Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
Програмне забезпечення системи інтелектуальної синхронізації IoT пристроїв з даними приватної Telegram групи

КбПЗ_2026

Виконав здобувач вищої освіти
IV курсу, групи _КН-22_____
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____Гончарук О.С.
«____» _____2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Дрєєв О.М.
«____» _____2026 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *бакалавр*
Галузь знань ._12 _____“Інформаційні технології”___
Спеціальність 122 “Комп’ютерні науки”
Освітньо-професійна (освітньо-наукова) програма _“Комп’ютерні науки”_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____Олексій СМІРНОВ
«____»_____2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Гончаруку Олександру Святославовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи інтелектуальної синхронізації IoT пристроїв з даними приватної Telegram групи*
2. Керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «06» лютого 2026 року №116-02
3. Строк подання роботи до захисту *08.06.2026 р.*
4. Мета та завдання випускної кваліфікаційної роботи. *Метою роботи є розробка алгоритму та програмної реалізації системи синхронізації IoT-пристроїв на базі ESP32 з даними приватної Telegram групи в якості Telegram-бота.*
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. *Призначення та область використання.*
2. Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проєктних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки.
6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Структурна схема системи *1 аркуш*
Функціональна схема системи *1 аркуш*
Діаграма процесів *1 аркуш*
Блок-схема алгоритму роботи системи *2 аркуші*
7. Дата видачі завдання «_06_» лютого 2026р.

КЦТЗ-2025

АНОТАЦІЯ

Гончарук О.С. Програмне забезпечення системи інтелектуальної синхронізації IoT пристроїв з даними приватної Telegram групи.

122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

Кваліфікаційна бакалаврська робота присвячена розробці алгоритму та програмної системи для синхронізації IoT-пристроїв на базі ESP32 з даними приватної Telegram-групи за допомогою Telegram-бота. У роботі реалізовано передавання індивідуальних і групових повідомлень від куратора студентам, а також функції перегляду списку груп, складу студентів і визначення типу навчального тижня.

Розроблено алгоритм обміну даними між Telegram-ботом та IoT-пристроями, що поєднує пряму передачу повідомлень через HTTP-запити та механізм відкладеної доставки із використанням черги повідомлень і періодичного опитування сервера. Визначено структуру повідомлень і забезпечено надійність їх доставки в умовах нестабільного мережевого з'єднання.

Досліджено можливості мікроконтролера ESP32, його мережеві інтерфейси та способи відображення інформації на дисплеї. Реалізовано режими активної роботи та енергозбереження із використанням механізму light sleep.

Програмне забезпечення реалізовано мовами Python і C++ з використанням середовища Visual Studio Code та платформи PlatformIO, що забезпечує зручність розробки, масштабованість і гнучкість конфігурації.

Розроблена система може бути використана як основа для створення IoT-рішень у сфері освіти, систем оповіщення та платформ для обміну даними між віддаленими пристроями.

Ключові слова: IoT, ESP32, Telegram Bot API, синхронізація даних, Wi-Fi, Python, C++, PlatformIO, алгоритм обміну.

КОПЗ_2026

ABSTRACT

Goncharuk O.S. Software for the intelligent synchronization of IoT devices with data from a private Telegram group.

122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

The qualification bachelor's thesis is devoted to the development of an algorithm and software system for synchronizing IoT devices based on ESP32 with data from a private Telegram group using a Telegram bot. The work implements the transmission of individual and group messages from the curator to students, as well as the functions of viewing the list of groups, the composition of students, and determining the type of academic week.

An algorithm for data exchange between a Telegram bot and IoT devices has been developed, combining direct message transmission via HTTP requests and a delayed delivery mechanism using a message queue and periodic server polling. The structure of messages has been determined and their delivery reliability has been ensured in conditions of unstable network connection.

The capabilities of the ESP32 microcontroller, its network interfaces, and methods of displaying information on the display were investigated. Active operation and energy saving modes were implemented using the light sleep mechanism.

The software is implemented in Python and C++ using the Visual Studio Code environment and the PlatformIO platform, which provides ease of development, scalability and flexible configuration.

The developed system can be used as a basis for creating IoT solutions in the field of education, alert systems and platforms for data exchange between remote devices.

Keywords: IoT, ESP32, Telegram Bot API, data synchronization, Wi-Fi, Python, C++, PlatformIO, communication algorithm.

КДТЗ_2026

ЗМІСТ

КОПЗ – 2026

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БР - Бакалаврська робота

ПЗ - Програмне забезпечення

ESP32 - мікроконтролер сімейства Espressif Systems

IoT (Internet of Things) - інтернет речей

Wi-Fi (Wireless Fidelity) - бездротова мережа

HTTP (HyperText Transfer Protocol) - протокол передачі гіпертексту

HTTPS (HyperText Transfer Protocol Secure) - захищений протокол HTTP

API (Application Programming Interface) - програмний інтерфейс додатків

NTP (Network Time Protocol) - протокол синхронізації часу

JSON (JavaScript Object Notation) - формат обміну даними

USB (Universal Serial Bus) - універсальна послідовна шина

GPIO (General Purpose Input/Output) - універсальні входи/виходи

SPI (Serial Peripheral Interface) - послідовний периферійний інтерфейс

I2C (Inter-Integrated Circuit) - двопровідний інтерфейс зв’язку

TLS/SSL (Transport Layer Security / Secure Sockets Layer) - протоколи захищеного з’єднання

OTA (Over-The-Air) - оновлення прошивки “по повітрю”

VS Code (Visual Studio Code) - середовище розробки

FSM (Finite State Machine) - скінченний автомат

IP (Internet Protocol) - інтернет-протокол (IP-адреса)

GET (HTTP GET) - метод запиту для отримання даних

POST (HTTP POST) - метод запиту для надсилання даних

UTF-8 (Unicode Transformation Format-8) - формат кодування символів

SSID (Service Set Identifier) - ім’я Wi-Fi мережі

BLE (Bluetooth Low Energy) - енергоефективний Bluetooth

КПІЗ_2026

ВСТУП

Актуальність теми. Розвиток IoT-технологій створює нові можливості для автоматизації процесів обміну інформацією та побудови розподілених систем керування пристроями в реальному часі. Одним із найбільш доступних і поширених рішень для організації дистанційної взаємодії є використання мікроконтролера ESP32 у поєднанні з веб-сервісами та месенджерами. Завдяки вбудованому Wi-Fi модулю та достатній обчислювальній потужності ESP32 дозволяє реалізовувати компактні IoT-пристрої з можливістю мережевої взаємодії.

Особливого значення набуває використання Telegram як платформи для керування та передачі повідомлень, оскільки Telegram Bot API забезпечує просту інтеграцію, високу швидкість доставки повідомлень та кросплатформеність. Поєднання Telegram-бота з ESP32 дозволяє створювати системи оперативного інформування, де повідомлення можуть передаватися безпосередньо на фізичні пристрої користувачів.

У навчальних та організаційних середовищах існує потреба у швидкій та зручній комунікації між викладачем і студентами. Використання IoT-пристроїв на базі ESP32 із TFT-дисплеєм дозволяє створити локальний інформаційний термінал, який приймає повідомлення через Wi-Fi та одразу відображає їх користувачу. Це підвищує оперативність обміну інформацією та зменшує залежність від мобільних пристроїв.

Додатково актуальним є питання енергоефективності IoT-рішень. Використання режиму light sleep у ESP32 зменшує енергоспоживання та підвищує автономність пристрою. Також важливим є застосування синхронізації часу через NTP-сервер, що забезпечує коректне відображення дати й часу навіть після перезавантажень або втрати живлення.

Мета і завдання дослідження. Метою роботи є розробка клієнт-серверної IoT-системи на базі ESP32 та Telegram-бота, яка забезпечує прийом, обробку та відображення повідомлень у локальній мережі з використанням HTTP-запитів.

КОПЗ 2026

Для досягнення поставленої мети необхідно проаналізувати можливості ESP32 як IoT-платформи, розглянути принципи роботи Telegram Bot API та розробити алгоритм обміну даними між Telegram-ботом і ESP32 із використанням HTTP-запитів. Також передбачається реалізація клієнтської частини системи на ESP32 з підтримкою TFT-дисплея, забезпечення обробки та відображення текстових повідомлень у реальному часі, впровадження синхронізації часу через NTP-сервер, а також реалізація режиму енергозбереження для підвищення автономності пристрою. Завершальним етапом є тестування стабільності роботи системи в умовах локальної мережі.

У межах роботи реалізовано систему, в якій Telegram-бот виступає інтерфейсом керування, а ESP32 - виконавчим пристроєм, що приймає HTTP POST-запити, обробляє отримані дані та відображає їх на TFT-дисплеї. Передача даних здійснюється у текстовому форматі, що забезпечує простоту та швидкість обміну інформацією. Такий підхід дозволяє організувати зручну та ефективну взаємодію між користувачем і апаратною частиною системи в режимі реального часу.

Крім основного функціоналу, система підтримує відображення поточного часу та дати, які синхронізуються через NTP-сервер. Передбачено обробку мережевих помилок, автоматичне повторне підключення до Wi-Fi та відновлення роботи після втрати зв’язку, що підвищує загальну надійність системи.

Практична цінність роботи полягає у можливості використання розробленої системи як навчального або інформаційного IoT-рішення для оперативного інформування користувачів. Система дозволяє передавати повідомлення віддалено через Telegram-бота та миттєво відображати їх на фізичному пристрої, що робить її придатною для використання в навчальних закладах та організаційних структурах.

Розроблена система є масштабованою, базується на стандартних мережевих технологіях та може бути розширена шляхом додавання нових пристроїв, сенсорів або функцій керування, без суттєвих змін у програмній архітектурі.

КБПЗ_2026

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Проєкт пов’язаний із організацією зв’язку між мікроконтролером ESP32 та месенджером Telegram. Основна задача - передавати повідомлення від адміністратора або куратора до користувачів. Як правило, це студенти, але в цілому це не принципово.

Ідея системи є досить простою та зрозумілою. Повідомлення, які надсилаються в Telegram-бот, передаються на окремий пристрій з ESP32 та TFT-дисплеєм. У результаті користувач може переглядати отриманий текст безпосередньо на екрані пристрою. Це може бути зручним у випадках, коли телефон знаходиться не поруч, сповіщення вимкнені або повідомлення було пропущене.

Передача даних реалізована стандартним способом - через HTTP-запити між Telegram-ботом та ESP32. У системі використовується polling-підхід, тобто мікроконтролер через певні проміжки часу самостійно перевіряє наявність нових повідомлень. Такий спосіб реалізації є достатньо простим, не потребує складної конфігурації та добре підходить для подібних систем обміну текстовими даними. Для поставленої задачі цього цілком достатньо.

Окрема серверна частина при цьому не розгортається, що дозволяє спростити загальну структуру системи та зменшити кількість необхідних ресурсів. Хоча такий підхід має певні обмеження, у більшості практичних випадків це не створює суттєвих проблем і забезпечує стабільну роботу системи.

Якщо з’єднання з мережею тимчасово зникає, повідомлення не втрачаються, а надходять із певною затримкою після відновлення підключення. ESP32 працює як клієнт: надсилає HTTP-запит, отримує відповідь від сервера та далі обробляє отримані дані відповідно до логіки роботи системи. Після цього текстова інформація виводиться на TFT-дисплей для подальшого перегляду користувачем. Такий підхід забезпечує стабільність обміну даними та просту

КПІЗ_2026

взаємодію між програмною і апаратною частинами системи.

Окремо використовується синхронізація часу через NTP-сервери. Це необхідно для коректного відображення поточного часу на пристрої та підтримання актуальності часових даних навіть після перезавантаження системи. Реалізація цього механізму є досить простою та не потребує значних обчислювальних або апаратних ресурсів. У цілому систему можна доповнювати новими функціями, модулями або пристроями без суттєвої зміни базового принципу її роботи та загальної архітектури.

1.2 Область застосування

Таке рішення можна використовувати в різних ситуаціях. У першу чергу - там, де потрібно швидко передавати інформацію. Наприклад, у навчальних закладах. Пристрій може бути модифіковано для показу розкладу або зміни. Це звичайна задача, але в такому вигляді вона вирішується трохи зручніше. Інформація дублюється, і шанс її пропустити менший.

На підприємствах ситуація схожа. Система підходить для повідомлень про зміну графіків або якихось внутрішніх оголошень. При цьому вона не залежить від особистих телефонів працівників. Ще один варіант - моніторинг. Якщо відбувається якась подія або спрацьовує датчик, повідомлення передається відповідальній особі. Тут важливо, щоб це відбувалося швидко.

Крім практичного використання, проєкт можна розглядати як навчальний. На ньому зручно пояснювати, як працюють IoT-системи, як відбувається обмін даними і як виглядає клієнт-серверна модель на практиці. Також він дозволяє демонструвати принципи мережевої взаємодії та обробки запитів.

У підсумку можна сказати, що система не прив’язана до конкретної сфери використання. Вона є достатньо простою за своєю структурою та принципом роботи, і саме завдяки цьому може застосовуватись у різних умовах та для різних задач. Систему можна адаптувати як для особистого використання, так і для навчальних, інформаційних або допоміжних проєктів. Крім цього, її архітектура дозволяє без значних труднощів додавати нові функції.

КОПЗ_2026

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Перед тим як починати розробку системи, потрібно було визначити, які саме засоби комунікації сьогодні найчастіше використовуються для обміну інформацією між людьми. У навчальному середовищі це переважно різноманітні месенджери, оскільки вони забезпечують швидку передачу повідомлень, простий доступ із мобільних пристроїв та зручність у повсякденному використанні. Електронна пошта в таких випадках використовується значно рідше, оскільки вона більше підходить для офіційного листування або передачі документів, а не для оперативного спілкування. Внутрішні системи навчальних закладів також існують, проте вони не завжди є достатньо зручними, швидкими або доступними для користувачів. Саме тому під час вибору основи для реалізації системи було логічно орієнтуватися на популярні месенджери, які вже активно використовуються більшістю користувачів у повсякденному житті.

Найчастіше використовується саме Telegram. З ним усе доволі просто - достатньо встановити застосунок, після чого можна одразу починати роботу та обмін повідомленнями. Інтерфейс є зрозумілим для більшості користувачів, а сам месенджер підтримується на різних платформах, що також робить його зручним у використанні. Однак основна причина, чому Telegram є цікавим саме для цієї роботи - це наявність ботів [1]. За їх допомогою можна автоматизувати передачу повідомлень, обробляти команди користувача або навіть використовувати бот як проміжну ланку між людиною та апаратним пристроєм, що значно розширює можливості інтеграції системи.

Фактично бот може виконувати роль окремого програмного інтерфейсу, через який здійснюється взаємодія із системою. Це дозволяє передавати текстові дані, отримувати відповіді від пристрою та реалізовувати різні сценарії

КПЗ_2026

керування без необхідності створення окремого мобільного застосунку чи складної серверної інфраструктури. Таким чином Telegram у межах цього проєкту виступає не лише як звичайний месенджер, а як повноцінний інструмент для реалізації технічної системи обміну даними та взаємодії з ESP32-пристроєм.

Також у Telegram є різні типи чатів: особисті повідомлення, групи та канали [2]. Це є досить зручним, оскільки не потрібно окремо реалізовувати власну систему взаємодії між користувачами - необхідна структура вже присутня в самому месенджері. Наприклад, можна організувати обмін повідомленнями як з одним користувачем, так і одночасно з групою людей або окремою аудиторією через канали [3]. Крім цього, Telegram підтримує швидку доставку повідомлень, роботу на різних пристроях та має зручний програмний інтерфейс для інтеграції ботів. У підсумку виходить достатньо гнучке та універсальне рішення, яке часто використовується під час реалізації подібних систем обміну даними та сповіщеннями.

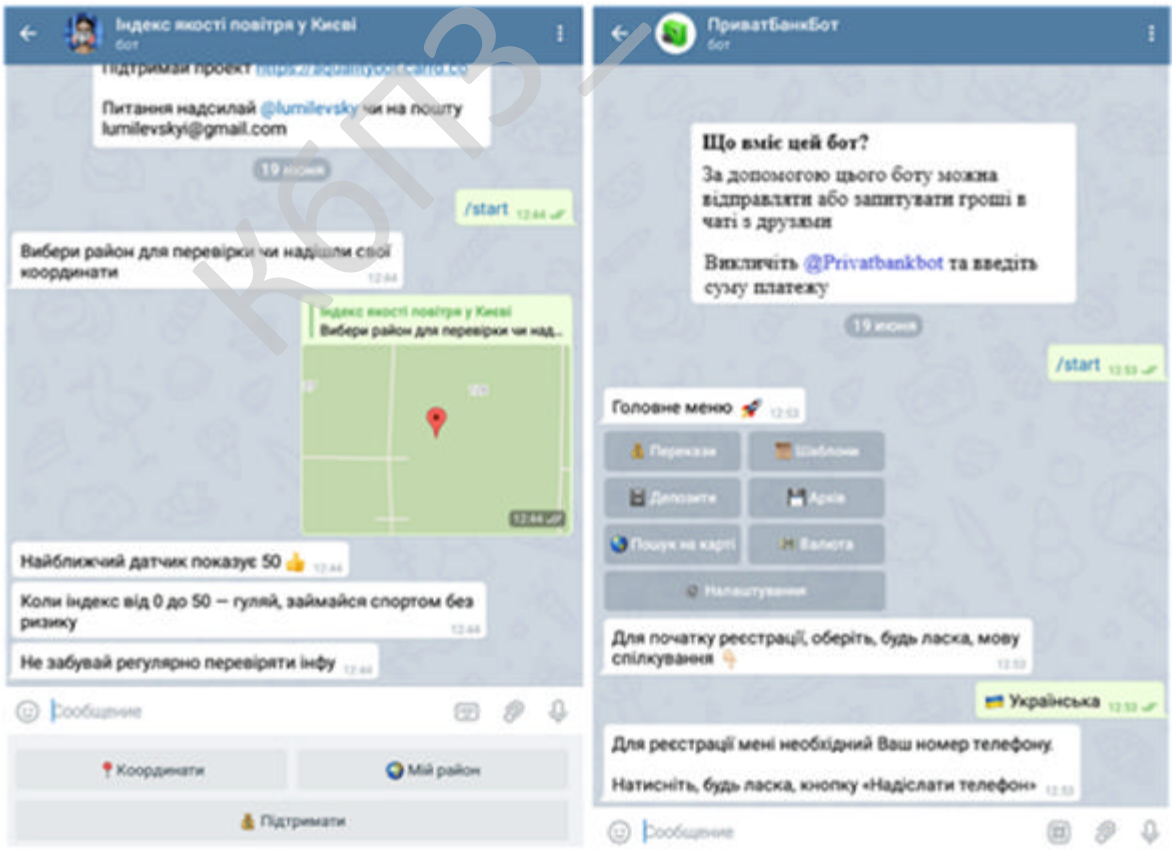
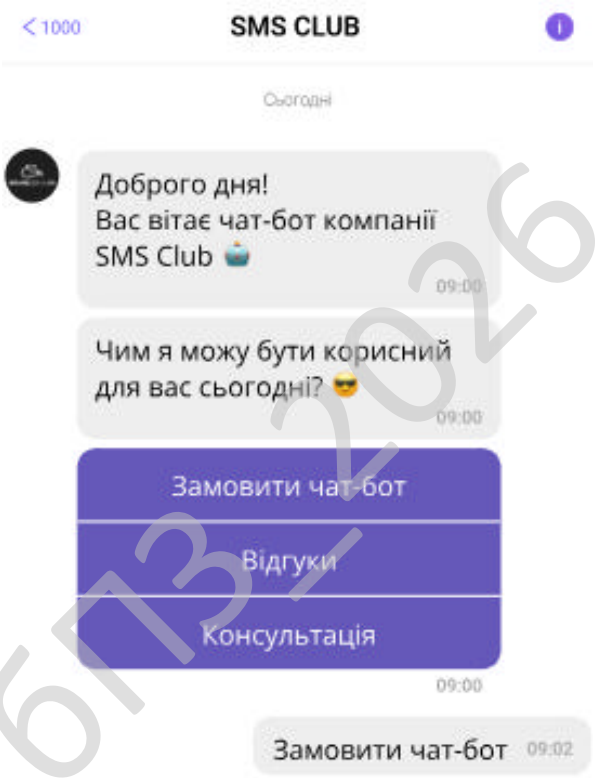


Рисунок 2.1 - Приклад вигляду Телеграм-боту

КПЗ 2026

Viber теж є популярним месенджером, але тут ситуація трохи інша. Він більше орієнтований на звичайне повсякденне спілкування [4]. Повідомлення, дзвінки та групові чати - усе це присутнє та працює стабільно. Однак якщо говорити про автоматизацію або інтеграцію з іншими системами, то таких можливостей значно менше [5]. Через це використовувати його в технічних або апаратних проєктах складніше, оскільки він не надає такого рівня відкритості та інструментів для розробників, як інші платформи.



