МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»
Циклова комісія *комп'ютерних систем та мереж*
(повна назва циклової комісії)

Допустити до захисту
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
(підпис) *Ірина КРАВЧУК*
(ім'я, ПРІЗВИЩЕ)
« 10 » 06 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬО-ПРОФЕСІЙНОГО СТУПЕНЯ
ФАХОВИЙ МОЛОДШИЙ БАКАЛАВР

Тема: *Система моніторингу курсу криптовалют BTC і ETH на базі NodeMCU ESP8266*

Група: *3-013* Спеціальність: *123 «Комп'ютерна інженерія»*

Здобувач освіти (підпис) *Руслан ЗАЙДУЛІН* (ім'я, ПРІЗВИЩЕ)

Керівник роботи (підпис) *Тетяна РУБАН* (ім'я, ПРІЗВИЩЕ)

Консультант з оформлення пояснювальної записки (підпис) *Оксана ОСАДЧА* (ім'я, ПРІЗВИЩЕ)

Кривий Ріг 2025 р.

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

Відділення *комп'ютерної та програмної інженерії*
Циклова комісія *комп'ютерних систем та мереж*
Освітньо-професійний ступінь *фаховий молодший бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Голова випускової циклової комісії
комп'ютерних систем та мереж
(повна назва циклової комісії)
(підпис) *Ірина КРАВЧУК* (ім'я. ПРІЗВИЩЕ)
« *01* » *03* 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ОСВІТИ

Зайдулін Руслан Рінатович
(прізвище, ім'я, по батькові)

1.Тема роботи *Система моніторингу курсу криптовалют BTC і ETH на базі NodeMCU ESP8266*

Керівник роботи *викладач першої категорії, Рубан Тетяна Миколаївна*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по коледжу від « *04* » *04* 2025 року № *50-ст*

2. Строк подання здобувачем освіти роботи з _____ по _____

3. Вихідні дані до роботи *Розробка пристрою, який збирає та візуалізує про криптовалюти (BTC, ETH, DOGE), національні курси валют (USD, EUR), поточну погоду в двох містах і статус повітряної тривоги.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *обґрунтування вибраного напрямку, проектування універсального IoT-гаджета DataBox на базі ESP8266 з TFT-дисплеєм ST7735S, створення веб-інтерфейса, розробка функціональної схеми, програмування мікроконтролера ESP8266.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація Microsoft PowerPoint

6. Консультанти розділів роботи (проекту)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Узгодження технічного завдання з керівником кваліфікаційної роботи*	*24.03.2025-27.03.2025*	*виконано*
2	*Підбір та вивчення науково-технічної літератури за темою кваліфікаційної роботи*	*28.03.2025-31.03.2025*	*виконано*
3	*Обґрунтування вибору програмних засобів*	*01.04.2025-04.04.2025*	*виконано*
4	*Опис компонентів. Обґрунтування їх вибору.*	*05.04.2025-08.04.2025*	*виконано*
5	*Розробка програмного забезпечення*	*09.04.2025-28.04.2025*	*виконано*
6	*Дослідження ефективності реалізованих методів.*	*29.04.2025-04.05.2025*	*виконано*
7	*Написання пояснювальної записки*	*12.05.2025-25.05.2025*	*виконано*
8	*Перевірка на плагіат пояснювальної записки*	*26.05.2025 - 01.06.2025*	*виконано*
9	*Попередній захист кваліфікаційної роботи*	*02.06.2025-06.06.2025*	*виконано*
10	*Захист кваліфікаційної роботи*		

Здобувач освіти _____ *Руслан ЗАЙДУЛІН*
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник роботи _____ *Тетяна РУБАН*
(підпис) (ім'я, ПРІЗВИЩЕ)

StrikePlagiarism.com

Дата звіту 5/28/2025

Дата редагування ---

Документ прийнятий

Звіт подібності

метадані

Назва організації

Ukrainian national aviation university

Заголовок

КПІ_2025_123_Зайдулін

Автор Науковий керівник / Експерт

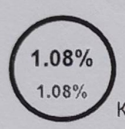
ЗайдулінРубан Т.М

підрозділ

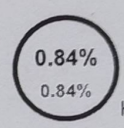
Криворізький Фаховий коледж

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25	10615	78697
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв		0
Інтервали		0
Мікропробіли		6
Білі знаки		0
Парафрази (SmartMarks)	a	5

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз

Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	http://elar.khmnu.edu.ua/bitstream/123456789/13927/1/%D0%91%D0%B0%D0%BA%D0%B0%D0%BB%D0%B0%D0%B2%D1%80%D1%81%D1%8C%D0%BA%D0%B0_%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0_%D0%97%D0%B0%D0%B1%D0%B0%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D0%9A%D0%862_19_2_R_1%20%281%29.pdf	19 0.18 %
2	http://asu.kpi.ua/wp-content/uploads/2018/12/ISTU-2018-ilovepdf-compressed.pdf	16 0.15 %
3	https://www.cs.ubbcluj.ro/files/curricula/2020/syllabus/IR_sem4_MLR5015_ro_bufny_2020_5156.pdf	11 0.10 %

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Система моніторингу курсу криптовалют *BTC* і *ETH* на баз і *NodeMCU ESP8266*»: 69 сторінок основного тексту, 43 рисунка, 25 таблиць, 15 використаних джерел, 1 додаток.

IoT, ESP8266, WEB-ІНТЕРФЕЙС, TFT-ДИСПЛЕЙ, JSON.

В даній кваліфікаційній роботі розроблено універсальний *IoT*-гаджет *DataBox* на базі *ESP8266* з *TFT*-дисплеєм *ST7735S* та веб-інтерфейсом. Пристрій збирає та візуалізує інформацію про криптовалюти (*BTC, ETH, DOGE*), національні курси валют (*USD, EUR*), поточну погоду в двох містах і статус повітряної тривоги. Конфігурація *Wi-Fi* здійснюється через форму в браузері без потреби перепрошивки.

Метою моєї дипломної роботи є розробка програмно-апаратного комплексу для віддаленого моніторингу фінансових і метео-даних з можливістю гнучкого налаштування користувачем. Актуальність роботи полягає в необхідності оперативного отримання актуальної інформації з різних джерел через одну компактну систему без застосування зовнішніх серверів.

Об’єктом дослідження є архітектура прошивки *ESP*8266 та реалізація вбудованого веб-інтерфейсу для віддаленого керування й відображення даних. Предметом дослідження є методи збору, парсингу та відображення динамічних даних (*JSON*/*XML*) на *TFT*-дисплеї і в браузері.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

IoT (англ. Internet of Things) – технологія «Інтернет речей», що дозволяє підключати до мережі різноманітні пристрої для обміну даними та управління.

ESP8266 – недорогий *Wi-Fi* мікроконтролер з інтегрованою мережею, широко використовуваний у проектах *IoT*.

TFT (англ. Thin Film Transistor) – тип *LCD*-дисплея з тонкоплівковими транзисторами для керування кожним пікселем, що забезпечує якісне відображення графіки.

ST7735S – контролер *TFT*-дисплея, сумісний з *Adafruit_ST*7735 бібліотекою, який забезпечує інтерфейс *SPI* і керування екраном 128×160 пікселів.

SPI (англ. Serial Peripheral Interface) – синхронний послідовний інтерфейс, що використовується для обміну даними між мікроконтролером та периферійними пристроями (дисплеєм, сенсорами тощо).

HTTP (англ. HyperText Transfer Protocol) – протокол передачі гіпертексту, основа веб-зв’язку між клієнтом (браузером) та сервером.

HTTPS (англ. HTTP Secure) – розширення *HTTP* з використанням *SSL*/*TLS* для захищеної передачі даних через мережу.

ArduinoJson – *C++* бібліотека для *Arduino*/*ESP*, що забезпечує зручний парсинг і генерацію *JSON*-документів у ресурсно-обмежених середовищах.

GPIO (англ. General Purpose Input/Output) – універсальні цифрові входи/виходи мікроконтролера, які використовуються для зчитування кнопок (*BUTTON_A_PIN, BUTTON_B_PIN*) та керування периферією.

ВСТУП

У сучасному світі криптовалюти, зокрема *Bitcoin* (*BTC*) та *Ethereum* (*ETH*), займають важливе місце в економічних та фінансових процесах. Їх вартість характеризується високою волатильністю, що викликає значний інтерес як у професійних трейдерів, так і у звичайних користувачів. Оперативний моніторинг змін курсу криптовалют дозволяє своєчасно приймати інвестиційні рішення та мінімізувати ризики.

Актуальність дослідження полягає у зростаючій потребі у доступних та зручних засобах моніторингу криптовалют у режимі реального часу. Використання мікроконтролерів, таких як *NodeMCU ESP*8266, відкриває можливість створення недорогих, автономних пристроїв для відображення актуальної інформації про курси *BTC* і *ETH* без необхідності постійного використання комп'ютера чи смартфона.

Метою роботи є розробка пристрою для моніторингу курсу *BTC* та *ETH* у реальному часі із можливістю сенсорного перемикання режимів та відображення інформації на *TFT*-дисплеї.

Об'єктом дослідження є процес відстеження курсу криптовалют у режимі реального часу.

Предметом дослідження є технології розробки апаратно-програмних систем на базі мікроконтролера *NodeMCU ESP*8266 для збору, обробки та візуалізації даних про курси криптовалют, а також інтеграція сенсорного керування та веб-інтерфейсу.

Завдання дослідження:

- Дослідити криптовалютний ринок та існуючі методи моніторингу курсів.
- Провести аналіз доступних рішень для моніторингу курсу криптовалют.
- Обрати апаратні та програмні компоненти для створення пристрою.

- Вивчити технічні можливості плати *NodeMCU ESP*8266 та відповідного периферійного обладнання.
- Розробити електронну схему підключення компонентів.
- Створити програмне забезпечення для отримання актуальних курсів через *API* та виведення їх на *TFT*-дисплей.
- Інтегрувати сенсорні кнопки *TTP*223 для перемикання між валютами.
- Реалізувати веб-інтерфейс для налаштування пристрою.
- Провести тестування пристрою та оцінити його ефективність.

Розроблений пристрій може бути використаний як персональний гаджет для трейдерів та інвесторів, надаючи їм швидкий доступ до актуальної інформації про курси криптовалют у зручному форматі. Крім того, він може слугувати демонстраційним зразком у навчальних закладах для вивчення основ Інтернету речей (*IoT*), принципів роботи з мікроконтролерами та розробки вбудованого програмного забезпечення. Такий підхід сприяє розвитку практичних навичок у здобувачів освіти, заохочує інноваційне мислення та розширює можливості застосування сучасних технологій у різних сферах діяльності.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз криптовалютного ринку: *BTC* та *ETH*

Ринок криптовалют - це швидко розвивається та технологічно розвинене сфера фінансів, постійно розширюється завдяки впровадженню нових технологій та зростаючій участі інвесторів. Криптовалюти побудовані на децентралізованих технологіях, таких як блокчейн, які пропонують прозорість, безпеку та незалежність від централізованих фінансових установ. Основними характеристиками ринку криптовалют є його надзвичайні коливання цін, доступність у всьому світі, відсутність централізованого контролю та потенціал для анонімних транзакцій. Біткойн (*BTC*) та *Ethereum* (*ETH*)-дві найвідоміші криптовалюти в цій екосистемі. [1]

Біткойн (рисунок 1.1) – це початкова і широко визнана криптовалюта, розроблена в 2009 році анонімною особою або групою розробників, відомих як Сатоші Накамото. Основна ідея цієї цифрової валюти полягає у встановленні автономної форми грошей, яка дозволяє трансакцій без участі банків чи інших фінансових посередників. Основою експлуатації *Bitcoin* є технологія *blockchain*, яка працює за принципом підтвердження роботи (*POW*), що потребує значної обчислювальної потужності для перевірки транзакцій. Загальна пропозиція *BTC* обмежена 21 мільйон монет, що створює дефіцит, подібну до золота. Цей фактор суттєво впливає на його ринкову вартість, оскільки дефіцит пропозиції та збільшення попиту призводять до стійкого зростання цін з часом.

Рисунок 1.1 – Ілюстративна форма біткойна у фізичному стані

Ethereum, розроблений у 2015 році *Vitalik Buterin*, є другим найціннішим криптовалютою з точки зору ринкової капіталізації. Основна відмінність між *Ethereum* та *Bitcoin* полягає в його здатності до розробки розумних контрактів та децентралізованих застосувань (*DAPPS*), що значно розширює його спектр можливостей. *Ethereum* перейшов до механізму доказового (*POS*) після впровадження *Ethereum* 2.0, що дозволяє значне зменшення споживання енергії мережі. Крім того, *Ethereum* має систему, де спалюється частина комісії, що допомагає контролювати інфляцію та робить актив більш цінним. На рисунку 1.2 ми можемо побачити логотип славнозвісної криптовалюти.

Рисунок 1.2 – Логотип *Ethereum* (ефіріум)

На ринок криптовалют впливає кілька факторів, причому найбільш значущими є рівень регулювання в різних країнах, загальний економічний стан, впровадження нових технологій, зміни в механізмах консенсусу та попит на послуги, що використовують блокчейн. Наприклад, на курс *BTC* сильно впливає події, пов'язані з вдвічі (зменшення винагороди за видобуток двічі на 4 роки), оскільки це безпосередньо впливає на наявність нових монет на ринку. *Ethereum* спирається на просування децентралізованих фінансових послуг (*DEFI*), зростання ринку непереможних маркерів (*NFT*) та масштабованості його мережі. Також, експерти говорять що якщо ім’я засновника біткоїна стане відома то його вартість дуже сильно впаде, а ім’я засновника Ефіріума відомо усім і це не як не заважає їй рости у ціні. На рисунку 1.3 зображено фото засновника Ефіріума. Також у таблиці 1.1, наведено основні відмінності між двома валютами.

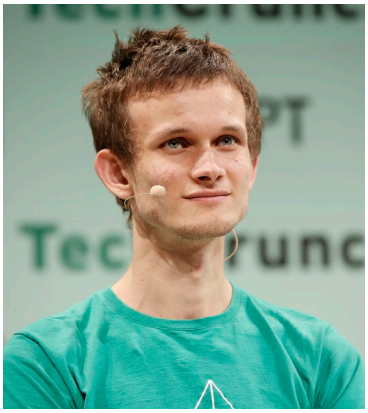


Рисунок 1.3 – Засновник Ефіріума (*Vitalik Buterin*)

Таблиця 1.1 – Основні відмінності між двома криптовалютами

Параметр	***Bitcoin* (*BTC*)**	***Ethereum* (*ETH*)**
Рік створення	2009	2015
Засновник	Сатоші Накамото	Віталік Бутерін
Механізм консенсусу	*Proof-of-Work* (*PoW*)	*Proof-of-Stake* (*PoS*)
Максимальна емісія	21 млн *BTC*	Необмежена, але з механізмом спалювання *ETH*
Час транзакції	~10 хвилин	~12 секунд
Середня комісія	$1 - $50	$0.5 - $10
Призначення	Цифрове золото, засіб збереження вартості	Смарт-контракти, децентралізовані додатки
Основні конкуренти	*Litecoin*, *Bitcoin Cash*	*Solana*, *Cardano*, *Binance Smart Chain*

Щодо ринкових показників, *Bitcoin* залишається найбільшою криптовалютою за капіталізацією, яка у 2024 році перевищує 500 млрд доларів США, тоді як капіталізація *Ethereum* складає близько 250 млрд доларів США. Обсяги щоденних торгів *Bitcoin* сягають 40 млрд доларів, тоді як *Ethereum* має середньодобовий обсяг торгів близько 25 млрд доларів. У таблиці 1.2, наведені основні фактори впливання на вартість криптовалюти.

Таблиця 1.2 – Фактори, що впливають на вартість *BTC* та *ETH*

Фактор впливу	***Bitcoin* (*BTC*)**	***Ethereum* (*ETH*)**
Регуляторні рішення	Обмеження у країнах, заборона або легалізація	Вплив законодавства на використання смарт-контрактів
Попит та пропозиція	Обмежена емісія сприяє підвищенню ціни	Динаміка використання в *DeFi* та *NFT*
Технологічні оновлення	Халвінг, оновлення протоколу	*Ethereum* 2.0, масштабованість, *Layer*-2 рішення
Вплив макроекономічних факторів	Кореляція з ринками золота та акцій	Популярність *DeFi*, стабільність платформи

Біткойн залишається популярним активом для інвесторів, які шукають надійний спосіб зберегти свою вартість на тривалий час. Він часто порівнюється із золотом через обмежену кількість монет та його стійкість до інфляції. *Ethereum* – це найбільша платформа для створення смарт-контрактів. Вона є основною криптовалютою, яка використовується для інновацій у сфері децентралізованих фінансів. *Bitcoin* і *Ethereum* виконують різні функції на ринку криптовалют. Біткоїн вважається активом для довгострокового збереження вартості, а ефір залишається лідером у сфері блокчейн-платформ для розробки децентралізованих додатків.

Обидві криптовалюти мають великий вплив на ринок і відіграють важливу роль у розвитку цифрової економіки. Далі зростання *Bitcoin* буде залежати від того, як в усьому світі люди прийматимуть його як альтернативу традиційним інвестиціям. Майбутнє *Ethereum* визначатиметься його здатністю підтримувати велику кількість транзакцій і мати низькі витрати на них. З огляду на вищезазначене, криптовалютний ринок є складним, але швидко розвивається. Біткойн і Ефір залишаються основними рушійними силами ринку. Кожна з цих криптовалют має свої унікальні властивості та сфери застосування.

1.2 Методи моніторингу криптовалютних курсів

Загальний огляд методів моніторингу. Моніторинг криптовалютного ринку дуже важливий для трейдерів, інвесторів та розробників фінансових технологій. Основні методи отримання актуальних курсів *BTC* і *ETH* можна розділити на два основні типи: Автоматизовані методи – це використання *API*, біржових платформ, мобільних додатків та ботів. Навчальні методи – це спостереження за курсами через веб-сайти, аналітичні огляди та фінансові новини. Кожен із цих методів має свої плюси і мінуси, які варто врахувати при створенні пристрою для моніторингу курсів. Отримання актуальних курсів за допомогою *API*. Один з найбільш ефективних способів отримувати дані - це автоматичне отримання їх через *API* (інтерфейс програмування додатків). Ви можете отримувати курси криптовалют в реальному часі, не потрібно відвідувати сайти або шукати інформацію вручну. У таблиці 1.3 наведено перелік популярних *API*-сервісів для моніторингу криптовалют. На рисунку 1.4 показано приклад отримання курсу через *API CoinLore*, а на рисунку 1.5 — інтерфейс самого веб-сайту цього сервісу.

Таблиця 1.3 – Популярні *API*-сервіси для моніторингу криптовалют

API	**Джерело даних**	**Основні особливості**
CoinGecko API	*CoinGecko*	Безкоштовний, доступ до понад 5000 монет, історичні дані
CoinMarketCap API	*CoinMarketCap*	Висока точність даних, платний та безкоштовний доступ
Binance API	*Binance*	Дані з найбільшої біржі, реальні ринкові ціни
CoinLore API	*CoinLore*	Простий *API* для отримання основних курсів
CryptoCompare API	*CryptoCompare*	Глибока аналітика, історичні дані, графіки

```
// Оновлюємо все
void updateAllData() {
  Serial.println("[Main] updateAllData() started.");
  fetchCrypto("https://api.coinlore.net/api/ticker/?id=90", cryptos[0]); // BTC
  fetchCrypto("https://api.coinlore.net/api/ticker/?id=80", cryptos[1]); // ETH
  fetchCrypto("https://api.coinlore.net/api/ticker/?id=2",  cryptos[2]); // DOGE

  fetchCurrencies();
  fetchWeather(0);
  fetchWeather(1);
  fetchAirAlerts();

  Serial.println("[Main] updateAllData() finished.");
}
```

Рисунок 1.4 – Приклад отримання курса криптовалюти через *API*

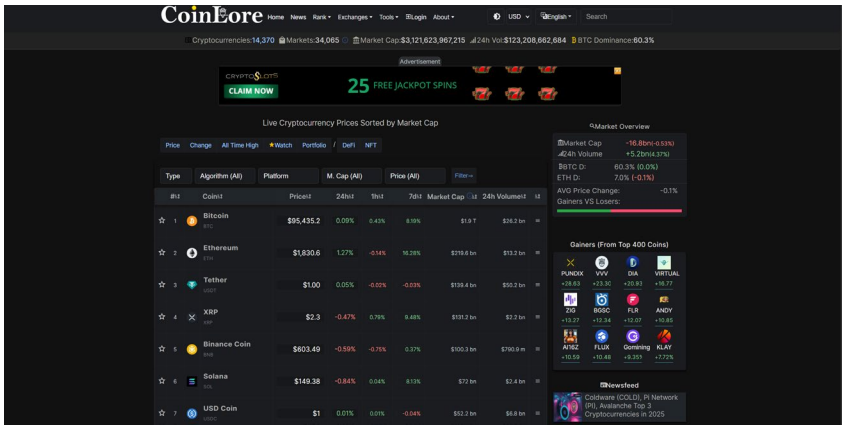


Рисунок 1.5 – Сайт *CoinLore*

Використання *API* є найбільш зручним методом для інтеграції в *IoT*-пристрій на базі *NodeMCU ESP*8266, що дозволяє оновлювати дані.

