МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка системи слідкування за сонцем з використанням інтернету речей

Здобувача вищої освіти другого (магістерського) рівня,
ІІ курсу, групи КМІ-24-2, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»

(підпис) Кульбаченка Віталія Юрійовича
(прізвище, ім'я, по батькові)

Науковий керівник (підпис) Горбовий Артур Юліанович, д-р техн. наук, професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Керівник освітньо-професійної програми (підпис) Омельчук Антон Анатолійович, канд. техн. наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту за рішенням кафедри протокол від «02» січня 2026 № 10

Завідувач кафедри (підпис) Філоненко Михайло Миколайович, канд. фіз.-мат. наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Ірпінь – 2026

Реєстрація № 4 16.01.2026 _____
(дата) (підпис відповідальної особи)

Результати перевірки до захисту 02.01.2026 _____
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 5 (відмінно) 05.02.2026 _____
(оцінка) (дата) (підпис)

Голова комісії: _____ О.С. Дюбанов
(підпис)

Члени комісії:

_____ А.Ю. Горбовий
(підпис)

_____ М.М. Філоненко
(підпис)

_____ І.Д. Погорєловська
(підпис)

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	30		05.02.2026 Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погорєловська І.Д. Суховецька Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	10		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	5		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	50		

Оцінювання кваліфікаційної роботи науковим керівником - 48 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалоюЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
98	A	5	05.02.2026	Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погорєловська І.Д. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую
Завідувач кафедри _____ Філоненко М.М.
«11» грудня 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Кульбаченку Віталію Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: розробка системи слідкування за сонцем з використанням інтернету речей.
Затверджена наказом ректора від «10» грудня 2024 № 1852
2. Строк здачі закінченої роботи «02» січня 2026
3. Мета роботи: метою кваліфікаційної роботи є розробка програмного забезпечення та програмно-орієнтованої апаратної архітектури системи слідкування за сонцем з використанням технологій Інтернету речей.
Об'єкт дослідження: процес автоматизованого керування та моніторингу положення сонячної панелі у фотоелектричних системах з використанням програмних і мережевих технологій.
Предмет дослідження: програмні засоби, алгоритми керування, методи моніторингу та програмно-апаратні рішення IoT для реалізації системи слідкування за сонцем.
Орієнтовна база дослідження: експериментальний програмно-апаратний прототип системи слідкування за сонцем на базі мікроконтролерної платформи ESP32-S3.
4. Вихідні дані до роботи: наукові публікації та навчальні матеріали з систем слідкування за сонцем; документація з програмування мікроконтролерів та IoT-платформ; технічні специфікації мікроконтролерів, сенсорів та виконавчих механізмів; програмні бібліотеки для реалізації веб-інтерфейсів, мережевих протоколів і багатозадачності.
5. Перелік графічного матеріалу: зображення апаратного блоку та його компонентів; інтерфейси веб-сторінок керування та моніторингу; приклад телеметричних даних фотоелектричної панелі; фрагменти програмного коду.
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом):
аналіз існуючих систем слідкування за сонцем та програмних підходів до їх реалізації; дослідження алгоритмів керування та сенсорної обробки даних; розробка програмно-апаратної архітектури IoT-системи; реалізація програмного забезпечення вбудованого контролера; розробка веб-інтерфейсу керування та системи моніторингу; експериментальні дослідження та аналіз результатів роботи програмного забезпечення; пропозиції щодо вдосконалення та масштабування програмної системи.

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2024	До 15.12.2024
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	30.05.2025	30.05.2025
3	**ІІ розділ** (аналіз експериментальної частини про методики досліджень)	10.09.2025	10.09.2025
4	**ІІІ розділ** (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	20.11.2025	20.11.2025
5	**Висновки**	11.12.2025	11.12.2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	25.12.2025	25.12.2025
7	Отримання відгуку керівника	28.01.2026	28.01.2026
8	Отримання зовнішньої рецензії	30.01.2026	30.01.2026

8. Дата видачі завдання «11» грудня 2024

9. Керівник кваліфікаційної роботи _____ (підпис) Горбовий А.Ю. (прізвище, ініціали)

10. Керівник освітньо-професійної програми _____ (підпис) Омельчук А.А. (прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти _____ (підпис) Кульбаченко.В.Ю. (прізвище, ініціали)

АНОТАЦІЯ

Розробка системи слідкування за сонцем
з використанням Інтернету речей.

(назва кваліфікаційної роботи)

Актуальність роботи зумовлена зростаючою роллю програмно-орієнтованих керуючих систем у сучасних енергетичних установках, зокрема у фотоелектричних системах. Підвищення ефективності використання сонячної енергії значною мірою залежить не лише від конструктивних характеристик обладнання, а й від якості програмного забезпечення, яке забезпечує автоматизоване керування, моніторинг, збір і аналіз даних, а також адаптацію системи до змінних умов експлуатації.

Сучасні системи слідкування за сонцем дедалі частіше розглядаються як програмні продукти, що функціонують у складі IoT-систем. У таких рішеннях програмне забезпечення відіграє ключову роль, визначаючи алгоритми керування, логіку взаємодії компонентів, надійність роботи, можливість віддаленого адміністрування та масштабування. Разом з тим значна частина існуючих реалізацій характеризується обмеженою гнучкістю програмної архітектури, недостатніми засобами моніторингу та складністю супроводу.

Об’єктом дослідження є програмне забезпечення керуючих IoT-систем, призначених для автоматизованого керування та моніторингу технічних об’єктів у режимі реального часу.

Предметом дослідження є архітектура, програмні модулі, алгоритми керування, мережеві сервіси та засоби моніторингу, що використовуються у вбудованих програмних системах слідкування за сонцем.

Метою кваліфікаційної роботи є розробка програмного продукту для керування та моніторингу системи слідкування за сонцем, реалізованого у вигляді автономного IoT-пристрою з вбудованим веб-інтерфейсом, підтримкою багатозадачності, віддаленого адміністрування та автоматизованого оновлення програмного забезпечення.

Для досягнення поставленої мети в роботі вирішено такі завдання:

• проаналізовано сучасні програмні підходи до реалізації систем слідкування за сонцем;

• досліджено алгоритми керування та обробки сенсорних даних з точки зору програмної реалізації;

• розроблено модульну архітектуру вбудованого програмного забезпечення;

• реалізовано веб-орієнтований інтерфейс керування без використання зовнішніх серверів;

• впроваджено механізми багатозадачності, віддаленого моніторингу та OTA-оновлення;

• проведено експериментальну перевірку стабільності та працездатності програмного продукту.

У роботі використано методи аналізу та узагальнення науково-технічних джерел, системного аналізу програмної архітектури, алгоритмічного моделювання, а також методи експериментального дослідження програмного забезпечення в умовах, наближених до реальної експлуатації.

Практичне значення роботи полягає у створенні універсального програмного продукту, який може бути використаний як основа для розробки вбудованих керуючих IoT-систем з веб-інтерфейсом, механізмами моніторингу та віддаленого адміністрування. Запропонована програмна архітектура є масштабованою та придатною для адаптації до інших задач автоматизованого керування.

Кваліфікаційна робота містить: 70 сторінок, 4 таблиці, 25 рисунків та 44 використаних джерел.

Ключові слова: сонячний трекер, IoT-система, програмне забезпечення, ESP32-S3, сенсорний модуль, веб-інтерфейс, WebSocket, OTA-оновлення, FreeRTOS, телеметрія, Prometheus, моніторинг.

Рік виконання кваліфікаційної роботи 2025-2026.

Рік захисту роботи 2026.

ABSTRACT

Development of a Solar Tracking System using the Internet of Things

(title of the qualification work)

The relevance of this work is driven by the increasing role of software-oriented control systems in modern energy installations, particularly in photovoltaic systems. Improving the efficiency of solar energy utilization depends not only on the hardware design of such systems but also on the quality of the software responsible for automated control, monitoring, data acquisition and analysis, as well as adaptation to variable operating conditions.

In contemporary solar tracking solutions, the system is increasingly viewed as a software-defined component of an IoT infrastructure. In such systems, software plays a central role, defining control algorithms, component interaction logic, operational reliability, as well as the possibilities for remote administration and system scalability. At the same time, many existing implementations remain limited by rigid software architecture, insufficient monitoring capabilities, and complex maintenance processes.

The object of the study is the software of IoT-based control systems designed for real-time automation and monitoring of technical objects.

The subject of the study is the architecture, software modules, control algorithms, network services, and monitoring mechanisms used in embedded solar-tracking systems.

The purpose of the qualification work is to develop a software product for controlling and monitoring a solar tracking system, implemented as an autonomous IoT device with an embedded web interface, multitasking support, remote administration capabilities, and OTA software updates.

To achieve this purpose, the following tasks were completed:

- an analysis of modern software approaches to implementing solar tracking systems was conducted;
- control algorithms and sensor-data processing methods were examined from the perspective of software implementation;
- a modular architecture of embedded software was designed;

- a web-based control interface was implemented without relying on external servers;
- mechanisms for multitasking, remote monitoring, and OTA updates were developed;
- experimental evaluation of system stability and software reliability was performed.

The work employs methods of scientific and technical analysis, system architecture analysis, algorithmic modelling, and experimental research of embedded software under conditions close to real operation.

The practical significance of the work lies in the creation of a universal software solution that can serve as a basis for developing embedded IoT control systems with a web interface, monitoring capabilities, and remote administration. The proposed software architecture is scalable and suitable for adaptation to various automation tasks.

The qualification work includes: 70 pages, 4 tables, 25 figures, and 44 referenced sources.

Keywords: solar tracker, IoT system, software, ESP32-S3, sensor module, web interface, WebSocket, OTA update, FreeRTOS, telemetry, Prometheus, monitoring.

Year of completion: 2025-2026.

Year of defense: 2026.

ЗМІСТ

ВСТУП **3**
РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ ТА МЕТОДИКА ДОСЛІДЖЕННЯ **6**
1.1. Системи слідкування за сонцем: класифікація, методи визначення положення та алгоритми керування класифікація 6
1.2. Сенсорні та інформаційні технології в сонячних трекерах 15
1.3. IoT у системах слідкування за сонцем та методика дослідження 20
РОЗДІЛ 2. ПРАКТИЧНІ РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ **25**
2.1. Архітектура розробленої системи 25
2.2. Розробка програмного забезпечення системи 33
2.3. Експериментальні дослідження та аналіз результатів 38
РОЗДІЛ 3. ПРОПОЗИЦІЇ ЩОДО УДОСКОНАЛЕННЯ РОЗРОБЛЕНОЇ СИСТЕМИ **43**
3.1. Напрями технічного та програмного удосконалення системи 43
3.2. Перспективи практичного використання та масштабування системи .. 45
ВИСНОВОКИ **47**
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **50**
ДОДАТКИ **55**

ВСТУП

Сучасні програмно-орієнтовані керуючі системи перебувають у стані активної трансформації, що зумовлено зростанням складності алгоритмів, підвищенням вимог до надійності, детермінованості виконання та необхідністю обробки подій у реальному часі. У цих умовах програмне забезпечення розглядається як ключовий компонент вбудованих і розподілених систем керування, що визначає їх функціональні можливості, гнучкість архітектури та здатність до інтеграції з цифровими середовищами.

Ефективність роботи таких систем істотно залежить від коректності реалізованих алгоритмів керування, точності обробки вхідних даних та здатності програмної логіки адаптуватися до змінних умов виконання.

Застосування систем автоматичного слідкування за сонцем дозволяє підвищити ефективність використання фотоелектричних панелей, проте супроводжується низкою технічних і експлуатаційних проблем, зокрема зростанням складності механічної конструкції, додатковими витратами енергії на роботу приводів та необхідністю забезпечення надійної роботи в умовах змінної освітленості й тривалої експлуатації. Особливої актуальності набуває розробка адаптивних алгоритмів керування та апаратно-програмних рішень, здатних поєднувати точність орієнтації, енергоефективність і функціональну гнучкість.

Розвиток мікроконтролерних платформ і технологій Інтернету речей (IoT) відкриває нові можливості для створення інтелектуальних систем слідкування за сонцем із підтримкою віддаленого моніторингу, збору телеметричних даних та адаптивного керування. Водночас аналіз існуючих рішень свідчить про відсутність універсальних доступних систем, які поєднували б енергоефективність, надійність і можливість масштабування. Це зумовлює актуальність теми даної магістерської роботи.

Проблематика підвищення ефективності фотоелектричних систем за рахунок використання сонячних трекерів широко представлена у працях

вітчизняних і зарубіжних науковців. Досліджуються конструктивні особливості одноосьових і двохосьових трекерів, методи визначення положення сонця на основі астрономічних моделей, сенсорні та гібридні алгоритми керування, а також питання енергоефективності й надійності таких систем.

Значну увагу приділено порівнянню модельних і сенсорних підходів до слідкування, впливу погодних умов на точність орієнтації панелей, а також вибору типів датчиків освітленості. Окремим напрямом досліджень є інтеграція сонячних трекерів у IoT-середовище з метою реалізації віддаленого моніторингу та аналізу експлуатаційних параметрів.

Разом з тим більшість промислових рішень орієнтована на використання у великих фотоелектричних станціях і характеризується високою вартістю та складністю впровадження, тоді як спрощені експериментальні реалізації часто не забезпечують достатньої точності та керованості. Це зумовлює необхідність подальших досліджень, спрямованих на створення універсальних апаратно-програмних систем.

Метою магістерської роботи є розробка та дослідження апаратно-програмної системи слідкування за сонцем з використанням мікроконтролерної платформи та технологій Інтернету речей, спрямованої на підвищення ефективності використання сонячної енергії та забезпечення можливостей моніторингу й масштабування.

Для досягнення поставленої мети в роботі передбачається вирішення таких завдань:

- систематизувати та проаналізувати існуючі підходи до побудови систем слідкування за сонцем;
- класифікувати методи визначення положення сонця та алгоритми керування сонячними трекерами;
- дослідити сенсорні та гібридні підходи до визначення напрямку сонячного випромінювання;
- обґрунтувати вибір апаратної платформи та основних електронних компонентів системи;

• розробити структуру апаратно-програмної архітектури сонячного трекера;

• реалізувати прототип системи слідкування та провести його експериментальне дослідження;

• оцінити ефективність і можливості масштабування розробленої системи.

Об’єктом дослідження є процес автоматичного слідкування за положенням сонця в фотоелектричних енергетичних системах.

Предметом дослідження є апаратно-програмні засоби та алгоритми керування сонячним трекером, що забезпечують визначення положення сонця, керування виконавчими механізмами та інтеграцію з IoT-середовищем.

У роботі використано методи теоретичного аналізу та узагальнення наукових джерел, методи класифікації й порівняльного аналізу технічних рішень, системний аналіз апаратно-програмної архітектури, алгоритмічне моделювання процесів керування, а також метод натурного експерименту з використанням фізичного прототипу сонячного трекера. Для обробки результатів експериментів застосовувалися елементи статистичного аналізу.

Базою дослідження є експериментальний прототип системи слідкування за сонцем, розроблений на основі мікроконтролерної платформи з бездротовим зв’язком, сенсорними модулями та виконавчими механізмами, що функціонує в умовах наближених до реальної експлуатації.

Практичне значення роботи полягає у можливості використання розробленої системи слідкування за сонцем у малих автономних та напівпромислових фотоелектричних установках. Отримані результати можуть бути використані для подальшого вдосконалення алгоритмів керування, розширення функцій моніторингу та масштабування системи.

Результати дослідження апробовано під час виступів на студентських наукових конференціях та відображено у наукових публікаціях, присвячених розробці систем слідкування за сонцем з використанням технологій (IoT).

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРИ ТА МЕТОДИКА ДОСЛІДЖЕННЯ

1.1. Системи слідкування за сонцем: класифікація, методи визначення положення та алгоритми керування класифікація

Сонячні системи слідкування широко застосовуються як інженерне рішення для підвищення ефективності роботи фотоелектричних установок. Їх основне призначення полягає у забезпеченні такого просторового положення сонячної панелі, за якого кут падіння сонячного випромінювання на активну поверхню є максимально наближеним до оптимального. На відміну від стаціонарних фотоелектричних систем, у яких орієнтація панелей залишається незмінною протягом усього часу експлуатації, сонячні трекери дозволяють компенсувати добові та сезонні зміни положення сонця на небосхилі, що безпосередньо впливає на величину виробленої електричної енергії [2], [3], [21], [39].

