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 систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютерна система моніторингу стану акумуляторів з прогнозуванням їх деградації*

Виконав: студент *IV* курсу, групи *СІ-41*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

		Брайляк Д.В.
	(підпис)	(прізвище та ініціали)
Керівник		*Тиш Є.В.*
	(підпис)	(прізвище та ініціали)
Нормоконтроль		*Луцик Н.С.*
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		*Осухівська Г.М.*
	(підпис)	(прізвище та ініціали)
Рецензент		*Дмитроца Л.П.*
	(підпис)	(прізвище та ініціали)

Тернопіль
2026

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 систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« *25* » *квітня* 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Брайляку Дмитру Володимировичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютерна система моніторингу стану акумуляторів з прогнозуванням їх деградації*

Керівник роботи *Тиш Євгенія Володимирівна, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *24* » *квітня* 2026 року № *4.9-188*

2. Термін подання студентом завершеної роботи *15.06.2026 р.*

3. Вихідні дані до роботи *Li-ion 18650 акумуляторний елемент, Raspberry PI 5, датчик температури, моделі машинного навчання, мова програмування Python*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз принципів функціонування та вимог до систем моніторингу стану акумуляторів. 2. Проєктування системи моніторингу стану акумуляторів на апаратному рівні. 3. Реалізація програмних модулів системи моніторингу стану акумуляторів. Безпека життєдіяльності, основи охорони праці. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Будова Li-ion акумуляторного елементу та графік його деградації
2. Взаємозв'язок параметрів моніторингу та показників стану акумулятора
3. Архітектура комп'ютерної системи моніторингу стану акумуляторів з прогнозуванням їх деградації
4. Структурна схема комп'ютерної системи моніторингу стану акумуляторів

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	*Сенчишин В.С., к.т.н., доц., каф. МТ*		
основи охорони праці			

7. Дата видачі завдання *25.04.2026 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Розробка технічного завдання*	*26.01 – 02.02*	
2.	*Робота над першим розділом «Аналіз принципів функціонування та вимог до систем моніторингу стану акумуляторів»*	*03.02 – 15.02*	
3.	*Робота над другим розділом «Проєктування системи моніторингу стану акумуляторів на апаратному рівні»*	*20.04 – 25.04*	
4.	*Робота над третім розділом «Реалізація програмних модулів системи моніторингу стану акумуляторів»*	*26.04 – 05.05*	
5.	*Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»*	*07.05 – 25.05*	
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*26.05 – 7.06*	
7.	*Перевірка на академічний плагіат, перевірка керівником та консультантами*	*8.06 – 14.06*	
8.	*Попередній захист кваліфікаційної роботи бакалавра*	*15.06 – 21.06*	
9.	*Захист кваліфікаційної роботи бакалавра*	*22.06.2026*	

Студент _____ *Брайляк Дмитро Володимирович*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Тиш Євгенія Володимирівна*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Брайляк Д.В. Комп’ютерна система моніторингу стану акумуляторів з прогнозуванням їх деградації: робота на здобуття ступеня бакалавра: спец. 123 – комп’ютерна інженерія. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя. 2026.

Ключові слова: комп’ютерна система, акумулятор, Li-ion 18650, моніторинг, деградація, Raspberry Pi 5.

У кваліфікаційній роботі розроблено комп’ютерну систему моніторингу стану акумулятора з прогнозуванням його деградації. Проаналізовано особливості роботи Li-ion акумуляторів типорозміру 18650, основні причини втрати їх початкових характеристик та показники оцінювання технічного стану акумулятора.

Для реалізації системи обрано Li-ion акумулятор 18650, одноплатний комп’ютер Raspberry Pi 5, модуль вимірювання напруги та струму INA219, цифровий датчик температури DS18B20, модуль заряджання TP4056 та навантажувальний резистор 7,5 Ом / 10 Вт для контрольованого розряду. Розроблено апаратну архітектуру системи, структурну схему підключення компонентів та програмне забезпечення для зчитування, нормалізації, збереження й аналізу вимірювальних даних.

Програмна частина системи забезпечує запис поточних вимірювань у CSV-файл, формування журналу циклів розряду, розрахунок фактичної ємності акумулятора та показника SoH. Для прогнозування деградації використано модель на основі дерева прийняття рішень, яка класифікує стан акумулятора як нормальний, деградований або критичний.

ANNOTATION

Brailiak D.V. Computer system for battery condition monitoring with degradation prediction: Bachelor's Graduation Thesis: speciality 123 — computer engineering. Ternopil: Ternopil Ivan Puluj National Technical University, 2026.

Keywords: computer system, battery, Li-ion 18650, monitoring, degradation, Raspberry Pi 5.

The qualification thesis develops a computer system for monitoring the state of a battery with prediction of its degradation. The operating features of Li-ion batteries of the 18650 form factor, the main causes of loss of their initial characteristics, and the indicators used to assess the technical state of a battery are analyzed.

To implement the system, a Li-ion 18650 battery, a Raspberry Pi 5 single-board computer, an INA219 voltage and current measurement module, a DS18B20 digital temperature sensor, a TP4056 charging module, and a 7.5 Ohm / 10 W load resistor for controlled discharge were selected. The hardware architecture of the system, the structural connection diagram of the components, and the software for reading, normalizing, storing, and analyzing measurement data were developed.

The software part of the system provides recording of current measurements to a CSV file, creation of a discharge cycle log, calculation of the actual battery capacity, and determination of the SoH indicator. A decision tree-based model is used to predict degradation, classifying the battery state as normal, degraded, or critical.

ЗМІСТ

ВСТУП

Акумуляторні джерела живлення сьогодні є важливою складовою більшості сучасних електронних, мобільних, вимірювальних, робототехнічних та енергетичних систем. Вони використовуються у портативній електроніці, системах резервного живлення, електротранспорті, автономних сенсорних вузлах, безпілотних пристроях, побутовій техніці та засобах інтернету речей. Одним із найпоширеніших типів акумуляторних елементів є літій-іонні елементи формату 18650, які відзначаються достатньо високою питомою енергоємністю, компактними розмірами, стабільними електричними характеристиками та можливістю багаторазового використання.

Разом із тим, у процесі експлуатації акумулятори поступово втрачають свої початкові характеристики. На їхній технічний стан впливають кількість циклів заряду-розряду, температурний режим, глибина розряду, величина струму навантаження, умови заряджання та тривалість зберігання. Деградація акумулятора проявляється у зменшенні фактичної ємності, зростанні внутрішнього опору, підвищенні нагрівання під час роботи, скороченні часу автономного живлення та збільшенні ризику нестабільної роботи пристрою. Тому своєчасний контроль параметрів акумулятора є важливою умовою безпечної та ефективної експлуатації електронних систем.

У багатьох практичних застосуваннях контроль стану акумуляторів обмежується лише вимірюванням поточної напруги або приблизним визначенням рівня заряду. Такий підхід не дає повної інформації про реальний технічний стан елемента, оскільки напруга не завжди прямо відображає залишкову ємність і ступінь зношення. Для більш обґрунтованого оцінювання необхідно враховувати сукупність параметрів, зокрема напругу, струм, температуру, тривалість роботи, характер зміни параметрів у часі та історію циклів заряду-розряду. Саме тому актуальним є створення комп’ютерних систем, які поєднують апаратний збір даних, програмну обробку вимірювань, візуалізацію результатів і прогнозування подальшої зміни стану акумулятора.

РОЗДІЛ 1 АНАЛІЗ ПРИНЦИПІВ ФУНКЦІОНУВАННЯ ТА ВИМОГ ДО СИСТЕМ МОНІТОРИНГУ СТАНУ АКУМУЛЯТОРІВ

1.1 Аналіз особливостей роботи Li-ion акумуляторів типорозміру 18650

Літій-іонні акумулятори належать до найбільш поширених перезаряджуваних джерел живлення, які використовуються у портативній електроніці, автономних пристроях, робототехнічних системах, електроінструментах, вимірювальному обладнанні та системах резервного живлення [1]. Їх широке застосування пояснюється високою питомою енергоємністю, порівняно малою масою, компактністю, низьким рівнем саморозряду та можливістю багаторазового циклічного використання.

Одним із поширених конструктивних форматів Li-ion акумуляторів є елемент типорозміру 18650. Таке позначення пов’язане з його геометричними параметрами: діаметр елемента становить приблизно 18 мм, а довжина — близько 65 мм [1]. Акумулятори цього формату мають циліндричну форму та можуть використовуватися як окремі джерела живлення або як складові частини акумуляторних батарей. У кваліфікаційній роботі елемент 18650 розглядається як окремий об’єкт моніторингу, що дає змогу зосередитися на вимірюванні його параметрів, аналізі стану та прогнозуванні деградації без ускладнення системи функціями балансування декількох комірок [1-3].

Типовий Li-ion акумулятор 18650 має номінальну напругу близько 3,6–3,7 В. У повністю зарядженому стані напруга одного елемента зазвичай становить близько 4,2 В. Під час розряду напруга поступово зменшується, а нижня межа безпечної роботи зазвичай перебуває в діапазоні 2,5–3,0 В залежно від типу елемента, виробника та умов експлуатації [2].

Вихід за допустимі межі напруги є небажаним, оскільки перезаряд або глибокий розряд можуть призвести до прискореного зношування акумулятора, зменшення його ємності або виникнення небезпечних режимів роботи [2].

Конструктивно Li-ion акумуляторний елемент складається з позитивного електрода, негативного електрода, сепаратора, електроліту, струмовідводів і зовнішнього металевого корпусу. Позитивний електрод зазвичай виготовляється на основі літійвмісних сполук, а негативний – на основі графіту або інших вуглецевих матеріалів. Сепаратор розділяє електроди, запобігаючи їх прямому контакту, але водночас забезпечує переміщення іонів літію між електродами. Електроліт створює середовище для перенесення іонів, а металевий корпус забезпечує механічний захист внутрішніх компонентів [3]. На рис. 1.1 наведено будову Li-ion акумуляторного елемента.

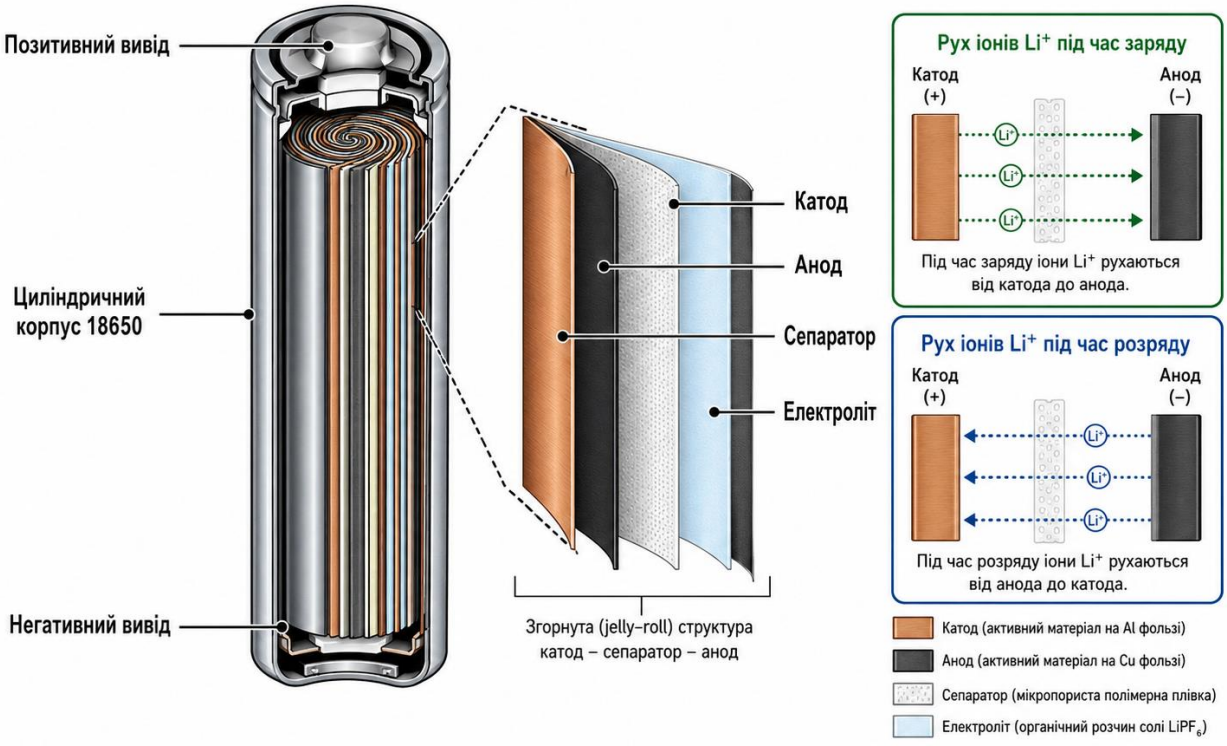

Рисунок 1.1 – Будова Li-ion акумуляторного елемента типорозміру 18650

Під час заряджання іони літію переміщуються від позитивного електрода до негативного, де вони накопичуються у структурі анодного матеріалу. Під час розряджання процес відбувається у зворотному напрямку: іони літію

повертаються до позитивного електрода, а у зовнішньому колі формується електричний струм, який живить навантаження. Саме повторюваність цих процесів забезпечує можливість багаторазового використання акумулятора [3].

Основними характеристиками Li-ion акумулятора є номінальна напруга, ємність, допустимий струм заряду та розряду, температурний діапазон роботи, внутрішній опір і кількість циклів експлуатації. Для системи моніторингу найбільше значення мають ті параметри, які можна вимірювати або розраховувати під час роботи елемента. До них належать напруга на клемах акумулятора, струм заряду або розряду, температура корпусу, тривалість роботи та зміна параметрів у часі [2, 3].

Напруга є одним із найпростіших для вимірювання параметрів, проте вона не дає повної інформації про реальний стан акумулятора. Два елементи можуть мати однакову напругу після заряджання, але різну фактичну ємність, різний внутрішній опір і різну поведінку під навантаженням. Тому в комп'ютерній системі моніторингу напруга повинна аналізуватися разом зі струмом, температурою та історією попередніх вимірювань.

Ємність акумулятора показує кількість електричного заряду, яку елемент може віддати під час розряду. Вона зазвичай вимірюється в мА·год. Для елементів 18650 типова ємність може перебувати в межах від 2000 до 3500 мА·год, хоча конкретне значення залежить від моделі та виробника. У процесі експлуатації фактична ємність поступово зменшується, тому її оцінювання є важливим для визначення технічного стану акумулятора [2].

Струм заряду та розряду впливає на тепловий режим і ресурс акумулятора. За надмірного струму елемент може нагріватися, а його характеристики погіршуються швидше. Температура також є важливим параметром, оскільки робота за несприятливих температурних умов може зменшувати доступну ємність і прискорювати старіння елемента. Тому в розроблюваній системі доцільно передбачити одночасний контроль напруги, струму та температури.

У табл. 1.1 наведено основні параметри Li-ion акумулятора типорозміру 18650, які є важливими для побудови системи моніторингу.

Таблиця 1.1 – Основні параметри Li-ion акумулятора типорозміру 18650

Параметр	Типове значення або діапазон	Значення для системи моніторингу
Номінальна напруга	3,6–3,7 В	Використовується як базова характеристика елемента
Максимальна напруга повного заряду	4,2 В	Дає змогу контролювати завершення заряду та уникати перезаряду
Нижня межа робочої напруги	2,5–3,0 В	Використовується для виявлення небезпечного глибокого розряду
Типова ємність	2000–3500 мА·год	Є основою для оцінювання втрати ємності та SoH
Робочий струм	Залежить від моделі елемента	Впливає на нагрівання, ресурс і швидкість розряду
Температура роботи	Орієнтовно від 0 до +45 °C під час заряду, ширший діапазон під час розряду	Контролюється для запобігання перегріву та прискореній деградації
Кількість циклів	Залежить від режимів експлуатації	Використовується як один із факторів прогнозування деградації
Внутрішній опір	Зростає під час старіння	Є непрямою ознакою погіршення технічного стану
Саморозряд	Невеликий порівняно з багатьма іншими типами акумуляторів	Впливає на зберігання та довготривалу експлуатацію

Поведінка Li-ion акумулятора під час розряду має нелінійний характер. Після повного заряджання напруга спочатку зменшується відносно швидко, далі протягом значної частини розряду залишається у робочому діапазоні, а наприкінці циклу знижується інтенсивніше. Саме кінцева ділянка розряду потребує особливої уваги, оскільки подальше використання акумулятора може призвести до глибокого розряду [4]. У проєктованій системі така ситуація повинна виявлятися програмно шляхом контролю напруги та формування відповідного попередження.

Для кваліфікаційної роботи вибір одного Li-ion елемента типорозміру 18650 є доцільним, оскільки він дозволяє побудувати безпечний і зрозумілий лабораторний прототип системи моніторингу. Такий підхід не потребує балансування кількох послідовно з’єднаних комірок і дає змогу основну увагу приділити збору даних, їх обробці, збереженню, візуалізації та прогнозуванню зміни технічного стану акумулятора.

1.2 Аналіз процесів деградації Li-ion акумуляторів та показників їх технічного стану

У процесі експлуатації Li-ion акумулятор поступово втрачає свої початкові характеристики. Такий процес називають деградацією акумулятора. Він пов’язаний із внутрішніми електрохімічними змінами, які виникають під час багаторазових циклів заряду-розряду, тривалого зберігання, роботи під навантаженням і впливу температурних факторів. Навіть за нормальних умов використання акумулятор з часом втрачає частину ємності, гірше утримує заряд, швидше нагрівається та має більшу просадку напруги під час роботи [4].

Для Li-ion акумулятора типорозміру 18650 деградація є важливою характеристикою, оскільки такі елементи часто використовуються у пристроях, де від стабільності джерела живлення залежить працездатність усієї системи. Якщо акумулятор втрачає значну частину початкової ємності, пристрій працює менше часу, може несподівано вимикатися або нестабільно працювати під навантаженням. Тому для практичного використання важливо контролювати не лише поточний рівень заряду, а й загальний технічний стан акумулятора [2].

Одним із найбільш наочних способів оцінювання поведінки Li-ion акумулятора є аналіз його розрядної характеристики. Під час розряду напруга елемента змінюється нерівномірно. Після повного заряджання вона перебуває поблизу верхньої межі робочого діапазону. На початковій ділянці розряду напруга зменшується відносно швидко, далі протягом значної частини циклу залишається у відносно стабільному робочому діапазоні, а наприкінці розряду починає знижуватися інтенсивніше [3]. Саме завершальна ділянка є особливо важливою для системи моніторингу, оскільки подальший розряд може призвести до небажаного глибокого розряду акумулятора. На рис. 1.2 продемонстровано криву розряду Li-ion акумулятора 18650.

