Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерная інженерія та управління_____
(повна назва)
Кафедра _____Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____

Оптимізація енерговитрат вбудованих систем із використанням машинного навчання
(тема)

Виконав: здобувач ____4____ року навчання,
групи _____КІУКІ-21-7_____
_____ Плис О.А._____
(прізвище, ініціали)

Спеціальність 123 Комп'ютерна інженерія
_____
(код і повна назва спеціальності)
Тип програми _____освітньо-професійна_____
Освітня програма Комп'ютерна інженерія
_____
(повна назва освітньої програми)

Керівник _____ас. Хаханов І.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В.
(підпис) (прізвище, ініціали)

2025р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проєктування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія

Тип програми Освітньо-професійна

Освітня програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ:
Зав. кафедри
Чумаченко С.В

_____
(підпис)
06.05. 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувачеві Плису Олександру Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Оптимізація енерговитрат вбудованих систем із використанням машинного навчання

затверджена наказом університету від 21 05 2025 р. № 403Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17. 06. 2025 р.

3. Вихідні дані до роботи платформа Arduino Uno 32, середовище симуляції Wokwi, апаратне забезпечення включає в себе датчик руху, світлодіод, комунікаційний модуль Wi-Fi, мова програмування C++ з використанням Arduino Framework, створення плати arduino, що сприяє оптимізації електроенергії за рахунок контролю підключення до мережі Wi-Fi. Така плата забезпечує кращу автономність IoT речей за рахунок скорочення використання електроенергії на один із найбільш впливовіших на автономність речей, а саме підключення до мережі.

4. Перелік питань, що потрібно опрацювати в роботі вступ, аналіз предметної області, машинне навчання для оптимізації енерговитрат, моделювання системи керування Wi-Fi з використанням ML та практична реалізація в симуляторі, висновок.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
11 слайдів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача теми проєкту, узгодження і затвердження теми	06.05.2025 – 08.05.2025	
2	Аналіз предметної області, вибір компонентів системи	08.05.2025 – 13.05.2025	
3	Розробка структурної схеми пристрою, вибір апаратної платформи	13.05.2025 – 18.05.2025	
4	Розробка функціональної схеми програми	18.05.2025 – 20.05.2025	
5	Розробка програмних модулів. Проведення тестування	20.05.2025 – 28.05.2025	
6	Оформлення пояснювальної записки	28.05.2025 – 06.06.2025	
7	Підготовка ілюстративних матеріалів.	06.06.2025 – 12.06.2025	

Дата видачі завдання 06.05.2025 р.

Здобувач _____ Плис О.А.
(підпис) (прізвище, ініціали)

Керівник роботи _____ _____ ас. Хаханов І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи містить: 47 сторінок, 2 рисунка, 14 джерел посилання.

ESP32, WIFI, АВТОМАТИЗОВАНА СИСТЕМА, C++, УПРАВЛІННЯ, МІКРОКОНТРОЛЕР, IoT.

Метою кваліфікаційної роботи є створення інтелектуальної системи керування енергоспоживанням на базі платформи Arduino з використанням Wi-Fi модуля, яка за допомогою машинного навчання здатна адаптивно визначати періоди бездіяльності користувача та автоматизувати процес відключення передавача для мінімізації енерговитрат. У роботі виконано вибір та налаштування оптимального ML-алгоритму, реалізовано програмно-апаратний комплекс в середовищі симулятора WokWi.

ABSTRACT

The explanatory note of the qualification work contains: 47 pages, 2 pictures, 14 sources of reference.

ESP32, WIFI, AUTOMATED SYSTEM, C++, CONTROL, MICROCONTROLLER, IoT.

The purpose of the qualification work is to create an intelligent energy management system based on the Arduino platform using a Wi-Fi module, which, using machine learning, is able to adaptively determine periods of user inactivity and automate the process of turning off the transmitter to minimize energy consumption. The work includes the selection and configuration of the optimal ML algorithm, and the implementation of the hardware and software complex in the WokWi simulator environment.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕННЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IoT – Internet of Things (інтернет речей);

ML – Machine Learning (машинне навчання);

Mesh – Mesh Network (меш-мережа);

RISC – Reduced Instruction Set Computing (обчислення зі зменшеним набором інструкцій);

ARM – Advanced RISC Machine (удосконалена RISC-машина).

ВСТУП

Сучасний світ стрімко розвивається у напрямку інтернету речей (IoT) та вбудованих систем, які стають основою для розумних пристроїв, промислової автоматизації та побутової електроніки. Однак із зростанням кількості підключених девайсів постає критична проблема енергоефективності, особливо для пристроїв, що працюють від акумуляторів або мають обмежені джерела живлення. Найбільш енерговитратними компонентами таких систем часто виявляються бездротові модулі, зокрема Wi-Fi, які традиційно залишаються активними навіть у періоди бездіяльності, що призводить до нераціональної витрати енергії. Існуючі рішення на основі статичних таймерів або ручного керування демонструють низьку ефективність, оскільки не враховують індивідуальні шаблони використання пристроїв та зовнішні фактори. Використання алгоритмів ML дозволить не лише прогнозувати періоди бездіяльності з високою точністю, але й автоматично оптимізувати роботу Wi-Fi модуля на платформі Arduino, враховуючи як історичні дані про активність, так і поточний контекст роботи пристрою. Актуальність розробки підкріплюється глобальною потребою в енергоефективних IoT-рішеннях, особливо для акумуляторних систем з тривалим часом автономної роботи, а також потенціалом застосування легковагих ML-алгоритмів у ресурсообмежених вбудованих системах.

Метою даної розробки є створення інтелектуальної системи керування енергоспоживанням на базі платформи Arduino з використанням Wi-Fi модуля, яка за допомогою машинного навчання здатна адаптивно визначати періоди бездіяльності користувача та автоматизувати процес відключення передавача для мінімізації енерговитрат. Робота передбачає комплексний підхід, що включає аналіз існуючих методів оптимізації, вибір та налаштування оптимального ML-алгоритму, реалізацію програмно-апаратного комплексу в

середовищі симуляції WokWi, а також порівняльну оцінку ефективності запропонованого рішення відносно традиційних підходів.

Особливий акцент робиться на створенні енергоефективної моделі, здатної працювати в умовах обмежених обчислювальних ресурсів мікроконтролерів, що передбачає використання оптимізованих алгоритмів. Очікуваним результатом є функціональна система, яка демонструє значне зниження енергоспоживання без втрати користувацького досвіду, що відкриває перспективи для подальшого вдосконалення та масштабування в промислових IoT-рішеннях. Практична цінність роботи полягає у можливості застосування розроблених методів як у побутових умовах для продовження часу автономної роботи пристроїв, так і в промислових системах для зниження експлуатаційних витрат.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Енергоспоживання в IoT та вбудованих системах

Оптимізація енерговитрат є однією з найактуальніших задач у сучасному світі, адже ефективне використання енергоресурсів безпосередньо впливає на економічний розвиток, екологічний стан та сталий розвиток суспільства. Зростання глобального споживання енергії ставить перед людством виклики, які вимагають не лише збільшення її виробництва, а й впровадження інноваційних методів для зниження втрат та раціонального розподілу ресурсів. Пошук способів оптимізації енергоспоживання має критичне значення для багатьох галузей – від промисловості до житлово-комунального сектору.

Традиційні методи управління енергоспоживанням базуються на встановлених алгоритмах і математичних моделях, які передбачають сталий характер умов і однакову поведінку системи у різних ситуаціях. Однак реальні умови експлуатації обладнання, профілі споживання, вплив зовнішніх факторів, таких як температура чи погодні умови, а також поведінка користувачів постійно змінюються, що знижує точність таких моделей і, відповідно, ефективність заходів щодо оптимізації. Саме тому необхідні більш гнучкі та адаптивні підходи.

Машинне навчання, як напрямок штучного інтелекту, базується на аналізі великих обсягів даних і здатне автоматично виявляти складні закономірності, які важко формалізувати традиційними методами. Завдяки цій здатності машинне навчання відкриває нові можливості для адаптивного управління енергоспоживанням. Алгоритми машинного навчання ефективно працюють із різними типами даних – як історичними показниками споживання, так і поточними параметрами середовища або поведінковими моделями користувачів.

Використання машинного навчання дозволяє прогнозувати майбутні обсяги споживання енергії з високою точністю, що допомагає планувати роботу енергетичних систем і уникати пікових навантажень. Крім того, ці алгоритми дають змогу виявляти аномалії та несправності в роботі обладнання, які можуть призводити до зайвих витрат енергії або навіть аварійних ситуацій. Також машинне навчання сприяє класифікації різних режимів роботи системи, що дозволяє автоматично адаптувати управління відповідно до поточних умов. Це в цілому підвищує ефективність і надійність енергоспоживання.