У сучасних дослідженнях сонячний трекер розглядається як складна електромеханічна система, що поєднує механічні елементи, виконавчі механізми, сенсорні вузли та програмні алгоритми керування. Ефективність такої системи визначається не лише точністю орієнтації панелі, але й стабільністю роботи в умовах змінної освітленості, рівнем енергоспоживання допоміжних приводів, надійністю апаратних компонентів та адаптивністю алгоритмів керування.

Крім того, важливим чинником є співвідношення між отриманим приростом енерговіддачі та додатковими витратами, пов’язаними з ускладненням конструкції та експлуатації системи слідкування. У ряді випадків надмірно складні механізми або алгоритми керування можуть нівелювати очікуваний ефект від застосування трекера, особливо в автономних або малопотужних енергетичних установках. Тому важливо забезпечити баланс між складністю апаратно-програмної реалізації та фактичним підвищенням енергетичної ефективності установки. [3], [16], [21].

Узагальнену класифікацію систем слідкування за сонцем за основними

ознаками наведено в таблиці 1.1 [2], [3], [39].

Таблиця 1.1

Класифікація систем слідкування за сонцем

Ознака класифікації	Тип системи	Характеристика	Переваги	Обмеження
Кількість осей обертання	Одноосьові	Обертання панелі навколо однієї осі (азимут або кут нахилу)	Проста конструкція, нижча вартість, менше енергоспоживання	Не враховують повністю сезонні зміни положення Сонця
	Двохосьові	Орієнтація за азимутом і кутом нахилу	Максимальна ефективність слідкування	Вища складність, чутливість до вітрових навантажень
Принцип керування	Відкритий контур	Астрономічна модель положення Сонця	Передбачуваність, стабільність, незалежність від засвіток	Не враховує хмарність і дифузне світло
	Замкнений контур	Керування за сигналами датчиків освітленості	Адаптація до реальних умов	Чутливість до шумів і паразитних відбиттів
	Гібридний	Поєднання модельного та сенсорного підходів	Компроміс між точністю та енергоефективністю	Вища складність алгоритмів
Тип виконавчих механізмів	Двигуни постійного струму	Приводи з редукторами	Простота керування, доступність	Потребують контролю крайніх положень
	Крокові / сервоприводи	Дискретне або позиційне керування	Вища точність позиціонування	Складніша схема, вища вартість
Сфера застосування	Промислові	Великі фотоелектричні станції	Надійність, висока ефективність	Висока ціна, закритість систем
	Напів-промислові	Приватні та автономні установки	Баланс вартості й функціональності	Обмежена гнучкість
	Аматорські	Навчальні та експериментальні системи	Простота, відкритість	Низька довготривала надійність

Джерело: [розроблено автором]

Однією з базових класифікаційних ознак є кількість осей обертання системи. За цим критерієм розрізняють одноосьові та двохосьові сонячні трекери [22], [39].

Одноосьові системи забезпечують зміну положення сонячної панелі в одній площині, найчастіше компенсуючи зміну азимутального положення сонця протягом дня або зміну кута нахилу.

Такі трекери характеризуються відносною простотою конструкції, нижчою вартістю реалізації та зменшеним енергоспоживанням виконавчих механізмів. Це робить їх доцільними для використання в автономних і малопотужних фотоелектричних установках. Водночас обмеження однією віссю обертання не дозволяє повною мірою враховувати сезонні зміни траєкторії Сонця, що зменшує потенційний приріст енерговіддачі порівняно з більш складними системами.

Двохосьові сонячні трекери забезпечують орієнтацію панелі за двома координатами - азимутом і кутом нахилу як показано на рисунку 1.1 [39].

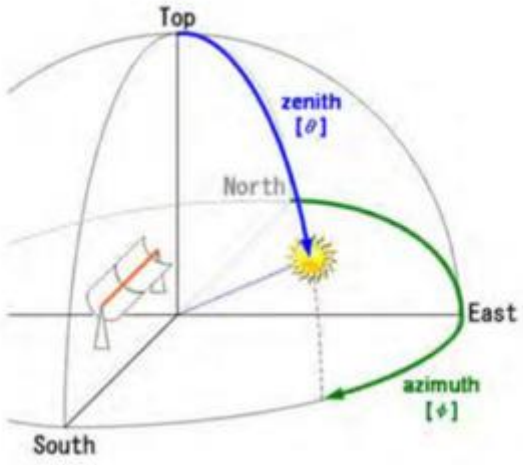


Рис. 1.1. Геометрична схема визначення положення сонця за азимутальним і зенітним кутами

Положення сонця відносно фотоелектричної панелі у двохосьових системах слідкування.

Такий підхід дозволяє підтримувати оптимальне положення фотоелектричної поверхні протягом усього дня незалежно від пори року. Двохосьові системи мають вищий потенціал збільшення виробітку

електроенергії, проте їх реалізація пов’язана зі зростанням конструктивної складності, кількості виконавчих механізмів та вимог до алгоритмів керування. Крім того, такі трекери є більш чутливими до зовнішніх механічних навантажень, зокрема вітрових, що потребує впровадження додаткових режимів захисту та підвищених вимог до надійності системи.

Іншою важливою ознакою класифікації є принцип керування системою слідкування. За цим критерієм виділяють системи з керуванням у відкритому контурі, замкненому контурі та гібридні системи. Керування у відкритому контурі базується на використанні математичних моделей руху сонця, які дозволяють визначати його положення на небосхилі за заданими часовими та географічними параметрами. Основною перевагою такого підходу є передбачуваність і незалежність від локальних оптичних завад, однак він не враховує реальні умови освітленості, зокрема хмарність і дифузне випромінювання [22], [39].

Системи із замкненим контуром керування використовують зворотний зв’язок від датчиків освітленості або електричних параметрів фотоелектричної панелі. Рішення про зміну положення панелі приймається на основі аналізу різниці сигналів між кількома сенсорами, розміщеними у визначеній геометричній конфігурації. Такий підхід дозволяє адаптувати роботу трекера до поточних умов навколишнього середовища, однак він є чутливим до дифузного світла, відбиттів та локальних джерел стороннього освітлення. У практичних реалізаціях це може призводити до нестабільної роботи та частих коригувальних рухів, що негативно впливає на енергоспоживання та довговічність механічних вузлів.

Гібридні системи слідкування поєднують модельні та сенсорні підходи до керування. У таких системах математична модель руху сонця використовується для визначення орієнтовного положення панелі, тоді як сенсорний зворотний зв’язок забезпечує корекцію локальних похибок. Подібна стратегія дозволяє зменшити кількість зайвих переміщень, підвищити стабільність роботи за змінних умов освітленості та знизити загальне енергоспоживання системи. Саме

гібридні підходи все частіше розглядаються як компромісне рішення для сучасних маломасштабних сонячних трекерів.

За типом виконавчих механізмів сонячні трекери можуть використовувати електродвигуни постійного струму, крокові двигуни, сервоприводи або лінійні актуатори. Вибір конкретного типу приводу залежить від вимог до крутного моменту, точності позиціонування, швидкості переміщення та енергоефективності. У малопотужних системах найбільш поширеними є редукторні двигуни постійного струму, які забезпечують достатній момент за відносно простого керування, однак потребують реалізації додаткових засобів обмеження ходу та контролю положення.

На сучасному етапі розвитку сонячної енергетики системи слідкування за сонцем представлені широким спектром технічних рішень, які відрізняються за принципами побудови, рівнем складності, точністю орієнтації та функціональними можливостями. Умовно такі рішення можна поділити на промислові, напівпромислові та аматорські (експериментальні), кожна з яких має свої переваги та суттєві обмеження.

Промислові системи слідкування за сонцем зазвичай характеризуються високою механічною надійністю, використанням точних приводів, промислових контролерів і складних алгоритмів керування. Такі рішення орієнтовані на роботу у складі великих фотоелектричних станцій і забезпечують значний приріст генерації електроенергії. Водночас їх недоліками є висока вартість, складність монтажу та обслуговування, а також обмежена гнучкість щодо модернізації або адаптації під специфічні умови експлуатації. Крім того, більшість промислових трекерів є закритими системами, що ускладнює інтеграцію з зовнішніми інформаційними платформами або користувацькими сервісами моніторингу.

Напівпромислові та комерційні рішення, орієнтовані на приватні або автономні установки, зазвичай поєднують спрощену механічну конструкцію з доступнішими електронними компонентами. У таких системах часто застосовуються мікроконтролери загального призначення, стандартні сенсори

освітленості та базові алгоритми керування. Основною перевагою цього класу рішень є відносно невисока вартість та можливість використання у невеликих енергетичних установках. Разом з тим вони нерідко демонструють обмежену точність позиціонування, недостатню стабільність роботи в умовах змінної освітленості та слабку адаптацію до нестандартних погодних умов.

Окрему групу становлять аматорські та експериментальні розробки, що широко представлені у наукових публікаціях, навчальних проєктах та відкритих спільнотах. Такі рішення, як правило, базуються на простих сенсорних схемах, відкритих програмних платформах і мінімальних апаратних ресурсах. Вони є цінними з точки зору дослідження принципів роботи сонячних трекерів та навчання, однак у більшості випадків не враховують довготривалу надійність, енергоспоживання, відмовостійкість та питання промислової експлуатації.

Суттєвим обмеженням значної частини існуючих рішень є недостатня інтеграція з сучасними технологіями Інтернету речей. Часто системи слідкування розглядаються ізольовано як автономні механізми керування орієнтацією панелі - без можливостей віддаленого моніторингу, збору історичних даних, аналізу ефективності або дистанційного налаштування параметрів роботи. У результаті втрачається потенціал використання накопичених даних для оптимізації алгоритмів керування та прогнозування роботи системи.

Крім того, в багатьох реалізаціях спостерігається компроміс між точністю слідкування та енергоспоживанням. Використання постійних коригувальних рухів підвищує точність орієнтації, проте призводить до зростання споживання енергії виконавчими механізмами та зниження загальної ефективності установки, особливо в автономних системах з обмеженим енергетичним балансом.

Отже, аналіз існуючих систем слідкування за сонцем свідчить про наявність низки невирішених проблем, пов’язаних із поєднанням точності позиціонування, енергоефективності, гнучкості архітектури та можливості інтеграції з IoT-платформами. Попри різноманіття реалізованих технічних

підходів, універсального рішення, придатного для всіх умов експлуатації, на сьогодні не існує.

Кожен клас сонячних трекерів має власні переваги та обмеження, а вибір конкретної архітектури повинен здійснюватися з урахуванням потужності фотоелектричної установки, умов навколишнього середовища, вимог до точності позиціонування та енергоефективності. Це обґрунтовує доцільність розробки універсальної апаратно-програмної системи сонячного трекера, яка поєднує адаптивні алгоритми керування, сучасні засоби збору й аналізу даних, а також підтримує масштабування відповідно до вимог конкретних умов експлуатації.

У зв’язку з цим виникає необхідність детального аналізу методів визначення положення сонця та алгоритмів керування.

Методи визначення положення сонця та алгоритми керування є ключовими чинниками ефективності роботи систем слідкування за сонцем. Ефективність таких систем безпосередньо залежить від точності визначення поточного положення світила та адекватності алгоритмів керування, які формують керувальні впливи на виконавчі механізми.

У сучасних сонячних трекерах застосовуються різні підходи до визначення орієнтації сонця, що відрізняються за складністю реалізації, точністю, чутливістю до зовнішніх умов і вимогами до апаратних ресурсів. Узагальнено ці підходи можна поділити на модельні, сенсорні та комбіновані (гібридні).

Модельні методи ґрунтуються на використанні астрономічних залежностей, які описують рух сонця відносно поверхні землі. Положення сонця визначається у вигляді азимутального та зенітного (або кутового) положення на основі поточного часу, дати та географічних координат місця встановлення системи. Подібні алгоритми дозволяють з високою точністю прогнозувати траєкторію руху Сонця протягом доби та року.

Основною перевагою модельних методів є їх передбачуваність і незалежність від локальних умов освітленості. На відміну від сенсорних систем, вони не реагують на короткочасні зміни, спричинені хмарністю, відбиттями або сторонніми джерелами світла. Це робить модельні алгоритми особливо

привабливими для систем, у яких важлива стабільність керування та мінімізація кількості коригувальних рухів.

Водночас використання модельних методів висуває підвищені вимоги до точності часових даних і коректного задання географічних координат. Похибки синхронізації часу, некоректний облік часових поясів або сезонного переведення годинника можуть призводити до систематичних помилок орієнтації. Крім того, модельні методи не враховують реальний стан атмосфери, зокрема дифузне випромінювання, що може знижувати ефективність слідкування за умов значної хмарності.

Сенсорні методи базуються на безпосередньому вимірюванні освітленості за допомогою світлочутливих елементів, таких як фоторезистори, фотодіоди або фототранзистори. Найбільш поширеним підходом є використання декількох сенсорів, розміщених у симетричній конфігурації, що дозволяє визначати напрямок на сонце за різницею сигналів між ними.

Перевагою сенсорних методів є їх здатність адаптуватися до поточних умов навколишнього середовища. Система реагує не на теоретичне положення сонця, а на фактичний розподіл світлового потоку, що дозволяє компенсувати часткове затінення або нестандартні умови освітлення. Це особливо актуально для автономних систем, що працюють у змінному середовищі.

Разом з тим сенсорні методи мають низку суттєвих обмежень. За умов суцільної хмарності або переважання дифузного світла різниця сигналів між сенсорами може бути недостатньою для формування стабільного керувального рішення. Крім того, система стає чутливою до паразитних відбиттів, локальних джерел освітлення та шумів вимірювального тракту. У практичних реалізаціях це може призводити до так званого “полювання” - частих дрібних переміщень панелі без істотного приросту корисної енергії.

Для зменшення впливу зазначених факторів у сенсорних системах застосовуються різні алгоритмічні прийоми, зокрема фільтрація сигналів, введення порогів нечутливості, часові затримки та обмеження мінімальної амплітуди коригувального руху. Застосування таких методів дозволяє

підвищити стабільність роботи системи. Важливу роль відіграє також конструктивне виконання сенсорного вузла, яке розглядається в наступному підрозділі.

Гібридні методи поєднують переваги модельних і сенсорних підходів. У таких системах астрономічна модель використовується для визначення орієнтовного положення сонця, тоді як сенсорний зворотний зв’язок застосовується для уточнення та корекції положення панелі. Це дозволяє зменшити залежність системи від нестабільних умов освітлення та одночасно зберегти адаптивність до локальних відхилень.

Як правило, гібрідний алгоритм, реалізуються у вигляді багаторежимних стратегій, де система може перемикатися між модельним і сенсорним режимами залежно від зовнішніх умов або внутрішніх критеріїв. Наприклад, за стабільної освітленості використовується сенсорний режим, тоді як за хмарної погоди або низького рівня сигналу система переходить до модельного керування. Такий підхід дозволяє знизити кількість зайвих переміщень і підвищити загальну енергоефективність системи.

Алгоритми керування сонячними трекерами формують логіку взаємодії між блоком визначення положення сонця та виконавчими механізмами. Найпростішим підходом є порогове керування, за якого рух панелі здійснюється лише у разі перевищення заданої різниці між виміряними або розрахованими параметрами. Такий алгоритм є простим у реалізації та не потребує значних обчислювальних ресурсів, проте може бути недостатньо гнучким для складних умов експлуатації [2], [3], [22].

Більш складні алгоритми передбачають використання регуляторів, що враховують динаміку системи, швидкість зміни сигналів і стан виконавчих механізмів. У практичних реалізаціях важливе значення має обмеження кількості та тривалості рухів, оскільки надмірна активність приводів призводить до зростання енергоспоживання та зносу механічних компонентів. Тому сучасні алгоритми керування часто орієнтовані не на максимальну миттєву точність, а на досягнення компромісу між точністю орієнтації та енергетичною доцільністю

рухів трекера.

Окрему увагу приділяють алгоритмам безпечного керування, які враховують крайні положення механізмів, процедури ініціалізації після запуску та аварійні режими роботи. Реалізація таких алгоритмів є необхідною умовою для забезпечення надійної та довготривалої експлуатації сонячного трекера, особливо в автономних системах [2], [10], [12], [19], [39].

Таким чином, вибір методу визначення положення сонця та відповідного алгоритму керування є ключовим етапом проєктування системи слідкування. Кожен з розглянутих підходів має власні переваги та обмеження, а їх комбінування дозволяє досягти більш збалансованих характеристик. Подальший розгляд датчиків освітленості та конструктивних рішень сенсорних вузлів дозволяє детальніше оцінити практичні аспекти реалізації таких систем.

