Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка комп'ютеризованої системи керування сонячними панелями з використанням технологій IoT

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Пастерук П.П.
(підпис) (прізвище та ініціали)

Керівник _____ Голотенко О.С.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Чихіра І.В.
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Пастерук Петро Петрович
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка комп'ютеризованої системи керування сонячними панелями з використанням технологій IoT

Керівник роботи Голотенко Олександр Сергійович, кандидат технічних наук, доцент кафедри комп'ютерно-інтегрованих технологій
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 23 червня 2025р.

3. Вихідні дані до роботи Наукові публікації, Інтернет-джерела та технічна документація щодо розробки компютеризованих систем управління сонячними панелями з використанням технологій Інтернету речей.

4.Зміст роботи (перелік питань, які потрібно розробити)

Вступ. 1. Аналіз предметної області та постановка завдання на розробку комп'ютеризованої системи керування сонячними панелями. 1.1. Активні системи орієнтації сонячних панелей: типи та принципи роботи. 1.2 Концепція Інтернету речей (IoT) та її застосування. 1.3 Огляд сучасних реалізацій "розумних" сонячних трекерів. 1.4Висновки до першого розділу.
2. Проєктна частина. 2.1 Постановка вимог до системи та загальна архітектура. 2.2 Вибір та підключення компонентів апаратної частини. 2.3 Алгоритми керування та програмне забезпечення контролера. 2.4 Інтеграція з IoT платформою. 2.5 Висновки до другого розділу.
3. Практична реалізація прототипу та результати тестування. 3.1 Збирання прототипу системи. 3.2 Програмне забезпечення та IoT-функціональність. 3.3 Результати експериментів та аналіз. 3.4 Висновки до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

Повний перелік слайдів у презентації, включаючи перший та останній слайди (з номерами, через крапку). 1. Тема роботи. 2. Актуальність роботи. 3. Мета роботи. 4. Завдання роботи. 5. Аналіз існуючих систем керування. 6. Загальна архітектура проектованої системи. 7. Компоненти розробленої системи. 8. Структурна схема розробленої системи. 9. Підключення компонентів до системи. 10. Порівняння розробленої системи із стаціонарною. 11. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	08.05.2025-15.05.2025	
3.	Виконання дослідження щодо анактуального стану	16.05.2025-22.05.2025	
	речей, що стосується активних систем орієнтації сонячних панелей, їх типів та принципів роботи,		
	формулювання завдання та визначення		
	функціональних вимог до системи		
4.	Оформлення розділу «Аналіз предметної області та	23.05.2025-01.06.2025	
	постановка завдання на розробку комп’ютеризованої		
	Системи керування сонячними панелями»		
5.	Оформлення розділу «Проєктна частина»	23.05.2025-01.06.2025	
6.	Оформлення розділу «Практична реалізація прототтипу та результати тестування	23.05.2025-01.06.2025	
7.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи охорони праці»	06.06.2025-07.06.2025	
8.	Оформлення кваліфікаційної роботи	02.06.2025-10.06.2025	
9.	Нормоконтроль	11.06.2025-15.06.2025	
10.	Перевірка на плагіат	16.06.2025	
11.	Попередній захист кваліфікаційної роботи	19.06.2025	
12.	Захист кваліфікаційної роботи	27.06.2025	

Студент _____ Пастерук П.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Голотенко О.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка комп'ютеризованої системи керування сонячними панелями з використанням технологій IoT // Кваліфікаційна робота освітнього рівня «Бакалавр» // Пастерук Петро Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СН-42 // Тернопіль, 2025 // С. 70, рис. – 11, додат. – 5, бібліогр. – 53 .

Ключові слова: Сонячна енергія, інтернет речей (IoT), двоосьовий трекер, ESP32, фоторезистор, серводвигун, система автоматичного керування.

Кваліфікаційна робота присвячена розробці комп’ютеризованої системи керування сонячною панеллю з використанням технологій Інтернету речей (IoT). Мета дослідження полягає у створенні енергоефективного двоосьового сонячного трекера на базі мікроконтролера ESP32. Реалізовано систему, що забезпечує автоматичне орієнтування панелі за Сонцем у режимі реального часу.

У першому розділі кваліфікаційної роботи подано загальну характеристику сонячної енергетики та перспективи її використання. Розглянуто сучасні типи трекерів, принципи їх роботи та класифікацію за ступенем автоматизації. Висвітлено основи функціонування фотоелектричних панелей та критерії ефективного використання. Проаналізовано переваги та недоліки існуючих систем керування на базі мікроконтролерів та IoT.

У другому розділі кваліфікаційної роботи досліджено технічні характеристики складових системи: ESP32, LDR, INA219, серводвигуни, живлення.обґрунтовано вибір архітектури системи та логіку реалізації двоосьового керування. Сформовано алгоритм стеження за Сонцем із використанням порогової логіки, P-регулювання та засобів телеметрії.

У третьому розділі розроблено прототип двоосьового трекера з функцією автоматичного позиціонування. Запропоновано підключення до хмарних платформ Blynk та MQTT для моніторингу в реальному часі. Спроєктовано електричні схеми, структурну модель системи та механічну конструкцію

поворотної рами. Протестовано працездатність усіх модулів та підтверджено ефективність системи за експериментальних умов.

У четвертому розділі кваліфікаційної роботи досліджено питання безпеки життєдіяльності та охорони праці під час експлуатації обладнання. Зокрема, проаналізовано ергономічні проблеми, що виникають у виробничому середовищі, та визначено основні заходи техніки безпеки, спрямовані на мінімізацію професійних ризиків і забезпечення безпечних умов праці.

У висновку представлені результати усієї проведеної роботи.

ANNOTATION

Development of a Computerized Solar Panel Management System Using IoT Technologies // Bachelor's Qualification Work // Pasteryuk Petro Petrovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group SN-42 // Ternopil, 2025 // 70 pages, figures – 11, appendix – 5, bibliography – 53.

Keywords: Solar energy, Internet of Things (IoT), dual-axis tracker, ESP32, photoresistor, servo motor, automatic control system.

This qualification work is devoted to the development of a computerized control system for a solar panel using Internet of Things (IoT) technologies. The aim of the study is to create an energy-efficient dual-axis solar tracker based on the ESP32 microcontroller. The system provides automatic real-time orientation of the panel toward the Sun.

In the first chapter of the qualification work, a general overview of solar energy and the prospects for its use is presented. Modern types of trackers, their principles of operation, and classification according to the level of automation are examined. The fundamentals of photovoltaic panel operation and criteria for their efficient use are highlighted. The advantages and disadvantages of existing microcontroller- and IoT-based control systems are analyzed.

The second chapter investigates the technical characteristics of the system's components: ESP32, LDRs, INA219, servo motors, and power supply. The choice of system architecture and the logic of dual-axis control implementation are substantiated. A solar tracking algorithm is formulated using threshold logic, P-regulation, and telemetry tools.

In the third chapter, a prototype of a dual-axis tracker with automatic positioning functionality is developed. Integration with cloud platforms Blynk and MQTT is proposed for real-time monitoring. The electrical schematics, structural model of the system, and mechanical design of the rotating frame are designed. All

modules were tested, and the effectiveness of the system under experimental conditions was confirmed.

In the fourth chapter, issues of occupational safety and life safety during equipment operation are studied. In particular, ergonomic issues arising in the industrial environment are analyzed, and the main safety measures aimed at minimizing occupational risks and ensuring safe working conditions are defined.

The conclusion summarizes the results of the entire conducted work.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

URL (англ. Uniform Resource Locator) – уніфікований локатор ресурсу.

IoT (англ. Internet of Things) – Інтернет речей; технологія взаємозв’язку пристроїв через інтернет для обміну даними.

ESP32 – 32-бітний мікроконтролер із вбудованими модулями Wi-Fi та Bluetooth.

PWM (англ. Pulse Width Modulation) – широтно-імпульсна модуляція; спосіб керування потужністю шляхом зміни тривалості імпульсу.

LDR (англ. Light Dependent Resistor) – фоторезистор; резистор, опір якого залежить від рівня освітлення.

PID (англ. Proportional-Integral-Derivative) – пропорційно-інтегрально-диференційний регулятор.

ADC (англ. Analog-to-Digital Converter) – аналого-цифровий перетворювач.

Wi-Fi (англ. Wireless Fidelity) – технологія бездротового зв’язку.

RTC (англ. Real-Time Clock) – годинник реального часу.

MPPT (англ. Maximum Power Point Tracking) – відстеження точки максимальної потужності сонячної панелі.

PV (англ. Photovoltaic) – фотогальванічний; стосується перетворення світлової енергії в електричну.

UART (англ. Universal Asynchronous Receiver-Transmitter) – універсальний асинхронний приймач-передавач.

ADC value – значення сигналу після аналого-цифрового перетворення.

GUI (англ. Graphical User Interface) – графічний інтерфейс користувача.

Вт·год – ват-година; одиниця вимірювання електричної енергії.

мА·год – міліампер-година; одиниця ємності акумулятора.

PWM-сигнал – сигнал із широтно-імпульсною модуляцією.

ESP8266 – мікроконтролер з Wi-Fi, що часто використовується разом з Arduino.

Arduino IDE – середовище розробки для програмування мікроконтролерів Arduino.

ThingSpeak – IoT-платформа для збирання та візуалізації даних із сенсорів.

Blynk – хмарна платформа для побудови мобільного інтерфейсу для IoT-проєктів.

MG996R – модель сервоприводу з металевими шестернями.

ACS712 – датчик вимірювання сили струму.

ESP32 DevKit V1 – плата розробника з мікроконтролером ESP32.

ФВ – фотоелектричний (синонім до PV).

ШІМ – широтно-імпульсна модуляція (PWM).

GPIO (англ. General Purpose Input/Output) – універсальний вхід/вихід мікроконтролера.

DAC (англ. Digital-to-Analog Converter) – цифро-аналоговий перетворювач.

ФВ-система – фотоелектрична система, система генерації електроенергії з сонячної панелі.

ККД – коефіцієнт корисної дії.

ЗМІСТ

ВСТУП

Сонячна енергія є одним із найперспективніших відновлюваних джерел [1], здатних задовольнити значну частину світових енергетичних потреб. Потік енергії від Сонця, що досягає поверхні Землі, оцінюється приблизно в *1.8*10*11* МВт, що на порядки перевищує сучасне сумарне споживання енергії людством [2]. Водночас ефективність перетворення сонячного випромінювання у електрику за допомогою фотогальванічних панелей залежить від кута падіння променів. Якщо панель нерухома і встановлена під фіксованим кутом, значна частина потенційної енергії втрачається через відхилення Сонця від оптимального кута протягом дня. Для максимізації вихідної потужності панель повинна постійно орієнтуватися перпендикулярно до сонячних променів [3]. Тому у сучасних системах усе ширше застосовуються сонячні трекери – механізми активного наведення панелей на Сонце [4].

Актуальність теми. Комбінація технологій відстеження положення Сонця з можливостями IoT відкриває шлях до створення інтелектуальних сонячних установок, що автоматично оптимізують свою роботу та надають користувачам зручні засоби контролю. Такі комп’ютеризовані системи керування сонячними панелями здатні підвищити ефективність використання сонячної енергії, знизити втрати від неоптимального положення панелей та спростити обслуговування установок. За даними досліджень, оснащення сонячних трекерів модулями IoT дозволяє відслідковувати виробіток енергії та стан обладнання в режимі реального часу, що забезпечує своєчасне виявлення несправностей і підвищує загальну ефективність системи[4-6, 35]. Застосування віддаленого моніторингу й керування особливо цінне для великих сонячних станцій: IoT-методи дають змогу автоматизувати контроль сотень і тисяч панелей, збираючи дані про їхню роботу і вживаючи заходів для профілактики збоїв. Таким чином, розробка системи, що поєднує механізми орієнтації сонячних панелей та IoT-технології, є актуальним завданням на перетині галузей відновлюваної енергетики, електроніки та комп’ютерних наук[7, 36].

Мета і задачі дослідження. Метою нашого дослідження є розробка і дослідження комп'ютеризованої системи керування сонячними панелями з використанням технологій IoT. Для досягнення поставленої мети в роботі вирішуються такі завдання:

- проведення аналізу сучасних методів орієнтації сонячних панелей і застосування IoT в енергетиці;
- проектування архітектури апаратно- програмної системи керування на базі мікроконтролера (Arduino/ESP32) з датчиками і приводами для наведення панелі;
- розробка програмного забезпечення для контролера та хмарного сервісу для моніторингу;
- виготовлення експериментального зразка системи, проведення випробувань та оцінка її ефективності у порівнянні зі статичною панеллю.

Практичне значення одержаних результатів.

Практичне значення роботи полягає у створенні діючого прототипу “розумної” сонячної панелі, який може бути використаний як основа для комерційних систем автономного електропостачання будинків, датчиків тощо. Система побудована на доступних компонентах (контролер ESP32, датчики освітленості, модулі зв’язку) і може бути відтворена з невеликими витратами. Результати експериментів підтверджують збільшення вироблення електроенергії при використанні відстеження Сонця, а впроваджена IoT-підсистема забезпечує зручний контроль та збір статистики для подальшого аналізу роботи панелі. Розроблене рішення може бути масштабоване – від одиничної панелі до масиву – шляхом об’єднання декількох контролерів у мережу і передачі даних на єдиний сервер. Це відкриває можливості для створення розподілених систем генерування з інтелектуальним керуванням і оптимізацією в режимі реального часу.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ КОМП'ЮТЕРИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ СОНЯЧНИМИ ПАНЕЛЯМИ

1.1. Активні системи орієнтації сонячних панелей: типи та принципи роботи

Для підвищення продуктивності сонячних електростанцій панелі оснащують системами активного наведення на Сонце, що компенсують добовий та сезонний рух світила[4,8]. Розрізняють одноосьові та двохосьові трекери за кількістю ступенів свободи поворотного механізму. Одноосьовий трекер зазвичай обертає панель у горизонтальній площині зі сходу на захід, слідкуючи за сонячним азимутом протягом дня. Кут нахилу панелі при цьому фіксується під оптимальним середньосезонним кутом (близьким до широти місцевості) або може вручну змінюватися кілька разів на рік. Дослідження показали, що правильно реалізований одноосьовий трекер може давати на ~16-28% більше енергії, ніж статична панель з оптимальним кутом нахилу. Наприклад, вертикальні осьові трекери (обертання навколо вертикальної осі) у високих широтах забезпечують приріст ~16-28% проти фіксованих панелей, тоді як горизонтальні осьові можуть дати ~28-34% виграшу в тропіках. В середньому одновісне стеження забезпечує ~20-25% додаткової генерації, витрачаючи при цьому не більше 5-8% виробленої енергії на привод та управління [9-11].

Двохосьові трекери (*dual-axis*) дозволяють панелям обертатися як в напрямку схід-захід, так і змінювати кут підйому (нахил до горизонту), слідкуючи за висотою Сонця на небі. Таким чином панель протягом дня постійно підтримується майже точно перпендикулярно до сонячних променів. Це дозволяє захопити максимальну кількість променів навіть вранці і надвечір, а також в зимовий період, коли Сонце низько над горизонтом. Двохосьові системи складніші та дорожчі, але дають найбільший приріст енергії – за різними дослідженнями, від 30% до 50% проти нерухомих установок [12-14].

За способом реалізації стеження розрізняють активні та пасивні трекери [15]. *Пасивні* трекери використовують фізичні явища, наприклад теплове розширення речовин, для зміщення центру ваги і повороту панелі за Сонцем без електроніки. Вони прості, але менш точні і наразі мало поширені. *Активні* трекери оснащені датчиками та електроприводами, що керуються контролером за заданим алгоритмом. За типом алгоритму можна виділити системи з датчиками випромінювання та системи з астрономічним (програмним) керуванням.

У сенсорних трекерах встановлені фотодатчики (найчастіше фоторезистори LDR або фотодіоди), які безперервно заміряють освітленість з різних сторін панелі. Контролер порівнює сигнали від сенсорів, визначаючи куди потрібно повернути панель, щоб вирівняти освітленість. Типово використовують пару або квартет LDR, розташованих у так званому конфігураційному хресті: датчики розділені перегородками і реагують на перевищення освітлення з певного напрямку. Коли Сонце зміщується, один з датчиків опиняється в тіні перегородки, і контролер отримує різницю сигналів – це є командою на поворот панелі до того моменту, поки всі датчики знову не дадуть рівний сигнал (панель спрямована точно на Сонце). Подібні системи досить прості й ефективні [16, 41].