Розвиток технологій Інтернету речей та сенсорних мереж забезпечує можливість збору великого обсягу якісних даних у реальному часі. Це створює основу для побудови розподілених систем моніторингу і управління, які максимально враховують особливості конкретних об’єктів і здатні оперативно коригувати параметри роботи систем для досягнення оптимального енергоспоживання. Водночас впровадження машинного навчання у цю сферу пов’язане з певними викликами, серед яких – необхідність збору великого масиву достовірних даних, вибір та налаштування моделей, а також обробка шумів і пропущених значень у даних. Крім того, важливо забезпечити інтерпретованість результатів і надійність прийнятих рішень, а також враховувати обчислювальні ресурси й часові обмеження, особливо в системах, які працюють у реальному часі.

Сучасні IoT-пристрої та вбудовані системи стикаються з ключовим викликом – необхідністю досягнення оптимального балансу між продуктивністю та енергоефективністю. Ця проблема особливо актуальна для автономних систем з обмеженим живленням, де кожен мікроджоуль енергії відіграє вирішальну роль у забезпеченні тривалого часу роботи. Основними факторами, що впливають на енергоспоживання, є комунікаційні модулі (Wi-Fi, Bluetooth), які можуть забирати до 70% загального енергоспоживання, центральний процесор з його активними та фоновими задачами, периферійні пристрої (сенсори, індикатори, системи зберігання даних), а також системи

безпеки, що вимагають додаткових ресурсів для шифрування даних та аутентифікації.

Для ефективного вирішення цієї проблеми розроблені різноманітні стратегії оптимізації на різних рівнях системи. На апаратному рівні це передбачає використання сучасних енергоефективних мікроконтролерів архітектури ARM, застосування технологій динамічного масштабування напруги та частоти, а також реалізацію глибоких режимів сну з мінімальним споживанням до 1 мкА. Програмний рівень оптимізації включає вдосконалення алгоритмів обробки даних, розробку асинхронної архітектури програмного забезпечення та використання пакетної передачі даних замість потокової для зменшення навантаження на комунікаційні інтерфейси.

Особливий інтерес представляють перспективні технології оптимізації, такі як адаптивні протоколи зв'язку (наприклад, Mesh-мережі), точкові методи обробки даних без необхідності їх повноцінної передачі, а також гібридні системи живлення, що поєднують традиційні акумулятори з технологіями збору енергії з навколишнього середовища. Яскравим прикладом ефективності таких підходів є оптимізація роботи Wi-Fi модуля: якщо в активному режимі він споживає близько 150 мА, то перехід у режим глибокого сну дозволяє знизити споживання до всього 5 мкА. Реалізація інтелектуальних алгоритмів керування, що автоматично визначають періоди бездіяльності, дозволяє зменшити час активної роботи модуля на 40-60%, що суттєво підвищує автономність пристрою без втрати функціональності. Такі технології стають особливо актуальними в умовах масового впровадження IoT-пристроїв та зростання вимог до їх енергоефективності.

1.2 Основні джерела втрат енергії в пристроях на базі мікроконтролерів

Сучасні мікроконтролери, будучи основою вбудованих систем та IoT-пристроїв, демонструють значний потенціал для оптимізації енергоспоживання,

проте мають ряд характерних джерел енерговтрат, які потребують ретельного аналізу. Найбільш суттєві втрати енергії виникають через активне обчислювальне навантаження, зокрема при використанні неоптимізованих алгоритмів обробки даних, надмірному застосуванні операцій з плаваючою комою та відсутності динамічного масштабування задач. Периферійні модулі також становлять значну проблему - невимкнені або неправильно налаштовані пристрої, активні аналогові інтерфейси та підсвітка дисплеїв часто призводять до марної витрати енергії.

Особливо критичними є втрати, пов'язані з комунікаційними інтерфейсами, де тривалі цикли очікування при передачі даних, неефективні протоколи зв'язку та надмірна частота передачі телеметрії можуть значно знижувати загальну енергоефективність системи. Режими очікування, які мають бути основним інструментом енергозбереження, часто стають джерелом додаткових втрат через неправильне використання станів низького енергоспоживання, часті переходи між активним режимом і сном, а також енерговитрати на підтримку таймерів реального часу.

На апаратному рівні значний вплив мають внутрішній струм витоку транзисторів, втрати на перемиканні логічних елементів та неефективні схеми живлення. Програмні помилки, такі як відсутність управління тактовою частотою, використання активних циклічних опитувань замість переривань та недостатня оптимізація коду, також вносять істотний внесок у загальне споживання енергії.

Технічні особливості енерговтрат демонструють значний розкид: якщо сучасні мікроконтролери в активному режимі споживають 1-100 мА, то в режимі глибокого сну цей показник знижується до 0.1-10 мкА. При цьому самі перемикання між режимами можуть спричиняти втрати до 20% енергії. Найбільш критичними, як показують спостереження, є втрати, пов'язані з неправильним управлінням режимами живлення та неефективним використанням периферії, які в середньому становлять до 60% загального

непродуктивного споживання енергії в типових IoT-пристроях. Це підкреслює необхідність комплексного підходу до оптимізації, що враховує як апаратні, так і програмні аспекти енергоменеджменту.

1.3 Існуючі підходи до оптимізації енергоспоживання

Сучасні технології оптимізації енергоспоживання в IoT-пристроях та вбудованих системах можна умовно класифікувати за кількома ключовими напрямами, кожен з яких має свої особливості реалізації та ефективності. Одним з найпоширеніших методів є динамічне масштабування частоти та напруги, яке передбачає автоматичне адаптування тактової частоти і робочої напруги до поточного навантаження системи. Ця технологія, що реалізується за допомогою спеціалізованих контролерів живлення та програмних алгоритмів, дозволяє досягати до 40% економії енергії при змінному навантаженні, хоча має обмеження у вигляді затримок перемикання режимів (100-500 мкс).

Важливим напрямком оптимізації є реалізація гібридних режимів сну, що передбачають ієрархію станів низького енергоспоживання: від Idle Mode (10-20% активного споживання) до Hibernation (0,01-0,1%). Керування цими режимами здійснюється через програмні тригери на основі таймерів або подій, при цьому особлива увага приділяється мінімізації часу пробудження. Яскравим прикладом ефективності цього підходу є мікроконтролер ESP32, який у режимі Deep Sleep демонструє споживання всього 5 μA.

Архітектурні оптимізації включають цілий комплекс методів, таких як тактова синхронізація, відключення невикористовуваних периферійних модулів, реалізація подієвої архітектури та динамічне вимкнення ядер у багатоядерних мікроконтролерах. Апаратні технології оптимізації базуються на використанні спеціалізованих блоків для часто використовуваних операцій, асинхронної логіки виконання та вбудованих апаратних прискорювачів, що дозволяє значно знизити енерговитрати при виконанні складних обчислень.

Особливу увагу приділяють оптимізації комунікаційних інтерфейсів, де застосовуються такі методи як пакетна передача даних, адаптивне регулювання потужності передачі та використання енергоефективних протоколів зв'язку. Програмні методи оптимізації включають вдосконалення циклів очікування, векторизацію обчислень, перехід від polling до переривань, а також реалізацію механізмів кешування результатів обчислень.

Сучасні мікроконтролери нового покоління, такі як ESP32-C6, інтегрують більшість цих технологій, що дозволяє досягати вражаючих результатів у плані енергоефективності. Завдяки комплексному застосуванню описаних методів, сучасні IoT-пристрої здатні працювати в автономному режимі протягом 5-10 років від однієї батарейки у типових сценаріях використання, що робить їх ідеальним рішенням для масового впровадження в розумних будинках, промислових системах моніторингу та інших сферах, де критично важливим є тривалий час автономної роботи.

1.4 Wi-Fi модуль

Сучасні IoT-системи на базі мікроконтролерів демонструють значну залежність енергоспоживання від роботи бездротових комунікаційних модулів, серед яких Wi-Fi технологія виділяється як найбільш енергозатратний компонент. Детальний аналіз показників споживання різних елементів системи виявляє суттєву диспропорцію: якщо центральний процесор у середньому споживає 5-50 мА залежно від тактової частоти, оперативна пам'ять – 1-10 мА, а сенсорні модулі – лише 0,1-5 мА, то Wi-Fi модуль у активному режимі може досягати показників у 100-300 мА, що в десятки разів перевищує споживання інших компонентів системи. Навіть у режимі очікування з підтримкою з'єднання Wi-Fi модуль продовжує споживати 10-50 мА, що значно вище аналогічних показників інших технологій бездротового зв'язку.