1.2. Сенсорні та інформаційні технології в сонячних трекерах

Датчики освітленості є ключовим елементом сенсорних і гібридних систем слідкування за сонцем, оскільки саме вони забезпечують отримання інформації про напрямок і інтенсивність світлового потоку. Якість роботи сонячного трекера значною мірою залежить від характеристик світлочутливих елементів, способу їх розміщення та конструктивного виконання оптичної частини сенсорного вузла. Неправильний вибір або реалізація датчиків може призвести до нестабільної роботи системи, підвищеного енергоспоживання та зниження загальної ефективності слідкування [1], [14], [30], [37], [41].

У практиці побудови сонячних трекерів найбільш поширеними є датчики освітленості на основі фоторезисторів, фотодіодів, фототранзисторів, а також, які використовуються як чутливі елементи для визначення напрямку та інтенсивності сонячного випромінювання.

Фоторезистори характеризуються простотою використання та низькою вартістю, що робить їх привабливими для маломасштабних і експериментальних систем [30].

Принцип їх роботи ґрунтується на зміні електричного опору залежно від

рівня освітленості. Водночас фоторезистори мають низку обмежень, серед яких нелінійна характеристика, значний температурний дрейф і відносно повільна реакція на зміну світлового потоку. Ці фактори необхідно враховувати під час розробки алгоритмів обробки сигналів.

Фотодіоди та фототранзистори, як показано на рисунку 1.2 [37,41].

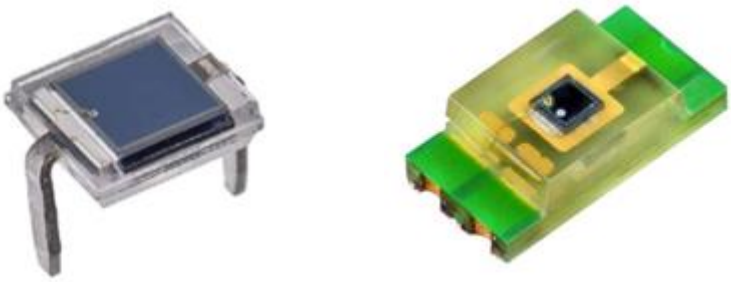


Рис. 1.2. Фотодіод і фототранзистор як світлочутливі елементи сенсорної системи

Вони дозволяють більш точно вимірювати зміни освітленості та ефективніше реалізовувати алгоритми фільтрації і нормалізації сигналів. Разом з тим використання напівпровідникових світлочутливих елементів часто потребує складнішої схеми узгодження та підсилення сигналу, що збільшує апаратну складність сенсорного вузла.

Окрему групу світлочутливих елементів, що застосовуються в системах слідкування за сонцем, становлять малі сонячні панелі, які використовуються як сенсори освітленості.

Принцип їх роботи ґрунтується на вимірюванні вихідної напруги або струму, що генерується панеллю під дією сонячного випромінювання. На відміну від класичних світлочутливих елементів, такі панелі безпосередньо перетворюють світлову енергію в електричну, що дозволяє одночасно використовувати їх як джерело сигналу та, в окремих випадках, як допоміжне джерело живлення для сенсорного вузла.

Перевагою малих сонячних панелей є простота інтеграції, відсутність потреби у спеціалізованих підсилювальних каскадах та висока стійкість до температурних впливів і старіння. Крім того, вихідний сигнал таких панелей добре корелює з реальним енергетичним потоком, що падає на поверхню

фотоелектричного модуля, що може бути корисним для оцінювання загальної ефективності системи [18].

Разом з тим використання малих сонячних панелей як датчиків освітленості має низку обмежень. Основним недоліком є низька кутова чутливість, оскільки вихідний сигнал змінюється повільніше порівняно з фотодіодами або фототранзисторами. Це ускладнює точне визначення напрямку на сонце та може призводити до інерційності системи слідкування, особливо за умов слабкої або дифузної освітленості. У зв’язку з цим малі сонячні панелі доцільно використовувати переважно в простих або допоміжних схемах керування, а також у поєднанні з іншими сенсорними елементами в гібридних системах.

Додатковим фактором, що обмежує застосування малих сонячних панелей як основних датчиків положення, є їх залежність від спектрального складу випромінювання та температурного режиму роботи. За змінних атмосферних умов, зокрема при частковій хмарності або під час сходу й заходу сонця, вихідні параметри панелі можуть змінюватися нерівномірно, що ускладнює інтерпретацію сенсорних даних.

З огляду на зазначені особливості, використання малих сонячних панелей як сенсорів доцільно розглядати як допоміжне або комплементарне рішення, яке ефективно поєднується з фотодіодами або фототранзисторами в рамках гібридних алгоритмів слідкування за сонцем.

Фоторезистори вирізняються простотою реалізації та низькою вартістю, однак їх нелінійна характеристика, значний температурний дрейф і повільна реакція на зміну освітленості обмежують можливість застосування в системах, що потребують стабільної та передбачуваної роботи.

Фотодіоди забезпечують високу точність і швидкодію, проте формують відносно малий вихідний сигнал, що зумовлює необхідність використання додаткових підсилювальних каскадів та ускладнює апаратну частину сенсорного вузла. Це є небажаним для компактних і енергоефективних вбудованих систем.

Малі сонячні панелі можуть використовуватися як сенсорні елементи у

простих системах, однак їх інерційність та низька точність визначення напрямку світлового потоку не дозволяють реалізувати стабільне позиціонування з необхідною роздільною здатністю.

Наведений порівняльний аналіз світлочутливих елементів в таблиці 1.2, свідчить, що кожен із розглянутих типів сенсорів має як переваги, так і обмеження, які визначають доцільність їх використання у системах слідкування за сонцем [1], [30], [37], [41].

Таблиця 1.2

Порівняльні характеристики світлочутливих сенсорів, що застосовуються в сонячних трекерах

Тип сенсора	Чутливість	Переваги	Недоліки
Фоторезистор	Середня	Простота використання, низька вартість	Нелінійність, температурний дрейф, повільна реакція
Фотодіод	Висока	Висока точність і швидкодія	Потребує підсилення сигналу
Фототранзистор	Висока	Висока чутливість, добрий компроміс між точністю та складністю	Обмежена лінійність у широкому діапазоні освітленості
Малі сонячні панелі	Низька–середня	Проста інтеграція, можливість одночасного живлення	Низька точність визначення напрямку, інерційність

Джерело: [розроблено автором]

З урахуванням наведених факторів у розроблюваній системі слідкування за сонцем обрано фототранзистор як компромісне рішення між чутливістю, простотою апаратної реалізації та стабільністю роботи в умовах змінної освітленості. Фототранзистори забезпечують достатній рівень вихідного сигналу без застосування складних схем підсилення та дозволяють ефективно реалізувати алгоритми фільтрації й нормалізації сигналів у вбудованому програмному забезпеченні.

Окрім вибору типу світлочутливого елемента, важливу роль відіграє геометрія розміщення датчиків. У сенсорних сонячних трекерах зазвичай

використовується кілька датчиків, розташованих симетрично відносно центральної осі. Така конфігурація дозволяє визначати напрямок на сонце за різницею сигналів між окремими каналами. Найпоширенішим рішенням є використання чотирьох датчиків, розміщених у вигляді квадрантів, що дає змогу формувати незалежні сигнали для керування за двома координатами.

Для підвищення чутливості до напрямку світлового потоку та зменшення впливу дифузного випромінювання застосовуються оптичні перегородки між датчиками. Таке конструктивне рішення дозволяє створити зону часткового затінення, в якій невелике відхилення напрямку світла викликає помітну різницю сигналів між сенсорами. Подібна конфігурація підвищує точність визначення напрямку на сонце за ясних погодних умов, проте за умов суцільної хмарності ефект затінення зменшується, що може призводити до зниження інформативності сенсорних сигналів.

Додатковим елементом оптичної частини сенсорного вузла можуть бути куполи або розсіювачі світла, які встановлюються над датчиками. Їх застосування дозволяє згладити різкі перепади освітленості та частково зменшити вплив локальних відбиттів. Водночас надмірне розсіювання світла може знизити контраст між сигналами окремих сенсорів, що негативно впливає на чутливість системи до напрямку світлового потоку. Таким чином, вибір та параметри розсіювача повинні узгоджуватися з алгоритмами керування та характеристиками застосованих датчиків.

Суттєвою проблемою сенсорних систем слідкування є вплив дифузного світла та паразитних засвіток. За умов хмарної погоди світловий потік надходить до сенсорів практично з усіх напрямків, унаслідок чого різниця сигналів між окремими датчиками зменшується. У таких режимах система може втрачати чітке уявлення про напрямок на сонце, що призводить до нестабільної роботи або до повної відмови від коригувальних рухів. Аналогічні проблеми можуть виникати внаслідок відбиттів від навколишніх поверхонь або впливу штучних джерел світла.

Для зменшення негативного впливу зазначених факторів у практичних

реалізаціях використовуються як конструктивні, так і алгоритмічні засоби. До конструктивних рішень належать оптимізація висоти та ширини перегородок, використання матових поверхонь і обмеження поля зору сенсорів. Алгоритмічні засоби включають фільтрацію сигналів, введення зон нечутливості, усереднення вимірювань у часі та обмеження частоти коригувальних рухів. Комплексне застосування цих підходів дозволяє підвищити стабільність роботи трекера без істотного ускладнення апаратної частини.

Таким чином, вибір датчиків освітленості та оптичних рішень є критичним етапом проєктування сонячного трекера. Ефективна сенсорна система повинна забезпечувати достатню чутливість до напрямку світлового потоку, водночас залишаючись стійкою до змін погодних умов і зовнішніх завад. Отримані висновки створюють основу для розгляду інтеграції сенсорних вузлів у системи з віддаленим моніторингом і керуванням, що розглядається в наступному підрозділі.

1.3. IoT у системах слідкування за сонцем та методика дослідження

Інтернет речей у системах слідкування за сонцем є одним із ключових напрямів розвитку сучасних енергетичних технологій. Розвиток концепції Інтернету речей (IoT) суттєво вплинув на підходи до проєктування сучасних енергетичних систем, зокрема систем слідкування за сонцем.

Інтеграція сонячних трекерів у IoT-середовище дозволяє перейти від локальних автономних рішень до розподілених інтелектуальних систем із можливістю віддаленого моніторингу, аналізу та адаптивного керування. У цьому контексті сонячний трекер розглядається не лише як механічний пристрій, а як повноцінний вузол кіберфізичної системи.

Ключовою особливістю IoT-орієнтованих систем слідкування є наявність мережевого з’єднання, яке забезпечує обмін даними між трекером, серверною інфраструктурою та користувацькими інтерфейсами. Завдяки цьому стає можливим збір інформації про поточне положення панелей, рівень освітленості, споживання енергії, стан виконавчих механізмів і загальну працездатність

системи. Отримані дані можуть зберігатися, аналізуватися в реальному часі або використовуватися для побудови статистичних і прогнозних моделей [1], [12], [17].

Однією з основних переваг використання IoT у сонячних трекерах є віддалений моніторинг і діагностика. Оператор системи отримує змогу контролювати роботу трекера без фізичної присутності на об’єкті, що особливо актуально для віддалених або важкодоступних локацій. Виявлення відхилень у роботі приводів, сенсорів чи системи живлення на ранніх етапах дозволяє зменшити час простою та підвищити загальну надійність енергетичної установки.

IoT-підхід також відкриває можливості для адаптивного керування. На основі зібраних даних система може автоматично змінювати режими роботи залежно від погодних умов, часу доби або технічного стану обладнання. Наприклад, за тривалої хмарності трекер може переходити в енергозберігаючий режим із мінімальною кількістю коригувальних рухів, а за сильного вітру - фіксувати панелі в безпечному положенні. Такі рішення підвищують не лише ефективність, але й довговічність механічних компонентів.

Важливим аспектом IoT-інтеграції є вибір протоколів обміну даними та архітектури системи. У практичних реалізаціях можуть використовуватися як локальні бездротові мережі, так і підключення до глобальних мереж передачі даних. Архітектура зазвичай включає периферійний рівень (датчики та контролери), рівень обробки даних та користувацький рівень, на якому реалізуються інтерфейси візуалізації та керування. Така багаторівнева структура забезпечує масштабованість і гнучкість системи.

Разом з перевагами IoT-системи слідкування за сонцем мають і певні обмеження. До них належать залежність від якості мережевого з’єднання, підвищені вимоги до енергоспоживання електронних компонентів і питання інформаційної безпеки. Передача даних і команд керування через мережу потребує захисту від несанкціонованого доступу, що ускладнює програмну частину системи та вимагає застосування відповідних механізмів автентифікації

й шифрування.

Порівняльну характеристику традиційних та IoT-орієнтованих систем слідкування за сонцем наведено в таблиці 1.3 [2], [7], [11,] [14], [17].

Таблиця 1.3

Порівняння традиційних та IoT-орієнтованих систем

Критерій	**Традиційні системи**	**IoT-орієнтовані системи**
Архітектура	Локальна, автономна	Розподілена, мережева
Моніторинг	Локальний або відсутній	Віддалений, у реальному часі
Збір даних	Обмежений	Централізований, зберігання історії
Адаптивність	Обмежена	Висока, з урахуванням умов середовища
Діагностика	Переважно ручна	Автоматизована
Масштабованість	Низька	Висока
Інформаційна безпека	Не актуальна	Потребує захисту та шифрування
Енергоспоживання	Менше	Дещо більше через мережеві модулі

Джерело: [розроблено автором]

Незважаючи на зазначені виклики, використання Інтернету речей у системах слідкування за сонцем є перспективним напрямом розвитку. IoT-рішення забезпечують підвищення керованості, прозорості та адаптивності роботи сонячних трекерів, що особливо важливо для сучасних децентралізованих енергетичних систем. У поєднанні з сенсорними та алгоритмічними підходами, розглянутими в попередніх підрозділах, IoT формує основу для створення інтелектуальних систем слідкування з розширеним функціоналом.

Методика дослідження, застосована в даній роботі, базується на комплексному поєднанні теоретичних, аналітичних та експериментальних підходів, що забезпечує всебічне вивчення принципів побудови та функціонування систем слідкування за сонцем. Такий підхід дозволяє не лише узагальнити наявні наукові напрацювання, а й отримати власні практичні результати, спрямовані на підвищення ефективності сонячної енергетичної

установки.

На початковому етапі дослідження було використано методи теоретичного аналізу та узагальнення, що передбачали опрацювання наукових публікацій, монографій, статей у фахових виданнях, матеріалів наукових конференцій та відкритих технічних джерел. У межах цього етапу здійснено критичний аналіз існуючих підходів до побудови сонячних трекерів, методів визначення положення сонця та алгоритмів керування, що дозволило сформувати концептуальні засади розроблюваної системи.

Для систематизації технічних рішень та порівняння різних класів сонячних трекерів застосовано метод класифікації та порівняльного аналізу. Оцінювання здійснювалося за такими критеріями, як точність слідкування, енергоспоживання, складність апаратної реалізації, адаптивність до змінних умов освітлення та можливості інтеграції з інформаційними системами. Це дало змогу виявити ключові обмеження існуючих рішень і обґрунтувати вибір архітектури власної розробки.

На етапі проєктування та реалізації системи використано методи системного аналізу та інженерного моделювання. Архітектура сонячного трекера розглядалася як сукупність взаємопов’язаних апаратних і програмних компонентів, що функціонують у межах єдиного керованого процесу. Для розробки алгоритмів керування застосовано елементи алгоритмічного аналізу, а також методи логічного моделювання процесів орієнтації сонячної панелі.

Експериментальна частина дослідження базувалася на методі натурного експерименту з використанням фізичного прототипу сонячного трекера. У процесі експлуатаційних випробувань здійснювалося спостереження за поведінкою системи в різних умовах освітленості, аналіз точності позиціонування панелі та оцінювання стабільності роботи алгоритмів керування. Отримані експериментальні дані оброблялися з використанням елементів статистичного аналізу та порівнювалися з теоретичними очікуваннями.

Інструментарій дослідження включав апаратні та програмні засоби. До апаратної частини належали мікроконтролерні платформи, сенсори освітленості,

виконавчі механізми, модулі живлення та комунікаційні інтерфейси. Програмний інструментарій охоплював середовища розробки вбудованого програмного забезпечення, бібліотеки для роботи з сенсорами та мережевими протоколами, а також засоби візуалізації та аналізу даних у межах IoT-платформи.

