# Звіт подібності

## Метадані

| Назва організації | підрозділ |
|---|---|
| **Ivano-Frankivsk National Technical University of Oil and Gas** | **Каф. ІТТС** |

Заголовок

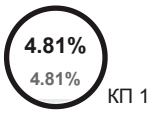
**2025_Бардюк І.І._ФІТ_ІТТС_АКСм-24-1**

| Автор | Науковий керівник / Експерт |
|---|---|
| **Бардюк І. І.** | **Паньків Ю. В.** |

| Кількість слів | Кількість символів | Дата звіту | Дата редагування | ІД документу |
|---|---|---|---|---|
| **11369** | **88542** | **12/24/2025** | **---** | **332968117** |

## Обсяг знайдених подібностей

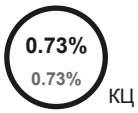
Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.




КП 1

**11369**

Кількість слів




КЦ

**88542**

Кількість символів

## Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

| | | |
|---|---|---|
| Заміна букв | ß | 3 |
| Інтервали | A→ | 0 |
| Мікропробіли | · | 0 |
| Білі знаки | ß | 54 |
| Парафрази (SmartMarks) | a | 22 |

## Джерела

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

| ПОРЯДКОВИЙ НОМЕР | НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ) | КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|
| 1 | https://nung.edu.ua/sites/default/files/2023-04/%D0%9C%D0%A0_%D0%9AI_%D0%9E%D1%84%D0%BE%D1%80%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_2023.pdf | 29 0.26 % |
| 2 | https://community.home-assistant.io/t/build-a-water-meter-with-esphome-and-proximity-sensor-no-soldering-required/387686?page=6 | 24 0.21 % |
| 3 | http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-fa833a00-1bd2-4d4c-a2f4-979e3afcc086 | 24 0.21 % |

| | | |
|---|---|---|
| 4 | http://repository.ub.ac.id/11813/8/Daftar%20Pustaka.pdf | 22 0.19 % |
| 5 | МР_Бардюк_І_І_АТм-23-1<br>[redacted] 12/20/2024<br>Ivano-Frankivsk National Technical University of Oil and Gas (Каф. АТ) | 22 0.19 % |
| 6 | 2024_Адамчак_М_І_ІІТ_ІТТС_АКСм_23_1<br>[redacted] 12/27/2024<br>Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС) | 22 0.19 % |
| 7 | https://dspace.dsau.dp.ua/bitstream/123456789/8977/1/%D0%92%D0%BE%D1%80%D0%BE%D0%BD%D0%B0%20%D0%9E.%20%D0%92..pdf | 19 0.17 % |
| 8 | https://nung.edu.ua/sites/default/files/2023-04/%D0%9C%D0%A0_%D0%9AI_%D0%9E%D1%84%D0%BE%D1%80%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_2023.pdf | 19 0.17 % |
| 9 | ФАЕТ_2022_171-2_Єнгуразов_Владислав<br>[redacted] 7/11/2024<br>Ukrainian national aviation university (Ukrainian national aviation university) | 18 0.16 % |
| 10 | https://ela.kpi.ua/bitstream/123456789/28097/1/Gogu_bakalavr.pdf | 18 0.16 % |

## з домашньої бази даних (1.06 %)

| ПОРЯДКОВИЙ НОМЕР | ЗАГОЛОВОК | КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|
| 1 | МР_Бардюк_І_І_АТм-23-1<br>[redacted] 12/20/2024<br>Ivano-Frankivsk National Technical University of Oil and Gas (Каф. АТ) | 87 (6) 0.77 % |
| 2 | 2024_Адамчак_М_І_ІІТ_ІТТС_АКСм_23_1<br>[redacted] 12/27/2024<br>Ivano-Frankivsk National Technical University of Oil and Gas (Каф. ІТТС) | 33 (3) 0.29 % |

## з програми обміну базами даних (0.78 %)

| ПОРЯДКОВИЙ НОМЕР | ЗАГОЛОВОК | КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|
| 3 | 2025_КІ_414_Підлубний_Б_О<br>[redacted] 6/12/2025<br>Ukrainian national aviation university (Фаховий коледж інженерії, управління та землевпорядкування Національного авіаційного університету) | 25 (2) 0.22 % |
| 4 | ФАЕТ_2022_171-2_Єнгуразов_Владислав<br>[redacted] 7/11/2024<br>Ukrainian national aviation university (Ukrainian national aviation university) | 18 (1) 0.16 % |
| 5 | Методика оцінки енергоефективності сенсорних мереж на етапі проєктування<br>[redacted] 6/19/2024<br>National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (РТФ, К-ра прикладної радіоелектроніки) | 12 (1) 0.11 % |
| 6 | Програмне забезпечення домашньої автоматизації для бездротової мережі сенсорів та актуаторів<br>[redacted] 5/25/2023<br>National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби") | 12 (1) 0.11 % |
| 7 | 2023_Б_АПОТ_КІУКІ-19-9_Деонега_О_В<br>[redacted] 5/31/2024<br>Kharkiv National University of Radio Electronics (Kharkiv National University of Radio Electronics) | 10 (1) 0.09 % |

| | | |
|---|---|---|
| 8 | БКР_Телюк_IP42<br>[redacted] 6/10/2025<br>National University "Lviv Politechnika" (NULP2) | 7 (1) 0.06 % |
| 9 | Програмне забезпечення для віддаленого моніторингу мікроклімату медичних приміщень<br>[redacted] 6/7/2022<br>National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби") | 5 (1) 0.04 % |