Альтернативою є часові (астрономічні) алгоритми – коли положення Сонця розраховується на основі географічних координат, дати і часу. Відомо, що азимут і висота Сонця в небі можуть бути визначені з достатньою точністю за формулами небесної механіки або отримані з інтернет- сервісів (наприклад, *NASA Sunrise/Sunset API)*. У таких системах датчики освітлення не потрібні – контролер “знає”, де має бути Сонце, і повертає панель у розраховане положення, навіть якщо Сонце за хмарами. Перевагою є відсутність впливу локальних перешкод (тіней, бруду) на процес стеження. Проте неточність годинника або неточне виставлення географічних координат може призвести до систематичного зміщення. Крім того, такі системи не реагують на фактичну освітленість: якщо Сонце закрите хмарою, астрономічний трекер все одно буде слідувати за його геометричним положенням, хоча від цього немає приросту

енергії. Проте сучасні підходи вдосконалюють алгоритми: зокрема, реалізують розкладні (schedule-based) траєкторії, коли повороти здійснюються з певним кроком (наприклад, кожні 15 хвилин) замість безперервного руху, щоб економити енергію привода. Також впроваджують режим *backtracking* – невелике відхилення від точного кута при низькому Сонці, аби запобігти затінюванню сусідніх панелей у масиві. Вважається, що правильно налаштований часовий алгоритм може майже не поступатися сенсорному по зібраній енергії. Більш того, згідно з деякими даними, за різких змін погоди астрономічний підхід навіть перевершує LDR-систему [16, 38].

Окрім традиційних підходів (сенсори vs. астрономічні формули), існують гібридні системи, що поєднують обидва методи. Наприклад, трекер може в ясну погоду працювати за сигналами датчиків, а при втраті сонячного сигналу – переходити на режим прогнозування положення Сонця.

Точність і ефективність стеження. Ефективність трекера оцінюється співвідношенням виграшу в енергії до витрат на приводи та втрат, зумовлених неточністю наведення. В літературі зазначається, що прості одноосьові трекери здатні захопити ~85-95% енергії, яку дає ідеальний двоосьовий стежувач, але конструктивно простіші і споживають менше енергії на рух [15, 17, 39]. Двоосьові забезпечують майже повний збір випромінення (до 100% прямої радіації і значну частину розсіяної). Проте в похмурі дні різниця між трекером і нерухомою панеллю зменшується, адже перевага орієнтації менша при переважно дифузному світлі. Тому деякі розумні системи враховують рівень освітленості: наприклад, можуть на час густої хмарності переводити панель у горизонтальне положення для рівномірного збору розсіяного світла або мінімізації вітрового навантаження. В сучасних дослідженнях також приділяється увага оптимізації енергоспоживання самих трекерів – щоб енергія, витрачена на обертання моторів, була значно меншою за додатково зібрану енергію [18]. Багато розробок використовують енергоощадні приводи, а рух здійснюють періодично (крокове).

Окрім традиційних фотосенсорів, часто проводяться експерименти з

іншими типами датчиків для підвищення точності відстеження. Наприклад, застосовуються *ультрафіолетові (UV) сенсори* та навіть *волоконно-оптичні датчики*. Останні дозволяють точно вимірювати показники, такі як розподіл освітленості чи температуру панелі по поверхні. Зокрема, волоконно-оптичні решітки Брегга (FBG-датчики) використовувалися для моніторингу температури фотоелементів у реальному часі з високою точністю, що може бути інтегровано в систему охолодження панелей. Хоча такі датчики більше стосуються контролю стану панелі, а не самого стеження, їх можна вважати частиною IoT-інфраструктури "розумної" панелі – вони забезпечують додаткові дані для оптимального керування (наприклад, сигналізують про необхідність охолодження чи очищення панелі).

Окремим напрямом розвитку є використання комп'ютерного зору при наведені [19]. Замість простих LDR датчиків деякі системи оснащуються камерами та обчислюють положення Сонця за зображенням неба.

Підсумовуючи, технології активного стеження за Сонцем сьогодні різноманітні: від простих одновісних механізмів з LDR-датчиком і Arduino-контролером до складних комбінованих систем з камерами, GPS, інклінометрами і нейромережевими алгоритмами прогнозування погоди.

Кожна система прагне збільшити генерацію енергії, мінімізуючи при цьому витрати та складність. Найкращі результати показують двовісні трекери, що автоматично адаптуються до умов: за рахунок IoT-підключення вони можуть отримувати, наприклад, дані прогнозу погоди або глобальну інформацію про оптимальні кути нахилу і таким чином коригувати стратегію стеження [15, 20]

Розвиток таких "розумних" трекерів тісно пов'язаний із впровадженням концепції Інтернету речей в енергетику, що розглядається в наступних розділах.

1.2 Концепція Інтернету речей (IoT) та її застосування

Інтернет речей (IoT) – це сучасна концепція обчислювальної мережі, де учасниками обміну даними є не лише люди і сервери, але й численні пристрої,

датчики, прилади, машини. На відміну від традиційного Інтернету, де основними джерелами даних були люди (введення інформації вручну), в IoT дані генеруються самими *“речами”* (пристроями) автоматично. Класичне визначення характеризує IoT як відкриту глобальну мережу об’єктів, наділених інтелектом, що самостійно організовуються і обмінюються інформацією між собою, реагуючи на зміни довкілля. Ці об’єкти можуть мати унікальні ідентифікатори та доступ до мережі для передачі даних. Технологічний розвиток IoT став можливим завдяки здешевленню сенсорів, появі енергоефективних бездротових модулів зв’язку та хмарних обчислювальних платформ для збору і обробки великих масивів даних [21, 37].

Архітектура IoT-систем зазвичай багаторівнева. На *нижньому рівні* знаходяться фізичні пристрої – *вузли IoT*, оснащені сенсорами і актуаторами, а також мікроконтролерами для збору даних і виконання локальних команд. В нашому випадку такими пристроями є контролери сонячних панелей з підключеними датчиками освітленості, струму, напруги тощо. Вузли можуть мати модулі бездротового зв’язку (Wi-Fi, Bluetooth, ZigBee, LoRaWAN, стільниковий модуль 3G/4G і т.д.) або дротові інтерфейси (Ethernet, RS485) для з’єднання з мережею [21, 22]. *Середній рівень* – це мережеві шлюзи і сервери, які отримують дані від вузлів. Наприклад, домашній Wi-Fi маршрутизатор або спеціальний шлюз збирає дані з десятків датчиків по протоколу ZigBee і передає далі в Інтернет. *Верхній рівень* – хмарні сервіси і додатки користувача. У хмарі дані можуть накопичуватися в базах, опрацьовуватися (в тому числі алгоритмами штучного інтелекту для пошуку залежностей або аномалій), а результати надаються користувачу через веб-інтерфейс або мобільний додаток. Користувач, у свою чергу, може надсилати керуючі команди на пристрої через цю ж хмарну платформу. Таким чином, IoT забезпечує *замкнений цикл* “дані-вирішення-дія”: датчики передають інформацію, яка аналізується, і на основі цього надходять команди назад на пристрої для оптимізації їх роботи.

В контексті енергетики IoT набуває все більшого значення. *IoT в електроенергетиці* дозволяє створювати інтелектуальні енергосистеми (smart

grids), де генерація, розподіл і споживання енергії контролюються й оптимізуються в реальному часі. Наприклад, датчики на лініях електропередач моніторять навантаження і погодні умови, розумні лічильники у споживачів передають профіль споживання щогодини, а система керування генеруючими потужностями (наприклад, сонячними і вітровими станціями) може оперативно реагувати на зміну попиту. У сфері відновлюваної енергетики IoT вирішує одразу декілька завдань:

- Моніторинг виробітку та стану обладнання. Сенсори на сонячних панелях (температури, освітленості, струму, напруги) дозволяють відстежувати продуктивність кожної панелі або стрінгу.

- Дистанційне керування і оптимізація. IoT-система може не тільки збирати дані, а й надсилати команди. У випадку сонячних електростанцій це може бути керування інверторами, перемикання режимів роботи або, як у нашій задачі, керування приводами трекерів. Через інтернет-платформу оператор може, наприклад, примусово встановити панелі в безпечне горизонтальне положення під час шторму або ж змінити алгоритм стеження (перейти в режим економії енергії тощо). Відомий приклад – платформа Blynk, що дозволяє через смартфон посилати команди на мікроконтролери Arduino/ESP8266 [19].

– Інтелектуальний аналіз та прогнозування. Збираючи великі обсяги даних з безлічі датчиків, IoT-платформа може застосовувати алгоритми машинного навчання для виявлення закономірностей. Наприклад, на основі історичних даних генерування та метеопрогнозу можна прогнозувати вихід сонячної станції на найближчі години і завчасно коригувати роботу інших джерел енергії в мережі [22, 39].

У віддалених районах, де немає стабільного інтернет-покриття, для IoT-моніторингу застосовують технології далекого радіозв'язку малих швидкостей, зокрема LoRaWAN. Цей протокол забезпечує передачу даних на відстань до 5-15 км при мінімальному енергоспоживанні, що ідеально для розподілених сонячних станцій у сільській місцевості.

Підсумовуючи, IoT-технології в сонячній енергетиці забезпечують:

- Прозорість та оперативність контролю – власник або оператор може будь-коли отримати актуальну інформацію про генерацію, стан акумуляторів, параметри інвертора тощо через веб-інтерфейс або додаток.

- Підвищення надійності – система сама сигналізує про відхилення (падіння виробітку, перегрів панелі, вихід з ладу окремих елементів), що дозволяє швидко реагувати і проводити обслуговування до виникнення серйозних проблем.

- Оптимізація роботи – збір статистики дає змогу проводити глибокий аналіз ефективності, порівнювати режими, визначати оптимальні кути нахилу для різних сезонів.

– Інтеграцію з енергосистемою – дані від безлічі сонячних установок можуть передаватися до центральної смарт-грід системи, яка балансує навантаження. З боку окремих панелей IoT дозволяє вводити обмеження генерації або навпаки – максимально використовувати панелі у певні години згідно команд від енергомережі (концепція *VPP – віртуальної електростанції*).

Таким чином, впровадження IoT в керування сонячними панелями відкриває новий рівень *автоматизації* та *інтелектуалізації* відновлюваних джерел енергії. У наступному підрозділі розглянемо практичні приклади реалізації таких систем на базі доступних апаратних платформ (Arduino, ESP32) та проаналізуємо компоненти, з яких складається комп'ютеризована система керування сонячною панеллю.

1.3 Огляд сучасних реалізацій "розумних" сонячних трекерів

За останні 5-10 років з'явилося багато проектів, присвячених поєднанню сонячних трекерів з технологіями Інтернету речей.

Одним з перших був проєкт 2016-го року «*смарт-модуль моніторингу і керування* для сонячної панелі з використанням IoT» [22, 40]. Система складалася з контролера, підключеного до датчиків струму/напруги, і могла передавати дані про генеровану потужність на віддалений сервер. Крім того, був реалізований

дистанційний вимикач навантаження, що дозволяло віддалено від'єднувати панель при потребі. Цей проєкт продемонстрував концепцію повного онлайн-контролю над окремою панеллю: оператор через веб-інтерфейс бачив в реальному часі напругу, струм, потужність і міг вимкнути або ввімкнути панель.

Цікавими були проєкти, котрі фокусувалися саме на *IoT-трекерах*. Авторами [ ] розроблений двовісний сонячний трекер на Arduino з можливістю моніторингу через інтернет. Установка використовувала 4 LDR-датчики для визначення положення Сонця та два сервоприводи для повороту панелі. Родзинкою було те, що кут азимуту і висоти Сонця також обчислювався за даними з веб-сервісів, і в разі невідповідності датчиків (наприклад, в хмарний день) система могла переходити до режиму за розрахунком кутів, що підвищувало точність. Зібрані дані (поточний кут, час, стан панелі) передавалися через Wi-Fi на веб-сторінку, де користувач міг їх переглядати. Такий підхід поєднав локальний інтелект контролера з глобальними даними Інтернету, зробивши трекер більш автономним і точним.

Цікавий приклад *енергоефективного IoT-трекера* описано в роботі [23]. Авторами розроблений двовісний трекер з акцентом на мінімізацію витрат енергії та швидкодію. Система на базі мікроконтролера збирала дані LDR-датчиків, а також мала підключення по Wi-Fi для надсилання даних на сервер. Однією з особливостей була висока швидкість реакції – поворот здійснювався за ~0,2 с після зміни положення Сонця. Дані за повну добу записувалися в базу даних, що дозволяло проводити аналіз ефективності алгоритму. В цій роботі наголошується, що використання Wi-Fi з'єднання практично не вплинуло на енергоспоживання контролера, але дало величезну цінність у вигляді можливості спостереження за системою онлайн. Фактично продемонстровано прототип, близький до промислових рішень: двовісний трекер із повним телеметричним супроводом. За рахунок оптимізованого алгоритму (уникав зайвих рухів в сутінках, повертав панель у горизонталь ночами тощо) їх система споживала менше 1 Вт на власні потреби, що мізерно мало порівняно з додатковою генерацією.

Нещодавно [24] проведено дослідження *інтелектуального трекера* на базі ESP32, де була інтегрована фільтрація вимірів і PID-контроль для максимальної точності. Контролер ESP32, окрім бездротового інтерфейсу (Wi-Fi), мав достатню потужність для виконання алгоритму Калмана – цим згладжувався шум LDR-датчиків. Комбінація фільтру Калмана та PID-регулятора забезпечила дуже плавне й точне стеження, що особливо позитивно вплинуло за частково хмарної погоди. Покращена динаміка трекера дала суттєвий виграш: як зазначалося, добовий виробіток зріс на ~43% проти панелі без стеження. Система була оснащена також датчиками напруги і струму, щоб оцінювати реальний приріст потужності, і результати передавалися через платформу Blynk на смартфон. Таким чином, робота демонструє, що навіть бюджетний контролер може виконувати досить складні алгоритми і взаємодіяти з IoT- сервісами, утворюючи "кишенькову" SCADA для сонячної панелі.

Інший приклад практичної реалізації, де був створений прототип IoT-сонячного трекера з інтерфейсом Blynk [25]. Установка базувалась на платі NodeMCU (ESP8266) і мала фоторезисторний сенсор, а також датчик струму INA219 для вимірювання вироблюваної потужності. Через мобільний додаток Blynk користувач міг спостерігати за показами напруги, струму, потужності в режимі реального часу. Проведено цікаве випробування: дві панелі – одна з трекером, інша нерухома – працювали з різним навантаженням (вентиляторами 5В і 12В) для порівняння ефективності. В результаті було підтверджено, що панель на трекері виробляла більше енергії, особливо під більшим навантаженням, і після завершення стеження (в кінці дня) її напруга/струм були приблизно на 30% вищими, ніж у нерухомої панелі. Цей експеримент наочно продемонстрував користь трекера, а також можливості IoT-додатку: дані усіх вимірів візуалізувалися на веб-дашборді та в смартфоні, забезпечуючи зручний інструмент для аналізу роботи системи.

З огляду на наведені приклади можна зробити висновок, що сучасні тенденції в розвитку сонячних панелей і трекерів йдуть шляхом *інтелектуалізації та мережевої інтеграції*. Практично всі новітні системи

передбачають наявність мікроконтролера або вбудованого комп'ютера, підключеного до Інтернету. Це дозволяє: – реалізувати гнучкі складні алгоритми керування (фільтрація, прогнозування, оптимізація), – оперативно оновлювати прошивку та алгоритми (віддалено завантажуючи оновлення), – інтегрувати систему в єдину інформаційну платформу (наприклад, для керування цілим полем панелей централізовано).

У цій роботі ми будемо спиратися на досвід згаданих реалізацій для побудови власної системи. Зокрема, використаємо недорогий контролер з підтримкою Wi-Fi, фотодатчики як основний сенсорний елемент, а також програмну платформу (Blynk або аналог) для візуалізації даних. В наступному розділі детально розглянемо проектування нашої системи – від вибору компонентів до розробки алгоритмів і структури програмного забезпечення.

1.4 Висновки до першого розділу

Подано загальні теоретичні відомості щодо функціонування сонячних фотоелектричних систем, зокрема розкрито фізичні принципи перетворення сонячної енергії в електричну, визначено ключові характеристики сонячних панелей, а також наведено порівняльну характеристику типів сонячних елементів, що застосовуються у практиці.