Джерельну базу дослідження становили наукові та навчальні публікації підтверджувальні сертифікати та документи показано на рисунках Додатку А, з проблематики сонячної енергетики, автоматизованих систем керування та Інтернету речей, нормативно-технічна документація на електронні компоненти, технічні описи й специфікації обладнання, а також результати власних експериментальних спостережень. Окрему групу джерел склали електронні ресурси та офіційна документація виробників апаратних і програмних платформ, що використовувалися під час розробки системи.

Таким чином, обрана методика дослідження, інструментарій та джерела інформації забезпечили наукову обґрунтованість і практичну спрямованість отриманих результатів, а також створили цілісну основу для реалізації та подальшого експериментального аналізу розробленої системи слідкування за сонцем. Такий підхід дав змогу об’єктивно оцінити роботу апаратної частини, перевірити коректність алгоритмів керування та підтвердити ефективність застосованих інженерних рішень. У наступному розділі детально розглянуто архітектуру системи, її функціональні можливості та результати практичних випробувань, що дозволяє пов’язати теоретичні положення з реальними параметрами роботи прототипу.

Крім того, отримані результати можуть бути використані як основа для подальшої оптимізації системи та її адаптації до різних конфігурацій сонячних установок. Це створює передумови для масштабування розробленого рішення та впровадження його у реальних енергетичних системах з підвищеними вимогами до надійності й ефективності.

РОЗДІЛ 2. ПРАКТИЧНІ РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

2.1. Архітектура розробленої системи

Архітектура розробленої системи слідкування за сонцем орієнтована на створення повноцінної вбудованої апаратно-програмної платформи, придатної для тривалої автономної експлуатації, масштабування та подальшого практичного використання. Система розглядається як кіберфізичний об’єкт, у якому механічні, електронні та програмні компоненти поєднані єдиною логікою керування, контролю стану та віддаленого моніторингу.

На відміну від типових аматорських рішень, архітектура системи проєктувалася з урахуванням реальних умов експлуатації, зокрема можливих збоїв живлення, необхідності дистанційного адміністрування, оновлення програмного забезпечення та контролю версій. Основною метою є формування універсальної платформи для промислового або напівпромислового використання сонячних трекерів із можливістю подальшої модернізації та розширення функціональних можливостей.

На початковому етапі розробки система була реалізована у вигляді експериментального макета, що дозволяв перевірити базову взаємодію сенсорів, алгоритму визначення напрямку освітлення та поведінку механізмів позиціонування. Такий підхід дав змогу оцінити працездатність основних компонентів ще до переходу до складніших апаратних конфігурацій, рисунок Б.2.1.

Подальше вдосконалення апаратної частини включало оптимізацію розміщення датчиків, покращення механічної конструкції та модернізацію електронних модулів, що забезпечило підвищення точності орієнтації та стабільності системи. На цьому етапі було уточнено конструктивні параметри механізмів, а також протестовано декілька варіантів сенсорної частини, рисунок Б.2.2.

Фінальна версія конструкції отримала захищену електронну частину, удосконалений механізм слідкування та систему датчиків, адаптовану до роботи

в умовах низьких температур та змінної інтенсивності сонячного випромінювання. Остаточний варіант забезпечує підвищену надійність, точність позиціонування та готовність до довготривалої автономної експлуатації, рисунок Б.2.3.

Додатково використання захисного корпусу дозволяє реалізувати впорядковану внутрішню компоновку електронних компонентів, що спрощує монтаж, обслуговування та подальшу модернізацію системи. Корпус виконує не лише захисну, а й організаційну функцію, забезпечуючи фіксацію модулів, розділення логічної та силової частин, а також коректне прокладання сигнальних і силових ліній. Такий підхід зменшує ризик випадкових механічних пошкоджень, коротких замикань і впливу електромагнітних завад. Крім того, модульна реалізація керуючого блоку дозволяє адаптувати систему до різних конфігурацій сонячних установок шляхом заміни або додавання окремих апаратних компонентів без зміни загальної архітектури.

Центральним елементом апаратної частини системи є мікроконтролер ESP32-S3, який виконує функції керування, обробки даних та мережевої взаємодії як показано на рисунку 2.4 [23], [24].

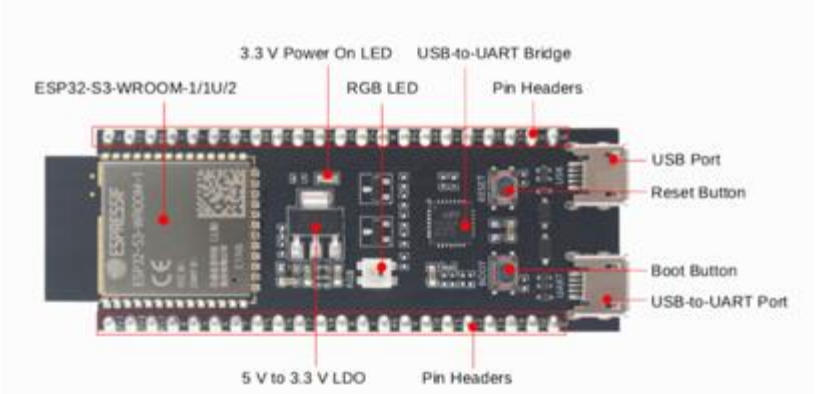


Рис. 2.4. Мікроконтролерний модуль ESP32-S3-DevKitC-1

Мікроконтролерний модуль ESP32-S3-DevKitC-1 з основними апаратними компонентами, зокрема модулем ESP32-S3-WROOM, інтерфейсами USB-UART, кнопками керування, світлодіодною індикацією та виводами для підключення зовнішніх пристроїв.

Для реалізації керуючого модуля системи слідкування за сонцем розглядалися різні мікроконтролерні платформи, які потенційно можуть використовуватися в подібних вбудованих системах. Зокрема, було проаналізовано платформи Arduino (на базі ATmega), STM32 та ESP32-S3. Порівняльні характеристики зазначених платформ наведено в таблиці 2.1 [10], [11], [12], [23]

Таблиця 2.1

Порівняння мікроконтролерних платформ

Критерій	**Arduino (ATmega)**	**STM32**	**ESP32-S3**
Обчислювальна потужність	Низька	Висока	Висока
Підтримка FreeRTOS	Обмежена	Так	Так
Вбудований Wi-Fi	Ні	Ні*	Так
OTA-оновлення	Ні	Обмежено	Так
Підтримка веб-інтерфейсу	Обмежена	Так	Так
Придатність для IoT	Низька	Середня	Висока

Джерело: [розроблено автором]

Як видно з таблиці 2.1, платформа ESP32-S3 забезпечує оптимальне поєднання обчислювальної потужності, можливостей паралельного виконання задач керування, стабільної роботи бездротового зв’язку Wi-Fi та підтримки оновлення програмного забезпечення «по повітрю» (OTA). Крім того, мікроконтролер підтримує роботу з зовнішньою оперативною пам’яттю PSRAM і файловими системами, що створює основу для реалізації складної багатозадачної логіки керування сонячним трекером без залучення додаткових обчислювальних вузлів або серверної інфраструктури.

Разом з тим мікроконтролер ESP32-S3 є лише одним із компонентів апаратної архітектури системи. Реалізація повноцінного функціонування сонячного трекера забезпечується сукупністю взаємопов’язаних апаратних модулів, розміщених у межах керуючого корпусу. До їх складу входять драйвери керування виконавчими механізмами, модулі живлення, комутаційні елементи, а також апаратні датчики положення.

Взаємодія зазначених модулів організована таким чином, щоб забезпечити надійну та безпечну роботу системи в умовах змінних навантажень і зовнішніх впливів. Кожен апаратний компонент виконує чітко визначену функцію в загальній структурі керування, що спрощує діагностику, технічне обслуговування та локалізацію можливих несправностей. Такий підхід відповідає принципам модульної побудови вбудованих систем і створює передумови для подальшого розширення або модернізації апаратної частини без порушення загальної архітектури.

У системі реалізовано два незалежні канали керування рухом:

• вертикальний канал (UP–DOWN), який керує лінійним актуатором і забезпечує зміну кута нахилу сонячної панелі;

• горизонтальний канал (LEFT–RIGHT), який керує колекторним електродвигуном постійного струму з редуктором та відповідає за азимутальне позиціонування панелі.

У межах корпусу розміщено мікроконтролер ESP32-S3, драйвери колекторних двигунів типу BTS7960, розрахованих на струм до 43 А, модулі живлення, клемні з’єднання та елементи внутрішньої комутації. Застосування драйверів цього типу забезпечує значний запас по потужності, стійкість до пікових навантажень та надійну роботу в умовах змінного механічного опору.

Драйвер BTS7960 підтримує реверсивне керування двигунами та регулювання швидкості за допомогою широтно-імпульсної модуляції. Це дозволяє реалізувати плавні профілі руху, зменшити ударні навантаження на механічні вузли та підвищити загальний ресурс системи. Така компоновка забезпечує електричну розв’язку логічної та силової частин системи, зменшує вплив імпульсних завад від електроприводів і підвищує загальну надійність роботи установки.

Загальний вигляд апаратної частини розробленої системи слідкування за сонцем, включаючи керуючий модуль у корпусі з розміщеними електронними компонентами, як показано на рисунку 2.5 [розроблено автором]. Конструктивне виконання модуля забезпечує компактність, та захист електронних компонентів.



Рис. 2.5. Загальний вигляд апаратної частини розробленої системи слідкування за сонцем (керуючий модуль у корпусі)

Логіка керування приводами та обробки подій у системі реалізована з використанням механізмів багатозадачності FreeRTOS. Основні функціональні процеси винесені в окремі задачі, що дозволяє уникати блокування системи, забезпечувати коректну реакцію на асинхронні події та безпечно завершувати операції переміщення. Такий підхід є критично важливим для системи, яка одночасно працює з виконавчими механізмами, кінцевими вимикачами, мережевими сервісами та системами моніторингу. Використання FreeRTOS дозволяє реалізувати передбачувану часову поведінку системи та чітке розмежування відповідальності між окремими програмними модулями. Для синхронізації роботи задач та зменшення навантаження на процесор застосовуються керовані затримки виконання, що дозволяє забезпечити

стабільну роботу системи без надмірного споживання обчислювальних ресурсів [10], [11], [12], [19].

Особливе місце в програмній архітектурі займає алгоритм ініціалізації та визначення початкового положення механізмів - Homing. Після кожного перезавантаження або відновлення живлення система автоматично визначає поточний стан приводів, виконує пошук базової HOME-позиції у двох напрямках та коректно обробляє ситуації, у яких базове положення не може бути знайдене. Результат виконання процедури Homing фіксується на програмному рівні та використовується як відправна точка для подальшої логіки керування, що дозволяє уникнути втрати координат і зменшити ризик механічних пошкоджень.

Важливою складовою програмного забезпечення є початкова сторінка керування системою, яка виступає основним інтерфейсом взаємодії користувача з сонячним трекером. Головне меню реалізоване у вигляді веб-сторінки, що завантажується безпосередньо з вбудованого HTTP-сервера мікроконтролера та доступна через стандартний веб-браузер без потреби встановлення додаткового програмного забезпечення.

Інтерфейс керування побудований із використанням технологій HTML та JavaScript і зберігається у файловій системі LittleFS. Такий підхід дозволяє відокремити логіку користувацького інтерфейсу від основної прошивки та змінювати зовнішній вигляд або поведінку сторінки без повного перепрошивання мікроконтролера. Це є важливим з погляду зручності експлуатації та подальшої модернізації системи [29].

Головна сторінка керування виконує роль центральної панелі оператора, через яку здійснюється:

- перегляд поточного режиму роботи системи;
- контроль стану ініціалізації (Homing);
- відображення активних напрямків руху;
- доступ до службової інформації та телеметричних даних;
- базове керування режимами роботи трекера;
- моніторинг поточних значень сенсорів та стану виконавчих механізмів.

Інтерфейс орієнтований на наочність і простоту використання, що дозволяє застосовувати систему як у сервісному режимі, так і під час повсякденної експлуатації. Оновлення інформації на сторінці відбувається без перезавантаження всієї сторінки, що підвищує зручність взаємодії та зменшує навантаження на мережеві ресурси мікроконтролера.

Реалізація веб-інтерфейсу безпосередньо на вбудованому контролері дозволяє розглядати систему слідкування за сонцем як самодостатній IoT-пристрій, готовий до інтеграції в локальні мережі без використання зовнішніх серверів або хмарних сервісів на початковому етапі експлуатації.

Програмне забезпечення підтримує два основні режими роботи: ручний режим керування та автоматичний режим позиціонування. У ручному режимі оператор має можливість безпосередньо керувати приводами, що використовується для налагодження, тестування та сервісного обслуговування системи. Автоматичний режим реалізує алгоритми слідкування за сонцем відповідно до принципів, розглянутих у попередніх розділах. Поточний режим роботи системи доступний для зовнішнього моніторингу, що дозволяє дистанційно контролювати поведінку трекера та оперативно виявляти некоректні сценарії.

Для взаємодії з користувачем і зовнішніми системами в програмному забезпеченні реалізовано вбудований HTTP-сервер із базовими механізмами автентифікації. Через веб-інтерфейс доступні технічні сторінки, точки доступу до телеметричних даних і службової інформації, а також окремі керуючі дії. Це дозволяє інтегрувати контролер у локальні або корпоративні мережі без використання додаткових шлюзів чи серверів.

Маємо чітке розділення прошивки та файлової системи. Для зберігання даних, використовується файлова система LittleFS, яка оновлюється незалежно від основної прошивки. Такий підхід дозволяє оновлювати програмний код без втрати даних та, навпаки, змінювати вміст файлової системи без перевстановлення прошивки. Підтримка окремих OTA-оновлень для прошивки та файлової системи суттєво підвищує гнучкість експлуатації системи та

спрощує процес супроводу.

Система моніторингу та збору метрик є важливою складовою архітектури програмного забезпечення та експортує телеметричні дані у форматі, сумісному з Prometheus, що дозволяє здійснювати подальшу візуалізацію та аналіз у середовищі Grafana. Доступні метрики охоплюють ідентифікацію пристрою, режими роботи, стан руху, результати процедури Homing, параметри мережевого з'єднання, використання оперативної пам'яті та PSRAM, стан Flash-пам'яті, файлових систем і температурні показники мікроконтролера. Це забезпечує повну спостережуваність системи в режимі реального часу [23], [12], [28], [42].

Для контролю версій і відстеження змін кожна збірка прошивки та файлової системи має унікальні ідентифікаційні параметри, зокрема номер версії, час складання, хеш репозиторію та контрольні суми. Збереження інформації про активну версію файлової системи дозволяє однозначно визначати, які дані використовуються системою в поточний момент, та уникати помилок під час оновлень.

Інтеграція розробленої системи слідкування за сонцем з IoT-платформою спрямована на забезпечення віддаленого моніторингу, підвищення керованості та прозорості роботи системи, а також створення умов для подальшого аналізу й оптимізації алгоритмів керування. Контролер функціонує як автономний мережевий вузол, що спрощує архітектуру системи та зменшує кількість потенційних точок відмови.

Таким чином, розроблене програмне забезпечення реалізує комплексну, надійну та масштабовану архітектуру керування сонячним трекером, що поєднує багатозадачність, адаптивну логіку позиціонування, віддалене адміністрування, автоматизоване оновлення та розвинені механізми моніторингу. Це створює міцну основу для інтеграції системи з IoT-інфраструктурою та подальшого розвитку функціональних можливостей, що розглядається у наступному підрозділі. У сукупності такі рішення підвищують ефективність експлуатації трекера та забезпечують стабільну роботу.

2.2. Розробка програмного забезпечення системи

Програмне забезпечення системи слідкування за сонцем реалізоване як вбудована багатомодульна програма для мікроконтролера ESP32-S3 та орієнтоване на безперервну роботу в автономному режимі з можливістю віддаленого керування, моніторингу та оновлення. Програмна реалізація побудована з урахуванням обмежених апаратних ресурсів, вимог до детермінованої поведінки приводів і необхідності одночасної роботи керуючої логіки, мережевих сервісів та сервісних підсистем.

Основним засобом взаємодії користувача з системою є вбудований веб-інтерфейс, який формується та обслуговується безпосередньо мікроконтролером ESP32-S3 і доступний через стандартний веб-браузер. Головна сторінка керування генерується програмно на рівні прошивки у вигляді HTML-документа та передається клієнту HTTP-сервером у відповідь на запит до кореневого маршруту [36].