КОПЗ-2026

Рисунок 2.2 - Приклад вигляду боту в месенджері Viber

Slack виглядає більш “серйозно”, оскільки це вже не просто месенджер, а повноцінний інструмент для організації командної роботи та взаємодії в межах проєктів [6]. У ньому реалізовані канали для комунікації, інтеграції з різними сервісами, розширені налаштування доступів та інші функції, що дозволяють ефективно координувати роботу в командах. Завдяки цьому Slack широко використовується в корпоративному середовищі та великих проєктах, де важлива структурованість і контроль комунікації. Проте через таку кількість можливостей він стає і складнішим у налаштуванні та використанні [7]:

потрібно створювати робочі простори, керувати користувачами, налаштовувати права доступу, а частина розширених функцій доступна лише у платних тарифах. Для простої системи передачі повідомлень або взаємодії з пристроєм на кшталт ESP32 такий підхід може бути надмірним, оскільки ускладнює реалізацію без суттєвих практичних переваг і потребує додаткових ресурсів для підтримки та налаштування.

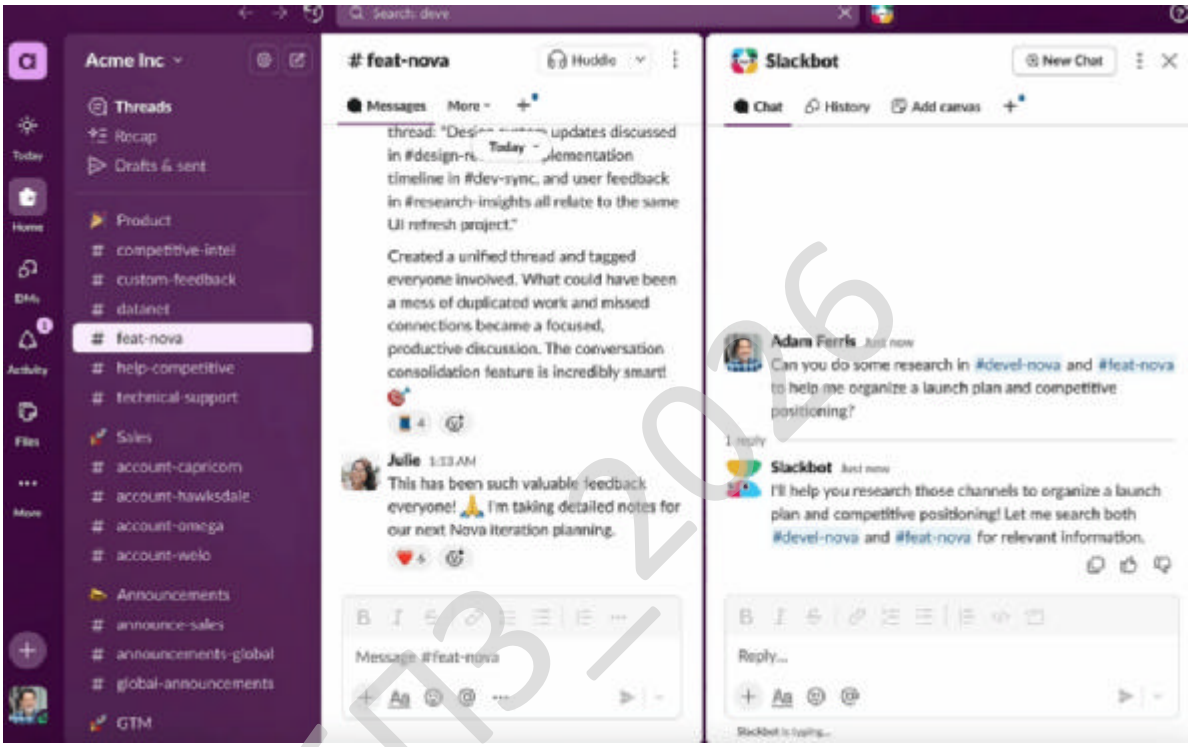


Рисунок 2.3 - Приклад вигляду боту в месенджері Slack

Discord часто використовується для різноманітних спільнот, оскільки він підтримує як текстові, так і голосові канали, а також дозволяє підключати ботів для автоматизації певних функцій [8]. Це робить його досить гнучким інструментом для організації взаємодії між користувачами та реалізації різних інтеграційних рішень. Водночас сама структура Discord більше орієнтована на неформальне спілкування та активні спільноти, де важлива швидка комунікація та велика кількість обговорень [9]. У навчальному або технічному контексті це не завжди є перевагою, оскільки велика кількість каналів, повідомлень і загального інформаційного шуму може відволікати та ускладнювати концентрацію на основній задачі.

КПЗ 2026

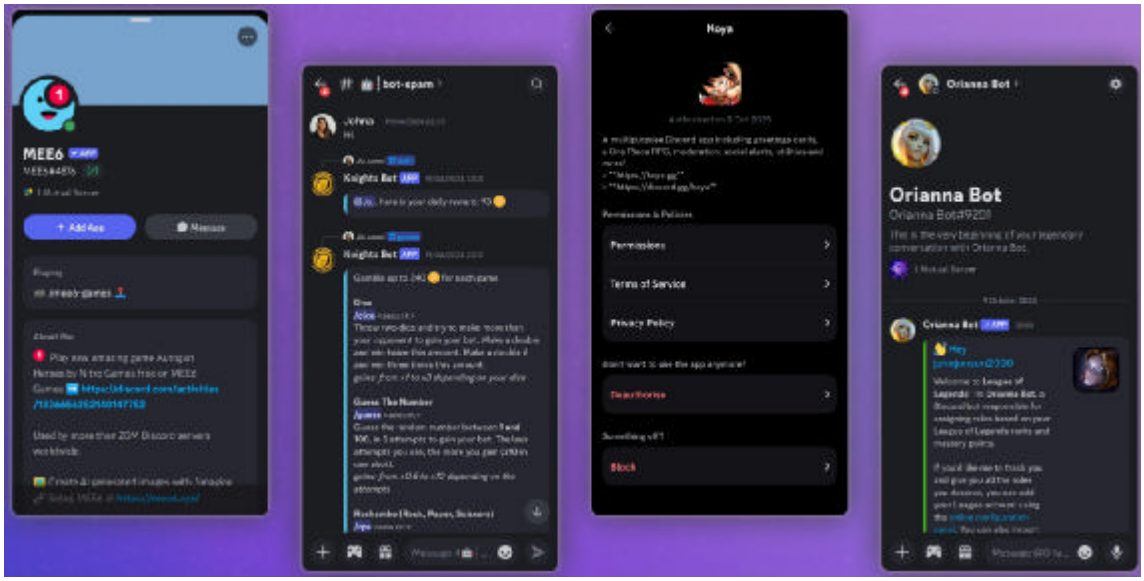


Рисунок 2.4 - Приклад вигляду Discord-боту

Якщо дивитись на всі ці сервіси разом, то стає зрозуміло, що вони відрізняються не тільки функціями, а й тим, як їх використовують. Десь простіше працювати, десь більше можливостей, але і складніше налаштування. Тому при виборі важливо враховувати не лише популярність, а й те, наскільки легко все це можна підключити до інших систем.

Таблиця 2.1 - Порівняльна матриця месенджерів за основними критеріями

Критерій	Telegram	Viber	Slack	Discord
Наявність API	Повний, відкритий	Обмежений	Повний	Частковий
Підтримка ботів	Розвинена	Обмежена	Розвинена	Середня
Популярність серед студентів	Висока	Середня	Низька	Середня
Вартість використання	Безкоштовно	Частково безкоштовно	Частково платно	Безкоштовно
Інтеграція з ESP32	Проста	Ускладнена	Середня	Середня
Простота використання	Висока	Висока	Середня	Середня
Придатність для навчального процесу	Висока	Середня	Середня	Низька

У підсумку можна сказати, що ідеального варіанту немає. Кожен сервіс підходить під свої задачі. Але якщо враховувати саме цю роботу, то більш

КОПЗ_2026

підходящим є варіант, який не потребує складного налаштування і при цьому дозволяє автоматизувати передачу повідомлень. Саме це і було враховано при подальшій розробці.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації проєкту була обрана плата LILYGO T-Display ESP32 (16MB) з вбудованим дисплеєм 1.14 дюйма та USB-UART перетворювачем CH9102F. Загалом ESP32 сьогодні широко використовується в різноманітних IoT-рішеннях, оскільки поєднує достатній рівень продуктивності, вбудовані модулі бездротового зв'язку та відносно низьку вартість [10]. Це робить його зручним вибором для проєктів, де необхідна робота з мережею, обробка даних і взаємодія з периферією. Якщо порівнювати його зі старішими платформами, наприклад Arduino Uno, то можна відзначити суттєві обмеження останньої: відсутність вбудованого Wi-Fi, меншу обчислювальну потужність та обмежені можливості для роботи з сучасними мережевими сервісами [11]. Через це Arduino Uno краще підходить для простих навчальних задач, тоді як ESP32 значно ефективніший для більш складних IoT-систем.

Сам ESP32 має двоядерний процесор Tensilica Xtensa LX6 із тактовою частотою до 240 МГц, а також вбудовані модулі Wi-Fi і Bluetooth [12]. Завдяки такій інтеграції він дозволяє реалізовувати мережеві та обчислювальні задачі без необхідності використання великої кількості додаткових компонентів, що суттєво спрощує апаратну схему пристрою та зменшує кількість зовнішніх з'єднань [13]. Крім того, це підвищує надійність системи, оскільки зменшується кількість потенційних точок відмови та спрощується процес розробки і подальшого масштабування проєкту.

Версія LILYGO T-Display побудована на базі ESP-WROOM-32 і по суті є вже готовим модулем для швидкої розробки пристроїв [14]. Найбільш зручним аспектом у цьому випадку є наявність вбудованого дисплея, завдяки чому немає потреби підключати окремий екран або додаткові контролери. Це суттєво

КОПЗ_2026

спрощує апаратну частину проєкту, оскільки зменшується кількість проводів, з’єднань і зовнішніх компонентів. У результаті економиться місце на платі, підвищується надійність конструкції та скорочується загальний час розробки і тестування пристрою.

Також важливою особливістю є підтримка режимів енергозбереження. Наприклад, режим deep sleep дозволяє суттєво зменшити енергоспоживання пристрою, коли він не виконує активних задач [15]. Це особливо актуально для автономних IoT-систем, де живлення може бути обмеженим або здійснюватися від акумулятора, оскільки дозволяє значно продовжити час роботи пристрою без підзарядки.

Рисунок 2.5 - LILYGO T-Display ESP32 (вигляд спереду)

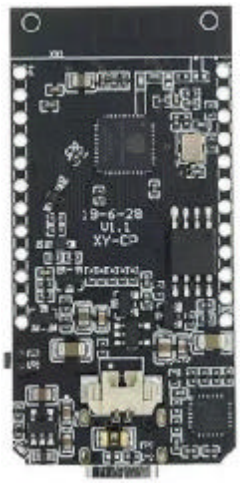


Рисунок 2.6 - LILYGO T-Display ESP32 (вигляд ззаду)

КПЗ-2026

Якщо коротко описати основні особливості плати, то можна виділити наступне:

- є вбудований TFT-дисплей 1.14", що дозволяє одразу виводити дані;
- підключення через CH9102F здійснюється звичайним USB кабелем;
- підтримується Bluetooth (Classic і BLE);
- Wi-Fi працює в різних режимах (клієнт, точка доступу тощо);
- є режими низького споживання енергії;
- підтримується OTA-оновлення прошивки;
- встановлено 16 MB Flash пам’яті [16].

Основні технічні характеристики виглядають так:

- живлення: 3.3-3.7 В;
- зарядка через micro-USB;
- середнє споживання близько 80 мА, пікове до 500 мА;
- RAM 520 KB;
- підтримка стандартних інтерфейсів GPIO, UART, SPI, I2C, ADC, PWM;
- робоча температура: -40…+85°C [17].

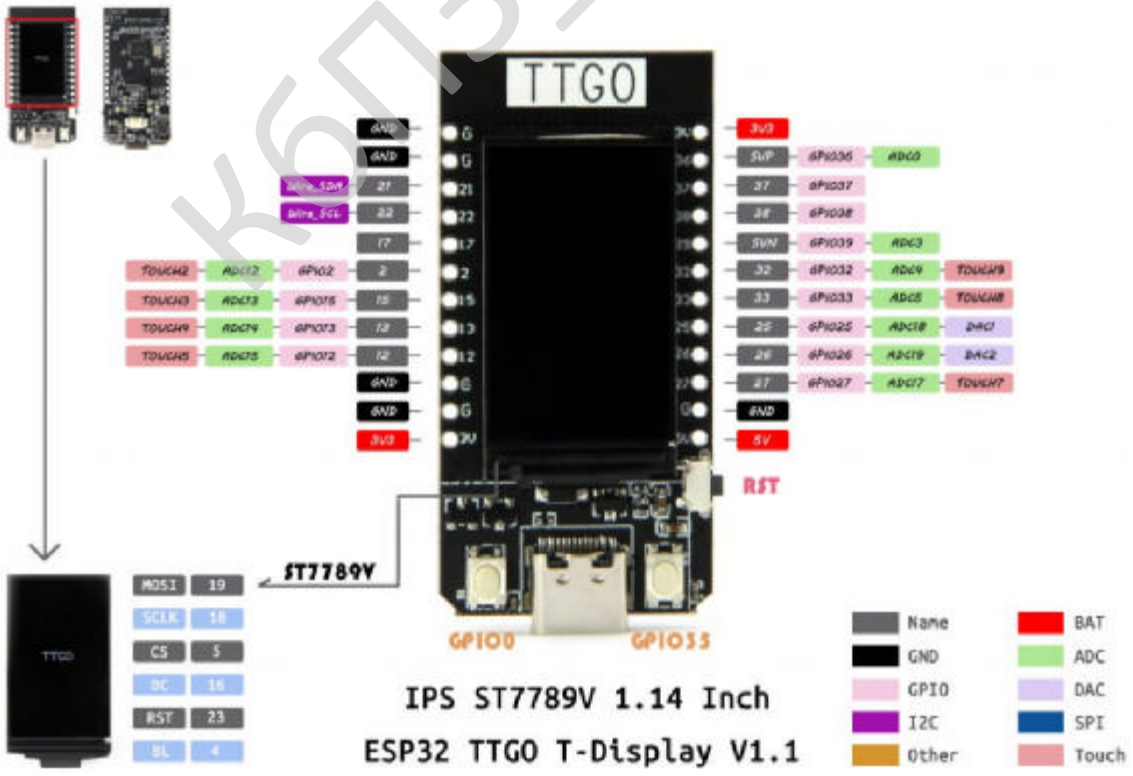


Рисунок 2.7 - Функціональне призначення пінів LILYGO T-Display ESP32

КБПЗ_2026

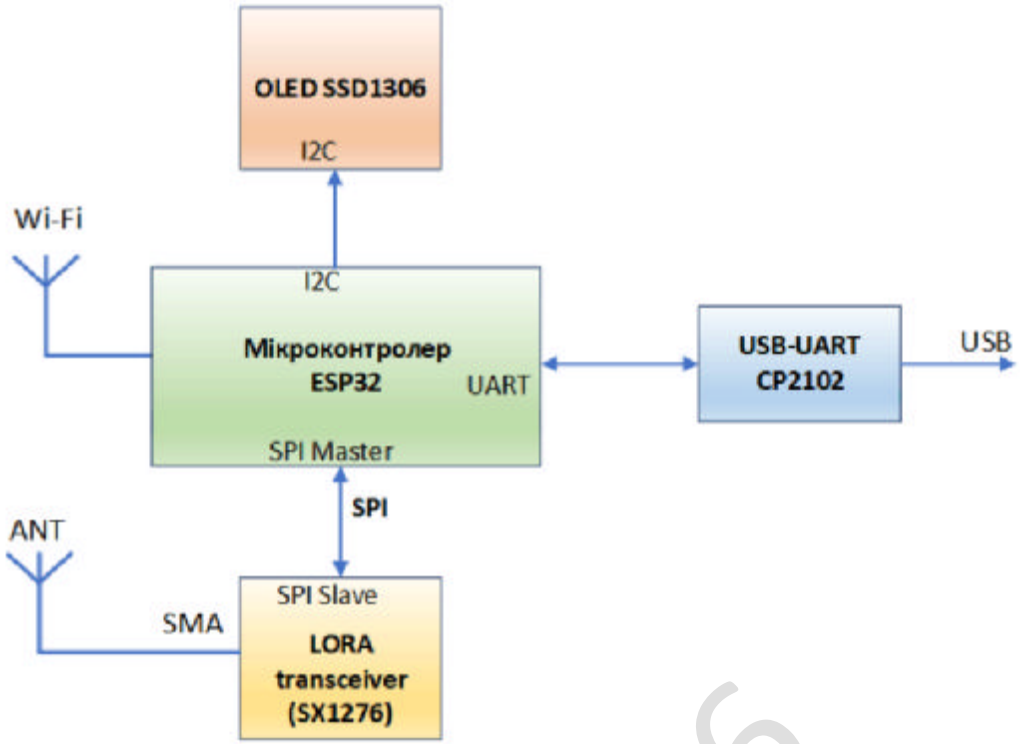


Рисунок 2.8 - Блок-схема мікроконтролера LILYGO T-Display ESP32

Для живлення використовується Li-Ion акумулятор 3.7 В, який підключається до плати напряму [18]. Це дозволяє зробити пристрій автономним, без постійного підключення до мережі.