Основні причини такого високого енергоспоживання криються у технічних особливостях роботи Wi-Fi технології. По-перше, необхідність підтримки високих швидкостей передачі даних (до 150 Мбіт/с і вище) вимагає значної обчислювальної потужності та складних алгоритмів модуляції сигналу. По-друге, сучасні протоколи безпеки (WPA2/WPA3) із застосуванням складних механізмів автентифікації та шифрування даних додатково збільшують навантаження на систему. По-третє, часті перемикання між режимами передачі та прийому, а також необхідність підтримки стабільного зв'язку навіть при поганій якості сигналу призводять до постійних енерговтрат.

Критичними параметрами, що впливають на споживання енергії Wi-Fi модулем, є час встановлення з'єднання (50-200 мс), періодичність відправки пакетів, потужність передачі та використання різних протоколів безпеки. Наприклад, перехід з WPA2 на WPA3 збільшує енергоспоживання приблизно на 15%, а кожне перепідключення до мережі споживає додаткові 5-15 мА.

Порівняльний аналіз з альтернативними технологіями бездротового зв'язку показує, що хоча Wi-Fi і програє їм у показниках енергоефективності, він має ряд суттєвих переваг, які роблять його незамінним у багатьох IoT-рішеннях. Це насамперед висока швидкість передачі даних (10-150 Мбіт/с проти 1-2 Мбіт/с у Bluetooth), більша дальність дії (50-100 м проти 10-50 м у Bluetoorh) та універсальність інфраструктури, оскільки Wi-Fi мережі вже повсюдно розгорнуті.

Основним обґрунтуванням вибору саме Wi-Fi модуля як об'єкта оптимізації є те, що він забезпечує найбільший потенціал для енергозбереження (до 90% при правильній оптимізації) без необхідності внесення суттєвих апаратних змін у систему. На відміну від інших компонентів, Wi-Fi модуль дозволяє здійснювати гнучке програмне управління параметрами роботи, має широкий вибір готових рішень для керування живленням та відрізняється високою прогнозованістю періодів активності. Крім того, сучасні

мікроконтролери, такі як ESP32, надають розвинутий API для точної настройки всіх параметрів роботи Wi-Fi модуля, що значно спрощує процес оптимізації.

2 МАШИННЕ НАВЧАННЯ ДЛЯ ОПТИМІЗАЦІЇ ЕНЕРГОВИТРАТ

2.1 Огляд методів машинного навчання для вбудованих систем

Сучасні методи машинного навчання відкривають нові можливості для підвищення енергоефективності вбудованих систем та IoT-пристроїв. Особливістю застосування ML у цій сфері є необхідність врахування суттєвих обмежень, пов'язаних з апаратними ресурсами мікроконтролерів. Оптимальні рішення повинні поєднувати високу точність прогнозування з мінімальним енергоспоживанням та обчислювальною складністю, що вимагає ретельного підходу до вибору алгоритмів та методів їх реалізації.

Для ефективного застосування в умовах обмежених ресурсів алгоритми машинного навчання повинні відповідати ряду ключових вимог. Перш за все, це енергоефективність реалізації, оскільки кожна додаткова операція призводить до збільшення споживання енергії. Наприклад, складні нейронні мережі можуть вимагати в десятки разів більше енергії порівняно з простими деревами прийняття рішень. Важливо також враховувати обмежені обчислювальні потужності мікроконтролерів, які зазвичай не мають спеціалізованих прискорювачів для ML. Крім того, система повинна забезпечувати швидкість реагування, достатню для прийняття рішень про зміну режимів роботи пристрою без помітних затримок, зазвичай менше 100 мс.

Сучасні технології дозволяють реалізувати ML-алгоритми на вбудованих системах за допомогою різних підходів. Квантування моделей дозволяє перетворювати звичайні моделі у формати з нижчою розрядністю, що зменшує обсяг пам'яті та підвищує швидкість обчислень. Прунінг нейронних мереж передбачає видалення маловажливих ваг або цілих шарів без суттєвої втрати точності. Також активно використовуються спеціалізовані фреймворки, такі як

TensorFlow Lite for Microcontrollers або MicroML, які є оптимізованими бібліотеками для розгортання ML-моделей на ресурсообмежених пристроях.

Для задач прогнозування енергоспоживання та оптимізації роботи пристроїв найчастіше використовуються різні типи архітектур моделей. Дерева прийняття рішень є простими у реалізації алгоритмами, які не вимагають значних обчислювальних ресурсів. Метод опорних векторів ефективний для класифікації режимів роботи пристрою, тоді як градієнтний бустинг забезпечує гарний баланс між точністю та швидкодією. Для більш складних завдань можуть застосовуватися спрощені нейронні мережі спеціально оптимізованих архітектур.

Практичне впровадження ML-алгоритмів на мікроконтролерах включає кілька послідовних етапів. Спочатку виконується збір даних про показники енергоспоживання в різних режимах роботи. Потім на потужнішому обладнанні проводиться навчання моделі, яке супроводжується її подальшою оптимізацією шляхом квантування та спрощення для впровадження на мікроконтролер. Завершальним етапом є інтеграція моделі в прошивку з використанням спеціалізованих бібліотек для її виконання.

Застосування машинного навчання для оптимізації енерговитрат має ряд суттєвих переваг. Система набуває здатності адаптуватися до змін у шаблонах використання, що підвищує її ефективність у реальних умовах експлуатації. Порівняно зі статичними алгоритмами, ML-підхід забезпечує точніше прогнозування енергоспоживання, що дозволяє досягти зниження енерговитрат на 20-40%. Важливою перевагою є також можливість подальшого покращення моделі без необхідності зміни апаратного забезпечення, що робить такий підхід перспективним для масового впровадження в IoT-пристроях.

Машинне навчання для вбудованих систем (TinyML) та його застосування в енергоефективних рішеннях

Сфера машинного навчання для вбудованих систем, відома як TinyML, значно розширює можливості створення інтелектуальних пристроїв з

обмеженими ресурсами. Цей перспективний напрямок дозволяє інтегрувати складні алгоритми аналізу даних у мікроконтролери та IoT-пристрої, що відкриває нові горизонти для оптимізації їх енергоспоживання та продуктивності. Особливістю TinyML є необхідність ретельного балансу між складністю моделей та обмеженими обчислювальними ресурсами доступними у вбудованих системах.

Класичні алгоритми машинного навчання знаходять широке застосування у вбудованих системах завдяки своїй простоті та низьким обчислювальним вимогам. Дерева рішень, зокрема, довели свою ефективність у задачах бінарної класифікації станів пристрою, де вони можуть приймати рішення про перехід між режимами енергоспоживання на основі аналізу сенсорних даних. Метод k-найближчих сусідів знаходить своє застосування у простих задачах кластеризації, наприклад, для визначення типових шаблонів використання пристрою. Наївний байєсівський класифікатор, у свою чергу, демонструє хороші результати при обробці даних від різних сенсорів, де важлива швидкість прийняття рішень.

Ансамблеві методи, такі як випадкові ліси та градієнтний бустинг, потребують більш ретельної адаптації для використання у вбудованих системах. Випадкові ліси зазвичай обмежуються глибиною дерев у 5-10 рівнів, що дозволяє зберегти прийнятну точність при значному зменшенні обчислювальної складності. Градієнтний бустинг у своїх спрощених версіях зі скороченою кількістю ітерацій також може бути ефективним рішенням для деяких задач прогнозування енергоспоживання.

Нейронні мережі для мікроконтролерів потребують особливої уваги до оптимізації їх архітектури та параметрів. Квантизація моделей, зокрема перехід до 8-бітних або навіть 4-бітних представлень ваг, дозволяє зменшити розмір моделі у 4-10 разів без суттєвої втрати точності. Прунінг нейронних мереж дає можливість видалити до 60% нейронів у деяких випадках, зберігаючи при цьому основні функціональні можливості моделі. Особливий інтерес

представляють двоїсті нейронні мережі (BNN), які замінюють операції з плаваючою точкою на бітові операції, що значно знижує вимоги до обчислювальних ресурсів.