Веб-ресурси з моніторингу криптовалют надають широкі аналітичні можливості, включаючи графіки, історичні дані та ринкову статистику. У таблиці 1.4, перелiковано список сайтів для перегляду курса криптовалют.

Таблиця 1.4 – Популярні сайти для перегляду курсів *BTC* та *ETH*

Веб-сайт	**Основні можливості**
CoinMarketCap	Оновлення цін, аналітика, рейтинг криптовалют
CoinGecko	Відображення цін, ринкова капіталізація, рейтинги
TradingView	Графіки та технічний аналіз для трейдерів
CryptoCompare	Детальні аналітичні огляди криптовалют
Binance	Реальні курси та торгові дані з біржі *Binance*

Ці ресурси є ефективними для аналітичного дослідження ринку, однак вони не підходять для автоматизованого моніторингу в реальному часі. Біржі криптовалют надають точні дані про поточні курси, глибину ринку та обсяги торгів, що робить їх важливим джерелом для моніторингу. У таблиці 1.5 перераховано список криптовалютних бірж.

Таблиця 1.5 – Популярні криптовалютні біржі

Біржа	**Добовий обсяг торгів**	**Основні особливості**
Binance	$50+ млрд	Найбільша біржа, підтримка *API*
Coinbase	$5+ млрд	Високий рівень безпеки, простий інтерфейс
Kraken	$3+ млрд	Регульована у США, маржинальна торгівля
OKX	$10+ млрд	Глибока ліквідність, підтримка *NFT*
Huobi	$8+ млрд	Велика кількість криптовалют, швидкі угоди

Біржі використовуються трейдерами для швидкого реагування на зміну ринку, а також як джерело *API* для автоматизованого отримання курсів, одну з крупних бірж ми можемо побачити на рисунку 1.6.

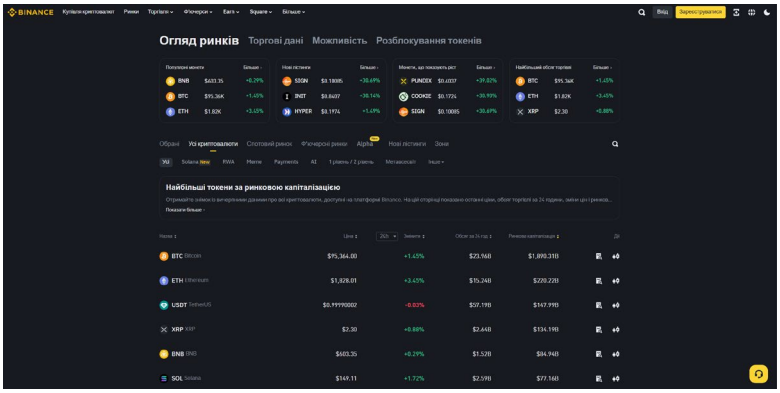


Рисунок 1.6 – *Binance* біржа

Завдяки мобільним додаткам користувачі можуть відстежувати курси криптовалют у режимі реального часу з будь-якого місця, корисний список додатків вказаний у таблиці 1.6.

Таблиця 1.6 – Популярні мобільні додатки для моніторингу *BTC* та *ETH*

Додаток	**Операційні системи**	**Основні функції**
CoinMarketCap	*iOS*, *Android*	Відображення курсів, аналітика, *push*-сповіщення
Binance	*iOS*, *Android*	Біржова торгівля, графіки, *API*
CryptoCompare	*iOS*, *Android*	Новини, аналітика, ринкові тренди
Blockfolio	*iOS*, *Android*	Управління криптопортфелем, моніторинг цін

Мобільні додатки підходять для ручного моніторингу курсів, але не можуть бути інтегровані в автономний пристрій, такий як *IoT*-гаджет. Багато трейдерів використовують *Telegram*-боти, які можуть надсилати сповіщення про зміну курсу, угоди на ринку або появу нової криптовалюти. Список корисних ботів для цього вказано у таблиці 1.7.

Таблиця 1.7 – Популярні *Telegram*-боти

Бот	**Основні функції**
@CryptoWhale	Відстеження змін курсу *BTC*, *ETH* та інших криптовалют
@CoinTrendz	Моніторинг цін у реальному часі
@WhaleAlert	Сповіщення про великі транзакції в блокчейні

Telegram-боти дозволяють швидко отримувати інформацію про зміни курсу без необхідності перевіряти додатки або веб-сайти. Один із ботів це *@CoinTrendz*, який ми можемо побачити на рисунку 1.7.

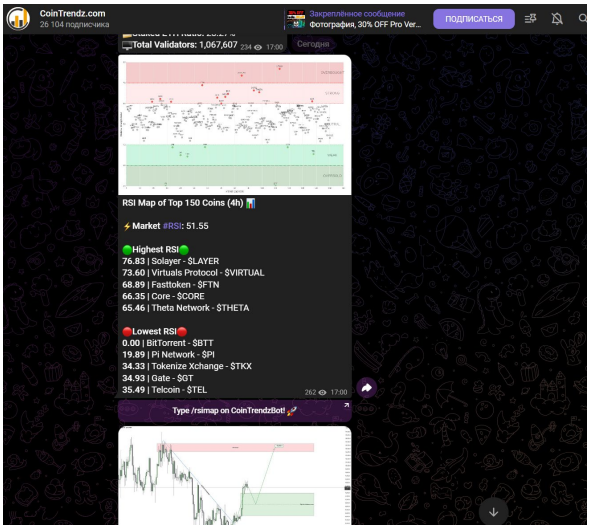


Рисунок 1.7 – Бот у *Telegram* (*@CoinTrendz*)

На основі аналізу методів моніторингу криптовалютних курсів можна зробити наступні висновки:

– *API*-сервіси (*CoinLore*, *Binance API*) – найкращий варіант для автоматизованих пристроїв (наприклад, *IoT*-гаджетів).
– Веб-сайти (*CoinMarketCap*, *TradingView*) – корисні для аналітики, але не підходять для автоматичного оновлення курсу.
– Біржі криптовалют – надають найточніші дані та можливість інтеграції через *API*.
– Мобільні додатки – оптимальні для користувачів, які хочуть відстежувати курс вручну.
– *Telegram*-боти – зручний інструмент для отримання швидких повідомлень про зміну курсу.

Для розробки пристрою, який буде стежити за курсами *BTC* і *ETH*, використовуючи *NodeMCU ESP*8266, найкраще використовувати *API* (такі як *CoinLore*, *CoinGecko*, *Binance API* запити). Це дозволяє отримувати актуальні курси в реальному часі з мінімальними затримками.

1.3 Огляд існуючих рішень для відстеження криптовалют

Існують як прості агрегатори, що в реальному часі демонструють лише курс основних монет, так і комплексні платформи з розширеним набором інструментів: історичні графіки, технічні індикатори, моніторинг портфеля та персоналізовані сповіщення. Більшість сервісів надають безкоштовний доступ до базових даних, проте за доступ до API або додаткових аналітичних функцій може знадобитися підписка. У таблиці 1.8 наведено найбільш популярні веб-сайти для моніторингу криптовалют, їхніми ключовими можливостями та особливостями використання.

Таблиця 1.8 – Популярні веб-сайти для моніторингу криптовалют

Сервіс	**Опис**	**Особливості**
CoinMarketCap	Найпопулярніший агрегатор криптовалютних курсів	Оновлення в реальному часі, рейтинги криптовалют
Coinlore	Конкурент *CoinMarketCap*, який надає *API* для моніторингу	Глибока аналітика, рейтинги проектів
CryptoCompare	Включає історичні графіки та аналітичні огляди	Можливість моніторингу портфеля
TradingView	Потужний інструмент для технічного аналізу криптовалют	Вбудовані індикатори, аналітичні інструменти
Live Coin Watch	Менш популярний сервіс, але з широким набором функцій	Персоналізація списків відстеження

Переваги: Доступність онлайн без необхідності встановлення додаткових програм. Зручний інтерфейс із великою кількістю даних та індикаторів.

Недоліки: Дані можуть оновлюватися із затримкою. Відсутність можливості автоматичної обробки інформації. Біржові платформи як інструмент моніторингу

Криптовалютні біржі є основним джерелом актуальних даних про курси цифрових активів. Вони не тільки дозволяють отримувати ціни в реальному часі, але й містять інформацію про ліквідність, глибину ринку та обсяги торгів.

Є дуже багато криптовалютних бірж, це наприклад *Binance*, *Coinbase*, *Kraken*, *OKX* та *Huobi*.

Переваги: Найактуальніші курси без затримки. Висока точність даних через пряму взаємодію з ринком. Можливість отримання глибини ринку та історичних даних.

Недоліки: Необхідність реєстрації для повноцінного доступу. Дані можуть відрізнятися між різними біржами. Мобільні додатки для відстеження курсів

Багато криптовалютних трейдерів використовують мобільні додатки для швидкого доступу до інформації про ринок, список цих додатків наведено у таблиці 1.9

Таблиця 1.9 – Популярні мобільні додатки для моніторингу курсів

Додаток	**Операційні системи**	**Особливості**
CoinMarketCap	*iOS*, *Android*	Агрегатор ринкових даних
Binance App	*iOS*, *Android*	Біржова торгівля, аналітика
CryptoCompare	*iOS*, *Android*	Аналітичні звіти, графіки
Blockfolio	*iOS*, *Android*	Відстеження портфеля криптовалют

Автоматизовані *API*-сервіси дозволяють інтегрувати актуальні курси криптовалют у програми, сайти чи *IoT*-пристрої.

Переваги: Автоматичне отримання оновлених курсів. Можливість використання в мобільних додатках та пристроях. Висока точність та швидкість оновлення.

Недоліки: Деякі *API* вимагають платної підписки для доступу до повного функціоналу.

На основі аналізу різних рішень для відстеження криптовалют можна зробити висновок, що на сьогоднішній день є великий вибір інструментів для моніторингу курсів цифрових активів. Кожен з цих інструментів має свої переваги та недоліки, які залежать від сфери використання. Усі методи можна

умовно поділити на дві групи: ручні (онлайн-сервіси, мобільні додатки, біржові платформи) та автоматизовані (*API*, програмні рішення). Це робить їх менш ефективними у випадках, коли потрібен постійний збір інформації для пристроїв або програм.

Використання біржових платформ — це один з найнадійніших способів отримувати актуальні курси, оскільки дані надходять безпосередньо з торгових майданчиків.

Мобільні додатки, такі як *CryptoCompare*, *Blockfolio* та *Binance App*, дозволяють швидко отримувати інформацію про курси криптовалют у будь-який момент. Вони дозволяють налаштовувати пуш-сповіщення, переглядати історію зміни цін та аналізувати ринок за допомогою створених графіків. Основна причина популярності мобільних додатків – це їхня зручність для користувачів, які хочуть швидко перевіряти стан ринку. Ці рішення не підходять для автоматичного моніторингу, оскільки працюють тільки в ручному режимі. Найефективніший спосіб отримати актуальні курси криптовалют – це використовувати *API* (інтерфейс програмування додатків). Ці *API* надаються різними платформами, такими як *Binance*, *CoinGecko*, *CryptoCompare* та *CoinLore*. *API* дозволяє автоматично надсилати запити до серверів і отримувати актуальну інформацію у форматах *JSON* або *XML*. Це значно спрощує інтеграцію даних у програмні рішення та пристрої Інтернету речей (*IoT*).

Сервіси забезпечують швидке оновлення курсів у реальному часі, а також просту інтеграцію з мікроконтролером *NodeMCU ESP*8266 і можливістю налаштування. Використання *API* дозволяє створити стабільну та ефективну систему збору даних, яка працює автоматично і не потребує постійного залучення користувача.

1.4 Висновки щодо вибору платформи для розробки

Основними критеріями вибору є: підключення до інтернету - пристрій повинен отримувати дані від *API* криптовалютних бірж, курс національних валют, погоди та тривоги в режимі реального часу, тому підтримка *Wi-Fi* є обов'язковою.

Продуктивність та енергоефективність - пристрій повинен мати достатню обчислювальну потужність для виконання *HTTP*-запитів та обробки отриманих даних з мінімальним енергоспоживанням.

Підтримка графічного дисплея - пристрій повинен бути сумісним з *TFT*-дисплеєм *ST*7735*S*, який використовується для відображення курсів криптовалют.

Можливість інтеграції сенсорної кнопки - сенсорна кнопка (*TTP*223) необхідна для зміни режиму відображення пристрою. Простота програмування та інтеграції - важливо, щоб платформа мала широке програмне забезпечення, бібліотеки та підтримку спільноти. Низька вартість і доступність компонентів - платформа повинна бути економічно ефективною і складатися з легкодоступних компонентів. Виходячи з цих вимог, було проведено порівняльний аналіз різних поширених платформ для розробки *IoT*-пристроїв.

При виборі платформи для розробки пристрою моніторингу курсів криптовалют важливо враховувати підтримку *Wi-Fi* для отримання актуальних курсів з мережі, можливості підключення дисплея *ST*7735*S*, інтеграцію сенсорної кнопки *TTP*223, низьке енергоспоживання, простоту програмування тощо та багато інших базових вимог.

На цьому тлі було оцінено кілька популярних апаратних платформ, кожна з яких має свої переваги та недоліки: *Arduino Uno* - один з найпопулярніших мікроконтролерів для створення електронних проектів, але його основним недоліком є те, що він не має вбудованого модуля *Wi-Fi*. Основним недоліком є те, що він не має вбудованого модуля *Wi-Fi*. Це означає, що для отримання актуальних курсів криптовалют з *API* біржі потрібні додаткові модулі,

наприклад, модуль *Wi-Fi ESP*8266, що ускладнює реалізацію проекту. Крім того, *Arduino Uno* має обмежену тактову частоту 16 МГц і лише 2 КБ оперативної пам'яті, що ускладнює обробку *HTTP*-запитів і роботу з дисплеєм. Через ці обмеження *Arduino Uno* не є найкращим вибором для розробки мережевих *IoT*-пристроїв.

Raspberry Pi Zero W - це потужний одноплатний комп'ютер з підтримкою *Wi-Fi* і значно вищою продуктивністю, ніж *Arduino*. Процесор *Broadcom BCM*2835 з тактовою частотою 1 ГГц робить *HTTP*-запити до *API*-сервісів, отримує *API*-сервіси і може легко робити *HTTP*-запити до *API*-сервісів, обробляти вхідні дані та відображати інформацію.

Крім того, налаштування *Raspberry Pi* для легкого отримання запитів *API* є більш складним, ніж у випадку з мікроконтролерами, такими як *ESP*8266. *ESP*32 є однією з найсучасніших і найпотужніших платформ для розробки проектів *IoT*. Завдяки двоядерному процесору з тактовою частотою 240 МГц і 520 КБ оперативної пам'яті він є більш потужним, ніж *ESP*8266; *ESP*32 також підтримує *Wi-Fi* і *Bluetooth*, що дозволяє створювати більш просунуті мережеві додатки. Однак, незважаючи на свої переваги, *ESP*32 є надто потужним для простого пристрою для моніторингу курсу криптовалют: Повний потенціал *ESP*32 не може бути використаний, а його енергоспоживання вище, ніж у *ESP*8266.

Крім того, *ESP*32 є дорогим і може не бути економічним рішенням для цього проекту; *NodeMCU ESP*8266 є найкращим варіантом для реалізації цього пристрою, оскільки він має вбудований *Wi-Fi* і може отримувати найсвіжіші курси *BTC* і *ETH* без необхідності додаткових модулів. Тактової частоти 80 МГц і 80 КБ оперативної пам'яті достатньо для виконання *HTTP*-запитів до *API* криптовалютних бірж; однією з головних переваг *ESP*8266 є низьке енергоспоживання, що робить його ідеальним для автономних *IoT*-рішень.

Крім того, підтримка *SPI* і *I2C* дозволяє легко підключати *TFT*-дисплеї *ST*7735*S* і сенсорні кнопки *TTP*223, які необхідні для зручного управління пристроєм. Ще одним важливим фактором є велика бібліотека та підтримка

документації, що спрощує розробку програмного забезпечення: *ESP*8266 дешевший за *ESP*32 та *Raspberry Pi*, що робить його ідеальним для реалізації пристроїв моніторингу криптовалют. Таким чином, серед розглянутих платформ *NodeMCU*, *ESP*8266 є найкращим вибором для розробки пристроїв моніторингу курсів криптовалют *BTC* та *ETH* завдяки низькому енергоспоживанню, вбудованому *Wi-Fi*, підтримці *API* запитів, простоті програмування та доступності.

Однією з головних особливостей, які роблять *NodeMCU ESP*8266 найкращим вибором, є вбудований модуль *Wi-Fi*. Це дозволяє пристрою підключатися до інтернету без необхідності використання додаткового модуля або зовнішнього контролера. Використання *Wi-Fi* необхідне для вирішення завдання отримання актуальних курсів криптовалют з відкритих *API* криптовалютних бірж (наприклад, *API CoinLore* та *Binance*).

Завдяки цьому пристрій може оновлювати дані в режимі реального часу і надавати користувачам актуальну інформацію без затримок.

Сумісність з *TFT*-дисплеєм *ST*7735*S* є важливим фактором, оскільки пристрій повинен відображати інформацію про курси криптовалют, а завдяки тому, що *NodeMCU ESP*8266 підтримує послідовний периферійний інтерфейс (*SPI*), використовуючи бібліотеку *Adafruit GFX* та бібліотеку *Adafruit ST*7735, дисплеї можна швидко інтегрувати в проекти для створення зручних візуальних інтерфейсів. Пристрій використовує сенсорну кнопку *TTP*223 для перемикання між режимами виведення даних для зручного управління, а *NodeMCU ESP*8266 має достатню кількість виходів *GPIO* для підключення сенсорного елемента без необхідності використання додаткових адаптерів або мікросхем.

Без необхідності в додаткових адаптерах або мікросхемах. Ще однією важливою перевагою є низьке енергоспоживання, що робить *ESP*8266 ідеальним для автономних рішень *IoT*. Його енергоспоживання значно нижче, ніж у *Raspberry Pi*, що дозволяє пристрою працювати протягом тривалого періоду часу без необхідності постійного підключення до джерела живлення. На додаток до технічних характеристик, низька вартість також є важливим

фактором при виборі: *ESP*8266 *NodeMCU* значно дешевший за *Raspberry Pi* та *ESP*32, що робить його економічно вигідним рішенням для вашого проекту.