У системі реалізовано автоматичне перемикання між USB та акумулятором. Також є стабілізація живлення на самій платі, тому додаткові модулі для цього не потрібні. Зарядка відбувається через micro-USB [19].

Рисунок 2.9 - Вид зовнішньої батареї

КБПЗ-2026

ESP32 має кілька практичних переваг, завдяки яким він і використовується в даному проєкті. Насамперед це стабільна робота модуля Wi-Fi, що дозволяє без проблем підключатися до серверів Telegram Bot API і отримувати або передавати дані в реальному часі з мінімальними затримками [20]. Також важливою перевагою є підтримка різноманітних апаратних інтерфейсів, таких як UART, SPI та I2C. Це дає можливість підключати додаткові сенсори, модулі або інші периферійні пристрої, якщо виникне потреба розширити функціональність системи або адаптувати її під інші задачі [21]. Ще одним суттєвим плюсом є підтримка захищених з'єднань TLS/SSL, що забезпечує безпечну передачу даних через інтернет і є особливо важливим при роботі з різними сервісами та API.

Окремо варто відзначити низьке енергоспоживання ESP32, оскільки в режимі deep sleep пристрій споживає мінімальну кількість енергії, що дозволяє використовувати його в автономних умовах або від акумулятора протягом тривалого часу без частого підзаряджання [22]. Це особливо важливо для IoT-систем, де пристрій значну частину часу може перебувати в неактивному режимі. Для програмування використовується C++ [23], що є зручним завдяки прямому доступу до апаратних ресурсів, ефективній роботі з пам'яттю та високій продуктивності виконання коду. У випадку ESP32 це має особливе значення, оскільки пристрій одночасно обробляє мережеві запити, працює з Wi-Fi-з'єднанням та виконує логіку відображення даних у реальному часі.

Середовище розробки - PlatformIO у Visual Studio Code [24]. Воно автоматично керує залежностями проєкту, підтягує необхідні бібліотеки, дозволяє зручно структурувати кодову базу та забезпечує швидке тестування роботи пристрою через серійний порт. Завдяки цьому процес розробки стає більш організованим і передбачуваним, а також спрощується відлагодження коду на етапі створення пристрою.

Для взаємодії з користувачем використовується Telegram Bot API [25]. Він дозволяє реалізувати обмін повідомленнями між користувачем і системою в реальному часі, забезпечує можливість надсилання та отримання текстових даних, а також підтримує використання кнопок і простих елементів керування. Це робить взаємодію з пристроєм більш зручною та інтуїтивною, не вимагаючи

КОПЗ_2026

створення окремого інтерфейсу користувача.

Основна логіка частково винесена на серверну частину, тому навантаження на сам мікроконтролер є невеликим. Це дозволяє спростити роботу пристрою, зменшити вимоги до обчислювальних ресурсів і водночас підвищити гнучкість системи, оскільки зміни у функціоналі можна вносити без необхідності перепрошивки мікроконтролера.

У підсумку поєднання LILYGO T-Display, C++, середовища PlatformIO і Telegram Bot API дає просту, але водночас стабільну та масштабовану систему. Вона може бути ефективно використана в різних IoT-проєктах, особливо в тих випадках, де потрібна віддалена передача даних, швидке реагування на повідомлення та мінімальна складність апаратної реалізації.

2.3 Розгорнута постановка завдання

Під час розробки системи дистанційного моніторингу через Telegram-бота основна ідея була досить простою. Потрібно було зробити не теоретичну модель, а реальний пристрій, який дійсно працює. Тобто щоб він міг передавати дані, підключатися до мережі і нормально поводитись у звичайних умовах.

Подібні рішення зараз зустрічаються доволі часто. Їх використовують і в навчальних роботах, і в різних невеликих проєктах. Особливо там, де важливо швидко отримати інформацію з пристрою без складної інфраструктури. Саме тому було вирішено реалізувати щось подібне, але без зайвих ускладнень, з акцентом на простоту архітектури та практичну зручність використання в реальних умовах.

Перед початком роботи розглядалися готові варіанти, такі як Blynk, MQTT, Firebase або Node-RED [26]. Вони дійсно підходять за можливостями, але на практиці виявилось, що або вони занадто перевантажені, або потребують додаткових сервісів [27]. У результаті було прийнято рішення не використовувати готові рішення, а зробити власну реалізацію, максимально адаптовану під конкретні вимоги проєкту та його архітектуру.

Як апаратну основу взято LILYGO T-Display ESP32. Вона досить зручна

КПІЗ_2026

для таких задач. Тут вже є Wi-Fi, Bluetooth і невеликий дисплей. Це означає, що не потрібно підключати додаткові модулі тільки для виводу інформації. Відповідно, схема стає простішою.

Програмна частина реалізована мовою C++ з використанням PlatformIO у Visual Studio Code. Це типовий і зручний підхід для розробки під ESP32, який забезпечує достатній рівень контролю над роботою мікроконтролера та дозволяє ефективно організувати структуру проєкту. Основна логіка системи включає отримання даних, їх подальшу обробку та передачу через мережу, а також паралельну роботу з периферійними компонентами, зокрема дисплеєм та інтерфейсами введення/виведення.

Окремо довелося приділити увагу питанню стабільності мережевого підключення [28]. У випадку втрати Wi-Fi-з'єднання система не зупиняє свою роботу повністю, а переходить у режим повторних спроб підключення і продовжує виконання основних функцій після відновлення зв'язку. Такий підхід є важливим для подібних IoT-пристроїв, оскільки дозволяє уникнути збоїв у роботі та забезпечує більш надійну й передбачувану поведінку системи в реальних умовах експлуатації.

Для взаємодії використовується Telegram Bot API [29]. У цій системі взаємодія організована досить просто: користувач надсилає команду через бот, а пристрій у відповідь повертає необхідну інформацію, наприклад статус системи або отримані дані. Такий підхід є зручним, оскільки не потребує розробки окремого інтерфейсу користувача і дозволяє використовувати вже готову платформу для комунікації, що спрощує реалізацію системи та прискорює процес її впровадження й тестування.

Під час тестування також було перевірено поведінку системи за різного рівня Wi-Fi сигналу, швидкість отримання відповідей та стабільність передачі даних у різних умовах роботи. Окремо аналізувалась коректність обробки запитів і відсутність втрат повідомлень під час навантаження. Крім того, для уникнення помилок у роботі довелося уточнити окремі моменти документації ESP32, що дозволило більш точно налаштувати взаємодію з апаратною частиною та підвищити стабільність системи.

КбПЗ_2026

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У даній роботі розглядається система, де IoT-пристрій працює у зв’язці з Telegram ботом. Ідея доволі проста - забезпечити обмін даними без використання складних серверних рішень та зайвих налаштувань інфраструктури. Тобто пристрій повинен отримувати команди від користувача та відповідати на них через месенджер у зручному текстовому форматі, що робить взаємодію швидкою та зрозумілою. Такий підхід дозволяє значно спростити архітектуру системи, оскільки замість окремого інтерфейсу або спеціалізованого програмного забезпечення використовується вже готова платформа обміну повідомленнями, яка забезпечує стабільну передачу даних і підтримує роботу в реальному часі. Крім того, це підвищує доступність системи, оскільки користувач може взаємодіяти з пристроєм з будь-якого місця, де є підключення до інтернету, без додаткових вимог до обладнання чи програмного забезпечення.

Перед тим як зупинитись на цьому варіанті, розглядалися й інші підходи та платформи. Загалом вони також підходять для реалізації подібних задач, однак мають певні обмеження: у деяких випадках необхідно розгортати окремий сервер і постійно підтримувати його роботу, в інших процес налаштування є доволі складним і потребує додаткових технічних знань, а іноді виникає значна залежність від сторонніх сервісів або платних рішень, що може ускладнювати масштабування та подальший розвиток системи. Окремо варто враховувати й фактор часу на впровадження, оскільки деякі платформи вимагають тривалої конфігурації перед початком роботи. Усе це в сукупності ускладнює розробку та підвищує вимоги до підтримки системи в майбутньому. Через це було прийнято рішення використати Telegram, оскільки він уже надає готову інфраструктуру для роботи з повідомленнями, має зручний інтерфейс для взаємодії через ботів і дозволяє швидко реалізувати необхідний функціонал без складнощів [30].

КПЗ – 2026

Обмін даними реалізований через регулярну перевірку нових повідомлень. Пристрій сам періодично звертається до Telegram і перевіряє, чи з'явилися нові дані - такий підхід називається polling [31]. Це дозволяє зробити систему простішою з точки зору архітектури, оскільки відсутня необхідність у налаштуванні відкритих портів, організації складних серверних рішень або використанні виділеного сервера з публічною IP-адресою. У результаті зменшується кількість залежностей від зовнішньої інфраструктури, а процес розгортання та подальшого обслуговування системи стає значно простішим і більш передбачуваним. Також такий підхід добре підходить для невеликих IoT-проєктів, де важлива стабільність роботи та мінімальна складність реалізації.

Робота системи виглядає наступним чином: спочатку запускається мікроконтролер ESP32 та ініціалізуються основні програмні модулі, включно з налаштуванням периферії, підготовкою мережевих компонентів і стартовою конфігурацією необхідних бібліотек. Це дозволяє привести пристрій у робочий стан і забезпечити коректну взаємодію всіх апаратних та програмних частин ще на початковому етапі запуску. Далі виконується підключення до Wi-Fi-мережі, після чого встановлюється взаємодія з сервісом Telegram через відповідний API. Після завершення етапу ініціалізації система переходить у безперервний робочий цикл, у якому періодично виконується перевірка наявності нових повідомлень або команд від користувача, що забезпечує стабільну роботу пристрою та дозволяє оперативно реагувати на отримані запити без затримок у межах заданої логіки програми.

У випадку надходження команди вона проходить попередню перевірку та обробку відповідно до заданої логіки програми, після чого виконується відповідна дія, наприклад оновлення даних на дисплеї, зміна режиму роботи пристрою. У процесі обробки також можуть враховуватися поточний стан системи та доступність мережевого з'єднання, що дозволяє коректно реагувати на різні сценарії роботи. Такий підхід забезпечує стабільну, передбачувану та автономну роботу пристрою навіть за умов нестабільного інтернет-з'єднання або тимчасової втрати зв'язку, оскільки критично важливі функції можуть виконуватися локально. Крім того, архітектура системи дозволяє легко

КОПЗ_2026

розширювати її функціональність шляхом додавання нових команд і сценаріїв роботи без суттєвих змін у базовій структурі програми, що підвищує гнучкість і масштабованість рішення.

Додатково така реалізація перетворює пристрій не лише на пасивний виконавець команд, а й на активний елемент системи, який може самостійно ініціювати повідомлення про події, зміни стану або натискання фізичних елементів керування. Це забезпечує двонаправлений обмін даними між користувачем і пристроєм у режимі близькому до реального часу. Завдяки цьому підвищується практична цінність рішення, оскільки забезпечується швидкий зворотний зв’язок у Telegram без постійного контролю з боку користувача, а також підвищується надійність системи в IoT-сценаріях. Крім того, така організація взаємодії дозволяє ефективніше масштабувати систему для складніших сценаріїв автоматизації.

Для тестування використано LILYGO T-Display на базі ESP32. Вона є зручною тим, що вже містить вбудований дисплей, тому відсутня потреба підключати додаткові модулі для відображення стану системи або налагоджувальної інформації. Це дозволяє значно спростити процес розробки та тестування, оскільки вся необхідна інформація може виводитися безпосередньо на екран пристрою, що прискорює перевірку логіки роботи та виявлення можливих помилок. Завдяки компактним розмірам плати можна швидко зібрати робочий прототип і перевірити основні сценарії роботи пристрою без складного монтажу та великої кількості з’єднань [32].

Сам програмний код реалізовано таким чином, щоб у майбутньому його можна було легко змінювати та масштабувати. Архітектура програми побудована з урахуванням можливого розширення функціональності, тому за потреби можна додавати нові команди, інтегрувати додаткові сенсори або змінювати логіку обробки даних без суттєвих змін у базовій структурі проєкту. Це робить систему більш гнучкою, спрощує її подальший розвиток і дозволяє адаптувати пристрій під різні сценарії використання без необхідності повної переробки коду. Додатково це сприяє зменшенню витрат часу на супровід і тестування системи під час її модернізації.

КОПЗ_2026

3.2 Розробка структурної схеми

Структурна схема є важливим елементом при проєктуванні програмно-апаратних систем, оскільки дозволяє наочно відобразити основні складові комплексу та зв’язки між ними [33]. За допомогою такої схеми можна зрозуміти, з яких функціональних блоків складається система та яким чином вони взаємодіють між собою під час обміну даними. У даній дипломній роботі структурна схема відображає процес передавання повідомлень від куратора через Telegram-бота та групу до кінцевого пристрою на базі ESP32, який використовується для відображення сповіщень студентам [34].

Крім відображення загальної архітектури системи, структурна схема дозволяє простежити повний шлях проходження інформації - від створення повідомлення до його отримання та виведення на дисплей пристрою. Вона наочно демонструє взаємодію між окремими функціональними блоками системи, зокрема Telegram-ботом, мережевими сервісами та мікроконтролером ESP32. Такий підхід допомагає краще зрозуміти принцип роботи системи та особливості обміну даними між її компонентами. Структурна схема спрощує аналіз взаємодії між програмною та апаратною частинами, а також полегшує подальшу модернізацію й тестування окремих компонентів комплексу. Завдяки чіткому поділу системи на функціональні блоки можна швидше виявляти можливі несправності, оптимізувати роботу окремих модулів і розширювати функціональні можливості системи без суттєвих змін її загальної структури.

Слід зазначити, що структурна схема не описує алгоритм роботи системи у вигляді послідовності виконуваних дій, а лише демонструє загальну архітектуру та взаємозв’язки між її компонентами. Вона використовується для узагальненого представлення системи, що є особливо важливим у випадках, коли проєкт включає як апаратну, так і програмну частини, а також взаємодію через мережу Інтернет [35]. За допомогою структурної схеми можна визначити основні модулі системи, канали передавання даних між ними та принцип організації обміну інформацією. Тому поділ системи на окремі функціональні блоки є доцільним, оскільки значно спрощує її розуміння, аналіз і подальше масштабування або

КПІЗ_2026

модернізацію. Крім того, такий підхід дозволяє окремо тестувати компоненти системи, швидше виявляти можливі несправності та оптимізувати роботу програмно-апаратного комплексу. Завдяки цьому забезпечується підвищення надійності системи та спрощується процес її подальшого вдосконалення.

Побудова схеми виконувалася поступово, починаючи з визначення загальних функцій системи, після чого відбувалося їх поетапне уточнення та деталізація до рівня окремих функціональних елементів. У результаті такого підходу було виділено основні компоненти системи, зокрема: керуючий модуль Telegram-бота, блок фільтрації та обробки повідомлень, модуль пересилання даних, IoT-пристрій із дисплеєм на базі ESP32, модуль бездротового зв’язку Wi-Fi, кнопка для локального керування, модуль відображення часу, а також блок визначення дня тижня. Така декомпозиція дозволяє чітко розділити відповідальність між різними частинами системи та спростити подальше проєктування й реалізацію.

Передавання даних між зазначеними компонентами здійснюється через мережу Інтернет із використанням бездротового підключення Wi-Fi. Telegram-бот отримує повідомлення від куратора, після чого ці дані проходять етап обробки та фільтрації відповідно до заданої логіки системи. На цьому етапі виконується перевірка коректності отриманої інформації, визначення її типу та підготовка до подальшого передавання на кінцевий пристрій. Такий підхід дозволяє забезпечити стабільну та безперебійну роботу системи навіть у разі великої кількості повідомлень або короткочасних затримок мережевого з’єднання. Далі інформація передається до кінцевого пристрою, де мікроконтролер ESP32 обробляє отримані дані та відображає їх на дисплеї. Після отримання повідомлення пристрій виконує оновлення інформації на екрані та, за необхідності, активує додаткові функції сповіщення користувача. У результаті студент отримує готове повідомлення у зручному для сприйняття вигляді, що забезпечує швидкий доступ до актуальної інформації, оперативність сповіщення та надійне функціонування всієї програмно-апаратної системи.

Основним елементом системи (рисунок 3.1) є Telegram-бот, оскільки саме через нього здійснюється основне керування всією логікою взаємодії. За його

КОПЗ_2026

допомогою користувач може обирати отримувачів повідомлень, переглядати списки доступних груп або студентів, а також визначати поточний навчальний тиждень і формувати відповідний тип сповіщення. Таким чином бот виступає центральною точкою керування, яка об’єднує всі функції системи в одному інтерфейсі та забезпечує зручну взаємодію без потреби у додаткових програмних засобах. Після виконання необхідних дій формується структурований запит, який передається до кінцевого пристрою через мережу Інтернет. Передавання даних здійснюється з використанням захищеного протоколу HTTPS, що забезпечує шифрування інформації та підвищує безпеку обміну даними між серверною частиною і пристроєм на базі ESP32. У результаті система отримує надійний канал зв’язку, який мінімізує ризики перехоплення або спотворення даних під час передачі, а також забезпечує стабільну роботу в умовах реального використання.

Окремо варто відзначити, що пристрій має певні автономні можливості, які дозволяють йому виконувати базові функції навіть без активного підключення до мережі Інтернет. Наприклад, за допомогою фізичної кнопки користувач може переглянути поточний час і дату, що відображаються на вбудованому дисплеї. Така функція реалізована локально на мікроконтролері ESP32 і не залежить від зовнішніх сервісів чи серверної частини системи. Це робить пристрій більш надійним у використанні, оскільки навіть у випадку втрати Wi-Fi-з’єднання основні базові функції залишаються доступними. Крім того, така автономність підвищує зручність експлуатації, адже користувач може швидко отримати необхідну інформацію без додаткових запитів до Telegram або інших сервісів, що особливо важливо в умовах нестабільного підключення до мережі.

Також у системі передбачено визначення типу навчального тижня. Для цього можуть використовуватися дані з локальної бази, попередньо заданих налаштувань або зовнішніх ресурсів, які містять актуальну інформацію про навчальний процес. Після отримання необхідних даних система автоматично визначає поточний тип тижня та передає цю інформацію на кінцевий пристрій для подальшого відображення.

КОПЗ_2026

Отримана інформація відображається користувачам і дозволяє уникнути плутанини у розкладі занять, особливо у випадках чергування чисельників і знаменників. Автоматизація цього процесу спрощує доступ студентів до актуальної інформації та зменшує ймовірність помилок. Крім того, використання автоматичного визначення типу тижня підвищує зручність користування системою та покращує її функціональність.