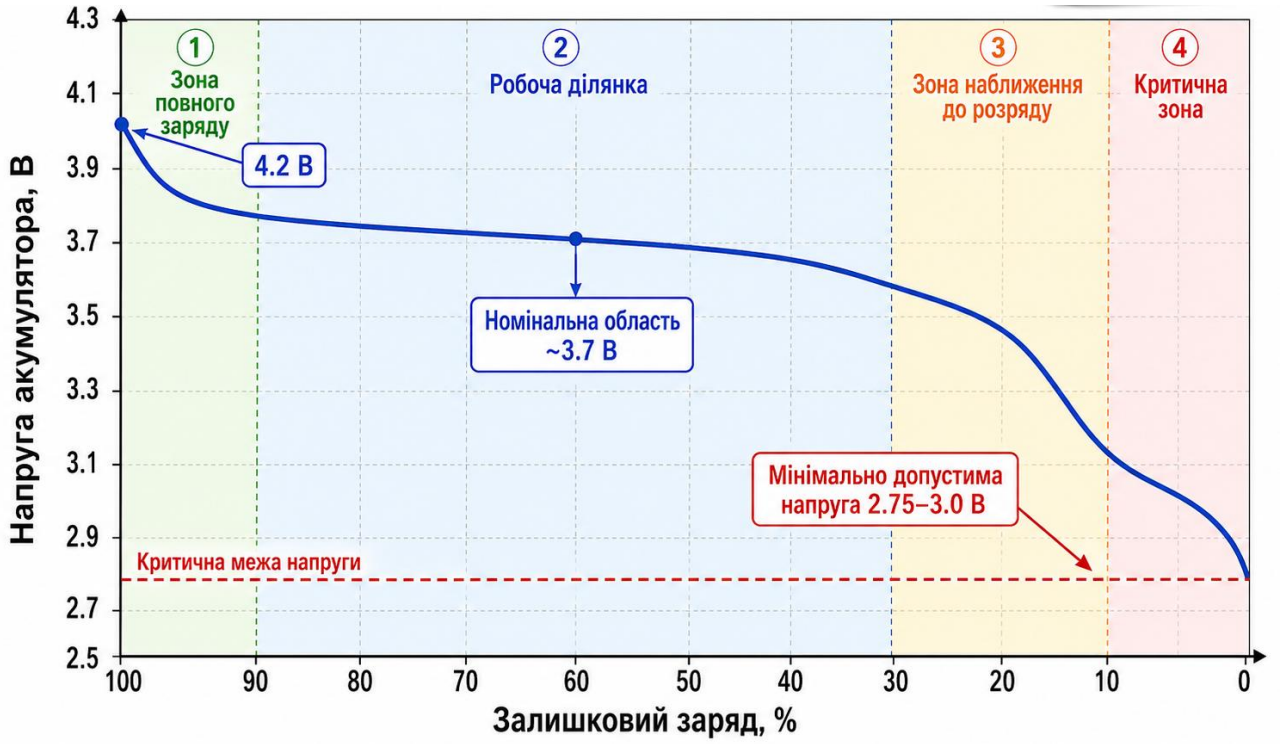

Рисунок 1.2 – Типова крива розряду Li-ion акумулятора 18650

Аналіз кривої розряду дозволяє зробити висновок, що поточна напруга акумулятора не завжди однозначно характеризує його технічний стан. Наприклад, два акумулятори можуть мати однакову напругу на певній ділянці розряду, але різну фактичну ємність і різний ступінь зношення [3]. Новий елемент здатний довше підтримувати робочу напругу під навантаженням, тоді як зношений акумулятор швидше переходить до ділянки різкого падіння напруги. Тому для більш повного оцінювання стану акумулятора необхідно враховувати не лише миттєве значення напруги, а й тривалість розряду, струм навантаження, температуру та історію попередніх циклів [4].

Основними факторами, що впливають на деградацію Li-ion акумулятора, є кількість циклів заряду-розряду, глибина розряду, величина струму, температурний режим, умови заряджання та тривалість перебування у повністю зарядженому або сильно розрядженому стані. У процесі багаторазового використання в акумуляторі поступово зменшується кількість активного матеріалу, змінюються властивості електродів і зростає внутрішній опір. У

результаті елемент уже не може накопичувати й віддавати таку саму кількість енергії, як у новому стані.

Одним із найбільш помітних проявів деградації є зменшення фактичної ємності акумулятора. Наприклад, елемент із номінальною ємністю 3000 мА·год після тривалої експлуатації може фактично віддавати значно меншу кількість заряду [4]. При цьому напруга повного заряду може залишатися близькою до 4,2 В, однак тривалість роботи під навантаженням буде меншою. Саме тому вимірювання лише напруги не дає повної інформації про реальний стан акумулятора.

Іншою важливою ознакою є збільшення внутрішнього опору. За його зростання під навантаженням спостерігається більша просадка напруги, а також інтенсивніше нагрівання елемента. Для користувача це проявляється у зменшенні часу автономної роботи, нестабільній роботі пристрою або швидкому падінні напруги під час підключення навантаження. У системі моніторингу така ознака може виявлятися непрямо – через аналіз зміни напруги, струму й температури у часі [3].

Температура також суттєво впливає на ресурс акумулятора. Підвищена температура прискорює процеси деградації, сприяє втраті ємності та може створювати небезпечні умови роботи. Низька температура, своєю чергою, зменшує доступну ємність і погіршує здатність акумулятора віддавати струм. Тому контроль температури є важливим елементом системи моніторингу, особливо під час заряду, розряду або роботи з підвищеним навантаженням.

Для оцінювання стану акумулятора використовують кілька основних показників. Найпоширенішими є State of Charge (SoC), State of Health (SoH) і Remaining Useful Life (RUL). Показник SoC характеризує поточний рівень заряду акумулятора. Він показує, яка частина енергії ще доступна для використання відносно повністю зарядженого стану [5]. Цей показник важливий для оперативного контролю, але він не показує, наскільки акумулятор зношений.

SoH характеризує загальний технічний стан акумулятора порівняно з його номінальним або початковим станом. Якщо новий акумулятор умовно має SoH

на рівні 100 %, то в процесі експлуатації цей показник поступово зменшується. У спрощеному вигляді SoH можна визначати через співвідношення фактичної та номінальної ємності акумулятора:

$$SoH = C_{\text{факт}} / C_{\text{ном}} \cdot 100\ \%, \quad (1.1)$$

де $C_{\text{факт}}$ – фактична ємність акумулятора, визначена під час тестування або розрахунку;

$C_{\text{ном}}$ – номінальна ємність, зазначена виробником.

RUL характеризує залишковий ресурс акумулятора до досягнення заданого критичного стану. На відміну від SoC і SoH, цей показник має прогнозний характер [5]. Для його визначення необхідно аналізувати не лише поточні значення параметрів, а й тенденцію їх зміни у часі. Саме тому прогнозування деградації потребує накопичення історії вимірювань.

Основні показники, які використовуються для оцінювання стану Li-ion акумулятора, наведено в табл. 1.2.

Таблиця 1.2 – Основні показники оцінювання стану Li-ion акумулятора

Показник	Повна назва	Зміст показника	Значення для проєктованої системи
SoC	State of Charge	Поточний рівень заряду акумулятора	Відображення поточного запасу енергії
SoH	State of Health	Технічний стан акумулятора порівняно з номінальним	Основний показник деградації
RUL	Remaining Useful Life	Залишковий ресурс до критичного стану	Використовується для прогнозування придатності
DoD	Depth of Discharge	Глибина розряду акумулятора	Впливає на кількість можливих циклів
C-rate	Струмовий режим відносно ємності	Характеризує інтенсивність заряду або розряду	Дає змогу оцінити навантаження на акумулятор
Внутрішній опір	—	Опір внутрішніх компонентів акумулятора	Непряма ознака старіння елемента

Для комп’ютерної системи моніторингу важливо розділити параметри на вимірювані та розрахункові. Напруга, струм і температура можуть бути отримані безпосередньо за допомогою датчиків, підключених до мікроконтролера. Кількість циклів заряду-розряду, фактична ємність, SoC, SoH і прогноз залишкового ресурсу визначаються програмно на основі накопичених вимірювальних даних [5, 6]. Процес деградації має накопичувальний характер, тому окреме вимірювання не дає повної картини технічного стану акумулятора. Наприклад, однакове значення напруги може відповідати різним станам елемента залежно від його фактичної ємності, температури, навантаження та попередньої історії використання. Натомість послідовність вимірювань у часі дозволяє виявити тенденції: швидше падіння напруги, скорочення тривалості розряду, підвищення температури або зменшення розрахованої ємності.

При проєктуванні системи деградацію акумулятора потрібно розглядати як поступове погіршення вимірюваних і розрахункових параметрів. Для одного Li-ion елемента 18650 це дозволяє побудувати безпечний лабораторний прототип, який накопичує дані про роботу акумулятора та використовує їх для подальшого аналізу. Отримані дані можуть бути застосовані для оцінювання SoH і побудови прогнозу зниження ресурсу акумулятора [6].

Таким чином, деградація Li-ion акумулятора є складним процесом, що залежить від електричних, температурних і часових факторів. При цьому необхідно комплексно аналізувати напругу, струм, температуру, кількість циклів, фактичну ємність і характер зміни параметрів у часі. Саме такий підхід покладено в основу комп’ютерної системи моніторингу стану акумулятора з прогнозуванням його деградації.

1.3 Аналіз вимог технічного завдання до комп’ютерної системи моніторингу стану акумуляторів

Основним завданням системи моніторингу стану акумуляторів з прогнозуванням їх деградації є автоматизований збір інформації про параметри

акумулятора під час його роботи. До таких параметрів належать напруга на клемах акумулятора, струм заряду або розряду, температура, тривалість циклу роботи та кількість циклів заряду-розряду. Саме ці дані є основою для подальшого розрахунку фактичної ємності, оцінювання показника SoH та прогнозування деградації акумулятора [5]. Отже, технічне завдання передбачає не лише вимірювання поточних значень, а й накопичення історії даних, що є важливим для виявлення змін у стані акумулятора в часі.

Апаратна частина системи повинна забезпечувати підключення акумулятора, засобів вимірювання та обчислювального модуля. Як основний керуючий та обчислювальний пристрій можна використати Raspberry Pi. Його застосування обґрунтовується тим, що цей одноплатний комп'ютер має достатню продуктивність для зчитування даних із датчиків, збереження результатів, виконання програмної обробки, побудови графіків і запуску нескладних моделей машинного навчання [5]. Крім того, Raspberry Pi підтримує інтерфейси підключення зовнішніх модулів, зокрема I2C, SPI, UART і GPIO, що спрощує інтеграцію з вимірювальними засобами.

Оскільки Raspberry Pi не має вбудованих аналогових входів, то необхідно передбачити використання цифрових вимірювальних модулів або аналогово-цифрових перетворювачів. Для вимірювання струму та напруги варто застосувати спеціалізований модуль, який може передавати дані до Raspberry Pi через цифровий інтерфейс [6]. Для контролю температури можна використати цифровий датчик температури або інший температурний сенсор, який забезпечує достатню точність для моніторингу акумулятора. Таким чином, апаратна частина повинна забезпечити безпечне та стабільне отримання вимірювальних даних без безпосереднього підключення аналогових сигналів до входів Raspberry Pi.

Важливою вимогою є безпечна робота з Li-ion акумулятором. Літій-іонні елементи чутливі до перезаряду, глибокого розряду, перегріву та перевищення допустимих струмів. Тому в системі необхідно передбачити програмний контроль граничних значень напруги та температури. У разі виходу параметрів

за допустимі межі система повинна формувати попередження користувачу. У кваліфікаційній роботі основний акцент робиться не на силовому захисті, як у повноцінній Battery Management System, а на моніторингу, аналізі та діагностиці стану акумулятора [6]. Проте дотримання безпечних режимів роботи є обов'язковою умовою експлуатації системи.

Програмна частина системи повинна виконувати кілька взаємопов'язаних функцій. Насамперед вона має забезпечувати регулярне зчитування даних із вимірювальних модулів. Отримані значення повинні містити часову мітку, напругу, струм, температуру та, за потреби, додаткові розрахункові параметри. Після зчитування дані мають проходити попередню обробку, що може включати перевірку коректності значень, усунення випадкових викидів, згладжування або усереднення вимірювань. Це необхідно для зменшення впливу шумів і підвищення якості подальшого аналізу [7].

Наступною вимогою є збереження вимірювальних даних. Для цього можуть використовуватися CSV-файли або локальна база даних, наприклад SQLite. Вибір конкретного способу збереження залежить від складності реалізації та обсягу даних. У кваліфікаційній роботі пропонується використання CSV-файлу. Це дозволить зберігати історію вимірювань, переглядати результати та використовувати їх для навчання або перевірки моделей прогнозування. Наявність історії даних є принципово важливою, оскільки деградація акумулятора проявляється не миттєво, а через зміну параметрів упродовж багатьох циклів роботи.

Окремим завданням програмної частини є розрахунок показників стану акумулятора. На основі виміряних значень струму та часу може визначатися кількість переданого заряду, а на основі цього – орієнтовна фактична ємність акумулятора. Порівняння фактичної ємності з номінальною дозволяє оцінити показник SoH. Крім того, система може розраховувати орієнтовний рівень заряду, фіксувати кількість циклів заряду-розряду та аналізувати зміну температури під час роботи.

Сукупність цих показників дає змогу сформувати більш повну картину технічного стану акумулятора, ніж звичайне вимірювання напруги.

Технічне завдання також передбачає реалізацію модуля прогнозування деградації акумулятора. У кваліфікаційній роботі можна застосувати просту модель машинного навчання, наприклад лінійну регресію, дерево рішень або інший алгоритм.

Інтерфейс користувача повинен забезпечувати зручне представлення результатів моніторингу. Доцільно передбачити відображення поточних значень напруги, струму, температури, орієнтовного рівня заряду та показника SoH. Також важливо реалізувати побудову графіків зміни параметрів у часі, оскільки графічне представлення дозволяє швидко оцінити тенденції роботи акумулятора. У разі досягнення критичних або наближених до критичних значень система повинна формувати попередження. Такий інтерфейс може бути реалізований у вигляді локального графічного застосунку або простої веб – сторінки, що працює на Raspberry Pi.

Розроблена система не є повноцінною промисловою BMS, оскільки не виконує балансування комірок і не реалізує складне силове керування. Водночас вона містить окремі діагностичні функції, характерні для систем моніторингу акумуляторів, і може бути використана як лабораторний стенд для дослідження стану окремих Li-ion елементів.

РОЗДІЛ 2 ПРОЄКТУВАННЯ СИСТЕМИ МОНІТОРИНГУ СТАНУ АКУМУЛЯТОРІВ НА АПАРАТНОМУ РІВНІ

2.1 Побудова архітектури системи моніторингу стану акумулятора

На основі проведеного аналізу предметної області, особливостей роботи Li-ion акумуляторів типорозміру 18650 та вимог технічного завдання можна сформувати загальну структуру комп'ютерної системи моніторингу стану акумулятора з прогнозуванням його деградації.

У проєктованій системі параметри акумулятора, які можуть бути безпосередньо виміряні або зафіксовані під час експлуатації, використовуються як вхідні дані для подальшої цифрової обробки. До таких параметрів, як зазначалось раніше, належать напруга на клемах акумулятора, струм заряду або розряду, температура та кількість або історія циклів заряду-розряду [8]. Саме ці величини є основою для формування узагальнених показників стану акумулятора, які більш повно характеризують його придатність до подальшої експлуатації.

Логіку переходу від вимірюваних параметрів до показників стану акумулятора наведено на рис. 2.1. Із цієї схеми видно, що напруга, струм, температура та цикли заряду-розряду надходять до модуля обробки даних. У межах цього модуля виконуються послідовні етапи збору даних, фільтрації, аналізу, оцінювання та моделювання. Результатом такої обробки є формування трьох основних вихідних показників: SoC, SoH і RUL [9]. Показник SoC характеризує поточний рівень заряду акумулятора, SoH відображає загальний технічний стан акумулятора порівняно з його номінальним станом, а RUL використовується для прогнозування залишкового ресурсу акумулятора.

Рисунок 2.1 – Взаємозв’язок параметрів моніторингу та показників стану акумулятора

Наведена на рис. 2.1 схема відображає інформаційну та аналітичну логіку функціонування системи, однак для її практичної реалізації необхідно визначити загальну апаратно-програмну архітектуру. У проєктованій системі об’єктом моніторингу є Li-ion акумулятор 18650, до якого підключаються засоби вимірювання напруги, струму та температури. Зібрані дані передаються до центрального обчислювального модуля, роль якого виконує Raspberry Pi [9]. Саме цей одноплатний комп’ютер забезпечує координацію роботи системи, зчитування даних, їх попередню обробку, збереження, аналітичні розрахунки та відображення результатів.

Загальна архітектура комп’ютерної системи повинна містити кілька взаємопов’язаних функціональних рівнів. Першим рівнем є рівень об’єкта моніторингу, тобто безпосередньо акумуляторного елемента. Другим є вимірювальний рівень, який включає модулі або датчики для контролю електричних і температурних параметрів. Третім рівнем є обчислювальний модуль на базі Raspberry Pi. Четвертий рівень становить програмне

забезпечення, яке виконує зчитування, фільтрацію, збереження та аналіз даних. П’ятим рівнем є модулі оцінювання стану акумулятора, прогнозування деградації та інтерфейс користувача.

Таким чином, система охоплює повний ланцюг перетворення інформації: від фізичних параметрів акумулятора до аналітичних показників і рекомендацій для користувача.

Архітектуру комп’ютерної системи моніторингу стану акумулятора наведено на рис. 2.2.

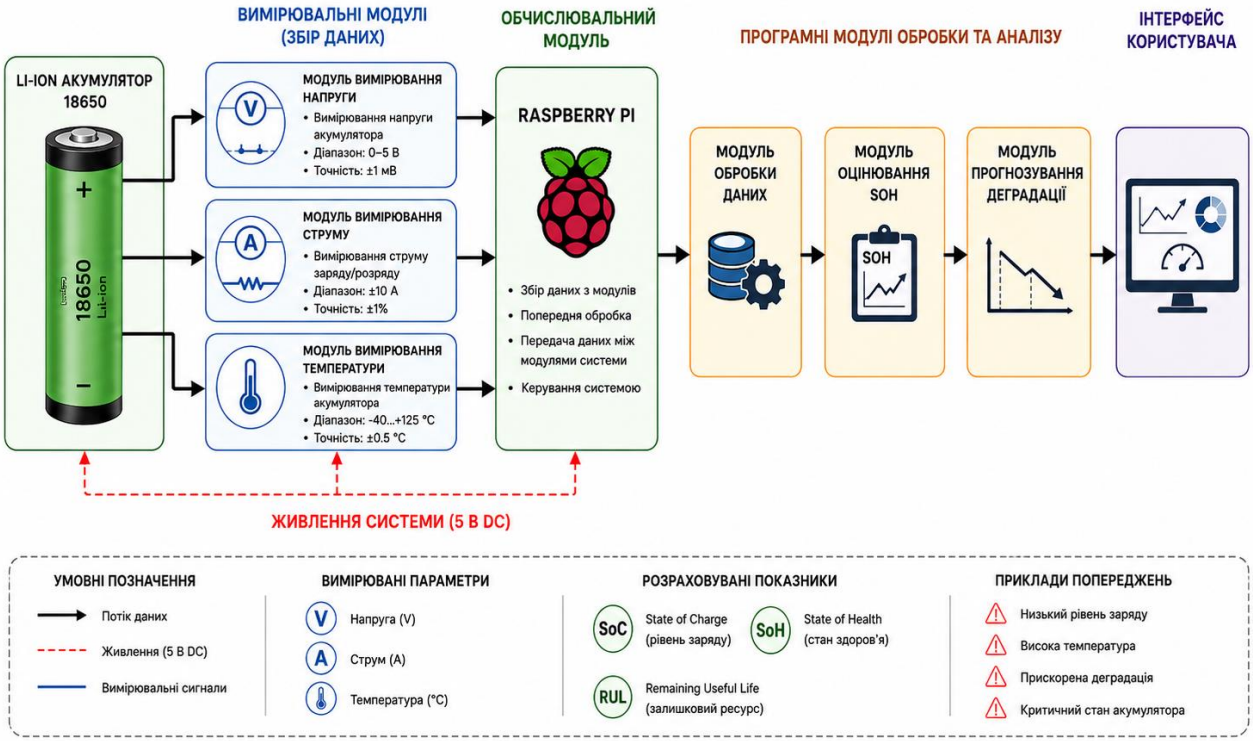

Рисунок 2.2 – Загальна архітектура комп’ютерної системи моніторингу стану акумулятора

Функціонально об’єкт моніторингу у системі представлений одним Li-ion акумуляторним елементом типорозміру 18650 [3]. Такий вибір дає можливість зосередитися на дослідженні процесів зміни електричних параметрів окремої комірки та аналізі її деградації в часі. Використання одного елемента також суттєво спрощує реалізацію системи, оскільки відсутня потреба у контролі

напруги кожної окремої комірки акумуляторного блока та у виконанні балансування.