## з Інтернету (2.97 %)

| ПОРЯДКОВИЙ НОМЕР | ДЖЕРЕЛО URL | КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|
| 10 | **https://nung.edu.ua/sites/default/files/2023-04/%D0%9C%D0%A0_%D0%9AI_%D0%9E%D1%84%D0%BE%D1%80%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_2023.pdf** | **69 (4) 0.61 %** |
| 11 | https://community.home-assistant.io/t/build-a-water-meter-with-esphome-and-proximity-sensor-no-soldering-required/387686?page=6 | 34 (2) 0.30 % |
| 12 | https://ela.kpi.ua/bitstream/123456789/28097/1/Gogu_bakalavr.pdf | 28 (2) 0.25 % |
| 13 | http://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-fa833a00-1bd2-4d4c-a2f4-979e3afcc086 | 24 (1) 0.21 % |
| 14 | https://dspace.dsau.dp.ua/bitstream/123456789/8977/1/%D0%92%D0%BE%D1%80%D0%BE%D0%BD%D0%B0%20%D0%9E.%20%D0%92..pdf | 24 (2) 0.21 % |
| 15 | https://repo.nung.edu.ua/bitstreams/b7eb01e0-75a4-46bc-84ab-413631085fc2/download | 22 (2) 0.19 % |
| 16 | http://repository.ub.ac.id/11813/8/Daftar%20Pustaka.pdf | 22 (1) 0.19 % |
| 17 | https://network-journal.mpei.ac.ru/ru/33/4/3/article.htm | 17 (1) 0.15 % |
| 18 | https://community.home-assistant.io/t/esphome-and-bme680-strange-voc-values/790186 | 14 (2) 0.12 % |
| 19 | https://link.springer.com/chapter/10.1007/978-981-99-3315-0_53 | 12 (1) 0.11 % |
| 20 | https://ela.kpi.ua/bitstreams/7a9bcf26-68b0-4b67-a866-27163e111cae/download | 11 (1) 0.10 % |
| 21 | http://tesi.eprints.luiss.it/14917/2/menicocci-giuseppe-sintesi-2015.pdf | 11 (1) 0.10 % |
| 22 | https://ifagrarncol.at.ua/sylabus2020/1metodichni_rekomendacii_do_vikonannja_kvlifikacij.pdf | 11 (1) 0.10 % |
| 23 | https://repo.nung.edu.ua/bitstreams/9ea12e2c-289e-4d04-b02e-1c3ae19ecd00/download | 10 (1) 0.09 % |
| 24 | https://elar.khmnu.edu.ua/bitstreams/defb1e9b-4b87-4189-a5f8-aaaae0de600c/download | 10 (1) 0.09 % |
| 25 | https://sejarahjarkom.blogspot.com/ | 7 (1) 0.06 % |
| 26 | https://community.home-assistant.io/t/using-bme680-sensor-with-bsec/632060 | 6 (1) 0.05 % |
| 27 | https://dou.ua/forums/topic/42488/ | 6 (1) 0.05 % |

## Список прийнятих фрагментів

| ПОРЯДКОВИЙ НОМЕР | ЗМІСТ | КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ) |
|---|---|---|

МАГІСТЕРСЬКА РОБОТА МР.АКС-01.00.00.000 ПЗ Група АКСм-24 [1] **Бардюк** Ігор 2025

Івано-Франківський **національний технічний університет нафти і газу Міністерства освіти і науки України**
Факультет [2] **інформаційних технологій**
**Кафедра інформаційно-телекомунікаційних** технологій та систем

Бардюк Ігор Ігорович
**(прізвище, ім’я, по батькові)**

**УДК 681**

(індекс)

МАГІСТЕРСЬКА РОБОТА

Тема: Розроблення системи збору інформації з розподілених об'єктів з використанням ESPhome
(назва роботи)

_____ Магістр _____
**(назва освітньої програми)**

**174- Автоматизація, комп’ютерно-інтегровані технології та робототехніка _____**
**(шифр і назва спеціальності)**

_____І.І. Бардюк_____
**(підпис, ініціали та прізвище здобувача освітнього ступеня)**

**Науковий керівник** Паньків Ю.В., к. т. н., доц.
**(прізвище, ім’я, по батькові, науковий ступінь, вчене звання) Допущено до захисту Завідувач** кафедри
_____Заміховський **Л.М.**
**(підпис) (дата) (ініціали та прізвище)**

**Рецензент**

_____
**(посада) (підпис) (дата) (ініціали та прізвище)**
**Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело**

Івано-Франківськ - 2025

**Івано-Франківський національний технічний університет нафти і газу**

**Інститут інформаційних технологій**
**Кафедра інформаційно-телекомунікаційних технологій та систем**
Напрям підготовки освітньо-професійна

ЗАТВЕРДЖУЮ:
Зав. **кафедри ІТТС**
**Проф. Заміховський Л.М.**
“___” 2025 р.