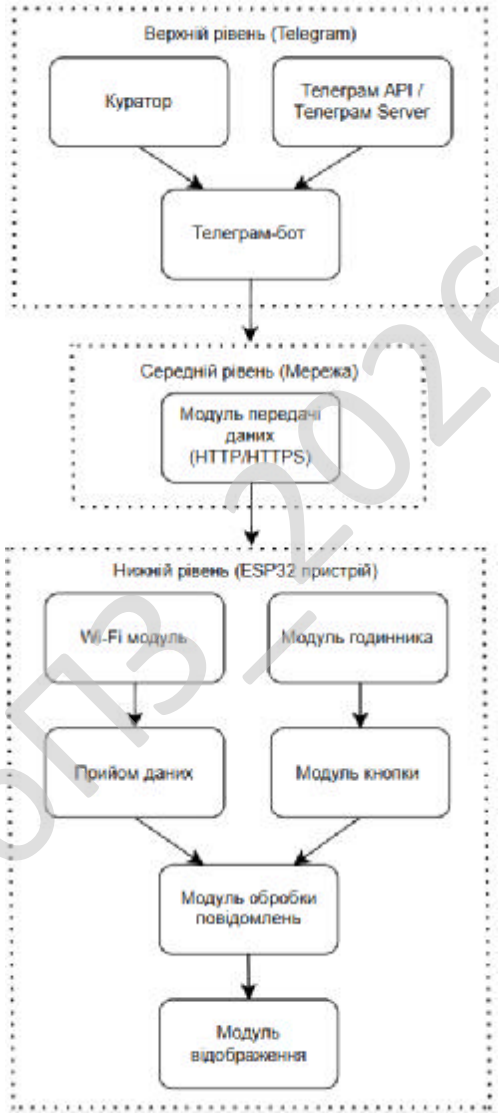


Рисунок 3.1 - Структурна схема пристрою

Запропонована структура є модульною, що дає можливість у подальшому розширювати систему. За потреби можуть додаватися нові компоненти без значних змін уже існуючих елементів. Це робить систему більш гнучкою та придатною до подальшого розвитку.

КбПЗ 2026

3.3 Розробка функціональної схеми

Функціональна схема IoT-системи синхронізації смартгодинника на базі ESP32 з Telegram-ботом призначена для наочного відображення основних функціональних вузлів системи та принципів їх взаємодії. Вона демонструє, яким чином здійснюється отримання інформації з серверної інфраструктури Telegram і подальше її передавання на пристрій користувача для відображення на екрані смартгодинника або іншого IoT-пристрою. Усі елементи системи об'єднані в єдину логічну структуру, що забезпечує безперервний обмін даними, узгоджену роботу компонентів та взаємодію з користувачем у режимі реального часу [36].

Telegram-бот у даній системі виконує роль проміжної програмної ланки між користувачем і пристроєм на базі ESP32. Він приймає повідомлення з групових чатів, виконує їх обробку відповідно до заданої логіки та забезпечує зберігання даних у хмарній інфраструктурі Telegram. Крім того, бот дозволяє організувати адресну або групову розсилку повідомлень, що робить систему більш гнучкою та зручною у використанні як для індивідуальних користувачів, так і для великих навчальних або організаційних груп.

Модуль доступу до Telegram Bot API забезпечує підключення до хмарної частини системи та організовує основний канал обміну даними між пристроєм і серверною інфраструктурою. За його допомогою виконується автентифікація пристрою, отримання нових повідомлень, а також обробка команд, що надходять від користувача. Передача даних відбувається через захищене з'єднання із використанням протоколу HTTPS, що забезпечує шифрування інформації та підвищує загальний рівень безпеки системи під час роботи в мережі Інтернет.

Обмін даними з боку мікроконтролера ESP32 реалізовано за клієнтським принципом. Пристрій періодично надсилає запити до Telegram Bot API та отримує відповідь у форматі JSON, який є стандартним для структурованого обміну даними. У цих відповідях міститься текст повідомлення, інформація про відправника, ідентифікатори повідомлень, а також інші службові параметри,

КОПЗ_2026

необхідні для коректної обробки. Отримані дані тимчасово зберігаються в пам’яті пристрою та надалі використовуються для обробки логікою програми і відображення на дисплеї або виконання відповідних дій.

Для підключення до мережі використовується вбудований Wi-Fi модуль ESP32. Він забезпечує доступ до Інтернету та є обов’язковим компонентом для коректної роботи всієї системи, оскільки саме через мережу здійснюється обмін даними з Telegram Bot API. При нестабільному або слабкому з’єднанні можуть виникати затримки в отриманні повідомлень або повторні спроби підключення, що безпосередньо впливає на швидкість реакції пристрою та загальну стабільність роботи системи.

Обробка отриманих повідомлень виконується безпосередньо на ESP32. Програмний модуль відповідає за розбір отриманих JSON-даних, виділення необхідних полів, відсіювання дубльованих або повторно отриманих повідомлень, а також підготовку тексту для подальшого відображення. Форматування виконується з урахуванням обмежень та роздільної здатності дисплея, що дозволяє коректно виводити інформацію без перекриттів або втрати частини тексту. Це забезпечує зручне сприйняття даних користувачем і стабільну роботу інтерфейсу пристрою.

У системі також передбачено використання фізичної кнопки. Вона використовується для перемикання режимів відображення. Зокрема, за її допомогою можна переглянути поточний час і дату.

Для відображення часу використовується синхронізація через NTP-сервери. Внутрішній таймер ESP32 періодично оновлюється, що дозволяє підтримувати актуальні значення часу навіть при тривалій роботі пристрою або після перезапуску системи. Завдяки цьому забезпечується коректна синхронізація з реальним часом, а також зменшується похибка внутрішнього годинника. Відображення часу виконується за запитом користувача або у визначені моменти роботи системи, залежно від логіки програми.

Виведення інформації здійснюється на TFT-дисплей ST7789, який підключений до ESP32. На екрані можуть відображатися текстові повідомлення, дата, актуальний час, а також службова інформація про стан підключення до

КОПЗ_2026

мережі або роботи системи. Це дозволяє користувачу візуально контролювати стан пристрою в режимі реального часу та швидко отримувати необхідні дані без додаткових запитів через Telegram.

Живлення пристрою реалізовано за допомогою літій-полімерного акумулятора. Передбачено модуль заряджання через інтерфейс USB Type-C. Система живлення враховує особливості роботи Wi-Fi модуля та забезпечує стабільну роботу пристрою.

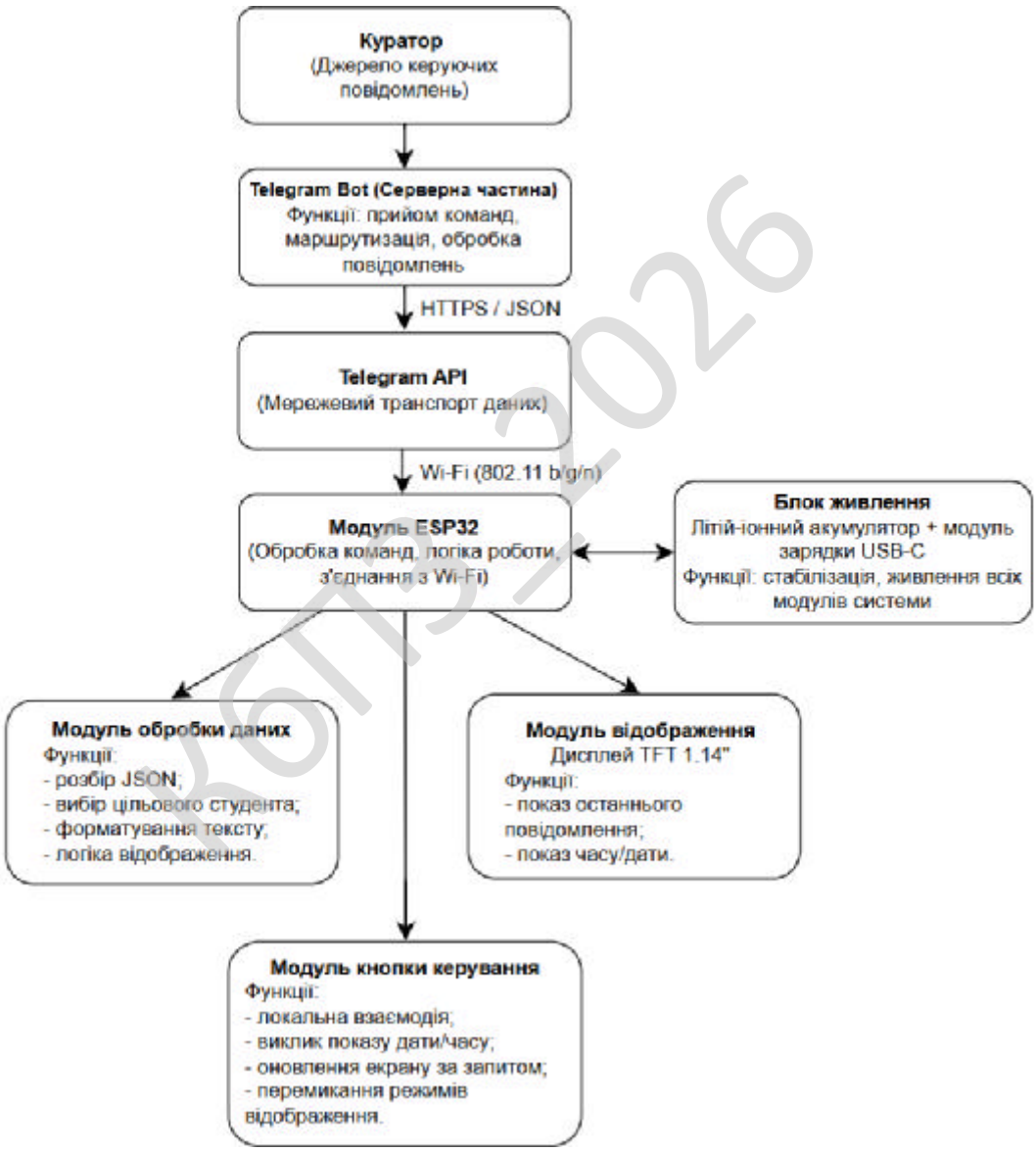


Рисунок 3.2 - Функціональна схема пристрою

У цілому функціональна схема відображає взаємодію основних компонентів системи. ESP32 у даному випадку виступає центральним

елементом, який забезпечує обробку даних та керування іншими модулями. Схема дозволяє простежити шлях інформації від сервера Telegram до кінцевого відображення на дисплеї. Такий підхід спрощує розуміння роботи системи та може бути використаний при її подальшому вдосконаленні.

Розробка принципової схеми пристрою

Принципова схема пристрою для синхронізації IoT-модуля ESP32 з даними приватної Telegram-групи побудована з використанням мінімальної кількості компонентів. Такий підхід дозволяє забезпечити стабільну роботу системи та коректне відображення отриманої інформації. Основою пристрою є плата LILYGO T-Display ESP32, яка має вбудований TFT-дисплей і підтримує роботу через Wi-Fi. Завдяки цьому немає необхідності використовувати окремі модулі для виведення інформації, що спрощує конструкцію пристрою [37].

Для взаємодії з мікроконтролером ESP32 передбачено лише один зовнішній елемент керування - фізична кнопка. Вона використовується для локальної взаємодії з пристроєм і дозволяє користувачу виконувати базові дії без необхідності підключення до мережі або використання Telegram. Такий підхід робить систему більш автономною та зручною у випадках, коли мережевий доступ обмежений або відсутній. Кнопка підключена до GPIO-виводу мікроконтролера з використанням внутрішнього підтягування (internal pull-up), що дозволяє уникнути використання додаткових резисторів та спрощує апаратну схему пристрою. При натисканні кнопки виконується обробка події в програмному коді, після чого на дисплеї відображається поточний час і дата, синхронізовані через NTP. Це забезпечує швидкий доступ до базової інформації та підвищує зручність використання пристрою в повсякденній експлуатації.

Плата LILYGO T-Display має вбудований перетворювач USB-UART, тому живлення і програмування здійснюється через стандартний USB-порт. Це означає, що немає потреби у використанні окремих джерел живлення або додаткових модулів. Також варто зазначити, що дисплей, Wi-Fi модуль та інші необхідні елементи вже з’єднані на платі виробником, тому у принциповій схемі відображаються лише основні підключення [38], що суттєво спрощує процес розробки системи загалом.

КОПЗ_2026

Рисунок 3.3 - Принципова схема IoT-пристрою з синхронізацією Telegram-бота

Рисунок 3.4 - Схема системи IoT-пристрою з синхронізацією Telegram-бота на реальному фото

У цілому принципова схема показує просту апаратну реалізацію пристрою, яка відповідає поставленій задачі. Використання мінімальної кількості зовнішніх компонентів зменшує ймовірність відмов та спрощує подальше обслуговування. При цьому функціональність пристрою залишається достатньою для прийому повідомлень і взаємодії з користувачем.

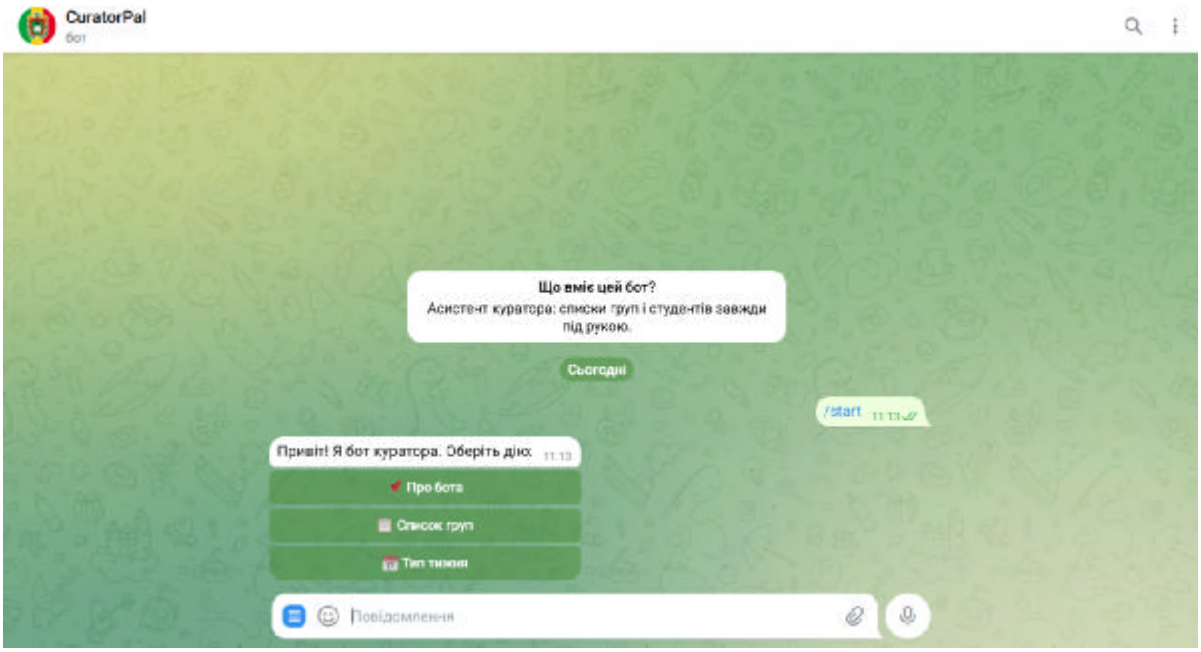


Рисунок 3.5 - Вигляд створеного Telegram-боту

На даному рисунку зображено зовнішній вигляд Telegram-бота та його стартовий інтерфейс. Після початку взаємодії з ним користувач отримує доступ до головного меню, яке містить основні функції системи та забезпечує зручну навігацію між доступними можливостями. Це дозволяє одразу розпочати роботу з ботом без додаткових налаштувань.

Рисунок 3.6 - Меню вибору конкретного студента для надіслання повідомлення на індивідуальний пристрій

КПЗ_2026

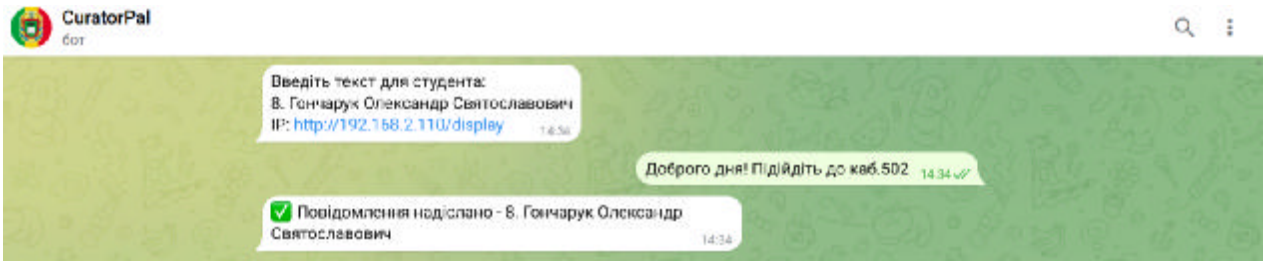


Рисунок 3.7 - Вигляд вікна відправлення індивідуального повідомлення

На рисунках 3.6 та 3.7 наочно продемонстровано меню вибору конкретного студента, а також інтерфейс, що відображається після його вибору. У першому випадку користувач бачить список доступних студентів для надсилання повідомлення, що забезпечує зручний пошук і вибір потрібного отримувача. Після вибору конкретного студента система переходить до режиму введення тексту та відображає повідомлення про можливість формування й відправки повідомлення на його пристрій. Це дозволяє чітко простежити логіку взаємодії користувача з ботом на кожному етапі та забезпечує інтуїтивно зрозумілий процес роботи з функцією персоналізованих повідомлень.

Рисунок 3.8 - Вигляд вікна інформації про тип тижня

На даному рисунку представлено результат натискання кнопки «Тип тижня» та виконання відповідної функції. Після обробки запиту користувач отримує інформацію про поточний тип навчального тижня - чисельник або знаменник. Це дозволяє швидко та зручно визначити актуальний розклад навчального процесу в межах системи.

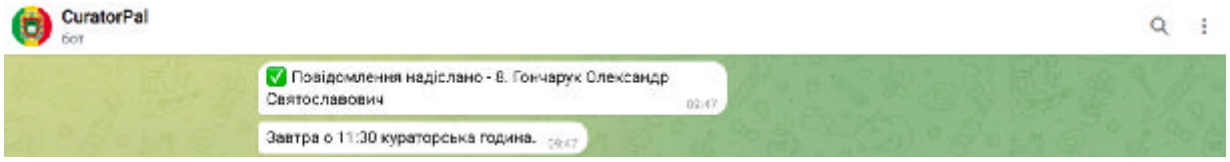


Рисунок 3.9 - Приклад нагадування про кураторську годину

КОПЗ_2026

Узагальнюючи можна описати створену систему що у Telegram-боті реалізовано низку функціональних можливостей, які забезпечують зручну взаємодію користувача із системою. Зокрема, через кнопку «Про бота» користувач може отримати коротку інформацію про призначення та функціонал бота. Кнопка «Список груп» дозволяє переглянути доступні групи та здійснити надсилання повідомлень як окремому студенту, так і всій обраній групі. За допомогою розділу «Тип тижня» можна дізнатися поточний навчальний тиждень - чисельник або знаменник.