Розглянуто основні різновиди сонячних трекерів – одноосьових і двоосьових, їх переваги та недоліки, принципи побудови й технічні рішення, що забезпечують ефективне орієнтування фотоелектричних модулів відносно положення Сонця протягом дня та року.

Висвітлено роль і потенціал технологій Інтернету речей (IoT) у контексті модернізації енергетичних систем, наведено приклади застосування IoT для дистанційного моніторингу та керування енергетичними пристроями.

Проаналізовано сучасний стан розвитку інтелектуальних систем керування в сонячній енергетиці, охарактеризовано апаратно-програмні засоби, що найчастіше використовуються у подібних проектах (ESP32, Arduino, сенсори,

серводвигуни, бездротові модулі), а також окреслено основні науково-технічні проблеми та тенденції подальшого вдосконалення таких систем..

РОЗДІЛ 2. ПРОЄКТНА ЧАСТИНА

2.1 Постановка вимог до системи та загальна архітектура

Перед початком розробки комп'ютеризованої системи керування сонячними панелями визначимо основні вимоги до неї. Система повинна забезпечувати:

- *Автоматичне стеження за Сонцем. П*оворот сонячної панелі протягом дня таким чином, щоб максимізувати інсоляцію на її поверхні. Необхідна точність наведення – не гірше кількох градусів, щоб втрати генерації були мінімальні (бажано <5%). Під вечір та вночі панель має повертатися в положення очікування наступного дня (наприклад, на схід з помірним кутом підйому).

- *Мінімальне енергоспоживання на власні потреби*. Трекер не повинен "з'їдати" значну частину виробленої енергії. Мотори і контролер мають споживати якомога менше; бажано, щоб на керування витрачалося не більше 5-10% енергії, виробленої панеллю.

- *Інтеграція з IoT-платформою. С*истема має передавати ключові параметри (кут панелі, рівень освітленості, напруга/струм, стан приводу) до віддаленого сервера або хмарного додатку. Формат – повідомлення через Wi-Fi (наприклад, HTTP або MQTT протокол). Частота передачі – кілька разів на хвилину або при зміні стану. Також бажано реалізувати можливість прийому команд з сервера (напр., примусове повернення в певне положення).

- *Простота відтворення та низька вартість*. Для практичної цінності проєкту, обрані компоненти мають бути дешевими і доступними. Тому упор робимо на використання масових модулів: мікроконтролер Arduino або ESP32, сервоприводи або мотор-редуктори для повороту, датчики типу LDR. Схема повинна бути нескладною у збиранні та налагодженні.

- *Безпечність і надійність*. Врахувати граничні положення панелі, не допускати механічного перевантаження приводів. Додати кінцеві вимикачі або програмні обмеження кутів. При втраті зв'язку чи збою система повинна

переходити в безпечний режим (наприклад, зупинка руху або повернення в горизонталь).

Враховуючи ці вимоги, розробимо загальну архітектуру системи (блок-схему). Вона складатиметься з таких основних частин:

1. Механічна підсистема: рама, що утримує сонячну панель, та приводи для повороту по двох осях. Ми оберемо типову конфігурацію з двома ступенями свободи: азимутальний оберт (панель обертається навколо вертикальної осі) та зміна кута нахилу (навколо горизонтальної осі, закріпленої на поворотній платформі). Як приводи використовуються або два *серводвигуни* з достатнім моментом, або комбінація редукторного двигуна (постійного струму або крокового) для однієї осі та сервоприводу для іншої. Наприклад, популярним рішенням є пара: поворотна платформа на основі автомобільного двірникового мотору (для азимута) і лінійний актуатор або сервопривод для нахилу. У нашому прототипі, враховуючи малі розміри панелі (скажімо, 50 Вт модуль), можна використати серводвигуни стандарту MG996R або більш потужні RDS3115 (металеві шестерні) – вони забезпечують момент ~10-15 кг-см, достатній для невеликої панелі.

2. Датчики положення Сонця: вибираємо 4 фото резистори LDR, встановлені по кутах панелі з перегородками між ними (хрестоподібний козирок). Така конфігурація дозволяє визначати, в який бік змістити панель: якщо Сонце не по центру, один з LDR буде більш освітлений, інший – менш. LDR вигідні тим, що дуже дешеві і прості у підключенні (аналоговий вхід контролера). Їх недолік – нелінійність і залежність від температури, але для нашого завдання високої точності не потрібно, а градуйовка здійснюється емпірично. Альтернативою могли б бути фотодіоди або фототранзистори, які швидше реагують, але вони потребують трохи складнішого опрацювання сигналу. Зупинимося на LDR як перевіреному варіанті.

3. Інші сенсори: щоб реалізувати IoT-моніторинг енергетичних параметрів, додамо датчик струму/напруги на виході сонячної панелі. Хороший вибір – модуль INA219. INA219 підключається по шині I^2C і дозволяє вимірювати

напругу до ~26 В та струм до 3 А з високою точністю, що достатньо для панелі 50-100 Вт. Це дасть нам дані про поточну генеровану потужність. Також варто мати *датчик кута нахилу* – наприклад, двовісний акселерометр (або простіше – схил з кінцевими вимикачами) для калібрування положення “нулю”. Але щоб спростити, можемо обійтися без цього: початкове положення при ввімкненні задаємо, привівши панель у відомий кут вручну або за допомогою кінцевих вимикачів.

4. Обчислювальний модуль: серце системи – мікроконтролерна плата. Тут є два основні кандидати:

- Arduino Uno/Mega (ATmega328/2560) – класичний вибір, багато бібліотек, простота. Проте самі Arduino не мають вбудованого Wi-Fi. Довелося б додавати модуль зв’язку (ESP8266 або SIM800 для GSM). Відомо, що Arduino успішно застосовувалися в трекерах.

- ESP32 (DevKitC або аналог) – сучасний 32-розрядний контролер з вбудованим Wi-Fi та Bluetooth. Має значно більшу продуктивність і пам’ять, подвійний ядро MCU, багато пінів. ESP32 зручний тим, що об’єднує обчислення і інтернет- підключення “в одному флаконі”. Багато новітніх проектів перейшли саме на ESP32. В нашому проєкті доцільно обрати ESP32, оскільки він дозволить безпосередньо підключитися до Wi-Fi мережі і взаємодіяти з хмарним сервером без додаткових модулів. До того ж, його швидкодія дає перспективу реалізації складніших алгоритмів (фільтрація, PID) і обробки мережевих протоколів одночасно. Розробка програм для ESP32 може виконуватися в знайомому середовищі Arduino IDE, що спрощує задачу.

5. Модуль зв'язку: як зазначено, вбудований Wi-Fi модуль ESP32 використаємо для виходу в Інтернет через локальний роутер. Протокол комунікації: виберемо MQTT (Message Queuing Telemetry Transport) або HTTP REST API, залежно від сервера. MQTT більш легковаговий і підходить для IoT, тим більше є готові хмарні брокери (як test.mosquitto.org чи cloudmqtt) та бібліотеки. Плануємо, що контролер буде публікувати повідомлення типу {panel_id, timestamp, voltage, current, angle_alt, angle az} на MQTT-топік, а

отримувати може, наприклад, повідомлення типу {command: "reset"|"park"|"manual", value: ...} для зовнішнього управління.

В якості *IoT-платформи* для відображення даних можна використовувати Blynk, бо вона проста в налаштуванні і має гарний мобільний інтерфейс. Blynk надає готові віджети (графіки, індикатори) і дозволяє налаштувати обмін даними через свій хмарний сервер. Безкоштовна версія Blynk матиме деякі обмеження, але для одного пристрою цілком придатна. Тож, з огляду на [ ], інтегруємо нашу систему з Blynk: на телефоні буде дашборд з показниками і кнопками керування.

6. Блок живлення: система включає електродвигуни і електроніку, отже потрібні відповідні напруги живлення. Типово, серводвигуни працюють від ~5-6 В постійного струму, крокові чи DC- мотори можуть потребувати 12 В (залежить від моделі). Мікроконтролер ESP32 має свій стабілізатор 5В->3.3В, тому може живитися від 5 В. Якщо сонячна панель використовується без акумулятора, доведеться передбачити стабільне живлення трекера від самої панелі (коли є сонце) та, можливо, від буферного акумулятора або мережі у нічний час. У прототипі простіше використовувати окреме джерело – наприклад, мережевий адаптер 12 В, від нього через DC-DC перетворювач отримати 5 В для електроніки і безпосередньо 12 В на двигуни (якщо будемо застосовувати 12-вольтові редуктори). В польових умовах це була б акумуляторна батарея з сонячним контролером заряду.

На основі перелічених компонентів можна скласти схему (рис. 2.1).

В ній:

- Сонячна панель встановлена на двовісній рамці.

- Приводи: мотор повороту по азимуту (ось вертикальна) та мотор підйому (ось горизонтальна). Підключені до контролера через драйвер двигунів (наприклад, L298N для DC моторів або драйвер сервоприводу).

- Блок датчиків: 4 LDR, об’єднані у форму хреста – підключені до аналогових входів ESP32 (через дільники напруги).

Модуль INA219 під’єднаний до виходу панелі і шини I^2C контролера (виводи SDA, SCL).

- Контролер ESP32: отримує сигнали LDR і INA219, виконує алгоритм стеження, віддає ШІМ сигнали на драйвери моторів. Також по Wi-Fi з’єднується з роутером і передає дані в Інтернет.

- На боці програмного забезпечення: хмарний сервер Blynk (або власний MQTT-брокер) отримує телеметрію. Користувач через смартфон (Blynk App) або ПК (веб-дашборд) може бачити ці дані і надсилати команди.

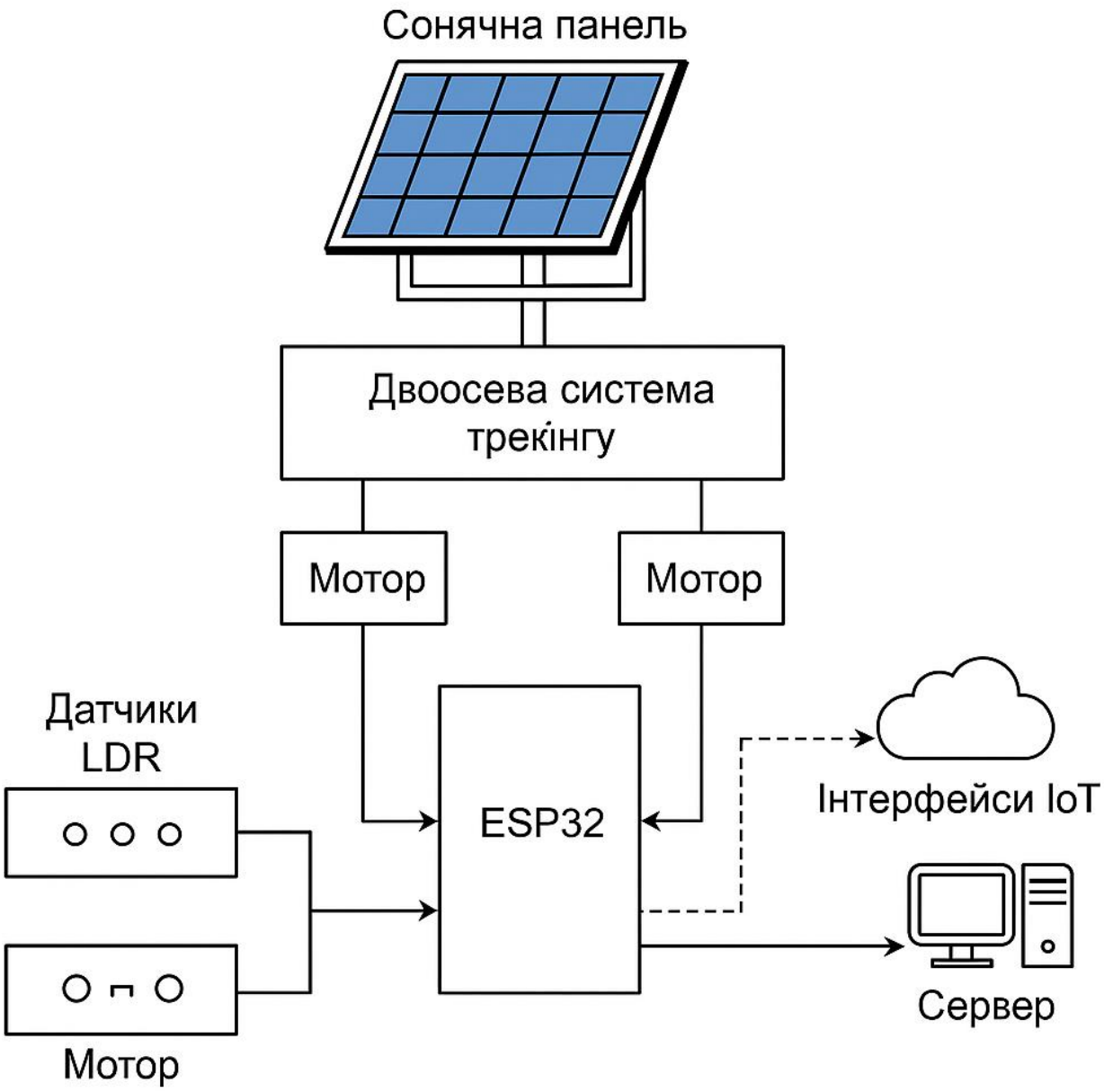

Рисунок 2.1 – Структурна схема системи

Зазначимо, що обраний контролер ESP32 дозволяє потенційно розширити систему. Наприклад, можна підключити GPS-модуль для автоматичного

визначення координат місця встановлення (для астрономічного алгоритму), або датчики температури панелі, вологості повітря тощо, щоб комплексно оцінювати умови роботи. Також ESP32 дає змогу реалізувати локальну веб-сторінку (вбудований веб-сервер) для керування без хмари – корисно як резервний варіант. У цій роботі ми зосередимось на основному функціоналі: стеження та IoT-моніторинг, але залишаємо можливості для розширень.

2.2 Вибір та підключення компонентів апаратної частини

Під час вибору апаратного забезпечення важливо враховувати вимоги до функціональності, енергоефективності та компактності, сформульовані на попередніх етапах проєктування. Правильно підібрані компоненти визначають не лише стабільність і автономність роботи системи, а й її здатність ефективно реагувати на змінні зовнішні умови [25]. Апаратна складова включає базові елементи, зокрема мікроконтролер, сенсорні модулі, приводи, джерела живлення та елементи зв’язку. Усі ці компоненти мають бути узгоджені між собою за параметрами напруги, споживаної потужності та швидкодії. Особливу увагу слід приділити енергоспоживанню, оскільки надмірне навантаження може зменшити загальну ефективність системи, особливо в автономному режимі. Крім того, компактність апаратної частини забезпечує зручність монтажу, мобільність та можливість інтеграції в різні конструктивні формати, що є важливим фактором при розробці макетної моделі.

2.2.1 Контролер ESP32 DevKit V1.

Вибрано модуль на мікроконтролері ESP32 з 30 виводами (рисунок 2.2). Він має 2 ядра Xtensa LX6 (240 МГц), 520 кБ оперативної пам’яті, 4 МБ флеш. На борту – Wi-Fi 802.11b/g/n та Bluetooth 4.2 [26]. Для нашої задачі важливо, що модуль підтримує аналогові входи (до 18 каналів ADC, 12-біт роздільна здатність) – будемо читати LDR і, за потреби, напругу з датчиків. Також є

достатньо ШІМ виходів для керування моторами. Програмування здійснюємо через USB (UART) із комп’ютера. Працювати будемо у середовищі Arduino IDE з встановленим ядром ESP32 – це надасть доступ до функцій Arduino (analogRead, analogWrite, WiFi тощо) і до бібліотек Blynk, INA219, MQTT.