Вимірювальна підсистема призначена для отримання первинних даних про роботу акумулятора. Напруга дозволяє контролювати поточний електричний стан елемента та визначати наближення до граничних режимів роботи. Струм є важливим параметром для оцінювання інтенсивності заряду або розряду, а також для подальшого розрахунку кількості переданого заряду й фактичної ємності. Температура характеризує тепловий режим акумулятора та дає змогу виявляти перевантаження, перегрів або інші небезпечні стани [9]. Додатковим інформаційним параметром є історія циклів заряду-розряду, яка відіграє важливу роль у визначенні ступеня зношування акумулятора [10].

Центральним елементом системи є Raspberry Pi. Його використання є обґрунтованим завдяки достатній обчислювальній потужності та широким можливостям інтеграції із зовнішніми модулями. Оскільки Raspberry Pi не має вбудованих аналогових входів, отримання електричних параметрів акумулятора повинно здійснюватися за допомогою зовнішніх вимірювальних модулів або аналогово-цифрових перетворювачів. Саме тому в архітектурі системи передбачається наявність окремого вимірювального вузла, який забезпечує коректне зчитування напруги та струму. Для контролю температури також передбачається використання окремого сенсора. Така побудова системи дозволяє чітко розділити рівень отримання фізичних параметрів і рівень їх цифрової обробки.

Програмна частина системи виконує ключову роль у забезпеченні функціонування всього комплексу. На першому етапі вона реалізує регулярне зчитування значень із вимірювальних модулів. Далі виконується попередня обробка даних, яка може включати перевірку коректності значень, усунення випадкових викидів, згладжування або усереднення результатів вимірювання. Після цього дані зберігаються у файл для формування історії спостережень. Наявність історичних даних є необхідною умовою для подальшого оцінювання стану акумулятора та прогнозування його деградації.

Окремий програмний модуль повинен забезпечувати розрахунок показників стану акумулятора. На основі вимірюваних напруги, струму, температури та часових характеристик система може визначати орієнтовний рівень заряду, фактичну ємність та показник SoH. Порівняння фактичної ємності з номінальною дозволяє оцінити ступінь зношення акумулятора. Також передбачено модуль прогнозування, який на основі накопичених даних і простої моделі машинного навчання забезпечує оцінювання та подальшу зміну стану акумулятора щодо критичного рівня деградації акумулятора.

Завершальною ланкою загальної структури системи є інтерфейс користувача. Він орієнтований на забезпечення зручного відображення поточних параметрів акумулятора, розрахункових показників стану та результатів прогнозування.

Запропонована загальна структура системи поєднує апаратні та програмні компоненти в єдину функціональну систему. З одного боку, вона охоплює вимірювання фізичних параметрів акумулятора, а з іншого – забезпечує їх цифрову обробку, накопичення історії вимірювань, оцінювання технічного стану та прогнозування деградації.

2.2 Вибір апаратних компонентів системи моніторингу стану акумулятора

Апаратна частина комп’ютерної системи моніторингу стану акумулятора повинна забезпечувати безпечне підключення Li-ion елемента, вимірювання його основних параметрів, передавання даних до обчислювального модуля та створення умов для подальшого програмного аналізу. Вибір компонентів виконується з урахуванням того, що система має працювати з одним акумуляторним елементом типорозміру 18650, а основні обчислювальні та керуючі функції покладаються на Raspberry Pi 5.

До складу апаратної частини системи входять: акумуляторний елемент 18650, одноплатний комп’ютер Raspberry Pi 5, модуль вимірювання напруги та струму, температурний датчик, модуль заряджання, навантаження для

контрольованого розряду, джерело живлення Raspberry Pi та допоміжні монтажні елементи.

2.2.1 Обчислювальний модуль Raspberry Pi 5

Центральним апаратним елементом системи обрано одноплатний комп’ютер Raspberry Pi 5 [11]. На нього покладаються функції зчитування даних із вимірювальних модулів, запуску програмного забезпечення, збереження результатів, побудови графіків, розрахунку показників стану акумулятора та виконання модуля прогнозування деградації.

Вибір Raspberry Pi 5 зумовлений його вищою продуктивністю порівняно з попередніми моделями Raspberry Pi, наявністю сучасних інтерфейсів підключення та можливістю роботи з повноцінною операційною системою. Для проєктованої системи це важливо, оскільки обчислювальний модуль повинен не лише опитувати датчики, а й виконувати програмну обробку вимірювань, зберігати часові ряди, будувати графіки та запускати алгоритми прогнозування деградації акумулятора [11].

Raspberry Pi 5 підтримує роботу з мовою програмування Python, бібліотеками для обробки даних, засобами візуалізації та простими моделями машинного навчання [12]. Це дозволяє реалізувати на одному пристрої повний програмний цикл роботи системи: зчитування параметрів акумулятора, перевірку коректності даних, запис результатів у файл, розрахунок показників SoC і SoH, формування прогнозу та відображення результатів користувачу.

Для взаємодії із зовнішніми вимірювальними модулями Raspberry Pi 5 має інтерфейси GPIO, I2C, SPI та UART. У даній системі особливе значення відіграє інтерфейс I2C, через який може бути підключений модуль вимірювання напруги та струму. Для температурного контролю може використовуватися цифровий датчик, підключений через GPIO [11]. Така комбінація дозволяє організувати збір вимірювальних даних без застосування складної проміжної електроніки.

Водночас потрібно врахувати, що Raspberry Pi 5, як і інші моделі цієї серії, не має вбудованих аналогових входів. Тому аналогові величини не можна подавати безпосередньо на його контакти. Для вимірювання напруги та струму

необхідно використовувати зовнішній цифровий вимірювальний модуль або аналогово-цифровий перетворювач. Саме тому в структурі системи передбачається застосування модуля для зняття показників напруги і струму, який передає результати вимірювань до Raspberry Pi 5 у цифровому вигляді [11].

Окрему увагу потрібно приділити живленню Raspberry Pi 5. Для стабільної роботи він повинен живитися від окремого джерела живлення 5 В з достатнім струмом. Досліджуваний Li-ion акумулятор 18650 у даній роботі не використовується як джерело живлення для Raspberry Pi, а розглядається як об’єкт моніторингу [12]. Такий підхід дозволяє уникнути впливу стану акумулятора на стабільність роботи обчислювального модуля та забезпечити коректність вимірювань. На рис. 2.3 проілюстровано Raspberry Pi 5.

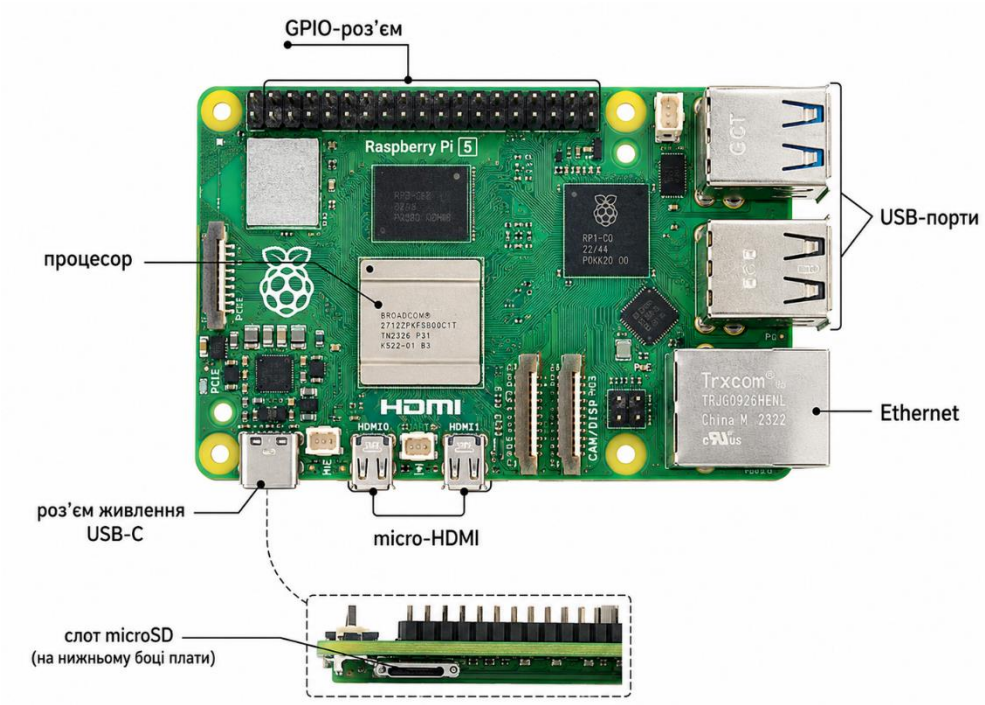

Рисунок 2.3 – Одноплатний комп’ютер Raspberry Pi 5

Raspberry Pi 5 побудований на основі процесора Broadcom BCM2712 із чотирма 64-бітними ядрами Arm Cortex-A76, що працюють на частоті 2,4 ГГц. Така апаратна основа забезпечує достатню продуктивність для локального виконання прикладних програм, обробки вимірювальних даних і побудови графіків без використання окремого персонального комп’ютера [12].

Плата оснащується оперативною пам’яттю LPDDR4X, слотом для microSD-карти, двома micro-HDMI виходами, USB-портами, мережевим інтерфейсом, бездротовими засобами зв’язку та 40-контактним GPIO-роз’ємом. Наявність цих інтерфейсів дозволяє використовувати Raspberry Pi 5 як компактний обчислювальний вузол у вбудованих системах моніторингу [13].

Особливістю Raspberry Pi 5 є поєднання високої обчислювальної продуктивності з можливістю прямої взаємодії із зовнішніми електронними модулями. Через GPIO-роз’єм доступні цифрові лінії введення-виведення та інтерфейси I2C, SPI і UART, що є важливим для підключення вимірювальних модулів системи.

У даній роботі це дає змогу підключити модуль вимірювання напруги та струму через I2C, а також температурний датчик через цифрову лінію даних. Водночас відсутність вбудованих аналогових входів потребує використання зовнішніх цифрових сенсорів або аналогово-цифрових перетворювачів, що враховується під час вибору інших апаратних компонентів системи.

2.2.2 Модуль вимірювання струму та напруги INA219

Для вимірювання електричних параметрів акумулятора запропоновано використати модуль INA219 [14]. Він призначений для цифрового вимірювання напруги, струму та потужності в колах постійного струму. У розроблюваній системі цей модуль виконує роль основного вимірювального вузла, через який отримуються дані про електричний режим роботи акумулятора.

Принцип роботи INA219 базується на вимірюванні падіння напруги на шунтовому резисторі. Коли через шунт протікає струм, на ньому виникає невелика різниця потенціалів, за якою мікросхема визначає величину струму. Одночасно модуль може вимірювати напругу на шині живлення. У результаті Raspberry Pi отримує цифрові значення напруги, струму та потужності через інтерфейс I^2C [14].

Використання INA219 є доцільним з кількох причин. По-перше, модуль сумісний із Raspberry Pi та передає дані у цифровому вигляді. По-друге, він дозволяє уникнути прямого вимірювання аналогових сигналів входами

Raspberry Pi. По-третє, отримані значення струму та напруги можуть бути безпосередньо використані для подальших розрахунків: визначення переданого заряду, оцінювання фактичної ємності та формування показника SoH [14].

Обмеженням INA219 є те, що він не визначає деградацію самостійно. Модуль лише надає первинні вимірювання, а всі висновки щодо технічного стану акумулятора формуються програмним забезпеченням [14]. Однак для запропонованого прототипу це є перевагою, оскільки дозволяє гнучко змінювати алгоритми обробки та прогнозування без зміни апаратної частини. На рис. 2.4 представлено модуль INA219.

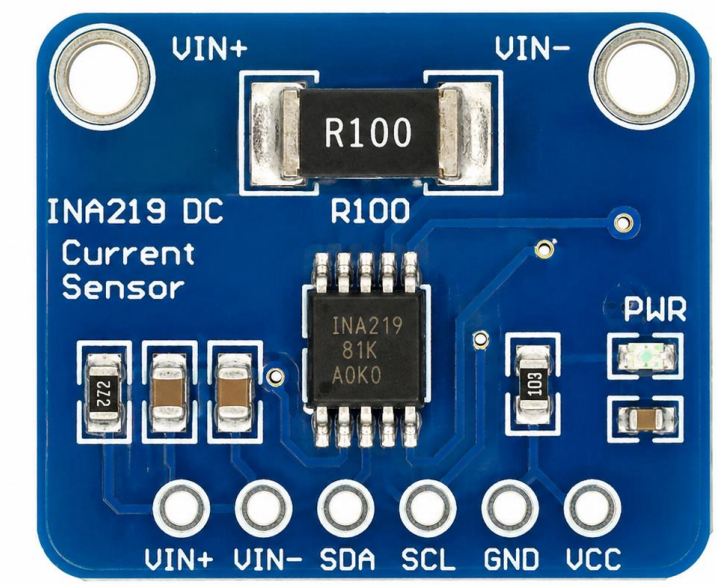

Рисунок 2.4 – Модуль вимірювання струму та напруги INA219

На рис. 2.4 показано мікросхему INA219, шунтовий резистор із маркуванням R100, світлодіод індикації живлення та контактні майданчики для підключення до вимірюваного кола і Raspberry Pi. Контакти VIN+ та VIN- призначені для включення модуля у силову частину кола між акумулятором і навантаженням, а контакти SDA та SCL використовуються для передавання даних через інтерфейс I2C. Контакти VCC і GND забезпечують живлення модуля.

У системі моніторингу стану акумуляторів INA219 може бути включений між акумулятором і навантаженням. Така схема (рис. 2.5) дає змогу контролювати струм розряду та напругу акумулятора під час тестування.

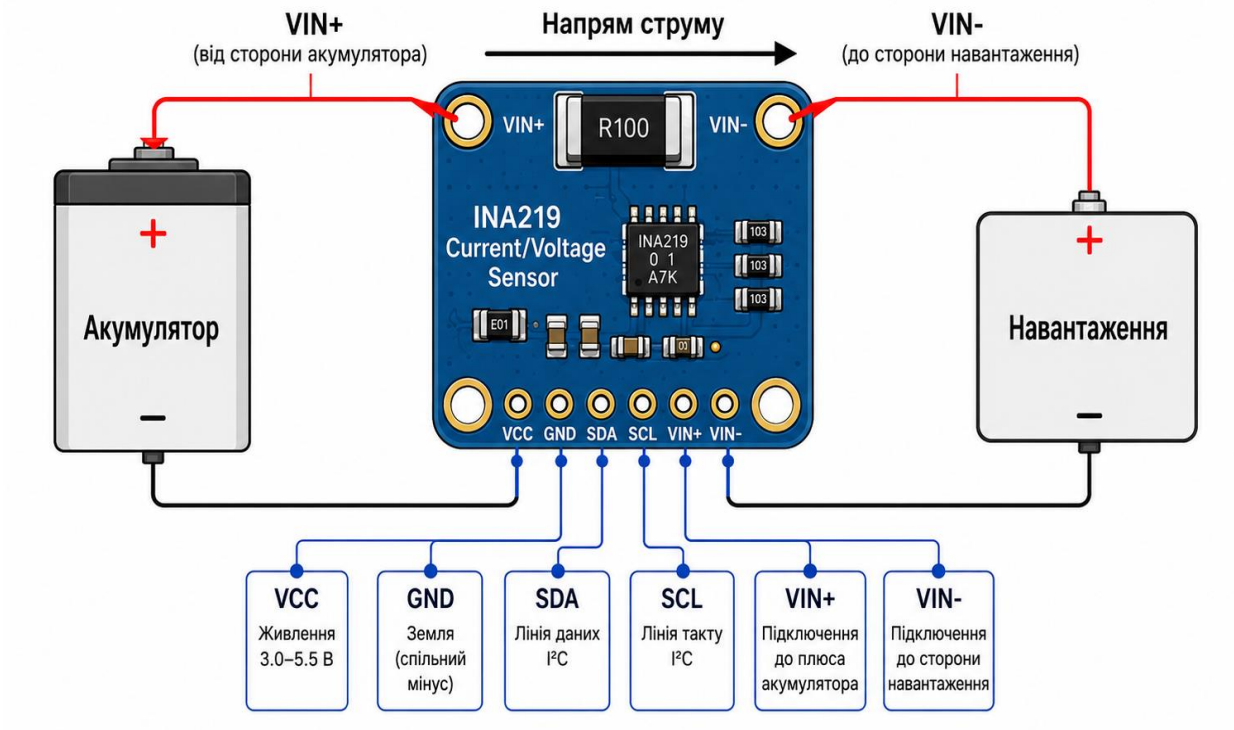

Рисунок 2.5 – Схема включення модуля INA219 у системі моніторингу стану акумулятора

Якщо додатково організувати періодичне зчитування даних, то можна отримати часовий ряд зміни електричних параметрів. Саме такі дані є основою для побудови графіків і аналізу деградаційних ознак акумулятора.

2.2.3 Датчик температури DS18B20

Температура є важливим параметром під час моніторингу Li-ion акумулятора [10, 13, 15]. Підвищення температури може свідчити про перевантаження, небажаний режим розряду, збільшення внутрішнього опору або погіршення технічного стану елемента. Тому в апаратній частині системи необхідно передбачити засіб температурного контролю.

Для вимірювання температури запропоновано використати цифровий датчик DS18B20. Він може передавати дані у цифровому вигляді, що зручно для

підключення до Raspberry Pi. На відміну від аналогових термісторів, DS18B20 не потребує окремого аналого-цифрового перетворювача. Це спрощує схему підключення та зменшує кількість додаткових компонентів.

Датчик DS18B20, який показаний на рис. 2.6, може бути розміщений безпосередньо біля корпусу акумулятора або зафіксований на його поверхні. Такий спосіб розміщення дозволяє контролювати зміну температури елемента під час розряду або заряджання [15]. У комп'ютерній системі моніторингу стану акумулятора достатньо вимірювати температуру корпусу акумулятора, оскільки саме вона відображає загальний тепловий стан елемента під час роботи.