Для створення програми, що забезпечує безпеку пристрою, потрібно вибрати найкраще середовище для розробки. Це середовище має дозволяти ефективно взаємодіяти з мікроконтролером *NodeMCU ESP*8266, працювати з *API* бірж, обробляти отримані дані та виводити їх на дисплей. Основними критеріями вибору програмної платформи є стабільність, підтримка необхідних бібліотек, простота розробки та інтеграції. Серед доступних засобів для розробки було розглянуто три основні варіанти: *Arduino IDE* – це одна з найпопулярніших програм для програмування мікроконтролерів. Вона підтримує *ESP*8266 і має багато готових бібліотек для роботи з *Wi-Fi*, екранами та *API*-запитами. *MicroPython* – це можливість писати код на мові *Python*, яка може бути зручнішою для деяких розробників. Проте вона вимагає більше пам'яті та складніша в налаштуванні для роботи з мікроконтролерами. *ESPHome* – зручна програма для автоматизації пристроїв розумного дому. Вона спрощує налаштування та інтеграцію, але не підходить для проєктів, які пов'язані з моніторингом криптовалют та взаємодією з *API* бірж. Для аналізу було обрано *Arduino IDE* (рисунок 1.8), оскільки це найстабільніше, просте у використанні та ефективне середовище для розробки програмного забезпечення для *NodeMCU ESP*8266. *Arduino IDE* містить всі необхідні бібліотеки, які дозволяють швидко і просто реалізувати функції пристрою та прошивати пристрій.

Рисунок 1.8 – Логотип *Arduino IDE*

Для реалізації програмного забезпечення пристрою моніторингу курсів криптовалют було використано низку спеціалізованих бібліотек. Бібліотека

*ESP*8266*WiFi.h* забезпечує підключення до *Wi-Fi*-мережі та доступ до Інтернету. *HTTPClient.h* дозволяє виконувати *HTTP*-запити до *API* бірж для отримання актуальних курсів *BTC* та *ETH*. Для роботи з *TFT*-дисплеєм *ST*7735*S* застосовано бібліотеки *Adafruit_GFX.h* та *Adafruit_ST*7735.*h*, які дають змогу виводити графіку та текстову інформацію.

У якості апаратної платформи було обрано *NodeMCU ESP*8266, що має вбудований *Wi-Fi*-модуль, низьке енергоспоживання, підтримку *API*-запитів, а також можливість підключення *TFT*-дисплею та сенсорних кнопок *TTP*223. Розробка програмного забезпечення здійснювалась у середовищі *Arduino IDE*, яка підтримує необхідні бібліотеки та спрощує процес налаштування і програмування пристрою.

Це рішення забезпечує компактність, стабільність, енергоефективність та простоту інтеграції, що робить його оптимальним для створення пристрою реального часу з відображенням курсів криптовалют.

РОЗДІЛ 2
ВИБІР АПАРАТНОЇ ТА ПРОГРАМНОЇ ЧАСТИНИ

2.1 Обґрунтування вибору мікроконтролера *NodeMCU ESP*8266

Розробка пристроїв для моніторингу криптовалют *BTC* та *ETH* потребує надійного та енергоефективного мікроконтролера, який забезпечує підключення до мережі, взаємодіє з периферійними пристроями (*TFT*-дисплеєм *ST*7735*S* та сенсорними кнопками *TTP*223), а також забезпечує обробку та відображення вхідних даних у реальному часі. Необхідно вибрати контролер, який дозволяє обробляти і відображати вхідні дані в реальному часі.

Враховуючи всі ці вимоги, порівняльний аналіз існуючих платформ призвів до вибору *NodeMCU ESP*8266, який найкраще відповідає технічним потребам проекту. Аналіз вимог до мікроконтролера Перед вибором мікроконтролера для пристрою відстеження криптовалют були визначені основні вимоги до апаратної платформи: вбудований модуль *Wi-Fi* - пристрій повинен мати можливість підключатися до мережі Інтернет для отримання останніх курсів криптовалют через *API* підтримка запитів *API* - мікроконтролер повинен мати можливість виконувати *HTTP*-запити до сервера криптообміну та отримувати відповідь у форматі *JSON*; сумісність з *TFT*-дисплеєм *ST*7735*S* - підтримується протокол *SPI* для підключення та керування графічним дисплеєм.

*TTP*223 Підключення сенсорних кнопок - пристрій повинен мати можливість перемикання між режимами відображення за допомогою ємнісних сенсорних кнопок. Низьке енергоспоживання - пристрій повинен споживати мінімум енергії для забезпечення автономної роботи. Доступність і простота інтеграції - вартість платформи повинна бути економічно обґрунтованою, а для прискорення розробки повинна бути хороша бібліотека і документація.

Arduino Uno – не підходить, оскільки не має вбудованого *Wi-Fi* та має надто малий об’єм пам’яті, що ускладнює роботу з *API*-запитами та відображенням графічних даних. Пристрій зображено на рисунку 2.1.

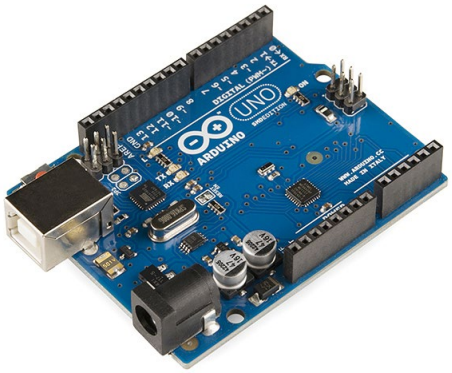


Рисунок 2.1 – *Arduino Uno*

Raspberry Pi Zero W – забезпечує потужність та можливість підключення до Інтернету, проте споживає багато енергії і вимагає складніших налаштувань для обробки *API*-запитів. Пристрій зображено на рисунку 2.2.

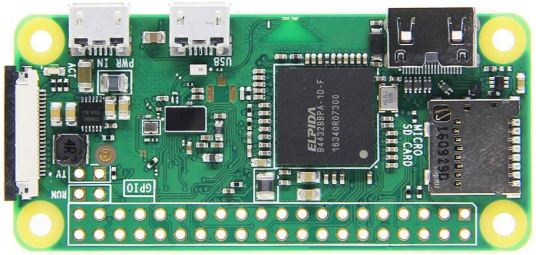


Рисунок 2.2 – *Raspberry Pi Zero W*

*ESP*32 – більш продуктивний, ніж *ESP*8266, і має двоядерний процесор, однак його можливості надлишкові для моніторингу курсів криптовалют. Крім того, вищий рівень енергоспоживання може бути недоліком у довготривалій автономній роботі пристрою. Пристрій зображено на рисунку 2.3.

Рисунок 2.3 – *ESP*32

*NodeMCU ESP*8266 – ідеально підходить для цього проекту, оскільки має вбудований *Wi-Fi*, низьке енергоспоживання, підтримку *API*-запитів та *SPI*-протокол для роботи з дисплеєм. Пристрій зображено на рисунку 2.4.

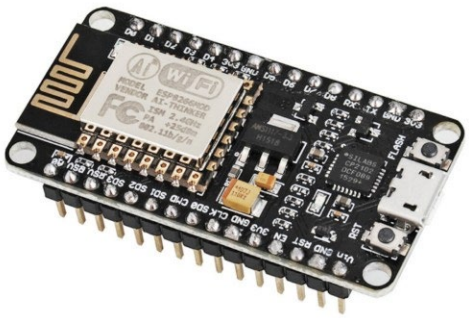


Рисунок 2.4 – *NodeMCU ESP*8266

На основі порівняльного аналізу було визначено, що *NodeMCU ESP*8266 є оптимальним мікроконтролером для реалізації проекту моніторингу курсів криптовалют *BTC* та *ETH*. Він пропонує стабільне інтернет-з'єднання, ефективну взаємодію з *API*, підтримку *TFT*-дисплея та сенсорних кнопок, низьке енергоспоживання та ідеально підходить для автономної роботи.

Мікроконтролер має вбудований модуль *Wi-Fi*, який дозволяє пристрою підключатися до Інтернету без необхідності використання додаткових модулів. Це дозволяє йому отримувати дані в режимі реального часу від *API* криптовалютних бірж, що важливо для моніторингу курсів *BTC* і *ETH*.

Підтримка *API* запитів - *NodeMCU ESP*8266 дозволяє робити *HTTP* запити до зовнішнього сервера і отримувати відповідь у форматі *JSON*. Це дозволяє пристрою інтегруватися з такими сервісами, як *CoinLore API* і *Binance API*, які надають актуальні курси криптовалют; бібліотека *HTTPClient.h* значно спрощує взаємодію з *API* і підвищує ефективність розробки. Сумісність з *TFT*-дисплеями *ST*7735*S* - пристрій підтримує протокол *SPI* і може підключати та керувати графічними дисплеями *ST*7735*S*.

Це забезпечує зручний візуальний інтерфейс, який відображає інформацію про курси криптовалют у зрозумілому форматі. Гнучка конфігурація дозволяє змінювати параметри дисплея та оновлювати дані в режимі реального часу.

Підключення сенсорної кнопки *TTP*223 - *NodeMCU ESP*8266 має достатньо виходів *GPIO* для інтеграції ємнісної сенсорної кнопки *TTP*223. Сенсорні кнопки використовуються для зміни режиму відображення та покращення взаємодії з користувачем. Це дозволяє користувачеві змінювати робочі параметри і перемикатися між відображенням курсу *BTC* і *ETH* без необхідності підключення до комп'ютера або додаткових пристроїв. Низьке енергоспоживання - мікроконтролер *NodeMCU ESP*8266 споживає значно менше енергії, ніж, наприклад, *Raspberry Pi Zero W* або *ESP*32. Це означає, що їх можна використовувати в автономному режимі без необхідності частого підключення до електромережі. У режимі очікування вони споживають менше 0,1 Вт, що значно знижує витрати на електроенергію і продовжує термін служби пристрою.

Доступність - *NodeMCU ESP*8266 значно дешевше, ніж *Raspberry Pi* і *ESP*32. При середній вартості $3-5, він пропонує відмінний баланс між ціною і продуктивністю і є економічно ефективним рішенням для розробки пристроїв моніторингу криптовалют. Широка бібліотечна та документаційна підтримка - *NodeMCU ESP*8266 поставляється з великою кількістю готових бібліотек і докладною документацією для прискорення процесу розробки. Оскільки цей мікроконтролер широко використовується в *IoT*-проектах, існує безліч

відкритих вихідних кодів і форумів, які спрощують налагодження пристрою і вирішення можливих технічних проблем.

Технічні характеристики *NodeMCU ESP*8266

- Мікроконтролер *NodeMCU ESP*8266 має такі ключові характеристики:
- Процесор: *Tensilica L*106 32-бітний, 80 МГц
- ОЗП (*RAM*): 80 КБ, що достатньо для обробки *API*-запитів та збереження отриманих даних
- Флеш-пам’ять: 4 МБ, що дозволяє зберігати код програми та налаштування
- Живлення: 3,3 В
- Кількість *GPIO*: 10 цифрових входів/виходів
- Комунікаційні протоколи: *SPI*, *I2C*, *UART*, *PWM*, *GPIO*, *ADC*
- *Wi-Fi*: 802.11 *b*/*g*/*n*, підтримка режиму *Access Point* та *Station*

Ці характеристики роблять *NodeMCU ESP*8266 ідеальним варіантом для *IoT*-проєктів, включаючи пристрої, що працюють із мережею, відправляють *HTTP*-запити та обробляють отриману інформацію.

Основні фактори, що визначили вибір цієї платформи:

1. Забезпечення з'єднання з Інтернетом для отримання даних про курс криптовалют у реальному часі.
2. Можливість інтеграції з *API* для автоматизованого отримання інформації без взаємодії користувача.
3. Підтримка графічного дисплея *ST*7735*S* для зручного відображення курсів *BTC* та *ETH*.
4. Автономність та енергоефективність, що забезпечує тривалий час роботи пристрою.
5. Доступність та економічна ефективність, що робить проект вигідним із фінансової точки зору.

Таким чином, завдяки своїм технічним характеристикам, низькому енергоспоживанню, простоті інтеграції та широкій підтримці бібліотек *NodeMCU ESP*8266 є оптимальним вибором для реалізації цього проекту.

Його використання дозволить створити компактний, продуктивний і економічно вигідний пристрій, який здатний отримувати, обробляти та відображати курси криптовалют у реальному часі без необхідності взаємодії користувача з зовнішніми програмами або додатками.

2.2 Модулі зв’язку та їх використання

Для коректної роботи пристроїв моніторингу криптовалют необхідно забезпечити стабільне з'єднання з інтернетом. Оскільки пристрої працюють в режимі реального часу, канал зв'язку повинен бути стабільним, енергоефективним і швидким: *NodeMCU ESP*8266 має вбудований модуль *Wi-Fi*, що є однією з його головних переваг. Це дозволяє пристрою підключатися до бездротової мережі та отримувати необхідну інформацію без використання додаткових модулів. У цьому розділі описано особливості використання *Wi-Fi* з'єднання, можливі альтернативи, принципи роботи мережі та програмну реалізацію з'єднання.

Сучасні пристрої Інтернету речей (*IoT*) використовують різні методи зв'язку для отримання та передачі даних. Вибір відповідного методу залежить від специфіки проекту, необхідної швидкості передачі, енергоспоживання та стабільності з'єднання. У цьому розділі описано основні технології зв'язку, що використовуються в пристроях Інтернету речей, та обґрунтовано вибір найбільш підходящого рішення для проектів моніторингу курсів криптовалют *BTC* та *ETH*. Методи зв'язку Інтернету речей можна розділити на три основні категорії: бездротові технології з підключенням до інтернету (*Wi-Fi*, *GSM*/4*G*/5*G*, *Ethernet*). Енергоефективні бездротові технології малого радіусу дії (*Bluetooth Low Energy*, *Zigbee*, *Z-Wave*). Бездротові технології великого радіусу дії (*LoRa*, *NB-IoT*, *Sigfox*). Їх порівняння можна побачити у таблиці 2.1

Таблиця 2.1 – Основні методів зв’язку, їхні переваги та недоліки

Метод зв’язку	**Переваги**	**Недоліки**
Wi-Fi (802.11 *b*/*g*/*n*)	Висока швидкість передачі даних, просте підключення до Інтернету, підтримка більшості мікроконтролерів	Високе енергоспоживання, залежність від зони покриття
Ethernet	Надійне з’єднання, висока швидкість, безпека	Вимагає проводів, обмежена мобільність
Bluetooth Low Energy (*BLE*)	Дуже низьке енергоспоживання, швидке підключення, висока сумісність зі смартфонами	Обмежений радіус дії, не підключається до Інтернету
GSM/3*G*/4*G*/*LTE*	Підключення у віддалених місцях, стабільний Інтернет	Потребує *SIM*-карту, високе енергоспоживання, додаткові витрати
LoRa (*Long Range*)	Велика дальність передачі, низьке енергоспоживання	Низька швидкість передачі даних, потребує *LoRaWAN*-шлюзу

Wi-Fi - один з найпоширеніших методів зв'язку для пристроїв *IoT*, які потребують високошвидкісного обміну даними та постійного підключення до інтернету. Його основними перевагами є висока швидкість передачі даних (до 600 Мбіт/с для 802.11*n*), простота створення мережі та підтримка більшості мікроконтролерів, таких як *ESP*8266 та *ESP*32 (рисунок 2.5).

Рисунок 2.5 – Логотип *Wi-Fi*

Однак він має відносно високе енергоспоживання, що може бути критичним для автономних пристроїв. *Ethernet* забезпечує надійне дротове з'єднання, що дозволяє уникнути перешкод у бездротових мережах, але не підходить для мобільних пристроїв, оскільки вимагає фізичного підключення до мережі

Bluetooth Low Energy (*BLE*) відрізняється низьким енергоспоживанням і швидким з'єднанням, але зв'язок обмежений відстанню (10-30 м) і не може підключатися до інтернету без шлюзу; *GSM*/3*G*/4*G*/*LTE* використовується для пристроїв, які повинні працювати у віддалених районах без *Wi-Fi*, але вимагає наявності *SIM*-карти і споживає більше енергії, що робить його менш привабливим в якості автономного *IoT* Не привабливий як рішення

Аналіз методів зв'язку для *IoT*-пристроїв, *Wi-Fi* (802. 11 *b*/*g*/*n*) є найбільш придатним для проекту. Він забезпечує високу швидкість передачі даних, необхідну для отримання курсів криптовалют в реальному часі; він є безкоштовним у використанні, оскільки не вимагає *SIM*-карти або додаткових платежів. Це стабільне з'єднання і сумісне з *API* криптовалютних бірж; воно працює на мікроконтролері *NodeMCU ESP*8266, що дозволяє легко інтегрувати його в системи.

Тому *Wi-Fi* є найкращим способом зв'язку для пристроїв для відстеження криптовалют *BTC* і *ETH*, забезпечуючи надійне з'єднання з *API* і стабільне оновлення даних у реальному часі.

Програмна реалізація підключення до *Wi-Fi*.

Для підключення до бездротової мережі використовується бібліотека *ESP*8266*WiFi.h*, яка дозволяє пристрою працювати через *Wi-Fi* з'єднання. Це дозволяє пристрою отримувати дані про курси криптовалют у реальному часі за допомогою *API* біржового сервісу. Наступний код на рисунку 2.6 підключає пристрій до *Wi-Fi*, перевіряє його стан та отримує *IP*-адресу. Таблиця 2.2, містить опис роботи цього коду.

```
bool attemptWiFiConnection() {
  if (strlen(wifiSettings.staSSID) == 0) {
    Serial.println("[WiFi] Нет сохранённого SSID для STA.");
    return false;
  }
  Serial.print("[WiFi] Подключаемся к: ");
  Serial.println(wifiSettings.staSSID);

  WiFi.begin(wifiSettings.staSSID, wifiSettings.staPass);

  unsigned long start = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - start < 15000) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  if (WiFi.status() == WL_CONNECTED) {
    Serial.print("[WiFi] STA Ok! IP:");
    Serial.println(WiFi.localIP());
    return true;
  } else {
    Serial.println("[WiFi] Не удалось подключиться.");
    return false;
  }
}

void setupAccessPoint() {
  // Если пароль пуст, AP без пароля
  if (strlen(wifiSettings.apPass) == 0) {
    Serial.print("[WiFi] AP открытая: ");
    Serial.println(wifiSettings.apSSID);
    WiFi.softAP(wifiSettings.apSSID);
  } else {
    Serial.print("[WiFi] AP с паролем: ");
    Serial.println(wifiSettings.apSSID);
    WiFi.softAP(wifiSettings.apSSID, wifiSettings.apPass);
  }
  Serial.print("[WiFi] AP IP: ");
  Serial.println(WiFi.softAPIP());
}
```

Рисунок 2.6 – Код програмної реалізація підключення до *Wi-Fi*

Таблиця 2.2 – Опис роботи коду

Крок	**Опис дії**
1	Визначаємо *SSID* та пароль *Wi-Fi*-мережі, до якої підключатиметься пристрій.
2	Використовуємо *WiFi.begin*(*ssid*, *password*); для ініціалізації підключення**.**
3	Очікуємо встановлення з’єднання, виводячи "..." у серійний порт**.**
4	Після успішного підключення виводимо *IP*-адресу пристрою в серійний порт.

Цей код дозволяє автоматично підключати пристрій до *Wi-Fi* при кожному ввімкненні, що забезпечує стабільний доступ до Інтернету для отримання актуальних курсів *BTC* та *ETH*.

У випадку, якщо *Wi-Fi*-з’єднання переривається, пристрій повинен автоматично відновлювати підключення.

Також у даному коді є функція *attemptWiFiConnection*(), вона відповідає за підключення до *Wi-Fi*, проте вона викликається лише в процесі ініціалізації *setup*()

Ця функція перевіряє статус підключення до *Wi-Fi* та автоматично виконує перепідключення у разі його втрати.

Якщо *Wi-Fi* недоступний або з’єднання нестабільне, можна розглянути інші методи зв’язку. У таблиці 2.3 наведено альтернативи *Wi-Fi* та їх основні характеристики.