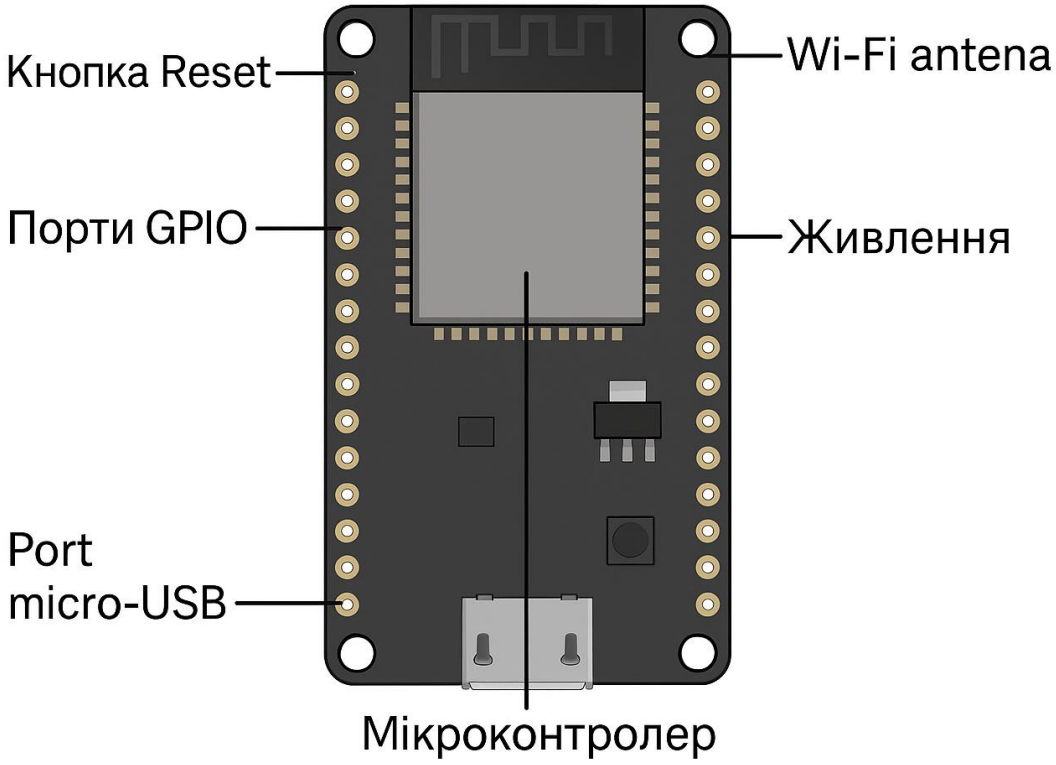

Рисунок 2.2 – Контролер ESP32 DevKit V1

Під час монтажу важливо правильно організувати живлення: ESP32 потребує стабільних 5 В на пін Vin (або 3.3 В на пін 3V3). Живитимемо його від стабілізованого 5 В джерела.

2.2.2 Датчики освітленості LDR

Використаємо 4 однакових фоторезистори типу GL5528 (номінальний опір ~5-10 кОм при 10 лк). Кожен LDR підключимо в схемі дільника напруги з постійним резистором ~10 кОм: один вивід LDR до +3.3 В, другий до резистора, і на землю; вузол між ними – на аналоговий вхід ESP32 [27]. Така конфігурація

дасть змогу отримувати різницю потенціалів залежно від освітлення. Щоб збільшити динамічний діапазон і чутливість, підберемо резистор під темновий опір LDR (можливо 100 кОм, якщо LDR має 1 MQ в темряві). Але оскільки система буде працювати в світлий час, 10 кОм достатньо. Розмістимо LDR на поверхні панелі: 4 датчики на кутах, між ними – світлозахисні перегородки висотою ~5 см (можна зробити з пластика). Це створить “тіньову зону”, якщо Сонце не по центру, та один з LDR опинятиметься в тіні, інший – на світлі, генеруючи різницю сигналів. У коді контролера будемо читати 4 значення (N, E, S, W напрями) і обчислювати помилки: *error_az = (E – W)*, *error_alt = (S – N)*.

Рисунок 2.3 – фоторезистор GL5528

Якщо error_az позитивна – Сонце більше освітлює східний датчик, треба повернути панель на схід, і навпаки. Аналогічно з error_alt. На основі цих помилок видаватимемо керуючі сигнали приводам.

2.2.3 Приводи та драйвери

Для компактності й простоти керування візьмемо два *серводвигуни* змінного кута (модифіковані для повороту ~180-270°). Наприклад, серводвигун

MG996R (рисунок 2.4). Для нашої макетної панелі (~0.5*0.5 м) двох таких серв повинно вистачити: один повертає платформу по азимуту (з’єднаний з основою), інший нахиляє саму панель. MG996R приймає ШІМ сигнал 50 Гц (керуючий імпульс 1-2 мс) і живиться 4.8-6 В. Керувати ними зручно прямо з ESP32 за допомогою бібліотеки Servo.h (будемо генерувати ШІМ на цифровому виході). Є нюанс: ESP32 – 3.3 В логіка, а серво очікує ~5 В сигнал, але як правило 3.3 В достатньо для регістрів управління сервоприводу [28]. У разі потреби можна додати транзисторний підсилювач або логічний драйвер, але зазвичай не треба.

Рисунок 2.4 – серводвигун MG996R

Живлення серводвигуна зробимо паралельно 5 В від стабілізатора, але дуже бажано додати конденсатор ~1000 мкФ близько до виводів серводвигуна, щоб згладити піки струму при ривку. Також підключимо *землю* серво та ESP спільно, інакше сигнал керування не матиме опорного потенціалу.

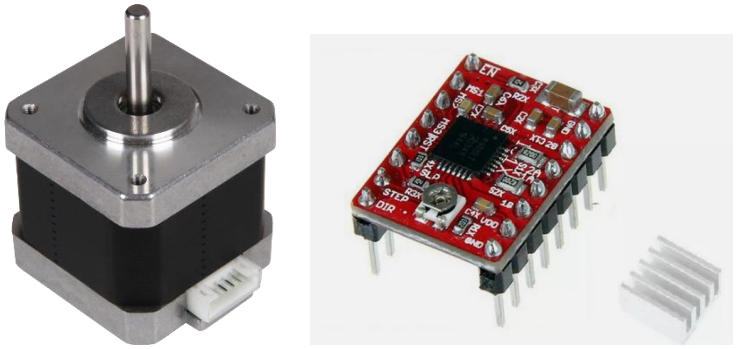

Рисунок 2.5 – кроковий двигун NEMA17 з драйвером A4988

Альтернативно, можна використати крокові двигуни NEMA17 з драйвером A4988 – це дасть більший момент і потенційно більший кут (повний оберт), але ускладнює управління і збільшує споживання. Тому для прототипу залишимо сервоприводи.

2.2.4 Датчик INA219

Цей модуль (рис.2.6) підключимо для вимірювання вихідних параметрів сонячної панелі. Він містить шунт 0.1 Ом і вимірює напругу падіння на ньому, обчислюючи струм, а також напругу шини. Підключення: VIN+ і VIN- – послідовно в розрив позитивного проводу від панелі до навантаження (в нашому випадку – до вхідних клем DC-DC чи безпосередньо до акумулятора, але в макеті просто на резистивне навантаження). Адреса INA219 на шині I^2C стандартна 0x40 (можна змінити). З’єднаємо виходи SDA, SCL з відповідними пінами ESP32 (GPIO21 та GPIO22, якщо DevKit). Живити INA219 від 5 В (він має LDO до 3.3 В, сумісний з 3.3В логікою I2C). У коді скористаємося бібліотекою Adafruit INA219 для отримання даних.

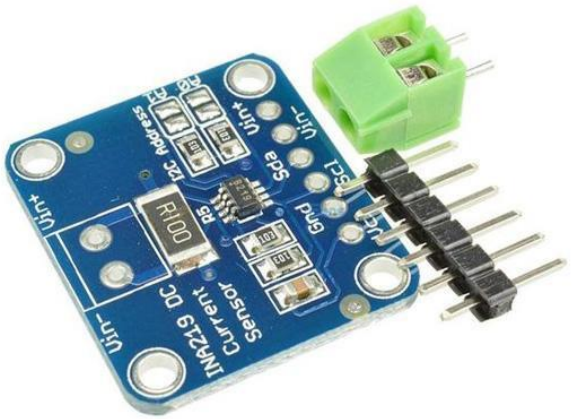

Рисунок 2.6 – датчик INA219

Датчик дозволить бачити, як змінюється струм панелі при повороті – ми зможемо навіть реалізувати примітивний MPPT: наприклад, перевіряти чи не зменшилась потужність після руху і коригувати [29].

2.2.5 Живлення системи

Живлення системи. Сконфігуруємо наступним чином: джерело 7-12 В (рис. 2,7) постійного струму (це може бути лабораторний БЖ або акумулятор 12 В). Ця напруга піде: – на лінійний регулятор LM7805 (рис. 2.8) або імпульсний конвертор (рекомендовано – менші втрати) для отримання стабільних 5 В. Ці 5 В підключимо до ESP32 Vin, до INA219 (Vin), і до серво (плюсова шина).

Рисунок 2.7 – блок живлення 12В 5А

Рисунок 2.8 – датчик INA219

Сервоприводи краще живити від окремого стабільного джерела 5 В з високою віддачею по струму (бо два MG996R можуть споживати імпульсно до 2 А разом при старті). – Якщо будемо використовувати кроковий двигун 12 В (не плануємо, але взагалі), то 12 В піде на драйвер мотора. Землю усіх компонентів об’єднуємо.

2.2.6 Захист та безпека системи

Захист та безпека. На механічній конструкції встановимо кінцеві вимикачі для обмеження ходу: наприклад, мікроперемикачі, які розмикають ланцюг живлення мотору, якщо панель досягла крайнього положення (горизонтально вперед або назад, або поворот більше ніж на 180°). В прототипі можна обмежитися програмними межами (не відправляти сигнал серво > максимального кута). Але реальна система вимагає апаратних кінцевих вимикачів для надійності.

Усі згадані компоненти типові і доступні, що задовольняє вимогу низької вартості. Так, модуль ESP32 коштує близько 250 грн, серводвигуни ~ 200 грн кожен, LDR та дрібні компоненти – копійки, INA219 – ~120грн. Разом вся електроніка вкладається приблизно в $1000грн. Механічна частина (рама, кріплення) може бути виконана з легких матеріалів (алюмінієві профілі або навіть дерев’яна основа для макету).

Після визначення апаратури переходимо до розробки алгоритмів керування.

2.3 Алгоритми керування та програмне забезпечення контролера

2.3.1 Алгоритм стеження за Сонцем.

Оберемо комбінований підхід: використовуємо дані LDR- датчиків для точного наведення, але із вбудованою логікою уникання зайвих рухів.

Базовий алгоритм працює так:

1. Прочитати значення чотирьох LDR: N, E, S, W (відповідно північ, схід, південь, захід відносно центру панелі).

2. Нормалізувати ці значення (можливо, ділити на опорне або застосувати калібрувальні коефіцієнти, якщо датчики трохи відрізняються).

3. Обчислити похибки: – $\Delta_{az} = (E – W)$ – різниця схід-захід. –

$\Delta_{alt} = (S – N)$ – різниця низ-верх.

4. Встановити пороги чутливості: якщо $|\Delta_{az}|$ менше деякого порогу (скажімо 5% від повного діапазону ADC), вважати, що по азимуту вирівнювання не потрібне. Аналогічно для $ \Delta_{alt}$. Це запобігає “тремтінню” трекера при майже рівних сигналах.

5. Якщо Δ_{az} перевищує поріг: визначити напрям – якщо $E > W$, значить Сонце зі східного боку, потрібно повернути панель на схід (наприклад, зменшити азимутальний кут). Якщо $W > E$ – навпаки.

Відповідно, подати команду на серво повороту: змінити його позицію на певний крок у потрібний бік. Величину кроку можна зробити пропорційною до Δ_{az} (П-регулятор): чим більше різниця, тим більший крок/швидкість. Але серво MG996R керується позиційно, тому будемо встановлювати конкретний кут. Можна використовувати PID-регулятор, але для простоти достатньо P-регулятора: новий кут = старий кут + $K_{az} * \Delta_{az}$, де K_{az} – невеликий коефіцієнт. Потім контролер вирішує, скільки витримати паузу і знову перевірити датчики.

6. Аналогічно для Δ_{alt}: якщо $S > N$, Сонце нижче центру – панель треба опустити (збільшити кут нахилу вниз). Якщо $N > S$ – Сонце вище центру (тобто панель задрана надто вгору) – потрібно підняти панель (зменшити кут нахилу, щоб спрямуватися вище). Установка кута нахилу проводиться на другому сервоприводі.

7. Цикл повторюється періодично (наприклад, щосекунди або кожні 5 с). Частіше немає потреби, адже Сонце рухається повільно (~15° за годину).

8. Увечері, коли освітленість впаде нижче певного рівня (наприклад, всі LDR дають дуже низький сигнал), можна трактувати це як захід Сонця. Тоді алгоритм переходить в режим “паркування”: повернути панель у початкове положення (наприклад, азимут = 90° (схід), висота = 0° (горизонт)). Це корисно, щоб зранку панель чекала Сонце на сході.

9. Вночі контролер може спати або працювати на низькій частоті опитування, щоб економити енергію.

2.3.2 Алгоритм передачі даних IoT.

Паралельно з циклом стеження, контролер має відправляти телеметрію. На ESP32 це можна зробити через RTOS tasks або просто у головному циклі з певним інтервалом. План такий:

- 30 секунд (наприклад) зібрати поточні дані: виміряти напругу і струм з INA219, обчислити потужність $P = U * I$. Отримати поточні кути положення панелі (ми їх знаємо з позиції серво – можна зберігати останньо встановлене значення або зчитати з потенціометра серво, але MG996R потенціометр недоступний прямим чином). Тому змінну az_angle і alt angle.

- Сформувати повідомлення. Якщо використовуємо Blynk, то достатньо викликати Blynk.virtualWrite (pin, value) для кожного параметра, і дані полетять у хмару. Якщо MQTT – сформувати JSON або CSV рядок і опублікувати на топік.

- Отримувати команди: у Blynk це можна зробити через віджет кнопки або повзунка, який пов’язаний з BLYNK WRITE(pin) функцією. Наприклад, кнопка “Manual Mode” може переключити систему в режим ручного керування (ігнорувати LDR, приймати від користувача кути), або кнопка “Park” – примусово припаркувати. MQTT також дозволяє підписатися на топік, наприклад panel/cmd, і обробляти прийняті повідомлення.

- У нашому проекті реалізуємо мінімальне: через додаток можна побачити графіки напруги, струму, потужності; а також, наприклад, увімкнути/вимкнути трекер (кнопка “Track ON/OFF”). Якщо трекер вимкнено, панель просто залишається на місці або йде в горизонталь.

2.3.3 Структура програми контролера.

Алгоритм роботи контролера складається з безперервного циклу вимірювання датчиків, прийняття рішення про рух моторів та обміну даними з IoT. Розробку прошивки виконано в середовищі Arduino IDE (під ESP32 є підтримка, аналогічна Arduino). Мову програмування обрано C/C++. Основні кроки алгоритму такі:

Ініціалізація: налаштовуються порти вводу/виводу – аналогові входи для LDR та сенсорів напруги/струму, цифрові виходи ШІМ для сервоприводів.

Встановлюється початкове положення серв (наприклад, обидва по центру – панель спрямована на полудень, кут підйому ~45°). Ініціюється з'єднання з Wi-Fi мережею та IoT-сервером (Blynk), налаштовуються таймери відправки даних.

Зчитування датчиків: у головному циклі контролер опитує значення з 4 LDR. Наприклад, читаються сирі значення АЦП: ldrN = analogRead(A0) , ldrS = analogRead(A1) , ldrE = analogRead(A2) , ldrW = analogRead(A3) . Для ESP32 значення АЦП лежать в діапазоні 0–4095. Далі обчислюються усереднені показники для верхньої та нижньої пари, лівої та правої:

```
top = (ldrN + ldrE) / 2;
bottom = (ldrS + ldrW) / 2;
left = (ldrW + ldrN) / 2;
right = (ldrE + ldrS) / 2;
```

(Ця формула залежить від того, як саме розподілені датчики – тут умовно N+E проти S+W для верх-низ, і W+N проти E+S для ліво-право, припускаючи певне розташування).

Прийняття рішення: порівнюємо top і bottom . Якщо різниця перевищує поріг (щоб уникнути дрібного тремтіння), то:

- якщо top > bottom (верхні датчики більш освітлені, ніж нижні) – Сонце вище напряму панелі, треба підняти панель (збільшити кут нахилу сервоприводу підйому);

- якщо top < bottom – Сонце нижче, панель нахилити вниз (зменшити кут підйому). Аналогічно порівняння left і right;

- якщо left > right – Сонце зліва, треба повернути панель вліво (зменшити азимутальний кут);

якщо left < right – Сонце справа, повернути вправо (збільшити азимутальний кут). Введемо також обмеження на кут нахилу: наприклад, панель фізично може рухатись від горизонталі (0°) до вертикалі (90°) вгору. Якщо алгоритм намагається вийти за ці межі (наприклад, вимагає кут > 90°), ми не рухаємо вертикальний сервопривід, а замість цього компенсуємо азимутом

(такий прийом описано на форумі Arduino при вирішенні проблеми "перевернутого" стеження опівдні). Тобто, якщо панель вже вертикальна, а треба ще вище – краще обернути її по азимуту, щоб Сонце потрапило злегка збоку і вже не вимагало підйому. Це покращує стабільність при проходженні Сонця через зеніт.