ЗАВДАННЯ
НА ВИКОНАННЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТОВІ

Бардюку Ігору Ігоровичу
**(прізвище, ім’я, по-батькові)**
**1 Тема роботи: Розроблення системи** збору інформації з розподілених об'єктів з використанням ESPhome
Керівник роботи Паньків Ю.В., к. т. н., доц.
**(прізвище, ім’я, по-батькові, науковий ступінь, вчене звання) затверджені** наказом по університету від_____
**2 Термін** здачі студентом закінчної роботи **3 Вихідні дані до роботи** системи збору інформації з розподілених об'єктів
**4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)** Проаналізувати область теми роботи та постановити задачу проекту, розглянути сучасні рішення по методах збору даних з розподілених об’єктів, провести огляд обраної платформи, розглянути ефективність контроллерів у вирішенні поставленої задачі, провести технічну та програмну реалізацію проекта, зробити висновки з проведеної роботи.
5 **Перелік графічного матеріалу**
**Презентація у** Power Point
6 Дата видачі завдання 30.10.2025

Керівник
**(підпис) Завдання прийняв до** виконання_____
_____(підпис)_____

КАЛЕНДАРНИЙ **ПЛАН No з/п** Назва етапівмагістерської роботи **Термін виконання етапів роботи Примітка** 1 Аналіз предметної області та постановка задачі 19.11.25 - 23.11.25 Виконано

2 Огляд методів реалізації збору інформації з об’єктів 24.11.25 - 28.11.25 Виконано

3 Здійснити технічну та програмну реалізацій проекту 29.11.25 - 15.12.25 Виконано

4 Оформлення результатів роботи 16.12.25 - 20.12.25 Виконано

5 Розробка графічного матеріалу 21.12.25 - 28.12.25 **Виконано**

**Студент** **Бардюк І.І.**

**(Особистий підпис)** **(Розшифровка підпису)**

**Керівник** Паньків Ю.В.

(Особистий підпис) (Розшифровка підпису)


АНОТАЦІЯ

Темою дипломної роботи є розроблення системи збору інформації з розподілених об'єктів з використанням платформи ESPHome на базі мікроконтролерів ESP32 та ESP8266. Особливу увагу приділено аналізу сучасних підходів до побудови розподілених IoT-систем, включаючи промислові SCADA-системи, платформи розумного будинку та бездротові сенсорні мережі. Проведено комплексний аналіз апаратних платформ для IoT, зокрема мікроконтролерів Arduino, Raspberry Pi, ESP8266, ESP32 та STM32, з визначенням їх переваг та обмежень для різних сценаріїв застосування.Досліджено програмні рішення та фреймворки для розробки IoT-систем, включаючи Arduino Framework, ESP-IDF, MicroPython, Lua/NodeMCU, Tasmota та ESPHome, з детальним порівнянням їх характеристик за критеріями складності розробки, гнучкості, продуктивності та швидкості створення додатків. Проаналізовано протоколи передачі даних у розподілених системах, зокрема MQTT, HTTP/HTTPS, WebSocket, CoAP, AMQP, Zigbee, Z-Wave та LoRaWAN, визначено їх особливості, переваги та придатність для різних типів IoT-застосувань.
Практична цінність роботи полягає в можливості використання розробленої системи для моніторингу мікроклімату в житлових та офісних приміщеннях, контролю параметрів у теплицях та сховищах, спостереження за станом обладнання на виробництві, екологічного моніторингу.
Ключові слова: система збору інформації, розподілені об'єкти, ESPHome, ESP32, ESP8266, Інтернет речей, IoT, MQTT, бездротові сенсорні мережі, моніторинг, Home Assistant, мікроконтролери, датчики.



ABSTRACTS

The topic of the diploma thesis is the development of an information collection system for distributed objects using the ESPHome platform based on ESP32 and ESP8266 microcontrollers. Special attention is paid to the analysis of modern approaches to building distributed IoT systems, including industrial SCADA systems, smart home platforms, and wireless sensor networks. A comprehensive analysis of hardware platforms for IoT is carried out, including Arduino, Raspberry Pi, ESP8266, ESP32, and STM32 microcontrollers, with the identification of their advantages and limitations for various application scenarios.
Software solutions and frameworks for IoT system development are investigated, including the Arduino Framework, ESP-IDF, MicroPython, Lua/NodeMCU, Tasmota, and ESPHome, with a detailed comparison of their characteristics according to the criteria of development complexity, flexibility, performance, and application development speed. Data transmission protocols in distributed systems are analyzed, including MQTT, HTTP/HTTPS, WebSocket, CoAP, AMQP, Zigbee, Z-Wave, and LoRaWAN, and their features, advantages, and suitability for different types of IoT applications are determined.
The practical significance of the work lies in the possibility of using the developed system for microclimate monitoring in residential and office buildings, control of parameters in greenhouses and storage facilities, monitoring the condition of equipment in industrial environments, and environmental monitoring.
Keywords: information collection system, distributed objects, ESPHome, ESP32, ESP8266, Internet of Things, IoT, MQTT, wireless sensor networks, monitoring, Home Assistant, microcontrollers, sensors.