Таблиця 2.3 – Порівняння методів зв’язку

Метод зв’язку	**Переваги**	**Недоліки**
GSM/4*G* модулі (*SIM*800*L*, *SIM*900)	Працює у віддалених місцях, не потребує *Wi-Fi*	Потребує *SIM*-карту, велике енергоспоживання
Ethernet (*ENC*28*J*60, *W*5500)	Надійне підключення, стабільний зв’язок	Вимагає проводів, не підходить для мобільних пристроїв
LoRa (*SX*1278, *RFM*95*W*)	Велика дальність передачі, низьке енергоспоживання	Низька швидкість, не підходить для швидкого оновлення даних

Проте, у цьому проекті *Wi-Fi* є найкращим вибором, оскільки:

- Забезпечує високу швидкість передачі даних.
- Просто інтегрується з мікроконтролером *NodeMCU ESP*8266.

*NodeMCU ESP*8266 оснащений вбудованим *Wi-Fi*-модулем, що дозволяє легко інтегрувати пристрій у бездротову мережу та забезпечити отримання актуальних курсів криптовалют у режимі реального часу.

Основні переваги використання *Wi-Fi* у цьому проекті: висока швидкість передачі даних, що дозволяє швидко отримувати курс *BTC* та *ETH*. Безкоштовний доступ до Інтернету, що не вимагає додаткових витрат. Простота

інтеграції з *API* криптовалютних сервісів. Можливість оновлення даних у режимі реального часу та їхнього виведення на дисплей.

Таким чином, використання *Wi-Fi* та бібліотеки *ESP*8266*WiFi.h* дозволяє реалізувати надійний та ефективний зв'язок для моніторингу курсів криптовалют у цьому проекті.

2.3 Вибір *API* для отримання даних про курс криптовалют

При розробці пристрію для моніторингу курсів криптовалют *BTC* і *ETH*, потрібно інтегрувати *API*-сервіс, який дозволяє отримувати актуальні дані в режимі реального часу.

При виборі *API* для проекту враховуються кілька ключових факторів, які впливають на його ефективність та простоту інтеграції. Основні критерії вказані у таблиці 2.4

Таблиця 2.4 – Критерії вибору *API*

Критерій	**Опис**
Достовірність та точність даних	*API* повинен отримувати дані з надійних джерел (біржі або перевірені агрегатори), щоб мінімізувати розбіжності у курсах.
Швидкість оновлення інформації	Чим менше затримка оновлення, тим точніший моніторинг ринку. Ідеальне оновлення – кожні кілька секунд або хвилин**.**
Доступність *API*	Важливо, щоб *API* був безкоштовним або мав доступну тарифну політику для необмеженого отримання курсів криптовалют.
Формат відповіді	*API* повинен повертати структуровані дані у форматі *JSON***,** оскільки це найбільш зручний формат для обробки *ESP*8266.
Простота інтеграції з *NodeMCU ESP*8266	*API* повинен підтримувати *HTTP*-запити, щоб дані можна було легко отримувати за допомогою бібліотеки *HTTPClient.h*.

Якщо *API* не відповідає хоча б одному з цих критеріїв, його інтеграція в *IoT*-пристрій стає складнішою або менш ефективною.

API криптовалют за джерелом отримання даних поділяються на дві основні категорії (таблиця 2.5).

Таблиця 2.5 – Джерела отримання даних *API*

Тип *API*	**Джерело даних**	**Особливості**
API бірж	Прямо з криптовалютних бірж (*Binance*, *Kraken*, *Coinbase*)	Дані отримуються безпосередньо з біржі, що забезпечує найвищу точність та оновлення в реальному часі.
Агрегатори *API*	*Coinlore*, *CoinMarketCap*, *CryptoCompare*	*API* використовує дані з кількох бірж, що дозволяє усереднювати курс і забезпечує стабільну роботу навіть при збої однієї з бірж.

Такі сервіси часто потребують *API*-ключів та автентифікації, що ускладнює інтеграцію.

На основі наведених критеріїв було розглянуто чотири найпопулярніші *API* в таблиці 2.6, які широко використовуються для отримання курсів криптовалют.

На основі аналізу було прийнято рішення використовувати *Coinlore API*, оскільки він найкраще підходить для інтеграції з *NodeMCU ESP*8266.

Основні причини вибору:

- Повністю безкоштовний, не потребує *API*-ключа.
- Простий у використанні – достатньо виконати *HTTP*-запит, без складної автентифікації.
- Стабільний сервер – швидке оновлення даних кожні 1-2 хвилини.
- Підтримує великий список криптовалют – понад 5000 активів, включаючи *BTC* та *ETH*.
- Формат *JSON*, що легко обробляється *ESP*8266.

Таблиця 2.6 – Найпопулярніші *API*

API	**Джерело даних**	**Переваги**	**Недоліки**
Coinlore API	Агреговані дані з кількох бірж	✅ Безкоштовний, ✅ Підтримує понад 5000 криптовалют, ✅ Не потребує *API*-ключа	❌ Дані оновлюються кожні 1-2 хвилини
CoinMarketCap API	Агреговані дані з понад 300 бірж	✅ Висока точність, ✅ Підтримка великої кількості активів	❌ Потребує *API*-ключа, ❌ Обмеження у безкоштовній версії
Binance API	Дані напряму з біржі *Binance*	✅ Оновлення в реальному часі, ✅ Висока точність	❌ Потребує реєстрації та *API*-ключа, ❌ Підтримує лише активи *Binance*
CryptoCompare API	Агреговані дані та аналітика	✅ Історичні та поточні дані, ✅ Додаткові графічні інструменти	❌ Потребує *API*-ключа, ❌ Деякі функції доступні лише у платній версії

Coinlore API дозволяє отримати актуальний курс *BTC, ETH та DOGE* за таким запитом коду, наведеного на рисунку 2.7.

```
fetchCrypto("https://api.coinlore.net/api/ticker/?id=90", cryptos[0]); // BTC
fetchCrypto("https://api.coinlore.net/api/ticker/?id=80", cryptos[1]); // ETH
fetchCrypto("https://api.coinlore.net/api/ticker/?id=2",  cryptos[2]); // DOGE
```

Рисунок 2.7 – Приклад отримання актуальних курсів криптовалют

Відповідь у форматі *JSON*:

```
{
  "bitcoin": {
    "usd": 62345
  }
}
```

Цей формат дозволяє легко витягнути значення курсу та вивести його на екран пристрою. Наступний код демонструє на рисунку 2.8, як можна отримати курс *BTC* за допомогою *NodeMCU ESP*8266 та бібліотеки *HTTPClient.h*, *WiFi.h*, *WiFiClientSecure.h*, *ArduinoJson.h*.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
const char* coinUrl = "https://api.coinlore.net/api/ticker/?id=90";

WiFiClientSecure client;

void setup() {
  Serial.begin(115200);
  delay(1000);
  Serial.println("Connecting to Wi-Fi...");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("Connected, IP: ");
  Serial.println(WiFi.localIP());

  client.setInsecure();
  HTTPClient http;
  http.begin(client, coinUrl);
  int httpCode = http.GET();
  if (httpCode == HTTP_CODE_OK) {
    String payload = http.getString();
    Serial.println("Received payload:");
    Serial.println(payload);

    DynamicJsonDocument doc(256);
    DeserializationError error = deserializeJson(doc, payload);
    if (error) {
      Serial.print("JSON parse error: ");
      Serial.println(error.c_str());
    } else {
      float price = doc[0]["price_usd"].as<float>();
      Serial.print("Цена Bitcoin (USD): ");
      Serial.println(price, 2);
    }
  } else {
    Serial.print("Ошибка запроса, HTTP код: ");
    Serial.println(httpCode);
  }
  http.end();
}

void loop() {
  delay(60000);
}
```

Рисунок 2.8 – Приклад реалізації запиту через *Coinlore API*

Ось чому *Coinlore API* є найкращим вибором для отримання останніх курсів криптовалют. Він простий у використанні, безкоштовний, стабільний і може бути легко інтегрований в *IoT*-пристрої на базі *ESP*8266 *NodeMCU*.

2.4 Огляд технологій для створення веб-інтерфейсу

Для підвищення функціональності пристроїв моніторингу курсів криптовалют *BTC* та *ETH* бажано реалізувати веб-інтерфейс, який надає можливість переглядати поточні курси без використання додаткових екранів або встановлення спеціального програмного забезпечення. Веб-інтерфейс

дозволяє користувачам отримувати інформацію безпосередньо з веб-браузера пристрою, підключеного до локальної мережі *Wi-Fi*. Це дозволяє користувачам переглядати поточні курси *BTC* і *ETH*, а також віддалено отримувати доступ до налаштувань пристрою і контролювати його поведінку.

Реалізація веб-інтерфейсу відкриває наступні важливі можливості:

1. Відображення поточних курсів *BTC* та *ETH* на веб-сторінці в режимі реального часу.
2. Інформація оновлюється автоматично без перезавантаження сторінки, тому користувачі можуть бачити актуальні курси без необхідності вручну оновлювати веб-сторінку.
3. Можливість змінювати налаштування пристрою, наприклад, частоту оновлення курсів.

Віддалений доступ через браузер з будь-якого пристрою (смартфона, планшета або комп'ютера), підключеного до локальної мережі *Wi-Fi*. Інтеграція веб-інтерфейсу дозволяє користувачам переглядати курси *BTC* і *ETH* в зручному форматі і керувати роботою пристрою без необхідності фізичного підключення до нього.

Основними технологіями, що були використані при розробці, є *ESP*8266*WebServer* - реалізує локальний веб-сервер на мікроконтролері та обробляє *HTTP*-запити від клієнта (браузера); *AJAX* + *JavaScript* - забезпечує динамічне оновлення даних та відповідає за візуалізацію графічного інтерфейсу сторінки *HTML* + *CSS* - робить веб-інтерфейс зрозумілим та зручним у використанні. [2]

Разом ці технології дозволяють *NodeMCU ESP*8266 виступати в ролі веб-сервера, дозволяючи користувачам отримувати оновлення курсів криптовалют та інших даних у зручному форматі.

Розглянемо основні технології, які будуть використані для реалізації веб-інтерфейсу, які представлені у таблиці 2.7

Таблиця 2.7 – Огляд технологій для створення веб-інтерфейсу

Технологія	**Призначення**
*ESP*8266*WebServer*	Реалізація локального веб-сервера на *ESP*8266, що дозволяє обробляти *HTTP*-запити.
AJAX + JavaScript	Забезпечує динамічне оновлення даних без перезавантаження сторінки.
HTML + CSS	Використовується для створення інтерфейсу користувача.

Кожна з цих технологій має важливе значення у розробці зручного та швидкого веб-інтерфейсу для пристрою.

Для створення веб-інтерфейсу пристрій *NodeMCU ESP*8266 повинен виконувати роль локального веб-сервера. Це дозволить користувачам підключатися до пристрою через *Wi-Fi* та переглядати інформацію про курс криптовалют у браузері.

Для реалізації використовується бібліотека *ESP*8266*WebServer.h*, яка дозволяє створювати веб-сервер та обробляти *HTTP*-запити. На рисунку 2.9 наведено приклад реалізації веб-сервера на *ESP*8266 та опис коду у таблиці 2.8.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

const char* ssid = "YourWiFiNetwork";
const char* password = "YourWiFiPassword";

ESP8266WebServer server(80); // Веб-сервер працює на порту 80

void handleRoot() {
  server.send(200, "text/html", "<h1>Crypto Monitor</h1><p>Курс BTC: 62000$</p>");
}

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Wi-Fi підключено!");

  server.on("/", handleRoot);
  server.begin();
}

void loop() {
  server.handleClient();
}
```

Рисунок 2.9 – Типова базова реалізація веб-сервера на *ESP*8266

Таблиця 2.8 – Опис коду

Крок	**Опис**
1	Ініціалізація *Wi-Fi*-з'єднання (*WiFi.begin*(*ssid*, *password*);).
2	Створення веб-сервера на порту 80.
3	Обробка *HTTP*-запиту на головну сторінку /, вивід курсу *BTC*.
4	Очікування *HTTP*-запитів у циклі *loop*() (*server.handleClient*();).

Результатом виконання цього коду є веб-сторінка, яка відображає курс *BTC*, доступна за локальною *IP*-адресою пристрою (наприклад, 192.168.4.1).

Таке поєднання технологій забезпечує ефективну, стабільну та автономну роботу пристрою і дозволяє користувачам переглядати інформацію за допомогою стандартного браузера.

РОЗДІЛ 3
РОЗРОБКА ПРИСТРОЮ ДЛЯ МОНІТОРИНГУ КУРСУ КРИПТОВАЛЮТ

3.1 Архітектура системи

Архітектура пристрою поєднує в собі апаратні та програмні компоненти, які взаємодіють і виконують поставлені завдання. Основними функціями системи є отримання даних з інтернету через *Wi-Fi* (підключення до *API* криптобіржі). Обробка та відображення отриманих *JSON* даних. Інтерактивна взаємодія з користувачем за допомогою сенсорних кнопок. Віддалений доступ через веб-інтерфейс, що дозволяє користувачеві переглядати інформацію з будь-якого пристрою. Система розроблена таким чином, щоб дозволити користувачам переглядати курси криптовалют та інші дані у двох режимах: локальному - через *TFT*-дисплей *ST*7735*S*, підключений до мікроконтролера. Віддалений - через веб-інтерфейс, розташований в локальній мережі *Wi-Fi*. Дані оновлюються автоматично, що дозволяє користувачам оперативно відстежувати зміни без необхідності оновлювати сторінку або вводити команди вручну. Основні компоненти апаратної частини, програмної частини та структуру програмного забезпечення наведено у таблицях 3.1 – 3.4.

Таблиця 3.1 – Основні апаратні компоненти системи

Компонент	**Функціональне призначення**
Мікроконтролер *NodeMCU ESP*8266	Відповідає за *Wi-Fi*-з'єднання, отримання та обробку *API*-даних, керування дисплеєм і веб-інтерфейсом.
Дисплей *TFT ST*7735*S*	Відображає поточний курс криптовалют *BTC* та *ETH*.
Сенсорні кнопки *TTP*223	Використовуються для зміни режимів відображення.
Мережевий модуль *Wi-Fi* (вбудований у *ESP*8266)	Виконує з'єднання з Інтернетом для отримання даних через *API*.
Живлення (*USB* 5*V* або акумулятор 3.7*V*)	Забезпечує стабільну роботу пристрою.

Система містить набір апаратних модулів, які взаємодіють між собою для виконання завдань.

Таблиця 3.2 – Основні компоненти програмної частини

Компонент	**Функціональне призначення**
Wi-Fi-з'єднання	Підключення до локальної мережі для отримання даних із зовнішніх серверів.
HTTP-запити до *API CoinLore*	Отримання курсу *BTC* та *ETH* у форматі *JSON*.
Обробка отриманих *JSON*-даних	Виділення потрібної інформації (курс *BTC* у доларах).
Відображення на дисплеї	Оновлення інформації кожні 60 секунд.
Реалізація веб-інтерфейсу	Можливість перегляду курсу через браузер.
Реакція на сенсорні кнопки	Зміна режиму відображення (наприклад, курс *BTC* → курс *ETH*).

Таблиця 3.3 – Структура програмного забезпечення

Етап	**Опис**
1. Ініціалізація мікроконтролера	Налаштування *Wi-Fi*, дисплея та кнопок.
2. Підключення до мережі *Wi-Fi*	Підключення пристрою до локальної мережі *Wi-Fi*.
3. Отримання даних про курс криптовалют через *API*	Запит до *API CoinLore* та отримання відповіді у форматі *JSON*.
4. Обробка отриманої *JSON*-відповіді	Виділення потрібних даних (курс *BTC*, *ETH*) для подальшого відображення.
5. Оновлення даних на дисплеї	Відображення отриманих курсів криптовалют на *TFT*-дисплеї.
6. Обробка натискання сенсорних кнопок	Зміна режиму перегляду інформації (перемикання між *BTC* і *ETH*).
7. Обслуговування веб-запитів у локальній мережі	Обробка запитів користувачів у браузері через веб-інтерфейс.
8. Оновлення інформації кожні 60 секунд	Автоматичне оновлення даних.

Таблиця 3.4 – Основні переваги архітектури системи

Перевага	**Опис**
Автономна робота	Пристрій працює без комп'ютера, отримуючи дані з Інтернету автоматично.
Безкоштовний *API*	Використання *CoinLore API* дозволяє отримувати курси криптовалют без додаткових витрат.
Гнучке керування	Можливість перемикання режимів за допомогою сенсорних кнопок.
Локальне та віддалене відображення	Інформація доступна як на дисплеї, так і у браузері через *Wi-Fi*.
Низьке енергоспоживання	*ESP*8266 дозволяє працювати від зовнішнього живлення або акумулятора.

Ці таблиці дають структурований опис програмної частини пристрою, його основних функцій та переваг архітектури. [3]

Принцип роботи пристрою

- Вмикається мікроконтролер *ESP*8266 та ініціалізуються всі модулі. Далі *NodeMCU* підключається до *Wi-Fi* мережі *ESP*8266 і перевіряє стан з'єднання.
- Після встановлення з'єднання мікроконтролер надсилає *HTTP*-запит до *API CoinLore*, НБУ *API*, погоди та повітряної тривоги для отримання поточних курсів *BTC* та *ETH*.
- *API* повертає відповідь у форматі *JSON*, в якому пристрій виводить необхідну інформацію (курси *BTC* та *ETH* в *USD*). Отримані дані передаються на *TFT*-дисплей *ST*7735*S*, що дозволяє користувачеві переглядати поточні дані.
- Ці дані також передаються у веб-інтерфейс, доступний з браузера на пристрої, підключеному до тієї ж мережі *Wi-Fi*. Користувачі можуть використовувати сенсорні кнопки для зміни режиму відображення і перемикання між потрібними режимами. Веб-інтерфейс дозволяє користувачам переглядати усю інформацію у браузері.

Основні переваги архітектури системи

- Автономна робота – пристрій функціонує незалежно від комп’ютера або смартфона, автоматично отримуючи дані з Інтернету.
- Безкоштовний *API* – використання *CoinLore API* дозволяє отримувати курси криптовалют без додаткових витрат.
- Гнучке керування – можливість перемикання режимів відображення курсів за допомогою сенсорних кнопок.
- Локальне та віддалене відображення – інформація доступна як на *TFT*-дисплеї, так і у веб-інтерфейсі через *Wi-Fi*.
- Низьке енергоспоживання – *ESP*8266 дозволяє працювати від зовнішнього живлення або акумулятора, що забезпечує енергоефективність системи

Архітектура системи моніторингу криптовалют *BTC* та *ETH* забезпечує ефективну взаємодію між апаратними та програмними компонентами для отримання, обробки та відображення фінансових даних в режимі реального часу. Пристрій побудований на базі *NodeMCU ESP*8266, підключений до *Wi-Fi* і отримує актуальну інформацію через *API CoinLore*. Отримані курси криптовалют відображаються як на *TFT*-дисплеї, так і на веб-інтерфейсі *ST*7735*S*, пропонуючи гнучкість використання та віддалений доступ до даних. Поєднання апаратних і програмних компонентів робить пристрій компактним, економічно вигідним і простим у використанні, дозволяючи користувачам легко відстежувати зміни на ринку криптовалют без необхідності використання комп'ютера або смартфона.

3.2 Схема підключення компонентів

Основними завданнями схеми з'єднання є наступні: забезпечення стабільного зв'язку між мікроконтролером і мережею *Wi-Fi* та отримання актуальних курсів криптовалют в режимі реального часу; швидке відображення

інформації для точної передачі даних на екран через шину *SPI* між *ESP*8266 і *TFT ST*7735*S*. Гарантована робота сенсорних кнопок *TTP*223, що дозволяє перемикати режим відображення на екрані.

Живлення всіх компонентів здійснюється від стабільного джерела напруги 3,3 В, що відповідає специфікаціям *ESP*8266 і периферійних пристроїв. Для підключення пристрою використовується стандартний набір інтерфейсів, включаючи *SPI* для обміну даними з дисплеєм і *GPIO* для взаємодії з сенсорною кнопкою. [5]

*NodeMCU ESP*8266 є основним обчислювальним модулем, який підключається до *Wi-Fi*-мережі для отримання актуальних курсів *BTC* та *ETH* через *CoinLore API*.