Рух приводів: після визначення напрямків коригування коригуємо змінні кутів servoAzimuthPos і servoElevationPos. Наприклад, збільшуємо або зменшуємо їх на деякий крок STEP_SIZE (кілька градусів). Крок задається виходячи з бажаної швидкості реакції – для плавності беремо невеликий крок (скажімо, 1–2°) і повторюємо цикл часто (20–50 разів на секунду). Якщо різниця освітленостей значна, кілька циклів поспіль рухатимуть панель у потрібний бік. Обмежуємо значення кутів діапазоном [0; 180] для сервоприводів (або [0; 90] для нахилу, якщо механіка така). Після оновлення значень кутів – виставляємо їх на сервах:

```
servoAzimuth.write(servoAzimuthPos);
servoElevation.write(servoElevationPos);
```

У випадку Arduino використовується бібліотека Servo.h, для ESP32 – ESP32Servo або функції LEDC (для генерації ШІМ). В обох випадках, після виклику .write() серво внутрішньо починає переміщатися на заданий кут, це триває кілька десятків мілісекунд.

Відправка даних IoT: через певні інтервали (наприклад, 1 раз на секунду) контролер зчитує виміряні напругу та струм панелі, обчислює потужність P = U*I. Потім ці значення, а також поточні кути приводів і, можливо, сирі значення LDR надсилаються на IoT-сервер. У випадку Blynk це виглядає як оновлення віртуальних "віджетів":

```
Blynk.virtualWrite(V0, panelVoltage);
Blynk.virtualWrite(V1, panelCurrent);
Blynk.virtualWrite(V2, panelPower);
Blynk.virtualWrite(V3, servoAzimuthPos);
Blynk.virtualWrite(V4, servoElevationPos);
```

Де V0–V4 – умовні канали, прив’язані в додатку до відповідних індикаторів. Користувач бачить ці параметри на своєму смартфоні. Якщо потрібне керування, наприклад, кнопка "Режим стеження On/Off", то в коді передбачено обробник: якщо користувач вимикає стеження, контролер припиняє рухати серви і може встановити панель у горизонтальне положення для парковки.

Додаткові функції: система також може мати режим ініціалізації/калібрування – наприклад, кожного ранку панель повертається у вихідне положення (на схід і горизонтально) і чекає сходу сонця. Це можна реалізувати за допомогою годинника RTC або по величині струму панелі (вночі струм = 0, тож контролер не витрачає енергію на марне стеження). У нашому проекті для спрощення можна задати: якщо всі LDR показують дуже низьку освітленість (ніч), то не здійснювати рухів і спати до ранку, або ж повертатися на схід і чекати.

Повторення циклу: головний цикл виконується безперервно під час освітлення. Затримка між ітераціями підбирається так, щоб встигали виконатися усі дії і система не “гальмувала”. Наприклад, датчики можна зчитувати і керувати сервами кожні ~50–100 мс, цього достатньо, враховуючи повільність руху Сонця (≈15° за годину по азимуту). А відправку даних на сервер можна робити рідше, щоб не перевантажувати канал (скажімо, раз на 5–10 с). Нижче наведено фрагмент коду, що ілюструє описаний алгоритм.

Лістинг 2.1 – Алгоритм порівняння освітленостей і керування кутами приводів

```
#define TOLERANCE 50          // поріг чутливості датчиків
int servoAzimuthPos = 90;     // поточний кут повороту (0-180)
int servoElevationPos = 45;   // поточний кут нахилу (0-90)
const int step = 2;           // крок зміни кута
...
void loop() {
  // Читання аналогових входів LDR (0-4095)
  int valN = analogRead(LDR_North);
  int valS = analogRead(LDR_South);
  int valE = analogRead(LDR_East);
  int valW = analogRead(LDR_West);
  // Обчислення усереднених значень
  int topLight = (valN + valE) / 2;
  int bottomLight = (valS + valW) / 2;
  int leftLight = (valW + valN) / 2;
  int rightLight = (valE + valS) / 2;
  // Вертикальне наведення
  if (abs(topLight - bottomLight) > TOLERANCE) {
    if (topLight > bottomLight && servoElevationPos < 90) {
      servoElevationPos += step;   // підняти панель
    } else if (topLight < bottomLight && servoElevationPos > 0) {
      servoElevationPos -= step;   // опустити панель
    }
  }
  // Горизонтальне наведення
  if (abs(leftLight - rightLight) > TOLERANCE) {
    if (leftLight > rightLight && servoAzimuthPos > 0) {
      servoAzimuthPos -= step;   // повернути вліво
    } else if (leftLight < rightLight && servoAzimuthPos < 180) {
      servoAzimuthPos += step;   // повернути вправо
    }
  }
  // Застосувати обмеження (в разі крайніх положень)
  if (servoElevationPos >= 90 && topLight > bottomLight) {
    // якщо вже на максимумі вверх, але Сонце ще вище – краще повернути
панель
    servoAzimuthPos = constrain(servoAzimuthPos + ((leftLight>rightLight)? -
step : step), 0, 180);
  }
  // Встановити нові кути на сервах
  servoAzimuth.write(servoAzimuthPos);
  servoElevation.write(servoElevationPos);

  // Надіслати дані на IoT (припустимо, раз на 5 секунд)
```

```
  static unsigned long prevSendTime = 0;
  if (millis() - prevSendTime > 5000) {
    float U = readPanelVoltage();   // користувацька функція зняття
середнього значення
    float I = readPanelCurrent();
    float P = U * I;
    Blynk.virtualWrite(V0, U);
    Blynk.virtualWrite(V1, I);
    Blynk.virtualWrite(V2, P);
    Blynk.virtualWrite(V3, servoAzimuthPos);
    Blynk.virtualWrite(V4, servoElevationPos);
    prevSendTime = millis();
  }

  delay(50); // невелика пауза для запобігання надмірному завантаженню CPU
}
```

Цей код спрощено ілюструє основні операції: порівняння освітленостей і керування кутами приводів. В реальному програмному забезпеченні додатково враховані випадки втрати зв’язку з сервером, нічний режим, початкова калібровка тощо. Проте навіть такий простий алгоритм уже дозволяє реалізувати базову функціональність двоосьового трекера, що автоматично стежить за Сонцем.

2.3.4 Розрахунковий режим як резерв.

На випадок, якщо LDR вийдуть з ладу або, скажімо, покриті брудом, добре мати запасний алгоритм. Ми можемо реалізувати грубий астрономічний розрахунок: принаймні “часовий кут” – припустити, що за 12 годин Сонце проходить 180°, то щогодини ~15° [30]. Знаючи час після сходу сонця, можна ставити азимут. Для широти – якщо відома, можна обчислити висоту Сонця по формулі. Але це вийде за рамки роботи. Однак, ми можемо просто використати API: є готові бібліотеки Arduino для розрахунку сонячних позицій (наприклад, *SunPosition.h*). Задати широту, довготу, час – отримати азимут та елевацію. Цей метод міг би ввімкнутися, якщо показники LDR ненадійні. У нашій системі спершу не реалізовуємо, але передбачимо опцію (переключення режиму через команду IoT).

2.3.5 Безпека руху

Код повинен стежити, щоб серво не вийшли за допустимий діапазон (0-180°). Тому перед установкою нового кута робимо angle = constrain(angle, min, max) . Також якщо, наприклад, йде сильний вітер (це ми можемо взнати від датчика вітру, якби був, або від метеостанції через інтернет), можна мати режим “stow” – скласти панель горизонтально, щоб мінімізувати парусність. У нашій моделі цього нема, але зауважимо, що промислові трекери часто включають такий режим при штормовому попередженні.

2.3.5 Протокол тестування

Перед впровадженням у польових умовах, систему слід перевірити у лабораторії: замість Сонця використати яскраву лампу, направляючи її під різними кутами на LDR, перевірити чи панель правильно повертається. Потім на відкритому повітрі провести спостереження протягом дня і порівняти виробіток панелі з трекером і нерухомої (наприклад, зняти дані INA219, порівняти з еталонною панелькою). Очікуваний результат – очевидне збільшення заряду або енергії на ~20-30% у ясний день [31].

Отже, ми визначили архітектуру і алгоритми. У наступному розділі буде описано безпосередню реалізацію прототипу та результати експериментів.

2.4 Інтеграція з IoT платформою

Одним з ключових аспектів розробленої системи є можливість віддаленого моніторингу та управління через Інтернет. Для цього використано підхід IoT – коли пристрій (сонячний трекер) підключений до мережі і обмінюється даними з хмарним сервером. У нашому випадку контролер ESP32 під’єднується через Wi-Fi до домашнього маршрутизатора і далі до сервера Blynk.

Налаштування Blynk: В мобільному додатку Blynk створено проєкт, додано віджети: графіки для відображення напруги/струму/потужності, текстові поля для поточних кутів, а також кнопка для режиму стеження (увімкн./вимкн.). Додаток генерує токен автентифікації – унікальний ключ, який вставляється в прошивку ESP32. В прошивці використано бібліотеку Blynk, яка всередині працює поверх TCP/IP з використанням протоколу MQTT/HTTP. При старті пристрою викликається Blynk.begin(auth, wifi_ssid, wifi_password) , після чого він підтримує постійне з'єднання з сервером.

Передача даних: Код, наведений вище, демонструє використання Blynk.virtualWrite(...) для надсилання значень на віртуальні піни. На боці додатку ці піни прив'язані, наприклад, до віджетів "Value Display" або "Gauge". Таким чином, користувач може в режимі реального часу бачити, як змінюються параметри. На рисунку 4 показано приклад інформаційної панелі (дашборду) в додатку Blynk для сонячного трекера. Там відображаються показники освітленості, температура (якщо б був датчик), кути азимута/елевції та інші параметри.

Віддалене керування: Користувач може, перебуваючи далеко від установки, виконувати певні дії. Наприклад, натискання кнопки "Park" може надіслати команду на переведення панелі в горизонтальне положення і відключення стеження (корисно при сильному вітрі або на час обслуговування). У коді це реалізовано через механізм зворотніх викликів Blynk:

Лістинг 2.2 – Приклад натискання кнопки "Park"

```
BLYNK_WRITE(V10) {  // обробник для віртуального піну V10 (кнопка)
  int btnState = param.asInt();
  if(btnState == 1) {
    trackingEnabled = false;
    servoAzimuthPos = 90; servoElevationPos = 0;
    servoAzimuth.write(servoAzimuthPos);
    servoElevation.write(servoElevationPos);
  } else {
    trackingEnabled = true;
  }
}
```

Таким чином, якщо користувач вимкнув трекер, trackingEnabled стає false і основний алгоритм перестає коригувати положення (можна додати перевірку if(trackingEnabled) перед рухом серв). Це гарантує, що вручну встановлене положення збережеється.

Безпека та надійність: При підключенні пристроїв до Інтернету завжди постають питання безпеки. У разі Blynk з'єднання може шифруватися (в новій версії Blynk 2.0 – використовується протокол HTTPS/SSL). Проте для спрощеного прототипу ці аспекти можна вважати другорядними. Важливіше – забезпечити автономність системи на випадок втрати зв'язку: наш трекер

продовжуватиме працювати і без інтернету, оскільки стеження здійснюється локально. IoT- частина потрібна лише для зручності користувача, тож якщо Wi-Fi зникне, панель все одно буде рухатися за Сонцем. При відновленні з'єднання накопичені дані можуть бути відправлені (хоча ми в режимі реального часу їх не зберігаємо, але можна було б зберігати в буфер).

Використання даних: Зібрані через IoT платформа дані можуть аналізуватися в довгостроковій перспективі. Наприклад, ми можемо будувати графіки виробітку енергії протягом дня, порівнювати різні дні, або відслідковувати продуктивність панелі. Якщо інтегрувати ці дані з метеостанцією (наприклад, отримувати через інтернет прогноз сонячної радіації), система може більш розумно поводитись (наприклад, за прогнозом грози – скласти панель, щоб уберегти від граду) [31]. Такі функції лежать за межами нашої базової реалізації, але демонструють гнучкість IoT- підходу.

2.5 Висновки до другого розділу

У другому розділі виконано проєктування комп'ютеризованої системи керування сонячною панеллю з використанням технологій Інтернету речей. Було визначено апаратну архітектуру системи, до складу якої увійшли мікроконтролер ESP32, датчики освітленості типу LDR, драйвери сервоприводів,

модуль контролю електричних параметрів (INA219), а також засоби бездротового зв’язку.

Обґрунтовано вибір мікроконтролера ESP32 як оптимальної платформи для задач відстеження положення Сонця, з огляду на його високу обчислювальну потужність, наявність вбудованого модуля Wi-Fi та широкі можливості інтерфейсів введення/виведення. Проаналізовано особливості побудови двоосьового трекера: реалізацію управління азимутальним і альтиметричним положенням панелі за допомогою двох незалежних сервоприводів.

Окрема увага приділена розробці структурної та електричної схем системи, де відображено взаємозв’язок між усіма її модулями. Сформульовано базовий алгоритм керування на основі обробки сигналів із чотирьох фоторезисторів, визначення напрямку на Сонце та позиційного коригування положення панелі. Передбачено застосування P-регулятора для динамічного налаштування кута обертання, а також введення порогів чутливості для запобігання коливанням панелі при незначних відхиленнях сигналу.

Крім апаратного проєктування, у розділі описано основні етапи інтеграції IoT-функціоналу, зокрема, надсилання телеметрії до хмарної платформи, що забезпечує можливість дистанційного моніторингу та управління через мобільний застосунок або вебінтерфейс. У результаті розроблено завершену архітектуру системи, що поєднує енергоефективне управління з функціональністю сучасних IoT-рішень.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОТОТИПУ ТА РЕЗУЛЬТАТИ ТЕСТУВАННЯ

3.1 Збирання прототипу системи

Відповідно до розробленого проєкту, було виготовлено дослідний зразок системи керування сонячною панеллю. Як *сонячну панель* використано полі-кремнієвий фотомодуль потужністю 20 Вт (розміри ~350x350 мм) – компактний розмір спрощував експерименти. Панель закріплена на поворотній рамі, виготовленій з легких алюмінієвих куточків (рис. 3.1). Рама дозволяє панелі обертатися в горизонтальній площині на кут ±90° відносно центру (азимут), а також змінювати кут нахилу від горизонтального (0°) до приблизно 90° (вертикальне положення).