Програмна генерація головного інтерфейсу керування дозволяє жорстко пов’язати критичні елементи керування з логікою прошивки та уникнути ситуацій, за яких помилки файлової системи або некоректне оновлення статичних ресурсів можуть призвести до втрати доступу до базових функцій керування. Такий підхід підвищує стійкість системи до збоїв, оскільки ключові елементи інтерфейсу створюються безпосередньо у пам’яті контролера та не залежать від стану зовнішніх ресурсів. Це особливо важливо для автономних систем, що працюють без постійного технічного супроводу.

Реалізація інтерфейсу на основі HTML та JavaScript забезпечує доступність і універсальність рішень: керування може здійснюватися через будь-який сучасний браузер без необхідності встановлення додаткового програмного забезпечення. У поєднанні з асинхронними механізмами обміну даними це дозволяє організувати оперативний моніторинг роботи системи, стану приводів, виконання алгоритму Homing та параметрів навколишнього середовища у реальному часі, забезпечуючи зручність взаємодії оператора з керуючим пристроєм.

Веб-інтерфейс, як показано на рисунку 2.6 [розроблено автором], забезпечує доступ до основних функцій керування режимами роботи, ручного керування приводами, аварійної зупинки та сервісних можливостей системи.

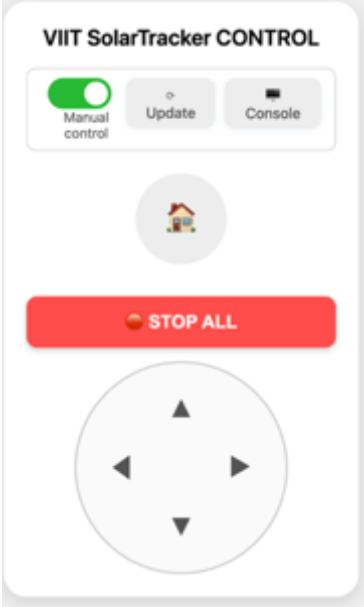


Рис. 2.7. Головна панель веб-інтерфейсу керування системою

Для забезпечення надійності та незалежності інтерфейсу код головної веб-сторінки інтегровано безпосередньо в програмну прошивку мікроконтролера, що дозволяє зберігати доступ до базових функцій керування навіть у разі збоїв або некоректної роботи файлової системи. Такий підхід дозволяє мінімізувати залежність від зовнішніх ресурсів, зменшити складність файлової системи та забезпечити гарантовану доступність основного інтерфейсу керування одразу після запуску контролера, незалежно від стану файлових розділів.

Головна сторінка керування виконує функцію центральної панелі оператора та забезпечує:

- відображення поточного режиму роботи системи (ручний або автоматичний);
- перемикання між ручним та автоматичним режимами керування;
- керування рухом приводів у ручному режимі;
- доступ до сервісних сторінок та веб-консолі;
- перегляд основних телеметричних показників у реальному часі;
- аварійну зупинку всіх виконавчих механізмів (STOP ALL).

Оновлення інформації на сторінці виконується динамічно без перезавантаження всієї сторінки, що зменшує мережеве навантаження та підвищує зручність експлуатації.

Для обробки запитів веб-інтерфейсу у системі використовується асинхронний HTTP-сервер. Асинхронна модель дозволяє обслуговувати запити без блокування основного циклу виконання та без впливу на часову поведінку керуючої логіки.

Серверна частина надає набір HTTP-маршрутів, через які реалізується:

- отримання поточного стану системи;
- керування напрямками руху приводів;
- перемикання режимів роботи;
- запуск процедури ініціалізації (Homing);
- виконання службових дій (перезавантаження, OTA-оновлення, доступ до метрик).

Кожен запит проходить перевірку автентифікації, що обмежує доступ до керуючих функцій та підвищує загальний рівень безпеки системи.

Веб-консоль забезпечує інтерактивний доступ до команд керування, перегляду журналів подій та виконання сервісних і діагностичних операцій у режимі реального часу.

Для цього в програмному забезпеченні реалізовано власний механізм логування з буферизацією повідомлень у пам’яті та синхронізацією їх із веб-інтерфейсом. Командний інтерпретатор обробляє текстові команди, розділяючи їх на категорії керування системою, файловою системою та виконавчими механізмами.

Окремою складовою програмного забезпечення є вбудована веб-консоль, реалізована з використанням WebSocket-з’єднання, інтерфейс як показано на рисунку 2.8 [розроблено автором]. Вона забезпечує інтерактивний обмін даними з контролером у режимі реального часу, що суттєво підвищує зручність діагностики та оперативного керування системою. Такий механізм дозволяє отримувати актуальні службові повідомлення та технічні параметри [36].



Рис. 2.8. Лістинг командного рядка керування та діагностики

Програмне забезпечення структуроване у вигляді окремих модулів, як показано на рисунках Додатку В кожен з яких відповідає за конкретну функціональну підсистему: керування приводами, обробку кінцевих вимикачів, мережеву взаємодію, зберігання даних, моніторинг, оновлення програмного забезпечення та ініціалізацію механізмів.

Програмний код файлу (main.cpp), який представлений в рисунках Додатку Г, реалізує базову логіку роботи контролера та координує взаємодію між окремими підсистемами.

Такий підхід дозволяє ізолювати логіку окремих компонентів, спростити тестування та зменшити взаємні залежності між частинами системи. Модулі взаємодіють між собою через спільні структури стану та чітко визначені інтерфейси.

Для процесів, що потребують фонового виконання та точного керування часом, у системі використовується операційна система реального часу FreeRTOS. При цьому FreeRTOS не застосовується для всієї логіки системи, а використовується вибірково — лише для критичних керуючих задач.

У програмному забезпеченні реалізовано окремі FreeRTOS-задачі для:

- виконання процедури ініціалізації механізмів (Homing);
- постійного моніторингу стану системи.

Задача Homing запускається у фоновому режимі та виконує поетапний алгоритм пошуку базового положення з використанням кінцевих вимикачів. Між етапами руху застосовуються керовані затримки, що дозволяє стабілізувати механічні вузли та уникнути блокування інших підсистем.

Задача моніторингу працює паралельно та забезпечує регулярне оновлення внутрішніх параметрів стану, які використовуються як для логіки керування, так і для експорту телеметричних даних.

Програмне забезпечення підтримує два основні режими роботи:

- ручний режим, у якому оператор безпосередньо керує приводами через веб-інтерфейс або консоль;
- автоматичний режим, у якому керування здійснюється відповідно до алгоритмів слідкування за сонцем.

Перемикання режимів виконується програмно, з обов’язковим завершенням активних рухів, що виключає конфлікт між ручним і автоматичним керуванням.

У системі реалізовано універсальний механізм OTA-оновлення, який підтримує незалежне оновлення основної прошивки та файлової системи. Процес оновлення супроводжується автентифікацією, контролем цілісності та автоматичним перезавантаженням контролера після завершення операції.

Кожна збірка прошивки та файлової системи має унікальні ідентифікаційні параметри, включно з номером версії, часом складання та контрольними сумами, що дозволяє однозначно визначати активну конфігурацію системи.

Для забезпечення спостережуваності система експортує телеметричні дані у форматі, сумісному з Prometheus, що дає змогу здійснювати повноцінний моніторинг роботи контролера у реальному часі. Набір метрик охоплює режими роботи системи, стан і напрям руху приводів, результати виконання процедури Homing, параметри мережевого з’єднання, рівень використання оперативної пам’яті та PSRAM, заповненість файлових систем, а також температурні характеристики мікроконтролера. Такий підхід дозволяє не лише своєчасно виявляти можливі відхилення у роботі, а й інтегрувати систему в зовнішні

аналітичні інструменти, зокрема Grafana, для побудови інформаційних панелей та аналізу довгострокових трендів.

Метрики як показано на рисунку 2.20 [розроблено автором], охоплюють режими роботи, стан руху, результати процедури Homing, параметри мережі, використання пам’яті, стан файлових систем та температурні характеристики мікроконтролера.



Рис. 2.20. Метрики експорту телеметричних системи

Такий підхід дозволяє аналізувати поведінку системи в часі, виявляти аномалії та оцінювати стабільність роботи без втручання у виконання основної керуючої логіки.

2.3. Експериментальні дослідження та аналіз результатів

Експериментальні дослідження розробленої системи слідкування за сонцем проводилися з метою перевірки працездатності апаратно-програмної архітектури, оцінювання коректності реалізованих алгоритмів керування та аналізу стабільності роботи системи в умовах, наближених до реальної експлуатації. Дослідження охоплювали як функціональні випробування окремих підсистем, так і комплексну перевірку роботи системи в цілому з урахуванням впливу зовнішніх факторів та асинхронних подій.

Першочергову увагу під час експериментів було зосереджено на перевірці коректності запуску системи після подачі живлення та її відновлення після

програмного перезавантаження. Зокрема, досліджувалася робота алгоритму визначення початкового положення механізмів (Homing), який є критично важливим для безпечної експлуатації механічної частини сонячного трекера. У ході випробувань підтверджено, що система стабільно виконує процедуру пошуку базової HOME-позиції, коректно обробляє сигнали від кінцевих вимикачів та фіксує результат ініціалізації, що дозволяє уникнути втрати координат і неконтрольованих переміщень приводів.

Окрему групу експериментів становили дослідження роботи системи в різних режимах керування. У ручному режимі перевірялася можливість безпосереднього керування виконавчими механізмами через веб-інтерфейс, коректність обробки керуючих команд та реакція системи на спрацювання кінцевих вимикачів. Особливу увагу приділено поведінці системи під час швидкої зміни напрямків руху та повторюваних команд. В автоматичному режимі аналізувалася стабільність алгоритмів слідкування за сонцем, зокрема здатність системи приймати рішення щодо доцільності коригувальних переміщень за змінних умов освітленості. Результати випробувань показали, що реалізована логіка дозволяє зменшити кількість зайвих рухів приводів без втрати загальної орієнтації сонячної панелі.

Важливим аспектом експериментальних досліджень було оцінювання взаємодії між паралельними програмними задачами, реалізованими на базі операційної системи реального часу FreeRTOS. Під час тривалої безперервної роботи системи не було зафіксовано блокувань, зависань або некоректної взаємодії між задачами керування приводами, моніторингу станів та мережевої комунікації. Навіть за умов підвищеної активності мережевих сервісів та одночасного виконання процедур руху система зберігала передбачувану поведінку, що підтверджує доцільність обраної багатозадачної архітектури.

Значну увагу приділено аналізу даних моніторингу, отриманих через вбудовані IoT-інтерфейси системи. Телеметричні метрики дозволили оцінити динаміку станів системи, частоту переміщень виконавчих механізмів, результати виконання процедури Homing, а також параметри використання апаратних

ресурсів мікроконтролера. Отримані дані підтвердили стабільну роботу програмного забезпечення та відсутність критичних перевантажень оперативної пам’яті, PSRAM або процесорних ресурсів у штатних режимах експлуатації.

Додатково в межах експериментів проводилася оцінка поведінки системи за умов часткової деградації сервісів, зокрема при тимчасовій втраті мережевого з’єднання або затримках у відповіді веб-інтерфейсу. Результати показали, що відсутність мережевої взаємодії не впливає на коректність виконання базових алгоритмів керування та не призводить до порушення логіки роботи виконавчих механізмів. Після відновлення з’єднання система автоматично відновлює доступ до сервісів без необхідності перезапуску.

Окрему серію експериментальних випробувань було проведено з метою оцінювання працездатності системи в несприятливих кліматичних умовах. Зокрема, система тестувалася при пониженій температурі навколишнього середовища до -15 °C, а також під час атмосферних опадів у вигляді дощу. Під час зазначених випробувань підтверджено коректний запуск контролера після подачі живлення, стабільну роботу мережевих сервісів та відсутність збоїв у керуванні виконавчими механізмами. Система успішно переходила в робочий стан, виконувала процедуру ініціалізації та реагувала на керуючі команди без деградації функціональності, що свідчить про достатній запас надійності апаратно-програмної реалізації. Також, перевірялася коректність роботи механізмів віддаленого адміністрування та оновлення програмного забезпечення. Випробування показали можливість оновлення як програмної прошивки, так і файлової системи без втрати даних та порушення працездатності системи. Після завершення оновлень контролер коректно відновлював роботу, а всі сервісні функції залишалися доступними.

Окрему увагу в рамках досліджень приділено оцінюванню надійності та відмовостійкості системи. На апаратному рівні ключову роль відіграють кінцеві вимикачі, які фізично обмежують діапазон руху виконавчих механізмів у напрямках LEFT, RIGHT, UP та DOWN. Наявність окремої HOME-позиції дозволяє системі однозначно визначати еталонне положення механізмів

незалежно від попереднього стану або аварійного завершення роботи. Застосування такої схеми позиціонування підвищує повторюваність роботи системи та зменшує ризик накопичення похибок під час тривалої експлуатації.

На програмному рівні відмовостійкість забезпечується алгоритмом Homing, який виконує відновлення коректного стану системи після перезавантаження або втрати живлення. У разі некоректного завершення процедури система переходить у помилковий стан, що унеможливлює подальший неконтрольований рух приводів. Додатковий рівень надійності забезпечується використанням багатозадачної моделі FreeRTOS, у якій окремі задачі керування, моніторингу та обробки зовнішніх подій працюють незалежно, запобігаючи блокуванню системи в разі локальних помилок.

Важливу роль у забезпеченні стабільності експлуатації відіграє розділення прошивки та файлової системи. Відокремлення firmware від LittleFS дозволяє виконувати оновлення програмного коду без втрати конфігураційних або сервісних даних, а також зберігати інформацію між оновленнями, що суттєво знижує ризик деградації системи під час експлуатації [23].

Однією з ключових відмінностей розробленої системи є високий рівень спостережуваності та керованості. Система експортує набір Prometheus-сумісних метрик, які охоплюють режими роботи, стан руху, результати виконання алгоритму Homing, параметри мережевого з’єднання, використання пам’яті, файлових систем та температурні характеристики мікроконтролера. Використання цих метрик дозволяє здійснювати безперервний моніторинг системи в реальному часі та інтегрувати контролер у зовнішні системи спостереження, зокрема інформаційні панелі Grafana [28].

Контроль версій програмного забезпечення забезпечується унікальними ідентифікаційними параметрами кожної збірки прошивки та файлової системи, включаючи номер версії, час складання, git-хеш і контрольні MD5-суми. Це дозволяє однозначно ідентифікувати активну конфігурацію системи та забезпечити відтворюваність експериментальних результатів [27].

Експериментальні дослідження підтвердили висунуту гіпотезу та дали

змогу оцінити фактичну енергетичну ефективність сонячної панелі номінальною потужністю 475 Вт у зимових умовах експлуатації. Вимірювання проводилися за температури навколишнього середовища близько -11 °C та наявності прямого сонячного випромінювання [9].

Значення напруги, струму та миттєвої потужності знімалися безпосередньо з інтерфейсу портативної електростанції Bluetti, до якої була підключена панель та зафіксовано такі вхідні дані як показано на рисунку 2.21 [розроблено автором].



Рис. 2.21. Поточні параметри генерації фотоелектричної панелі під навантаженням

За зафіксованими даними вхідна напруга становила 35,6 В, струм - 6,6 А, а миттєва потужність генерації - 234 Вт. Орієнтовний коефіцієнт використання номінальної потужності панелі (відносний ККД у даних умовах) визначався за формулою 2.1:

$$\text{ККД} = \left(\frac{P_{\text{фактична}}}{P_{\text{номінальна}}} \right) \times 100\%. \tag{2.1}$$

Підставляємо та обчислюємо наші значення:

$$\text{ККД} = \left(\frac{234}{475} \right) \times 100\% \approx 49{,}2\%.$$

Отримане значення свідчить, що фактичний коефіцієнт корисної дії фотоелектричної панелі взимку становив приблизно 49,2%. Такий рівень ефективності є типовим для зимових умов, коли на генерацію суттєво впливають низька інсоляція, знижена інтенсивність освітлення та малий кут падіння сонячного проміння.

РОЗДІЛ 3. ПРОПОЗИЦІЇ ЩОДО УДОСКОНАЛЕННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

3.1. Напрями технічного та програмного удосконалення системи

Одним із ключових напрямів подальшого удосконалення розробленої системи слідкування за сонцем є модернізація підсистеми визначення положення рухомих механізмів. У поточній реалізації для фіксації крайніх положень та базової HOME-позиції використовуються механічні кінцеві вимикачі, підключені до аналогових входів контролера, що забезпечує просту та надійну реалізацію апаратного захисту.