Завдяки вбудованому *Wi-Fi ESP*8266 може:

– Виконувати *HTTP*-запити та отримувати відповіді у форматі *JSON*. Працювати автономно, оновлюючи дані кожні 60 секунд. Забезпечувати веб-доступ до курсу криптовалют через браузер.
– Дисплей *TFT ST*7735*S* використовується для відображення даних. Він працює за *SPI*-протоколом, що дозволяє швидко оновлювати зображення на екрані. Можемо побачити його на рисунку 3.1
– Висока швидкість передачі – забезпечує швидке оновлення екрану. Менше проводів у порівнянні з паралельними інтерфейсами.

Рисунок 3.1 – Дисплей *TFT ST*7735*S*

3. Підключення сенсорних кнопок *TTP*223 до *ESP*8266

Сенсорні кнопки *TTP*223 (рисунок 3.2) дозволяють перемикати режими роботи пристрою без механічних натискань, що підвищує їхню надійність.

Рисунок 3.2 – Сенсорна кнопка *TTP*223

Таблиця 3.5 – Підключення та фунціонал сенсорних кнопок

Сигнал кнопки	***GPIO ESP*8266**	**Функція**
Кнопка *A*	*D*1 (*GPIO*5)	Перемикання між основними вкладками (наприклад *BTC* та *ETH*)
Кнопка *B*	*D*6 (*GPIO*12)	Перемикання між підрозділами основних вкладок

Підключення здійснюється напряму до *GPIO* (таблиця 3.5) без використання додаткових резисторів, оскільки *TTP*223 працює в режимі відкритого колектора.

4. Живлення пристрою

Усі компоненти системи працюють від 3.3*V*, що забезпечує їхню стабільну роботу без перегріву та перевантаження мікроконтролера.

Живлення може подаватися через *MicroUSB* або зовнішній акумулятор 3.7*V*, що забезпечує можливість автономної роботи пристрою.

Принцип роботи схеми підключення

*ESP*8266 отримує живлення та ініціалізує всі підключені компоненти.

NodeMCU встановлює з’єднання з мережею *Wi-Fi* та надсилає *HTTP*-запит до *Coinlore API*. Отримана *JSON*-відповідь обробляється, і необхідні дані

передаються: на *TFT*-дисплей для локального перегляду, у веб-інтерфейс для віддаленого перегляду через браузер.

Переваги запропонованої схеми підключення

- Автономна робота – пристрій працює без комп'ютера, автоматично отримуючи дані з Інтернету. Швидкий обмін даними – *SPI*-з'єднання забезпечує миттєве оновлення інформації на дисплеї. Гнучке керування – користувач може перемикати режими через сенсорні кнопки.
- Надійність та економічність – використання *Wi-Fi* та безкоштовного *API* мінімізує додаткові витрати. Розроблена схема підключення забезпечує ефективну взаємодію між всіма компонентами системи, що дозволяє пристрою отримувати, обробляти та відображати актуальні курси криптовалют *BTC* та *ETH* у реальному часі.

Схему підключення основних компонентів, можна побачити у таблиці 3.6.

Таблиця 3.6 – Підключення TFT ST7735S до ESP8266

Пін дисплея	**Пін *NodeMCU***	**Функція**
LED	3.3*V*	Живлення підсвітки дисплея
SCK	*D*5 (*GPIO*14)	Тактовий сигнал *SPI*
SDA (*MOSI*)	*D*7 (*GPIO*13)	Передача даних на дисплей
AO (*DC*)	*D*4 (*GPIO*2)	Вибір режиму: передача даних або команд
RESET	*D*3 (*GPIO*0)	Апаратний скидання (*Reset*) дисплея
CS	*D*8 (*GPIO*15)	Вибір пристрою (*Chip Select*)
GND	*GND*	Загальне заземлення

*TFT ST*7735*S* працює по *SPI*-шині, яка забезпечує високу швидкість обміну даними, що необхідно для миттєвого оновлення інформації на екрані.

Таблиця 3.7 – Підключення сенсорних кнопок TTP223 до ESP8266

Компонент	***ESP*8266 (*GPIO*)**	**Призначення**
Кнопка *A*	*D*1 (*GPIO*5)	Перемикання між основними вкладками (наприклад *BTC* та *ETH*)
Кнопка *B*	*D*6 (*GPIO*12)	Перемикання між підрозділами основних вкладок
VCC	3.3*V*	Живлення кнопок
GND	*GND*	Загальне заземлення

Сенсорні кнопки *TTP*223 працюють за принципом ємнісного сенсора, що дозволяє управляти пристроєм без механічного контакту, схему підключень можна побачити у таблиці 3.7. Це покращує надійність і зручність експлуатації.

Таблиця 3.8 – Живлення пристрою

Компонент	***ESP*8266 (*GPIO*)**	**Призначення**
*NodeMCU ESP*8266	*VIN*, *GND*	Живлення контролера
*TFT ST*7735*S*	3.3*V*, *GND*	Живлення дисплея
Сенсорні кнопки *TTP*223	3.3*V*, *GND*	Живлення кнопок

Усі компоненти працюють від 3.3*V*, що є стандартним рівнем напруги для *NodeMCU ESP*8266 (таблиця 3.8).

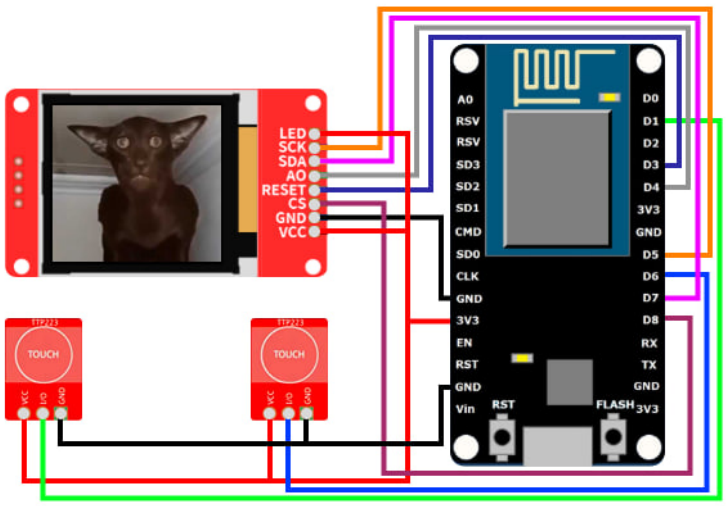


Рисунок 3.3 – Електрична схема пристрою

Розроблена електрична схема підключення на рисунку 3.3 забезпечує ефективну взаємодію між усіма компонентами системи, дозволяючи пристрою отримувати, обробляти і відображати поточні курси криптовалют *BTC* і *ETH* в режимі реального часу. Завдяки правильному підключенню: *ESP*8266 стабільно отримує і передає дані на дисплей через *Wi-Fi*; *TFT ST*7735*S* відображає останні фінансові дані без затримок; а *TFT ST*7735*S* може відображати останні курси *BTC* і *ETH* в режимі реального часу.

Режим відображення можна легко перемикати за допомогою сенсорних кнопок. Веб-інтерфейс забезпечує доступ до інформації через браузер.

3.3 Розробка програмного забезпечення мікроконтролера

Для належного функціонування пристрою необхідно розробити програмне забезпечення, яке забезпечує підключення до *Wi-Fi*, отримує курси криптовалют *BTC* та *ETH* через *API*, обробляє отримані *JSON*-дані, відображає їх на *TFT*-дисплеї *ST*7735*S* та реалізує веб-інтерфейс для віддаленого перегляду інформації. Програмне забезпечення базується на використанні *NodeMCU ESP*8266, який взаємодіє з компонентами пристрою та дозволяє йому працювати в автономному режимі. Основні функції програмного забезпечення.

Програмне забезпечення виконує наступні основні завдання: *Wi-Fi* з'єднання - встановлення стабільного мережевого з'єднання для доступу до *API* криптобіржі. Отримання даних про курс *BTC* і *ETH* в форматі *JSON* - здійснення *HTTP*-запитів до *API CoinLore* і обробка отриманих відповідей. Відображення отриманих даних на екрані - графічний інтерфейс оновлюється актуальною інформацією кожні 60 секунд.

Сенсорна кнопка B - Перемикання між підрозділами основних вкладок (перемикання між *BTC* і *ETH*).

Сенсорна кнопка A - Перемикання між основними вкладками (наприклад *BTC* та *ETH*), погода та інше. Для реалізації цих функцій використовуються

бібліотеки *ESP*8266*WiFi*, *HTTPClient*, *Adafruit GFX* і *ST*7735*S* для мережевих запитів, форматування *JSON* і відображення.

1. Налаштування *Wi-Fi*-з'єднання

Для підключення *ESP*8266 до бездротової мережі використовується бібліотека *ESP*8266*WiFi.h* (рисунок 3.5), яка дозволяє виконувати автоматичне підключення до *Wi-Fi* після запуску пристрою, налаштування підключення до *Wi-Fi* здійснюється через точку доступу нашого пристрою «*DataBox*», налаштування підключення здійснюється у нашому веб-інтерфейсі за адресою 192.168.4.1 (рисунок 3.4).

Рисунок 3.4 – Вкладка веб-інтерфейса з налаштуванням *Wi-Fi*

Код підключення до *Wi-Fi*

```
// Функція спроби підключення в режимі STA
bool attemptWiFiConnection() {
  if (strlen(wifiSettings.staSSID) == 0) {
    return false;
  }
  WiFi.begin(wifiSettings.staSSID, wifiSettings.staPass);
  unsigned long start = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - start < 15000) {
    delay(500);
  }
  return (WiFi.status() == WL_CONNECTED);
}

// У setup() - ініціалізація режиму і сама спроба
void setup() {
  // …
  WiFi.mode(WIFI_AP_STA);
  wifiStatus("Connecting STA...", ST77XX_YELLOW);
  isConnected = attemptWiFiConnection();
  if (isConnected) {
    wifiStatus("STA OK!", ST77XX_GREEN);
  } else {
    wifiStatus("STA Fail", ST77XX_RED);
  }
  // …
}
```

Рисунок 3.5 – Реалізація підключення до *Wi-Fi*

Опис коду на рисунку 3.5: Переводить модуль в режим одночасної роботи як станція (*STA*) і точка доступу (*AP*). Використовує *SSID* та пароль для підключення до локальної *Wi-Fi* мережі. Виводить *IP*-адресу пристрою після успішного підключення. У циклі *loop*() можна виконувати запити до *API*.

2. Отримання курсу *BTC* через *Coinlore API*

Для отримання актуального курсу *BTC* використовується *HTTP*-запит до *CoinLore API*. Запит повертає *JSON*-відповідь, яка містить курс *BTC* у доларах (рисунок 3.6).

```
// У функції updateAllData():
fetchCrypto("https://api.coinlore.net/api/ticker/?id=90", cryptos[0]);

// Саме fetchCrypto():
bool fetchCrypto(const String &url, CryptoData &data) {
  clientSecure.setInsecure();
  http.begin(clientSecure, url);
  int httpCode = http.GET();
  if (httpCode == HTTP_CODE_OK) {
    DynamicJsonDocument doc(256);
    DeserializationError error = deserializeJson(doc, http.getString());
    http.end();
    if (!error) {
      data.price = doc[0]["price_usd"].as<float>();
      return true;
    }
  }
  http.end();
  return false;
}
```

Рисунок 3.6 – Реалізація отримання курсу *BTC* через *Coinlore API*

Опис коду: Виконує *HTTPS*-запит до *API Coinlore* за адресою *https://api.coinlore.net/api/ticker/?id=*90. Отримує *JSON*-масив з об’єктів, парсить перший елемент і читає поле *price_usd*.

Записує значення ціни *Bitcoin* у структуру *CryptoData*. Виводить отриману інформацію у серійний монітор.

Функція викликається з *updateAllData*(), яка оновлює всі дані раз на хвилину.

3. Відображення курсу *BTC* на *TFT ST7735S*

Для виведення отриманих даних на дісплей (рисунок 3.7) використовується бібліотека *Adafruit GFX* та *Adafruit ST7735S,* що наведені у коді на рисунку 3.8.

Рисунок 3.7 – Відображення курсу крипти на дісплею *TFT*

Код підключення дисплея та виводу курсу *BTC*.

```
// Піни та об'єкт TFT-дисплея
#define TFT_CS   15
#define TFT_DC    2
#define TFT_RST   0
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

void setup() {
  // Ініціалізація дисплея
  tft.initR(INITR_144GREENTAB);
  tft.setRotation(4);
  tft.fillScreen(ST77XX_BLACK);
  tft.setTextWrap(false);
  // …інші налаштування…
}

void displayCrypto(int index) {
  // Очищуємо екран
  tft.fillScreen(ST77XX_BLACK);
  // Малюємо індикатори
  int indicatorWidth = 128 / 3;
  for (int i = 0; i < 3; i++) {
    tft.fillRect(i * indicatorWidth, 18, indicatorWidth - 2, 3,
                 (i == index) ? cryptos[index].color : ST77XX_WHITE);
  }
  // Виводимо символ (BTC, ETH або DOGE)
  tft.setTextColor(cryptos[index].color);
  drawCenteredText(cryptos[index].symbol, 2, 2);
  // Формуємо рядок з курсом і виводимо по центру
  String priceStr = formatNumber(cryptos[index].price) + " USD";
  tft.setTextSize(2);
  int16_t x1, y1; uint16_t w, h;
  tft.getTextBounds(priceStr, 0, 0, &x1, &y1, &w, &h);
  tft.setCursor((128 - w) / 2, 30);
  tft.print(priceStr);
}
```

Рисунок 3.8 – Код для виведення курсу на дісплей

Опис коду: Отримує курс *BTC* через *API Coinlore*: встановлює захищене *SSL*-з'єднання через *WiFiClientSecure*, виконує *HTTPS GET*-запит за допомогою *HTTPClient*, парсить *JSON* у *DynamicJsonDocument* і зчитує поле *price_usd*.

Виводить отриману інформацію на *TFT*-дисплей: очищує екран (*fillScreen*), малює індикатор активного пункту (три смужки вгорі), відображає символ "*BTC*" та форматовану ціну по центру (*drawCenteredText* і *print*).

Оновлює дані кожні 60 секунд: у *loop*() перевіряє інтервал часу, викликає *updateAllData*() для повторного запиту всіх *API* (крипта, валюти, погода, тривога) і одразу викликає *showCurrentPage*() для перемальовки дисплея із новими значеннями.

4. Відображення курсу національних валют (*USD*/*UAH*)

Рисунок 3.9 – Відображення курсу національних валют на дісплей

```
float fetchSingleCurrency(const char* valcode) {
  String url = "https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode=" + String(valcode);
  clientSecure.setInsecure();
  http.begin(clientSecure, url);
  int httpCode = http.GET();
  if (httpCode == HTTP_CODE_OK) {
    String payload = http.getString();
    http.end();
    int start = payload.indexOf("<rate>");
    int end   = payload.indexOf("</rate>");
    if (start >= 0 && end > start) {
      String rateStr = payload.substring(start + 6, end);
      rateStr.trim();
      return rateStr.toFloat();
    }
  } else {
    http.end();
  }
  return 0.0;
}

bool fetchCurrencies() {
  float usd = fetchSingleCurrency("USD");
  float eur = fetchSingleCurrency("EUR");
  bool success = false;
  if (usd > 0.0) { currencies[0].rate = usd; success = true; }
  if (eur > 0.0) { currencies[1].rate = eur; success = true; }
  return success;
}
```

Рисунок 3.10 – Код для виведення курсу національних валют на дісплей

Опис коду на рисунку 3.10: Відображає курс національних валют (*USD* або *EUR* у гривнях) на дисплей (рисунок 3.9). Очищує екран *TFT*-дисплея, малює індикатор активного пункту меню (дві смужки зверху), виводить символ валюти (*USD* або *EUR*) та форматований рядок із курсом у форматі “*XXX.XX*

UAH” по центру екрану. Викликається при перемиканні підменю або після оновлення даних для миттєвого відображення актуального курсу.

5. Відображення поточної погоди у місті.

Рисунок 3.11 – Відображення поточної погоди у місті на дісплей

```
// Запит погоди по координатах (API WeatherAPI)
bool fetchWeather(int index) {
  String locations[2] = {"47.90,33.38", "46.65,32.61"};
  String url = "https://api.weatherapi.com/v1/current.json?key=" +
               String(WEATHER_API_KEY) + "&q=" + locations[index] + "&lang=en";

  clientSecure.setInsecure();
  http.begin(clientSecure, url);
  int httpCode = http.GET();
  if (httpCode == HTTP_CODE_OK) {
    DynamicJsonDocument doc(1024);
    DeserializationError error = deserializeJson(doc, http.getString());
    http.end();
    if (!error) {
      JsonObject current = doc["current"];
      weather[index].temp      = current["temp_c"];
      weather[index].humidity  = current["humidity"];
      weather[index].cloud     = current["cloud"];
      weather[index].feelslike = current["feelslike_c"];
      weather[index].condition = current["condition"]["text"].as<String>();
      return true;
    }
  } else {
    http.end();
  }
  return false;
}

// Включення запиту в оновлення всіх даних
void updateAllData() {
  // ...
  fetchWeather(0);
  fetchWeather(1);
  // ...
}

// Відображення погоди на TFT-дисплеї
void displayWeather(int index) {
  tft.fillScreen(ST77XX_BLACK);

  int indicatorWidth = 128 / 2;
  for (int i = 0; i < 2; i++) {
    tft.fillRect(i * indicatorWidth, 18, indicatorWidth - 2, 3,
                 (i == index) ? weather[index].color : ST77XX_WHITE);
  }
  tft.setTextColor(weather[index].color);
  drawCenteredText(weather[index].city, 2, 1);

  tft.setTextSize(4);
  String tempStr = String(weather[index].temp, 0) + "C";
  int16_t x1, y1;
  uint16_t w, h;
  tft.getTextBounds(tempStr, 0, 0, &x1, &y1, &w, &h);
  tft.setCursor((128 - w) / 2, 25);
  tft.print(tempStr);

  tft.setTextSize(1);
  String condition = weather[index].condition;
  if (condition.length() > 18) {
```

Рисунок 3.12 – Код виведення для відображення поточної погоди у місті

```
  uint16_t w, h;
  tft.getTextBounds(tempStr, 0, 0, &x1, &y1, &w, &h);
  tft.setCursor((128 - w) / 2, 25);
  tft.print(tempStr);

  tft.setTextSize(1);
  String condition = weather[index].condition;
  if (condition.length() > 18) {
    condition = condition.substring(0, 18) + "...";
  }
  drawCenteredText(condition, 60, 1, ST77XX_WHITE);

  tft.drawFastHLine(0, 70, 128, ST77XX_WHITE);

  tft.setTextColor(ST77XX_WHITE);
  tft.setCursor(10, 80);
  tft.print("Feels: ");
  tft.setTextColor(ST77XX_YELLOW);
  tft.print(String(weather[index].feelslike, 1) + "C");

  tft.setTextColor(ST77XX_WHITE);
  tft.setCursor(10, 95);
  tft.print("Hum: ");
  tft.setTextColor(ST77XX_CYAN);
  tft.print(String(weather[index].humidity) + "%");

  tft.setTextColor(ST77XX_WHITE);
  tft.setCursor(10, 110);
  tft.print("Cloud: ");
  tft.setTextColor(ST77XX_GREEN);
  tft.print(String(weather[index].cloud) + "%");
}
```

Рисунок 3.13 – Код виведення для відображення поточної погоди у місті (2)

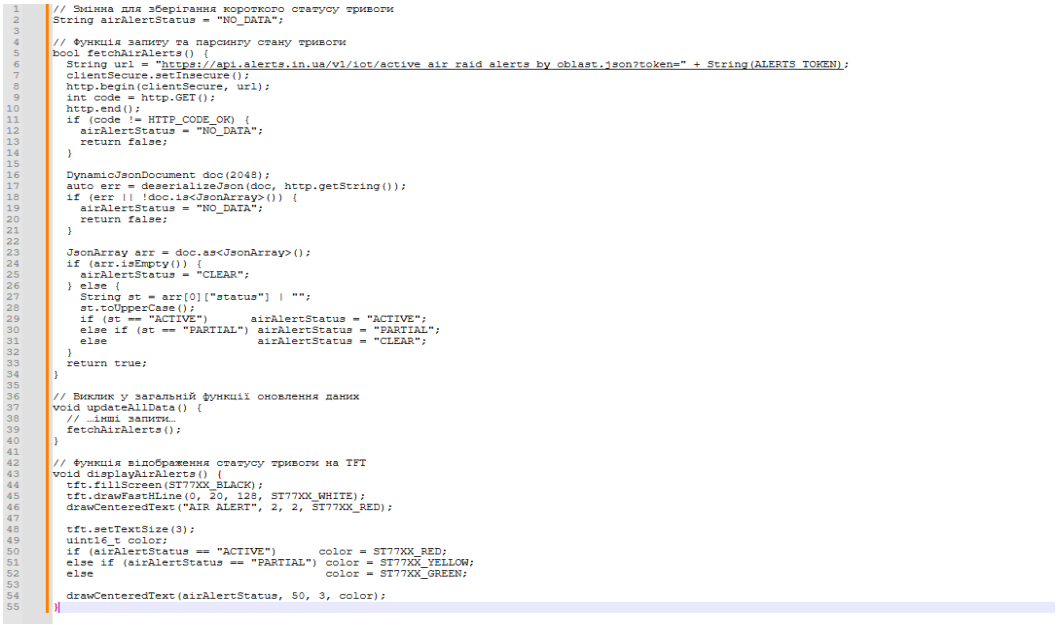
Опис коду на рисунку 3.12 та рисунку 3.13: Отримує поточну погоду через *API WeatherAPI* за координатами для двох міст. Парсить *JSON*, зчитує температуру, вологість, відчувається, хмарність та текстовий опис умови. [15]

Відображає назву міста, великий поточний температурний рядок, короткий опис умови та додаткові параметри (відчувається, вологість, хмарність) на *TFT*-дисплеї з індикаторами вибору міста (рисунок 3.11).