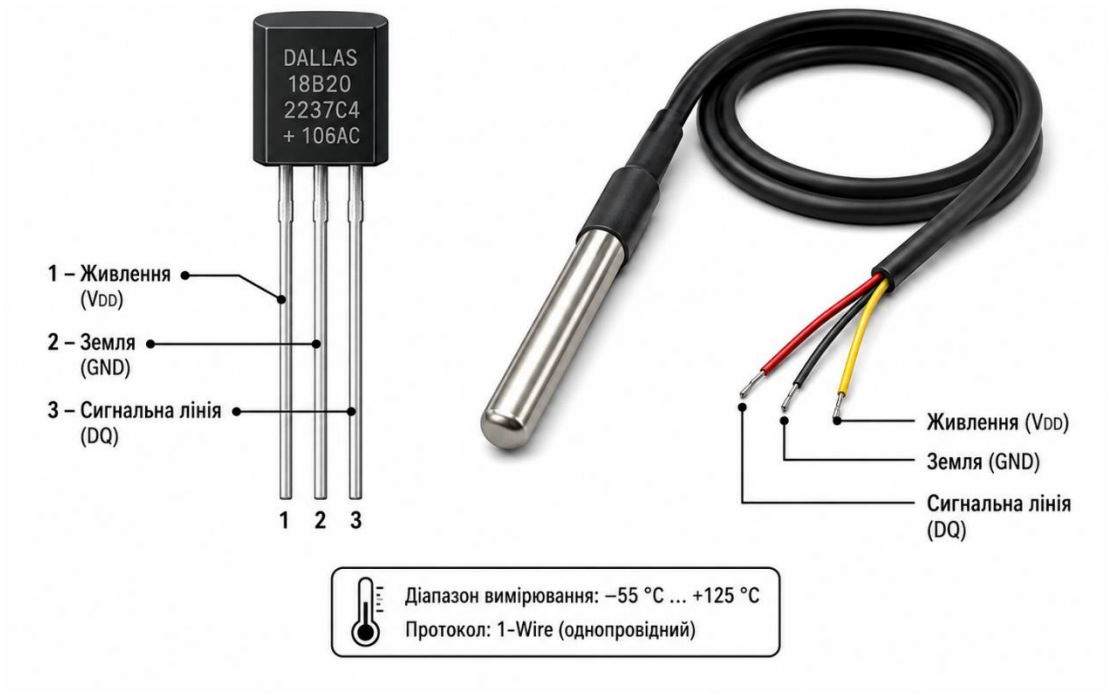

Рисунок 2.6 – Цифровий датчик температури DS18B20

Використання температурного датчика має не лише інформаційне, а й захисне значення. Якщо температура перевищує допустиме значення, програмне забезпечення може сформувати попередження для користувача. У подальшому така функція може бути розширена до автоматичного вимкнення навантаження або припинення тестового циклу.

2.2.4 Вибір модуля заряджання TP4056

Для заряджання одного Li-ion акумуляторного елемента 18650 використано модуль TP4056 [16]. Він призначений для заряджання однокоміркових літій-іонних акумуляторів і широко застосовується у портативних електронних пристроях. При проєктуванні системи моніторингу стану акумулятора цей модуль застосовується як окремий вузол для підготовки акумулятора до тестування або організації циклів заряду.

Модуль TP4056 (рис. 2.7) дозволяє заряджати акумулятор до напруги повного заряду. Для практичної реалізації використовується версія плати із захистом, яка має додаткові виводи OUT+ та OUT-. Такі модулі можуть містити схему захисту від перерозряду, перезаряду та короткого замикання. Це підвищує безпечність роботи з Li-ion елементом, хоча повністю не замінює дотримання правил експлуатації акумулятора [16].

У структурі розроблюваної системи TP4056 не виконує функції аналізу стану акумулятора. Його основне призначення полягає в забезпеченні заряджання елемента перед проведенням вимірювань або між циклами тестування. Дані про стан акумулятора формуються не зарядним модулем, а вимірювальними компонентами та програмними алгоритмами Raspberry Pi.

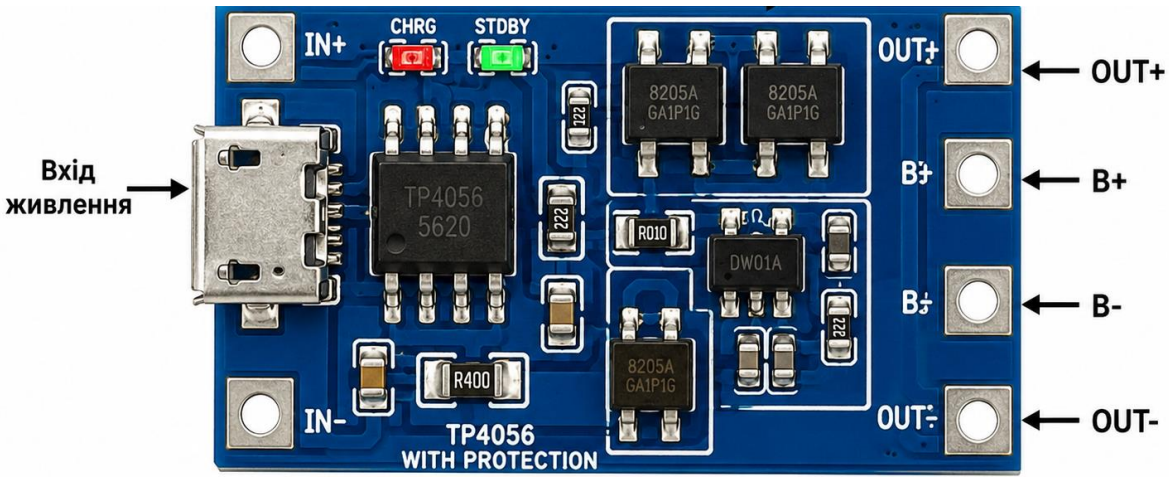

Рисунок 2.7 – Модуль заряджання Li-ion акумулятора TP4056

Під час використання TP4056 потрібно враховувати, що Raspberry Pi не слід живити безпосередньо від одного елемента 18650 через цей модуль без стабілізуючого перетворювача [16].

Напруга одного Li-ion елемента змінюється в широкому діапазоні та не відповідає стабільним вимогам живлення Raspberry Pi. Тому акумулятор у даній системі розглядається як об’єкт моніторингу, а не як основне джерело живлення одноплатного комп’ютера.

2.2.5 Вибір навантаження для контрольованого розряду

Для оцінювання фактичної ємності Li-ion акумулятора 18650 необхідно організувати контрольований процес розряду. Під час такого розряду система фіксує зміну напруги, струму та температури акумулятора в часі. На основі цих даних можна визначити кількість заряду, яку акумулятор здатний віддати до досягнення нижньої межі напруги, а також оцінити характер його роботи під навантаженням [17].

У розроблюваній системі в якості навантаження обрано потужний резистор номіналом 7,5 Ом з допустимою потужністю розсіювання 10 Вт. Такий вибір є достатнім для безпечного розряду одного Li-ion елемента 18650 без створення надмірного струму. Якщо акумулятор повністю заряджений і має напругу близько 4,2 В, то початковий струм через резистор становитиме приблизно: 0,56 А [17].

У процесі розряду напруга акумулятора поступово зменшується, тому струм через резистор також буде знижуватися. Наприклад, при напрузі 3,7 В струм становитиме близько 0,49 А, а при напрузі 3,0 В – близько 0,4 А. Такий режим є придатним для тестування, оскільки він не створює різкого навантаження на акумулятор і водночас дозволяє отримати достатньо помітну розрядну характеристику [17].

Потужність, яка виділяється на навантажувальному резисторі, визначається за формулою:

$$P = U^2 / R, \quad (2.1)$$

де U – напруга живлення акумулятора;

R – опір навантажувального резистора.

Для максимальної напруги акумулятора 4,2 В потужність становитиме приблизно 2,35 Вт.

Отже, використання резистора потужністю 10 Вт забезпечує необхідний запас за тепловим навантаженням. Це важливо, оскільки під час тривалого розряду резистор нагрівається, і застосування елемента з малим запасом потужності може призвести до його перегріву або зміни параметрів [18]. Резистор бажано розміщувати так, щоб навколо нього був вільний доступ повітря, а його корпус не контактував із акумулятором або пластиковими елементами макета.

Навантажувальний резистор вмикається в коло розряду послідовно з акумулятором через вимірювальний модуль INA219. У такій схемі позитивний вивід акумулятора підключається до входу VIN+ модуля INA219, з виходу VIN- струм проходить до навантажувального резистора, а другий вивід резистора з’єднується з негативним виводом акумулятора. Завдяки такому включенню весь струм розряду проходить через шунт модуля INA219, що дозволяє вимірювати його значення та одночасно контролювати напругу акумулятора [18].

Для зручності проведення експериментів у коло розряду доцільно додати вимикач або електронний ключ. Це дозволить запускати та зупиняти розряд без від’єднання провідників. У найпростішому варіанті можна використати механічний перемикач, розрахований на струм не менше 1 А. У складнішому варіанті розряд може вмикатися за допомогою MOSFET-ключа, яким керуватиме Raspberry Pi []18.

Резистивне навантаження не забезпечує абсолютно сталого струму, оскільки струм залежить від поточної напруги акумулятора. Проте це не є критичним для даної системи, оскільки модуль INA219 вимірює фактичний струм у кожен момент часу. Отже, програмне забезпечення може розраховувати

переданий заряд не за припущенням про сталий струм, а на основі реальних вимірювань струму та часу. Це підвищує коректність оцінювання фактичної ємності акумулятора.

Під час контрольованого розряду необхідно обмежити нижню межу напруги акумулятора. Для Li-ion елемента 18650 доцільно припиняти тестування при досягненні напруги близько 3,0 В. Подальший розряд може бути небажаним, оскільки він наближає акумулятор до режиму глибокого розряду. Тому програмне забезпечення повинно постійно контролювати напругу та формувати попередження або команду на зупинку розряду при досягненні заданої межі.

2.2.6 Засоби живлення та допоміжні елементи

Для стабільної роботи Raspberry Pi необхідне окреме джерело живлення напругою 5 В з достатнім струмом. Використання окремого блока живлення є доцільним, оскільки акумулятор 18650 у даній системі виконує роль об’єкта дослідження, а не джерела живлення всієї системи. Це дозволяє уникнути ситуації, коли зміна стану досліджуваного акумулятора впливає на стабільність роботи обчислювального модуля [19].

До допоміжних елементів системи належать тримач акумулятора 18650, з’єднувальні провідники, макетна плата, резистори, клемники та елементи механічного кріплення. Незважаючи на допоміжний характер, ці компоненти впливають на надійність роботи прототипу. Якісний контакт, правильна полярність підключення та надійна фіксація датчика температури є важливими умовами отримання коректних вимірювань.

Для зручності роботи передбачено окремі клеми для підключення акумулятора, навантаження та вимірювального модуля. Це спрощує перевірку схеми, заміну елементів і проведення повторних вимірювань. Крім того, бажано використовувати провідники достатнього перерізу для ділянок, через які протікає струм розряду. Узагальнений перелік апаратних компонентів системи наведено в табл. 2.1.

Таблиця 2.1 – Апаратні компоненти комп’ютерної системи

Компонент	Основне призначення	Обґрунтування
Li-ion 18650	Об’єкт моніторингу	Поширений формат акумулятора, придатний для лабораторного тестування
Raspberry Pi 5	Центральний обчислювальний модуль	Підтримує зчитування даних, збереження результатів, побудову графіків і запуск програмних модулів прогнозування
INA219	Вимірювання напруги, струму та потужності	Працює через I2C і не потребує аналогових входів Raspberry Pi
DS18B20	Контроль температури акумулятора	Передає цифрові значення температури та просто підключається
TP4056	Заряджання одного Li-ion елемента	Дозволяє організувати заряджання акумулятора перед тестуванням
Резистор 7,5 Ом / 10 Вт	Проведення контрольованого розряду	Забезпечує помірний струм розряду та має запас за потужністю
Блок живлення 5 В	Живлення Raspberry Pi	Забезпечує стабільну роботу обчислювального модуля
Тримач 18650 та монтажні елементи	Підключення й фіксація компонентів	Спрощують складання прототипу системи

Таким чином, для побудови апаратної частини системи обрано компоненти, які забезпечують вимірювання електричних і температурних параметрів акумулятора, стабільну роботу обчислювального модуля та можливість проведення контрольованих циклів розряду. Вибраний набір апаратних засобів є достатнім для реалізації програмно-апаратного прототипу, у якому Raspberry Pi виконує функції збору, обробки, збереження та аналізу даних, а акумулятор 18650 виступає об’єктом моніторингу і діагностики.

2.3 Розробка структурної схеми апаратної частини системи

Після вибору основних апаратних компонентів необхідно визначити спосіб їх взаємного підключення та функціональну роль кожного вузла у складі комп’ютерної системи моніторингу стану акумулятора [19]. Структурна схема апаратної частини повинна відображати фізичну взаємодію між акумулятором, вимірювальними модулями, навантаженням, обчислювальним модулем і засобами живлення.

Загальна структурна схема апаратної частини комп'ютерної системи моніторингу стану акумулятора наведена на рис. 2.8.

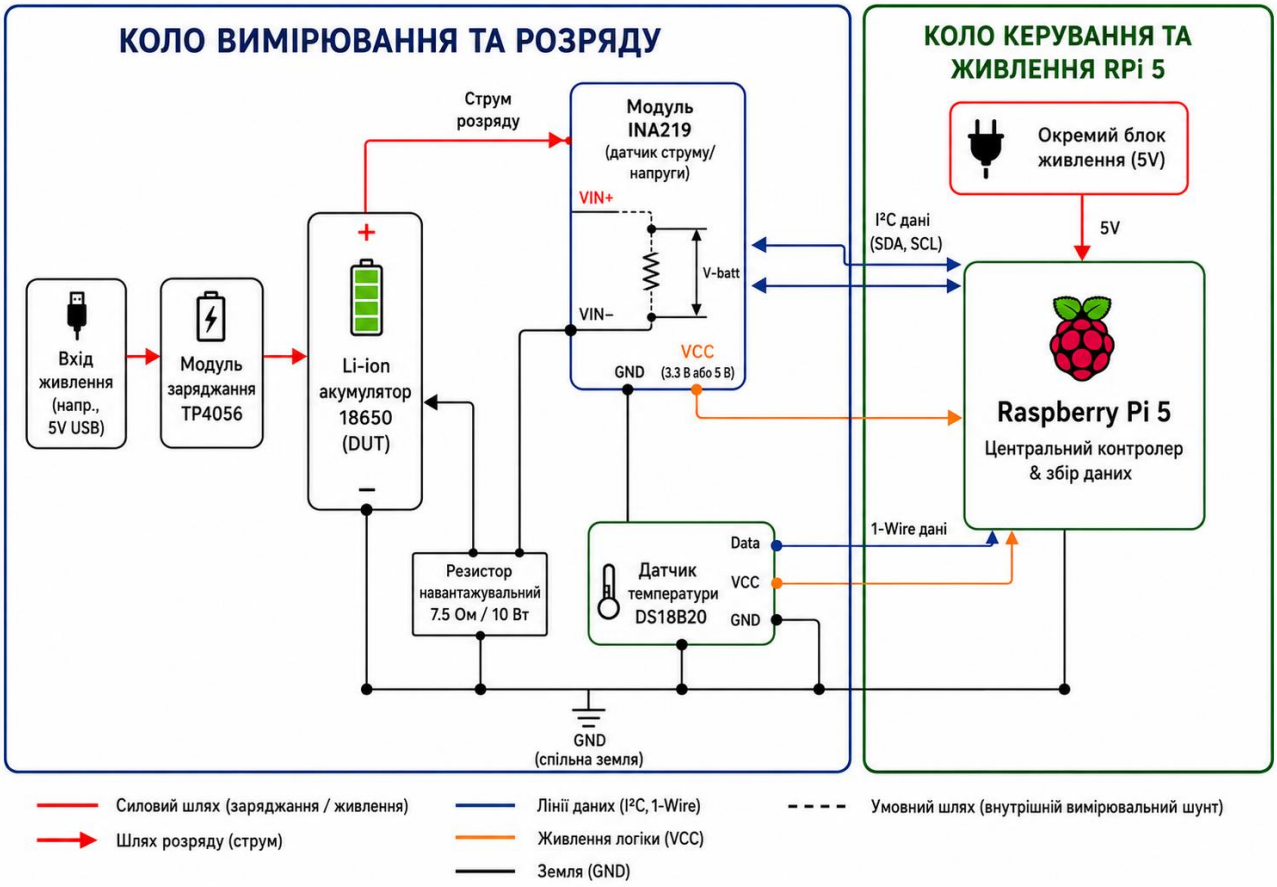

Рисунок 2.8 – Структурна схема апаратної частини системи моніторингу стану акумулятора

Основне розрядне коло формується з акумулятора 18650, модуля INA219 і навантажувального резистора. Позитивний вивід акумулятора підключається до входу VIN+ модуля INA219. Вивід VIN- модуля з'єднується з одним виводом навантажувального резистора 7,5 Ом / 10 Вт. Другий вивід резистора повертається до негативного контакту акумулятора. У такій конфігурації весь струм розряду проходить через шунтовий резистор модуля INA219, тому модуль може вимірювати фактичний струм у колі та напругу акумулятора під час тестування [19].

Модуль INA219 підключається до Raspberry Pi 5 через інтерфейс I^2C. Для цього використовуються контакти SDA та SCL модуля, які з'єднуються з

відповідними GPIO-контактами Raspberry Pi 5. Контакт VCC модуля підключається до лінії живлення 3,3 В або 5 В відповідно до особливостей конкретного модуля, а контакт GND – до загальної землі Raspberry Pi. Важливо, щоб вимірювальний модуль і Raspberry Pi мали спільний потенціал землі, інакше цифровий обмін даними може бути некоректним.

Сигнальний вивід температурного датчика DS18B20 підключається до одного з GPIO-контактів Raspberry Pi 5, а між сигнальною лінією та живленням встановлюється резистор змінного опору, який необхідний для коректної роботи інтерфейсу 1-Wire. У табл. 2.2 наведено особливості підключення основних вузлів апаратної частини системи.

Таблиця 2.2 – Підключення основних вузлів апаратної частини системи

Вузол системи	Основні підключення
Li-ion 18650	До модуля INA219, TP4056 або тримача акумулятора
INA219	VIN+ і VIN- у розрядне коло, SDA/SCL до Raspberry Pi 5
Резистор 7,5 Ом / 10 Вт	Послідовно в коло розряду
DS18B20	Живлення, GND, сигнальна лінія до GPIO
Raspberry Pi 5	I^2C, GPIO, живлення 5 В
TP4056	B+/B- до акумулятора, вхід живлення до USB або 5 В
Блок живлення 5 В	До Raspberry Pi 5
Тримач 18650 і клемники	До акумулятора та розрядного кола

Запропонована структурна схема апаратної частини дозволяє реалізувати базовий цикл роботи системи. Спочатку акумулятор заряджається за допомогою модуля TP4056. Після цього він підключається до розрядного кола, де через модуль INA219 і резистор 7,5 Ом / 10 Вт виконується контрольований розряд. Під час розряду Raspberry Pi 5 зчитує електричні параметри з INA219 і температуру з DS18B20, після чого отримані значення передаються до програмної частини для обробки та збереження.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ МОНІТОРИНГУ СТАНУ АКУМУЛЯТОРІВ

3.1 Побудова багаторівневої архітектури програмного забезпечення системи моніторингу стану акумуляторів

Програмне забезпечення комп'ютерної системи моніторингу стану акумулятора повинно забезпечувати не лише зчитування даних із вимірювальних модулів, а й повний цикл їх перетворення у корисну діагностичну інформацію [20]. У проєктованій системі первинні вимірювання напруги, струму та температури мають пройти кілька послідовних етапів: отримання з апаратних модулів, перевірку, нормалізацію, збереження, розрахунок показників стану, прогнозування деградації та відображення результатів користувачу. Тому програмну частину доцільно будувати не як один монолітний скрипт, а як багаторівневу архітектуру з чітким поділом функцій.

Багаторівнева архітектура дозволяє відокремити роботу з апаратними інтерфейсами від аналітичної обробки та веб-інтерфейсу. Це важливо для даної системи, оскільки різні частини програмного забезпечення мають різну природу. Наприклад, модуль зчитування даних повинен взаємодіяти з I^2C та GPIO, модуль збереження – працювати з файлами, аналітичний модуль – виконувати розрахунки, а веб-сервер – обробляти запити користувача [20].

У кваліфікаційній роботі програмне забезпечення пропонується реалізувати таким чином, щоб воно функціонувало на Raspberry Pi 5. На цьому пристрої розміщуються всі основні програмні компоненти: модулі взаємодії з датчиками, обробки даних, збереження результатів, оцінювання стану акумулятора, прогнозування деградації та веб-інтерфейс користувача.