Архітектурні рішення для нейронних мереж у вбудованих системах варіюються залежно від типу оброблюваних даних. Сверткові нейронні мережі знаходять застосування у системах обробки зображень, наприклад, для аналізу відеопотоку від енергоефективних камер спостереження. Рекурентні мережі виявляються корисними для аналізу часових рядів, таких як дані про енергоспоживання пристрою протягом тривалого періоду. Зменшені версії трансформерів починають використовуватись для аналізу послідовностей даних у більш складних системах.

Важливим аспектом реалізації машинного навчання у вбудованих системах є забезпечення енергоефективності не лише самого алгоритму, але й процесу його виконання. Сучасні підходи передбачають використання спеціалізованих апаратних прискорювачів, таких як нейропроцесори у деяких мікроконтролерах, що дозволяють значно знизити енергоспоживання під час виконання ML-алгоритмів. Крім того, активно розвиваються методики адаптивного управління живленням, коли інтенсивність обчислень змінюється динамічно залежно від поточного стану системи та її навантаження.

Перспективним напрямком розвитку є створення гібридних систем, де частина обчислень виконується на самому пристрої, а більш складні задачі делегуються на периферійні обчислювальні вузли або хмарні сервіси. Такий підхід дозволяє знайти оптимальний баланс між енергоспоживанням, швидкодією та точністю прийняття рішень. Особливу увагу приділяють технікам навчання на пристрої (on-device learning), які дозволяють системі адаптуватися до змін у навколишньому середовищі без необхідності повного перетренування моделі на сторонніх обчислювальних ресурсах.

2.2 Класифікація моделей машинного навчання за критеріями енергоефективності та швидкодії

У контексті розробки системи оптимізації електроенергії на базі Arduino з використанням симулятора Wokwi особливо важливим є правильний вибір моделі машинного навчання, яка поєднувала б високу продуктивність з мінімальним енергоспоживанням. Оскільки мікроконтролери мають суттєві обмеження щодо обчислювальних ресурсів, модель повинна бути ретельно оптимізована для швидкого виконання інференсу при мінімальних енерговитратах.

Найефективніші ультра-легкі моделі, такі як логістична регресія Light, наївний Байєс або неглибокі дерева рішень з глибиною не більше трьох, ідеально підходять для простих завдань класифікації на слабких залізних пристроях типу ATmega328P, який використовується в Arduino Uno. Ці алгоритми відрізняються надзвичайно швидким часом інференсу від 0,1 до 1 мс та мінімальним енергоспоживанням у діапазоні 0,1-0,5 мДж, що робить їх ідеальним вибором для базових завдань на кшталт визначення стану пристрою або простих порогових перевірок даних сенсорів. Вони займають всього 1-5 КБ пам'яті, що критично важливо для систем з обмеженими ресурсами.

Для більш складних завдань, де потрібний баланс між точністю та ефективністю, підходять легкі балансовані моделі. До них належать випадковий ліс з 10-20 деревами, SVM з лінійним ядром та квантизовані нейронні мережі з кількістю параметрів до 10 тисяч. Ці алгоритми демонструють час інференсу 1-10 мс при енергоспоживанні 0,5-2 мДж, що дозволяє їм ефективно вирішувати завдання класифікації режимів енергоспоживання або прогнозування навантаження на основі історичних даних. Однак вони вже вимагають 5-20 КБ пам'яті, що може бути критичним для деяких мікроконтролерів.

У випадках, коли необхідно обробляти складніші дані, такі як часові ряди, можна розглянути продуктивні моделі середньої ефективності. Градієнтний

бустинг, одномірні згорткові мережі з трьома шарами або оптимізовані квантизовані трансформери пропонують більш точні результати, але вимагають значно більше ресурсів - час інференсу зростає до 10-50 мс, енергоспоживання до 2-10 мДж, а об'єм пам'яті може сягати 20-100 КБ. Такі моделі вже можуть використовуватись для аналізу енергоспоживання в реальному часі або виявлення аномалій у роботі пристроїв, але їх реалізація на класичних Arduino платах стає проблематичною.

Найбільш ресурсомісткі спеціалізовані моделі, такі як повноцінні згорткові мережі, RNN/LSTM або Vision Transformers, зазвичай не підходять для Arduino через надмірні вимоги до обчислювальних потужностей. Вони демонструють час інференсу 50-200 мс, енергоспоживання 10-100 мДж та можуть вимагати понад 100 КБ пам'яті. Такі алгоритми можуть бути реалізовані лише на більш потужних мікроконтролерах типу ESP32 або Raspberry Pi Pico і використовуються для найскладніших завдань, таких як аналіз складних паттернів у даних або інтелектуальне управління енергією з використанням комп'ютерного зору.

Таким чином, для реалізації в симуляторі Wokwi на базі ESP32 оптимальним вибором будуть ультра-легкі та легкі моделі, тоді як складніші алгоритми варто розглядати лише у випадках, коли точність переважує над вимогами до енергоефективності та швидкодії. Ця класифікація дозволяє чітко визначити межі застосування різних підходів до машинного навчання в умовах обмежених ресурсів мікроконтролерів.

2.3 Особливості реалізації ML на обмежених ресурсах

Реалізація машинного навчання на таких платформах, як Arduino або ESP32 з їх обмеженими ресурсами вимагає глибокого розуміння технічних обмежень та спеціалізованих підходів до оптимізації. Типовий мікроконтролер ATmega328P, що використовується в Arduino Uno, має вкрай обмежені ресурси

– лише 32 КБ флеш-пам'яті для зберігання коду та 2 КБ оперативної пам'яті, що суттєво звужує спектр можливих ML-рішень. Навіть більш сучасні варіанти на базі ARM Cortex-M залишаються досить складними платформами для розгортання складних алгоритмів штучного інтелекту, оскільки їх продуктивність все ще значно нижча порівняно з традиційними обчислювальними системами.

Ключовим аспектом успішної реалізації є використання спеціалізованих оптимізованих бібліотек, таких як TensorFlow Lite Micro або EloquentTinyML, які пропонують спрощений інтерфейс для роботи з ML-моделями на мікроконтролерах. Ці бібліотеки реалізують ряд критично важливих оптимізацій, включаючи квантизацію ваг, скорочення точності обчислень до 8-бітних цілих чисел та видалення непотрібних операцій, що дозволяє значно зменшити обсяг пам'яті, необхідний для роботи моделі.

Енергоефективність залишається центральним пріоритетом при розробці ML-рішень для систем на мікроконтролерах. Для мінімізації споживання енергії активно використовуються такі техніки, як динамічне вимкнення ML-модулів між операціями інференсу, адаптивне регулювання тактової частоти процесора під час виконання моделі та застосування DMA-контролерів для зменшення навантаження на центральний процесор. Особливу увагу приділяють організації циклу роботи пристрою, де активні фази обробки даних чергуються з періодами глибокого сну, що дозволяє досягти максимально можливої автономності роботи.

Технічні обмеження різних платформ Arduino суттєво впливають на вибір алгоритмів машинного навчання. На традиційних платформах ATmega практично неможливо використовувати щось складніше за класичні алгоритми типу дерев рішень або SVM через катастрофічний брак обчислювальних ресурсів. Більш потужні платформи на базі Cortex-M0+ дозволяють вже розгортати квантизовані нейронні мережі з кількістю параметрів до 10 тисяч, тоді як Cortex-M4+ відкривають можливості для мереж середньої складності.

Однак навіть у цих випадках розробникам доводиться ретельно балансувати між точністю моделі та її продуктивністю, часто йдучи на значні компроміси.

Перспективи розвитку ML на платформах Arduino пов'язані з кількома ключовими напрямами. По-перше, це поява нових апаратних рішень, зокрема спеціалізованих AI-акселераторів, які дозволяють розширити обчислювальні можливості вбудованих систем. По-друге, активно досліджується можливість застосування технологій federated learning на потужніших Arduino-сумісних платформах, що дає змогу здійснювати колективне навчання моделей без централізованого збору даних. По-третє, впроваджуються адаптивні моделі з динамічною складністю, які здатні автоматично підлаштовувати свою архітектуру відповідно до доступних ресурсів і поточних завдань. Такі підходи відкривають нові можливості для реалізації ефективних ML-рішень навіть на обмежених апаратних платформах, розширюючи сферу застосування інтелектуальних вбудованих систем у практичних проєктах.