Оновлює дані щохвилини в складі функції *updateAllData*() та автоматично перерисовує екран через *showCurrentPage*().

6. Відображення поточного статусу тривоги у регіоні.

Рисунок 3.14 – Відображення поточного статусу тривоги у регіоні на дісплей



```
// Змінна для зберігання короткого статусу тривоги
String airAlertStatus = "NO_DATA";

// Функція запиту та парсингу стану тривоги
bool fetchAirAlerts() {
  String url = "https://api.alerts.in.ua/v1/iot/active_air_raid_alerts_by_oblast.json?token=" + String(ALERTS_TOKEN);
  clientSecure.setInsecure();
  http.begin(clientSecure, url);
  int code = http.GET();
  http.end();
  if (code != HTTP_CODE_OK) {
    airAlertStatus = "NO_DATA";
    return false;
  }

  DynamicJsonDocument doc(2048);
  auto err = deserializeJson(doc, http.getString());
  if (err || !doc.is<JsonArray>()) {
    airAlertStatus = "NO_DATA";
    return false;
  }

  JsonArray arr = doc.as<JsonArray>();
  if (arr.isEmpty()) {
    airAlertStatus = "CLEAR";
  } else {
    String st = arr[0]["status"] | "";
    st.toUpperCase();
    if (st == "ACTIVE")       airAlertStatus = "ACTIVE";
    else if (st == "PARTIAL") airAlertStatus = "PARTIAL";
    else                      airAlertStatus = "CLEAR";
  }
  return true;
}

// Виклик у загальній функції оновлення даних
void updateAllData() {
  // …інші запити…
  fetchAirAlerts();
}

// Функція відображення статусу тривоги на TFT
void displayAirAlerts() {
  tft.fillScreen(ST77XX_BLACK);
  tft.drawFastHLine(0, 20, 128, ST77XX_WHITE);
  drawCenteredText("AIR ALERT", 2, 2, ST77XX_RED);

  tft.setTextSize(3);
  uint16_t color;
  if (airAlertStatus == "ACTIVE")       color = ST77XX_RED;
  else if (airAlertStatus == "PARTIAL") color = ST77XX_YELLOW;
  else                                  color = ST77XX_GREEN;

  drawCenteredText(airAlertStatus, 50, 3, color);
}
```

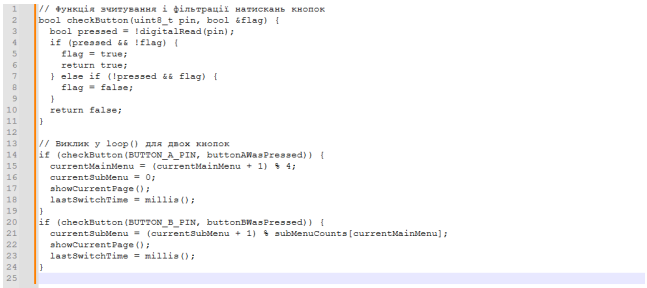
Рисунок 3.15 – Код виведення для поточного статусу тривоги у регіоні

Опис коду на рисунку 3.15: Отримує поточний статус повітряної тривоги через *HTTPS*-запит до *API alerts.in.ua*, парсить *JSON*-масив і зберігає короткий статус у змінну *airAlertStatus* ("*ACTIVE*", "*PARTIAL*" або "*CLEAR*", "*NO_DATA*" при помилці або відсутності даних).

У *updateAllData*() викликає *fetchAirAlerts*() разом з іншими запитами, щоб оновити статус щохвилини.

Функція *displayAirAlerts*() очищує екран *TFT*, малює заголовок і відображає *airAlertStatus* великим шрифтом по центру, підсвічуючи текст червоним (*ACTIVE*), жовтим (*PARTIAL*) або зеленим (див. рисунок 3.14).

7. Реалізація роботи сенсорних кнопок. Сенсорні кнопки використовуються для перемикання між нашими режимами та вкладками.



```
// Функція зчитування і фільтрації натискань кнопок
bool checkButton(uint8_t pin, bool &flag) {
  bool pressed = !digitalRead(pin);
  if (pressed && !flag) {
    flag = true;
    return true;
  } else if (!pressed && flag) {
    flag = false;
  }
  return false;
}

// Виклик у loop() для двох кнопок
if (checkButton(BUTTON_A_PIN, buttonAWasPressed)) {
  currentMainMenu = (currentMainMenu + 1) % 4;
  currentSubMenu = 0;
  showCurrentPage();
  lastSwitchTime = millis();
}
if (checkButton(BUTTON_B_PIN, buttonBWasPressed)) {
  currentSubMenu = (currentSubMenu + 1) % subMenuCounts[currentMainMenu];
  showCurrentPage();
  lastSwitchTime = millis();
}
```

Рисунок 3.16 – Код обробки зчитування сенсорних кнопок

Опис коду на рисунку 3.16: Реалізує зчитування стану двох кнопок з внутрішнім підтягуванням та фільтрацію «дребезгу» контактів через функцію *checkButton*(), яка повертає *true* лише в момент спрацювання.

– Кнопка *A* перемикає головні екрани (крипта/валюта/погода/тривога),
– Кнопка *B* підменю поточного екрану, після чого викликається *showCurrentPage*() для оновлення дисплея.

Виконується на кожній ітерації *loop*(), що забезпечує швидку реакцію на натискання.

3.4 Реалізація веб-інтерфейсу

Веб-інтерфейс забезпечує віддалений доступ до *ESP*8266 через браузер у локальній мережі. Він дозволяє переглядати курси криптовалют, національних валют, погоду та статус повітряної тривоги, а також налаштовувати *Wi-Fi* параметри, не підходячи безпосередньо до пристрою. Він реалізований на базі вбудованого веб-сервера *ESP*8266 і дозволяє передавати дані через локальну мережу *Wi-Fi*. Ключовими особливостями веб-інтерфейсу є доступ через браузер для перегляду курсів криптовалют, не відходячи від пристрою, автоматичне оновлення курсів для забезпечення постійної актуальності даних про *BTC* і *ETH* без перезавантаження сторінки, а веб-сервер *ESP*8266 забезпечує легку інтеграцію для простого веб-доступу без необхідності використання зовнішнього сервера.

Для реалізації веб-інтерфейсу використовуються наступні технології: *ESP*8266*WebServer* - бібліотека для створення веб-сервера на *ESP*8266; *HTML* + *CSS* - забезпечує основний інтерфейс в браузері; *ArduinoJson* – для парсингу *JSON*- даних від *API* (погода, тривога), *JSON* - формат для отримання даних з *API CoinLore*. Завдяки цим технологіям веб-інтерфейс працює швидко, легко інтегрується з *ESP*8266 і не вимагає додаткового серверного обладнання.

Використані технології

– *ESP*8266*WebServer* – бібліотека для створення веб-сервера на *ESP*8266.

– *HTML* + *CSS* – забезпечують базовий інтерфейс у браузері.

– *ArduinoJson* – для парсингу *JSON*-даних від *API* (погода, тривога).

– *JSON* – формат, у якому дані отримуються з *CoinLore API*.

Перед початком роботи необхідно підключити *ESP*8266 до *Wi-Fi* і створити локальний веб-сервер. Підключення до *Wi-Fi* у нашому пристрої здійснюється за допомогою збереженного налаштування у *EEPROM*, нижче наведено принцип за яким працює програмний код.

Пристрій зчитує збережені параметри з *EEPROM* (якщо вони є), приклад коду на рисунку 3.17.

```
#include <EEPROM.h>

struct WifiSettings {
  char staSSID[33];
  char staPass[33];
  char apSSID[33];
  char apPass[33];
} wifiSettings;

void loadSettingsFromEEPROM() {
  EEPROM.begin(512);
  EEPROM.get(0, wifiSettings);
  EEPROM.end();
  // Гарантуємо нуль-термінатор і перевіряємо валідність
  wifiSettings.staSSID[32] = 0;
  wifiSettings.staPass[32] = 0;
  // Якщо apSSID порожній — задаємо "DataBox" за замовчуванням
  if (strlen(wifiSettings.apSSID) == 0) {
    strcpy(wifiSettings.apSSID, "DataBox");
  }
}
```

Рисунок 3.17 – Зчитування збережених параметрів з *EEPROM*

Пробує підключитися до цієї мережі (*STA*), приклад коду на рисунку 3.18

```
#include <ESP8266WiFi.h>

bool attemptWiFiConnection() {
  if (strlen(wifiSettings.staSSID) == 0) return false;
  WiFi.mode(WIFI_STA);
  WiFi.begin(wifiSettings.staSSID, wifiSettings.staPass);

  unsigned long start = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - start < 15000)
    delay(500);
  }
  return (WiFi.status() == WL_CONNECTED);
}
```

Рисунок 3.18 – Підключення до мережі (*STA*)

Якщо підключення збережених даних немає або воно не вдале, переходить у режим *AP* + *WebConfig Portal*, де користувач через браузер вводить свої *SSID*/пароль 192.168.4.1 (рисунок 3.19).

```ino
#include "WebConfig.h"   // містить startWebConfigPortal() з формою

void setup() {
  Serial.begin(115200);
  loadSettingsFromEEPROM();

  // Спроба STA
  if (!attemptWiFiConnection()) {
    // Переходимо у режим AP для конфігурації
    startWebConfigPortal();
  } else {
    // Якщо STA є — піднімаємо власну AP та веб-сервер
    setupAccessPoint();
    startWebServer();
  }
}
```

Рисунок 3.19 – Піднімання *AP + WebConfig*

Після вдалого підключення до мережі *Wi-Fi*, у нашому коді створюється головна сторінка з вкладками. Фрагменти коду представлено на рисунках 3.20 та 3.21.

```ino
void handleRoot() {
  String tab = server.arg("tab");
  if (tab != "wifi" && tab != "crypto" && tab != "currency" && tab != "weather" && tab != "alerts")
    tab = "wifi";

  // HTML + CSS для панелі вкладок
  String html = R"rawliteral(
  <!DOCTYPE html><html><head>
    <meta charset="UTF-8"><meta name="viewport" content="width=device-width,initial-scale=1">
    <title>DataBox</title>
    <style>
      body{font-family:Arial;background:#f0f0f0;margin:0;padding:20px;}
      .tabs{margin-bottom:15px;}
      .tabBtn{display:inline-block;padding:8px 12px;margin-right:6px;
              background:#eee;color:#333;text-decoration:none;border-radius:4px;}
      .tabBtn.active{background:#2196F3;color:#fff;}
      .info-box{background:#fff;padding:12px;border-radius:6px;box-shadow:0 0 5px rgba(0,0,0,0.1);}
    </style>
  </head><body>
    <div class="tabs">
      <a href="/?tab=wifi"    class="TAB_WIFI">Wi-Fi</a>
      <a href="/?tab=crypto"  class="TAB_CRYPTO">Crypto</a>
      <a href="/?tab=currency"class="TAB_CURRENCY">Currency</a>
      <a href="/?tab=weather" class="TAB_WEATHER">Weather</a>
      <a href="/?tab=alerts"  class="TAB_ALERTS">Alerts</a>
    </div>
  )rawliteral";

  auto makeActive = [&](String name){
    return (tab==name) ? "tabBtn active" : "tabBtn";
  };
  html.replace("class=\"TAB_WIFI\"",    "class=\"" + makeActive("wifi") + "\"");
  html.replace("class=\"TAB_CRYPTO\"",  "class=\"" + makeActive("crypto") + "\"");
  html.replace("class=\"TAB_CURRENCY\"", "class=\"" + makeActive("currency") + "\"");
  html.replace("class=\"TAB_WEATHER\"",  "class=\"" + makeActive("weather") + "\"");
  html.replace("class=\"TAB_ALERTS\"",   "class=\"" + makeActive("alerts") + "\"");
```

Рисунок 3.20 – Створення головної сторінки з вкладками

```ino
  // --- вкладка Wi-Fi ---
  if(tab == "wifi"){
    html += R"rawliteral(
    <div class="info-box">
      <h3>Wi-Fi Setup</h3>
      <form action="/save" method="POST">
        <label>STA SSID:</label>
        <input name="sta_ssid" value=")rawliteral" + wifiSettings.staSSID + R"rawliteral(">
        <label>STA PASS:</label>
        <input name="sta_pass" value=")rawliteral" + wifiSettings.staPass + R"rawliteral(">
        <label>AP SSID:</label>
        <input name="ap_ssid"  value=")rawliteral" + wifiSettings.apSSID + R"rawliteral(">
        <label>AP PASS:</label>
        <input name="ap_pass"  value=")rawliteral" + wifiSettings.apPass + R"rawliteral(">
        <button type="submit">Save</button>
      </form>
    </div>
    )rawliteral";

  }
  // --- вкладка Crypto ---
  else if(tab == "crypto"){
    html += "<div class=\"info-box\"><h3>Crypto (USD)</h3>"
          + "<p>BTC: "  + String(cryptos[0].price,2) + " USD</p>"
          + "<p>ETH: "  + String(cryptos[1].price,2) + " USD</p>"
          + "<p>DOGE: " + String(cryptos[2].price,4) + " USD</p>"
          + "</div>";

  }
  // --- вкладка Currency ---
  else if(tab == "currency"){
    html += "<div class=\"info-box\"><h3>Currency (UAH)</h3>"
          + "<p>USD: " + String(currencies[0].rate,4) + " UAH</p>"
          + "<p>EUR: " + String(currencies[1].rate,4) + " UAH</p>"
          + "</div>";

  }
  // --- вкладка Weather ---
  else if(tab == "weather"){
    html += "<div class=\"info-box\"><h3>Weather</h3>"
          + "<p><b>" + weather[0].city + ":</b> "
          + String(weather[0].temp,1) + "°C, " + weather[0].condition + "</p>"
          + "<p><b>" + weather[1].city + ":</b> "
          + String(weather[1].temp,1) + "°C, " + weather[1].condition + "</p>"
          + "</div>";

  }
  // --- вкладка Alerts ---
  else if(tab == "alerts"){
    html += "<div class=\"info-box\"><h3>Air Alerts</h3>"
          + "<p>Status: " + airAlertStatus + "</p>"
          + "</div>";

  }

  html += "</body></html>";
  server.send(200, "text/html", html);
}
```

Рисунок 3.21 – Більш дітальний опис вмісту вкладок

Надалі можно заглянути, як взагалі виглядають ці вкладки у веб-інтерфейси та сам цей інтерфейс.

Вкладка *Wi-Fi* дозволяє налаштовувати параметри мережі без перепрошивки самого пристрою. Тут можна налашувати назву мережі нашого пристрою та *Wi-Fi* мережу до якої він буде підключатися, приклад показано на рисунку 3.22.

Рисунок 3.22 – Вкладка “*Wi-Fi*”

Також, у веб-інтерфейсі є вкладка “*Crypto*”, що показує актуальні ціни трьох криптовалют у доларах США (рисунок 3.23).

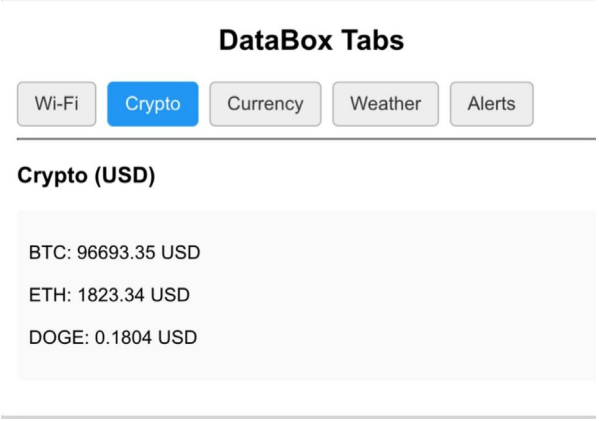


Рисунок 3.23 – Вкладка “*Crypto*”

Вкладка “*Currency*” у веб-інтерфейсі *DataBox* показує актуальні курси долара і євро в гривнях (рисунок 3.24).

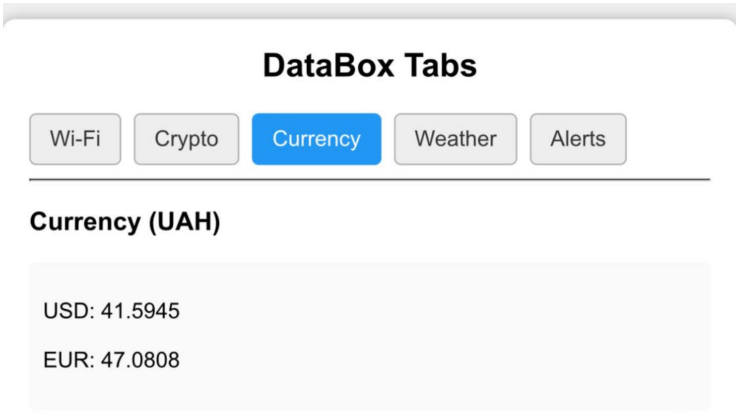


Рисунок 3.24 – Вкладка “*Currency*” з курсами *USD* та *EUR*

Вкладка “*Weather*” у веб-інтерфейсі *DataBox* відображає поточну погоду для двщх поточних міст (рисунок 3.25).

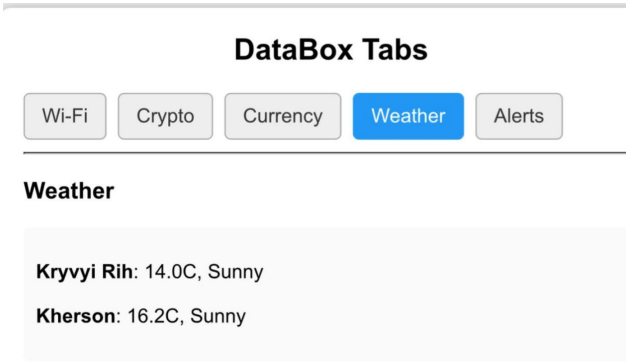


Рисунок 3.25 – Вкладка “*Weather*” з поточною погодою

Вкладка “*Alerts*” у веб-інтерфейсі *DataBox* відображає поточний статус повітряної тривоги у Криворізькому районі. Це дуже актуально з веденням зональних тривог (рисунок 3.26).