Разом із тим застосування механічних кінцевих вимикачів має низку обмежень, пов’язаних із фізичним зношуванням контактів, чутливістю до вологи, пилу та температурних коливань. За тривалої експлуатації в умовах відкритого простору ці фактори можуть негативно впливати на стабільність спрацювання та вимагати періодичного технічного обслуговування.

У зв’язку з цим перспективним напрямом розвитку є перехід до безконтактних датчиків положення, зокрема герконових датчиків, які працюють у поєднанні з постійними магнітами. Такий підхід дозволяє усунути механічний контакт між елементами, зменшити зношування та підвищити надійність системи без суттєвого ускладнення апаратної частини. Герконові датчики можуть бути інтегровані в існуючу архітектуру з мінімальними змінами програмної логіки, зберігаючи принципи роботи алгоритму Homing.

Подальшим етапом еволюції системи може стати використання датчиків положення на основі ефекту Холла, які забезпечують вищу стабільність спрацювання та більшу довговічність у порівнянні з механічними й герконовими рішеннями. Застосування таких датчиків відкриває можливість реалізації додаткових точок контролю положення, що дозволяє підвищити точність визначення координат і зменшити накопичення похибок під час тривалої експлуатації. Інтеграція датчиків Холла не потребує кардинальної перебудови програмної архітектури та може бути реалізована шляхом адаптації існуючих

механізмів обробки сигналів [1].

Важливим напрямом удосконалення є подальший розвиток системи моніторингу. У поточній реалізації забезпечено експорт базових Prometheus-сумісних метрик, що дозволяє здійснювати віддалене спостереження за станом системи та її апаратними ресурсами. Розширення набору метрик доцільно здійснювати з урахуванням специфіки механічної та програмної частин системи.

Зокрема, перспективним є впровадження метрик, що характеризують роботу приводів, таких як сумарна тривалість активних переміщень, кількість запусків двигунів, статистика зміни напрямків руху та частота спрацювання датчиків положення. Отримані дані можуть використовуватися для аналізу навантаження на механічні компоненти та прогнозування їх зносу.

Окрему групу метрик доцільно присвятити роботі алгоритму Homing, зокрема фіксації кількості успішних і неуспішних ініціалізацій, часу виконання процедури та причин переходу системи в безпечний стан. Це дозволяє оцінювати стабільність роботи системи після перезапусків і своєчасно виявляти потенційні проблеми в апаратній або програмній частині.

Інтеграція з сервером Prometheus у поєднанні з використанням системи візуалізації Grafana створює умови для централізованого моніторингу кількох пристроїв, що є особливо актуальним у випадку масштабування системи. Наявність історичних даних дозволяє виконувати аналіз тенденцій, порівнювати режими роботи та приймати обґрунтовані рішення щодо оптимізації алгоритмів керування [28], [42].

Точність і стабільність слідкування за сонцем значною мірою залежать від якості сенсорної частини системи. У подальших реалізаціях доцільно приділити особливу увагу вибору та конструктивному виконанню датчиків освітленості. Перспективним є використання спеціалізованих сонячних датчиків із оптичною ізоляцією, світловими каналами або екрануванням від бокового та розсіяного світла, що дозволяє зменшити вплив сторонніх джерел освітлення та погодних факторів.

Стабільність програмної частини є важливим чинником надійності всієї

системи. Подальше удосконалення може включати впровадження більш формалізованої моделі станів, яка чітко визначає допустимі переходи між режимами ініціалізації, ручного керування, автоматичного позиціонування та аварійного стану. Це сприятиме підвищенню передбачуваності поведінки системи та зменшенню кількості неоднозначних сценаріїв.

Окрему увагу доцільно приділити оптимізації взаємодії між задачами FreeRTOS, зокрема механізмам тайм-аутів, пріоритетів і захисту спільних ресурсів. Подальший розвиток сервісної консолі та інтерфейсів керування, включаючи розширене журналювання подій і діагностичні повідомлення, створює передумови для підвищення керованості системи та спрощення її обслуговування в умовах віддаленої експлуатації.

3.2. Перспективи практичного використання та масштабування системи

Перспективним напрямом практичного розвитку розробленої системи слідкування за сонцем є її масштабування шляхом збільшення встановленої потужності без пропорційного ускладнення апаратно-програмної платформи. Запропонована архітектура спочатку проєктувалася з урахуванням можливості роботи в різних конфігураціях, що створює основу для поступового нарощування потужності установки без суттєвих змін логіки керування.

Початковим етапом такого масштабування доцільно розглядати використання однієї сонячної панелі підвищеної потужності, наприклад у діапазоні 600–650 Вт (зокрема 645 Вт), встановленої на одному трекері зі спільною системою керування. Дана конфігурація дозволяє дослідити граничні режими роботи механічної частини системи, оцінити відповідність обраних електроприводів і редукторів підвищеним навантаженням, а також проаналізувати вплив збільшеної маси та моменту інерції на стабільність позиціонування [9].

Особливу увагу при масштабуванні необхідно приділити аналізу динамічних характеристик системи. Збільшення потужності панелі неминуче призводить до зростання механічних навантажень, що може впливати на

плавність руху, частоту коригувальних переміщень і загальний ресурс приводів. У цьому контексті важливим є коригування параметрів керування, зокрема швидкості переміщення, прискорення та допустимої частоти змін положення, без зміни загальної архітектури програмного забезпечення.

Реалізована система моніторингу створює сприятливі умови для кількісної оцінки ефективності масштабування. Prometheus-сумісні метрики дозволяють аналізувати тривалість активних режимів, кількість і амплітуду переміщень, температуру мікроконтролера, а також використання апаратних ресурсів у різних конфігураціях. Порівняння телеметричних даних для панелей різної потужності дає змогу обґрунтовано оцінити запас надійності системи та виявити потенційні обмеження ще на етапі експериментальної експлуатації.

У разі успішного функціонування конфігурації з однією панеллю підвищеної потужності подальшим етапом розвитку може стати перехід до конфігурацій із двома панелями, встановленими на одному трекері. Такий підхід дозволяє підвищити сумарну продуктивність установки за збереження єдиного контролера та спільної системи керування. При цьому вже наявна програмна логіка може бути адаптована переважно на рівні параметрів, без суттєвого перепроєктування архітектури.

Практичне використання розробленої системи є доцільним для малих і середніх автономних сонячних установок, експериментальних стендів, а також напівпромислових рішень, де особливо важливими є модульність, віддалений моніторинг та можливість подальшого розвитку. Гнучкість архітектури дозволяє адаптувати систему до різних типів приводів, сенсорів та умов монтажу, що розширює сферу її застосування - від побутових установок приватного сектору до невеликих енергетичних комплексів у сфері аграрних, телекомунікаційних чи дослідницьких об’єктів. Завдяки можливості масштабування та інтеграції з хмарними сервісами контролю й аналітики, система здатна працювати як у складі одиничних трекерів, так і в багатовузлових кластерних конфігураціях. Поєднання модульності, відмовостійкості та розвинених засобів спостережуваності створює передумови для подальшої комерціалізації [14].

ВИСНОВОКИ

У кваліфікаційній роботі виконано науково-практичне завдання розробки та дослідження програмного забезпечення для керування та моніторингу системи слідкування за сонцем, реалізованого у вигляді автономного IoT-пристрою. Основний акцент роботи зроблено на архітектурі програмного продукту, механізмах віддаленої взаємодії, зберіганні даних, оновленні програмного забезпечення та спостережуваності системи.

У першому розділі здійснено аналіз сучасних систем слідкування за сонцем з точки зору алгоритмічних та програмних підходів до керування, а також вимог до програмної реалізації в автономних системах. Розглянуто відкриті, замкнені та гібридні алгоритми керування, що дозволило обґрунтувати вибір програмної архітектури, орієнтованої на стабільну роботу за змінних умов освітленості та обмежених апаратних ресурсів.

У другому розділі розроблено програмну архітектуру керуючого контролера, яка реалізує функції локального керування, віддаленого адміністрування та моніторингу. Розробка програмного забезпечення виконувалася з використанням Visual Studio Code як основного середовища розробки, що забезпечило зручну роботу з кодом, керування версіями та інтеграцію з інструментами збирання.

Основна логіка керування реалізована мовою C/C++ для вбудованих систем, із використанням модульного підходу до організації коду. Програмна структура побудована таким чином, щоб чітко відокремити логіку керування, мережеву взаємодію, роботу з файловими системами та сервісні функції, що спрощує супровід і розширення програмного продукту.

Ключовим елементом програмного забезпечення є вбудований HTTP-сервер, який забезпечує доступ до системи через стандартний веб-браузер без використання зовнішніх серверів. Веб-інтерфейс реалізовано з використанням HTML, CSS та JavaScript і інтегровано безпосередньо в прошивку контролера. Такий підхід дозволяє розглядати пристрій як самодостатній веб-застосунок,

доступний у локальній мережі.

Для розширених функцій та сервісних сторінок використовується файлова система LittleFS, яка дозволяє зберігати статичні ресурси, конфігураційні файли та службові дані. Окрему увагу приділено організації таблиці розділів (partition table), у межах якої чітко розділено області прошивки, файлової системи та службових розділів. Це забезпечує безпечну експлуатацію та можливість незалежного оновлення окремих компонентів програмного забезпечення.

У системі реалізовано механізм OTA-оновлення, який дозволяє віддалено оновлювати як основну прошивку, так і файлову систему без втрати даних. Кожна збірка програмного забезпечення має унікальні ідентифікаційні параметри, зокрема номер версії, час складання та контрольні суми, що забезпечує контроль життєвого циклу програмного продукту та відтворюваність конфігурацій.

Для зберігання додаткових даних та можливого розширення функціональності передбачено підтримку зовнішніх носіїв інформації, зокрема SD-карти, що може використовуватися для логування подій, збереження історичних даних або резервного копіювання конфігурацій. Такий підхід підвищує гнучкість програмного рішення та відкриває можливості для подальшого розвитку системи.

Значну увагу приділено питанням моніторингу та спостережуваності. Програмне забезпечення експортує телеметричні дані у форматі, сумісному з Prometheus, що дозволяє інтегрувати систему з зовнішніми інструментами аналізу та візуалізації. Реалізований набір метрик охоплює режими роботи системи, стан керування, параметри використання пам’яті та мережевого з’єднання, що відповідає сучасним підходам до розробки керованих програмних систем.

Експериментальні дослідження підтвердили стабільну роботу програмного забезпечення у ручному та автоматичному режимах, коректність обробки мережевих запитів і передбачувану поведінку системи за умов втрати

живлення контролера. Система коректно відновлює роботу після перезавантаження системи чи аварійного вимкнення.

У роботі реалізовано програмну архітектуру автономного керуючого IoT-пристрою, яка поєднує в собі вбудований веб-сервер, механізми OTA-оновлення, файлові системи та засоби віддаленого моніторингу без використання зовнішньої серверної інфраструктури. Такий підхід дає змогу створити повністю самодостатню систему, здатну функціонувати у віддалених або автономних умовах, забезпечуючи надійне керування виконавчими механізмами, збір телеметрії та обслуговування користувача через зручний інтерфейс керування. Запропонована програмна архітектура може слугувати універсальною основою для розробки інших вбудованих керуючих комплексів, зокрема систем автоматизації, енергоощадних рішень, сенсорних вузлів або компактних кіберфізичних об’єктів.

Практичне значення роботи полягає у можливості багаторазового використання створеної програмної платформи для реалізації різноманітних IoT-рішень, систем керування та навчальних проєктів у сфері вбудованих інформаційних технологій. Архітектура програмного забезпечення побудована таким чином, щоб бути легко масштабованою, а модульний підхід забезпечує просте додавання нових функціональних компонентів, зокрема нових алгоритмів керування, додаткових сенсорів, нових мережевих протоколів чи механізмів діагностики. Це робить систему не лише гнучкою, але й придатною для довгострокового розвитку в умовах змін апаратної бази або нових вимог експлуатації.

Таким чином, у кваліфікаційній роботі досягнуто поставленої мети та виконано всі визначені завдання. Розроблений програмний продукт демонструє сучасний підхід до створення вбудованих веб-орієнтованих IoT-систем та може слугувати основою для подальших розробок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ванько В. М., Поліщук Є. С., Дорожовець М. М., Яцук В. О., Яцук Ю. В. Вимірювальні перетворювачі (сенсори) : підручник/ за ред. Є. С. Поліщука, В. М. Ванька ; Нац. ун-т «Львівська політехніка». - Львів, 2015. - 580 с.

2. Жолубак І. М., Матвієць В. Ю. Трекер для сонячних електростанцій [Електронний ресурс] // Computer Systems and Networks. - 2022. - Vol. 4, № 1. - С. 38-48. - Режим доступу: URL [https://science.lpnu.ua/sites/default/files/journal-paper/2022/dec/29469/zhurnal-ksm41verstka1-39-48.pdf]

3. Жуков Р. О., Петрик О. А. Сонячні трекери в умовах обмеженого простору для підвищення ефективності сонячних електростанцій [Електронний ресурс] // Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки. - 2025. - № 3. - Режим доступу: URL [https://www.tech.vernadskyjournals.in.ua/journals/2025/3_2025/part_1/22.pdf]

4. Колонтаєвський Ю. П., Сосков А. Г. Електроніка і мікросхемотехніка : підручник для студентів закладів вищої освіти / за ред. А. Г. Соскова. - 2-ге вид. - Київ : Каравела, 2009. - 416 с.

5. Кульбаченко В. Ю. Збірник матеріалів науково-практичної конференції / Львівський національний університет імені Івана Франка, Факультет фінансів та бізнес-менеджменту. - Львів : ЛНУ ім. І. Франка, 2025. - Електронний ресурс. – Режим доступу: URL [https://financial.lnu.edu.ua/wp- content/uploads/2025/06/Zbirnyk-konferentsii-15-travnia-2025.pdf]

6. Кульбаченко В. Ю., Команда ДПУ. Інноваційний IoT-проєкт у сфері сонячної енергетики [Електронний ресурс] // Державний податковий університет. - Режим доступу: URL [https://dpu.edu.ua/novyny/student-dpu-predstavyv-vlasnu-iot-systemu-dlia-soniachnoi-enerhetyky]

7. Кульбаченко В. Ю., Горбовий А. Ю. Розробка системи слідкування за сонцем за допомогою Інтернету речей [Електронний ресурс]. - Черкаси, 2025. - Режим доступу: URL [https://drive.google.com/file/d/1vfK7HzALRZHFTE8SKi6P_c-D3X4K3YPK/view]

8. Кульбаченко В. Ю. Розробка системи слідкування за сонцем за допомогою Інтернету речей [Електронний ресурс]. - Вінниця, 2025. - Режим доступу: URL [https://drive.google.com/file/d/1XEdZDbm41x4qN5wgAY03D_GhdNwXdfMr/view]

9. Кульбаченко В. Ю., Горбовий А. Ю. Сонячна панель LONGi Solar LR7-54HTH-465M: конструктивні особливості, кристалічна структура та технологічні інновації [Електронний ресурс]. - Хмельницький, 2025. - Режим доступу: URL [https://xti.khmnu.edu.ua/wp-content/uploads/sites/40/itmie-2025-sr.pdf]

10. Лисенков М. О., Ключник І. І. Мікроконтролери в приладах і пристроях : підручник / МОН України, Харків. нац. ун-т радіоелектроніки. - Харків : ХНУРЕ, 2014. - 368 с.

11. Нікітчук А. В. Програмування вбудованих систем Інтернету речей [Електронний ресурс] : навч. посіб. - Київ : КПІ ім. Ігоря Сікорського, 2024. - 120 с. - Режим доступу: URL [https://ela.kpi.ua/bitstreams/63d3d8fc-d191-405e-a61f-4118251e02a7/download]

12. Пархоменко А. В., Гладкова О. М., Залюбовський Я. І., Пархоменко А. В. Інженерія вбудованих систем : навчальний посібник. - Запоріжжя : Дике Поле, 2017. - 220 с.

13. Синєглазов В. М., Зеленков О. А., Аскеров Ш. І., Дмитренко Б. І. Відновлювальна енергетика : навчальний посібник. - Київ : НАУ, 2015. - 278 с.