Окремо реалізовано систему автоматичних нагадувань про кураторську годину. Такі повідомлення надсилаються раз на місяць, у перший вівторок місяця, та мають кілька часових етапів: за день до події - о 09:00 та 18:00, а також у день проведення - о 09:00 (основне нагадування) і о 11:25, тобто за 5 хвилин до початку заходу.

Таким чином, реалізований функціонал робить Telegram-бот зручним інструментом для оперативної комунікації та організації навчального процесу, оскільки дозволяє своєчасно інформувати студентів, зменшує потребу в ручному розсиланні повідомлень і підвищує загальну ефективність взаємодії між куратором та групами.

3.4 Розробка діаграми процесів

Для опису роботи системи використовується покрокове представлення, яке дозволяє чітко відобразити послідовність виконуваних дій і взаємодію між окремими компонентами. Такий підхід застосовується для того, щоб спростити розуміння загального принципу функціонування системи та показати, як саме дані проходять через усі етапи обробки - від користувача до кінцевого пристрою і назад. Завдяки цьому можна детально простежити процес передавання, обробки та відображення інформації в межах програмно-апаратного комплексу. Покрокове представлення також дозволяє наочно продемонструвати логіку роботи системи в реальних умовах використання. Кожен етап описує певну функцію: надсилання повідомлення користувачем, отримання даних Telegram-

КбПЗ_2026

сервером, обробку інформації мікроконтролером ESP32, а також виведення повідомлення на дисплей пристрою. Такий спосіб опису є зручним для аналізу роботи системи, тестування окремих модулів і подальшого вдосконалення функціональних можливостей проєкту.

В основі системи знаходиться мікроконтролер ESP32, який виконує ключові функції обробки та передачі даних. Він підключається до мережі Інтернет через Wi-Fi і отримує інформацію з Telegram. Передача даних реалізована через HTTPS-запити до Telegram Bot API, що забезпечує захищений обмін інформацією та дозволяє працювати без розгортання додаткових серверів або складної інфраструктури. Завдяки вбудованому Wi-Fi-модулю ESP32 може самостійно встановлювати з'єднання з мережею, надсилати запити та отримувати відповіді від сервера Telegram у режимі реального часу. У результаті така архітектура значно спрощує систему, робить її більш зрозумілою, стабільною та зручною для практичної реалізації в IoT-проєктах.

Система складається з кількох основних частин, які між собою взаємодіють і утворюють єдиний робочий цикл обміну даними. Центральним елементом у цьому процесі є Telegram-сервер, який зберігає повідомлення користувачів і надає доступ до них через API. Коли користувач надсилає повідомлення, воно спочатку потрапляє на сервер, де зберігається та очікує запиту від пристрою. Таким чином реалізується асинхронна модель обміну даними, де відправлення і отримання не відбуваються одночасно. ESP32 періодично надсилає запити до Telegram Bot API, перевіряючи наявність нових повідомлень, після чого отримані дані обробляються та відображаються на дисплеї. Такий підхід дозволяє забезпечити стабільну роботу системи навіть при нестабільному мережевому з'єднанні та зменшує навантаження на пристрій.

Мікроконтролер ESP32 працює як клієнт і періодично звертається до сервера через Wi-Fi-з'єднання, яке забезпечує доступ до Інтернету. Запити виконуються до Telegram Bot API, звідки пристрій отримує відповідь у форматі JSON. Такий формат є зручним для передавання структурованих даних, оскільки дозволяє передавати не лише текст повідомлення, а й додаткову службову інформацію, зокрема час надсилання, ідентифікатори повідомлень та інші

КОПЗ–2026

параметри. Далі ці дані обробляються безпосередньо на мікроконтролері: з них виділяється текст повідомлення, службова інформація та інші необхідні параметри.

Після цього інформація готується для виводу на дисплей відповідно до заданого формату відображення, що забезпечує її структуроване та зручне сприйняття користувачем. На цьому етапі дані приводяться до єдиного вигляду, за потреби скорочуються або групуються, щоб оптимально розміститися на екрані пристрою. Окремо враховується ситуація дублювання повідомлень - у разі повторного отримання однакових даних частина з них автоматично відсіюється, що дозволяє уникнути повторного відображення та забезпечує коректність і чистоту інформаційного потоку. Такий механізм запобігає перевантаженню інтерфейсу зайвими оновленнями та підвищує стабільність відображення даних. У результаті такий підхід зменшує кількість зайвих оновлень екрана, оптимізує роботу системи та покращує зручність сприйняття інформації користувачем, особливо під час частих або повторюваних запитів.

Робота системи побудована циклічно, що означає безперервне повторення одного й того самого алгоритму через задані проміжки часу. Мікроконтролер ESP32 у такому режимі працює як клієнт, який регулярно ініціює запити до сервісу Telegram через Telegram Bot API. Спочатку пристрій формує та відправляє запит, після чого очікує відповідь від сервера, отримуючи актуальні дані у разі їх наявності. Якщо нових повідомлень у черзі немає, система завершує поточну ітерацію циклу та переходить до наступної перевірки через визначений інтервал часу. У випадку, коли повідомлення присутні, вони проходять етап обробки, після чого підготовлена інформація виводиться на дисплей пристрою. Такий підхід є досить простим і надійним для реалізації, хоча може передбачати невелику затримку між відправкою повідомлення та його отриманням, що пов’язано з періодичним характером перевірки. Водночас для даного типу IoT-системи така затримка є прийнятною і не впливає критично на загальну функціональність.

Окрім роботи з повідомленнями, пристрій має також локальні функції керування, які дозволяють взаємодіяти з ним без використання мережі.

КОПЗ_2026

Наприклад, передбачена фізична кнопка, при натисканні якої змінюється режим відображення інформації на дисплеї: користувач може переглядати поточний час і дату або інші системні дані залежно від реалізованої логіки програми. Це робить пристрій більш універсальним і зручним у повсякденному використанні. Така функція повністю реалізована на мікроконтролері ESP32 і працює автономно, незалежно від підключення до мережі Інтернет. Завдяки цьому система залишається функціональною навіть у випадках відсутності Wi-Fi-з'єднання або тимчасових збоїв у роботі серверної частини Telegram, що підвищує її надійність та практичну придатність у реальних умовах експлуатації.

Також варто врахувати енергоспоживання пристрою під час роботи. Wi-Fi модуль мікроконтролера ESP32 у активному режимі, особливо під час передачі або отримання даних, споживає значно більше енергії порівняно з режимами очікування. Найбільше навантаження на систему живлення виникає під час встановлення з'єднання з мережею та виконання HTTPS-запитів до Telegram Bot API. У зв'язку з цим для стабільної роботи системи важливо забезпечити надійне живлення без просідань напруги, оскільки нестабільне живлення може призводити до перезавантажень, втрати з'єднання або інших збоїв у роботі пристрою.

Правильна організація енергоспоживання позитивно впливає на загальну стабільність і довговічність системи, забезпечуючи коректну роботу всіх її компонентів навіть за умов змінного навантаження. Для цього можуть використовуватися якісні джерела живлення, фільтрація напруги та додаткові конденсатори для компенсації короткочасних пікових навантажень, що виникають під час передачі даних або роботи периферійних модулів.

Додатково ці аспекти доцільно враховувати при подальшій оптимізації системи, зокрема шляхом впровадження режимів енергозбереження, автоматичного вимкнення окремих модулів у періоди бездіяльності або зменшення частоти запитів до сервера. Також ефективним рішенням може бути використання адаптивних алгоритмів роботи, які змінюють інтенсивність взаємодії із сервером залежно від поточного стану пристрою та рівня активності користувача. Це дозволяє більш гнучко керувати ресурсами системи без втрати

КПЗ_2026

її функціональності.

Такий підхід дозволяє суттєво знизити загальне енергоспоживання та підвищити ефективність роботи пристрою, що є особливо важливим для IoT-систем із тривалим часом автономної роботи та обмеженими ресурсами живлення. Крім того, це позитивно впливає на стабільність роботи системи в умовах нестабільного енергопостачання.

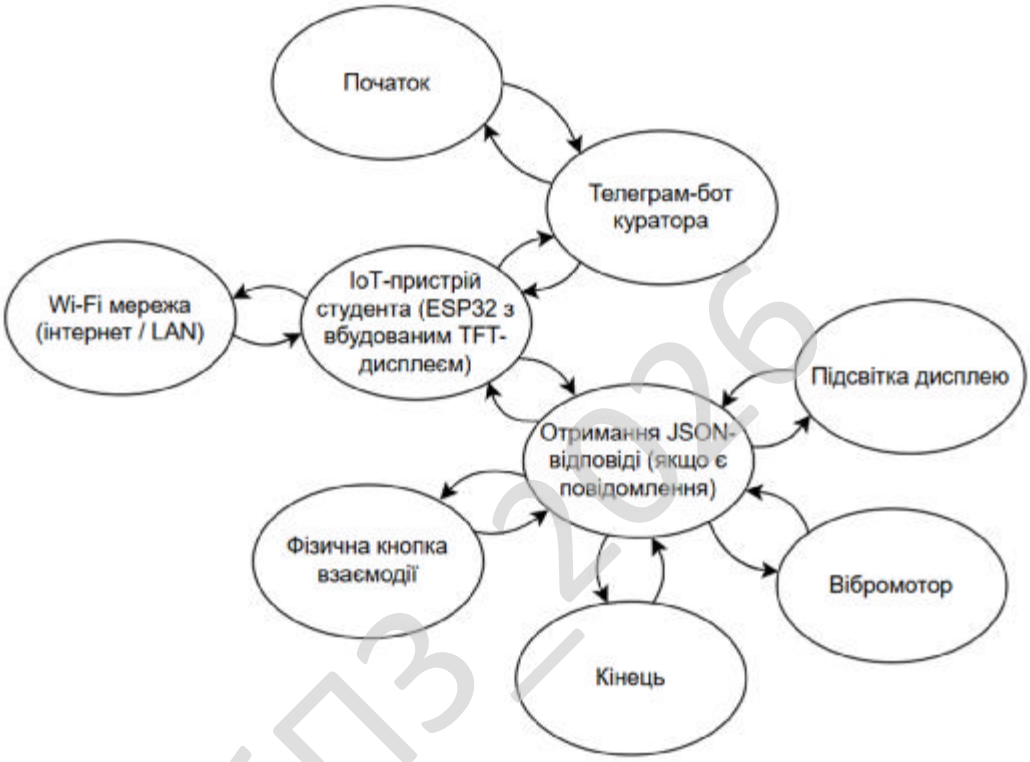


Рисунок 3.10 - Діаграма взаємодії процесів у системі Telegram-бот - IoT-смартгодинник ESP32

КПЗ 2026

На діаграмі показано загальний принцип того, як саме організований обмін даними між основними компонентами системи. Мікроконтролер ESP32 у циклічному режимі постійно звертається до сервера через мережеве з’єднання, отримує відповідь, далі виконує її обробку та формує результат для відображення на дисплеї. У цьому процесі важливу роль відіграє стабільність мережевого підключення, оскільки саме від нього залежить швидкість отримання актуальних даних і загальна реакція системи. В цілому запропонована схема є простою та логічною, без зайвих ускладнень або надлишкових проміжних компонентів.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЄКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

У межах виконання дипломного проєкту було створено IoT-систему, призначену для оперативної взаємодії між куратором і студентами із застосуванням Telegram-бота та індивідуальних пристроїв на базі мікроконтролера ESP32. Запропоноване рішення орієнтоване на швидку доставку повідомлень і зручність керування, що є актуальним для освітнього середовища, де важливо забезпечити своєчасне інформування. Архітектура системи побудована за гібридним принципом, який поєднує класичну клієнт-серверну модель, механізм періодичного опитування та асинхронну обробку подій із використанням черги повідомлень. Такий підхід дозволяє досягти балансу між швидкодією, надійністю та гнучкістю.

Система включає дві ключові складові: серверну частину у вигляді Telegram-бота, реалізованого з використанням бібліотеки aiogram, та апаратно-програмний модуль на основі ESP32 з TFT-дисплеєм. Бот виконує роль центрального елемента керування, забезпечуючи взаємодію з користувачами, обробку команд і розподіл повідомлень між кінцевими пристроями. Його функціональність охоплює керування станами користувачів за допомогою FSM, паралельну обробку запитів, ведення списків студентів і груп, а також роботу з механізмом відкладеної доставки. Наявність вбудованого HTTP-сервера дозволяє організувати взаємодію з пристроями через чергу повідомлень, що розширює можливості системи.

Обмін даними реалізовано на основі HTTP-протоколу, що спрощує інтеграцію та забезпечує сумісність між компонентами. Пристрій ESP32 виконує подвійну функцію: приймає вхідні запити як сервер і одночасно ініціює звернення до серверної частини для отримання нових повідомлень. Періодичне

Копія 2026

опитування дає змогу перевіряти наявність даних у черзі та отримувати їх навіть після тимчасової втрати зв’язку. У разі недоступності пристрою повідомлення зберігаються та доставляються пізніше, що гарантує їх отримання без втрати інформації.

Запропонована організація взаємодії підвищує стійкість системи до мережевих збоїв і забезпечує безперервність роботи в реальних умовах експлуатації. Завдяки використанню чіткої логіки обміну даними та періодичної перевірки стану з’єднання система здатна відновлювати роботу після тимчасових втрат зв’язку без критичних збоїв у функціонуванні. Це особливо важливо для IoT-рішень, де стабільність передачі даних напряму впливає на коректність відображення інформації та загальну надійність пристрою на базі ESP32. Така архітектура створює основу для подальшого розвитку системи, зокрема розширення функціоналу, підключення додаткових пристроїв або інтеграції з іншими інформаційними сервісами, включно з Telegram та іншими платформами обміну даними. При цьому зберігається цілісність і узгодженість роботи всієї системи, а також її ефективність, оскільки основні принципи взаємодії між компонентами не потребують суттєвих змін навіть при масштабуванні або модернізації.

Функціонування Telegram-бота базується на асинхронній подієвій моделі, що дозволяє обробляти запити користувачів без блокування основного циклу роботи. Після запуску бот ініціалізує необхідні структури даних, які містять інформацію про студентів, їхні ідентифікатори, а також IP-адреси або адреси пристроїв, після чого переходить у режим постійного очікування подій та команд від користувача. Такий підхід забезпечує оперативну реакцію на дії користувача та дозволяє одночасно обробляти кілька запитів без втрати даних або затримок у роботі. Взаємодія з користувачем реалізована через інлайн-меню, яке забезпечує зручний інтерфейс для вибору групи, конкретного студента або виконання групової розсилки повідомлень. Для кожного повідомлення визначається тип доставки: безпосереднє надсилання на пристрій або додавання до черги pending-повідомлень для подальшої обробки. Такий механізм дозволяє гнучко керувати потоком повідомлень, оптимізувати навантаження на систему

КПІЗ_2026

та забезпечувати коректну доставку інформації до кінцевих пристроїв на базі ESP32.

У разі недоступності пристрою повідомлення не втрачається, а зберігається у серверній черзі, реалізованій у Telegram-боті. Надалі ESP32 періодично виконує GET-запити до ендпоінта /pending, отримуючи відкладені повідомлення у форматі JSON. Після успішного отримання повідомлення воно видаляється з черги, що забезпечує одноразову доставку даних.

Робота пристрою ESP32 організована у вигляді багаторежимної системи з підтримкою енергозбереження [39]. Виділяються два основні режими: активний режим та режим глибокого сну. У активному режимі пристрій підключається до Wi-Fi мережі, синхронізує час через NTP-сервер, ініціалізує дисплей та запускає HTTP-клієнт для отримання повідомлень. У режимі сну пристрій переходить у light sleep із періодичним пробудженням за таймером або апаратною кнопкою.

Після пробудження ESP32 виконує перевірку наявності нових повідомлень через сервер черги. У разі отримання даних повідомлення відображається на TFT-дисплеї, супроводжується вібросигналом та переводить пристрій у активний режим. Якщо повідомлення відсутні, пристрій повертається у режим енергозбереження.

Окремим функціональним модулем є система синхронізації часу через NTP-сервер, яка забезпечує отримання точних значень часу з мережі та подальше їх використання для відображення актуальної дати і часу на пристрої [40]. Такий підхід дозволяє підтримувати коректність системного часу навіть після перезапуску мікроконтролера або тривалого відключення живлення, оскільки синхронізація відбувається автоматично після підключення до мережі. Додатково реалізовано обробку апаратної кнопки, підключеної до мікроконтролера ESP32, при натисканні якої користувачу виводиться окремий екран годинника з поточними часовими даними. Це дозволяє швидко отримати доступ до базової інформації без необхідності звернення до мережевих сервісів або використання Telegram. Така функціональність підвищує зручність використання пристрою та забезпечує його автономність у базових сценаріях роботи.

КОПЗ_2026

Графічне представлення алгоритму функціонування системи наведено на рисунку 4.1. Блок-схема відображає взаємодію між Telegram-ботом, сервером черги повідомлень та ESP32-пристроями, включаючи процес формування повідомлень, їх збереження, опитування пристроями та відображення на дисплеї.