Raspberry Pi 5 у такій архітектурі виконує роль локального обчислювального вузла та веб-сервера [21]. Користувач взаємодіє із системою через браузер, відкриваючи веб-сторінку моніторингу за локальною адресою пристрою.

Запропоновану архітектуру програмного забезпечення, яка представлена на рис. 3.1, поділено на такі рівні:

– рівень апаратної взаємодії;
– рівень збору та нормалізації даних;
– рівень збереження та історії вимірювань;
– аналітичний рівень;
– прогностичний рівень;
– серверний рівень;
– рівень користувацького інтерфейсу.

Рисунок 3.1 – Багаторівнева архітектура програмного забезпечення моніторингу стану акумуляторів з прогнозуванням їх деградації

Першим є рівень апаратної взаємодії. Він відповідає за безпосередній обмін даними з вимірювальними компонентами системи. На цьому рівні працюють драйвери та прикладні модулі, які отримують значення з INA219 та DS18B20. Модуль INA219 передає до Raspberry Pi 5 дані про напругу та струм через інтерфейс I^2C, а датчик DS18B20 передає значення температури через цифрову лінію 1-Wire [22]. Завдання цього рівня полягає не в аналізі стану акумулятора, а в надійному отриманні первинних значень із апаратних засобів.

Другим є рівень збору та нормалізації даних. Він приймає первинні значення з апаратного рівня та перетворює їх у єдиний формат запису. На цьому етапі до вимірювань додається часова мітка, виконується приведення одиниць вимірювання, перевіряється наявність пропущених або некоректних значень. Наприклад, напруга може зберігатися у вольтах, струм – в амперах, температура – у градусах Цельсія, а потужність може розраховуватися як добуток напруги на струм [23]. Такий рівень є проміжним між фізичними вимірюваннями та подальшою аналітикою.

Третім є рівень збереження та історії вимірювань. Його призначення полягає у формуванні накопиченого набору даних про роботу акумулятора. У межах лабораторного прототипу доцільно використовувати CSV-файл, оскільки він є простим, відкритим і зручним для подальшої обробки у Python. На цьому рівні зберігаються часові ряди напруги, струму, температури, потужності, номера циклу та службових параметрів. Окремо може формуватися підсумковий журнал циклів, у якому для кожного циклу фіксуються тривалість розряду, розрахована ємність, середня температура та оцінений SoH [18].

Четвертим є аналітичний рівень. Він відповідає за перетворення накопичених вимірювань у показники стану акумулятора. На цьому рівні розраховується кількість переданого заряду, фактична ємність, орієнтовний рівень заряду та показник SoH. Крім того, аналітичний рівень контролює граничні значення напруги та температури. Якщо напруга акумулятора наближається до нижньої допустимої межі або температура перевищує заданий поріг, система повинна сформувати відповідний стан попередження. Таким

чином, аналітичний рівень виконує роль локального діагностичного ядра системи.

П’ятим є прогностичний рівень. Він використовує результати аналітичного рівня та історію попередніх циклів для прогнозування деградації акумулятора. Основними вхідними ознаками для цього рівня є номер циклу, фактична ємність, SoH, середній струм розряду, середня температура та тривалість циклу [23]. У межах кваліфікаційної роботи використовується модель прогнозування дерево прийняття рішень. Завдання цього рівня полягає у визначенні тенденції зменшення SoH і наближенні акумулятора до критичного рівня зношення.

Шостим є серверний рівень. Він забезпечує зв’язок між внутрішніми програмними модулями та веб-інтерфейсом користувача. Для реалізації цього рівня використовується локальний веб-сервер на основі Flask. Серверний рівень надає веб - сторінку користувачу, обробляє HTTP-запити, передає поточні значення параметрів, результати розрахунків і дані для графіків. Крім того, через серверний рівень можна реалізувати окремі API-маршрути, наприклад для отримання останнього вимірювання, історії розряду, результатів прогнозування або стану попереджень [24].

Сьомим є рівень користувацького інтерфейсу. Він реалізується у вигляді веб-сторінки, яка відкривається у браузері. На цьому рівні користувач бачить поточну напругу, струм, температуру, розраховану потужність, орієнтовну ємність, SoH і прогнозований залишковий ресурс. Також веб-інтерфейс повинен відображати графіки зміни параметрів у часі. Якщо система виявляє небажаний режим роботи, наприклад зниження напруги нижче допустимого рівня або підвищення температури, на сторінці повинно з’являтися попередження.

Перевагою такої архітектури є чітке розмежування відповідальності між програмними частинами системи. Якщо потрібно змінити спосіб збереження даних, наприклад перейти від CSV-файлу до реляційної бази даних, це можна зробити на рівні збереження без зміни логіки апаратного зчитування. Якщо в майбутньому буде використана інша модель прогнозування, зміни стосуватимуться переважно прогностичного рівня. Якщо потрібно змінити

вигляд веб-сторінки, це не потребує внесення змін до модулів вимірювання. Такий підхід робить систему більш гнучкою та придатною до розширення.

Для реалізації програмного забезпечення пропонується використати мову програмування Python, оскільки вона підтримує роботу з апаратними інтерфейсами Raspberry Pi, файлами даних, веб-фреймворками та бібліотеками аналізу даних.

Таким чином, запропонована архітектура програмного забезпечення комп'ютерної системи моніторингу стану акумуляторів з прогнозуванням їх деградації створює основу для подальшої реалізації модулів зчитування, збереження, оцінювання стану та прогнозування деградації акумулятора.

3.2 Реалізація рівня апаратної взаємодії та збору даних

Перед початком роботи програмного забезпечення на Raspberry Pi 5 необхідно активувати відповідні апаратні інтерфейси. Для роботи з INA219 потрібно увімкнути інтерфейс I2C, а для DS18B20 – підтримку 1-Wire. Після цього програма може звертатися до підключених модулів через відповідні бібліотеки або системні файли. На цьому етапі важливо перевірити, чи доступні пристрої, оскільки відсутність з'єднання з одним із датчиків може призвести до некоректної роботи всієї системи.

Загальна логіка рівня апаратної взаємодії передбачає виконання таких дій: ініціалізація інтерфейсів, перевірка доступності вимірювальних модулів, зчитування напруги та струму з INA219, зчитування температури з DS18B20, формування первинного набору даних і передавання його на рівень нормалізації.

Для програмної реалізації доцільно створити окремий модуль sensors.py, у якому будуть зосереджені функції роботи з вимірювальними компонентами. Це дозволяє не змішувати код зчитування датчиків із кодом обробки, збереження або веб-інтерфейсу. На рис. 3.2 наведено фрагмент програмного коду для ініціалізації модуля INA219.

```
import time
from datetime import datetime

try:
    from ina219 import INA219
except ImportError:
    INA219 = None


SHUNT_OHMS = 0.1
MAX_EXPECTED_AMPS = 1.0
```

Рисунок 3.2 – Програмна ініціалізація модуля INA219

Для зчитування показників струму і напруги реалізовано клас з відповідним методом, які представлено на рис. 3.3.

```
class Ina219Reader:
    def __init__(self):
        if INA219 is None:
            raise RuntimeError("Бібліотеку INA219 не встановлено")

        self.sensor = INA219(
            shunt_ohms=SHUNT_OHMS,
            max_expected_amps=MAX_EXPECTED_AMPS
        )
        self.sensor.configure()

    def read_electrical_values(self):
        voltage = self.sensor.voltage()
        current = self.sensor.current() / 1000.0
        power = self.sensor.power() / 1000.0

        return {
            "voltage_v": round(voltage, 3),
            "current_a": round(current, 4),
            "power_w": round(power, 4)
        }
```

Рисунок 3.3 – Зчитування значень з модуля INA219

У наведеному фрагменті клас Ina219Reader відповідає за роботу з модулем INA219. Під час створення об’єкта виконується ініціалізація датчика та його конфігурація. Метод read_electrical_values() повертає напругу у вольтах, струм в амперах і потужність у ватах. Оскільки деякі бібліотеки для INA219 повертають струм у міліамперах, у коді виконується перетворення значення у ампери. Таке приведення одиниць важливе для подальших розрахунків ємності та потужності.

Для зчитування температури з DS18B20 можна використати файловий інтерфейс Linux, через який Raspberry Pi отримує значення з датчика [24]. Після активації 1-Wire у системі з’являється каталог із даними температурного сенсора. Програма може зчитувати відповідний файл і перетворювати отримане значення у градуси Цельсія. На рис. 3.4 представлено оголошення класу Ds18b20Reader та метод, який реалізує пошук датчика температури.

```python
from pathlib import Path


class Ds18b20Reader:
    def __init__(self, base_path="/sys/bus/w1/devices"):
        self.base_path = Path(base_path)
        self.device_file = self._find_sensor_file()

    def _find_sensor_file(self):
        sensors = list(self.base_path.glob("28-*"))

        if not sensors:
            raise RuntimeError("Датчик DS18B20 не знайдено")

        return sensors[0] / "w1_slave"
```

Рисунок 3.4 – Оголошення класу для роботи з датчиком температури

Для зчитування значень температури у класі Ds18b20Reader реалізовано метод read_temperature(self). Програмний код цього методу проілюстровано на рис. 3.5.

Клас Ds18b20Reader виконує пошук підключеного температурного датчика та зчитує значення температури з системного файлу. Перевірка наявності рядка YES дає змогу переконатися, що дані були прочитані коректно. Якщо датчик не знайдено або формат даних є неправильним, програма формує

виняток. Такий підхід дозволяє виявляти помилки апаратного підключення ще на рівні зчитування даних.

```
def read_temperature(self):
    content = self.device_file.read_text().splitlines()

    if len(content) < 2 or "YES" not in content[0]:
        raise RuntimeError("Помилка зчитування температури")

    marker = "t="
    position = content[1].find(marker)

    if position == -1:
        raise RuntimeError("Некоректний формат даних DS18B20")

    temperature_raw = content[1][position + len(marker):]
    temperature_c = float(temperature_raw) / 1000.0

    return round(temperature_c, 2)
```

Рисунок 3.5 – Метод зчитування значень температури

Після отримання значень з INA219 і DS18B20 необхідно сформувати єдиний первинний запис вимірювання. Такий запис повинен містити часову мітку, напругу, струм, потужність і температуру [25]. Саме він передається на наступний рівень програмної архітектури – рівень збору та нормалізації даних. На рис. 3.6 показано реалізацію класу MeasurementCollector.

```
class MeasurementCollector:
    def __init__(self):
        self.ina219 = Ina219Reader()
        self.temperature_sensor = Ds18b20Reader()

    def collect(self):
        electrical_data = self.ina219.read_electrical_values()
        temperature_c = self.temperature_sensor.read_temperature()

        measurement = {
            "timestamp": datetime.now().isoformat(timespec="seconds"),
            "voltage_v": electrical_data["voltage_v"],
            "current_a": electrical_data["current_a"],
            "power_w": electrical_data["power_w"],
            "temperature_c": temperature_c
        }

        return measurement
```

Рисунок 3.6 – Клас MeasurementCollector

Клас MeasurementCollector поєднує дані з двох джерел: модуля INA219 і датчика DS18B20. Метод collect() створює один структурований запис, який містить усі первинні параметри акумулятора на певний момент часу. Така структура є зручною, оскільки надалі її можна безпосередньо передавати до модулів перевірки, збереження або відображення [25].

У реальному режимі роботи система повинна виконувати зчитування не один раз, а циклічно. Інтервал опитування можна вибирати з врахуванням потреб. У системі контролю розряду акумулятора достатньо виконувати вимірювання один раз на декілька секунд. Занадто малий інтервал опитування призведе до надмірного обсягу даних, а занадто великий може не дозволити вчасно виявити різке падіння напруги чи зростання температури. На рис. 3.7 показано функцію, що забезпечує циклічний запуск на зчитування показників стану акумулятора.

```
def run_measurement_loop(interval_seconds=1):
    collector = MeasurementCollector()

    while True:
        try:
            measurement = collector.collect()
            print(measurement)
            time.sleep(interval_seconds)

        except RuntimeError as error:
            print(f"Помилка зчитування даних: {error}")
            time.sleep(interval_seconds)
```

Рисунок 3.7 – Функція циклічного запуску на зчитування показників стану акумулятора

Наведений на рис. 3.7 фрагмент коду демонструє базову логіку циклічного збору даних. У кожній ітерації програма зчитує параметри акумулятора, формує запис вимірювання та виводить його у консоль. У повній версії системи замість простого виведення на екран цей запис передається до модуля нормалізації та збереження. Обробка винятків дає змогу уникнути аварійного завершення програми у випадку короткочасної помилки зчитування [25].

Важливою вимогою до рівня апаратної взаємодії є стійкість до помилок. Під час роботи системи моніторингу стану акумуляторів можуть виникати ситуації, коли один із датчиків тимчасово недоступний, має поганий контакт або повертає некоректне значення. Тому програмний код повинен передбачати перевірку доступності датчиків, обробку винятків і передачу інформації про помилки на вищі рівні системи. Наприклад, якщо температура не зчитується, система може тимчасово позначити це значення як відсутнє, але водночас сформувати повідомлення для користувача [23].

Результатом роботи рівня апаратної взаємодії є первинний набір даних, отриманий безпосередньо з апаратних компонентів. Цей набір ще не є повністю готовим для прогнозування деградації, оскільки він потребує перевірки, нормалізації та збереження. Проте саме на цьому рівні формується основа для всіх подальших розрахунків. Якщо дані з INA219 або DS18B20 будуть неточними або неповними, то аналітичний і прогностичний рівні не зможуть сформувати коректні результати.

Реалізація рівня рівень апаратної взаємодії та збору даних у вигляді окремих класів для INA219, DS18B20 і колектора вимірювань робить програмне забезпечення більш зрозумілим, модульним і придатним до подальшого розширення.

3.3 Реалізація рівня нормалізації, збереження даних та журналювання циклів заряду-розряду акумуляторів

Після отримання первинних значень із вимірювальних модулів необхідно забезпечити їх підготовку до подальшого використання. Дані, які надходять із рівня апаратної взаємодії, ще не можна безпосередньо застосовувати для аналізу деградації акумулятора, оскільки вони можуть містити випадкові відхилення, пропущені значення або мати різний формат представлення [24]. Тому наступним етапом роботи програмного забезпечення є нормалізація, перевірка та збереження вимірювальних даних.

Для реалізації перевірки та нормалізації створено окрему функцію, яка приймає початковий запис вимірювання і повертає підготовлений запис у єдиному форматі. На рис. 3.8 наведено програмний код функції нормалізації виміряних значень стану акумулятора.

```
def normalize_measurement(raw_measurement, cycle_id):
    voltage = float(raw_measurement["voltage_v"])
    current = float(raw_measurement["current_a"])
    power = float(raw_measurement["power_w"])
    temperature = float(raw_measurement["temperature_c"])

    if voltage < 0 or voltage > 5.0:
        raise ValueError("Некоректне значення напруги акумулятора")

    if abs(current) > 2.0:
        raise ValueError("Некоректне значення струму")

    if temperature < -20 or temperature > 80:
        raise ValueError("Некоректне значення температури")

    normalized = {
        "timestamp": raw_measurement["timestamp"],
        "cycle_id": cycle_id,
        "voltage_v": round(voltage, 3),
        "current_a": round(current, 4),
        "power_w": round(power, 4),
        "temperature_c": round(temperature, 2),
        "status": "ok"
    }

    return normalized
```

Рисунок 3.8 – Функція нормалізації значень показників струму, напруги і температури

У наведеному на рис. 3.8 фрагменті програмного коду, функція normalize_measurement() виконує перетворення первинного запису до уніфікованого формату. Додатково вона перевіряє, чи не виходять параметри за допустимі межі. Верхня межа напруги 5 В вибрана із запасом, оскільки напруга одного Li-ion елемента у нормальному режимі не повинна перевищувати 4,2 В. Перевірка струму та температури дозволяє відсіяти очевидно помилкові

значення, які можуть виникнути внаслідок збою датчика або некоректного зчитування [25].

Для збереження вимірювань використовується CSV-файл, структуру якого показано у табл. 3.1.

Таблиця 3.1 – Структура файлу measurements.csv

Назва поля	Призначення	Приклад значення
timestamp	Дата і час вимірювання	2026-06-07T12:30:15
cycle_id	Номер циклу заряду-розряду	4
voltage_v	Напруга акумулятора, В	3.742
current_a	Струм розряду, А	0.4812
power_w	Потужність, Вт	1.800
temperature_c	Температура акумулятора, °C	31.6
status	Стан запису	ok

Для запису даних у CSV-файл доцільно створити окремий клас. Це дозволяє відокремити логіку збереження від логіки зчитування датчиків і подальшої аналітики. На рис. 3.9 наведено оголошення класу для запису даних у CSV-файл.

```
class CsvMeasurementStorage:
    def __init__(self, file_path="measurements.csv"):
        self.file_path = Path(file_path)
        self.fieldnames = [
            "timestamp",
            "cycle_id",
            "voltage_v",
            "current_a",
            "power_w",
            "temperature_c",
            "status"
        ]
        self._create_file_if_needed()
```

Рисунок 3.9 – Оголошення класу CsvMeasurementStorage

У класі CsvMeasurementStorage реалізовано методи для створення файлу та запису даних. На рис. 3.10 представлено метод _create_file_if_needed.

```python
def _create_file_if_needed(self):
    if not self.file_path.exists():
        with self.file_path.open("w", newline="", encoding="utf-8") as file:
            writer = csv.DictWriter(file, fieldnames=self.fieldnames)
            writer.writeheader()
```

Рисунок 3.10 – Метод створення CSV-файлу (за необхідності)

Програмна реалізація методу запису даних про параметри стану акумулятора проілюстрована на рис. 3.11.

```python
def save(self, measurement):
    with self.file_path.open("a", newline="", encoding="utf-8") as file:
        writer = csv.DictWriter(file, fieldnames=self.fieldnames)
        writer.writerow(measurement)
```

Рисунок 3.11 – Метод запису даних

Клас CsvMeasurementStorage відповідає за створення CSV-файлу та додавання нових рядків. Під час ініціалізації перевіряється наявність файлу. Якщо файл ще не створений, програма формує його заголовок. Метод save() додає новий нормалізований запис у кінець файлу.

Окрім основного файлу вимірювань, передбачено окремий журнал циклів. Якщо measurements.csv зберігає кожне окреме вимірювання, то журнал циклів містить узагальнені результати для кожного циклу розряду. Такий журнал зберігається у файлі cycles.csv. Структура файлу подана у табл. 3.2.

Таблиця 3.2 – Структура файлу cycles.csv

Поле	Зміст поля
cycle_id	Номер циклу
start_time	Початок циклу
end_time	Завершення циклу
duration_s	Тривалість циклу, с
capacity_mah	Розрахована ємність, мА·год
avg_current_a	Середній струм, А
avg_temperature_c	Середня температура, °C
soh_percent	Оцінений SoH, %

Програмна логіка реалізації класу для роботи з журналювання циклів заряду-розряду акумуляторів подібна до тієї, яка стосується запису даних з датчика температури та модуля вимірювання струму і напруги.

Журнал циклів має важливе значення для прогностичного рівня. Якщо основний файл вимірювань містить велику кількість точок, то журнал циклів подає дані у більш компактній формі. Саме такі узагальнені дані зручно використовувати як вхідні ознаки для моделі прогнозування деградації. Тому алгоритм дерева прийняття рішень може працювати не з кожним секундним вимірюванням, а з підсумковими параметрами окремих циклів.