14. Скіба М., Говорущенко Т., Мартинюк В., Засорнов О., Павлова О. Інтелектуальна сонячна слідкуюча система як перспектива розвитку альтернативної енергетики в Україні [Електронний ресурс] // CEUR Workshop Proceedings. - 2021. - № 2853. - С. 1-12. - Режим доступу: URL [https://ceur-ws.org/Vol-2853/paper2.pdf]

15. Скришевський В. А. Що таке сонячна енергетика і чи потрібна вона Україні? [Електронний ресурс] // SolarSystem. - Режим доступу: URL [https://solarsystem.com.ua/shho-take-sonyachna-energetyka-chy-potribna-vona-ukrayini]

16. Сонячний трекер. Переваги і недоліки встановлення [Електронний

ресурс] // Електротехнічна продукція. – Режим доступу: URL [https://prel.prom.ua/a279654-sonyachnij-treker-scho.html]

17. Технології інтернету речей : навчальний посібник [Електронний ресурс] / Б. Ю. Жураковський, І. О. Зенів ; КПІ ім. Ігоря Сікорського. - Київ, 2021. - 271 с.

18. Як працює сонячна батарея? [Електронний ресурс] // Сонячна енергія. – Режим доступу: URL [https://sun-energy.com.ua/articles/yak_pratsuyut_sonyachni_paneli]

19. Barr M., Massa A. Programming Embedded Systems: With C and GNU Development Tools. - 2nd ed. - Sebastopol : O'Reilly Media, 2006. - 328 p.

20. Beazley D., Jones B. Python Cookbook. - 3rd ed. - Sebastopol : O'Reilly Media, 2013. - 706 p.

21. Boxwell M. Solar Electricity Handbook: A Practical Guide to Solar Energy. - Birmingham : Greenstream Publishing, 2017. - 212 p.

22. Design of tracking photovoltaic systems with a single vertical axis // Progress in Photovoltaics: Research and Applications. - 2002. - Vol. 10. - P. 533-543.

23. Espressif Systems. ESP-IDF Programming Guide [Електронний ресурс]. - Режим доступу: URL : [https://docs.espressif.com/projects/esp-idf/en/latest/esp32/]

24. Espressif Systems. ESP32-S3-DevKitC-1 v1.1 User Guide [Електронний ресурс]. - Режим доступу: URL : [https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/]

25. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Doctoral dissertation. – University of California, Irvine, 2000. - 180 p.

26. Furuta K. Mastering ESP32 Practical Projects with Arduino IDE Programming. - 2024. - 229 p.

27. GNU Project. Bash Reference Manual [Electronic resource]. - Free Software Foundation. - [Електронний ресурс]. - Режим доступу: URL https://www.gnu.org/software/bash/manual/

28. Grafana Labs. Grafana. - [Електронний ресурс]. - Режим доступу: URL [https://grafana.com/docs/]

29. Keith J., Andrew R. HTML5 for Web Designers. - New York : A Book Apart, 2010. - 98 p.

30. Light Dependent Resistor (LDR) or Photoresistor . - [Електронний ресурс]. - Режим доступу: URL https://www.electronics-notes.com/articles/electronic_components/resistors/light-dependent-resistor-ldr.php

31. Lutz M. Learning Python. - 5th ed. - Sebastopol : O'Reilly Media, 2013. - 1648 p.

32. Margolis M. Arduino Cookbook. - 3rd ed. - Sebastopol : O'Reilly Media, 2020. - 726 p.

33. Meier A. Developing IoT Projects with ESP32. - Birmingham : Packt Publishing, 2021. - 398 p.

34. Meyer E. A., Weyl E. CSS: The Definitive Guide. - 4th ed. - Sebastopol : O'Reilly Media, 2018. – 544 p.

35. Microsoft. Visual Studio Code Documentation - [Електронний ресурс]. - Режим доступу: URL : [https://code.visualstudio.com/docs]

36. Mozilla Developer Network. HTML Living Standard . - [Електронний ресурс]. - Режим доступу: URL [https://developer.mozilla.org/en-US/docs/Web/HTML]

37. Photodiode Basics [Electronic resource] // Electronics For You. . - [Електронний ресурс]. - Режим доступу: URL [https://www.electronicsforu.com/technology-trends/learn-electronics/photodiode-working-applications]

38. PlatformIO. PlatformIO IDE Documentation. - [Електронний ресурс]. - Режим доступу: URL [https://docs.platformio.org/]

39. Prinsloo G. J., Dobson R. T. Solar Tracking. - Stellenbosch : SolarBooks, 2015. - 542 p.

40. Richardson L., Ruby S. RESTful Web Services. - Sebastopol : O'Reilly Media, 2007. - 448 p.

41. TEMT6000 Ambient Light Sensor // Components101 - [Електронний ре

сурс]. - Режим доступу: URL https://components101.com/sensors/temt6000-ambient-light-sensor

42. Turnbull J. The Prometheus Monitoring System. - Turnbull Press, 2018. - 196 p.

43. W3C. CSS Snapshot 2023 - [Електронний ресурс]. - Режим доступу: URL https://www.w3.org/TR/css-2023/

44. Yu J. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. - 3rd ed. - Oxford : Newnes, 2013. - 864 p.

ДОДАТКИ

Додаток А



Джерело: [розроблено автором]

Рис. А.1.3. Сертифікат ЧДТУ Черкаси

Продовження додатку А



Джерело: [розроблено автором]

Рис. А.1.4. Сертифікат ХНУ Хмельницький

Продовження додатку А



Джерело: [розроблено автором]

Рис. А.1.5. Сертифікат ВНТУ Вінниця

Додаток Б



Джерело: [розроблено автором]

Рис. Б.2.1. Початковий прототип трекера



Джерело: [розроблено автором]

Рис. Б.2.2. Фінальний прототип сонячного трекера на етапі налагодження (без панелі)

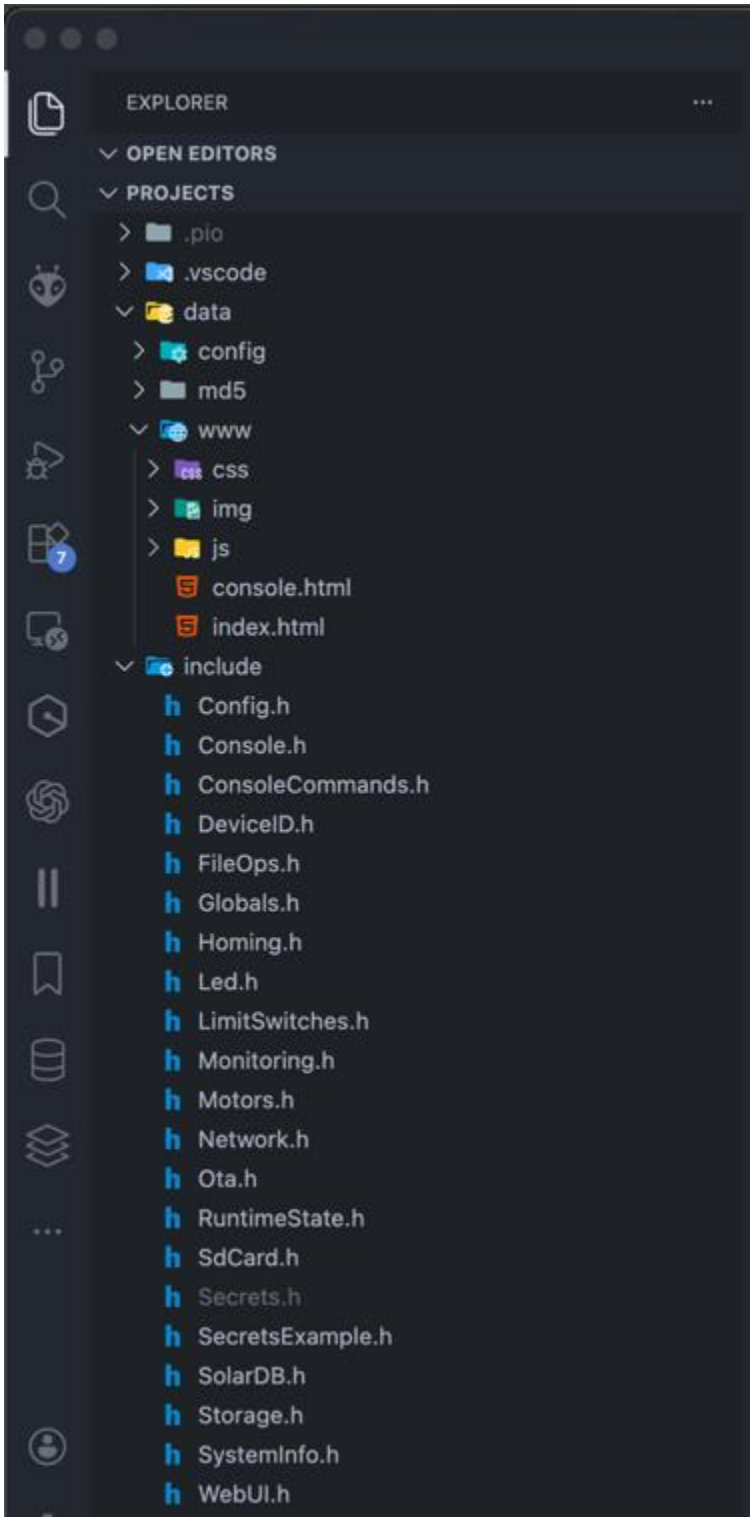
Продовження додатку Б



Джерело: [розроблено автором]

Рис. Б.2.3. Фінальний прототип сонячного трекера в умовах реальної експлуатації

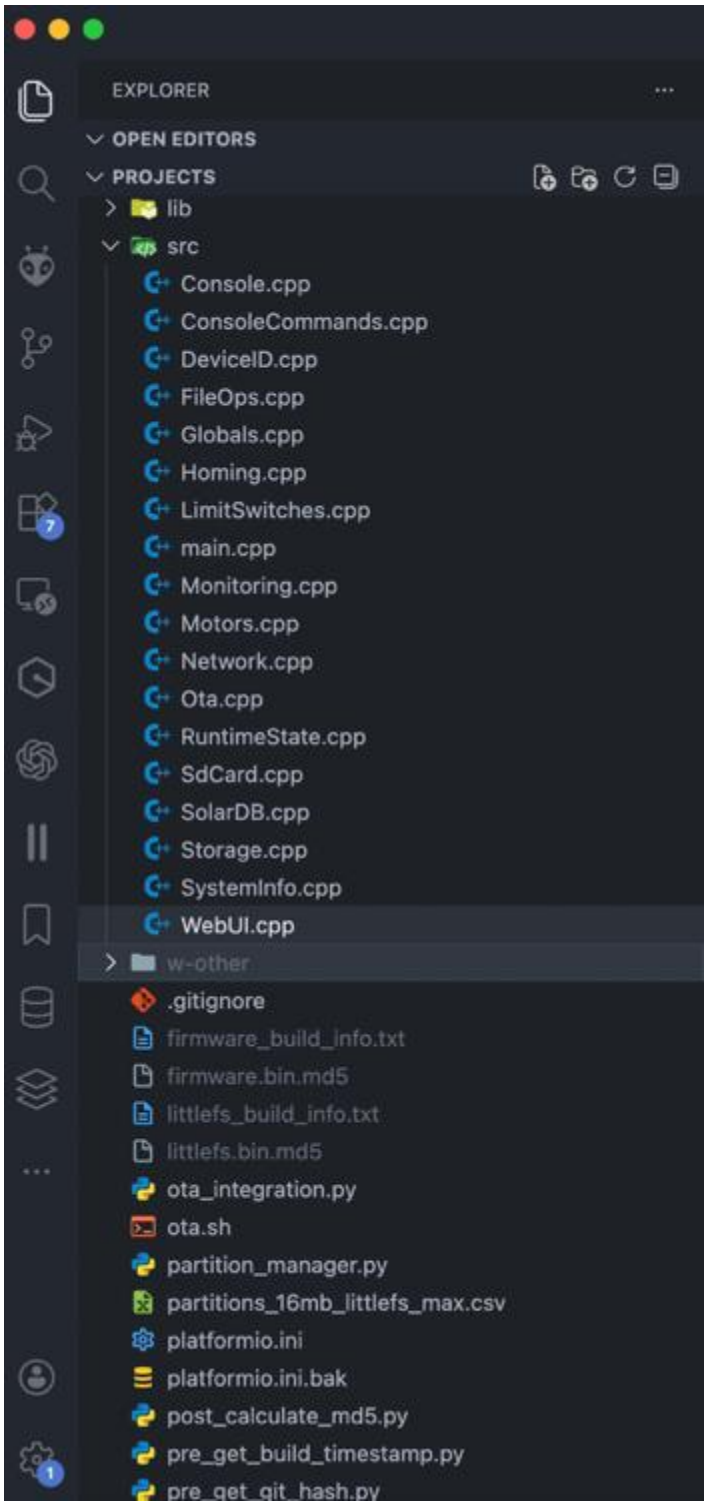
Додаток В



Джерело: [розроблено автором]

Рис. 2.9. Структура проекту

Продовження додатку В



Джерело: [розроблено автором]

Рис. 2.10. Структура проекту (продовження)

Додаток Г

```
// Modules
#include <Arduino.h>
#include <ESPAsyncWebServer.h>
#include <AsyncTCP.h>
#include <WiFi.h>
#include <Adafruit_NeoPixel.h>
#include <AsyncElegantOTA.h>
#include <ESPmDNS.h>
#include <SPI.h>
#include <SD.h>
#include <ArduinoJson.h>
#include <WebSocketsServer.h>

// include
#include "DeviceID.h"
#include "RuntimeState.h"
#include "Globals.h"
#include "Config.h"
#include "LimitSwitches.h"
#include "Motors.h"
#include "Network.h"
#include "WebUI.h"
#include "Led.h"
#include "SdCard.h"
// #include "Time.h"
#include "Storage.h"
#include "SystemInfo.h"
#include "Console.h"
#include "Ota.h"
#include "ConsoleCommands.h"
#include "Monitoring.h"
#include "Homing.h"

// using namespace
using namespace RuntimeState;

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  delay(2000);

  // Print Banner
  printBanner();

  ConsolePrintf ("\n%-16s %-44s\n", "", "[ DEVICE ]");
  DEVICE_CHIP_ID  = getChipId();
  DEVICE_FLASH_ID = getFlashId();
  DEVICE_UID      = getDeviceUID();

  ConsolePrintf("[DEVICE] CHIP ID:  %s\n", DEVICE_CHIP_ID.c_str());
  ConsolePrintf("[DEVICE] FLASH ID: %s\n", DEVICE_FLASH_ID.c_str());
  ConsolePrintf("[DEVICE] UID:      %s\n", DEVICE_UID.c_str());

```

Джерело: [розроблено автором]

Рис. Г.2.11. Вихідний код файлу main.cpp

Продовження додатку Г

```
54    esp_reset_reason_t reason = esp_reset_reason();
55
56    ConsolePrintf ("\n%-16s %-44s\n", "", "[ BOOT ]");
57    ConsolePrintf("[BOOT] Reset reason: %d\n", reason);
58
59    // Якщо це холодний старт (живлення тільки подали)
60    if (reason == ESP_RST_POWERON) {
61      ConsolePrintln("[BOOT] Power-on detected -> stabilizing...");
62    }
63
64    // Якщо рестарт чи reset — звичайний запуск
65    ConsolePrintln("[BOOT] Normal startup. Initializing system...");
66
67    // Init Storage == LittleFS,FATFS,SD
68    Storage::begin();
69
70    // Init Pixel
71    neoPixel.begin();
72    neoPixel.setBrightness(32);
73    setLedColor(0, 0, 255);
74
75    // WI-FI
76    wifi_scan();
77    bool wifiConnected = connectToConfiguredWifi(WIFI_CONNECT_TIMEOUT_MS);
78
79    if (wifiConnected) {
80      ConsolePrintf("[WI-FI] Connected with IP: %s\n", WiFi.localIP().toString().c_str());
81    } else {
82      ConsolePrintln("[WI-FI] STA connection failed, starting fallback AP");
83      startFallbackAccessPoint();
84    }
85
86    // Ws Console start
87    wsConsole.begin();
88    wsConsole.onEvent([](uint8_t num, WStype_t type, uint8_t* payload, size_t length){
89        if (type == WStype_CONNECTED) {
90            // Send the entire history of logs
91            String hist = ConsoleGetBufferedLog();
92            wsConsole.sendTXT(num, hist.c_str());
93            // return;
94        }
95        // Add text command processing
96        if (type == WStype_TEXT) {
97            String cmd = String((char*)payload);
98            handleConsoleCommand(cmd);
99            return;
100       }
101   });
102
103   // mDNS
104   startMdns();
105   ledHoldUntil = millis() + STATUS_LED_HOLD_TIME_MS;
```