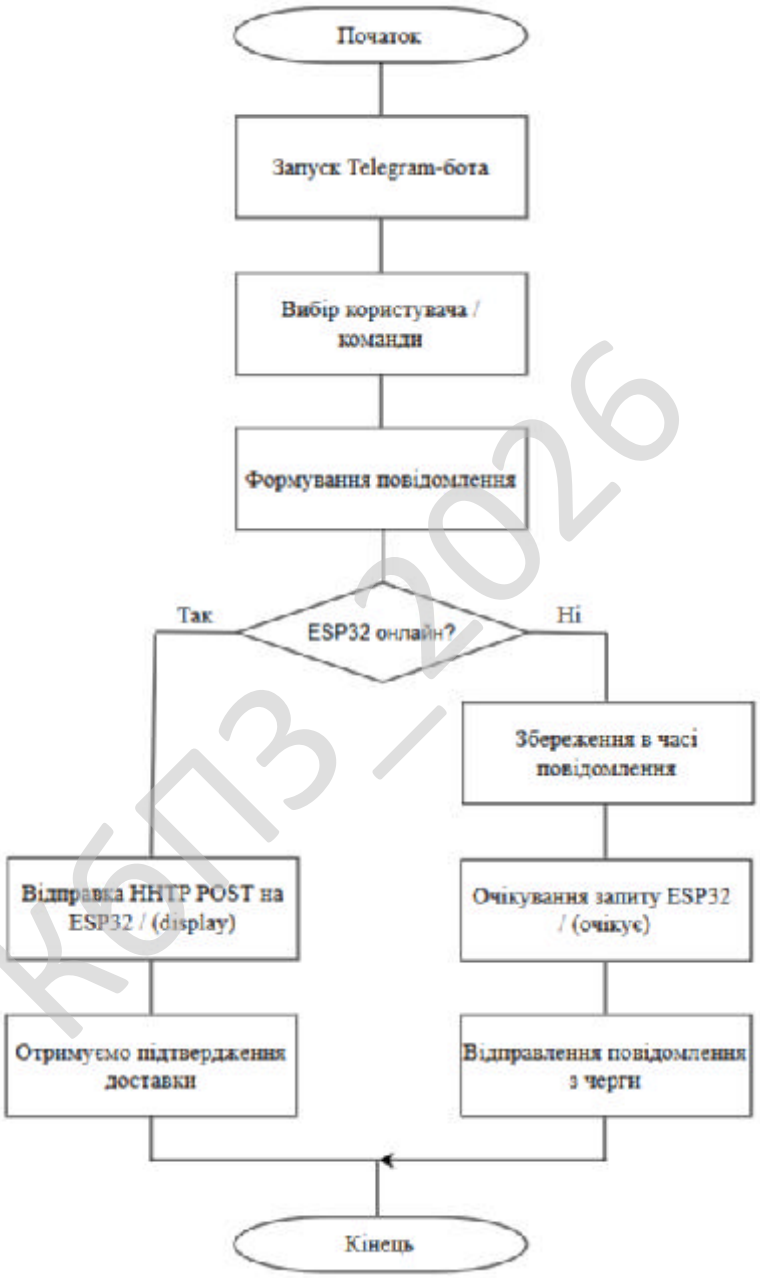


Рисунок 4.1 - Блок-схема Telegram-бота

Обмін даними між компонентами системи здійснюється через HTTP-запити із використанням JSON-формату для передачі повідомлень [41], що забезпечує уніфіковану та зручну структуру даних для обробки як на серверній,

так і на пристроєвій частині. Такий підхід спрощує взаємодію між компонентами та дозволяє ефективно працювати навіть на обмежених за ресурсами пристроях на базі ESP32. Telegram-бот у даній архітектурі виконує роль центрального координуючого вузла, який керує розсилкою даних, формує чергу повідомлень і забезпечує асинхронну взаємодію з пристроями, а також виступає посередником між користувачем і виконавчою частиною системи. Успішність доставки повідомлень контролюється за допомогою HTTP-статусів, що дозволяє відслідковувати стан запитів.

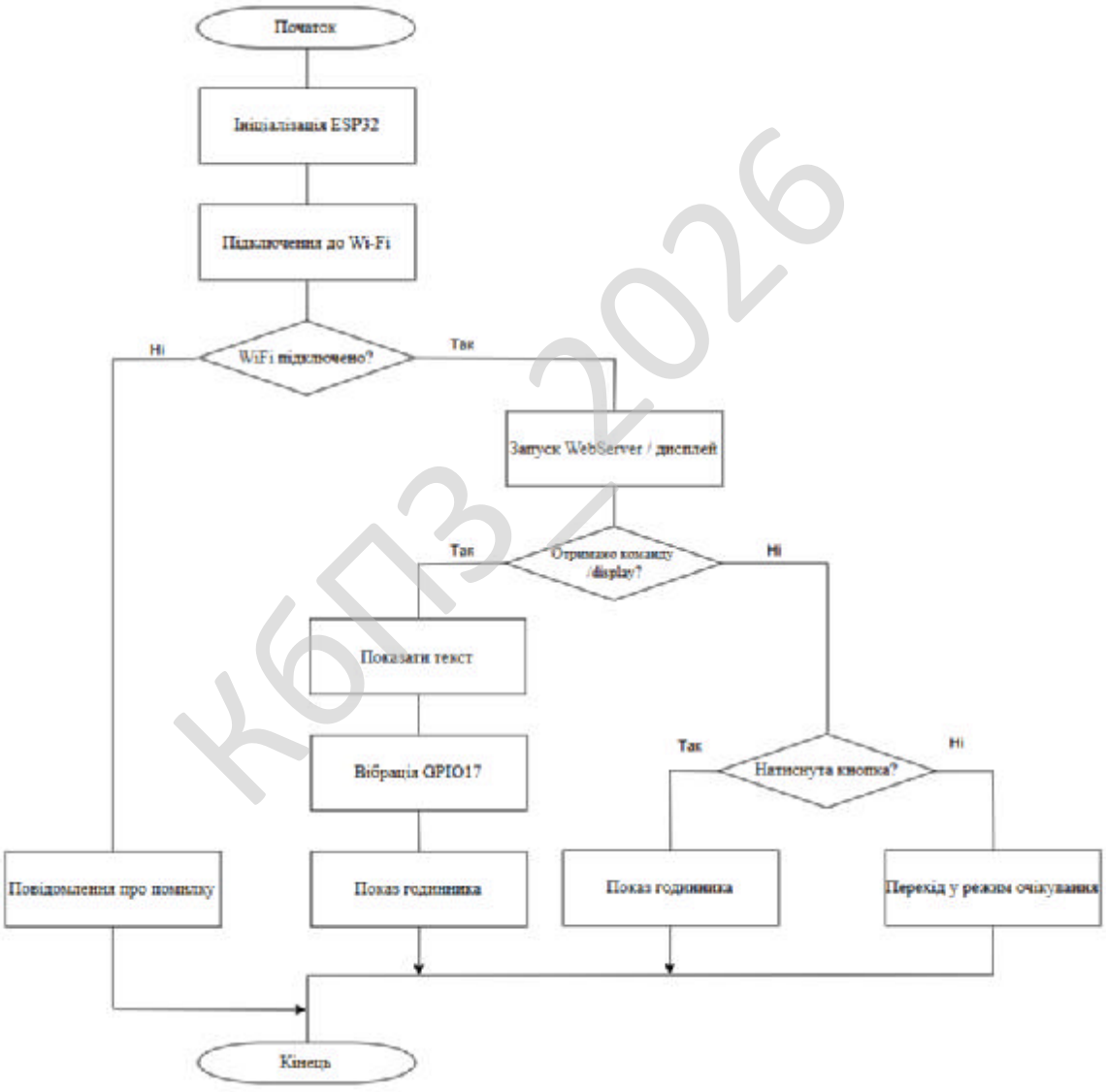


Рисунок 4.2 - Блок-схема ESP32 пристрою

На рисунку 4.3 наведено приклад відображення повідомлення на TFT-дисплеї мікроконтролера ESP32, що підтверджує коректну роботу алгоритму

КБПЗ 2026

отримання та обробки даних у реальному часі. Це демонструє успішне приймання інформації з мережі, її подальшу обробку та коректне форматування для відображення на екрані у зручному для користувача вигляді. Також результат підтверджує стабільну взаємодію системи з Telegram, де обмін даними відбувається без суттєвих затримок, втрат повідомлень або помилок у передачі, що свідчить про надійну роботу всієї архітектури.

Рисунок 4.3 - Приклад виведення повідомлення на дисплей ESP32

Запропонований алгоритм забезпечує високу надійність доставки повідомлень завдяки використанню черги повідомлень, асинхронної обробки запитів та механізму повторного опитування пристроїв. Така комбінація дозволяє мінімізувати ризики втрати даних навіть у випадках тимчасових перебоїв зв’язку або перевантаження мережі, оскільки повідомлення не обробляються одноразово, а зберігаються до моменту успішної доставки та виконання. Це особливо важливо для IoT-систем, де стабільність передачі інформації є критичним фактором коректної роботи.

Архітектура системи є масштабованою та дозволяє легко розширювати функціонал без суттєвих змін базової логіки. Зокрема, вона підтримує можливість додавання нових пристроїв на базі ESP32, підключення додаткових

КбПЗ_2026

сенсорів або впровадження нових режимів взаємодії з користувачем через Telegram. Завдяки цьому система зберігає свою гнучкість і може адаптуватися до різних сценаріїв використання без необхідності повної переробки програмної або апаратної частини.

4.2 Захист розробленого програмного забезпечення

Відповідно до Закону України «Про авторське право і суміжні права», розроблене в межах дипломного проєкту програмне забезпечення належить до об’єктів інтелектуальної власності та підлягає правовому захисту. Створена система є програмно-апаратним комплексом, що складається з серверної частини у вигляді Telegram-бота, реалізованого мовою Python із використанням асинхронного фреймворку aiogram, та клієнтської частини на базі мікроконтролера ESP32 з TFT-дисплеєм, що виконує функції приймання, обробки та візуалізації інформації.

Архітектура системи побудована за гібридним принципом взаємодії компонентів і поєднує модель клієнт-сервер та механізм опитування (polling). Telegram-бот виступає центральним керуючим вузлом, який забезпечує формування повідомлень, обробку команд користувача, управління групами студентів, а також маршрутизацію даних до відповідних пристроїв. Пристрої на базі ESP32 виконують роль кінцевих вузлів, що приймають інформацію через HTTP-запити та додатково періодично опитують сервер для отримання повідомлень із черги.

Обмін даними між компонентами реалізовано через локальну мережу WiFi із використанням HTTP-протоколу [42]. При цьому застосовано два механізми передачі інформації: пряме надсилання HTTP POST-запитів від Telegram-бота до ESP32 для оперативної доставки повідомлень, а також механізм черги повідомлень (pending_messages), який забезпечує збереження даних у випадку недоступності пристрою. У такому разі повідомлення накопичуються на сервері та доставляються пізніше через HTTP GET-запит.

Серверна частина системи реалізована з використанням асинхронної

КПІЗ 2026

моделі обробки подій. Telegram-бот підтримує інтерактивне меню на основі inline-кнопок, FSM (Finite State Machine) для керування сценаріями введення даних, а також планувальник задач APScheduler для формування періодичних повідомлень, зокрема нагадувань про події. Асинхронна реалізація дозволяє обробляти одночасні запити користувачів без блокування основного потоку виконання.

Клієнтська частина на ESP32 реалізує багаторежимну логіку роботи пристрою. Після запуску виконується підключення до Wi-Fi мережі, ініціалізація TFT-дисплея та запуск режиму активної роботи або енергозбереження. У режимі активної роботи пристрій може приймати HTTP POST-запити через вбудований веб-сервер, а також періодично здійснювати HTTP GET-запити до серверного ендпоінта /pending для отримання нових повідомлень.

Окремою особливістю є реалізація енергозберігаючого режиму light sleep, у який пристрій автоматично переходить після періоду бездіяльності для зменшення енергоспоживання. Пробудження здійснюється за таймером або натисканням апаратної кнопки. Після пробудження система виконує перевірку нових повідомлень, синхронізацію часу через NTP та повертається до активного режиму роботи.

Передача даних у системі організована у вигляді текстових повідомлень із використанням кодування UTF-8, що гарантує коректне відображення кириличних символів без спотворень [43]. Такий підхід забезпечує універсальність обміну інформацією та сумісність між усіма компонентами системи, незалежно від середовища виконання.

Для обробки структурованих даних на стороні ESP32 застосовується бібліотека ArduinoJson, яка надає зручні засоби для десеріалізації JSON-повідомлень і доступу до їх окремих полів [44]. Це дозволяє ефективно витягувати необхідні параметри, зокрема текст повідомлення, службову інформацію або ідентифікатори, та використовувати їх у подальшій логіці програми. Така організація спрощує роботу з даними і підвищує читабельність коду.

Виведення інформації на TFT-дисплей реалізовано за допомогою бібліотек

КБПЗ_2026

TFT_eSPI та U8g2_for_TFT_eSPI, які забезпечують відображення графічних елементів і тексту з підтримкою кирилиці [45]. Завдяки цьому інтерфейс залишається зрозумілим і зручним для користувача, а текстові повідомлення відображаються чітко та без втрати символів. Використані інструменти дозволяють формувати структурований екранний вміст, включаючи різні шрифти, розміри тексту та базові елементи оформлення, що покращує сприйняття інформації.

Мережевий функціонал ESP32 реалізовано з використанням бібліотек WiFi.h, HTTPClient.h та WebServer.h, які забезпечують повний цикл взаємодії з мережею: від встановлення з'єднання до обробки запитів і відповіді на них. Завдяки цьому мікроконтролер здатний не лише ініціювати HTTP-запити до зовнішніх компонентів системи, а й приймати вхідні звернення, виступаючи як локальний сервер [46]. Така організація дозволяє поєднати в одному пристрої функції клієнта і сервера, що забезпечує ефективний двосторонній обмін даними. Зокрема, ESP32 може отримувати повідомлення від Telegram-бота, обробляти їх і відображати, а також ініціювати запити до сервера для перевірки наявності нових даних у черзі.

Робота Telegram-бота побудована на асинхронній моделі обробки подій із використанням бібліотек aiogram та aiohttp, що дозволяє ефективно обслуговувати кілька запитів одночасно без блокування виконання програми. Основна логіка включає формування повідомлень як для окремих користувачів, так і для визначених груп, управління списками адресатів і гнучку маршрутизацію даних. Передача здійснюється безпосередньо за IP-адресами пристроїв у разі їх доступності або через механізм черги, якщо з'єднання тимчасово відсутнє. Такий підхід забезпечує безперервність доставки інформації навіть за нестабільних умов мережі.

Крім цього, реалізовано збереження даних про активних користувачів, що дає змогу надсилати службові повідомлення, сповіщення та нагадування без необхідності повторного визначення адресатів [47]. Це підвищує зручність керування системою та дозволяє швидко реагувати на події. У сукупності такі рішення формують гнучку й ефективну взаємодію між програмними

КБПЗ_2026

компонентами, забезпечуючи надійний обмін даними та можливість подальшого розширення функціоналу.

З позиції інформаційної безпеки розроблена система функціонує в межах локальної Wi-Fi мережі, що істотно обмежує можливості зовнішнього втручання. Відсутність прямого доступу з Інтернету знижує ймовірність атак типу сканування портів або несанкціонованого підключення. Хоча передача даних здійснюється за протоколом HTTP без використання шифрування, це рішення є виправданим у контексті ізольованого середовища експлуатації, де мережа не має публічної доступності [48]. Додатковий рівень контролю забезпечується використанням адресної моделі взаємодії: кожен пристрій ESP32 ідентифікується за унікальною IP-адресою, що дозволяє обмежити коло дозволених підключень і підвищує передбачуваність мережевої взаємодії.

Разом із цим архітектура системи не є жорстко обмеженою поточними рішеннями й передбачає можливість поступового посилення захисту. Зокрема, може бути реалізовано перехід на HTTPS із використанням TLS, що забезпечить шифрування переданих даних [49]. Доцільним також є впровадження механізмів автентифікації пристроїв, наприклад за допомогою токенів або ключів доступу, а також налаштування контролю доступу до серверних ендпоінтів. Це дозволить обмежити виконання запитів лише авторизованими клієнтами та підвищить загальний рівень довіри до системи у разі її масштабування або інтеграції з зовнішніми сервісами.

Результати тестування підтвердили працездатність програмного забезпечення в умовах, наближених до реальних [50]. Система коректно обробляє як прямі HTTP POST-запити, що надходять від Telegram-бота, так і повідомлення, отримані через механізм черги /pending. Усі сценарії доставки виконуються без помилок, включаючи ситуації з короткочасною втратою мережевого з’єднання, після чого обмін даними автоматично відновлюється. Це свідчить про достатній рівень надійності реалізованих механізмів взаємодії та обробки запитів.

ESP32 забезпечує стабільне функціонування як в активному режимі, так і під час використання режиму енергозбереження light sleep, що дозволяє

КПЗ_2026

зменшити споживання енергії без погіршення роботи ключових компонентів. Перехід між станами відбувається коректно, а після пробудження пристрій швидко відновлює виконання основних задач із повторною ініціалізацією необхідних модулів [51]. Такий підхід є ефективним для автономних IoT-рішень, де важливо раціонально використовувати ресурси.

Виведення інформації на TFT-дисплей реалізовано без помилок: повідомлення та елементи інтерфейсу відображаються чітко, а оновлення здійснюється своєчасно після надходження нових даних [52]. Інтерфейс зберігає цілісність і читабельність незалежно від тривалості роботи або циклів переходу в режим сну.

Синхронізація системного часу через NTP-сервери працює коректно, що гарантує відображення актуальної дати та годинника. Це дозволяє використовувати пристрій у задачах, де важлива точна фіксація подій і узгодженість часових параметрів.

Програмне забезпечення побудоване за модульним принципом, що спрощує його підтримку та подальший розвиток. Окремі компоненти, такі як обробка мережевих з’єднань, взаємодія з Telegram-ботом, керування дисплеєм і робота з часом, функціонують незалежно, забезпечуючи зручність внесення змін. Передача повідомлень виконується надійно завдяки механізмам повторного підключення та буферизації даних.

Система витримує короткочасні перебої зв’язку, автоматично відновлюючи з’єднання без участі користувача, що підвищує її надійність у реальних умовах експлуатації. У разі тимчасової втрати доступу до мережі пристрій на базі ESP32 продовжує роботу в локальному режимі та повторює спроби підключення до сервера після відновлення зв’язку. Такий механізм дозволяє мінімізувати вплив мережевих збоїв на загальну функціональність системи. Така архітектура також відкриває можливості для подальшого розширення функціоналу, включаючи інтеграцію додаткових модулів, сенсорів або реалізацію складніших сценаріїв взаємодії через Telegram. При цьому зберігається ефективність роботи системи, її масштабованість та адаптивність до різних умов використання.

КОПЗ_2026

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Впровадження розробленої IoT-системи, що поєднує програмний компонент у вигляді Telegram-бота та апаратний модуль на базі мікроконтролера ESP32 із TFT-дисплеєм, є логічним завершенням дипломного проєкту та передбачає поступове перенесення рішення у реальні умови експлуатації. На цьому етапі здійснюється налаштування мережевої інфраструктури, розгортання серверної частини, підключення кінцевих пристроїв і перевірка їх взаємодії в єдиному середовищі. Основна мета полягає у створенні стабільного та керованого каналу обміну повідомленнями між адміністратором системи та користувацькими пристроями в межах локальної бездротової мережі Wi-Fi, що забезпечує оперативне інформування та централізоване керування.

Процес інтеграції передбачає адаптацію системи до конкретних умов використання, включаючи конфігурацію параметрів мережі, призначення IP-адрес, формування списків користувачів і налаштування сценаріїв взаємодії. У межах цього етапу також виконується узгодження роботи апаратної частини на базі ESP32 із серверною логікою та сервісом Telegram, що забезпечує коректний обмін даними між усіма компонентами системи. Особлива увага приділяється перевірці стабільності роботи під навантаженням, коректності доставки повідомлень і відновленню функціонування після можливих збоїв, включаючи тимчасову втрату мережевого з’єднання. Такий підхід дозволяє виявити потенційні обмеження ще до повноцінного введення системи в експлуатацію та забезпечити її надійність у повсякденному використанні. Додатково це дає можливість оптимізувати параметри роботи, зменшити затримки передачі даних і підвищити загальну стабільність системи в умовах змінного навантаження та нестабільного мережевого середовища.