Для поєднання зчитування, нормалізації та збереження використовується окремий код управління, який показаний на рис. 3.12.

```
def collect_and_store_measurement(collector, measurement_storage, cycle_id):
    raw_measurement = collector.collect()

    try:
        normalized = normalize_measurement(raw_measurement, cycle_id)
        measurement_storage.save(normalized)

        return normalized

    except ValueError as error:
        error_record = {
            "timestamp": raw_measurement.get("timestamp"),
            "cycle_id": cycle_id,
            "voltage_v": raw_measurement.get("voltage_v"),
            "current_a": raw_measurement.get("current_a"),
            "power_w": raw_measurement.get("power_w"),
            "temperature_c": raw_measurement.get("temperature_c"),
            "status": f"error: {error}"
        }

        measurement_storage.save(error_record)

        return error_record
```

Рисунок 3.12 – Функція def collect_and_store_measurement()

Функція collect_and_store_measurement() отримує первинний запис, виконує нормалізацію та зберігає результат. Якщо значення не проходить перевірку, у файл записується службовий запис зі статусом помилки. Це

дозволяє не втрачати інформацію про проблемні вимірювання і водночас не передавати їх без перевірки до аналітичного рівня.

Збереження помилкових записів зі службовим статусом є корисним для налагодження системи. Наприклад, якщо під час експерименту періодично виникають помилки зчитування температури або струму, це можна буде побачити у файлі вимірювань. Надалі такі записи були виключені з розрахунків або використані для оцінювання стабільності апаратного підключення.

Важливою вимогою до рівня збереження є неперервність роботи. Під час тривалого розряду акумулятора система повинна регулярно записувати дані, не перериваючи процес моніторингу. Тому операції запису повинні бути простими й швидкими. CSV-файл добре відповідає цій вимозі, оскільки додавання одного рядка не потребує складних операцій і може виконуватися без значного навантаження на Raspberry Pi 5.

3.4 Аналітичний рівень оцінювання стану акумулятора

У розроблюваній системі аналітичний рівень працює з даними, які зберігаються у файлі поточних вимірювань measurements.csv та журналі циклів cycles.csv. Основним джерелом для розрахунків є часовий ряд значень напруги, струму, потужності та температури [22]. Кожен запис має часову мітку, тому програмне забезпечення може визначати тривалість роботи акумулятора під навантаженням і розраховувати кількість заряду, переданого під час розряду [].

Одним із ключових розрахунків на аналітичному рівні є визначення кількості заряду, який акумулятор віддав під час розряду. Оскільки система фіксує струм у певні моменти часу, переданий заряд можна оцінити шляхом підсумовування добутків струму на часові інтервали між вимірюваннями. Якщо струм вимірюється в амперах, а час – у секундах, то результат можна перевести у мА·год. При практичній реалізації враховуються тільки ті записи, які належать до одного розрядного циклу та мають статус ok. Це дозволяє не включати до розрахунків помилкові або службові записи [21].

Загальну логіку роботи аналітичного рівня наведено на рис. 3.13.

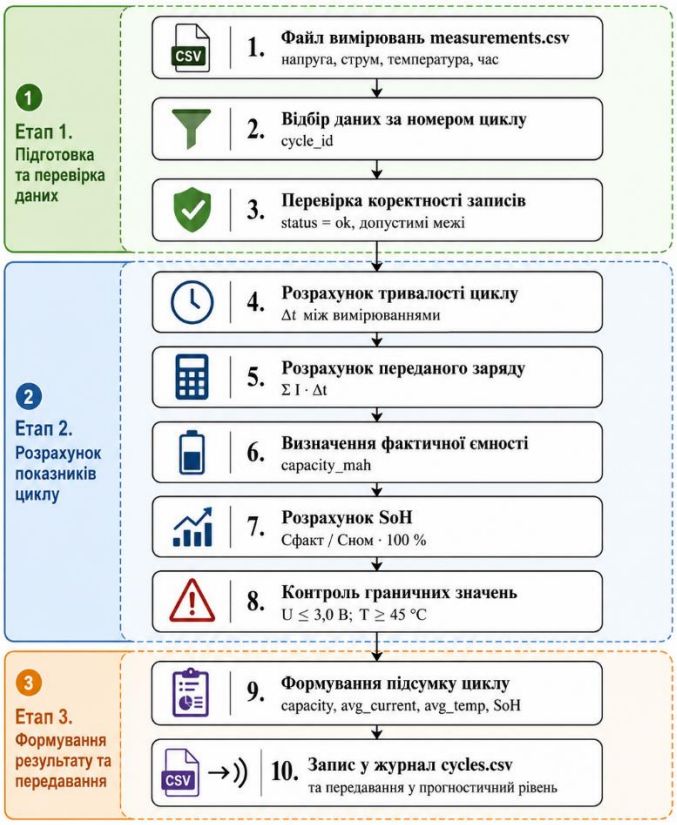

Рисунок 3.13 – Загальна логіка роботи аналітичного рівня

Фрагмент програмної реалізації розрахунку ємності на основі списку вимірювань показано на рис. 3.14.

```python
def calculate_capacity_mah(measurements):
    if len(measurements) < 2:
        return 0.0

    capacity_as = 0.0

    for index in range(1, len(measurements)):
        previous = measurements[index - 1]
        current = measurements[index]

        time_previous = parse_time(previous["timestamp"])
        time_current = parse_time(current["timestamp"])

        delta_t = (time_current - time_previous).total_seconds()

        if delta_t <= 0:
            continue

        current_a = abs(float(previous["current_a"]))
        capacity_as += current_a * delta_t

    capacity_mah = capacity_as * 1000.0 / 3600.0

    return round(capacity_mah, 2)
```

Рисунок 3.14 – Функція розрахунку ємності акумулятора

Як видно з рис. 3.15, функція calculate_capacity_mah() приймає список виміряних показників, які належать до одного циклу розряду. Для кожної пари сусідніх вимірювань визначається часовий інтервал, після чого обчислюється внесок струму в загальну кількість переданого заряду. Результат повертається у мА·год.

Після визначення фактичної ємності можна розрахувати показник State of Health. У межах даної системи SoH визначається як відношення розрахованої фактичної ємності до номінальної ємності акумулятора [23]. Для прикладу, номінальна ємність Li-ion елементу становить 3000 мА·год, то саме це значення застосовується як базове для порівняння. Функція для розрахунку SoH проілюстрована на рис. 3.15.

```
def calculate_soh_percent(
    capacity_mah,
    nominal_capacity_mah=NOMINAL_CAPACITY_MAH
):
    if nominal_capacity_mah <= 0:
        raise ValueError("Номінальна ємність повинна бути більшою за нуль")

    soh = capacity_mah / nominal_capacity_mah * 100.0

    return round(min(soh, 100.0), 2)
```

Рисунок 3.15 – Функція розрахунку SoH

Функція calculate_soh_percent() повертає оцінений показник SoH у відсотках. Додатково використовується обмеження верхнього значення на рівні 100 %, оскільки в межах діагностичної інтерпретації акумулятор не повинен мати стан, кращий за номінальний. Програмний код усіх інших методів і класів, які використовуються на аналітичному рівні наведено у додатку Б.

Важливо, що аналітичний рівень не виконує прогнозування деградації безпосередньо. Його завданням є підготовка якісних розрахункових ознак для наступного рівня [25]. Саме прогностичний рівень використовує журнал циклів і побудовані аналітичні показники для оцінювання майбутнього стану акумулятора.

3.5 Прогнозування деградації акумулятора на основі методу дерева прийняття рішень

У розроблюваній системі для прогнозування деградації використовується модель на основі дерева прийняття рішень. Алгоритм дерева прийняття рішень працює за принципом послідовного поділу даних за певними умовами. На кожному вузлі дерева перевіряється значення однієї з ознак, наприклад фактичної ємності, кількості циклів, середньої температури або показника SoH. Залежно від результату перевірки запис переходить до наступного вузла, поки не буде отримано кінцевий результат. У даній системі таким результатом є прогнозований клас стану акумулятора.

У кваліфікаційній роботі для прогнозування рівня деградації акумулятора вирішується задача класифікації стану акумулятора. У цьому випадку модель дерева прийняття рішень визначає один із можливих станів: нормальний стан, помітна деградація або критичне зношення [24].

Як вхідні ознаки для рівня прогнозування використовуються дані з журналу циклів cycles.csv. Для навчання моделі кожному запису циклу необхідно поставити у відповідність цільовий клас стану. У межах комп'ютерної системи моніторингу стану та прогнозування деградації акумуляторів у відповідність цільовій змінній ставиться значення SoH. Наприклад, якщо SoH більший або дорівнює 85 %, акумулятор можна вважати таким, що перебуває у нормальному стані. Якщо SoH знаходиться в межах від 70 % до 85 %, це може відповідати помітній деградації. Якщо SoH менший ніж 70 %, акумулятор доцільно віднести до класу критичного зношення. Загальну схему роботи дерева прийняття рішень для прогнозування деградації акумулятора наведено на рис. 3.16. Така схема є прикладом інтерпретації дерева прийняття рішень. У програмній реалізації конкретна структура дерева формується автоматично під час навчання моделі на основі підготовлених даних. Проте наведена логіка добре демонструє, як модель може поєднувати кілька параметрів для формування висновку про стан акумулятора.

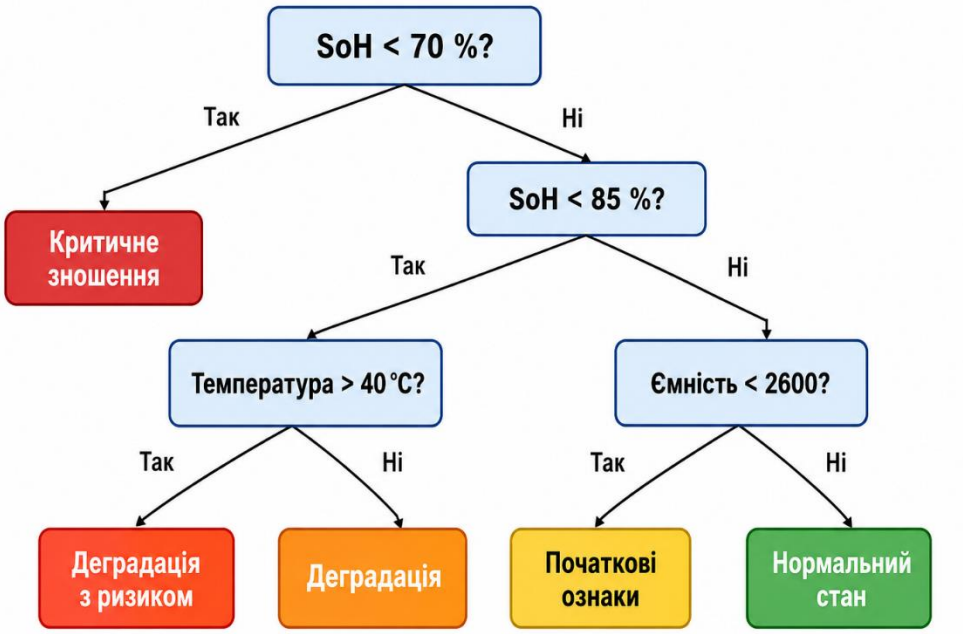

Рисунок 3.16 – Приклад дерева прийняття рішень

Для реалізації рівня прогнозування деградації акумулятора створено окремий програмний модуль degradation_model.py. Фрагмент коду для формування класу стану акумулятора на основі значення SoH представлено на рис. 3.17.

```
def define_degradation_class(soh_percent):
    if soh_percent >= 85.0:
        return "normal"

    if soh_percent >= 70.0:
        return "degraded"

    return "critical"
```

Рисунок 3.17 – Формування класів деградації акумулятора

Функція define_degradation_class() перетворює числове значення SoH у текстовий клас стану акумулятора. Такий підхід дозволяє використати задачу класифікації, де модель дерева прийняття рішень навчається визначати клас за набором вхідних ознак. У межах системи використано три класи: normal, degraded і critical.

Для підготовки навчального набору необхідно зчитати файл cycles.csv, вибрати потрібні ознаки та сформувати цільовий стовпець. Для цього можна використати бібліотеку pandas. Визначення ознак, за якими буде проводитися класифікація стану акумулятора показана на рис. 3.18.

```
import pandas as pd


FEATURE_COLUMNS = [
    "cycle_id",
    "capacity_mah",
    "avg_current_a",
    "avg_temperature_c",
    "duration_s",
    "soh_percent",
]
```

Рисунок 3.18 – Визначення вхідних ознак

Функція завантаження вхідних ознак для навчання моделі представлена на рис. 3.19.

```
def load_cycle_dataset(file_path="cycles.csv"):
    data = pd.read_csv(file_path)
    data = data.dropna(subset=FEATURE_COLUMNS)

    data["degradation_class"] = data["soh_percent"].apply(
        define_degradation_class
    )

    features = data[FEATURE_COLUMNS]
    target = data["degradation_class"]

    return features, target
```

Рисунок 3.19 – Функція load_cycle_dataset ()

На рис. 3.19 функція load_cycle_dataset() зчитує журнал циклів і формує набір ознак для навчання. До ознак включено номер циклу, фактичну ємність, середній струм, середню температуру, тривалість циклу та SoH. Цільова змінна degradation_class формується автоматично за значенням SoH.

Для побудови дерева прийняття рішень використовується клас DecisionTreeClassifier з бібліотеки scikit-learn [23]. Щоб оцінити працездатність моделі, набір даних поділяється на навчальну та тестову частини. На рис. 3.20 проілюстровано функцію навчання моделі дерева прийняття рішень.

```python
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier


def train_degradation_model(features, target):
    x_train, x_test, y_train, y_test = train_test_split(
        features,
        target,
        test_size=0.25,
        random_state=42,
        stratify=target,
    )

    model = DecisionTreeClassifier(
        max_depth=4,
        random_state=42,
    )

    model.fit(x_train, y_train)

    predictions = model.predict(x_test)
    accuracy = accuracy_score(y_test, predictions)

    return model, accuracy
```

Рисунок 3.20 – Функція навчання дерева прийняття рішень

У наведеному на рис. 3.20 програмному коді, функція train_degradation_model() виконує навчання дерева прийняття рішень. Параметр max_depth=4 обмежує глибину дерева, щоб модель не стала надто складною та не запам’ятовувала навчальні дані без узагальнення. Після навчання модель перевіряється на тестовій частині даних, а результат оцінюється за показником точності класифікації.

Після навчання модель використовується для прогнозування стану акумулятора за даними нового циклу. Для цього потрібно сформувати запис із

такими самими ознаками, які використовувалися під час навчання. Функція прогнозування стану деградації акумулятора показана на рис. 3.21.

```
def predict_degradation_state(model, cycle_summary):
    input_data = pd.DataFrame(
        [
            {
                "cycle_id": cycle_summary["cycle_id"],
                "capacity_mah": cycle_summary["capacity_mah"],
                "avg_current_a": cycle_summary["avg_current_a"],
                "avg_temperature_c": cycle_summary["avg_temperature_c"],
                "duration_s": cycle_summary["duration_s"],
                "soh_percent": cycle_summary["soh_percent"],
            }
        ]
    )

    predicted_class = model.predict(input_data)[0]
    probabilities = model.predict_proba(input_data)[0]

    class_probabilities = dict(zip(model.classes_, probabilities))

    return {
        "predicted_class": predicted_class,
        "probabilities": class_probabilities,
    }
```

Рисунок 3.21 – Функція прогнозування стану деградації акумулятора

Функція predict_degradation_state() приймає навчену модель і кінцевий запис циклу. На основі цих даних формується прогнозований клас стану акумулятора. Додатково повертаються ймовірності для кожного класу. Це дає змогу не лише показати кінцевий висновок, але й оцінити впевненість моделі у прогнозі [24].

Варто зазначити, що якість прогнозування залежить від кількості та різноманітності даних у журналі циклів. Якщо система має лише кілька записів, модель може працювати умовно й використовуватися переважно для демонстрації принципу прогнозування. Для отримання більш надійних результатів необхідно накопичити більшу кількість циклів або використати відкриті набори даних про деградацію Li-ion акумуляторів.

3.6 Серверний рівень та веб-інтерфейс користувача

Завершальним рівнем програмного забезпечення системи є серверний рівень і веб-інтерфейс користувача.Для реалізації серверного рівня використано фреймворк Flask. Він забезпечує обробку HTTP-запитів, формування веб-сторінки та передавання даних у форматі JSON. Такий підхід дозволяє відокремити внутрішню логіку системи від способу відображення результатів. Модулі зчитування, збереження, аналітики та прогнозування працюють на Raspberry Pi 5, а користувач отримує доступ до результатів через браузер.

Графічний інтерфейс реалізується у вигляді локальної веб-сторінки. Користувач може відкрити її з Raspberry Pi 5 або з іншого пристрою, підключеного до тієї самої мережі, за адресою локального веб-сервера. Це зручно, оскільки не потребує встановлення окремої настільної програми та дозволяє переглядати результати моніторингу з будь-якого пристрою з браузером [25].

На веб-сторінці передбачено кілька основних інформаційних зон. У верхній частині розміщується назва системи та поточний стан акумулятора. Нижче виводяться картки з основними параметрами: напругою, струмом, температурою, потужністю, SoH та прогнозованим класом деградації. Центральна частина інтерфейсу призначена для графіків зміни напруги, струму та температури в часі. Окремо відображаються попередження про небажані режими роботи, наприклад наближення напруги до нижньої межі або перевищення температури.

Інформаційна панель користувача містить поточні значення параметрів акумулятора, графіки зміни вимірюваних величин і блок прогнозування деградації. Такий інтерфейс дозволяє користувачу швидко оцінити поточний стан акумулятора, переглянути характер його розряду та отримати попередження у разі небажаного режиму роботи [24].

Загальний вигляд вебінтерфейсу системи моніторингу стану акумулятора наведено на рис. 3.22.