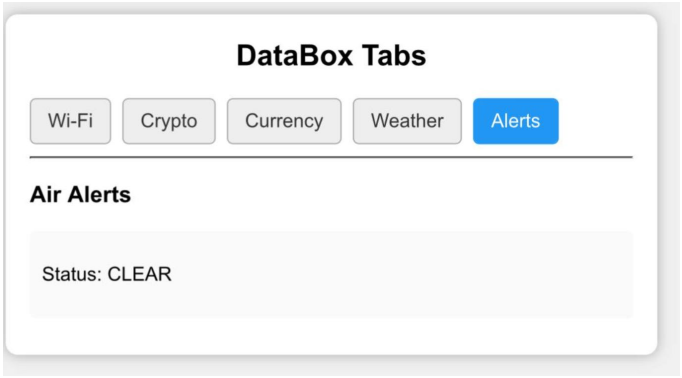


Рисунок 3.26 – Вкладка “*Alerts*” з поточним статусом повітряної тривоги

Алгоритм роботи веб-інтерфейсу:

1. Вхід у браузері

Користувач підключається до *Wi-Fi AP* пристрою (“*DataBox*”) або вводить його *STA-IP* (192.168.4.1) у браузері.

2. Панель вкладок

При зверненні на кореневий маршрут (*GET* /?*tab*=…) відображається сторінка з навігаційними вкладками: *Wi-Fi*, *Crypto*, *Currency*, *Weather*, *Alerts*.

3. Динамічна генерація *HTML*

Параметр *tab* у *URL* передається в *ESP*8266*WebServer*. Функція *handleRoot*() формує відповідний блок:

– Вкладка *Wi-Fi* — форма для введення та збереження *SSID*/*PASS*,

– *Crypto* — курси *BTC*/*ETH*/*DOGE*,

– *Currency* — курси *USD*/*EUR*,

– *Weather* — погода для двох міст,

– *Alerts* — статус повітряної тривоги.

4. Оновлення даних у бекенді

Функція *updateAllData*() опитує *API* усієї інформації (крипта, валюти, погода, тривоги) раз на хвилину. Отримані значення зберігаються в масивах/змінних і підставляються в *HTML* при кожному запиті.

5. Збереження налаштувань *Wi-Fi*

Вкладка *Wi-Fi* надсилає *POST*-запит на /*save*, обробник *handleSave*() зберігає нові *SSID*/*PASS* у *EEPROM* і перезавантажує пристрій для застосування змін.

3.5 Тестування та інструкція користувача

Стабільність з'єднання, коректність відображення інформації та функціональність веб-інтерфейсу. Тестування включало тест підключення до *Wi-Fi*, який перевіряв стабільність мережевого з'єднання і можливість автоматичного перепідключення в разі втрати сигналу. Також було протестовано отримання даних з *API CoinLore*, щоб переконатися, що отримані курси *BTC* і *ETH* відповідають реальним ринковим значенням. Крім того, було протестовано дисплей *ST*7735*S*, щоб переконатися в точності виведення інформації, включаючи правильні числові значення і регулярне оновлення. Завершальним етапом тестування стала оцінка веб-інтерфейсу, перевірка точності відображення курсу в браузері, точності оновлення даних без перезавантаження сторінки, а також можливість віддаленого доступу до інформації через локальну мережу *Wi-Fi*.

Інструкція користувача:

1. Живлення

   Підключіть пристрій до *USB*-джерела 5 В.

2. Початкове підключення до *Wi-Fi*

   При увімкненні *ESP8266* автоматично спробує під’єднатися до збереженої мережі *STA* (повідомлення “*Wi-Fi connected: …” у Serial Monitor*).

3. Режим *AP* та веб-конфігурація

   Якщо *STA*-підключення відсутнє або не налаштоване, *ESP* відкриває точку доступу (*AP SSID: DataBox*). Підключіться до цієї мережі та перейдіть у браузері на http://192.168.4.1.

4. Налаштування *Wi-Fi* у браузері

   На вкладці *Wi-Fi* введіть або перегляньте:

   STA SSID / PASS – домашня мережа,

   AP SSID / PASS – ім’я та пароль власної точки доступу.

   Натисніть *Save Wi-Fi*, дочекайтесь перезавантаження.

5. Перегляд даних

Відкрийте браузер і перейдіть за *IP*-адресою (*STA-IP* або 192.168.4.1) або переглядайте дані на дісплеї. Перейдіть на потрібну вкладку:

Crypto – ціни *BTC/ETH/DOGE*,

Currency – курси *USD/UAH, EUR/UAH*,

Weather – погода в обраних містах,

Alerts – поточний статус повітряної тривоги.

6. Перемикання режимів на *TFT*-дисплеї
   - Кнопка *A* (*D6/GPIO12*) – перемикає між основними вкладками (*Crypto* ↔ *Currency* ↔ *Weather* ↔ *Alerts*),
   - Кнопка *B* (*D1/GPIO5*) – циклічно змінює підрозділи в межах поточної вкладки.

ВИСНОВОК

Кваліфікаційна робота присвячена розробці універсального *IoT*-гаджета *DataBox* на базі *ESP8266*, який здатний у реальному часі збирати та відображати дані з різних джерел: курси криптовалют (*BTC, ETH, DOGE*), курси національних валют (*USD, EUR*), поточну погоду в обраних містах та статус повітряної тривоги. Використання TFT-дисплея *ST7735S* і сенсорних кнопок забезпечило зручний локальний інтерфейс, а вбудований веб-сервер зі сторінками-вкладками дозволяє віддалено переглядати й налаштовувати пристрій без перепрошивки.

Реалізація веб-інтерфейсу базується на бібліотеці ESP8266WebServer і використовує HTML + CSS для побудови адаптивних вкладок (“*Wi-Fi*”, “*Crypto*”, “*Currency*”, “*Weather*”, “*Alerts*”). Конфігурація *Wi-Fi* (*STA + AP*) відбувається через веб-форму, дані зберігаються у *EEPROM*, що гарантує збереження налаштувань після перезавантаження. Для обміну з *API* застосовано *WiFiClientSecure + HTTPClient*, а парсинг *JSON/XML* здійснюється через *ArduinoJson* та простий пошук тегів.

У ході тестування підтверджено стабільність мережевого з’єднання з автоматичним перепідключенням при втраті сигналу, коректність отримання та відображення курсів криптовалют і валютних курсів із *API Coinlore* та НБУ, точність погодних даних із *WeatherAPI* та статусу тривоги з *alerts.in.ua*. *TFT*-дисплей забезпечує своєчасне оновлення екрану, а веб-інтерфейс демонструє актуальні значення у браузері з можливістю віддаленого доступу.

Вибір технологій та їх інтеграція дозволили створити компактне, енергоефективне та гнучке рішення. Що робить його перспективним інструментом для домашнього та промислового моніторингу.

Отже, реалізований *IoT*-гаджет з веб-інтерфейсом і локальним дисплеєм є ефективним засобом для віддаленого моніторингу фінансових і метео-даних, забезпечує простоту управління та налаштування, а також високу надійність і масштабованість рішення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Криптовалюти та блокчейн. *Nakamoto S. Bitcoin: A Peer-to-Peer Electronic Cash System.* [Електронний ресурс]. – Режим доступу до ресурсу: *https://bitcoin.org/bitcoin.pdf.* (дата звернення: 20.05.2025)
2. Веб-сервер на *ESP8266. Web Server with ESP8266 NodeMCU*. [Електронний ресурс]. – Режим доступу до ресурсу: *https://randomnerdtutorials.com/esp8266-web-server/.* (дата звернення: 18.05.2025)
3. Використання *ESP8266* для *IoT*-проєктів. *Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and WiFi Enabled Devices. – Neil Cameron, 2020. –* 62 с.
4. Енергоспоживання *ESP8266. Low Power Modes of ESP8266*. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.espressif.com/en/products/socs/esp8266. (дата звернення: 20.05.2025)
5. Робота з дисплеєм *ST7735S. Adafruit GFX Library Documentation.* [Електронний ресурс]. – Режим доступу до ресурсу: *https://learn.adafruit.com/adafruit-gfx-graphics-library/.* (дата звернення: 16.05.2025)
6. Моніторинг курсів криптовалют. *CoinLore. Cryptocurrency API for Developers.* [Електронний ресурс]. – Режим доступу до ресурсу: *https://www.coinlore.com/cryptocurrency-data-api.* (дата звернення: 21.05.2025)
7. Принципи побудови *IoT*-пристроїв. *Margolis M. Arduino Cookbook, 2nd Edition. – O'Reilly Media,* 2012. – 724 с.

8. Програмування *ESP8266. ESP8266WiFi Library – Arduino Reference*. [Електронний ресурс]. – Режим доступу до ресурсу: *https://www.arduino.cc/en/Reference/WiFi.* (дата звернення: 16.05.2025)

9. *AJAX* та *JSON* для оновлення веб-інтерфейсів. *Duckett J. JavaScript and jQuery: Interactive Front-End Web Development. – Wiley, 2014. –* 640 с.

10. «A NodeMCU Based Low-Cost Portable Device.» *[Електронний ресурс]. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/10462140 (дата звернення: 16.05.2025)*

11. *Adafruit Industries. Adafruit GFX Library. GitHub Repository,* 2025. [Електронний ресурс]. – Режим доступу до ресурсу: *https://github.com/adafruit/Adafruit-GFX-Library* (дата звернення: 16.05.2025)

12. Blanchon, Benoît. ArduinoJson Library. [Електронний ресурс] Режим доступу до ресурсу: *https*://*arduinojson.org*/ (дата звернення: 16.05.2025)

13. *Espressif Systems. ESP8266EX Datasheet V3. Espressif Systems, June 2015.* [Електронний ресурс]. – Режим доступу до ресурсу: *https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf* (дата звернення: 16.05.2025)

14. *Grokhotkov, Ivan. ESP8266 Arduino Core Documentation — EEPROM Library. Espressif via Read the Docs, 2017.* [Електронний ресурс]. – Режим доступу до ресурсу: *https://esp8266.readthedocs.io/en/latest/libraries.html#eeprom* (дата звернення: 16.05.2025)

15. *WeatherAPI.com. WeatherAPI Documentation.* [Електронний ресурс]. – Режим доступу до ресурсу: *https://www.weatherapi.com/docs/* (дата звернення: 16.05.2025)