Запропоноване рішення має широкий спектр застосування. Окрім освітнього середовища, система може використовуватись в офісах, на підприємствах та в організаціях, де потрібно швидко передавати текстову

КОПЗ_2026

інформацію на розподілені пристрої. Завдяки використанню мікроконтролера ESP32 та інтеграції з Telegram забезпечується простий обмін даними без складної інфраструктури. Гнучкість архітектури дозволяє адаптувати систему під різні задачі, зокрема інформування персоналу, службові повідомлення та внутрішні сповіщення. Система також може бути розширена для більш складних сценаріїв, таких як моніторинг обладнання, диспетчеризація або оповіщення про критичні події в реальному часі. Завдяки модульній структурі можливе додавання нових пристроїв і сервісів без суттєвих змін базової логіки, що робить рішення придатним як для невеликих проєктів, так і для масштабованих IoT-систем.

В основі впровадження лежать принципи модульності, масштабованості та зручності інтеграції. Це дає змогу розширювати систему шляхом додавання нових пристроїв або функціональних можливостей без необхідності суттєвого перепроєктування. Завдяки такому підходу забезпечується довготривала актуальність рішення, його адаптивність до змінних умов і ефективне використання в різних сферах діяльності, включаючи освітні, промислові та побутові застосування, а також підтримку подальшого розвитку системи.

Методика впровадження передбачає реалізацію комплексу послідовних етапів, кожен з яких спрямований на забезпечення коректної роботи як програмної, так і апаратної частини системи. Загальна структура процесу впровадження включає:

- підготовку апаратно-програмного середовища;
- налаштування локальної мережі Wi-Fi;
- конфігурацію Telegram-бота та логіки взаємодії;
- прошивку та налаштування ESP32-пристроїв;
- тестування передачі даних і відображення інформації;
- інтеграцію системи в експлуатаційне середовище;
- навчання користувачів;
- моніторинг та технічний супровід;
- оновлення функціоналу та оптимізацію роботи;
- аналіз ефективності впровадження.

КПЗ_2026

1. Підготовка апаратно-програмного середовища

На початковому етапі здійснюється комплексна підготовка всіх складових системи, що забезпечують її подальше розгортання та стабільне функціонування. Апаратна частина включає мікроконтролери ESP32, оснащені TFT-дисплеями, елементи живлення та органи керування, зокрема кнопки для взаємодії з користувачем. Виконується перевірка правильності підключення компонентів, відповідності схемі та працездатності кожного елемента окремо, що дозволяє уникнути апаратних помилок на наступних етапах.

Програмна складова передбачає розгортання Telegram-бота, реалізованого мовою Python із використанням асинхронних бібліотек, а також підготовку прошивки для ESP32. Налаштовується середовище розробки, встановлюються необхідні залежності та конфігуруються параметри, пов’язані з мережею та обробкою запитів. Окремо перевіряється коректність роботи програмних модулів, що відповідають за взаємодію між компонентами системи.

Важливим аспектом є забезпечення сумісності використаних бібліотек і програмних інструментів. Проводиться тестування модулів, пов’язаних із підключенням до Wi-Fi, обробкою HTTP-запитів та відображенням інформації на дисплеї. Така перевірка дозволяє виявити можливі конфлікти або обмеження ще до початку повноцінної експлуатації та забезпечує узгоджену роботу всіх елементів системи в єдиному середовищі.

2. Налаштування локальної мережі Wi-Fi

Для повноцінного функціонування системи необхідно організувати стабільну локальну мережу Wi-Fi, у межах якої здійснюється взаємодія між серверною частиною та пристроями ESP32. На етапі налаштування визначаються параметри мережі, зокрема SSID, пароль доступу та спосіб адресації. Кожному пристрою призначається унікальна IP-адреса, що дає змогу реалізувати адресну доставку HTTP POST-запитів і забезпечити чітку ідентифікацію вузлів під час обміну даними.

Особлива увага приділяється оцінці характеристик мережі. Проводиться перевірка рівня сигналу, швидкості передачі, часу відгуку та стабільності з’єднання в різних умовах роботи. Це дозволяє виявити потенційні проблеми,

КОПЗ_2026

пов’язані з покриттям або перевантаженням, і своєчасно їх усунути. За потреби виконується оптимізація розташування точок доступу або коригування параметрів мережі.

У випадку впровадження системи в навчальному закладі доцільно використовувати окрему Wi-Fi мережу, призначену для IoT-пристроїв. Такий підхід зменшує вплив інших користувачів на якість зв’язку, підвищує стабільність роботи та спрощує адміністрування. Крім того, це створює більш контрольоване середовище для передачі даних і сприяє надійній роботі всієї системи.

3. Конфігурація Telegram-бота

Telegram-бот є центральним програмним компонентом системи, що забезпечує взаємодію користувача з пристроями ESP32. На цьому етапі виконується налаштування логіки його роботи, яка визначає принципи обробки команд та передачі даних.

Формуються структуровані списки студентів і навчальних груп, що дозволяє реалізувати адресну доставку повідомлень. На їх основі організовується механізм вибору отримувача та подальшої маршрутизації даних.

Реалізується інтерактивне меню на основі inline-кнопок, а також FSM (Finite State Machine) для послідовного введення даних і керування сценаріями взаємодії. Це підвищує зручність використання та зменшує ймовірність помилок.

КОПЗ_2026

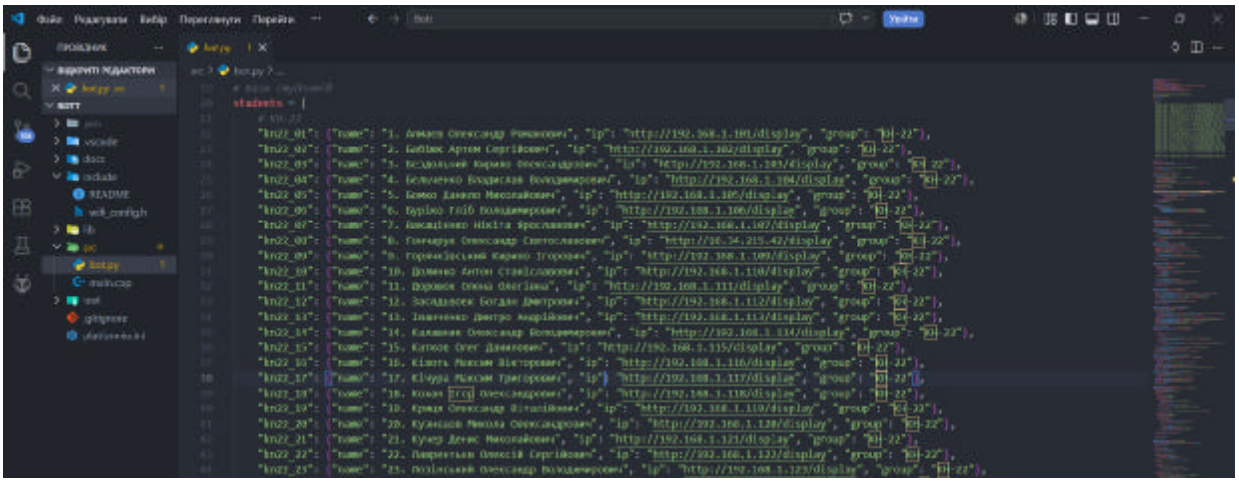


Рисунок 5.1 – Зміна налаштувань перед впровадженням

Передача повідомлень здійснюється через HTTP POST-запити до відповідних IP-адрес ESP32. Бот підтримує як індивідуальні, так і групові розсилки, а використання асинхронної моделі обробки дозволяє одночасно працювати з кількома користувачами без затримок.

4. Налаштування та програмування ESP32

Мікроконтролер ESP32 у складі системи виконує функції клієнтського вузла, відповідального за приймання, обробку та візуалізацію повідомлень. Після завантаження прошивки здійснюється початкове налаштування апаратних і програмних компонентів, що забезпечує готовність пристрою до роботи в мережевому середовищі. На цьому етапі перевіряється коректність конфігураційних параметрів, а також ініціалізуються основні модулі, необхідні для подальшого функціонування.

Першочергово виконується підключення до локальної Wi-Fi мережі із заданими параметрами доступу. Після встановлення з’єднання активується вбудований HTTP-сервер, який забезпечує приймання вхідних POST-запитів. Налаштовується обробка маршрутів, визначаються точки доступу до функціоналу пристрою, а також логіка реагування на отримані дані. Паралельно здійснюється ініціалізація TFT-дисплея, включаючи встановлення параметрів відображення, шрифтів і режимів роботи, а також конфігурація апаратної кнопки, яка може використовуватися для взаємодії або керування режимами.

Після надходження запиту пристрій виконує обробку отриманої інформації, зокрема розбір структури даних і виділення текстового вмісту. Повідомлення виводиться на дисплей у зручному для сприйняття вигляді з урахуванням форматування. Додатково реалізовано відображення поточного часу та дати, що підвищує інформативність інтерфейсу. Для цього використовується синхронізація з NTP-сервером, яка забезпечує актуальність часових даних незалежно від тривалості роботи пристрою.

У результаті виконаних налаштувань ESP32 стає повноцінним елементом системи, здатним стабільно взаємодіяти з серверною частиною, обробляти вхідні повідомлення та відображати їх у зрозумілому вигляді, забезпечуючи зручність використання та надійність роботи.

КОПЗ_2026

5. Тестування системи

Тестування є одним із ключових етапів впровадження, оскільки дозволяє оцінити працездатність усіх складових рішення в умовах, максимально наближених до реальної експлуатації. На цьому етапі виконується комплексна перевірка взаємодії між апаратною та програмною частинами, а також аналіз коректності обміну даними в різних сценаріях використання. Основна увага приділяється стабільності роботи системи, узгодженості її модулів і відсутності критичних помилок під час навантаження.

У межах тестування перевіряється доставка HTTP POST-запитів від Telegram-бота до пристрою ESP32, включаючи правильність формування, передачі та обробки повідомлень. Оцінюється точність відображення отриманої інформації на TFT-дисплеї, зокрема коректність виведення тексту, відсутність спотворень і своєчасне оновлення інтерфейсу. Додатково тестується робота інтерактивних елементів Telegram-бота, що відповідають за формування команд і керування пристроями, а також перевіряється стабільність обміну даними в умовах змінного навантаження мережі та повторних запитів.

Рисунок 5.2 - Перевірка працездатності пристрою перед експлуатацією

Окремо проводиться аналіз мережевих характеристик системи, включаючи затримки під час передачі даних, стабільність Wi-Fi-з’єднання та поведінку

КПІ – 2026

системи при змінних умовах сигналу. Такий аналіз дозволяє оцінити ефективність реалізованого механізму обміну даними між серверною частиною та пристроєм на базі ESP32, а також визначити швидкість реакції системи на надходження нових повідомлень. Додатково перевіряється коректність роботи повторних запитів і механізмів відновлення з'єднання. У результаті виявляються потенційні вузькі місця, зокрема затримки або нестабільність сигналу, та оцінюється їх вплив на продуктивність системи. Це важливий етап, який дозволяє оптимізувати роботу системи та забезпечити стабільність

Також перевіряється функціональність відображення системного часу та дати, а також коректність синхронізації з NTP-серверами. Важливим елементом є тестування відновлення роботи після короткочасних втрат з'єднання, що демонструє здатність системи автоматично відновлювати обмін даними без втручання користувача. Отримані результати підтверджують стійкість рішення до мережевих збоїв і його придатність до використання в умовах реального середовища.

6. Інтеграція в експлуатаційне середовище

Після завершення етапу тестування та підтвердження стабільної роботи всіх компонентів система переходить у фазу реального впровадження. На цьому етапі Telegram-бот стає основним каналом взаємодії між адміністратором і користувачами, забезпечуючи централізоване керування повідомленнями та їх оперативний розподіл. Одночасно пристрої на базі ESP32 розгортаються у кінцевих користувачів і виконують роль інформаційних вузлів, що відображають отримані дані в режимі реального часу.

Процес інтеграції включає налаштування робочих параметрів відповідно до конкретного середовища експлуатації, перевірку підключення пристроїв до локальної мережі та їх синхронізацію з серверною частиною. За потреби виконується адаптація списків користувачів, груп доступу та сценаріїв взаємодії, що дозволяє забезпечити коректну роботу системи в умовах реального використання, а також врахувати можливі зміни конфігурації мережі, навантаження та розширення функціоналу в майбутньому.

У навчальному середовищі запропоноване рішення може ефективно

КПІЗ_2026

застосовуватися для оперативного інформування студентів. Серед основних сценаріїв використання - передавання розкладу занять, повідомлень про зміни в навчальному процесі, нагадувань щодо подій, а також інших організаційних оголошень. Такий підхід дозволяє скоротити час доведення інформації, зменшити навантаження на адміністративний персонал і підвищити загальну ефективність комунікації в освітньому процесі.

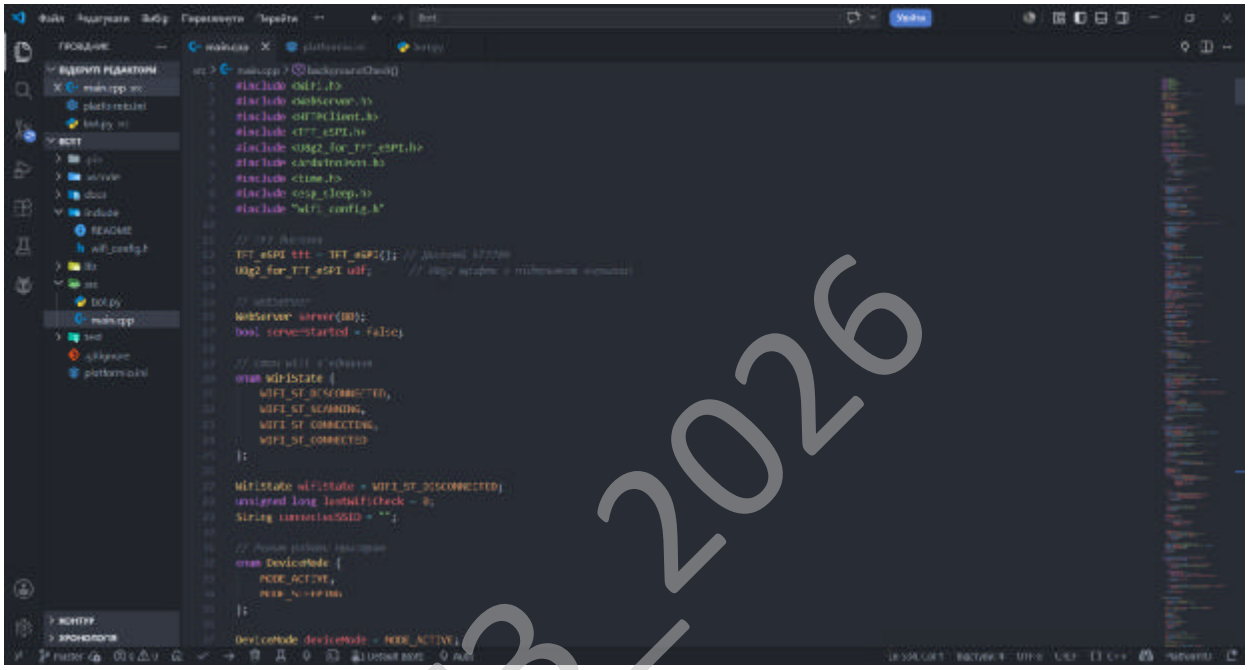


Рисунок 5.3 - Зовнішній вигляд середовища програмування Visual Studio Code з розширенням PlatformIO

КОПЗ_2026

7. Навчання користувачів

Важливою складовою впровадження системи є підготовка кінцевих користувачів до її коректного та ефективного використання. На цьому етапі проводиться ознайомлення кураторів і студентів із функціональними можливостями рішення, а також з основними принципами взаємодії між компонентами системи. Це дозволяє забезпечити швидке освоєння інтерфейсу та зменшити кількість помилок під час експлуатації.

Користувачам надаються структуровані інструкції, що охоплюють роботу з Telegram-ботом, включаючи формування та надсилання повідомлень, вибір груп або окремих адресатів, а також особливості отримання інформації в

реальному часі. Окремо пояснюється логіка маршрутизації повідомлень і принцип їх доставки до кінцевих пристроїв.

Додатково розглядається взаємодія з пристроями ESP32, зокрема їх базові функції та сценарії використання у складі системи. Користувачі отримують інформацію щодо перегляду повідомлень на TFT-дисплеї, а також відображення поточного часу та додаткових системних даних. Це дозволяє сформувати цілісне уявлення про роботу пристрою та його роль у загальній архітектурі.

Навчальний етап сприяє зниженню кількості користувацьких помилок, підвищенню рівня самостійності при роботі із системою та забезпечує більш ефективне використання її можливостей у повсякденній експлуатації.

8. Моніторинг та технічна підтримка

Після впровадження системи організовується безперервне спостереження за її функціонуванням, що дозволяє підтримувати стабільну роботу всіх компонентів у реальних умовах експлуатації. Такий підхід спрямований на своєчасне виявлення відхилень у роботі, запобігання критичним збоям і забезпечення загальної надійності рішення як на рівні апаратної, так і програмної складової.

Моніторинг охоплює контроль працездатності мікроконтролерів ESP32, включаючи перевірку їх підключення до мережі, коректність обробки запитів і стабільність виконання основних функцій. Паралельно здійснюється спостереження за роботою Telegram-бота, зокрема доступністю сервісу, швидкістю обробки команд і коректністю маршрутизації повідомлень. Додатково аналізуються лог-файли та системні журнали, що дозволяє виявляти приховані помилки або повторювані збої на ранніх етапах.

Окремим напрямом є контроль мережевої інфраструктури, включаючи стабільність Wi-Fi-з’єднання, рівень сигналу та затримки передачі даних. За необхідності виконується актуалізація конфігурацій, оновлення списків користувачів і коригування параметрів взаємодії між компонентами системи.

У випадку виникнення несправностей передбачено оперативне реагування, спрямоване на швидке усунення причин збоїв і відновлення працездатності системи. Це забезпечує безперервність її функціонування, мінімізує час простою

КПІЗ_2026

та підвищує загальний рівень надійності в умовах реальної експлуатації.

9. Оновлення та масштабування системи

Розроблена архітектура має закладений потенціал для подальшого розвитку, що дозволяє розширювати функціональність без кардинального перероблення базової структури. Це досягається завдяки модульному поділу компонентів, у якому програмні та апаратні частини працюють автономно, зберігаючи чітко визначені інтерфейси взаємодії. Такий підхід спрощує внесення змін і мінімізує ризики порушення існуючої логіки системи.

Подальше вдосконалення може включати впровадження захищених протоколів обміну даними, зокрема HTTPS, що підвищить рівень конфіденційності та цілісності інформації під час передачі. Також доцільним є додавання механізмів автентифікації пристроїв і користувачів, що дозволить обмежити доступ до системи лише авторизованим учасникам. Це особливо актуально у випадку розширення системи за межі локального середовища та використання хмарної інфраструктури.