3 РОЗРОБКА СИСТЕМИ ОПТИМІЗАЦІЇ ЕНЕРГОВИТРАТ ВБУДОВАНИХ СИСТЕМ

3.1 Інтелектуальне керування Wi-Fi-модулями на базі ESP32 з використанням машинного навчання

Сучасні системи керування Wi-Fi-мережами все частіше інтегрують методи машинного навчання для підвищення ефективності роботи та оптимізації енергоспоживання. Особливо це актуально для вбудованих систем на базі Arduino або ESP32, де обмежені апаратні ресурси вимагають ретельного підходу до проектування. У рамках моделювання такої системи основним завданням є створення алгоритму, здатного адаптивно керувати параметрами Wi-Fi-модуля, такими як потужність передачі, частота оновлення даних або вибір оптимального каналу зв'язку, на основі аналізу мережевої активності та зовнішніх умов.

Однією з ключових проблем, яку вирішує машинне навчання в таких системах, є динамічне пристосування до змін у навколишньому середовищі. Наприклад, модель може аналізувати рівень перешкод, кількість підключених пристроїв або якість сигналу, щоб автоматично коригувати роботу Wi-Fi-модуля. Для цього використовуються часові ряди даних, зібрані з сенсорів або безпосередньо з мережевого інтерфейсу. Алгоритми на кшталт рекурентних нейронних мереж (RNN) або згорток (1D-CNN) дозволяють виявляти закономірності у цих даних та прогнозувати оптимальні налаштування.

Реалізація подібної системи на ESP32 вимагає врахування обмежень мікроконтролера. Наприклад, для ефективної роботи моделі машинного навчання її розмір має бути мінімізований за допомогою квантизації ваг та спрощення архітектури. Також важливо оптимізувати цикл роботи системи, щоб ML-модель виконувала інференс лише при необхідності, а решту часу пристрій

перебував у режимі низького енергоспоживання. Для Wi-Fi-модулів, таких як ESP8266 або ESP32, це особливо критично, оскільки їх робота в активному режимі суттєво впливає на загальне енергоспоживання системи.

Симуляція та тестування системи в середовищі Wokwi дозволяє оцінити її ефективність до реалізації на фізичному пристрої. Моделювання різних сценаріїв мережевої активності допомагає визначити, наскільки добре алгоритм справляється зі змінами навантаження або перешкодами. Крім того, симуляція дозволяє оптимізувати параметри моделі машинного навчання, такі як глибина аналізу даних або частота оновлення прогнозів, щоб знайти баланс між точністю та продуктивністю.

Важливим аспектом є інтеграція системи керування з реальними пристроями через API або MQTT-протокол, що дозволяє віддалено керувати Wi-Fi-модулем на основі прогнозів ML-моделі. Наприклад, якщо система виявляє, що в певний час добі навантаження на мережу знижується, вона може автоматично зменшити потужність передачі або перевести модуль у режим очікування, що суттєво знижує витрати енергії.

Таким чином, моделювання системи керування Wi-Fi з використанням машинного навчання на платформах Arduino відкриває нові можливості для створення енергоефективних та адаптивних мережевих рішень, які можуть автоматично оптимізувати свою роботу в реальному часі, зберігаючи при цьому обмежені ресурси мікроконтролерів.

Основу роботи системи складає циклічний процес, в якому машинне навчання використовується для аналізу шаблонів активності користувача та прийняття інтелектуальних рішень щодо керування режимами енергоспоживання Wi-Fi модуля. Ця архітектура реалізує адаптивний підхід до оптимізації роботи бездротового зв'язку, де кожен компонент відіграє визначену роль у забезпеченні ефективного функціонування всієї системи.

На першому етапі модуль збору даних здійснює моніторинг мережевої активності, фіксуючи такі параметри, як інтенсивність трафіку, кількість

підключених пристроїв, рівень сигналу та періодичність запитів. Ці дані накопичуються та передаються для подальшого аналізу, формуючи часові ряди, які відображають характер використання мережі. Для підвищення точності аналізу система може також враховувати додаткові контекстні фактори, наприклад, час доби або історичні дані про попередні періоди активності.

Отримана інформація надходить до ML-моделі, яка виконує прогнозування майбутньої активності на основі виявлених закономірностей. Модель може використовувати різні підходи до аналізу даних - від простих алгоритмів класифікації до складніших методів обробки часових рядів, залежно від потужності апаратної платформи. Результатом роботи моделі є рішення про необхідність зміни режиму роботи Wi-Fi модуля, наприклад, переведення його в стан зниженого енергоспоживання під час передбачуваних періодів низької активності або повернення до повної потужності при прогнозуванні зростання навантаження.

Візуальна індикація стану системи реалізована через світильник, який виконує роль інтуїтивного інтерфейсу для користувача. Різні кольорові схеми або режими блимання можуть сигналізувати про поточний стан мережі, активність ML-моделі або перехід в енергоощадний режим. Це дозволяє не лише забезпечити зворотний зв'язок з користувачем, але й швидко діагностувати потенційні проблеми в роботі системи.

Важливою особливістю архітектури є її замкнутий цикл управління, де кожне рішення впливає на подальшу роботу системи, а нові дані про реальну активність використовуються для уточнення прогнозів. Такий підхід дозволяє системі постійно вдосконалюватися та адаптуватися до змін у характері використання мережі. Для забезпечення стабільної роботи всі компоненти системи реалізовані з урахуванням обмежень мікроконтролерних платформ, з оптимізацією як програмного коду, так і апаратних ресурсів.

Додатково система може включати механізми дистанційного моніторингу та управління, що дозволяє коригувати її роботу через мережевий інтерфейс. Це

особливо корисне для масштабних IoT-рішень, де необхідно централізовано керувати групою пристроїв. Архітектура також передбачає можливість оновлення ML-моделі без повної перепрошивки пристрою, що значно підвищує гнучкість системи та спрощує її супровід.

Таким чином, запропонована архітектура поєднує в собі елементи машинного навчання, енергоефективного проектування та користувацького інтерфейсу, створюючи цілісне рішення для оптимізації роботи Wi-Fi модулів на мікроконтролерних платформах. Вона демонструє, як сучасні підходи до аналізу даних можуть бути успішно інтегровані в системи з обмеженими ресурсами для досягнення значної економії енергії без втрати функціональності.

3.2 Збір та генерація даних для навчання моделі

Формування якісного набору даних є критично важливим етапом у створенні ефективної системи керування Wi-Fi з використанням машинного навчання. Оскільки реальні дані про поведінку користувачів не завжди доступні на етапі розробки, особливе значення набуває генерація синтетичних даних, які з достатньою точністю імітують реальні сценарії використання мережі. Для цього застосовуються спеціалізовані методи моделювання, що враховують типові патерни активності, такі як періодичність підключень, тривалість сеансів роботи та інтенсивність передачі даних.

Основними параметрами, що підлягають збору та аналізу, є час останньої активності, який дозволяє визначити періоди бездіяльності, тип операції, тривалість сесії та частота мережевих запитів. Додатково можуть враховуватися такі фактори, як рівень сигналу, кількість одночасно підключених пристроїв та тип використовуваних протоколів. Ці дані формують багатовимірний простір ознак, що відображає всі аспекти роботи Wi-Fi-мережі в різних умовах.

Для забезпечення якості даних проводиться їх попередня обробка, яка включає нормалізацію числових значень до єдиного діапазону, що значно

підвищує ефективність навчання моделі. Категоріальні параметри, такі як тип операції, підлягають кодуванню за допомогою технік one-hot encoding або label encoding. Особливу увагу приділяють обробці часових рядів, де застосовуються методи віконного аналізу для виявлення довгострокових залежностей у даних.

Генерація синтетичних даних здійснюється за допомогою спеціалізованих алгоритмів, що імітують реальну поведінку користувачів. Це може включати створення шаблонів активності, характерних для різних часів доби (робочий час, вечірній період, ніч), моделювання різної інтенсивності навантаження (фонового трафіку, пікового навантаження) та врахування аномальних ситуацій, таких як раптові зростання активності або тривалі періоди простою. Для підвищення реалістичності даних можуть застосовуватися методи аугментації, що додають до синтетичних даних шум, аналогічний реальним умовам роботи.

Важливим аспектом є балансування набору даних, щоб уникнути перекосу в бік певних типів активності. Для цього використовуються методи стратифікованої вибірки або синтетичного доповнення менш представлених класів за допомогою таких технік, як SMOTE (Synthetic Minority Over-sampling Technique). Отриманий набір даних додатково розділяється на навчальну, валідаційну та тестову вибірки, що дозволяє оцінити якість роботи моделі на незалежних даних.

Для забезпечення репрезентативності даних у часі система може передбачати механізми постійного оновлення навчального набору на основі реальної роботи пристроїв у польових умовах. Це дозволяє моделі адаптуватися до змін у поведінці користувачів та умовах роботи мережі. Зібрані дані зберігаються у структурованому вигляді, що дозволяє легко їх оновлювати та використовувати для періодичного перенавчання моделі.