ДОДАТОК А

Код програми головного модуля

```
#include <Arduino.h>
#include <EEPROM.h>
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <WiFiClientSecure.h>
#include <ESP8266HTTPClient.h>
#include <ArduinoJson.h>
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#include <SPI.h>

// ------------------ API ------------------
#define WEATHER_API_KEY  "0aca65826f304eec8f0141954252401"
#define ALERTS_TOKEN     "b62e143d313dd4dae015732324fa3a6155c5182dab2203"

// ----------------------------------
#define TFT_CS   15
#define TFT_DC    2
#define TFT_RST   0
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);

// ----------------------------------------
#define BUTTON_A_PIN  5
#define BUTTON_B_PIN  12

bool buttonAWasPressed = false;
bool buttonBWasPressed = false;

// ------------------   -----------
unsigned long lastSwitchTime = 0;
const unsigned long AUTO_SWITCH_INTERVAL = 10000; // 10

int currentMainMenu = 0;
int currentSubMenu = 0;
const int subMenuCounts[4] = {3, 2, 2, 1};

bool isConnected = false;

// ------------------------------------

//
struct CryptoData {
  float price;
  const char* symbol;
  uint16_t color;
} cryptos[3] = {
  {0, "BTC",  0x07E0}, // ST77XX_BLUE (i)
  {0, "ETH",  0x07E0},
  {0, "DOGE", 0x07E0}
};

//
struct CurrencyData {
  float rate;
  const char* symbol;
  uint16_t color;
} currencies[2] = {
  {0, "USD", 0x001F}, // BLUE
  {0, "EUR", 0x001F}
```

```
};

//
struct WeatherData {
  float temp;
  int   humidity;
  int   cloud;
  float feelslike;
  String condition;
  const char* city;
  uint16_t color;
} weather[2] = {
  {0, 0, 0, 0, "", "Kryvyi Rih", 0xFFFF}, // WHITE
  {0, 0, 0, 0, "", "Kherson",    0xFFFF}
};

//
String airAlertStatus = "NO_DATA";

// ------------------ WiFiSettings + EEPROM -------------
struct WifiSettings {
  char staSSID[33];
  char staPass[33];
  char apSSID[33];
  char apPass[33];
};

WifiSettings wifiSettings;

// ------------------ HTTP  WebServer ----------
WiFiClientSecure clientSecure;
HTTPClient http;
ESP8266WebServer server(80);

// ========================================================
//           EEPROM
// ========================================================
void loadSettingsFromEEPROM() {
  EEPROM.begin(512);
  EEPROM.get(0, wifiSettings);
  EEPROM.end();

  //
  wifiSettings.staSSID[32] = 0;
  wifiSettings.staPass[32] = 0;
  wifiSettings.apSSID[32]  = 0;
  wifiSettings.apPass[32]  = 0;

  //
  auto fixString = [&](char* s) {
    if (s[0] < 32 || s[0] > 126) {
      memset(s, 0, 33);
    }
  };
  fixString(wifiSettings.staSSID);
  fixString(wifiSettings.staPass);
  fixString(wifiSettings.apSSID);
  fixString(wifiSettings.apPass);

  if (strlen(wifiSettings.apSSID) == 0) {
    strcpy(wifiSettings.apSSID, "DataBox");
  }

  Serial.println("[EEPROM] :");
  Serial.print("  STA SSID: "); Serial.println(wifiSettings.staSSID);
```

```
  Serial.print("  STA PASS: "); Serial.println(wifiSettings.staPass);
  Serial.print("  AP  SSID: "); Serial.println(wifiSettings.apSSID);
  Serial.print("  AP  PASS: "); Serial.println(wifiSettings.apPass);
}

void saveSettingsToEEPROM(const WifiSettings &settings) {
  EEPROM.begin(512);
  EEPROM.put(0, settings);
  EEPROM.commit();
  EEPROM.end();
  Serial.println("[EEPROM]  !");
}

// ======================================================
//              Wi-Fi  (STA+AP)
// ======================================================
bool attemptWiFiConnection() {
  if (strlen(wifiSettings.staSSID) == 0) {
    Serial.println("[WiFi]   SSID  STA.");
    return false;
  }
  Serial.print("[WiFi]  : ");
  Serial.println(wifiSettings.staSSID);

  WiFi.begin(wifiSettings.staSSID, wifiSettings.staPass);

  unsigned long start = millis();
  while (WiFi.status() != WL_CONNECTED && millis() - start < 15000) {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  if (WiFi.status() == WL_CONNECTED) {
    Serial.print("[WiFi] STA Ok! IP:");
    Serial.println(WiFi.localIP());
    return true;
  } else {
    Serial.println("[WiFi]   .");
    return false;
  }
}

void setupAccessPoint() {
  //   , AP
  if (strlen(wifiSettings.apPass) == 0) {
    Serial.print("[WiFi] AP : ");
    Serial.println(wifiSettings.apSSID);
    WiFi.softAP(wifiSettings.apSSID);
  } else {
    Serial.print("[WiFi] AP  : ");
    Serial.println(wifiSettings.apSSID);
    WiFi.softAP(wifiSettings.apSSID, wifiSettings.apPass);
  }
  Serial.print("[WiFi] AP IP: ");
  Serial.println(WiFi.softAPIP());
}

// ======================================================
//           FETCH (CRYPTO, CURRENCY, WEATHER, ALERTS)
// ======================================================

bool fetchCrypto(const String &url, CryptoData &data) {
  Serial.println("[fetchCrypto] " + url);
  clientSecure.setInsecure();
```

```
  http.begin(clientSecure, url);
  int code = http.GET();
  if (code == HTTP_CODE_OK) {
    DynamicJsonDocument doc(256);
    DeserializationError err = deserializeJson(doc, http.getString());
    http.end();
    if (!err) {
      data.price = doc[0]["price_usd"].as<float>();
      Serial.printf("[fetchCrypto] %s=%.2f\n", data.symbol, data.price);
      return true;
    }
  }
  http.end();
  return false;
}

float fetchSingleCurrency(const char* valcode) {
  String
url="https://bank.gov.ua/NBUStatService/v1/statdirectory/exchange?valcode=" +
String(valcode);
  clientSecure.setInsecure();
  http.begin(clientSecure, url);
  int code=http.GET();
  if(code==HTTP_CODE_OK){
    String payload=http.getString();
    http.end();
    int start=payload.indexOf("<rate>");
    int end=payload.indexOf("</rate>");
    if(start>=0 && end>start){
      String rateStr=payload.substring(start+6,end);
      rateStr.trim();
      float rate=rateStr.toFloat();
      Serial.printf("[fetchSingleCurrency] %s=%.3f\n",valcode, rate);
      return rate;
    }
  }
  http.end();
  return 0.0;
}

bool fetchCurrencies() {
  float usd=fetchSingleCurrency("USD");
  float eur=fetchSingleCurrency("EUR");
  bool ok=false;
  if(usd>0){ currencies[0].rate=usd; ok=true; }
  if(eur>0){ currencies[1].rate=eur; ok=true; }
  return ok;
}

bool fetchWeather(int idx) {
  String coords[2]={"47.90,33.38","46.65,32.61"};
  String url="https://api.weatherapi.com/v1/current.json?key=" +
String(WEATHER_API_KEY)
           + "&q=" + coords[idx] + "&lang=en";

  clientSecure.setInsecure();
  http.begin(clientSecure, url);
  int code=http.GET();
  if(code==HTTP_CODE_OK){
    DynamicJsonDocument doc(1024);
    DeserializationError err=deserializeJson(doc,http.getString());
    http.end();
    if(!err){
      auto cur=doc["current"];
      weather[idx].temp=cur["temp_c"];
```

```
      weather[idx].humidity=cur["humidity"];
      weather[idx].cloud=cur["cloud"];
      weather[idx].feelslike=cur["feelslike_c"];
      weather[idx].condition=cur["condition"]["text"].as<String>();
      return true;
    }
  }
  http.end();
  return false;
}

/**
 *   JSON  alerts.in.ua,
 *    status  airAlertStatus:
 *   "ACTIVE", "PARTIAL", "CLEAR"  "NO_DATA"
 */
bool fetchAirAlerts() {
  String
url="https://api.alerts.in.ua/v1/iot/active_air_raid_alerts_by_oblast.json?token
=" + String(ALERTS_TOKEN);
  Serial.println("[fetchAirAlerts] " + url);

  clientSecure.setInsecure();
  http.begin(clientSecure, url);
  int code=http.GET();
  Serial.printf("[fetchAirAlerts] code=%d\n", code);

  if(code==HTTP_CODE_OK){
    String payload=http.getString();
    http.end();

    //  JSON
    DynamicJsonDocument doc(2048);
    auto err = deserializeJson(doc, payload);
    if(err){
      Serial.print("[fetchAirAlerts] parse error: ");
      Serial.println(err.c_str());
      airAlertStatus = "NO_DATA";
      return false;
    }

    if(!doc.is<JsonArray>()){
      //
      airAlertStatus="CLEAR";
      return true;
    }
    JsonArray arr = doc.as<JsonArray>();
    if(arr.size()==0){
      //  =>
      airAlertStatus="CLEAR";
      return true;
    }

    //
    JsonObject obj=arr[0];
    // ,  "status": "ACTIVE"/"PARTIAL"/"CLEAR"
    //   - NO_DATA
    String st = obj["status"] | "";
    st.toUpperCase();

    if(st=="ACTIVE"){
      airAlertStatus="ACTIVE";
    } else if(st=="PARTIAL"){
      airAlertStatus="PARTIAL";
    } else if(st=="CLEAR" || st==""){
```

```
      airAlertStatus="CLEAR";
    } else {
      //
      airAlertStatus=st;
    }

    Serial.print("[fetchAirAlerts] => ");
    Serial.println(airAlertStatus);
    return true;
  }
  else {
    http.end();
    airAlertStatus="NO_DATA";
    return false;
  }
}

// є
void updateAllData() {
  Serial.println("[Main] updateAllData() started.");
  fetchCrypto("https://api.coinlore.net/api/ticker/?id=90", cryptos[0]); // BTC
  fetchCrypto("https://api.coinlore.net/api/ticker/?id=80", cryptos[1]); // ETH
  fetchCrypto("https://api.coinlore.net/api/ticker/?id=2",  cryptos[2]); // DOGE

  fetchCurrencies();
  fetchWeather(0);
  fetchWeather(1);
  fetchAirAlerts();

  Serial.println("[Main] updateAllData() finished.");
}

// ========================================================
//           :
// ========================================================
String formatNumber(float v){
  String s = String(v, (v<1000)?2:0);
  int dp=s.indexOf('.');
  if(dp<0) dp=s.length();
  for(int i=dp-3;i>0;i-=3){
    s=s.substring(0,i)+" "+s.substring(i);
  }
  return s;
}

void drawCenteredText(const String &txt,int y,
  uint8_t size=1, uint16_t color=0xFFFF
){
  tft.setTextSize(size);
  int16_t x1,y1; uint16_t w,h;
  tft.getTextBounds(txt,0,0,&x1,&y1,&w,&h);
  int x=(128-w)/2;
  tft.setCursor(x,y);
  tft.setTextColor(color);
  tft.print(txt);
}

void wifiStatus(const String &msg, uint16_t color=0xFFFF){
  Serial.println("[Main] wifiStatus: "+msg);
  tft.fillScreen(0x0000);
  drawCenteredText(msg,54,1,color);
}

void displayCrypto(int index){
  tft.fillScreen(0x0000); // BLACK
```

```
  int indicatorWidth=128/3;
  for(int i=0;i<3;i++){
    tft.fillRect(i*indicatorWidth,18,indicatorWidth-2,3,
                 (i==index)?cryptos[index].color:0xFFFF);
  }
  tft.setTextColor(cryptos[index].color);
  drawCenteredText(cryptos[index].symbol,2,2);

  String priceStr=formatNumber(cryptos[index].price)+" USD";
  tft.setTextSize(2);
  int16_t x1,y1; uint16_t w,h;
  tft.getTextBounds(priceStr,0,0,&x1,&y1,&w,&h);
  tft.setCursor((128-w)/2,30);
  tft.print(priceStr);
}

void displayCurrency(int index){
  tft.fillScreen(0x0000);
  int w2=128/2;
  for(int i=0;i<2;i++){
    tft.fillRect(i*w2,18,w2-2,3,(i==index)?currencies[index].color:0xFFFF);
  }
  tft.setTextColor(currencies[index].color);
  drawCenteredText(currencies[index].symbol,2,2);

  String s=formatNumber(currencies[index].rate)+" UAH";
  tft.setTextSize(2);
  int16_t x1,y1; uint16_t w,h;
  tft.getTextBounds(s,0,0,&x1,&y1,&w,&h);
  tft.setCursor((128-w)/2,30);
  tft.print(s);
}

void displayWeather(int idx){
  tft.fillScreen(0x0000);
  int w2=128/2;
  for(int i=0;i<2;i++){
    tft.fillRect(i*w2,18,w2-2,3,(i==idx)?weather[idx].color:0xFFFF);
  }
  tft.setTextColor(weather[idx].color);
  drawCenteredText(weather[idx].city,2,1);

  tft.setTextSize(4);
  String tempStr=String(weather[idx].temp,0)+"C";
  int16_t x1,y1; uint16_t w,h;
  tft.getTextBounds(tempStr,0,0,&x1,&y1,&w,&h);
  tft.setCursor((128-w)/2,25);
  tft.print(tempStr);

  tft.setTextSize(1);
  String cond=weather[idx].condition;
  if(cond.length()>18){
    cond=cond.substring(0,18)+"...";
  }
  drawCenteredText(cond,60,1,0xFFFF);

  tft.drawFastHLine(0,70,128,0xFFFF);
  tft.setTextColor(0xFFFF);
  tft.setCursor(10,80);
  tft.print("Feels:");
  tft.setTextColor(0xFFE0); // YELLOW
  tft.print(String(weather[idx].feelslike,1)+"C");

  tft.setTextColor(0xFFFF);
  tft.setCursor(10,95);
```

```
  tft.print("Hum:");
  tft.setTextColor(0x07FF); // CYAN
  tft.print(String(weather[idx].humidity)+"%");

  tft.setTextColor(0xFFFF);
  tft.setCursor(10,110);
  tft.print("Cloud:");
  tft.setTextColor(0x07E0); // GREEN
  tft.print(String(weather[idx].cloud)+"%");
}

void displayAirAlerts(){
  tft.fillScreen(0x0000);
  tft.drawFastHLine(0,20,128,0xFFFF);
  drawCenteredText("AIR ALERT",2,2,0xF800); // RED

  // airAlertStatus: "ACTIVE","PARTIAL","CLEAR","NO_DATA"
  if(airAlertStatus=="ACTIVE"){
    tft.setTextSize(3);
    int16_t x1,y1; uint16_t w,h;
    String st="ACTIVE";
    tft.getTextBounds(st,0,0,&x1,&y1,&w,&h);
    tft.setCursor((128-w)/2,50);
    tft.setTextColor(0xF800); // RED
    tft.print(st);
  } else if(airAlertStatus=="PARTIAL"){
    tft.setTextSize(3);
    String st="PARTIAL";
    int16_t x1,y1; uint16_t w,h;
    tft.getTextBounds(st,0,0,&x1,&y1,&w,&h);
    tft.setCursor((128-w)/2,50);
    tft.setTextColor(0xFFE0); // YELLOW
    tft.print(st);
  } else if(airAlertStatus=="CLEAR"){
    tft.setTextSize(3);
    String st="CLEAR";
    int16_t x1,y1; uint16_t w,h;
    tft.getTextBounds(st,0,0,&x1,&y1,&w,&h);
    tft.setCursor((128-w)/2,50);
    tft.setTextColor(0x07E0); // GREEN
    tft.print(st);
  } else {
    // NO_DATA or other
    tft.setTextColor(0xFFE0);
    drawCenteredText("NO DATA",60,1);
  }
}

void showCurrentPage(){
  switch(currentMainMenu){
    case 0: displayCrypto(currentSubMenu);   break;
    case 1: displayCurrency(currentSubMenu); break;
    case 2: displayWeather(currentSubMenu);  break;
    case 3: displayAirAlerts();              break;
  }
  //   (4 )
  for(int i=0;i<4;i++){
    tft.fillRect(i*32,120,30,8,(currentMainMenu==i)?0x07E0:0xFFFF);
  }
}

// ======================================================
//     :    ()
// ======================================================
bool checkButton(uint8_t pin, bool &flag) {
```

```
  bool pressed=!digitalRead(pin);
  if(pressed && !flag){
    flag=true;
    return true;
  }
  if(!pressed && flag){
    flag=false;
  }
  return false;
}

// =======================================================
//   -:
// =======================================================
/**
 * handleRoot() => /?tab=wifi|crypto|currency|weather|alerts
 */
void handleRoot() {
  String tab = server.arg("tab");
  if(tab=="") tab="wifi"; //   - WiFi

  //  HTML
  String html = R"(
  <!DOCTYPE html>
  <html><head>
    <meta charset="UTF-8"/>
    <title>DataBox Web</title>
    <style>
      body { font-family:Arial; background:#f0f0f0; margin:0; padding:20px; }
      .container { max-width:500px; margin:auto; background:#fff; padding:20px;
                   border-radius:10px; box-shadow:0 0 10px rgba(0,0,0,0.3);}
      h2 { text-align:center; margin-top:0; }
      .tabs { margin-bottom:10px; }
      .tabBtn {
        display:inline-block; padding:8px 14px; margin-right:5px; border-
radius:5px;
        background:#eee; color:#333; text-decoration:none; border:1px solid
#aaa;
      }
      .tabBtn.active { background:#2196F3; color:#fff; border-color:#2196F3; }
      label{ font-weight:bold; }
      input, button { width:100%; padding:10px; margin:5px 0; box-sizing:border-
box; }
      button {
        background:#2196F3; color:#fff; border:none; border-radius:5px;
        font-size:16px; cursor:pointer;
      }
      .info-box { background:#fafafa; padding:10px; margin:10px 0; border-
radius:5px; }
      .info-title { font-weight:bold; margin-bottom:5px; }
    </style>
  </head>
  <body>
    <div class="container">
      <h2>DataBox Tabs</h2>
      <div class="tabs">
        <a href="/?tab=wifi"    TABWIFI>Wi-Fi</a>
        <a href="/?tab=crypto"  TABCRYPTO>Crypto</a>
        <a href="/?tab=currency"TABCURR>Currency</a>
        <a href="/?tab=weather" TABWEA>Weather</a>
        <a href="/?tab=alerts"  TABALR>Alerts</a>
      </div>
  )";

  //  placeholders  class="tabBtn ..." + "active"
```

```
auto setActive = [&](const String &what){
  if(tab==what) return "class='tabBtn active'";
  return "class='tabBtn'";
};

html.replace("TABWIFI",   setActive("wifi"));
html.replace("TABCRYPTO", setActive("crypto"));
html.replace("TABCURR",   setActive("currency"));
html.replace("TABWEA",    setActive("weather"));
html.replace("TABALR",    setActive("alerts"));

//      tab
if(tab=="wifi"){
  //  WiFi
  html += R"(
    <hr/><h3>Wi-Fi Setup</h3>
    <form action="/save" method="POST">
      <label>STA SSID:</label>
      <input type="text" name="sta_ssid" value=")";
  html += wifiSettings.staSSID;
  html += R"(">
      <label>STA PASS:</label>
      <input type="text" name="sta_pass" value=")";
  html += wifiSettings.staPass;
  html += R"(">
      <label>AP SSID:</label>
      <input type="text" name="ap_ssid" value=")";
  html += wifiSettings.apSSID;
  html += R"(">
      <label>AP PASS (opt):</label>
      <input type="text" name="ap_pass" value=")";
  html += wifiSettings.apPass;
  html += R"(">
      <button type="submit">Save Wi-Fi</button>
    </form>
  )";
}
else if(tab=="crypto"){
  // Crypto
  html += R"(
    <hr/><h3>Crypto (USD)</h3>
    <div class="info-box">
      <p>BTC: )";
  html += String(cryptos[0].price,2);
  html += R"( USD</p>
      <p>ETH: )";
  html += String(cryptos[1].price,2);
  html += R"( USD</p>
      <p>DOGE: )";
  html += String(cryptos[2].price,4);
  html += R"( USD</p>
    </div>
  )";
}
else if(tab=="currency"){
  // Currency
  html += R"(
    <hr/><h3>Currency (UAH)</h3>
    <div class="info-box">
      <p>USD: )";
  html += String(currencies[0].rate,4);
  html += "</p><p>EUR: ";
  html += String(currencies[1].rate,4);
  html += R"(</p>
    </div>
```

```
    )";
  }
  else if(tab=="weather"){
    // Weather
    html += R"(
      <hr/><h3>Weather</h3>
      <div class="info-box">
        <p><b>)";
    html += weather[0].city;
    html += "</b>: ";
    html += String(weather[0].temp,1);
    html += "C, ";
    html += weather[0].condition;
    html += R"(</p>
        <p><b>)";
    html += weather[1].city;
    html += "</b>: ";
    html += String(weather[1].temp,1);
    html += "C, ";
    html += weather[1].condition;
    html += R"(</p>
      </div>
    )";
  }
  else if(tab=="alerts"){
    // Alerts
    html += R"(
      <hr/><h3>Air Alerts</h3>
      <div class="info-box">
        <p>Status: )";
    html += airAlertStatus;
    html += R"(</p>
      </div>
    )";
  }
  else {
    //  tab
    html += "<hr/><p>Unknown tab!</p>";
  }

  //  HTML
  html += R"(
    </div> <!-- container -->
  </body>
  </html>
  )";

  server.send(200, "text/html", html);
}

/**
 * handleSave() -  Wi-Fi
 */
void handleSave() {
  String sta_ssid=server.arg("sta_ssid");
  String sta_pass=server.arg("sta_pass");
  String ap_ssid=server.arg("ap_ssid");
  String ap_pass=server.arg("ap_pass");

  sta_ssid.trim(); sta_pass.trim();
  ap_ssid.trim();  ap_pass.trim();

  memset(&wifiSettings,0,sizeof(wifiSettings));
  sta_ssid.toCharArray(wifiSettings.staSSID,33);
  sta_pass.toCharArray(wifiSettings.staPass,33);
```

```
  ap_ssid.toCharArray(wifiSettings.apSSID,33);
  ap_pass.toCharArray(wifiSettings.apPass,33);

  saveSettingsToEEPROM(wifiSettings);

  String resp=R"(
    <!DOCTYPE html>
    <html><head><meta http-equiv="refresh" content="3;url=/?tab=wifi"/>
    <style>body{text-align:center;font-family:Arial;}</style></head>
    <body><h2>Settings saved</h2><p>Rebooting...</p></body>
    </html>
  )";
  server.send(200, "text/html", resp);

  delay(2000);
  ESP.restart();
}

//  -
void startWebServer() {
  server.on("/",HTTP_GET,handleRoot);
  server.on("/save",HTTP_POST,handleSave);
  server.begin();
  Serial.println("[WebServer] ");
}

// ========================================================
// SETUP & LOOP
// ========================================================
void showSplashScreen(){
  Serial.println("[Main] Splash...");
  tft.fillScreen(0x0000);
  tft.setTextColor(0xFFFF);
  drawCenteredText("DataBox",40,3);
  delay(1500);
}

void setup() {
  Serial.begin(115200);
  Serial.println("\n[Setup] Starting...");

  tft.initR(INITR_144GREENTAB);
  tft.setRotation(4);
  tft.fillScreen(0x0000);
  tft.setTextWrap(false);

  pinMode(BUTTON_A_PIN,INPUT_PULLUP);
  pinMode(BUTTON_B_PIN,INPUT_PULLUP);

  showSplashScreen();

  loadSettingsFromEEPROM();
  WiFi.mode(WIFI_AP_STA);

  wifiStatus("Connecting STA...",0xFFE0);
  isConnected=attemptWiFiConnection();
  if(isConnected){
    wifiStatus("STA OK!",0x07E0);
    delay(1000);
  } else {
    wifiStatus("STA Fail",0xF800);
    delay(2000);
  }

  setupAccessPoint();
```

```
  startWebServer();

  if(isConnected){
    updateAllData();
    showCurrentPage();
  }

  Serial.println("[Setup] Finished.");
}

void loop() {
  server.handleClient();

  if(!isConnected){
    delay(100);
    return;
  }

  // :
  if(checkButton(BUTTON_A_PIN,buttonAWasPressed)){
    currentMainMenu=(currentMainMenu+1)%4;
    currentSubMenu=0;
    showCurrentPage();
    lastSwitchTime=millis();
  }
  if(checkButton(BUTTON_B_PIN,buttonBWasPressed)){
    currentSubMenu=(currentSubMenu+1)%subMenuCounts[currentMainMenu];
    showCurrentPage();
    lastSwitchTime=millis();
  }

  //
  if(millis()-lastSwitchTime> AUTO_SWITCH_INTERVAL){
    currentSubMenu=(currentSubMenu+1)%subMenuCounts[currentMainMenu];
    showCurrentPage();
    lastSwitchTime=millis();
  }

  //
  static unsigned long lastUpdate=0;
  if(millis()-lastUpdate>60000){
    Serial.println("[Loop] 1 minute => updateAllData");
    updateAllData();
    showCurrentPage();
    lastUpdate=millis();
  }

  delay(50)
```

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Руслан ЗАЙДУЛІН
(ім'я, прізвище)

Обрана тема кваліфікаційної роботи «Система моніторингу курсу криптовалют BTC і ETH на базі NodeMCU ESP8266» є надзвичайно актуальною в умовах стрімкого розвитку ринку криптовалют, зростання попиту на персональні IoT-рішення та необхідності швидкого доступу до інформації в реальному часі. Реалізація моніторингу фінансових даних через компактний пристрій без залучення зовнішніх серверів відповідає сучасним тенденціям автономності та безпеки в побутових і професійних системах.

Кваліфікаційна робота повністю відповідає заявленій темі, затвердженій відповідним наказом. Всі поставлені завдання були виконані у повному обсязі, зокрема: розробка апаратної частини на базі ESP8266, реалізація відображення інформації на TFT-дисплеї, створення вбудованого веб-інтерфейсу та забезпечення гнучкого налаштування параметрів без перепрошивки.

В результаті виконання кваліфікаційної роботи створено функціональний IoT-гаджет DataBox, здатний відображати курси криптовалют, національні валюти, погоду та повітряну тривогу. Робота демонструє вміння здобувача освіти поєднувати апаратну розробку з програмуванням, обробкою API-запитів та розгортанням інтерактивного інтерфейсу користувача.

Якість оформлення пояснювальної записки та графічного матеріалу відповідає вимогам ДСТУ. Присутній логічний виклад матеріалу, а також послідовна структура подання результатів. У роботі наведено фрагменти практичної реалізації, що підтверджує експериментальний підхід до виконання завдань.

Результати роботи можуть бути використані як основа для розробки персональних індикаторів стану фінансів, погодних умов або як елемент розумного дому. Проєкт має прикладне значення і демонструє навички роботи з сучасними мікроконтролерами та веб-технологіями.

Кваліфікаційна робота виконана на належному рівні, засвідчує достатній рівень підготовки здобувача освіти.

Робота заслуговує на оцінку "відмінно".

Рецензент _____ *викладач* _____
(науковий ступінь, посада)

«30» 05 2025 р. _____ Сергій РУДИЙ
(підпис) (ім'я, прізвище)

З рецензією ознайомлений _____ Руслан ЗАЙДУЛІН
(підпис) (ім'я, прізвище)

КРИВОРІЗЬКИЙ ФАХОВИЙ КОЛЕДЖ
ДЕРЖАВНОГО НЕКОМЕРЦІЙНОГО ПІДПРИЄМСТВА
«ДЕРЖАВНИЙ УНІВЕРСИТЕТ «КИЇВСЬКИЙ АВІАЦІЙНИЙ ІНСТИТУТ»

ВІДГУК
керівника кваліфікаційної роботи

випускника спеціальності: 123 «Комп'ютерна інженерія»

відділення: комп'ютерної та програмної інженерії

циклова комісія: комп'ютерних систем та мереж

Руслан ЗАЙДУЛІН
(ім'я, прізвище)

1. *Кваліфікаційна робота на тему «Система моніторингу курсу криптовалют BTC і ETH на базі NodeMCU ESP8266» виконана здобувачем самостійно за ініціативою.*
2. *Метою роботи є проєктування та реалізація пристрою та веб-інтерфейсу для віддаленого моніторингу актуальних курсів криптовалют BTC і ETH.*
3. *Кваліфікаційна робота відповідає темі, затвердженій наказом керівництва навчального закладу.*
4. *Виконання проєкту здійснене здобувачем освіти самостійно з дотриманням вимог інженерії та програмування для вбудованих систем.*
5. *Руслан ЗАЙДУЛІН показав високий рівень володіння літературними джерелами, аналізу API Coinlore та інших сервісів, прийняття обґрунтованих технічних рішень та застосування сучасних IoT-технологій (ESP8266, ArduinoJson, Adafruit_ST7735, ESP8266WebServer).*
6. *Рівень виконаної кваліфікаційної роботи заслуговує оцінку «відмінно», відповідає набутим знанням, умінням і навичкам та дозволяє можливість присвоєння йому кваліфікації фахівця освітньо-кваліфікаційного ступеню «Фаховий молодший бакалавр» спеціальності 123 «Комп'ютерна інженерія».*

Керівник кваліфікаційної роботи

«13» 06 2025 р. _____ Тетяна РУБАН
(підпис) (ім'я, прізвище)