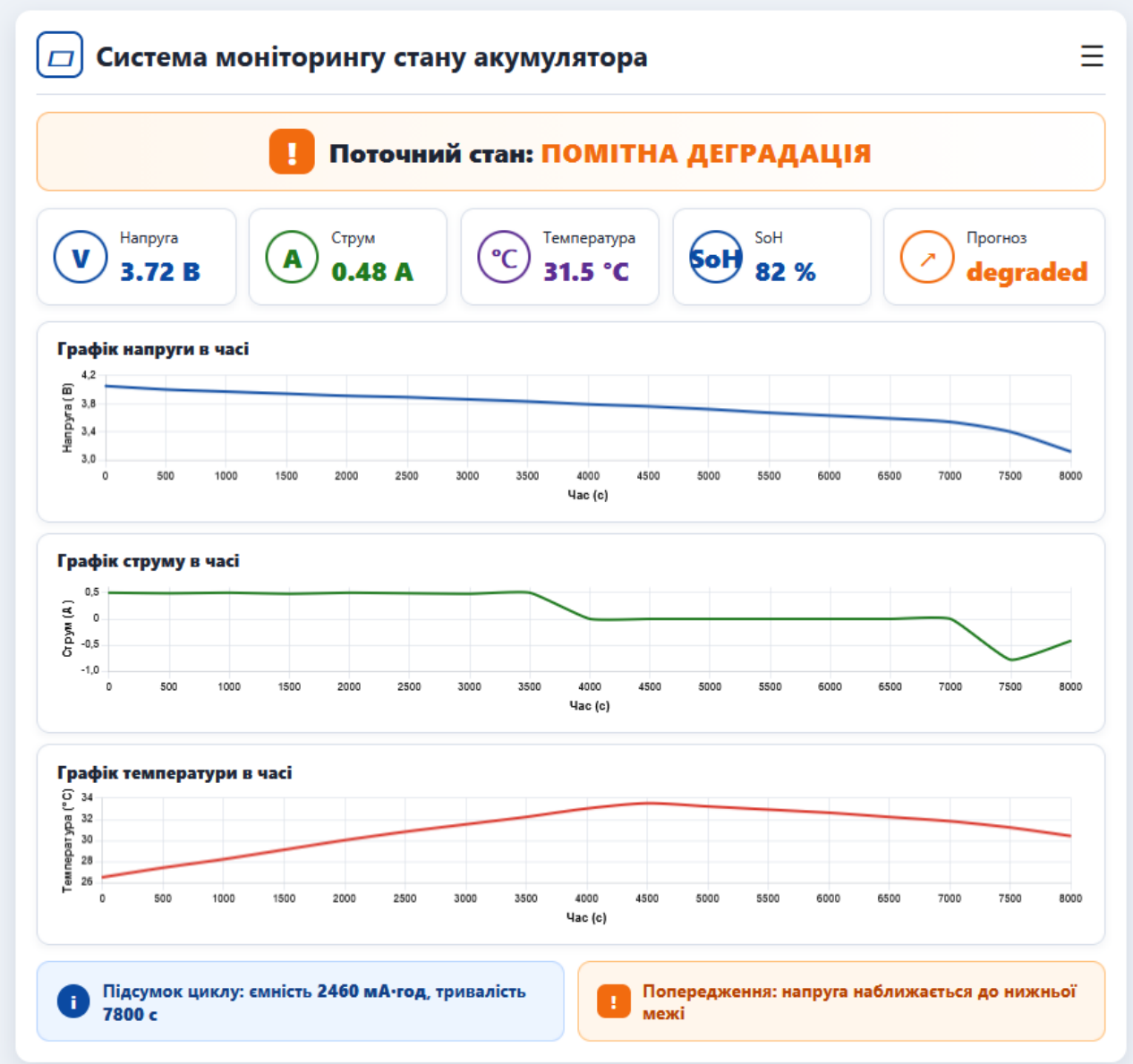

Рисунок 3.22 – Веб-інтерфейс користувача

Принцип роботи серверного рівня полягає в тому, що Flask-сервер періодично отримує дані з файлів вимірювань і результатів аналітичного модуля. Для веб-сторінки передбачено окремі маршрути: головна сторінка інтерфейсу, отримання останнього вимірювання, отримання історії параметрів для графіків, отримання результатів останнього циклу та отримання прогнозу деградації [23].

Веб-сторінка звертається до цих маршрутів і оновлює інформацію без перезавантаження. Основні маршрути локального веб-сервера наведено в табл. 3.3, а їх програмний код у додатку В.

Таблиця 3.3 – Основні маршрути серверного рівня

Маршрут	Призначення
/	Відкриття головної веб-сторінки моніторингу
/api/current	Отримання останніх значень напруги, струму, температури та потужності
/api/history	Передавання історії вимірювань для побудови графіків
/api/last-cycle	Отримання підсумкових параметрів останнього циклу розряду
/api/prediction	Отримання прогнозованого класу деградації акумулятора

Результати прогнозування, отримані з даного рівня, відображаються у веб-інтерфейсі у вигляді текстового стану. Наприклад, акумулятор може бути віднесений до одного з класів: нормальний стан, помітна деградація або критичне зношення. Це робить результат роботи дерева прийняття рішень зрозумілим для користувача, оскільки він бачить не лише числові параметри, а й узагальнений висновок щодо придатності акумулятора.

Таким чином, серверний рівень і веб-інтерфейс забезпечують зручну взаємодію користувача з комп’ютерною системою моніторингу. Flask-сервер, що працює на Raspberry Pi 5, передає у браузер поточні вимірювання, графіки, підсумки циклів і результати прогнозування. Веб-інтерфейс дозволяє контролювати стан Li-ion акумулятора 18650 у наочній формі та своєчасно отримувати попередження про небажані режими роботи.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при ураженні електричним струмом.

Небезпека ураження струмом може чекати людину як вдома, так і на вулиці. Виявити пошкоджений або оголений провід, що знаходиться під напругою дуже важко – ні за звуком, ні за запахом, ні візуально провід під напругою не відрізняється від того, який не заживлений у мережі. Тому вкрай необхідно пам’ятати правила електробезпеки, щоб уникнути травматизму [26].

Приміщення з робочими місцями користувачів комп’ютерів для забезпечення електробезпеки обладнання, а також для захисту від ураження електричним струмом самих користувачів ПК повинні мати достатні технічні засоби захисту відповідно до НПАОП 40.1-1.07-01 “Правила експлуатації електрозахисних засобів”, НПАОП 40.1-1.21-98 “Правила безпечної експлуатації електроустановок споживачів”, НПАОП 40.1-1.32-01 “Правила будови електроустановок. Електрообладнання спеціальних установок”.

З метою запобігання ушкодженням, що можуть статися через ураження електричним струмом, загоряння, коротке замикання тощо, розроблено загальний стандарт безпеки IEC 950. Загальним стандартом електробезпечності для країн Європейської співдружності є Cemark [26].

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, перейти на негорючу ізоляцію.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ПЕОМ, периферійні пристрої ПЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ повинні підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Необхідно унеможливити з'єднання контактів фазових провідників з контактами нульового захисного провідника.

Неприпустимим є підключення ПЕОМ та периферійних пристроїв ПЕОМ до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв [27].

Людина, яка піддається дії струму, не може покликати на допомогу та самостійно звільнитись від предмету, через який її ударило струмом. Дотик до струмопровідних частин у більшості випадків призводить до судом м’язів, які обмежують здатність рухатись чи говорити. Як правило, про те, що людина у небезпеці свідчить її несподіване падіння на вулиці або неприродне відкидання від джерела струму невидимою силою, раптова втрата свідомості, судоми, яскраво виражене мимовільне скорочення м’язів, опіки на тілі з різко окресленими межами.

При ураженні електричним струмом в першу чергу потрібно звільнити потерпілого від струмопровідних частин обладнання. Необхідно швидко відключити від мережі ту частину електрообладнання, до якої доторкається людина.

Будь-яке зволікання при наданні допомоги призводить до загибелі людини, яка знаходиться під дією струму. При звільненні потерпілих від струмопровідних частин або проводу вимикають струм, використовуючи сухий одяг, палицю, дошку, шапку, сухі рукавиці, рукав одягу, діелектричні рукавиці.

Провідники перерізають інструментом з ізольованими ручками, перерубують сокирою з дерев'яним сухим топорищем. Потерпілого можна відтягнути від струмопровідних частин за сухий одяг. При цьому, людина, яка відтягує потерпілого повинна уникати дотику до навколишніх металевих предметів та до відкритих частин тіла потерпілого. Відтягуючи потерпілого за ноги, не можна торкатися його взуття, оскільки воно може бути сирим і стає провідником електричного струму. Той, хто надає допомогу, повинен одягнути діелектричні рукавиці або обмотати руки шарфом, натягнути на них рукав піджака або пальта. Можна також ізолювати себе, ставши на гумовий килимок, суху дошку тощо.

Після звільнення потерпілого від дії струму потрібно відразу ж надати йому першу медичну допомогу та негайно сповістити службу екстреної медичної допомоги. Далі потрібно почати серцево-легеневу реанімацію людини, що не подає ознак життя [27].

Якісний непрямий масаж серця (якомога швидше починати виконувати непрямий масаж серця, виконуючи натискання у нижній половині грудини («в центрі грудної клітки»), натисненням на глибину не менше 5, але не більше 6 см, натисненням на грудну клітку зі швидкістю 100–120 натисків/хв з якомога меншою кількістю переривань зможливістю грудної клітки повністю випрямитися після кожного стиснення. Не потрбіно спиратися на груди (грудну клітку), настиснення необхдіно виконувати на твердій поверхні.

Штучне дихання – чергуються 30 натискань та 2 вдихи. Якщо не можливо забезпечити штучну вентиляцію легенів, виконується безперервне натискання на грудну клітку.

Автоматичний зовнішній дефібрилятор (як тільки АЗД надійде (буде доступний), або якщо такий вже є на місці, де наявний постраждалий з зупинкою серця, потрібно увімкнути його, прикріпіти електродні прокладки (наліпки) до оголеної грудної клітки постраждалого відповідно до положення, вказаного на АЗД або на прокладках (наліпках) та дотримуватися голосових (та/або візуальних) підказок АЗД.

4.2 Оцінка майбутнього фізичного та психологічного навантаження на людину, яка обслуговує пристрій

Розумова праця об'єднує роботи пов'язані зі сприйняттям та опрацюванням інформації, необхідністю переважного навантаження сенсорного апарату, уваги, пам'яті, а також активації процесів мислення, емоційної сфери [27].

Виділяють такі різновиди розумової праці:

– операторська;
– адміністративно-керівна;
– творча;
– праця викладачів і медичних працівників;
– праця учнів і студентів.

Вказані види роботи відрізняються по організації трудового процесу, рівномірності навантаження, ступеню емоційного напруження. При розумовій діяльності загострюється сприйняття, увага, пам'ять. Посилюється кровопостачання мозку, підвищується енергетичний обмін нервових клітин, змінюються показники біоелектричної активності мозку.

При інтенсивній інтелектуальній діяльності споживання кисню 100 г кори головного мозку в 5-6 разів більше, ніж споживання скелетного м'язу такої ж ваги при максимальному навантаженні. Розумова праця, а особливо, робота оператора супроводжується деякою нервово-емоційною напругою. Вона

призводить до посилення серцево-судинної діяльності, дихання, енергообміну, підвищення м'язового тонусу.

Після закінчення розумової праці "робоча домінанта" повністю не згасає, зумовлюючи більш тривале втомлення та виснаження ЦНС при розумовій праці, ніж при фізичній [27].

Практичне значення заходів щодо підвищення працездатності випливає із закономірностей її динаміки і зводиться до:

– збільшення фази стійкого стану в фонді робочого часу;

– прискорення процесу впрацювання;

– віддалення фази розвитку втоми;

– забезпечення високої продуктивності праці за нормальних фізіологічних затрат.

Комплекс заходів щодо підвищення і збереження працездатності працівників на оптимальному рівні реалізується на техніко-організаційному, соціально-економічному, санітарно-гігієнічному, медико-біологічному, психологічному напрямках.

Могутнім фактором високої працездатності і продуктивності праці є оптимізація трудових навантажень на основі механізації і автоматизації виробничих процесів, удосконалення технології, скорочення і ліквідації важкої ручної праці. Доведено, що при правильній організації праці на легких роботах спостерігається найбільша тривалість фази стійкого стану, а на важких роботах вона нетривала [27].

Високий рівень працездатності безпосередньо залежить від умов праці, оскільки поліпшення їх супроводжується зменшенням енергетичних затрат організму на подолання несприятливого впливу факторів виробничого середовища.

Важливим напрямком підвищення працездатності працюючих є ритмізація трудових процесів, оптимізація темпу роботи, а також раціоналізація трудових рухів на фізіологічній основі, що сприяє формуванню і закріпленню робочих динамічних стереотипів, а отже зменшенню м’язових і вольових зусиль.

Ритмічна робота підвищує функціональні можливості організму, сприяє його тренованості і забезпечує економізацію енергетичних затрат.

Економізація функціональних затрат досягається завдяки стійкій домінанті і автоматизму дій, що виключає зайві рухи, розсіювання уваги тощо.

Особливе значення для підтримання працездатності працівників на високому рівні має раціональний режим праці і відпочинку.

Дослідження показують, що впровадження раціонального режиму праці і відпочинку на підприємствах забезпечує підвищення продуктивності праці на 8—10%, сприяє поліпшенню фізіологічного стану працівників (зменшується частота пульсу в процесі роботи, підвищується м'язова витривалість в кінці зміни, покращується координація рухів). Високій працездатності працівників сприяє і раціоналізація робочих місць на основі врахування антропометричних, біомеханічних і психофізіологічних вимог, що обумовлює раціональну робочу позу, зменшення статичних навантажень, оптимізацію робочої зони та інформаційних потоків. Висока працездатність забезпечується за рахунок використання факторів естетичного впливу на працюючих. Такими факторами є колір, світло, музика. Особливо слід підкреслити значення функціональної музики, яка впливає на емоційну сферу людини, підвищує збудливість і лабільність центральної нервової системи [27].. На початку роботи вона прискорює процес, а в кінці робочого дня зменшує суб'єктивне відчуття стомленості. Вплив функціональної музики посилюється, якщо вона поєднується з фізичними вправами. Останні підвищують лабільність органів, які безпосередньо беруть участь у виконанні роботи, активізують роботу органів дихання і кровообігу.

Особливе значення в підвищенні працездатності працівників має створення сприятливого соціально-психологічного клімату в організації, високий рівень мотивації праці, ефективна система стимулювання результатів діяльності, рівень життя в цілому і охорона здоров'я населення. Ефективність заходів, спрямованих на підвищення працездатності працівників, можна оцінити приростом продуктивності праці, який досягається за рахунок збільшення фази стійкого стану в загальній тривалості робочої зміни.

ВИСНОВКИ

У кваліфікаційній роботі розроблено комп’ютерну систему моніторингу стану акумуляторів з прогнозуванням їх деградації. Запропоноване рішення орієнтоване на контроль Li-ion акумуляторного елемента типорозміру 18650 і призначене для отримання вимірювальних даних, їх програмної обробки, оцінювання технічного стану акумулятора та відображення результатів користувачу.

У першому розділі проаналізовано особливості роботи Li-ion акумуляторів 18650, основні причини їх деградації та показники, які застосовуються для оцінювання стану акумулятора.

У другому розділі спроєктовано апаратну частину системи. Як центральний обчислювальний модуль обрано Raspberry Pi 5, який забезпечує зчитування даних, їх обробку, збереження та роботу веб-інтерфейсу. Для вимірювання напруги, струму й потужності використано модуль INA219, для контролю температури – датчик DS18B20, для заряджання акумулятора – модуль TP4056. Для контрольованого розряду обґрунтовано використання резистора 7,5 Ом / 10 Вт, який забезпечує помірний струм розряду та має достатній запас за потужністю.

У третьому розділі реалізовано програмну частину системи. Програмне забезпечення побудовано за багаторівневою архітектурою, яка включає рівень апаратної взаємодії, нормалізації та збереження даних, аналітичний рівень, прогностичний рівень, серверний рівень і веб-інтерфейс користувача. Для збереження результатів передбачено CSV-файли: measurements.csv для поточних вимірювань і cycles.csv для підсумкових даних циклів розряду.

У четвертому розділі розглянуто питання охорони праці і безпеки життєдіяльності.

Розроблена система не є повноцінною промисловою BMS, оскільки не виконує балансування комірок і складного силового захисту. Водночас вона реалізує основні діагностичні функції, необхідні для моніторингу стану Li-ion елемента.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Клименко І.А., Таранюк В.А., Ткаченко В.В., Каплунов А.В. Мікропроцесорні системи. Частина 1. Програмування для процесора Cortex M4: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2022. 100 с.

2. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп'ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ, 2019. 150 с.

3. Жаровський Р.О., Луцик Н.С., Осухівська Г.М., Паламар А.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів першого (бакалаврського) рівня вищої освіти за спеціальністю 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль: ТНТУ, 2024. 39 с.

4. Нікітчук А.В. Програмування вбудованих систем Інтернету речей: навч. посіб. Запоріжжя: ЗНУ, 2024. 140 с.

5. Колонтаєвський Ю.П. Комп'ютерна електроніка: навч. посіб. Харків: ХНУМГ ім. О.М. Бекетова, 2018. 160 с.

6. Yao L., Xu S., Tang A., Zhou F., Hou J., Xiao Y., Fu Z. A Review of Lithium-Ion Battery State of Health Estimation and Prediction Methods. World Electric Vehicle Journal. 2021. Vol. 12, No. 3. Article 113. DOI: https://doi.org/10.3390/wevj12030113.

7. Li X., Wang Z. State of Health Estimation for Lithium-Ion Battery by Combining Incremental Capacity Analysis with Gaussian Process Regression. Energy. 2019. Vol. 174. P. 971–982.

8. Uddin K., Schofield J., Widanage W.D. State of Health Estimation of Lithium-Ion Batteries in Vehicle-to-Grid Applications Using Recurrent Neural Networks for Learning the Impact of Degradation Stress Factors. Batteries. 2022. Vol. 8, No. 12. Article 262. DOI: https://doi.org/10.3390/batteries8120262.

9. NASA Ames Prognostics Center of Excellence. Li-ion Battery Aging Datasets. URL: https://data.nasa.gov/dataset/li-ion-battery-aging-datasets (дата звернення: 07.06.2026 р.).

10. E-One Moli Energy Corp. INR-18650-P26A Lithium-Ion Rechargeable Battery: Product Data Sheet. URL: https://www.molicel.com/wp-content/uploads/INR18650P26A-V2-80087.pdf (дата звернення: 10.06.2026 р.).

11. Тиш Є.В., Гончаренко О.Р. Алгоритм автоматизованого режиму роботи сонячного трекера. International Scientific Journal Grail Of Science. №10, Vinnytsia-Vienna, 2021. P.268-271.

12. Тиш Є.В., Лупенко С.А. Математичне моделювання, методи аналізу та комп'ютерної імітації серцевого ритму при фізичних навантаженнях. Наукова монографія. Львів : «Магнолія-2006», 2020. 150с.

13. Тиш Є.В. Узагальнений алгоритм синтезу компонентів комп'ютерних систем на основі мікропрограмних автоматів. Вчені записки Таврійського національного університету імені В.І. Вернадського. Том 36 (75), № 1, 2025. С. 247-253.

14. Tysh Ie. Methods for calculating the reliability of computer networks. Theoretical and Practical Scientific Achievements: Research and Results of their Implementation. Collection of Scientific Papers «SCIENTIA» with Proceedings of the X International Scientific and Theoretical Conference, February 13, 2026. Liverpool, England; United Kingdom: International Center of Scientific Research. P.173-175.

15. Voloshchuk A., Velychko D., Osukhivska H., Palamar A. Computer system for energy distribution in conditions of electricity shortage using artificial intelligence. CEUR Workshop Proceedings, 2nd International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2024), Ternopil, Ukraine, June 12-14, 2024. Vol. 3742 P. 66-75.

16. Raspberry Pi Ltd. Raspberry Pi 5 Product Brief. 2023. URL: https://pip.raspberrypi.com/documents/RP-008348-DS-raspberry-pi-5-product-brief.pdf (дата звернення: 12.06.2026 р.).

17. Raspberry Pi Ltd. Raspberry Pi Documentation. URL: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html (дата звернення: 12.06.2026 р.).

18. PyPI. pi-ina219 1.4.1: Python Library for Raspberry Pi and INA219 Current Sensor. 2023. URL: https://pypi.org/project/pi-ina219/ (дата звернення: 12.06.2026 р.).

19. EDN. INA219 Current Sensor Module Primer. 2021. URL: https://www.edn.com/ina219-current-sensor-module-primer/ (дата звернення: 13.06.2026 р.).

20. OSOYOO. Lesson 1: DS18B20 Temperature Sensor for Raspberry Pi. 2024. URL: https://osoyoo.com/2024/09/08/lesson-1-ds18b20-temperature-sensor-for-raspberry-pi/ (дата звернення: 13.06.2026 р.).

21. Blikai. What is TP4056 Module? All You Need to Know. 2024. URL: https://www.blikai.com/blog/what-is-tp4056-module-all-you-need-to-know (дата звернення: 07.06.2026 р.).

22. Python Software Foundation. Python 3 Documentation. URL: https://docs.python.org/3/ (дата звернення: 13.06.2026 р.).

23. Pallets Projects. Flask Documentation. URL: https://flask.palletsprojects.com/ (дата звернення: 14.06.2026 р.).

24. pandas development team. pandas Documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 14.06.2026 р.).

25. scikit-learn developers. Decision Trees. URL: https://scikit-learn.org/stable/modules/tree.html (дата звернення: 14.06.2026 р.).