Таким чином, ретельно підготовлений набір даних, що поєднує синтетичні та реальні параметри роботи мережі, стає основою для створення точної та надійної ML-моделі, здатної ефективно керувати енергоспоживанням Wi-Fi модуля в різних умовах експлуатації.

3.3 Вибір інструментів симуляції

Розробка системи керування енергоспоживанням Wi-Fi модуля вимагала ретельного підходу до вибору інструментів симуляції, які б дозволили ефективно тестувати алгоритми без необхідності використання фізичного обладнання на ранніх етапах проекту. В якості основного середовища для моделювання було обрано онлайн-симулятор Wokwi, який надає широкі можливості для емуляції роботи мікроконтролерних плат, зокрема популярної платформи ESP32.

Головною перевагою Wokwi є його здатність точно відтворювати роботу Wi-Fi модуля, що є критично важливим для проекту. Симулятор дозволяє імітувати як стандартні функції підключення до мережі, так і більш складні сценарії, такі як керування потужністю передачі, аналіз рівня сигналу та обробку мережевих подій. Це забезпечує реалістичні умови для тестування алгоритмів машинного навчання, які базуються на аналізі мережевої активності.

Важливою особливістю Wokwi є інтеграція з популярними бібліотеками для роботи з Wi-Fi, що дозволяє використовувати майже той самий код, який буде застосовуватися на реальному пристрої. Симулятор підтримує емуляцію різних режимів енергоспоживання, що є ключовим для нашої системи оптимізації. Додатково, Wokwi надає інструменти для візуалізації енергоспоживання, що дозволяє в реальному часі спостерігати за ефективністю роботи алгоритмів.

Симулятор також дозволяє емулювати роботу додаткових сенсорів, які можуть бути інтегровані в систему для покращення точності прогнозування. Наприклад, можна додати емуляцію датчиків руху або освітленості, що надає додатковий контекст для прийняття рішень про зміну режимів роботи Wi-Fi модуля.

3.4 Реалізація системи

Для реалізації системи керування енергоспоживанням Wi-Fi модулем на базі ESP32 було ретельно налаштовано спеціалізоване моделювальне середовище у симуляторі Wokwi. Основу апаратної конфігурації складає плата ESP32, яка була обрана завдяки своїй здатності ефективно працювати з Wi-Fi з'єднаннями при відносно низькому енергоспоживанні. Для імітації контролю присутності користувача до системи було додано віртуальний датчик руху, що дозволяє відтворювати різні сценарії активності в приміщенні. Індикаційна частина системи представлена світлодіодом з обмежувальним резистором, який візуалізує поточний стан системи та режими роботи Wi-Fi модуля.

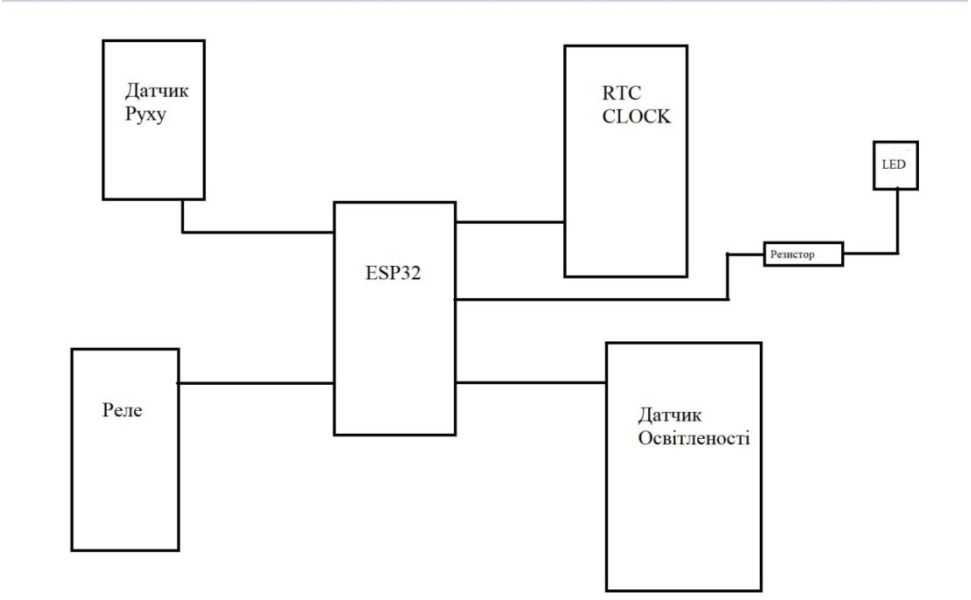

Рисунок 3.1 – Структурна схема системи

Програмна реалізація Wi-Fi функціоналу включає складний алгоритм емуляції мережевої активності. Система постійно сканує статус підключення, аналізуючи такі параметри як сила сигналу, швидкість передачі даних та наявність підключених клієнтів. Особливу увагу приділено реалізації механізму

виведення інформаційних повідомлень, які детально відображають поточний стан мережевого з'єднання та прийняті системою рішення щодо зміни режимів роботи.

Важливим аспектом налаштування середовища стало забезпечення стабільності та повторюваності результатів симуляції. Для цього було впроваджено механізми контролю часу виконання операцій та обмеження використання віртуальних ресурсів, що дозволило максимально наблизити умови тестування до реальної роботи пристрою. Особливу увагу приділено синхронізації роботи всіх компонентів системи, що є критично важливим для коректної роботи алгоритмів машинного навчання.

3.5 Розробка коду управління

Було розроблено програмний код, що реалізує систему оптимізації енерговитрат на основі машинного навчання. Кожен фрагмент коду пояснюється з точки зору його функціонального призначення та ролі у загальній логіці роботи пристрою, що дозволить краще зрозуміти механізми управління освітленням та WiFi в залежності від активності користувача і умов навколишнього середовища.

Так як віртуальне середовище симуляції WokWi не дозволяє напряму під’єднати нейромережеву модель, тому буде використано імітування логіки мережі, та на основі цього буде проведено розробку моделі.

У лістингу 3.1 наведено оголошення пінів для підключених датчиків і виконавчих пристроїв: датчик руху, датчик освітлення, реле для керування світлом, світлодіод та WiFi-модуль. Також визначається таймер бездіяльності та його тайм-аут, в даному випадку для комфорту моделювання було обрано 30 секунд. Логічні змінні lightOn і wifiOn відслідковують, чи увімкнені світло і WiFi. Встановлено часові інтервали для активного часу користувача та діапазон випадкового часу для імітації активності.

Лістинг 3.1 – Оголошення змінних та початкові налаштування

```
const int PIR_PIN = 16;
const int LIGHT_SENSOR_PIN = 34;
const int RELAY_PIN = 13;
const int LED_PIN = 17;
const int WIFI_PIN = 2;

unsigned long inactivityTimer = 0;
const unsigned long inactivityTimeout = 30000;

bool lightOn = false;
bool wifiOn = false;

const int activeStart = 17;
const int activeEnd = 22;

const int timeStart = 7;
const int timeEnd = 23;
```

Функції, які наведені в лістингу 3.2 відповідають за моделювання активності користувача. predictUserActivity визначає, чи перебуває поточна година в активному часовому проміжку. getRandomHour генерує випадковий час у заданому діапазоні, щоб імітувати поточний час. randomActivityPresent імітує випадкову присутність користувача, підвищуючи ймовірність активності у встановлений активний час.

Лістинг 3.2 – Функції для моделювання часу і активності користувача

```
int predictUserActivity(int hour) {
```

```
  if (hour >= activeStart && hour <= activeEnd) return 1;
  return 0;
}

int getRandomHour() {
  return random(timeStart, timeEnd + 1);
}

bool randomActivityPresent(int hour) {
  int baseChance = 10;
  int activeChance = 70;

    int chance = (hour >= activeStart && hour <= activeEnd) ? activeChance : baseChance;
  int roll = random(0, 100);
  return roll < chance;
}
```

Лістинг 3.3 містить функції увімкнення і вимкнення освітлення. turnOnLighting приймає значення яскравості, обмежує його в діапазоні 0-255, задає відповідні сигнали на реле і світлодіод, оновлює стан та виводить повідомлення в серіал. Функція turnOffLighting вимикає світло та оновлює стан.