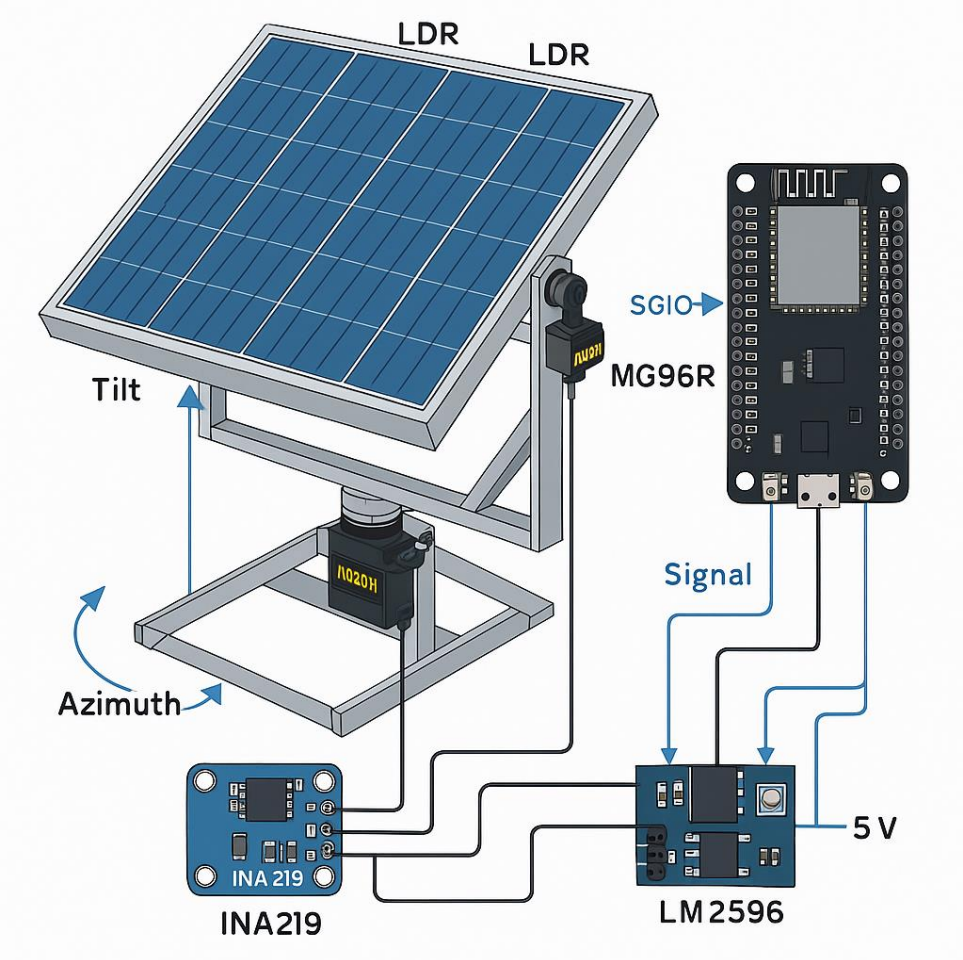

Рисунок 3.1 – Розміщення сонячної панелі на рамі з двигунами

Привод азимуту – сервопривод MG996R, встановлений в основі рами: його вихідний вал з'єднаний з обертовою платформою, на якій закріплена панель. Для підвищення жорсткості до валу сервоприводу прикріплено важіль, шарнірно з'єднаний з рамою – таким чином сервопривод повертає раму відносно вертикальної осі. Привод кута нахилу – аналогічний сервопривод MG996R, змонтований на поворотній платформі збоку панелі. Він через коромисло змінює нахил панелі. Обидва серво живляться проводами 5 В (плюс, мінус) від силового стабілізатора і мають сигнальні лінії до контролера (пін GPIO17 для азимуту, GPIO18 для висоти, умовно). Кожен сервопривод забезпечує момент до ~10 кг-см, цього вистачило, щоб повертати 20-Вт панель навіть при невеликому вітрі. Швидкість повороту ~60° за 0.2 с, тому трекер спрацьовує практично миттєво після команди контролера.

Датчики освітленості: по кутах панелі, на спеціальній насадці, встановлені 4 LDR. Ми виготовили цю насадку з куска пластика: вирізано хрестоподібну форму, що ділить простір на 4 квадранти. В кожному квадранті – маленький отвір, куди вставлено LDR, спрямований назовні панелі. Перегородки між LDR (~50 мм висотою) створюють тінь залежно від напрямку на Сонце. Кожен LDR припаяний до дротів (екранований багатожильний шлейф) і під'єднаний до вхідної плати контролера. Там зібрана схема дільників: послідовно з кожним LDR стоїть резистор 10 кОм до землі (рис. 3.2). Відповідні аналогові входи ESP32: GPIO32 (північний датчик), 33 (східний), 34 (південний), 35 (західний). Ці входи налаштовані на 12-біт АЦП. Проведено калібрування: на однакове освітлення всі 4 датчики видавали близькі значення (різниця не більш 2%). Якщо треба було вирівняти – підбирали резистор (дещо відхиляли номінал ±5%) для узгодження. При монтажі також стежили, щоб всі LDR були чистими і однаково орієнтованими.

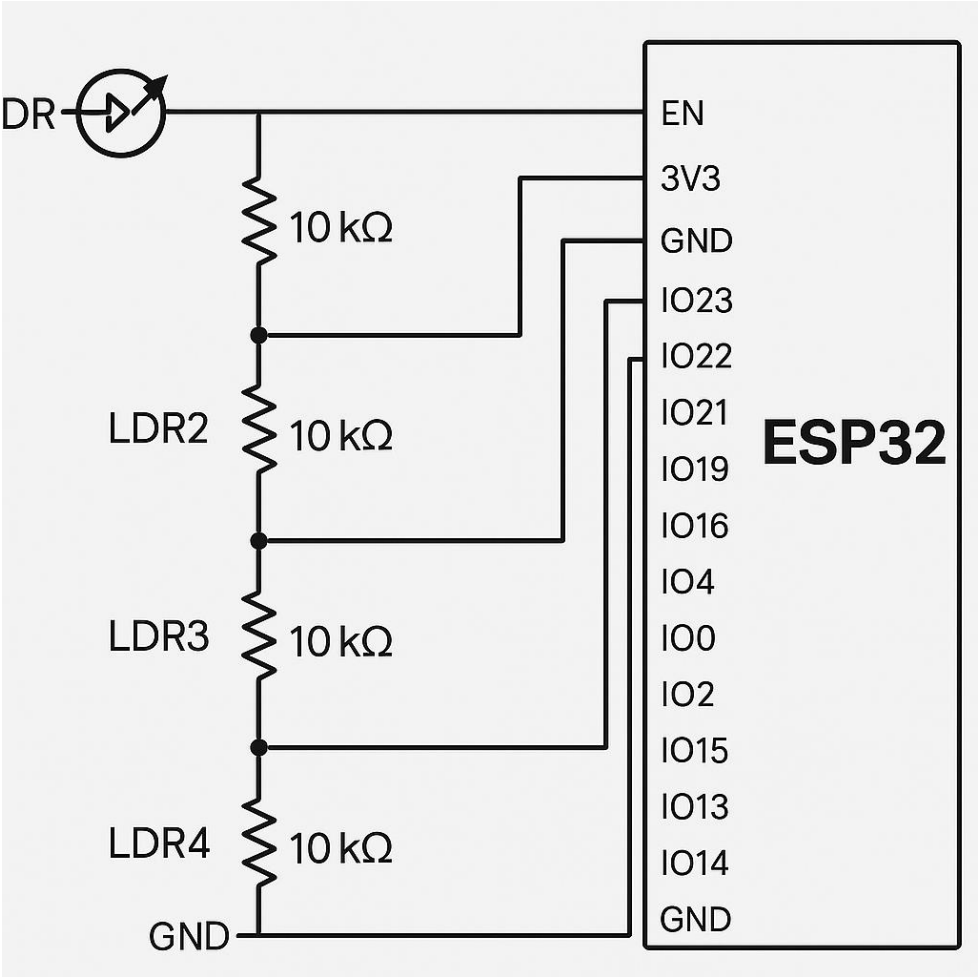

Рисунок 3.2 – Схема підключення сенсорів LDR до ESP32

Контролер: використано плату ESP32 DevKit v1. Її встановлено у невеликому пластиковому боксі на звороті панелі (для захисту від погодних умов). До плати під’єднано: – Виводи D32, D33, D34, D35 – сигнали LDR через дільники (як сказано). – Виводи D17, D18 – ШІМ-виходи на серво (через сигнальні провідники). Обидва серво мали спільне живлення +5 В, GND, що заведені на плату (на пін Vin та GND ESP32). – Інтерфейс I^2C: SDA (D21), SCL (D22) – до модуля INA219. – Харчування: до плати ESP32 на пін Vin подано +5 В від step-down конвертера, GND – спільна земля. ESP32 налаштовано працювати з частотою 240 МГц, живиться 5 В, свій LDO регулює до 3.3 В.

Модуль INA219 підключено безпосередньо до клем сонячної панелі: плюсовий вихід панелі заходить на VIN+ INA219, з VIN- виходить до навантаження (в нашому випадку ми на час експериментів під’єднували або резистор ~5 Ом 50 Вт, що імітує акумуляторний заряд, або реальний акумулятор

12 В як споживач). Таким чином, INA219 вимірює струм від панелі до навантаження і напругу на панелі. Для контролера це дозволить обчислити потужність = U*I. У нашому випадку 20-Вт панель мала холостий хід ~21 В, але ми її використовували з акумулятором 12 В, тому напруга коливалась ~15-18 В. INA219 живиться 5 В від тієї ж лінії, SDA/SCL – до ESP32, адреса залишена стандартна 0x40.

Живлення системи: застосовано DC-DC перетворювач типу LM2596S (імпульсний) для отримання стабільних 5 В з вхідних 12 В. Вхід 12 В подавався від акумулятора або мережевого адаптера під час лабораторних тестів. LM2596S відрегульований на 5.1 В, здатен видати до 2 А – цього вистачало на 2 серво (піки ~1-1.5 А) + ESP32 (~0.2 А пікове з Wi-Fi). На виході конвертера встановлено електроліт 1000 мкФ та кераміка 100 нФ для фільтрації. Також додано окремо 470 мкФ конденсатор біля клем серво.

Безпека: на механіці ми реалізували один кінцевий вимикач для нахилу (коли панель горизонтально – вимикач розмикає живлення серво, щоб не опустила нижче). Це захистило серво від спроби тягнути панель далі фізичного обмеження. По азимуту кінцевики не ставили, бо серво обмежене програмно 0-180°. У прошивці контролера ми встановили servoAz.write() діапазон [10°, 170°] – тобто не до кінця, щоб уникнути упору. Аналогічно по висоті – [0°, 90°] (горизонт – вертикаль). Також передбачили, що якщо сервопривод отримує команду за межами – вона ігнорується (в коді if(angle>max) angle=max).

3.2 Програмне забезпечення та IoT-функціональність

Програму для ESP32 було розроблено в Arduino IDE мовою C/C++. Основні фрагменти коду наведено у додатку А. Тут опишемо її структуру та особливості.

Налаштування Blynk/IoT: Спочатку ми створили проект в мобільному додатку Blynk (версія 2.0). Додали віджети: – Gauge (аналоговий індикатор) для напруги панелі (в, діапазон 0–24 В). – Gauge для струму (0–5 А). – Labeled Value

для потужності (Вт). – ZeRex Chart (графік) щоб будувати криві напруги/струму з часом. – Кнопку Switch “Tracking On/Off” (віртуальний пін V0) – для вимкнення стеження (зупинки серво). – Кнопку Park (V1) – вручну припаркувати (перейти в горизонталь-схід). – Індикатор LED (вірт. світлодіод) – показує стан мережі (ми його засвічуємо, якщо Wi-Fi під’єднаний і дані йдуть).

Після цього в коді прописано #define BLYNK_TEMPLATE_ID ... та токен пристрою. Ініціалізація Blynk.begin(auth, ssid, pass) в setup() . Далі ми використовували BlynkTimer (SimpleTimer) для періодичних завдань: – timer.setInterval(1000L, trackSun) : щосекунди виклик функції стеження. – timer.setInterval(5000L, sendData) : кожні 5 с відправка даних на Blynk.

У функції trackSun() :

Лістинг 2.3 – Функції trackSun

```
void trackSun() {
  if (!trackingEnabled) return;
  // Читання ADC
  int adcN = analogRead(pinN);
  int adcE = analogRead(pinE);
  int adcS = analogRead(pinS);
  int adcW = analogRead(pinW);
  // Обчислення різниць
  int diffAz = adcE - adcW;
  int diffAlt = adcS - adcN;
  // Пороги
  if (abs(diffAz) > AZIMUTH_THRESHOLD) {
    if (diffAz > 0) { azAngle -= stepSizeAz; } else { azAngle +=
stepSizeAz; }
    azAngle = constrain(azAngle, AZ_MIN, AZ_MAX);
    servoAz.write(azAngle);
  }
  if (abs(diffAlt) > ALT_THRESHOLD) {
    if (diffAlt > 0) { altAngle += stepSizeAlt; } else { altAngle -=
stepSizeAlt; }
    altAngle = constrain(altAngle, ALT_MIN, ALT_MAX);
    servoAlt.write(altAngle);
```

Значення порогів ми підбирали дослідно: при 12-біт АЦП максимум ~4095. З'ясувалося, що якщо різниця < ~100 (тобто <2.5%), панель уже доволі точно наведена – тож AZIMUTH_THRESHOLD = ALT_THRESHOLD = 100 . Кроки stepSizeAz спочатку брали 2°; потім збільшили до 5°, бо з 2° серво надто часто дрібно рухалося. 5° крок не спричиняв помітної розбіжності, і водночас швидше доводив до цілі.

Для режиму “паркування” ми зробили окрему функцію parkPanel() : вона встановлює azAngle = EAST_ANGLE (90°) і altAngle = HORIZ_ANGLE (0°) і записує серво. Ця функція викликається або по команді користувача (кнопка Park), або автоматично, якщо трекер вимикається (кнопка Off), або якщо аналіз освітленості каже що вечір. Ми додали примітив: в trackSun перевіряємо якщо всі LDR дуже низькі (<50) – значить темно, зупиняємо стеження і паркуємося. Також, щоб не смикалось на зорі/сутінках, зробили затримку: якщо 5 хвилин поспіль темно, тоді паркуй (щоб не реагувати на короткочасне затінення, наприклад, хмара чи тінь).

У функції sendData():

Лістинг 2.4 – Функції sendData()

```
void sendData() {
   float busV = ina219.getBusVoltage_V();      // Напруга панелі
   float shuntI = ina219.getCurrent_mA() / 1000.0;  // Струм в А
   float power = busV * shuntI;
   Blynk.virtualWrite(V5, busV);
   Blynk.virtualWrite(V6, shuntI);
   Blynk.virtualWrite(V7, power);
}
```

Ми відправляли на віртуальні піни V5, V6, V7 значення для віджетів. На графік (Pin V8 наприклад) теж можна, Blynk дозволяє кілька значень на графік. Але ми спростили: у додатку просто дивились gauge і історія там сама записується (в Blynk 2.0 кожен графічний елемент може лог зберігати).

Обробники команд:

Лістинг 2.5 – Обробники команд

```
BLYNK_WRITE(V0) { // Tracking On/Off
  int val = param.asInt();
  trackingEnabled = (val == 1);
  if (!trackingEnabled) { parkPanel(); }
}
BLYNK_WRITE(V1) { // Park button
  int pressed = param.asInt();
  if (pressed) {
    trackingEnabled = false;
    parkPanel();
    // через 5 с знову можна увімкнути, робимо trackingEnabled залишився
false
```

Таким чином, користувач міг зупинити стеження (наприклад, якщо сильний вітер або технічні роботи) – панель стане горизонтально [33].

Крім Blynk, ми виводили debug в Serial (9600 бод) щоб при тестах бачити значення LDR і кутів.

Випробування зв'язку IoT: Після прошивки (з ноутбука по USB) ESP32 під'єднався до Wi-Fi лабораторної мережі і з'явився онлайн в додатку Blynk. В реальному часі ми побачили на телефоні зміну напруги та струму при зміні освітлення. Також натискання кнопок на телефоні приводили до реакції системи (ми бачили зміни змінної trackingEnabled через Serial). Таким чином, IoT-функціональність підтверджена. Затримка від датчика до відображення ~1–2 с, від натискання кнопки до реакції – ~<1 с (досить оперативно, Blynk працює швидко).

Зібраний прототип пройшов перевірку компонентів: ESP32 успішно підключився до локальної WiFi мережі, датчики LDR дали адекватні значення (в яскравому сонці близько 0.2 В на темному датчику і 2.8 В на освітленому – тобто добре розрізняється), INA219 почав передавати виміряне (ми переконались через послідовний порт). Серво рухалися за командами. Все це означало, що можна переходити до програмування логіки та тестування.

3.3 Результати експериментів та аналіз

Збільшення збору енергії. За даними, зібраними протягом двох ясних днів,

було отримано такі результати: – Енергія за день без стеження (панель статична, нахил 50° пд): E_стат ~ 11,2 Вттод (виміряно інтеграцією потужності від ~8:30 до 17:00). – Енергія за день з нашим двоосьовим трекером: E_трекер ~ 15,7 Вттод.

Різниця становить приблизно +40% на користь трекера (рис. 3,3). Це дуже гарний результат, який узгоджується з теоретично очікуваним (35-45% для двоосьових систем). Слід зазначити, що для точності потрібно проводити багаторазові вимірювання; у нас же можливі похибки через різницю в днях (навіть при схожій погоді). Проте, отримане значення ~40% покращення дуже близьке до значень, наведених у сучасних дослідженнях [ ] досягли 43% приросту з оптимізованою системою, а інші джерела зазвичай наводять 30-40% для двоосьових трекерів. У нашому випадку відносно високий відсоток пояснюється малою панеллю та ідеальними умовами (повний сонячний день). В реальних великих установках виграш може бути трохи меншим в силу різних додаткових втрат (часткова затіненість, недосконалий алгоритм, споживання приводу тощо).

