45,
  0x66, 0x6D, 0x74, 0x20, 0x10, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00,
  0x44, 0xAC, 0x00, 0x00, 0x44, 0xAC, 0x00, 0x00, 0x01, 0x00, 0x08, 0x00,
  0x64, 0x61, 0x74, 0x61, 0x74, 0x13, 0x06, 0x00, 0x7F, 0x80, 0x80, 0x80,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
  0x7F, 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
  0x80, 0x80, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x80,
  0x80, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x7F, 0x7F,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80,
  0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F,
  0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x80, 0x80,
  0x80, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x7F,
  0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
  0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x7F, 0x7F, 0x7F, 0x80,
  0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x7F, 0x7F, 0x7F,
  0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x80,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
  0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x80,
  0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
  0x80, 0x80, 0x80, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x80, 0x80,
  0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x7F, 0x7F,
  0x7F, 0x80, 0x80, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F,
  0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F …
```

… 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x7F,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F,
0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x80,
0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F, 0x7F,
0x7F, 0x7F, 0x7F, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x7F, 0x7F,
0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x7F, 0x7F, 0x7F, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x7F, 0x80 };

ДОДАТОК Д

```
const unsigned char sample_clear[43022] = {
  0x52, 0x49, 0x46, 0x46, 0x06, 0xA8, 0x00, 0x00, 0x57, 0x41, 0x56, 0x45,
  0x66, 0x6D, 0x74, 0x20, 0x10, 0x00, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00,
  0x80, 0x3E, 0x00, 0x00, 0x80, 0x3E, 0x00, 0x00, 0x01, 0x00, 0x08, 0x00,
  0x64, 0x61, 0x74, 0x61, 0x2D, 0xA7, 0x00, 0x00, 0x80, 0x7D, 0x7F, 0x83,
  0x81, 0x7C, 0x7D, 0x83, 0x82, 0x7D, 0x7C, 0x81, 0x84, 0x80, 0x7A, 0x7D,
  0x85, 0x84, 0x7A, 0x79, 0x85, 0x88, 0x79, 0x72, 0x83, 0x90, 0x80, 0x6F,
  0x7C, 0x92, 0x88, 0x6F, 0x74, 0x8F, 0x92, 0x77, 0x6C, 0x84, 0x97, 0x83,
  0x69, 0x77, 0x95, 0x90, 0x70, 0x6B, 0x8A, 0x98, 0x7C, 0x67, 0x7D, 0x99,
  0x8B, 0x69, 0x6E, 0x91, 0x97, 0x75, 0x65, 0x82, 0x9C, 0x86, 0x66, 0x72,
  0x97, 0x95, 0x70, 0x66, 0x8A, 0x9E, 0x80, 0x62, 0x78, 0x9C, 0x90, 0x6A,
  0x6B, 0x91, 0x9A, 0x78, 0x65, 0x81, 0x9D, 0x88, 0x66, 0x71, 0x97, 0x96,
  0x6F, 0x66, 0x8B, 0x9E, 0x7E, 0x62, 0x7B, 0x9F, 0x8E, 0x67, 0x6E, 0x96,
  0x99, 0x72, 0x63, 0x87, 0x9F, 0x82, 0x63, 0x78, 0x9C, 0x91, 0x69, 0x6A,