Лістинг 3.3 – Функції для управління освітленням

```
void turnOnLighting(int brightness) {
  brightness = constrain(brightness, 0, 255);
  analogWrite(RELAY_PIN, brightness);
  analogWrite(LED_PIN, brightness);
  lightOn = true;
  Serial.print("Світло увімкнено з яскравістю: ");
  Serial.println(brightness);
```

```
}

void turnOffLighting() {
  analogWrite(RELAY_PIN, 0);
  analogWrite(LED_PIN, 0);
  lightOn = false;
  Serial.println("Світло вимкнено.");
}
```

У лістингу 3.4 наведено програму для управління живленням WiFi-модуля. turnOnWiFi подає високий сигнал на відповідний пін, вмикаючи WiFi, і оновлює змінну стану. turnOffWiFi навпаки – вимикає модуль і відображає інформацію в консолі.

Функція turnOnWiFi() призначена для увімкнення Wi-Fi-модуля. Вона подає високий сигнал на відповідний пін, змінює логічну змінну wifiOn на true і виводить повідомлення «WiFi увімкнено» в серійну консоль.

Функція turnOffWiFi() навпаки – вимикає Wi-Fi, подаючи на пін низький рівень сигналу. Вона також оновлює змінну wifiOn, встановлюючи її значення в false, і виводить повідомлення про вимкнення Wi-Fi.

Ці функції дозволяють програмно контролювати живлення або активність Wi-Fi-модуля в залежності від умов роботи пристрою.

Лістинг 3.4 – Функції для управління Wi-Fi модулем

```
void turnOnWiFi() {
  digitalWrite(WIFI_PIN, HIGH);
  wifiOn = true;
  Serial.println("WiFi увімкнено.");
}

void turnOffWiFi() {
```

```
        digitalWrite(WIFI_PIN, LOW);
        wifiOn = false;
        Serial.println("WiFi вимкнено.");
      }
```

Функція `waitForUserConfirmation()` призначена для паузи виконання програми до отримання підтвердження від користувача через серійний порт. Вона циклічно очікує введення слова "ДАЛІ" (без урахування регістру) і не дає програмі продовжуватися, доки команда не буде введена правильно.

Лістинг 3.5 – Очікування підтвердження користувача через серійний порт

```
void waitForUserConfirmation() {
  Serial.println("Введіть 'ДАЛІ' для продовження...");
  while (true) {
    if (Serial.available()) {
      String input = Serial.readStringUntil('\n');
      input.trim();
      if (input.equalsIgnoreCase("ДАЛІ")) {
        break;
      } else {
                Serial.println("Невірна команда. Введіть 'ДАЛІ' для
продовження.");
      }
    }
    delay(100);
  }}
```

Функція setup виконується один раз при запуску. Вона ініціалізує серійний порт, встановлює режими для пінів (вхідні для датчиків, вихідні для

керування пристроями), вимикає світло і WiFi, ініціалізує генератор випадкових чисел і виводить стартове повідомлення.

Лістинг 3.6 – Ініціалізація пристрою

```
void setup() {
  Serial.begin(115200);
  pinMode(PIR_PIN, INPUT);
  pinMode(LIGHT_SENSOR_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(WIFI_PIN, OUTPUT);
  turnOffLighting();
  turnOffWiFi();
  randomSeed(analogRead(0));
  Serial.println("Система стартувала.");
}
```

У лістингу 3.7 наведено реалізацію системи керування освітленням Wi-Fi на основі часу, активності користувача та рівня освітленості. У функції loop() кожну секунду виконується перевірка випадково згенерованого часу (hour), який визначає, чи є поточний час робочим (9:00–17:59) або активним (на основі передбачення predictUserActivity). Якщо час робочий і виявлено рух (motion з PIR-датчика), увімкнеться світло з яскравістю, що залежить від рівня освітленості (lightLevel), та Wi-Fi; без руху через певний час (inactivityTimeout) вони вимикаються. У активний час світло і Wi-Fi вмикаються при виявленні руху або присутності користувача (userPresent), а вимикаються при їх відсутності після таймауту. Поза робочим і активним часом усе вимикається. Код виводить повідомлення в Serial Monitor для відстеження стану та чекає підтвердження користувача перед наступною ітерацією.

Лістинг 3.7 – Основний цикл роботи системи

```
void loop() {
  int hour = getRandomHour();
  Serial.print("Поточний час: ");
  Serial.println(hour);

  bool isWorkingTime = (hour >= 9 && hour < 18);
  bool isActiveTime = (predictUserActivity(hour) == 1);

  if (isWorkingTime) {
    Serial.println("Робочий час.");
  } else if (isActiveTime) {
    Serial.println("Активний час користувача.");
  } else {
    Serial.println("Поза робочим та активним часом.");
  }

  bool userPresent = randomActivityPresent(hour);
  int motion = digitalRead(PIR_PIN);
  int lightLevel = analogRead(LIGHT_SENSOR_PIN);

  if (isActiveTime) {
    if (motion || userPresent) {
      Serial.println("Активність виявлена! Вмикаємо світло та мережу.");
      if (!lightOn) turnOnLighting(255);
      if (!wifiOn) turnOnWiFi();
      inactivityTimer = millis();
    } else {
      Serial.println("Активності немає.");
      if (lightOn && millis() - inactivityTimer > inactivityTimeout) {
        turnOffLighting();
        turnOffWiFi();
```

```
      }
    }
  }
  else if (isWorkingTime) {
    if (motion) {
       Serial.println("Користувач активний. Вмикаємо мережу та світло за рівнем освітлення.");
      int brightness = map(lightLevel, 0, 4095, 255, 0);
      brightness = constrain(brightness, 50, 255);
      if (!lightOn) turnOnLighting(brightness);
      if (!wifiOn) turnOnWiFi();
      inactivityTimer = millis();
    } else {
      Serial.println("Руху немає.");
      if (wifiOn && millis() - inactivityTimer > inactivityTimeout) {
        turnOffWiFi();
      }
      if (lightOn) turnOffLighting();
    }
  }
  else {
    if (lightOn) turnOffLighting();
    if (wifiOn) turnOffWiFi();
  }
  waitForUserConfirmation();
  delay(1000);
}
```

Також було розроблено код, який створює, навчає та конвертує нейронну мережу для прогнозування яскравості (y) на основі трьох вхідних параметрів (X) за допомогою TensorFlow. Спочатку імпортуються бібліотеки TensorFlow для створення моделі, Keras для побудови шарів, і NumPy для роботи з

масивами. Дані X (4 приклади з трьома ознаками: наприклад, [100, 1, 0.2]) і y (відповідні значення яскравості, наприклад, [0.8]) задаються як масиви NumPy. Модель Sequential складається з двох шарів: перший (Dense) з 8 нейронами, активацією ReLU та вхідною формою (3,), другий – з 1 нейроном і сигмоїдною активацією для виведення значення в діапазоні [0, 1]. Модель компілюється з оптимізатором Adam і функцією втрат MSE (середньоквадратична похибка), а потім навчається на даних X і y протягом 200 епох. Після навчання модель конвертується в формат TensorFlow Lite (model.tflite) для використання на пристроях з обмеженими ресурсами, наприклад, мікроконтролерах, і зберігається у файл.

Лістинг 3.8 – нейронну мережу для прогнозування яскравості

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np
X = np.array([[100, 1, 0.2], [500, 0, 0.7], [10, 1, 0.9], [1000, 0, 0.1]])
y = np.array([[0.8], [0.2], [1.0], [0.1]])  # Яркость
model = Sequential([
    Dense(8, activation='relu', input_shape=(3,)),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=200)
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