Окремий напрям розвитку стосується функціоналу Telegram-бота, який може бути доповнений розширеними сценаріями взаємодії, гнучкими правилами маршрутизації повідомлень та інструментами автоматизації адміністративних процесів. Паралельно доцільним є впровадження системи аналітики та розширеного логування, що дозволить глибше аналізувати роботу компонентів і швидше виявляти нештатні ситуації.

Завдяки закладеній гнучкості система може масштабуватися як у межах одного навчального закладу, так і на рівні декількох груп або окремих організаційних структур. Додавання нових пристроїв не потребує суттєвих змін у логіці роботи, що робить рішення універсальним і придатним для використання в різних сценаріях. У результаті система зберігає адаптивність, розширюваність і довгострокову актуальність. Крім того, модульна структура програмного забезпечення спрощує подальше оновлення функціоналу та інтеграцію з іншими інформаційними сервісами. Це дозволяє забезпечити стабільну роботу системи навіть при збільшенні кількості користувачів або розширенні її функціональних можливостей.

КбПЗ_2026

10. Оцінка ефективності впровадження

Завершальним етапом впровадження системи є оцінка її ефективності в умовах реальної експлуатації. На цьому етапі аналізується якість роботи як програмної, так і апаратної частини, а також загальна придатність системи для практичного використання.

Основними критеріями оцінювання є час доставки повідомлень від Telegram-бота до пристроїв ESP32, стабільність роботи мікроконтролерів, а також відсоток успішно виконаних HTTP-запитів. Додатково враховується зручність використання інтерфейсу Telegram-бота та загальний рівень надійності системи в умовах тривалої роботи, а також її поведінка при змінних мережевих навантаженнях і нестабільному сигналі.

Отримані результати свідчать про достатньо високу ефективність запропонованого рішення. Система демонструє стабільну роботу, прийнятну швидкість доставки повідомлень і зручність у використанні, що підтверджує її придатність для застосування у навчальних та організаційних середовищах. Використання мікроконтролера ESP32 та інтеграції з Telegram забезпечує простий і надійний обмін даними без складної інфраструктури, що підвищує практичну цінність рішення. Також підтверджується можливість подальшого впровадження системи в реальних проєктах різного рівня складності завдяки її модульній архітектурі та потенціалу масштабування.

Запропонована методика впровадження забезпечує поетапну та контрольовану інтеграцію IoT-системи в експлуатаційне середовище. Завдяки використанню ESP32, Telegram-бота та HTTP-комунікації система характеризується простотою розгортання, високою гнучкістю та можливістю масштабування. Це робить її ефективним інструментом для організації сучасної цифрової взаємодії в навчальних закладах, а також підвищує загальну оперативність обміну інформацією між користувачами. Крім того, використання доступних апаратних і програмних компонентів дозволяє зменшити витрати на впровадження та технічне обслуговування системи. У перспективі така архітектура може бути адаптована для автоматизації інших процесів, пов’язаних із передаванням та відображенням інформації в реальному часі.

КПЗ_2026

ОСНОВНІ ВИСНОВКИ

У ході виконання дипломної роботи було спроєктовано та реалізовано IoT-систему для організації оперативної комунікації між куратором і студентами на базі мікроконтролера ESP32 та Telegram-бота. У процесі аналізу предметної області встановлено, що використання архітектури активного опитування (polling) є доцільним для роботи в умовах локальних Wi-Fi мереж із динамічними IP-адресами, оскільки воно дозволяє уникнути складних мережевих налаштувань, зокрема використання статичних адрес або зовнішніх серверів, та забезпечує простоту розгортання і супроводу системи.

Програмна частина системи реалізована з використанням мови програмування C++ на стороні ESP32 та Python із застосуванням асинхронної бібліотеки aiogram на стороні Telegram-бота. Такий підхід забезпечив логічний поділ функціональних обов'язків між клієнтською та серверною частинами, що позитивно вплинуло на продуктивність та стабільність роботи системи. Реалізований функціонал включає передачу повідомлень, обробку структурованих даних у форматі JSON, маршрутизацію запитів до конкретних пристроїв, а також синхронізацію часу через NTP-сервер, що дозволяє підтримувати актуальність інформації на пристроях у режимі реального часу, забезпечуючи зручність експлуатації та точність відображення даних.

Проведене тестування показало, що система стабільно функціонує в умовах локальної мережі та забезпечує коректну передачу повідомлень між компонентами. Підтверджено правильність обробки HTTP-запитів і відображення даних на пристроях ESP32, а також те, що час доставки повідомлень є прийнятним для використання в освітньому середовищі. Додатково встановлено, що реалізовані алгоритми дозволяють підтримувати працездатність системи навіть за умов короткочасних збоїв або нестабільності мережевого з'єднання, забезпечуючи її відновлення без критичних помилок. У результаті система продемонструвала загальну надійність і готовність до подальшого практичного використання.

КОПЗ_2026

Додатково підтверджено ефективність використання режимів енергозбереження ESP32, що дозволяє зменшити енергоспоживання та підвищити автономність пристроїв у довготривалому режимі роботи. Застосування таких режимів є особливо важливим для IoT-рішень, які функціонують без постійного живлення або працюють від акумуляторних джерел. Система демонструє достатній рівень надійності при обробці індивідуальних та групових повідомлень, а також при роботі зі структурованими даними, що передаються у форматі JSON. Це є критично важливим для мікроконтролерних рішень з обмеженими обчислювальними ресурсами, де необхідно забезпечити баланс між продуктивністю, стабільністю та енергоефективністю.

Практичне впровадження розробленої системи в навчальний процес показало її прикладну цінність, зручність використання та ефективність у реальних умовах експлуатації. Використання Telegram-бота як основного інтерфейсу взаємодії дозволило значно спростити процес обміну інформацією між куратором і студентами, підвищити оперативність комунікації та зменшити навантаження на викладача в частині підготовки та розсилки організаційних повідомлень. Додатково було відзначено підвищення швидкості реагування студентів на важливі повідомлення, що позитивно вплинуло на загальну організацію навчального процесу. У результаті система підтвердила свою ефективність і доцільність подальшого використання в подібних освітніх сценаріях.

Узагальнюючи, розроблена система поєднує просту та ефективну архітектуру, достатній рівень функціональності та надійності, що підтверджує правильність обраних технічних і програмних рішень. Отримані результати свідчать про практичну придатність системи для використання в освітніх закладах різного рівня, а також про її адаптивність до реальних умов експлуатації. Система має потенціал до подальшого розвитку шляхом розширення функціональних можливостей, підвищення рівня інформаційної безпеки та інтеграції з іншими інформаційними та освітніми платформами, що забезпечує її довгострокову актуальність і перспективність.

КБПЗ_2026

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Учасники проєктів Вікімедіа. Telegram - Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Telegram (дата звернення: 14.03.2026).

2. Типи приватних чатів у Telegram - Foxy-IT. Foxy-IT. URL:https: //www.foxy-it.com.ua/uk/blog/tipi-privatnih-chativ-u-telegram/ (дата звернення: 15.03.2026).

3. 15 фішок Telegram, завдяки яким ви полюбите месенджер ще більше. Netpeak Journal - Медіа про інтернет-маркетинг та онлайн-бізнес у деталях. URL: https://netpeak.net/uk/blog/15-fishok-telegram-zavdyaki-yakim-vi-polyubite-mesendzher-shche-bil-she/ (дата звернення: 17.03.2026).

4. iTechua. Топ-6 неприємних проблем Viber та способи їх вирішення. UKR.NET.URL: https://www.ukr.net/news/details/technologies/103821055.html (дата звернення: 20.03.2026).

5. Contributors to Wikimedia projects. Viber - Вікіпедія. Вікіпедія - вільна енциклопедія. URL: https://uk.wikipedia.org/wiki/Viber (дата звернення: 20.03.2026).

6. NEWS P. Лайфхаки: як насправді варто користуватися Slack - 8 порад, які вас здивують - ProIT. ProIT: медіа для профі в IT. URL: https://proit.ua/laifkhaki-iak-naspravdi-varto-koristuvatisia-slack-8-porad-iaki-vas-zdivuiut/ (дата звернення: 20.03.2026).

7. Slack. Освітня платформа ВЧИМО. URL: https://vchymo.com/app/application/Slack (дата звернення: 20.03.2026).

8. Чому спільноти в Discord стають популярнішими за групи в інших соцмережах. Львівський портал | Новини Львова. URL: https://portal.lviv.ua/news/2025/06/13/chomu-spilnoty-v-discord-staiut-populiarnishymy-za-hrupy-v-inshykh-sotsmerezhakh (дата звернення: 21.03.2026).

9. Усе в одному місці: як програма Discord допоможе організувати дистанційне навчання. Нова українська школа | Веб-ресурс НУШ. URL: https://nus.org.ua/2020/05/04/use-v-odnomu-mistsi-yak-programa-discord-

КПЗ-2026

dopomozhe-organizuvaty-dystantsijne-navchannya/ (дата звернення: 22.03.2026).

10. Модуль ESP32 LILYGO TTGO T-Display TFT 1.14" 16MB - Купити в Україні, Харків. 1wire.com.ua. URL: https://1wire.com.ua/modul-esp32-lilygo®-ttgo-t-display-tft-1-14-16mb.html (дата звернення: 22.03.2026).

11. Програмне «покращення» Arduino UNO до двоядерної плати. Arduino в Україні. URL: https://arduino.ua/art190-programne-pokrashhennya-arduino-uno-do-dvoyadernoi-plati?srsltid=AfmBOorfXKZIHAxR.SYpQsyOVAnn3Njltjc3vhZjeQalSjMXgTvquQU_ (дата звернення: 22.03.2026).

12. Учасники проєктів Вікімедіа. ESP32 - Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/ESP32 (дата звернення: 23.03.2026).

13. ESP32 - вступ - IT Master - електроніка та програмування. Головна - IT Master - електроніка та програмування. URL: https://itmaster.biz.ua/electronics/esp32/esp32-s2.html (дата звернення: 23.03.2026).

14. LILYGO Paxcounter LoRa V2.1_1.6.1 ESP32 868MHz. РКС Компоненти - РАДІОМАГ. URL: https://www.rcscomponents.kiev.ua/search?q=Lilygo&srsltid=AfmBOoqPYaMRJWSPdXlUP_-deWdP9AQ1G5odZxNWOdg5xvFbibJzLsay (дата звернення: 23.03.2026).

15. Reddit - Please wait for verification. Reddit - Please wait for verification. URL: https://www.reddit.com/r/arduino/comments/17swhma/esp32_ble_ultra_low_consumption_from_deep_sleep/?tl=uk (дата звернення: 24.03.2026).

16. LILYGO TTGO T-Display ESP32 WiFi і Bluetooth модуль з дисплеєм купити в Києві та Україні. Arduino в Україні. URL: https://arduino.ua/prod4360-ttgo-t-display-esp32-wifi-e-bluetooth-module-development-board?srsltid=AfmBOopXTWTSUewealDLZEhiygIjBv8xeS95o5x5eKXgHv6wSsnA9ahf (дата звернення: 24.13.2026).

17. Мікроконтролер ESP32 - IT Master - електроніка та програмування. Головна - IT Master - електроніка та програмування.URL: https://itmaster.biz.ua/directory/microcontrollers/esp32.html (дата звернення: 24.03.2026).

18. Плата захисту li-ion 18650 3.7в акумуляторів - купити недорого, Prom.ua: ціни, акції і відгуки | Україна, Київ. Prom.ua. URL: https://prom.ua/ua/Plata-zaschity-li-ion-18650-37v-akkumulyatorov.html (дата звернення: 25.03.2026).

КОПЗ_2026

19. Аккумулятор Li-Po 200мАч 3.7В формату 501240 купити в Києві та Україні. Arduino в Україні. URL: https://arduino.ua/ru/prod4573-akkymylyator-li-po-200mach-3-7v-formata-501240?srsltid=AfmBOopwb_pob11pjOEPwUQ04dewKvPIGIJjaWFH4vk5lpEWsd07qbqS (дата звернення: 26.03.2026).

20. Порівняння Arduino, ESP8266 і ESP32: яку платформу вибрати для проєкту - gazteh. gazteh. URL: https://gazteh.com.ua/porivnyannya-arduino-esp8266-i-esp32-shho-obraty-dlya-proyektu/ (дата звернення: 27.03.2026).

21. Як спілкуються мікросхеми: I2C, SPI, UART - повний гід по інтерфейсах зв'язку. Мій Проект. URL: https://myproject.com.ua/i2c-spi-uart-shcho-tse-take-i-iak-pratsiuiut-interfeisy-mikroskhem.html?srsltid=AfmBOoo4lnHOreYQRHH0T_6Mz9mRzpk3kGqn4jhNlZE9y8naSaVU8uQb (дата звернення: 28.03.2026).

22. Repository at ChSTU: Дослідження IoT-систем для автоматизації розумного будинку. Repository at ChSTU: Home. URL: https://er.chdtu.edu.ua/handle/ChSTU/6435 (дата звернення: 28.03.2026).

23. Reddit - Please wait for verification. Reddit - Please wait for verification. URL: https://www.reddit.com/r/esp32/comments/81q0al/help_understand_if_programming_esp32_in_c_is/?tl=uk (дата звернення: 03.04.2026).

24. Налаштування середовища розробки - Лілка documentation. Вітаємо вас у документації проєкту Lilka! - Лілка documentation. URL: https://docs.lilka.dev/uk/latest/programming/environment/ (дата звернення: 03.04.2026).

25. Налаштування інтеграції з Telegram-ботом | Support eSputnik. eSputnik. URL: https://docs-ua.esputnik.com/docs/nalashtuvannya-integraciyi-z-telegram-botom (дата звернення: 04.04.2026).

26. Repository of academic texts of the Ukrainian State University of Railway Transport: Home. URL: http://lib.kart.edu.ua/bitstream/123456789/28565/1/Петренко.pdf (дата звернення: 04.04.2026).

27. NodeREDGuidUKR. NodeREDGuidUKR. URL: https://pupenasan.github.io/NodeREDGuidUKR/bots/telegrambot.html (дата звернення: 04.04.2026).

28. Making sure you're not a bot!. Making sure you're not a bot!. URL: https://forum.amperka.ru/threads/Проблема-wi-fi-esp32.21565/ (дата звернення: 05.04.2026).

КОПІЯ_2026

29. ESP32 та Telegram-бот: керування пристроями через інтернет. Любительська автоматика. URL: https://geekmatic.in.ua/ua/esp32telegram-bot (дата звернення: 05.04.2026).

30. Telegram Applications. Telegram. URL: https://telegram.org/apps?setln=ru (дата звернення: 06.04.2026).

31. Long Polling vs. Webhooks | grammY. grammY. URL: https://grammy.dev/guide/deployment-types (дата звернення: 06.04.2026).

32. Введення в тестування коду в C++ / aCode. aCode. URL: https://acode.com.ua/urok-76-vvedennya-v-testuvannya-kodu/ (дата звернення: 07.04.2026).

33. Структура та характеристика складових частин інформаційних систем - Бібліотека BukLib.net. Головна - Бібліотека BukLib.net. URL: https://buklib.net/books/24045/ (дата звернення: 07.04.2026).

34. Учасники проектів Вікімедіа.Структурна схема - Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Структурна_схема (дата звернення:08.04.2026).

35. Розробка структурної схеми. StudFiles. URL: https://studfile.net/preview/7314994/page:2/ (дата звернення: 08.04.2026).

36. Учасники проектів Вікімедіа. Функціональна схема - Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Функціональна_схема (дата звернення: 09.04.2026).

37. Учасники проектів Вікімедіа. Принципова схема - Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Принципова_схема (дата звернення: 09.04.2026).

38. Купити Плата розробки TTGO T-Display ESP32 Wi-Fi та Bluetooth для Arduino з доставкою по Україні - radiostore.com.ua. Активні та пасивні електронні компоненти, паяльні станції, паяльники, блоки живлення, витратні матеріали в radiostore.com.ua. URL: https://radiostore.com.ua/p2167084634-plata-razrabotki-ttgo.html (дата звернення: 09.04.2026).

39. Посібник з мікроконтролерів ESP32. Глобальний дистриб'ютор електронних компонентів |ARIAT TECHNOLOGY LIMITED. URL: https://ua.ariat-tech.com/blog/ESP32-Microcontroller-Guide.html (дата звернення: 10.04.2026).

40. Опис налаштуваннь NTP (SNTP) клієнтів. Точний час - синхронізація

ЯбПЗ_2026

часу у мережі. URL: http://time.in.ua/setup.html (дата звернення: 12.04.2026).

41. ProgIngContrSystems. ProgIngContrSystems. URL: https://pupenasan.github.io/ProgIngContrSystems/Лекц/6_httpapi.html (дата звернення: 12.04.2026).

42. Київський професійно-педагогічний фаховий коледж імені Антона Макаренка. URL: https://kppk.com.ua/ELLIB/ebook/Gorbenko/IKT/13/13.htm (дата звернення: 12.04.2026).

43. Учасники проєктів Вікімедіа. UTF-8 - Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/UTF-8 (дата звернення: 13.04.2026).

44. BenoitBlanchon. ArduinoJson: Efficient JSON serialization for embedded C++. ArduinoJson. URL: https://arduinojson.org/ (дата звернення: 14.04.2026).

45. GitHub - Bodmer/U8g2_for_TFT_eSPI. GitHub. URL: https://github.com/Bodmer/U8g2_for_TFT_eSPI (дата звернення: 14.04.2026).

46. Arduino - Home. URL: https://www.arduino.cc/en/Reference/WiFi (дата звернення: 14.04.2026).

47. Preferences - Arduino ESP32 latest documentation. Technical Documents | Espressif Systems. URL: https://docs.espressif.com/projects/arduino-esp32/en/latest/api/preferences.html (дата звернення: 14.04.2026).

48. Ризики публічного Wi-Fi: що потрібно знати кожному користувачу. Слобідський край. URL: https://www.slk.kh.ua/novini-kompanij/cim-nebezpecnij-publicnij-wi-fi-i-ak-zahistiti-sebe.html (дата звернення: 15.04.2026).

49. ITForce. Навіщо переходити на HTTPS?. ITForce. URL: https://itforce.ua/blog/zachem-perehodit-na-https/ (дата звернення:16.04. 2026).

50. ElAr KhNURE:: Головна. URL: https://openarchive.nure.ua/ bitstreams/e09fd563-991a-4bd0-ab56-f03c68ee425f/download (дата звернення: 18.04.2026).

51. Introduction to Low Power Mode for Systemic Power Management - ESP32 - ESP-IDF Programming Guide v6.0.1 documentation. Technical Documents | Espressif Systems. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/low-power-mode/low-power-mode-soc.html (дата звернення:18.04.2026).

52. Lab6 - Wokwi ESP32, STM32, Arduino Simulator. Wokwi - World's most advanced ESP32 Simulator. URL: https://wokwi.com/projects/412356874 2190728193 (дата звернення: 19.04.2026).

КОПЗ_2026