РЕФЕРАТ

Магістерська робота містить: 56 сторінок, 18 рисунків, 2 таблиці, 33 посилань на літературні джерела.
Об'єкт дослідження: процеси збору, передачі та обробки даних у розподілених системах моніторингу на основі бездротових сенсорних мереж та IoT-технологій.
Мета роботи: розроблення та реалізація розподіленої системи збору інформації з віддалених об'єктів на базі платформи ESPHome з використанням мікроконтролерів ESP32/ESP8266 для забезпечення ефективного моніторингу параметрів в режимі реального часу.
У магістерській роботі проведено дослідження процесів розроблення системи збору інформації з розподілених об’єктів з використання платформи ESPHome. Розглянуто сучасні підходи до побудови розподілених IoT-систем, проаналізовано архітектуру таких систем та особливості застосування мікроконтролерів сімейств ESP32 та ESP8266. Оцінено можливості платформи ESPHome для інтеграції сенсорів, передачі та обробки даних, а також її придатність для моніторингу параметрів у різних сферах застосування.
Основними завданнями роботи є:
1. Провести аналіз предметної області та постановка задачі.
2. Здійснити апаратну та програмну реалізацію системи.
3. Ознайомлення з охороною праці та безпеки в надзвичайних ситуаціях.
4. Провести висновки по спроектованій системі.


МІКРОКОНТРОЛЛЕР, МОНІТОРИНГ, ДАТЧИК, СИСТЕМА, ПОКАЗНИК, ОБ’ЄКТ

Паньків Ю.В.

Розробив **Перев. Н. Контр. Затв. М Р.А**
КС
м
-
0
1
.00.00.000 ПЗ
Розроблення системи збору інформації з розподілених об'єктів з використання ESPhome
ІФ
Н
ТУНГ,
А
КС
м-2
4
1
н
7
Аркушів
Арк.
Літ.
) ЗМІСТ
TOC \o "1-3" \h \z \u



(
1
**Арк.**
**Зм.**
**Підпис**
**Дата**
**No докум.**
**БР.АКС -**
**24**
**.00.00.000 ПЗ**
**Арк.**
**1**
)

ВСТУП

У зв’язку з швидким розвитком науки та технологій зростає потреба у сучасних інструментах, здатних забезпечувати надійний збір, передавання та аналіз інформації з розподілених об’єктів. У багатьох галузях - від промислових процесів і енергетики до екологічного моніторингу та систем «розумного» середовища - використання таких систем стає ключовою умовою ефективної роботи. Своєчасне отримання даних дозволяє оптимізувати керування, підвищувати безпеку, запобігати аварійним ситуаціям і приймати обґрунтовані рішення.
Швидкий прогрес сенсорної та обчислювальної техніки створив можливість масштабного впровадження розподілених систем моніторингу, що складаються з великої кількості автономних пристроїв. Сучасні сенсори стали компактними, енергоефективними та доступними, що дозволяє

розміщувати їх практично на будь-яких об’єктах - як стаціонарних, так і мобільних.
Метою даного дослідження є розроблення ефективної системи збору та обробки інформації з розподілених об'єктів на базі платформи ESPHome.
Об'єктом дослідження є система моніторингу віддалених об'єктів з використанням мікроконтролерів ESP та платформи ESPHome.
Предметом дослідження є технології конфігурування та інтеграції пристроїв на базі ESPHome.
Такі рішення відіграють важливу роль у побудові сучасних інфраструктур, оскільки дозволяють отримувати актуальні дані в реальному часі, аналізувати тенденції та підвищувати ефективність роботи різних процесів. Саме тому їх розроблення та дослідження залишаються актуальним завданням, що має практичне значення та широкий спектр можливих застосувань.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Огляд сучасних систем збору інформації з розподілених об'єктів

Сучасні системи збору інформації з розподілених об'єктів є невід'ємною частиною цифрової трансформації промисловості, сільського господарства, транспорту та побутової сфери. Ці системи забезпечують моніторинг параметрів навколишнього середовища, контроль стану обладнання, управління ресурсами та прийняття обґрунтованих рішень на основі зібраних даних [1].
Розподілені системи збору інформації можна класифікувати за кількома ключовими критеріями [2]:
1. За сферою застосування:
1. промислові системи моніторингу (scada, iiot);
2. системи розумного будинку та будівлі (smart home, building automation);
3. агропромислові системи моніторингу;
4. транспортні та логістичні системи;
5. екологічний моніторинг;
6. медичні системи віддаленого спостереження.
2. За архітектурою побудови:
1. централізовані системи з єдиним центром обробки даних;
2. децентралізовані системи з розподіленою обробкою;
3. гібридні системи з edge-computing;
4. хмарні рішення з віддаленим зберіганням даних.
3. За типом з'єднання:
1. дротові системи (ethernet, rs-485, modbus);
2. бездротові системи (wi-fi, bluetooth, lorawan, zigbee);
3. комбіновані системи.
INCLUDEPICTURE "https://www.researchgate.net/profile/Michael-Brodie/publication/220094963/figure/fig1/AS:339555996717064@1457967866767/Distributed-Object-Management.png" * MERGEFORMATINET