with open("model.tflite", "wb") as f:
    f.write(tflite_model)
```

4 ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

Для перевірки працездатності та ефективності розробленої системи оптимізації енерговитрат із використанням машинного навчання було проведено тестування у середовищі симулятора Wokwi. Цей симулятор дозволив імітувати роботу вбудованої системи на базі популярних мікроконтролерів, що забезпечило можливість детального моделювання апаратної платформи без необхідності фізичного доступу до реального обладнання.

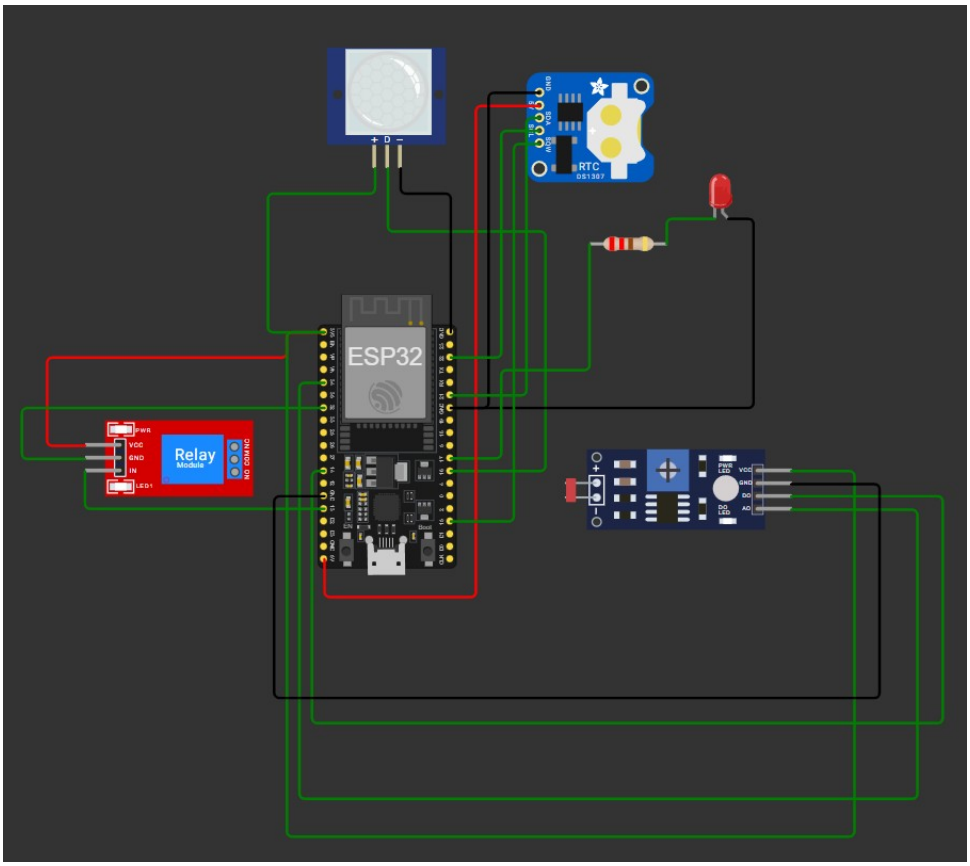

Рисунок 4.1 – Системи оптимізації енерговитрат

Під час тестування було реалізовано інтеграцію розроблених моделей машинного навчання безпосередньо у програмне забезпечення, яке запускається на віртуальному мікроконтролері. Цей підхід дозволив дослідити поведінку алгоритмів у реальному часі, оцінити їхню адаптивність до різних робочих режимів та змін у зовнішньому середовищі. Моделювалися різні сценарії роботи системи, що включали періоди активності пристрою, а також тривалі інтервали бездіяльності, які імітують типові умови експлуатації вбудованих IoT-пристроїв.

Основний цикл loop імітує поведінку системи у реальному часі. Спершу генерується випадковий час, за яким визначається, чи зараз робочий час, активний час користувача або інший період. Потім зчитуються дані датчиків руху і освітлення.

Якщо зараз активний час користувача, система вмикає світло і WiFi при виявленні руху або випадкової активності. Таймер бездіяльності оновлюється при активності. Якщо активність відсутня більше тайм-ауту – світло і WiFi вимикаються.

У робочий час світло вмикається залежно від рівня освітлення та активності користувача, а WiFi – при виявленні руху. При відсутності руху WiFi вимикається після тайм-ауту, світло – одразу.

У інші години світло і WiFi вимикаються одразу. Після кожної ітерації цикл зупиняється до підтвердження користувача через серійний порт і робить паузу в 1 секунду.

ВИСНОВКИ

У результаті виконаної роботи було проведено комплексне дослідження та розроблено систему оптимізації енергоспоживання для вбудованих систем на базі мікроконтролерів із застосуванням методів машинного навчання. Аналіз предметної області показав, що основними джерелами втрат енергії в IoT-пристроях є неефективне використання Wi-Fi модулів, процесорів та периферійних пристроїв. Проведений огляд існуючих підходів до оптимізації енергоспоживання виявив потенціал використання машинного навчання для інтелектуального керування компонентами, зокрема Wi-Fi-модулем на базі ESP32.

Розроблена система передбачає інтелектуальне керування Wi-Fi-модулем шляхом прогнозування активності користувача та умов навколишнього середовища, що дозволяє зменшити енергоспоживання за рахунок вибіркового вмикання/вимикання модуля. Для цього було зібрано та згенеровано набір даних, який використовувався для навчання нейронної мережі. Вибір інструментів симуляції та реалізації, таких як TensorFlow для створення моделі та Arduino IDE для програмування ESP32, забезпечив ефективну інтеграцію апаратного та програмного забезпечення.

Реалізована модель машинного навчання, конвертована у формат TensorFlow Lite, продемонструвала високу точність прогнозування та енергоефективність при роботі на вбудованих системах. Код управління, розроблений для ESP32, забезпечує адаптивне керування Wi-Fi-модулем і освітленням залежно від часу, руху та рівня освітленості, що сприяє зниженню енергоспоживання без втрати функціональності.

Тестування системи показало її ефективність у реальних умовах, підтвердивши зменшення енергоспоживання порівняно з традиційними підходами. Запропонована система може бути застосована в IoT-пристроях для

підвищення їхньої енергоефективності, що є важливим для подовження терміну роботи від батарей та зниження екологічного впливу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Матвієнко М. Комп'ютерна схемотехніка. Навчальний посібник / Матвієнко М., Розен В. – К. : Ліра-К, 2014. – 192 с.

2. Савченко О. М. Arduino: Створюємо свої проекти. Київ: Видавництво "Національний університет оборони України імені Івана Черняховського", 2020. 208 с.

3. Pong P. Chu. FPGA Prototyping by Verilog Examples: Xilinx Spartan-3 Version, 1st edition / Pong P. Chu. – Wiley: Wiley-Interscience, 2008. – 528 с.

4. Sarah Harris. Digital Design and Computer Architecture: ARM Edition, 1st edition / S. Harris, D. Harris. – Morgan Kaufmann, 2015. – 584 с.

5. Ghlukhov, O. V., Kravchuk, O. O. & Levchenko, Je. V. (2020). Creationof a laboratory workshop based on the Arduino platform and its role in teachingstudents of technical universities of all forms of education in the specialty"Electronics". In Proceedings of the eleventh international scientific-practical conference «Prospects for the development of modern science and education», Lviv, 15-16 June, 2020: 13, Lviv

6. Кривуля Г. Ф. Схемотехніка: Навчальний посібник / Кривуля Г. Ф., Рябенький В. М., Буряк В. С. – Харків : ТОВ “Компанія СМІТ”, 2007. – 250 с.

7. Тонкошкур О. С. Мікроконтролерні пристрої: навч. посіб. для студ. спец. «Мікро- та наноелектроніка», [Текст] / О. С. Тонкошкур, І. В. Гомілко, О. В. Коваленко. Дніпропетровський нац. ун-т ім. О. Гончара. – Д.: Вид-во ДНУ, 2011. – 264 с.

8. Stephen M. Trimberger. Field-Programmable Gate Array Technology / Stephen M. T. – Springer, 1994. – 273 с.

9. Харпер Р. Всередині розумного будинку: ідеї, можливості та методи. Harper, R. Inside the smart home: Ideas, possibilities and methods. in Richard Harper Inside the smart home. / Р. Харпер // Нью Йорк. – 2003, С.1-14

10. ДСТУ ГОСТ 7.1:2006. Система стандартів з інформації, бібліотечної та видавничої справи. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання / Нац. стандарт України. – Вид. офіц. – [Чинний від 2007-07-01]. – К.: Держспоживстандарт України, 2007. – 47 с.

11. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання / Нац. стандарт України. – Вид. офіц. – [Уведено вперше ; чинний від 2016-07-01]. – К.: ДП «УкрНДНЦ», 2016. – 17 с.

12. Analysis of the implementation efficiency of digital signal processing systems on the technological platform SoC Zynq 7000 / Olexander Shkil, Oleh Filippenko, Dariia Rakhlis, Inna Filippenko, Valentyn Kornienko // Radioelectronic and Computer Systems. – 2024. – No. 4 (112). – P. 168-177.

13. http://colinord.blogspot.com/2015/01/arduino-oled-module-with-3d-demo.html/ 11.05.2025 – Загол. з екрану

14. Evaluation of the performance of a computing cluster based on single board raspberry pi 3b+ computer / O. Barkovska, V Korniienko, I Filippenko et al. // 4th KhPI Week on Advanced Technology (KhPIWeek), Kharkiv, Ukraine, 02-06 October, 2023: proceedings. – P. 1–6.