26. Chart.js Documentation. URL: https://www.chartjs.org/docs/latest/ (дата звернення: 15.06.2026 р.).

27. Гурик О.Я., Окіпний І.Б. Методичні вказівки для написання розділу «Безпека життєдіяльності, основи охорони праці» в кваліфікаційних роботах здобувачів освітнього рівня «бакалавр». Тернопіль: ТНТУ імені Івана Пулюя, 2021. 20 с.

28. Катренко Л.А., Катренко А.В. Охорона праці в галузі комп’ютинґу. Львів: Магнолія-2006. 2012. 544 с.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем і програмної інженерії

Кафедра комп’ютерних систем та мереж

“Затверджую”

Завідувач кафедри КС

_____ Осухівська Г.М.

“_*02*_” __*лютого*_____2026 р

КОМП’ЮТЕРНА СИСТЕМА МОНІТОРИНГУ СТАНУ АКУМУЛЯТОРІВ З ПРОГНОЗУВАННЯМ ЇХ ДЕГРАДАЦІЇ

ТЕХНІЧНЕ ЗАВДАННЯ

на _*11*_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп’ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІ-41
_____ к.т.н., доц. Тиш Є.В.	_____ Брайляк Д.В.
«_*02*_» ____*лютого*__ 2026 р.	«_*02*_» __*лютого*__ 2026 р.

Тернопіль 2026

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: «Комп’ютерна система моніторингу стану акумуляторів з прогнозуванням їх деградації».

Умовне позначення кваліфікаційної роботи: КС КРБ 123.149.00.00

1.2 Виконавець

Студент групи СІ-41, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Брайляк Дмитро Володимирович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№4/9-188 від 24.04.2026 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 26.01.2026 р.

Плановий термін завершення виконання кваліфікаційної роботи – 21.06.2026 р.

1.5 Порядок оформлення та пред’явлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ISO, ЕСКД, ЕСПД та ДСТУ.

Пред’явлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи.

Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Комп’ютерна система моніторингу стану акумуляторів з прогнозуванням їх деградації розробляється як апаратно-програмний комплекс для вимірювання, збереження, аналізу та візуалізації параметрів Li-ion акумуляторного елемента. Система призначена для спостереження за зміною електричних і температурних характеристик акумулятора під час контрольованого розряду та подальшого оцінювання його технічного стану.

Система повинна забезпечувати зчитування напруги, струму, потужності та температури акумулятора, збереження результатів вимірювання, розрахунок фактичної ємності, визначення показника SoH та прогнозування рівня деградації. Результати роботи системи повинні відображатися у веб-інтерфейсі у вигляді поточних значень, графіків, підсумкових показників циклу та попереджень.

2.2 Мета створення системи

Метою створення комп’ютерної системи моніторингу стану акумуляторів є розроблення програмно-апаратного рішення, яке забезпечує контроль параметрів Li-ion акумулятора 18650, оцінювання його фактичної ємності та прогнозування деградації на основі накопичених даних.

Для досягнення мети необхідно розв’язати такі задачі:

– проаналізувати особливості роботи Li-ion акумуляторів типорозміру 18650;

– визначити основні фактори деградації акумулятора;

– обґрунтувати вибір параметрів моніторингу: напруги, струму, температури та циклів розряду;

– обрати апаратні компоненти системи;

– розробити структурну схему підключення компонентів;

– реалізувати зчитування даних із модуля INA219 і датчика DS18B20;

– організувати збереження вимірювань у CSV-файл;

– реалізувати розрахунок фактичної ємності та показника SoH;

– розробити модуль прогнозування деградації на основі дерева прийняття рішень;

– створити локальний вебінтерфейс для відображення результатів моніторингу.

2.3 Характеристика об'єкту

Об’єктом проєктування є комп’ютерна система моніторингу стану Li-ion акумулятора 18650 з прогнозуванням його деградації. Система виконує вимірювання параметрів акумулятора під час контрольованого розряду, зберігає отримані дані, розраховує діагностичні показники та відображає результати користувачу.

Система складається з апаратної та програмної частин. Апаратна частина забезпечує підключення акумулятора, вимірювання напруги, струму та температури, організацію контрольованого розряду й передавання даних до обчислювального модуля. Програмна частина забезпечує зчитування, нормалізацію, збереження, аналіз, прогнозування та візуалізацію результатів.

Raspberry Pi 5 виконує роль центрального обчислювального вузла. Він зчитує дані з вимірювальних модулів, виконує програмну обробку, зберігає результати у файли, запускає модуль прогнозування та забезпечує роботу локального вебінтерфейсу.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Комп’ютерна система моніторингу стану акумуляторів повинна забезпечувати повний цикл роботи з вимірювальними даними: від зчитування параметрів акумулятора до оцінювання його стану та відображення результатів користувачу.

Основними вимогами до системи є:

– вимірювання напруги акумулятора;

– вимірювання струму розряду;

– розрахунок потужності навантаження;

– вимірювання температури акумулятора;

– організація контрольованого розряду через резистор 7,5 Ом / 10 Вт;

– збереження поточних вимірювань у CSV-файл;

– формування журналу циклів розряду;

– розрахунок фактичної ємності акумулятора;

– визначення показника SoH;

– прогнозування стану акумулятора на основі дерева прийняття рішень;

– відображення результатів у веб-інтерфейсі;

– формування попереджень при зниженні напруги або підвищенні температури;

– модульна структура апаратної та програмної частин.

Система повинна бути придатною для моніторингу стану одного Li-ion акумуляторного елемента 18650.

3.1.1 Вимоги до способів та засобів зв’язку між компонентами системи

Обмін даними між компонентами системи повинен здійснюватися через прості та надійні інтерфейси, придатні для реалізації лабораторного прототипу.

Взаємодія між компонентами повинна виконуватися такими способами:

– Li-ion акумулятор 18650 підключається до розрядного кола через модуль INA219;

– навантажувальний резистор 7,5 Ом / 10 Вт підключається послідовно в коло розряду;

– модуль INA219 підключається до Raspberry Pi 5 через інтерфейс I2C;

– датчик температури DS18B20 підключається до Raspberry Pi 5 через цифрову лінію 1-Wire;

– модуль TP4056 використовується для заряджання одного Li-ion елемента;

– Raspberry Pi 5 підключається до локальної мережі через Ethernet або Wi-Fi;

– користувач взаємодіє із системою через браузер і локальний веб-сервер Flask.

З’єднання між компонентами повинні забезпечувати стабільне зчитування даних. Провідники в розрядному колі повинні бути розраховані на струм, що виникає під час контрольованого розряду.

3.1.2 Вимоги по діагностуванню системи

Діагностування системи повинно передбачати перевірку працездатності основних апаратних і програмних компонентів перед початком моніторингу.

Під час запуску системи повинні перевірятися:

– наявність живлення Raspberry Pi 5;

– доступність інтерфейсу I2C;

– доступність модуля INA219;

– доступність датчика температури DS18B20;

– коректність зчитування напруги, струму та температури;

– наявність файлу для збереження вимірювань або можливість його створення;

– можливість запису даних у CSV-файл;

– доступність локального вебсервера;

– коректність передавання даних у вебінтерфейс.

У разі помилки система повинна формувати повідомлення про відсутність датчика, некоректні дані, неможливість запису файлу або недоступність вебсервера.

3.1.3 Перспективи розвитку, модернізація системи

Перспективи розвитку системи пов’язані з розширенням функціональних можливостей, підвищенням точності прогнозування та переходом до більш автоматизованого режиму роботи.

Основними напрямами модернізації є:

– додавання підтримки кількох акумуляторних елементів;

– використання електронного навантаження замість резистора;

– автоматичне вмикання та вимикання розряду за допомогою MOSFET-ключа;

– збереження результатів у базу даних SQLite;

– передавання даних на віддалений сервер;

– використання складніших моделей машинного навчання;

– розширення набору вхідних ознак для прогнозування;

– реалізація мобільного або хмарного інтерфейсу;

– додавання функцій повноцінної BMS для багатокоміркових акумуляторних батарей.

Подальше вдосконалення системи може бути спрямоване на використання у навчальних лабораторних стендах, дослідженні деградації акумуляторів та системах технічної діагностики джерел живлення.

3.1.4 Вимоги до надійності системи

Система повинна забезпечувати стабільне зчитування, збереження, обробку та відображення даних під час проведення контрольованого розряду акумулятора.

Надійність системи повинна забезпечуватися такими засобами:

– використанням окремого стабілізованого живлення для Raspberry Pi 5;

– правильним підключенням акумулятора, навантаження та вимірювальних модулів;

– контролем полярності підключення Li-ion елемента;

– використанням навантажувального резистора з достатнім запасом потужності;

– перевіркою допустимих меж напруги, струму та температури;

– обробкою помилок зчитування даних;

– збереженням службового статусу вимірювань;

– регулярним записом результатів у CSV-файл;

– формуванням попереджень при небажаних режимах роботи.

Програмне забезпечення повинно продовжувати роботу при появі окремих некоректних значень. Такі значення не повинні використовуватися для розрахунку ємності, SoH та прогнозування деградації.

3.1.5 Вимоги до функцій системи

Комп’ютерна система моніторингу стану акумуляторів повинна забезпечувати такі функції:

– запуск програмного забезпечення на Raspberry Pi 5;

– ініціалізацію інтерфейсів I2C та 1-Wire;

– зчитування напруги й струму з модуля INA219;

– зчитування температури з датчика DS18B20;

– розрахунок потужності навантаження;

– формування запису вимірювання з часовою міткою;

– перевірку коректності отриманих значень;

– нормалізацію вимірювальних даних;

– збереження поточних вимірювань у файл measurements.csv;

– формування журналу циклів у файлі cycles.csv;

– розрахунок фактичної ємності акумулятора;

– розрахунок показника SoH;

– перевірку граничної напруги розряду;

– перевірку температурного режиму;

– прогнозування класу стану акумулятора;

– запуск локального Flask-сервера;

– відображення параметрів у вебінтерфейсі;

– побудову графіків зміни напруги, струму та температури;

– формування попереджень для користувача.

Raspberry Pi 5 повинен виконувати такі задачі:

– взаємодіяти з вимірювальними модулями;

– виконувати обробку вимірювальних даних;

– зберігати результати у файли;

– запускати модуль прогнозування;

– забезпечувати роботу веб-сервера та веб-інтерфейсу.

3.1.6 Вимоги до апаратного забезпечення

Апаратне забезпечення системи повинно бути доступним, сумісним між собою та достатнім для реалізації лабораторного прототипу моніторингу стану акумулятора.

Основними апаратними компонентами системи повинні бути:

– Li-ion акумуляторний елемент 18650;

– одноплатний комп’ютер Raspberry Pi 5;

– модуль INA219 для вимірювання напруги, струму та потужності;

– цифровий датчик температури DS18B20;

– модуль заряджання TP4056;

– навантажувальний резистор 7,5 Ом / 10 Вт;

– блок живлення 5 В для Raspberry Pi 5;

– тримач акумулятора 18650;

– з’єднувальні провідники, клемники та монтажні елементи.

Апаратна частина повинна забезпечувати безпечне підключення акумулятора, проведення контрольованого розряду, вимірювання електричних і температурних параметрів та передавання даних до Raspberry Pi 5.

3.1.7 Вимоги до програмного забезпечення

Програмне забезпечення системи повинно мати модульну структуру та працювати на Raspberry Pi 5. Основною мовою реалізації повинна бути Python.

Програмне забезпечення повинно включати такі модулі:

– модуль зчитування даних із INA219;

– модуль зчитування температури з DS18B20;

– модуль формування первинного запису вимірювання;

– модуль нормалізації та перевірки даних;

– модуль збереження даних у CSV-файл;

– модуль журналювання циклів розряду;

– аналітичний модуль для розрахунку ємності та SoH;

– модуль прогнозування деградації на основі дерева прийняття рішень;

– серверний модуль на основі Flask;

– вебі-нтерфейс користувача.

Програмна частина повинна забезпечувати:

– регулярне зчитування вимірювальних даних;

– перевірку допустимих меж параметрів;

– збереження даних у файли measurements.csv і cycles.csv;

– розрахунок фактичної ємності акумулятора;

– визначення показника SoH;

– класифікацію стану акумулятора як нормального, деградованого або критичного;

– передавання результатів у веб-інтерфейс;

– відображення поточних значень, графіків і попереджень;

– коректну обробку помилок зчитування та запису даних.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

1 Будова Li-ion акумуляторного елементу та графік його деградації.

2 Взаємозв’язок параметрів моніторингу та показників стану акумулятора.

3 Архітектура комп’ютерної системи моніторингу стану акумуляторів з прогнозуванням їх деградації.

4 Структурна схема комп’ютерної системи моніторингу стану акумуляторів.

*Примітка: У комплект документації можуть вноситися міни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка технічного завдання	26.01 – 02.02
2	Робота над першим розділом «Аналіз принципів функціонування та вимог до систем моніторингу стану акумуляторів»	03.02 – 15.02
3	Робота над другим розділом «Проєктування системи моніторингу стану акумуляторів на апаратному рівні»	20.04 – 25.04
4	Робота над третім розділом «Реалізація програмних модулів системи моніторингу стану акумуляторів»	26.04 – 05.05
5	Робота над четвертим розділом «Безпека життєдіяльності, основи охорони праці»	07.05 – 25.05
6	Оформлення пояснювальної записки і графічного матеріалу	26.05 – 7.06
7	Перевірка на академічний плагіат, перевірка керівником та консультантами	8.06 – 14.06
8	Попередній захист кваліфікаційної роботи бакалавра	15.06 – 21.06
9	Захист кваліфікаційної роботи бакалавра	22.06.2026

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

Програмний код функцій та класу аналітичного рівня

```
def evaluate_limits(measurement):
    warnings = []

    voltage = float(measurement["voltage_v"])
    temperature = float(measurement["temperature_c"])

    if voltage <= MIN_VOLTAGE_V:
        warnings.append("Напруга акумулятора наблизилася до нижньої
    межі")

    if temperature >= MAX_TEMPERATURE_C:
        warnings.append("Температура акумулятора перевищує
    допустимий рівень")

    if warnings:
        return {
            "state": "warning",
            "messages": warnings
        }

    return {
        "state": "normal",
        "messages": []
    }

def summarize_cycle(cycle_id, measurements):
    if not measurements:
        raise ValueError("Список вимірювань циклу порожній")

    start_time = parse_time(measurements[0]["timestamp"])
    end_time = parse_time(measurements[-1]["timestamp"])
    duration_s = (end_time - start_time).total_seconds()

    capacity_mah = calculate_capacity_mah(measurements)
    soh_percent = calculate_soh_percent(capacity_mah)

    currents = [
        abs(float(item["current_a"]))
        for item in measurements
    ]

    temperatures = [
        float(item["temperature_c"])
        for item in measurements
    ]

    avg_current = sum(currents) / len(currents)
    avg_temperature = sum(temperatures) / len(temperatures)

    return {
```

```
        "cycle_id": cycle_id,
        "start_time": start_time.isoformat(timespec="seconds"),
        "end_time": end_time.isoformat(timespec="seconds"),
        "duration_s": int(duration_s),
        "capacity_mah": round(capacity_mah, 2),
        "avg_current_a": round(avg_current, 4),
        "avg_temperature_c": round(avg_temperature, 2),
        "soh_percent": round(soh_percent, 2)
    }


class BatteryAnalyzer:
    def __init__(self, nominal_capacity_mah=3000.0):
        self.nominal_capacity_mah = nominal_capacity_mah

    def calculate_capacity(self, measurements):
        return calculate_capacity_mah(measurements)

    def calculate_soh(self, capacity_mah):
        return calculate_soh_percent(
            capacity_mah,
            self.nominal_capacity_mah
        )

    def check_current_state(self, measurement):
        return evaluate_limits(measurement)

    def build_cycle_summary(self, cycle_id, measurements):
        summary = summarize_cycle(cycle_id, measurements)
        summary["soh_percent"] =
    self.calculate_soh(summary["capacity_mah"])

        return summary
```

Додаток В

Програмний код основних компонентів веб-сервера

```
from flask import Flask, jsonify, render_template

from analytics import BatteryAnalyzer
from degradation_model import DegradationPredictor
from storage import (
    read_last_cycle,
    read_last_measurement,
    read_measurement_history,
)


app = Flask(__name__)

analyzer = BatteryAnalyzer(nominal_capacity_mah=3000.0)
predictor = DegradationPredictor(dataset_path="cycles.csv")


@app.route("/")
def index():
    return render_template("index.html")

@app.route("/api/current")
def get_current_measurement():
    measurement = read_last_measurement("measurements.csv")

    if measurement is None:
        return jsonify({
            "status": "no_data",
            "message": "Дані вимірювань відсутні"
        })

    state_info = analyzer.check_current_state(measurement)

    response = {
        "status": "ok",
        "measurement": measurement,
        "state": state_info["state"],
        "warnings": state_info["messages"]
    }

    return jsonify(response)

@app.route("/api/history")
def get_measurement_history():
    history = read_measurement_history(
        file_path="measurements.csv",
        limit=300
    )

    return jsonify({
```

```
        "status": "ok",
        "items": history
    })


@app.route("/api/last-cycle")
def get_last_cycle():
    cycle_summary = read_last_cycle("cycles.csv")

    if cycle_summary is None:
        return jsonify({
            "status": "no_data",
            "message": "Журнал циклів ще не сформовано"
        })

    return jsonify({
        "status": "ok",
        "cycle": cycle_summary
    })


@app.route("/api/prediction")
def get_prediction():
    cycle_summary = read_last_cycle("cycles.csv")

    if cycle_summary is None:
        return jsonify({
            "status": "no_data",
            "message": "Немає даних для прогнозування"
        })

    prediction = predictor.predict(cycle_summary)

    return jsonify({
        "status": "ok",
        "prediction": prediction
    })


if __name__ == "__main__":
    app.run(
        host="0.0.0.0",
        port=5000,
        debug=False
    )

import csv
from pathlib import Path


def read_csv_rows(file_path):
    path = Path(file_path)

    if not path.exists():
        return []
```

```
    with path.open("r", encoding="utf-8") as file:
        reader = csv.DictReader(file)
        return list(reader)


def read_last_measurement(file_path):
    rows = read_csv_rows(file_path)

    if not rows:
        return None

    return rows[-1]


def read_measurement_history(file_path, limit=300):
    rows = read_csv_rows(file_path)

    if not rows:
        return []

    return rows[-limit:]


def read_last_cycle(file_path):
    rows = read_csv_rows(file_path)

    if not rows:
        return None

    return rows[-1]

<script>
    async function updateCurrentValues() {
        const response = await fetch("/api/current");
        const data = await response.json();

        if (data.status !== "ok") {
            return;
        }

        const measurement = data.measurement;

        document.getElementById("voltage").textContent =
    measurement.voltage_v;
        document.getElementById("current").textContent =
    measurement.current_a;
        document.getElementById("temperature").textContent =
    measurement.temperature_c;
        document.getElementById("power").textContent =
    measurement.power_w;

        const warningList = document.getElementById("warning-list");
        warningList.innerHTML = "";

        data.warnings.forEach(function (message) {
            const item = document.createElement("li");
```

```
            item.textContent = message;
            warningList.appendChild(item);
        });
    }

    setInterval(updateCurrentValues, 2000);
    updateCurrentValues();
</script>

<script>
    async function updatePrediction() {
        const response = await fetch("/api/prediction");
        const data = await response.json();

        if (data.status !== "ok") {
            return;
        }

        const prediction = data.prediction;
        const predictedClass = prediction.predicted_class;

        document.getElementById("prediction").textContent =
    predictedClass;
    }

    setInterval(updatePrediction, 10000);
    updatePrediction();
</script>
```