Рисунок 1.1 - Управління розподіленими об'єктами

Майбутні середовища обробки інформації складатимуться з величезної мережі гетерогенних, автономних та розподілених обчислювальних ресурсів, включаючи комп'ютери (від мейнфреймів до персональних), інформаційно-ємні програми, та дані (файли та бази даних). Ключовим завданням у цьому середовищі є забезпечення можливостей об'єднання цієї різноманітної колекції ресурсів в інтегровану розподілену систему, що дозволить гнучко комбінувати ресурси, а їхню діяльність координувати для вирішення складних нових вимог до обробки інформації [2].
Промислові системи збору інформації традиційно базуються на SCADA (Supervisory Control and Data Acquisition) архітектурі. Ці системи забезпечують моніторинг та управління технологічними процесами у реальному часі [3]. Основними компонентами SCADA-систем є:
1. польовий рівень: датчики, виконавчі механізми, контролери (plc);
2. комунікаційний рівень: протоколи передачі даних (modbus, profibus, opc ua);
3. диспетчерський рівень: сервери збору даних, робочі станції операторів;
4. рівень управління: системи аналізу та прийняття рішень.
Сучасні промислові системи еволюціонують у напрямку Industry 4.0, інтегруючи технології Інтернету речей (Industrial IoT). Це дозволяє збирати значно більші обсяги даних, застосовувати методи машинного навчання для прогнозування відмов обладнання та оптимізації виробничих процесів [2, 3].
Однак традиційні SCADA-системи мають суттєві обмеження: висока вартість впровадження, складність масштабування, залежність від **propriet**арних рішень окремих виробників та обмежена гнучкість конфігурації.

INCLUDEPICTURE "https://www.reverecontrol.com/wp-content/uploads/Scada-System-Components-1024x899.jpg" * MERGEFORMATINET

Рисунок 1.2 - SCADA система

Системи розумного будинку (Smart Home) представляють собою інтегровані рішення для автоматизації побутових приміщень. Ці системи дозволяють контролювати освітлення, опалення, кондиціонування, безпеку та інші аспекти життєдіяльності. Найпоширеніші платформи розумного будинку включають [4]:
1. Home Assistant - відкрита платформа з підтримкою понад 2000 різних пристроїв та сервісів. Система підтримує локальну обробку даних, що забезпечує конфіденційність та незалежність від хмарних сервісів [4].
2. OpenHAB - гнучка система автоматизації з модульною архітектурою, що дозволяє інтегрувати пристрої різних виробників через єдиний інтерфейс.
3. Domoticz - легка система моніторингу та автоматизації з підтримкою широкого спектру протоколів (Z-Wave, Zigbee, MQTT, HTTP).
4. Комерційні рішення, такі як Apple HomeKit, Google Home, Amazon Alexa, Samsung SmartThings, пропонують зручність використання, але

обмежують гнучкість конфігурації та вимагають використання хмарних сервісів виробника [10].

Бездротові сенсорні мережі (Wireless Sensor Networks, WSN) стали основою сучасних систем збору інформації з розподілених об'єктів. Ці мережі складаються з автономних сенсорних вузлів, які здатні збирати, обробляти та передавати дані [3, 4]. Основні характеристики WSN:

1. самоорганізація та адаптивність топології мережі;
2. енергоефективність та тривалий час автономної роботи;
3. масштабованість від десятків до тисяч вузлів;
4. надійність передачі даних через резервування маршрутів.

Технології бездротового зв'язку для WSN:

1. Wi-Fi (IEEE 802.11) - забезпечує високу швидкість передачі даних (до 1 Гбіт/с) та широке покриття, але характеризується підвищеним енергоспоживанням;
2. Bluetooth/BLE (Bluetooth Low Energy) - оптимізований для низького енергоспоживання, підходить для персональних мереж з радіусом до 100 метрів;
3. Zigbee (IEEE 802.15.4) - спеціалізований протокол для mesh-мереж з низьким енергоспоживанням, підтримує до 65000 вузлів у мережі;
4. LoRaWAN - технологія дальнього радіусу дії (до 15 км) з мінімальним енергоспоживанням, ідеальна для сільськогосподарського та екологічного моніторингу [4];
5. Z-Wave - протокол для домашньої автоматизації з низькою інтерференцією та високою надійністю.

Сучасне сільське господарство активно впроваджує технології точного землеробства (Precision Agriculture), які базуються на збиранні та аналізі даних про стан ґрунту, метеорологічні умови, здоров'я рослин та ефективність використання ресурсів. Типові параметри моніторингу включають:

1. температура та вологість ґрунту на різних глибинах;
2. температура та вологість повітря;
3. рівень опадів та швидкість вітру;
4. освітленість та uv-індекс;
5. рівень ph та поживних речовин у ґрунті;
6. стан іригаційних систем.

Системи агромоніторингу часто використовують комбінацію бездротових технологій: LoRaWAN для передачі даних на великі відстані у полі, Wi-Fi для локальних концентраторів даних, та мобільні мережі (4G/5G) для передачі інформації до центральних серверів [5].

Хмарні платформи для Інтернету речей надають комплексні рішення для збору, зберігання, обробки та візуалізації даних з розподілених об'єктів [17]. AWS IoT Core - платформа Amazon Web Services з підтримкою мільйонів пристроїв, інтеграцією з сервісами машинного навчання та аналітики. Microsoft Azure IoT Hub - рішення з акцентом на промислові застосування, підтримка протоколів MQTT, AMQP, HTTPS. Google Cloud IoT - платформа з потужними інструментами аналізу даних та машинного навчання. ThingSpeak - відкрита платформа для збору та аналізу даних IoT з підтримкою MATLAB для обробки. Blynk - платформа з простим конструктором мобільних додатків для IoT-пристроїв [5].