Джерело: [розроблено автором]

Рис. Г.2.12. Вихідний код файлу main.cpp (продовження)

Продовження додатку Г

```
107  // ---------- Routes ----------
108  // httpServer.serveStatic("/", LittleFS, "/www/");
109
110  // ======= Registration of OTA routes =======
111  //  /update_fs, /update_fw
112  registerOtaRoutes(&httpServer);
113
114  httpServer.on("/console", HTTP_GET, [](AsyncWebServerRequest *request){
115    if (!ensureAuth(request)) return;
116    request->redirect("/console.html");
117  });
118
119  httpServer.on("/console.html", HTTP_GET, [](AsyncWebServerRequest *request){
120    if (!ensureAuth(request)) return;
121    request->send(LittleFS, "/www/console.html", "text/html");
122  });
123
124  httpServer.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
125    if (!ensureAuth(request)) return;
126    handleRoot(request);
127  });
128
129  httpServer.on("/api/state", HTTP_GET, [](AsyncWebServerRequest *request) {
130    if (!ensureAuth(request)) return;
131    String payload = String("{\"autoMode\":") + (autoMode ? "true" : "false") + "}";
132    request->send(200, "application/json", payload);
133  });
134
135  httpServer.on("/api/move", HTTP_POST, [](AsyncWebServerRequest *request) {
136    if (!ensureAuth(request)) return;
137    if (autoMode) {
138      request->send(409, "application/json", jsonStatus("error", "auto_mode_active"));
139      return;
140    }
141    if (!request->hasParam("dir")) {
142      request->send(400, "application/json", jsonStatus("error", "missing_dir"));
143      return;
144    }
145    String dir = request->getParam("dir")->value();
146    bool handled = true;
147    if (dir == "left") {
148      moveLeft();
149    } else if (dir == "right") {
150      moveRight();
151    } else if (dir == "up") {
152      moveUp();
153    } else if (dir == "down") {
154      moveDown();
155    } else {
156      handled = false;
157    }
158
```

Джерело: [розроблено автором]

Рис. Г.2.13. Вихідний код файлу main.cpp (продовження)

Продовження додатку Г

```
    if (!handled) {
      request->send(400, "application/json", jsonStatus("error", "invalid_dir"));
      return;
    }
    request->send(200, "application/json", jsonStatus("status", "moving"));
  });

  httpServer.on("/api/stop", HTTP_POST, [](AsyncWebServerRequest *request) {
    if (!ensureAuth(request)) return;
    stopAll();
    request->send(200, "application/json", jsonStatus("status", "stopped"));
  });

  httpServer.on("/api/mode", HTTP_POST, [](AsyncWebServerRequest *request) {
    if (!ensureAuth(request)) return;
    if (!request->hasParam("state")) {
      request->send(400, "application/json", jsonStatus("error", "missing_state"));
      return;
    }
    String state = request->getParam("state")->value();
    autoMode = !(state == "manual" || state == "0" || state == "false");
    stopAll();
    String payload = String("{\"autoMode\":") + (autoMode ? "true" : "false") + "}";
    request->send(200, "application/json", payload);
  });

  httpServer.on("/api/home", HTTP_POST, [](AsyncWebServerRequest *request) {
      if (!ensureAuth(request)) return;

      // AUTO mode ON -> HOME disabled (Homing allowed only in MANUAL mode)
      if (RuntimeState::autoMode) {
          request->send(409, "application/json",
              jsonStatus("error", "auto_mode_active"));
          return;
      }

      // Homing already running
      if (RuntimeState::homingInProgress) {
          request->send(409, "application/json",
              jsonStatus("error", "homing_in_progress"));
          return;
      }

      Homing::begin();

      request->send(200, "application/json",
          jsonStatus("status", "homing_started"));
  });

  httpServer.on("/api/reboot", HTTP_POST, [](AsyncWebServerRequest *request) {
    if (!ensureAuth(request)) return;
    request->send(200, "text/plain", "Rebooting...");
    request->client()->close();
    delay(200);
```

Джерело: [розроблено автором]

Рис. Г.2.14. Вихідний код файлу main.cpp (продовження)

Продовження додатку Г

```
212     delay(200);
213     ESP.restart();
214   });
215
216   httpServer.on("/metrics", HTTP_GET, [](AsyncWebServerRequest *request) {
217       if (!ensureAuth(request)) return;
218
219     String out = "";
220     out += "# ================================================\n";
221     out += "# DEVICE IDENTIFIERS\n";
222     out += "# ================================================\n";
223     out += "device_chip_id{id=\"" + DEVICE_CHIP_ID + "\"} 1\n";
224     out += "device_flash_id{id=\"" + DEVICE_FLASH_ID + "\"} 1\n";
225     out += "device_uid{id=\"" + DEVICE_UID + "\"} 1\n\n";
226
227     //------------------------------------------------
228     // FIRMWARE → INFO
229     //------------------------------------------------
230     out += "# ================================================\n";
231     out += "# FIRMWARE BUILD INFO\n";
232     out += "# ================================================\n";
233     out += "firmware_version{value=\"" + String(FW_VERSION) + "\"} 1\n";
234     // raw timestamp (epoch)
235     out += "firmware_build_timestamp " + String(FW_BUILD) + "\n";
236     // formatted human-readable time
237     out += "firmware_build_human{date=\"" + formatBuildTime(FW_BUILD) + "\"} 1\n";
238     out += "firmware_git_hash{hash=\"" + String(FW_GIT_HASH) + "\"} 1\n";
239     // MD5 hash of compiled firmware (sketch)
240     out += "firmware_md5{value=\"" + ESP.getSketchMD5() + "\"} 1\n\n";
241
242     // bool fwMatches = (ESP.getSketchMD5() == String(FW_MD5));
243
244     // out += "firmware_integrity{ok=\"" + String(fwMatches ? "true" : "false") +
245     //        "\",expected=\"" + String(FW_MD5) +
246     //        "\",actual=\"" + ESP.getSketchMD5() +
247     //        "\"} 1\n\n";
248
249     //------------------------------------------------
250     // LITTLEFS → INFO
251     //------------------------------------------------
252     out += "# ================================================\n";
253     out += "# LITTLEFS  BUILD INFO\n";
254     out += "# ================================================\n";
255     out += "littlefs_version{value=\"" + String(FS_LITTLEFS_VERSION) + "\"} 1\n";
256     out += "littlefs_build_timestamp " + String(FS_LITTLEFS_BUILD) + "\n";
257     out += "littlefs_build_human{date=\"" + formatBuildTime(FS_LITTLEFS_BUILD) + "\"} 1\n";
258     out += "littlefs_git_hash{hash=\"" + String(FS_LITTLEFS_GIT_HASH) + "\"} 1\n";
259     out += "littlefs_md5{value=\"" + String(FS_LITTLEFS_MD5) + "\"} 1\n\n";
260
261     out += "# ================================================\n";
262     out += "# SYSTEM METRICS\n";
263     out += "# ================================================\n";
264     unsigned long uptime_minutes = millis() / 1000 / 60;
265     out += "uptime_minutes " + String(uptime_minutes) + "\n";
```

Джерело: [розроблено автором]

Рис. Г.2.15. Вихідний код файлу main.cpp (продовження)

```
267     //------------------------------------------------
268     // WIFI LABELS
269     //------------------------------------------------
270     if (WiFi.isConnected()) {
271         out += "wifi_connected 1\n";
272         // Single metric with all labels
273         out += "wifi_info{ssid=\"" + WiFi.SSID() +
274               "\",mac=\"" + WiFi.macAddress() +
275               "\",ip=\"" + WiFi.localIP().toString() +
276               "\"} 1\n";
277         // Signal (dbm)
278         out += "wifi_rssi_dbm " + String(WiFi.RSSI()) + "\n";
279     } else {
280         out += "wifi_connected 0\n";
281         // Empty labels when not connected
282         out += "wifi_info{ssid=\"\",mac=\"\",ip=\"\"} 0\n";
283         out += "wifi_rssi_dbm 0\n";
284     }
285
286     //------------------------------------------------
287     // STORAGE: LITTLEFS
288     //------------------------------------------------
289     out += "littlefs_mounted " + String(Storage::isLittleFSMounted()) + "\n";
290
291     if (Storage::isLittleFSMounted()) {
292         size_t lfs_total = LittleFS.totalBytes();
293         size_t lfs_used  = LittleFS.usedBytes();
294         size_t lfs_free  = lfs_total - lfs_used;
295
296         out += "littlefs_total_kb " + String(lfs_total / 1024) + "\n";
297         out += "littlefs_used_kb "  + String(lfs_used / 1024) + "\n";
298         out += "littlefs_free_kb "  + String(lfs_free / 1024) + "\n";
299     }
300
301     //------------------------------------------------
302     // STORAGE: FATFS (FFat)
303     //------------------------------------------------
304     out += "fatfs_mounted " + String(Storage::isFatMounted()) + "\n";
305
306     if (Storage::isFatMounted()) {
307         size_t fat_total = FFat.totalBytes();
308         size_t fat_used  = FFat.usedBytes();
309         size_t fat_free  = fat_total - fat_used;
310
311         out += "fatfs_total_kb " + String(fat_total / 1024) + "\n";
312         out += "fatfs_used_kb "  + String(fat_used / 1024) + "\n";
313         out += "fatfs_free_kb "  + String(fat_free / 1024) + "\n";
314     }
315
316     //------------------------------------------------
317     // STORAGE: SD CARD
318     //------------------------------------------------
319     out += "sdcard_mounted " + String(Storage::isSDMounted()) + "\n";
320
```

Джерело: [розроблено автором]

Рис. Г.2.16. Вихідний код файлу main.cpp (продовження)

Продовження додатку Г

```
    if (Storage::isSDMounted()) {

        uint64_t cardBytes = SD.cardSize();
        uint64_t cardMB = cardBytes / 1024 / 1024;
        uint64_t cardGB = cardBytes / 1024 / 1024 / 1024;

        uint64_t totalMB = SD.totalBytes() / 1024 / 1024;
        uint64_t usedMB  = SD.usedBytes()  / 1024 / 1024;
        uint64_t freeMB  = totalMB - usedMB;

        out += "sdcard_capacity_mb " + String(cardMB) + "\n";
        out += "sdcard_capacity_gb " + String(cardGB) + "\n";

        out += "sdcard_total_mb " + String(totalMB) + "\n";
        out += "sdcard_used_mb "  + String(usedMB) + "\n";
        out += "sdcard_free_mb "  + String(freeMB) + "\n\n";
    }

    // -1: never executed, 0: failed or false, 1: success or true
    out += "# ================================================\n";
    out += "# RUNTIME STATE METRICS\n";
    out += "# ================================================\n";
    out += "auto_mode " + String(RuntimeState::autoMode) + "\n";
    out += "active_direction{dir=\"" + directionToLabel(RuntimeState::activeDirection) + "\"} 1\n";
    out += "homing_in_progress " + String(RuntimeState::homingInProgress) + "\n";
    out += "last_homing_status " + String(RuntimeState::lastHomingStatus) + "\n";
    out += "error_state " + String(RuntimeState::errorState) + "\n\n";

    out += "# ================================================\n";
    out += "# LIMIT SWITCHES METRICS\n";
    out += "# ================================================\n";
    out += "limit_left "  + String(LimitSwitches::limitLeftTriggered()) + "\n";
    out += "limit_right " + String(LimitSwitches::limitRightTriggered()) + "\n";
    out += "limit_up "    + String(LimitSwitches::limitUpTriggered()) + "\n";
    out += "limit_down "  + String(LimitSwitches::limitDownTriggered()) + "\n";
    out += "limit_home "  + String(LimitSwitches::limitHomeTriggered()) + "\n\n";

    out += "# ================================================\n";
    out += "# CHIP AND CPU INFO\n";
    out += "# ================================================\n";
    out += "chip_revision " + String(ESP.getChipRevision()) + "\n";
    out += "chip_cores " + String(ESP.getChipCores()) + "\n";
    out += "chip_model_info{model=\"" + String(ESP.getChipModel()) + "\"} 1\n";

    //------------------------------------------
    // SYSTEMINFO → CPU TEMPERATURE
    //------------------------------------------
    float temp;
    temp_sensor_config_t cfg = TSENS_CONFIG_DEFAULT();
    temp_sensor_set_config(cfg);
    temp_sensor_start();
    temp_sensor_read_celsius(&temp);
    out += "cpu_temperature_celsius " + String(temp) + "\n\n";
```

Джерело: [розроблено автором]

Рис. Г.2.17. Вихідний код файлу main.cpp (продовження)

Продовження додатку Г

```
    out += "# ================================================\n";
    out += "# RAM INFO\n";
    out += "# ================================================\n";
    //-----------------------------------------------
    // SYSTEMINFO → RAM
    //-----------------------------------------------
    out += "heap_total_kb " + String(ESP.getHeapSize() / 1024) + "\n";
    out += "heap_free_kb " + String(ESP.getFreeHeap() / 1024) + "\n";
    out += "heap_min_free_kb " + String(ESP.getMinFreeHeap() / 1024) + "\n";
    out += "heap_max_alloc_kb " + String(ESP.getMaxAllocHeap() / 1024) + "\n";
    out += "psram_total_kb " + String(ESP.getPsramSize() / 1024) + "\n";
    out += "psram_free_kb " + String(ESP.getFreePsram() / 1024) + "\n\n";

    out += "# ================================================\n";
    out += "# FLASH / SDK\n";
    out += "# ================================================\n";
    //-----------------------------------------------
    // SYSTEMINFO → FLASH
    //-----------------------------------------------
    out += "flash_size_mb " + String(spi_flash_get_chip_size() / 1024 / 1024) + "\n";
    out += "flash_speed_mhz " + String(ESP.getFlashChipSpeed() / 1000000) + "\n";
    //-----------------------------------------------
    // SYSTEMINFO → SDK
    //-----------------------------------------------
    out += "sdk_info{version=\"" + String(ESP.getSdkVersion()) + "\"} 1\n\n";

    //-----------------------------------------------
    // SYSTEMINFO → SKETCH
    //-----------------------------------------------
    out += "# ================================================\n";
    out += "# SKETCH\n";
    out += "# ================================================\n";
    out += "sketch_size_kb " + String(ESP.getSketchSize() / 1024) + "\n";
    out += "sketch_free_kb " + String(ESP.getFreeSketchSpace() / 1024) + "\n";
    out += "sketch_info{md5=\"" + ESP.getSketchMD5() + "\"} 1\n\n";

    request->send(200, "text/plain", out);
});


AsyncElegantOTA.begin(&httpServer, ADMIN_USER, ADMIN_PASSWORD);
httpServer.begin();
ConsolePrintf ("\n%-16s %-44s\n", "", "[ LIMIT/MOTORS ]");
// Init Limit Switches
LimitSwitches::begin();

// Init Motors
setupPins();
stopAll();
delay(50);

ConsolePrintf ("\n%-16s %-44s\n", "", "[ INFO ]");
ConsolePrintln("[INFO] System started successfully");
```

Джерело: [розроблено автором]

Рис. Г.2.18. Вихідний код файлу main.cpp (продовження)

Продовження додатку Г

```
429     // Partitions show info
430     ConsolePrintln("\n[INFO] Showing ESP32 system information...");
431     SystemInfo::printPartitions();
432     Storage::printFilesystemInfo();
433     SystemInfo::printAll();
434
435     // Print Banner
436     printBanner();
437
438     Monitoring::begin();
439     Homing::begin();
440   }
441
442   void loop() {
443     // WebSocketsServer
444     wsConsole.loop();
445
446     // ElegantOTA
447     AsyncElegantOTA.loop();
448
449     if (ledHoldUntil && static_cast<long>(millis() - ledHoldUntil) >= 0) {
450       setLedColor(0, 0, 0);
451       ledHoldUntil = 0;
452     }
453
454     // if (RuntimeState::errorState) {
455     //     ConsolePrintln("SYSTEM ERROR - STOPPING ALL");
456     //     stopAll();
457     // }
458   }
```

Джерело: [розроблено автором]

Рис. Г.2.19. Вихідний код файлу main.cpp (продовження)