  0x92, 0x9D, 0x76, 0x61, 0x83, 0xA1, 0x87, 0x62, 0x73, 0x9C, 0x96, 0x6B,
  0x65, 0x8E, 0xA0, 0x7B, 0x60, 0x7D, 0xA1, 0x8D, 0x63, 0x6B, 0x97,0x9C,
  0x71, 0x60, 0x87, 0xA3, 0x83, 0x5F, 0x74, 0x9E, 0x94, 0x67, 0x63, 0x90,
  0xA1, 0x7A, 0x5D, 0x7C, 0xA1, 0x8D, 0x60, 0x68, 0x98, 0x9E, 0x70,0x5C,
  0x85, 0xA5, 0x83, 0x5A, 0x70, 0xA0, 0x96, 0x64, 0x60, 0x90, 0xA2, 0x77,
  0x59, 0x7C, 0xA3, 0x8C, 0x5E, 0x68, 0x98, 0x9C, 0x6D, 0x5C, 0x86,0xA3,
  0x81, 0x5B, 0x71, 0x9D, 0x94, 0x65, 0x61, 0x8E, 0x9F, 0x77, 0x5B, 0x7A,
  0x9E, 0x8A, 0x60, 0x68, 0x95, 0x99, 0x6F, 0x5E, 0x83, 0x9E, 0x80, 0x5C,
  0x70, 0x99, 0x92, 0x67, 0x62, 0x8B, 0x9C, 0x76, 0x5B, 0x79, 0x9D, 0x89,
  0x60, 0x68, 0x95, 0x98, 0x6C, 0x5D, 0x84, 0x9E, 0x7E, 0x5C, 0x71, 0x9A,
  0x91, 0x65, 0x62, 0x8D, 0x9D, 0x75, 0x5A, 0x7B, 0xA0, 0x89, 0x5D, 0x67,
  0x97, 0x9A, 0x69, 0x59, 0x86, 0xA2, 0x7D, 0x57, 0x72, 0xA0, 0x93, 0x60,
  0x5F, 0x92, 0xA2, 0x72, 0x54, 0x7D, 0xA7, 0x89, 0x56, 0x66, 0x9E, 0x9E,
  0x66, 0x56, 0x89, 0xA8, 0x7E, 0x51, 0x70, 0xA6, 0x96, 0x5C, 0x5B …
```

```
… 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80,0x7F,
 0x80, 0x7F, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80,
 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
 0x80, 0x80, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80,
 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x7F,
 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80,
 0x80, 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80,
 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x80,
 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80,
 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80,
 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80,
 0x7F, 0x80, 0x7F, 0x80, 0x80, 0x80, 0x7F, 0x80, 0x7F, 0x80, 0x7F, 0x80,
 0x7F, 0x00, 0x4C, 0x49, 0x53, 0x54, 0x3C, 0x00, 0x00, 0x00, 0x49, 0x4E,
 0x46, 0x4F, 0x49, 0x43, 0x52, 0x44, 0x06, 0x00, 0x00, 0x00, 0x32, 0x30,
 0x32, 0x31, 0x00, 0x00, 0x49, 0x53, 0x46, 0x54, 0x22, 0x00, 0x00, 0x00,
 0x4C, 0x61, 0x76, 0x66, 0x35, 0x38, 0x2E, 0x32, 0x39, 0x2E, 0x31, 0x30,
 0x30, 0x20, 0x28, 0x6C, 0x69, 0x62, 0x73, 0x6E, 0x64, 0x66, 0x69, 0x6C,
 0x65, 0x2D, 0x31, 0x2E, 0x30, 0x2E, 0x33, 0x31, 0x29, 0x00, 0x69, 0x64,
 0x33, 0x20, 0x68, 0x00, 0x00, 0x00, 0x49, 0x44, 0x33, 0x03, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x5D, 0x54, 0x49, 0x54, 0x32, 0x00, 0x00, 0x00, 0x23,
 0x00, 0x00, 0x01, 0xFF, 0xFE, 0x1E, 0x04, 0x34, 0x04, 0x3D, 0x04, 0x3E,
 0x04, 0x3A, 0x04, 0x40, 0x04, 0x30, 0x04, 0x42, 0x04, 0x3D, 0x04, 0x4B,
 0x04, 0x39, 0x04, 0x20, 0x00, 0x3F, 0x04, 0x38, 0x04, 0x41, 0x04, 0x3A,
 0x04, 0x54, 0x44, 0x52, 0x43, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00,
 0x32, 0x30, 0x32, 0x31, 0x54, 0x58, 0x58, 0x58, 0x00, 0x00, 0x00, 0x17,
 0x00, 0x00, 0x00, 0x53, 0x6F, 0x66, 0x74, 0x77, 0x61, 0x72, 0x65, 0x00,
 0x4C, 0x61, 0x76, 0x66, 0x35, 0x38, 0x2E, 0x32, 0x39, 0x2E, 0x31, 0x30,
 0x30, 0x00
};
```