Основними перевагами хмарних платформ є масштабованість, відсутність необхідності підтримувати власну інфраструктуру, вбудовані інструменти аналізу та візуалізації. До недоліків відносяться залежність від Інтернет-з'єднання, щомісячна плата за використання, питання конфіденційності даних [6].

Концепція граничних обчислень (Edge Computing) та туманних обчислень (Fog Computing) передбачає обробку даних безпосередньо на пристроях або локальних шлюзах, а не у віддаленому хмарному центрі [6]. Переваги edge computing:

1. зменшення затримок обробки даних;
2. економія пропускної здатності каналів зв'язку;
3. підвищення конфіденційності та безпеки даних;
4. можливість роботи при відсутності з'єднання з центральним сервером;
5. зменшення навантаження на хмарні сервери.

Сучасні мікроконтролери, такі як ESP32, достатньо потужні для виконання локальної обробки даних, фільтрації, агрегації та навіть виконання простих алгоритмів машинного навчання (TinyML) [6].

Аналіз сучасних систем збору інформації дозволяє виділити наступні тенденції розвитку:

1. конвергенція технологій - об'єднання IT та OT (Operational Technology), інтеграція промислових та споживчих IoT-рішень.
2. штучний інтелект та машинне навчання - впровадження алгоритмів прогнозування, аномалій та оптимізації безпосередньо на рівні сенсорних вузлів [7].
3. підвищення енергоефективності - використання технологій energy harvesting (збирання енергії з навколишнього середовища), покращені алгоритми sleep mode.
4. стандартизація протоколів - розвиток відкритих стандартів (Matter, Thread) для забезпечення сумісності пристроїв різних виробників.
5. безпека за дизайном - впровадження криптографічного захисту на апаратному рівні, secure boot, шифрування даних за замовчуванням.
6. відкриті рішення - зростання популярності open-source платформ, які дозволяють повний контроль над даними та гнучкість налаштування.

Розглянуті системи демонструють широкий спектр підходів до збору інформації з розподілених об'єктів. Для побутових та невеликих промислових застосувань особливий інтерес представляють відкриті рішення на базі доступних мікроконтролерів, що поєднують гнучкість конфігурації, низьку вартість та можливість локальної обробки даних. Саме до таких рішень належить платформа ESPHome, детальний аналіз якої буде представлено у наступних розділах [8].

## 1.2 Аналіз існуючих платформ та технологій для побудови IoT-систем

Побудова ефективних систем Інтернету речей вимагає комплексного підходу, який охоплює як апаратні компоненти, так і програмне забезпечення для їх конфігурування, управління та інтеграції. У цьому розділі проводиться детальний аналіз сучасних платформ та технологій, що використовуються для створення IoT-систем [8].

Arduino є однією з найпопулярніших платформ для прототипування IoT-пристроїв. Екосистема Arduino включає широкий спектр мікроконтролерних плат на базі чіпів Atmel AVR та ARM. Основні характеристики:

1. процесор: 8-бітні AVR (Arduino Uno, Nano) або 32-бітні ARM Cortex (Arduino Due, MKR);
2. тактова частота: 16-84 МГц;