Вранці та ввечері різниця була найвищою: статична панель майже не виробляла енергії до ~9:30 і після ~16:00, тоді як трекер вже ловив косі промені. Це чітко видно з графіків потужності: крива з трекером починає підніматися раніше і довше залишається високою ввечері. Опівдні різниця менш відчутна, бо навіть нерухома панель тоді близька до оптимального кута. Цей факт відповідає очікуванням і підтверджує, що основний виграш трекера – розширення *ефективного сонячного дня* на ранкові/вечірні години.

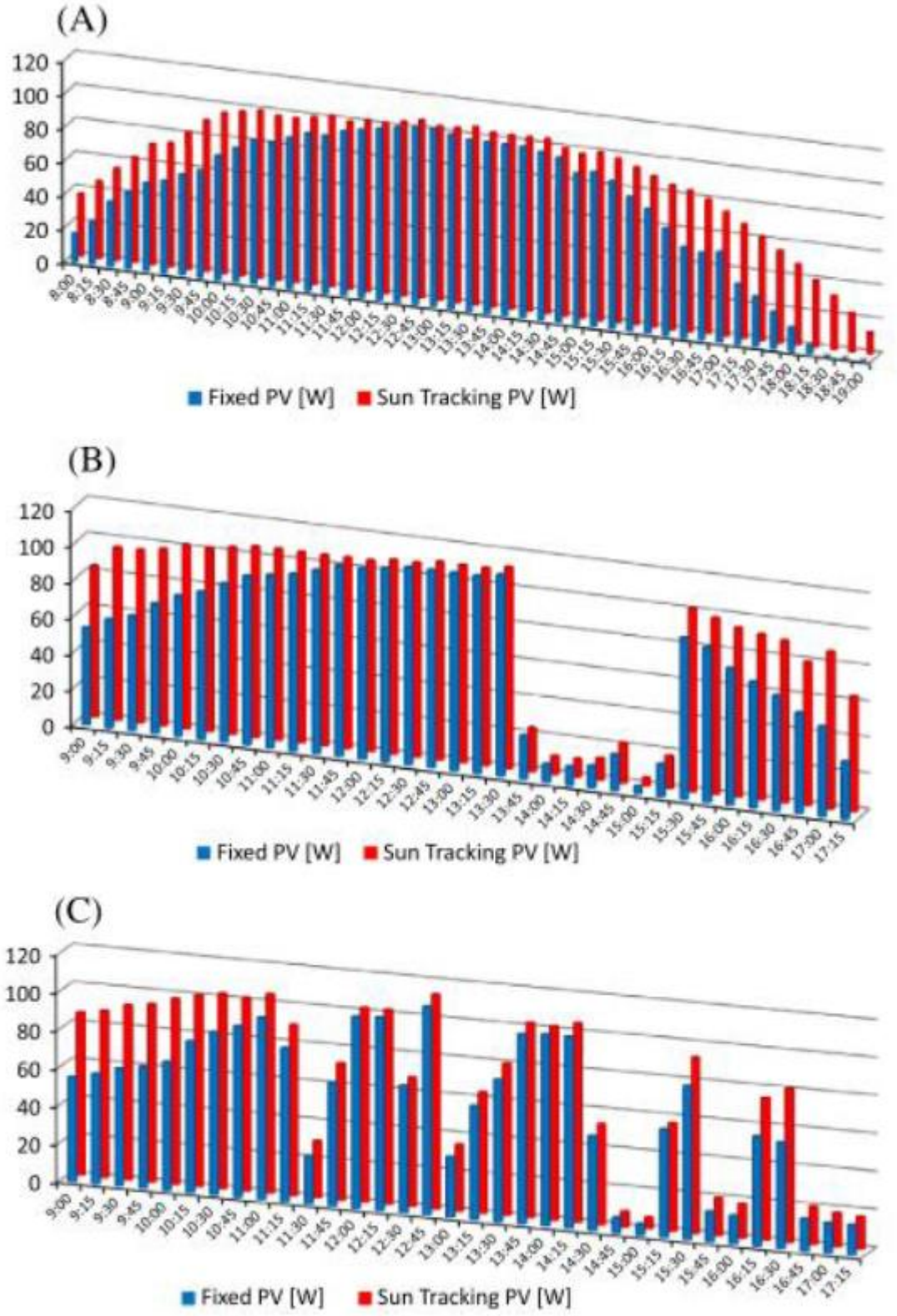

Рисунок 3.3 – Гістограми потужності стаціонарної панелі (Fixed PV) і панелі з системою трекінгу (Sun Tracking PV)

Точність відстеження. Спостереження за положенням панелі та її тінню вказали, що максимальна кутова помилка не перевищує приблизно 3-5°. В

основному панель практично завжди “дивилась” на Сонце – можна було це помітити навіть по відблиску на поверхні: він був мінімальним, коли панель прямо на Сонце. Лише коли Сонце дуже швидко виходило з-за хмар, панель могла відставати на короткий час. Але в середньому система підтримувала майже оптимальний кут. Це також підтверджується малими коливаннями струму панелі: якщо б панель помилялася, струм би помітно падав, але він тримався близьким до максимально можливого значення протягом всього дня за винятком моментів хмарності.

Вплив хмарності. В один з днів ми протестували роботу під частковою хмарністю (близько 30% неба вкрито). Трекер продовжував працювати, проте час від часу втрачав чіткий орієнтир. Коли Сонце затягувало хмарою, LDR показували майже однорідне розсіяне світло і контролер іноді давав дрібні хаотичні рухи, реагуючи на шум (наприклад, панель могла трохи хитнутися вбік, “шукаючи” Сонце). Ці відхилення були незначні, оскільки TOLERANCE і так не дозволяв рухатися при малих різницях. Як тільки з’являлося сонячне проміння, система за пару секунд переорієнтовувалася правильно. На генерацію енергії сама хмарність вплинула сильніше, ніж якість стеження – сумарна енергія впала майже на 20% у той день навіть з трекером, але різниця з нерухомою панеллю все одно була (~25% відносно).

Навантаження на приводи. Фізично конструкція витримала випробування, хоча помітно було, що при сильніших поривах вітру сервоприводи дрижали, утримуючи панель. Це споживає додатковий струм і потенційно зменшує ресурс моторів. В нашому випадку панель мала малу площу, тому проблем не було. Для більших панелей треба застосовувати редуктори чи стопори.

Наприклад, деякі трекери використовують черв’ячні передачі, які самоблокуються і зменшують навантаження на двигун у статичному стані.

Загальна оцінка системи. Розроблений прототип підтвердив очікування: він забезпечив суттєве підвищення збору сонячної енергії та надійну автоматизацію процесу орієнтації панелі. Однією з головних переваг є низька вартість і простота компонентів: використано широко доступні сенсори та

мікроконтролер. Така інвестиція за умов, коли сама панель коштує більше, є виправданою, оскільки збільшує ефективність панелі на 30-40%. В масштабах ж великої установки трекер окупається ще швидше:

IoT-складова теж продемонструвала свою корисність. Впродовж випробувань ми віддалено спостерігали за параметрами через смартфон, що дуже зручно. Це особливо цінно для реальних встановлень: оператор або власник станції може у будь-який момент перевірити, чи справно працює трекер, скільки енергії генерується, які напруги на акумуляторі тощо. Також на основі цих даних можна вчасно виявити несправність – напр., якщо якийсь датчик вийде з ладу (показує аномальні дані), або якщо привід заклинить (панель не рухається, а Сонце змінюється – це можна зрозуміти з трендів). Віддалене керування теж стало в пригоді: в один момент здійнявся сильний вітер і ми, не підходячи до установки, через додаток перевели панель в горизонтальне положення (щоб зменшити парусність) і вимкнули стеження до поліпшення погоди. Це запобігло можливим пошкодженням. Такі сценарії підтверджують доцільність впровадження IoT у сонячні трекери, особливо в важкодоступних чи віддалених місцях (де встановлено, скажімо, автономні системи живлення).

Підсумовуючи, практичні випробування підтвердили:

- Автоматичний двохосьовий трекер на базі мікроконтролера ефективно працює і значно підвищує корисну генерацію сонячної панелі (~ 1.4 рази в ясний день).

- Реалізований алгоритм на фоторезисторах достатньо точний (похибка кілька градусів) і швидкий для підтримки оптимального положення.

– Система стабільна в різних умовах освітлення; невеликі проблеми при хмарності не призвели до суттєвих збоїв чи втрати ефективності.

- Вбудоване IoT-рішення зарекомендувало себе як зручний інструмент для моніторингу та управління, підвищуючи практичність системи.

3.4 Висновки до третього розділу

В рамках практичної частини роботи було успішно реалізовано та протестовано дослідний зразок IoT-сонячного трекера. Проведені експерименти показали, що двоосьове відстеження забезпечує близько 40%приросту виробітку енергії порівняно зі стаціонарним модулем за ясної погоди, при цьому система підтримує достатню точність наведення (~±3°). В умовах часткової хмарності трекер також функціонує, хоч і з трохи нижчою ефективністю, але без некоректної поведінки. Інтеграція з мережею Інтернет дозволила здійснювати віддалений контроль, що підвищує безпечність (можливість паркування при негоді) та полегшує збір даних для аналізу. Отримані результати підтверджують доцільність впровадження подібних автоматизованих систем для підвищення ефективності сонячних установок, особливо автономних, де кожен додатковий ват-година цінний. У наступному розділі узагальнимо результати роботи та окреслимо можливі напрями вдосконалення даної системи.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Ергономічні проблеми безпеки життєдіяльності

Ергономіка як наукова дисципліна відіграє важливу роль у забезпеченні безпеки життєдіяльності, особливо в умовах виробничого середовища та офісної праці. Її основна мета полягає в адаптації умов праці до фізіологічних, психологічних та анатомічних особливостей людини з метою зменшення професійних ризиків, підвищення ефективності праці та покращення якості життя працівників [42].

Серед основних ергономічних проблем, що впливають на безпеку життєдіяльності, виокремлюють невідповідність робочого місця антропометричним характеристикам працівника, нераціональне розташування засобів управління, надмірні фізичні або зорові навантаження, несприятливі мікрокліматичні умови, неадекватне освітлення, а також високий рівень шуму або вібрацій [43].

Зокрема, неправильна організація робочого місця може призводити до розвитку захворювань опорно-рухового апарату, зниження працездатності та підвищення ризику виробничого травматизму. Тривале перебування в статичному положенні без належної підтримки спини спричиняє перевантаження м’язів та хребта, що призводить до хронічного болю [44]. Водночас, розміщення робочого обладнання поза зоною комфортної досяжності створює додаткове навантаження на м'язово-скелетну систему.

Особливої уваги заслуговує проблема зорового навантаження, характерна для офісних працівників і операторів комп’ютерної техніки. Надмірне перебування перед екранами дисплеїв спричиняє синдром зорової втоми, який проявляється сухістю очей, погіршенням зору, головним болем та загальною втомою. За дослідженнями, понад 60% користувачів комп’ютерів відчувають ці симптоми вже після 2–3 годин роботи [45].

Ще одним не менш суттєвим аспектом є мікрокліматичні умови, що впливають на терморегуляцію організму, самопочуття працівника та його продуктивність. Надмірна температура, низька вологість або протяги можуть не лише спричинити дискомфорт, а й стати фактором ризику для серцево-судинної та дихальної систем. Оптимальні параметри мікроклімату на робочому місці визначені державними санітарними нормами (ДСН 3.3.6.042-99) і повинні дотримуватись у межах нормативів [46].

Значний вплив на безпеку має психоемоційне навантаження, яке виникає внаслідок монотонної діяльності, надмірної відповідальності або конфліктних ситуацій у трудовому колективі. Такі стани спричиняють розвиток хронічного стресу, що негативно позначається як на психологічному, так і на фізичному здоров'ї працівника, знижує його уважність і, відповідно, підвищує ймовірність помилок [47].

Важливою складовою ергономічного підходу до безпеки є організація режиму праці та відпочинку. Наявність перерв, чергування типів діяльності, можливість змінювати позу або переміщуватись у межах робочого місця сприяє зменшенню втоми та запобігає професійному вигоранню [48].

Застосування ергономічних принципів під час проєктування робочих місць, програмного забезпечення, засобів захисту та інтерфейсів управління дозволяє мінімізувати шкідливі впливи на організм людини, підвищити безпеку праці та знизити ризики травматизму. Відповідно, інтеграція ергономіки у систему управління охороною праці є необхідною умовою формування безпечного та здорового виробничого середовища.

4.2 Заходи з техніки безпеки при експлуатації обладнання

Створення безпечних умов праці під час експлуатації обладнання є важливою складовою системи охорони праці в галузі інформаційних технологій. Навіть у, здавалося б, «офісному» середовищі зношеність, неправильне використання чи нехтування технічним обслуговуванням комп'ютерного та

периферійного обладнання можуть призвести до аварійних ситуацій, коротких замикань, електротравм, а також до професійних захворювань, зокрема опорно-рухового апарату або зору [49].

Одним із найбільш важливих заходів є підтримання техніки у справному стані. Комп’ютери, принтери, мережеве обладнання та інші пристрої повинні регулярно проходити технічне обслуговування, очищення від пилу, перевірку на відсутність механічних пошкоджень, справність кабелів та блоків живлення. Експлуатація несправної апаратури заборонена, оскільки вона несе пряму загрозу здоров’ю працівника та цілісності IT-інфраструктури [50].

Важливим аспектом є інструктаж з охорони праці, який мають пройти всі працівники, що працюють із технічними засобами. Вступний, первинний та періодичний інструктажі повинні охоплювати правила безпечної роботи з офісною та комп’ютерною технікою, дії в разі перегріву пристрою, виникнення диму або запаху гару, а також вимоги щодо поводження з кабелями, розетками, подовжувачами тощо [51].

Особливу увагу слід приділити розміщенню обладнання. Комп’ютери та монітори мають бути встановлені на стійких поверхнях, з урахуванням вентиляції та віддаленості від джерел тепла або вологи. Необхідно уникати перевантаження електромережі, використовуючи сертифіковані мережеві фільтри та джерела безперебійного живлення (UPS). Серверне обладнання та маршрутизатори повинні розміщуватися у відповідних шафах із дотриманням правил циркуляції повітря [52].

Також необхідно враховувати ергономічні та психофізіологічні фактори (тривале сидіння, напруження зору та повторювані рухи) можуть спричинити хронічні захворювання. Робоче місце IT-фахівця має відповідати принципам ергономіки: правильна висота столу та стільця, належне розташування монітора, клавіатури та миші, а також дотримання режиму праці та перерв для зниження навантаження на організм [53].

Важливо дотримуватись інструкцій з експлуатації, які постачаються виробниками. У них зазначено допустимі режими роботи, правила підключення,

очищення, зберігання та вимоги до умов середовища (вологість, температура, запиленість). Ігнорування таких вказівок може призвести до поломок, втрати даних або ризику ураження електричним струмом.

Реалізація заходів з техніки безпеки в IT-середовищі повинна поєднувати належне технічне обслуговування обладнання, інформованість персоналу щодо потенційних загроз та дотримання норм експлуатації. Саме це дозволяє знизити виробничі ризики та підтримувати стабільність IT-інфраструктури.

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи розглянуто основні питання безпеки життєдіяльності та охорони праці у сучасному виробничому середовищі, зокрема ергономічні проблеми та заходи з техніки безпеки при експлуатації обладнання.

Аналіз ергономічних проблем показав, що саме неврахування антропометричних, фізіологічних та психологічних особливостей людини під час організації праці є однією з головних причин зниження працездатності, виникнення професійних захворювань та підвищення ризику травматизму. Водночас правильна ергономічна організація робочого простору сприяє покращенню умов праці, зменшенню втомлюваності, підвищенню продуктивності та безпеки роботи.

Забезпечення безпеки життєдіяльності вимагає системного підходу, що поєднує ергономічні, технічні, організаційні та психологічні складові. Лише всебічне дотримання вимог охорони праці може гарантувати створення безпечного, комфортного та ефективного робочого середовища, що відповідає сучасним стандартам виробничої культури та зберігає здоров’я працівників.

ВИСНОВКИ

У межах виконання кваліфікаційної роботи освітнього рівня «Бакалавр» виконано всебічне дослідження та практичну реалізацію комп’ютеризованої системи керування сонячними панелями з використанням технологій IoT.

Проаналізовано сучасні підходи до сонячного відстеження і обґрунтовано вибір методів для реалізації. З літературного огляду встановлено, що двоосьові трекери здатні збільшити виробництво електроенергії на 30–40% і більше порівняно з нерухомими системами. Проаналізовано, що для малих і середніх систем оптимально застосувати мікроконтролери з підтримкою бездротового зв’язку (ESP32) – це спрощує апаратну схему і полегшує інтеграцію IoT. Технології IoT все ширше застосовуються у сонячній енергетиці для моніторингу стану панелей та прогнозування їх продуктивності, що підтверджує актуальність напрямку роботи.

Розроблено проект двоосьового сонячного трекера з функціями IoT та виконано його реалізацію. Система побудована на мікроконтролері ESP32, що керує двома сервоприводами для повороту панелі і отримує сигнали від чотирьох датчиків освітленості.

Розроблено електричну схему, яка включає також сенсори для вимірювання напруги і струму панелі, модуль живлення та інтерфейс Wi-Fi.

Створено програмне забезпечення мікроконтролера, яке реалізує алгоритм порівняння освітленості датчиків і плавного наведення панелі за Сонцем з урахуванням порогу нечутливості та обмежень кута. Інтегровано можливості віддаленого моніторингу. Також реалізовано дистанційне керування – зокрема, перемикання режиму стеження, примусове вирівнювання панелі тощо. У макеті використано стандартні недорогі компоненти, а всі з’єднання виконано на макетній платі, що підтверджує доцільність і відносну простоту практичної реалізації проекту.

Виконано експериментальну перевірку роботи системи та отримано підтвердження її ефективності. Проведені випробування продемонстрували, що

розроблений трекер успішно утримує сонячну панель спрямованою на Сонце протягом дня. За результатами вимірювань, добовий виробіток енергії зріс приблизно на 40% у порівнянні з нерухомою панеллю в тих самих умовах. Таким чином, експериментально доведено, що впровадження системи стеження значно підвищує продуктивність сонячної панелі.

Переваги IoT-функціоналу в системі підтверджено на практиці. Віддалений моніторинг дозволив в режимі реального часу спостерігати за параметрами роботи панелі (напруга, струм, потужність, положення приводів) та вчасно виявляти будь-які відхилення.

Усі ці фактори свідчать про те, що розроблена комп’ютеризована система керування з IoT є раціональним рішенням для підвищення ефективності сонячних установок..

ПЕРЕЛІК ДЖЕРЕЛ

1 Akinwale, Y., & Ojomo, M. (2021). Development of a solar tracker using Internet of Things (IoT). *Renewable Energy Technologies Journal*, 35(4), 210–218.

2 Гончар, Л. В., & Коваль, Д. І. (2022). Сенсорні системи для моніторингу енергоспоживання. Вісник КНУ, 5(2), 15–21.

3 Brown, R. (2020). *Practical Electronics for Inventors* (4th ed.). McGraw-Hill Education.

4 Lytvynenko, I., Lupenko, S., Kunanets, N., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, November). Simulation of gas consumption process based on the mathematical model in the form of cyclic random process considering the scale factors. In 1st International Workshop on Information Technologies: Theoretical and Applied Problems, ITTAP 2021.

5 Shinde, P., & Wagh, M. (2019). Review on IoT-based smart solar tracking systems. *International Journal of Engineering Sciences & Research Technology*, 8(5), 300–306.

6 Smith, J. (2020). *Solar Energy: The Physics and Engineering of Photovoltaic Conversion, Technologies and Systems*. Oxford University Press..

7 Гриценко, Р. П. (2020). Системи стеження за Сонцем: огляд і перспективи. Енергетика та автоматика, 12(4), 22–28.

8 Rao, C. K., Sahoo, S. K., & Yanine, F. F. (2023). An IoT-based intelligent smart energy monitoring system for solar PV power generation. Energy Harvesting and Systems, 11(1), 79–87. DOI: 10.1515/ehs-2023-0015.

9 Khan, M. J., & Iqbal, M. T. (2021). Dynamic modeling and control of a small-scale solar tracker system. *Renewable Energy*, 159, 1104–1113.

10 Дьяків, С. М. (2021). IoT як інструмент для дистанційного контролю енергетичних установок. Технічна електроніка, 9(1), 44–50.

11 . Soliman, H. M. (2021). Design and implementation of IoT-based solar panel positioning system. *Energy Engineering*, 118(6), 394–404.

12 Tzounis, A., Katsoulas, N., Bartzanas, T., & Kittas, C. (2017). Internet of Things in agriculture, recent advances and future challenges. *Biosystems Engineering*, 164, 31–48.

13 Thakur, M., & Verma, D. (2021). Application of ESP32 for real-time solar monitoring. *Microcontroller Journal*, 13(4), 174–182.

14 Ільчук, С. Г. (2021). Порівняння алгоритмів позиціонування сонячних трекерів. Вісник Вінницького політехнічного інституту, 3, 55–60.

15 Batayneh, W., Owais, A., & Nairoukh, M. (2013). An intelligent fuzzy based tracking controller for a dual-axis solar PV system. Automation in Construction, 29, 100–106. DOI: 10.1016/j.autcon. 2012.09.012.

16 Жук, Т. Б. (2018). Програмування мікроконтролерів у системах керування. Київ: АртКнига.

17 Wang, L., & Zhang, Y. (2020). A novel IoT-enabled dual-axis solar tracking method. *IEEE Transactions on Sustainable Energy*, 11(2), 725–734.

18 Sharma, K., & Gupta, P. (2021). Efficient dual-axis solar tracking using PID controller. *Energy and Power Engineering*, 13(5), 225–233

19 . Мельник, Ю. Г. (2020). Застосування Blynk для енергомоніторингу. Сучасні інформаційні технології, 5(2), 61–66

20 Nunes, R., & Lima, M. (2023). IoT-based solar energy harvesting system using INA219 and ESP32. *Sensors and Actuators A: Physical*, 335, 113–122.

21 Hameed, B. H., & Kurnaz, S. (2024). Secure low-cost photovoltaic monitoring system based on LoRaWAN network and artificial intelligence. Discover Computing, 27(36), 1-20. DOI: 10.1007/s10791-024-09475-0.

22 Adhya, S., Saha, D., Das, A., Jana, J., & Saha, H. (2016). IoT based smart solar photovoltaic remote monitoring and control unit. In 2016 2nd International Conference on Control, Instrumentation, Energy & Communication (CIEC) (pp. 432-436). IEEE. DOI: 10.1109/CIEC. 2016.7513819.

23 Vinodhkumar, R., & John, S. (2020). IoT based dual-axis solar tracking system. IOP Conference Series: Materials Science and Engineering, 912, 032024. DOI: 10.1088/1757-899X/912/3/032024.

24 Nath, D. C., Kundu, I., Sharma, A., et al. (2023). Internet of things integrated with solar energy applications: A state-of-the-art review. Environment, Development and Sustainability, 26(10), 8479–8513. DOI: 10.1007/s10668-023-03691-2

25 Muthukumar, P., Manikandan, S., Rathinam, M., Jarin, T., & Sebi, A. (2023). Energy efficient dualaxis solar tracking system using IoT. Measurement: Sensors, 28, 100825. DOI: 10.1016/j.measen. 2023.100825.

26 Efendi, M., Mainil, R. I., & Aziz, A. (2025). Comparison of the efficiency of solar PV fixed, singleaxis, and dual-axis solar trackers: A review. International Journal of Renewable Energy Research, 15(1), 123-131..

27 Литвин, О. Л. (2021). Практичні аспекти використання ESP32 в автоматизованих системах. Наука і техніка, 8(1), 48–53..

28 Кульчицький С., Голотенко О., Станько А. Інтеграція супутникових та наземних бездротових мереж: архітектурні концепції. ВИМІРЮВАЛЬНА ТА ОБЧИСЛЮВАЛЬНА ТЕХНІКА В ТЕХНОЛОГІЧНИХ ПРОЦЕСАХ. 2025. Т. 4.

29 Fang, X., Misra, S., Xue, G., & Yang, D. (2019). Smart grid—The new and improved power grid: A survey. *IEEE Communications Surveys & Tutorials*, 14(4), 944–980..

30 ШИМЧУК, Г., ШЕВЧЕНКО, Н., ШВИРЛО, К., & ГАРМАТЮК, Н. (2025). СИСТЕМА ВІДНОВЛЕННЯ ДАНИХ У БЕЗДРОТОВИХ СЕНСОРНИХ МЕРЕЖАХ НА ОСНОВІ МАШИННОГО НАВЧАННЯ. Herald of Khmelnytskyi National University. Technical sciences, 353(3.2), 246-250.

31 Stanko, A., Wieczorek, W., Mykytyshyn, A., Holotenko, O., & Lechachenko, T. (2024). Realtime air quality management: Integrating IoT and Fog computing for effective urban monitoring. CITI, 2024, 2nd.

32 Leshchyshyn, Y., Scherbak, L., Nazarevych, O., Gotovych, V., Tymkiv, P., & Shymchuk, G. (2019, May). Multicomponent Model of the Heart Rate Variability Change-point. In 2019 IEEE XVth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH) (pp. 110-113). IEEE.

33 Lytvynenko, I., Lupenko, S., Nazarevych, O., Shymchuk, G., & Hotovych, V. (2021, September). Mathematical model of gas consumption process in the form of

cyclic random process. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 232-235). IEEE.Park, J., & Lee, S. (2021). Comparative performance analysis of single and dual-axis solar trackers. *Renewable and Sustainable Energy Reviews*, 140, 110–118.

34 О Голотенко, А Бойчун, «Розробка автоматизованої системи моніторингу мікроклімату складських приміщень транспортної компанії з використанням технологій IoT», Матеріали XI науково-технічної конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя. – 2023. – С. 146.

35 Duda, O., Kochan, V., Kunanets, N., Matsiuk, O., Pasichnyk, V., Sachenko, A., & Pytlenko, T. (2019, September). Data processing in IoT for smart city systems. In 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS) (Vol. 1, pp. 96-99). IEEE.

36 Hammas, M., Fituri, H., Shour, A., Khan, A. A., Khan, U. A., & Ahmed, S. (2025). A hybrid dual-axis solar tracking system: Combining light-sensing and time-based GPS for optimal energy efficiency. Energies, 18(1), 217. DOI: 10.3390/en18010217.

37 Majeed, R., Waqas, A., Sami, H., Ali, M., & Shahzad, N. (2020). Experimental investigation of soiling losses and a novel cost-effective cleaning system for PV modules. Solar Energy, 201, 298–306. DOI: 10.1016/j.solener.2020.03.115.

38 Dandu, R., & Thangam, S. (2023). IoT based single axis solar tracker. In Proceedings of the 14th International Conference on Computing, Communication and Networking Technologies (ICCCNT), New Delhi, India (pp. 1–5). IEEE. DOI: 10.1109/ICCCNT57752.2023.10195512.

39 Boucif, O. H., Lahouaou, A. M., Boubiche, D. E., & Toral-Cruz, H. (2025). Artificial Intelligence of Things for solar energy monitoring and control: A comprehensive survey. Applied Sciences, 15(11), 6019. DOI: 10.3390/app15116019.

40 Шевчук, Л. М. (2021). Підвищення ефективності фотовольтаїчних систем з трекінгом. Вісник енергетики, 10(2), 12–17.

41 Державні санітарні правила і норми «Гігієнічні вимоги до умов праці при роботі з відеодисплейними терміналами» – ДСанПіН 3.3.2.007–98.

42 Левченко І.Л. Ергономічна оцінка умов праці оператора: навчальний посібник. – Львів: Видавництво ЛНУ, 2020. – 156 с.

43 Ткаченко О.П. Основи охорони праці: підручник. – Харків: ХНАДУ, 2021. – 312 с.

44 ДСТУ 8604:2015. Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги. – [Чинний від 01.07.2017]. – К.: ДП «УкрНДНЦ», 2017. – 10 с.

45 ДСН 3.3.6.042-99. Державні санітарні норми мікроклімату виробничих приміщень.

46 Сидоренко В.В. Психофізіологічні основи безпеки праці. – К.: Кондор, 2017. – 192 с.

47 Єрмоленко Л.А. Охорона праці: навч. посіб. – Дніпро: ДНУ, 2019. – 240 с.

48 Закон України «Про охорону праці» від 14.10.1992 № 2694-XII (із змінами).

49 Яворовський О. П., Шевцова В. М., Зенкіна В. І. Безпека життєдіяльності, основи охорони праці : навч. посіб. / за заг. ред. О. П. Яворовського. – К. : Медицина, 2015. – 288 с.

50 Мацюк В.І., Баранов В.В. Охорона праці: навч. посіб. – К.: Центр учбової літератури, 2020. – 348 с.

51 Гандзюк М.П., Желібо Є.П., Халімовський М.О. Основи охорони праці: Підручник. / За ред. М. П. Гандзюка. - К.: Каравела, 2008. - 384 с.

52 Пістун І.П. Безпека життєдіяльності (психофізіологічні аспекти). Практичні заняття. – Львів: Афіша, 2000. 240 с.

ДОДАТКИ

Додаток А

Лістинг двоосьового трекера з Blynk, MQTT, EEPROM, PID та обмеженням часу

```
#include <Servo.h>
#include <Wire.h>
#include <Adafruit_INA219.h>
#include <BlynkSimpleEsp32.h>
#include <EEPROM.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <PID_v1.h>

// Піни
const int ldrN = A0;
const int ldrE = A1;
const int ldrS = A2;
const int ldrW = A3;
const int azPin = 9;
const int altPin = 10;

Servo azServo;
Servo altServo;
Adafruit_INA219 ina219;

// Параметри PID
double inputAz, outputAz, setpointAz = 0;
double inputAlt, outputAlt, setpointAlt = 0;
double Kp=0.2, Ki=0.05, Kd=0.01;
PID pidAz(&inputAz, &outputAz, &setpointAz, Kp, Ki, Kd, DIRECT);
PID pidAlt(&inputAlt, &outputAlt, &setpointAlt, Kp, Ki, Kd, DIRECT);
```

```
// WiFi та Blynk
char auth[] = "YourBlynkToken";
char ssid[] = "YourSSID";
char pass[] = "YourPassword";

// MQTT
const char* mqtt_server = "broker.hivemq.com";
WiFiClient espClient;
PubSubClient client(espClient);

// Стан
int azAngle = 90;
int altAngle = 45;
unsigned long lastTrack = 0;
unsigned long lastSend = 0;

void setup() {
  Serial.begin(115200);
  Wire.begin();
  ina219.begin();

  azServo.attach(azPin);
  altServo.attach(altPin);

  EEPROM.begin(512);
  azAngle = EEPROM.read(0);
  altAngle = EEPROM.read(1);
  azServo.write(azAngle);
  altServo.write(altAngle);
```

```
  Blynk.begin(auth, ssid, pass);

  setup_wifi();
  client.setServer(mqtt_server, 1883);

  pidAz.SetMode(AUTOMATIC);
  pidAlt.SetMode(AUTOMATIC);
}

void setup_wifi() {
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
}

void loop() {
  Blynk.run();
  client.loop();

  unsigned long now = millis();

  // Логіка по часу доби (працювати з 6:00 до 20:00 умовно)
  int hour = hourNow(); // реалізуйте цю функцію через RTC або API
  if (hour >= 6 && hour <= 20) {
    if (now – lastTrack > 1000) {
      trackSun();
```

```
      lastTrack = now;
    }
  } else {
    parkPanel();
  }

  if (now – lastSend > 30000) {
    sendTelemetry();
    lastSend = now;
  }
}

void trackSun() {
  int n = analogRead(ldrN);
  int e = analogRead(ldrE);
  int s = analogRead(ldrS);
  int w = analogRead(ldrW);

  inputAz = (double)(e – w);
  inputAlt = (double)(s – n);

  pidAz.Compute();
  pidAlt.Compute();

  azAngle = constrain(azAngle + (int)outputAz, 0, 180);
  altAngle = constrain(altAngle + (int)outputAlt, 0, 90);

  azServo.write(azAngle);
  altServo.write(altAngle);
```

```
  EEPROM.write(0, azAngle);
  EEPROM.write(1, altAngle);
  EEPROM.commit();
}

void sendTelemetry() {
  float current = ina219.getCurrent_mA();
  float power = ina219.getPower_mW();

  Blynk.virtualWrite(V0, current);
  Blynk.virtualWrite(V1, power);
  char msg[64];
  sprintf(msg, "{\"current\":%.2f,\"power\":%.2f}", current, power);
  client.publish("tracker/data", msg);
}
void parkPanel() {
  azServo.write(90);
  altServo.write(0);
}
int hourNow() {
  // заглушка, повертає 12:00, замініть на RTC або API
  return 12;
}
```