3. оперативна пам'ять: 2-96 КБ;
4. Flash-пам'ять: 32-512 КБ;
5. середовище розробки: Arduino IDE з спрощеним C/C++.
Основні переваги: велика спільнота розробників та величезна база прикладів коду, проста архітектура, ідеальна для навчання, широкий вибір сумісних модулів та датчиків (shields), відкрита апаратна платформа.
Основні недоліки: обмежені обчислювальні ресурси для складних завдань, відсутність вбудованого wi-fi/bluetooth у базових моделях, відносно високе енергоспоживання, необхідність додаткових модулів для бездротового зв'язку [8, 9].
Raspberry Pi представляє собою одноплатний комп'ютер, який працює під управлінням повноцінної операційної системи Linux [9]. Основні характеристики (Raspberry Pi 4):
1. процесор: 64-бітний ARM Cortex-A72, 1.5 ГГц (4 ядра);
2. оперативна пам'ять: 2-8 ГБ LPDDR4;
3. вбудовані інтерфейси: Wi-Fi 802.11ac, Bluetooth 5.0, Gigabit Ethernet;
4. GPIO: 40 pin для підключення периферії;
5. підтримка HDMI, USB 3.0, камери.
Основні переваги: потужність, достатня для складних обчислень та машинного навчання, повноцінна ОС з доступом до величезної кількості Linux-пакетів, можливість використання як edge-сервер або шлюз, підтримка Python, C/C++, Java та інших мов програмування [9].
Основні недоліки: високе енергоспоживання (5-8 Вт), неприйнятне для автономних пристроїв, надмірні ресурси для простих сенсорних вузлів, відсутність аналогових входів без додаткових модулів, потребує стабільного живлення та proper shutdown [9].
ESP8266 - мікроконтролер від компанії Espressif Systems, який революціонізував ринок IoT завдяки вбудованому Wi-Fi модулю та низькій вартості [10]. Основні характеристики:
1. процесор: 32-бітний Tensilica L106, 80/160 МГц;
2. оперативна пам'ять: 80 КБ (користувачу доступно ~36 КБ);
3. Flash- пам'ять: зазвичай **4 МБ;**
4. Wi-Fi: **802.11 b/g/n 2.4 ГГц;**
5. GPIO: 16 цифрових входів/виходів (деякі з обмеженнями);
6. АЦП: 1 канал, 10-біт;
7. інтерфейси: UART, SPI, I2C.
Основні переваги: дуже низька вартість ($2-4 за модуль), вбудований Wi-Fi з можливістю роботи як станція або точка доступу, можливість OTA-оновлень (Over-The-Air), підтримка Arduino IDE та інших середовищ розробки, низьке споживання в режимі deep sleep (20 мкА) [10].
Основні недоліки: обмежена кількість GPIO, лише один ацп обмежений обсяг оперативної пам'яті, відсутність bluetooth деякі GPIO мають обмеження використання.
ESP32 - наступне покоління мікроконтролерів від Espressif, що усуває багато недоліків ESP8266 [10]. Основні характеристики:
1. процесор: Dual-core Tensilica Xtensa LX6, до 240 МГц;
2. оперативна пам'ять: **520** КБ **SRAM;**
3. **Flash-** пам'ять: зазвичай **4-** 16 **МБ;**
4. Wi-Fi: **802.11 b/g/n 2.4 ГГц;**
5. Bluetooth: Classic та BLE 4.2/5.0;
6. GPIO: 34 програмовані входи/виходи;
7. АЦП: 18 каналів, 12-біт;
8. ЦАП: 2 канали, 8-біт;
9. інтерфейси: UART (3x), SPI (4x), I2C (2x), I2S, CAN, Ethernet MAC.
Основні переваги: два ядра процесора для паралельної обробки, багато GPIO та периферійних інтерфейсів, підтримка Wi-Fi та Bluetooth одночасно, апаратне шифрування (AES, SHA, RSA), низьке споживання в deep sleep (5-10 мкА), сенсорні входи (touch sensors), можливість використання FreeRTOS Велика спільнота та бібліотеки [10].
Основні недоліки: дещо вища вартість порівняно з ESP8266 ($4-8), більш складна архітектура для початківців, деякі GPIO мають обмеження та спільне використання з flash.
Сімейство 32-бітних мікроконтролерів від STMicroelectronics на базі ARM Cortex-M ядер [10]. Основні характеристики (залежно від серії):
1. процесор: **ARM** Cortex-M0/M 3/M4/ **M7,** до 480 МГц;
2. оперативна пам'ять: 4 КБ - 1 МБ;
3. Flash-пам'ять: 16 КБ - 2 МБ;
4. низьке енергоспоживання;
5. широкий діапазон периферії.
Основні переваги: промислова надійність та сертифікація, низьке споживання енергії, потужні обчислювальні можливості, широкий вибір моделей під різні задачі.
Основні недоліки: складніше програмування порівняно з Arduino, потребує зовнішніх модулів для Wi-Fi/Bluetooth, більш крута крива навчання [10]. Коротке порівняння наведено в таблиці 1.1.
Таблиця 1.1 - Порівняння апаратних платформ

| Характеристика | Arduino Uno | Raspberry Pi 4 | ESP8266 | ESP32 | STM32F4 |
|---|---|---|---|---|---|
| Процесор | 8-біт, 16 МГц | 4x1.5 ГГц | 80-160 МГц | 2x240 МГц | 168 МГц |
| RAM | 2 КБ | 2-8 ГБ | 80 КБ | 520 КБ | 192 КБ |
| Flash | 32 КБ | SD-карта | 4 МБ | 4-16 МБ | 1 МБ |
| Wi-Fi | - | + | + | + | - |
| Bluetooth | - | + | - | + | - |
| Споживання | ~50 мА | 600+ мА | 80-170 мА | 80-260 мА | 50-100 мА |
| Складність | Низька | Висока | Середня | Середня | Висока |

Arduino Framework забезпечує спрощений інтерфейс для роботи з мікроконтролерами через абстракцію апаратного рівня. Її особливості:

1. проста структура: setup() та loop();
2. величезна бібліотека готових функцій;
3. підтримка багатьох апаратних платформ;
4. інтеграція з PlatformIO та інших IDE.
Переваги: простота використання, швидкий старт для початківців.
Недоліки: обмежений контроль над апаратурою, неоптимальний код для складних проектів.
ESP-IDF - офіційний фреймворк від Espressif для розробки на ESP32/ESP8266 з використанням FreeRTOS [35]. Особливості: повний доступ до всіх можливостей апаратури, підтримка багатозадачності через FreeRTOS, оптимізований код та низьке споживання, професійні інструменти налагодження [11].
Переваги: максимальна продуктивність та контроль, підходить для промислових застосувань, регулярні оновлення від виробника.
Недоліки: крута крива навчання, більш складна структура проекту, потребує знання RTOS-концепцій.
Інтерпретовані мови Python для мікроконтролерів, що дозволяють швидку розробку та прототипування. Наприклад MicroPython:
1. реалізація Python 3 для обмежених ресурсів;
2. інтерактивний REPL для налагодження;
3. підтримка ESP32, ESP8266, STM32, RP2040.
Переваги: швидка розробка та тестування, простота синтаксису Python, не потребує компіляції.
Недоліки: вища затримка виконання, більше споживання пам'яті, обмежений доступ до низькорівневих функцій.
Tasmota - альтернативна прошивка з відкритим кодом для пристроїв на ESP8266/ESP32, орієнтована на домашню автоматизацію.
Особливості: веб-інтерфейс для конфігурації, підтримка MQTT, HTTP, KNX, готові шаблони для популярних пристроїв, планувальник та правила автоматизації.
Переваги: не потребує програмування, готове рішення для типових задач, активна спільнота.
Недоліки: обмежена гнучкість для специфічних проектів, велика прошивка може не поміститися на ESP8266 з обмеженою пам'яттю, складність додавання власної логіки.
ESPHome - система конфігурування IoT-пристроїв на ESP через YAML-файли без написання коду. Особливості: декларативний підхід: опис бажаного стану системи, автоматична генерація C++ коду, нативна інтеграція з Home Assistant, підтримка понад 100 типів датчиків та компонентів [11].
Lua - легка скриптова мова програмування, яка знайшла широке застосування у вбудованих системах завдяки своїй простоті, невеликому розміру інтерпретатора та ефективності виконання [11].
NodeMCU firmware - це прошивка з відкритим кодом для ESP8266/ESP32, яка включає інтерпретатор Lua та надає високорівневий API для роботи з апаратними можливостями мікроконтролера. Архітектура NodeMCU: NodeMCU надає модульну структуру, де кожен модуль відповідає за певну функціональність: WiFi, GPIO, UART, I2C, SPI, MQTT, HTTP та інші [11].
Переваги: простота конфігурації, швидкий розвиток проекту ota-оновлення, готові компоненти для більшості датчиків.
Недоліки: обмеження для дуже специфічних завдань, потребує компіляції при кожній зміні, залежність від home assistant екосистеми.
PlatformIO - професійна платформа для розробки вбудованих систем з підтримкою понад 50 платформ. Особливості: інтеграція з Visual Studio Code, Atom, Clion, менеджер бібліотек, підтримка різних фреймворків (Arduino, ESP-IDF, STM32Cube), просунуті можливості налагодження CI/CD інтеграція [11].
Переваги: професійне середовище розробки, одна платформа для різних мікроконтролерів, потужний менеджер залежностей.
Недоліки: більш складне налаштування порівняно з Arduino IDE, споживає більше ресурсів комп'ютера.
Node-RED - візуальний інструмент програмування для з'єднання IoT-пристроїв, API та онлайн-сервісів. Особливості: браузерний редактор flow-based програмування Велика бібліотека готових вузлів, підтримка MQTT, HTTP, WebSocket, можливість створення dashboard [11].
Переваги: не потребує традиційного програмування, швидке прототипування логіки, візуалізація потоків даних.
Недоліки: працює на окремому сервері (Raspberry Pi, хмара), не підходить для програмування самих ESP-пристроїв, може бути складним для дуже великих проектів. Коротке порівняння наведено в таблиці 1.2.
Таблиця 1.2 - Порівняння програмних рішень

| Фреймворк | Мова | Складність | Гнучкість | Продуктивність | Швидкість розробки |
|---|---|---|---|---|---|
| Arduino | C/C++ | Низька | Середня | Середня | Висока |
| ESP-IDF | C | Висока | Висока | Висока | Низька |
| MicroPython | Python | Низька | Середня | Низька | Висока |
| Tasmota | Конфігурація | Низька | Низька | Середня | Висока |
| ESPHome | YAML | Низька | Середня | Висока | Висока |
| PlatformIO | Різні | Середня | Висока | Висока | Середня |

При виборі платформи для побудови системи збору інформації необхідно враховувати наступні критерії:
1. Функціональні вимоги: типи та кількість підключених датчиків, необхідність бездротового зв'яз ку (Wi-Fi, Bluetooth, LoRa), обсяг даних та частота їх передачі, необхідність локальної обробки даних, потреба в автономному живленні [12];
2. Технічні вимоги: обчислювальна потужність, обсяг пам'яті, кількість та типи інтерфейсів, енергоспоживання, можливість OTA-оновлень;
3. Економічні фактори: вартість апаратних компонентів, вартість розробки та підтримки, масштабованість рішення, доступність компонентів на ринку;
4. Експлуатаційні характеристики: надійність та стійкість до відмов, простота налаштування та обслуговування, можливість віддаленого управління, безпека та захист даних [12].
На основі проведеного аналізу можна зробити висновок, що для побудови гнучкої, масштабованої та економічно ефективної системи збору інформації з розподілених об’єктів оптимальним вибором є поєднання апаратної платформи ESP з програмним рішенням ESPHome. Така комбінація забезпечує баланс між функціональністю, простотою розробки та експлуатаційними характеристиками, що робить її придатною як для автоматизації дому, так і для невеликих промислових застосувань [12].

## 1.3 Протоколи передачі даних у розподілених системах

Ефективність роботи розподіленої системи збору інформації значною мірою залежить від вибору протоколів передачі даних між компонентами системи. Протоколи визначають правила обміну інформацією, формати повідомлень, методи забезпечення надійності та безпеки передачі даних. У розподілених IoT-системах використовуються різноманітні протоколи, кожен з яких має свої особливості, переваги та обмеження, що робить їх придатними для різних сценаріїв застосування [12].

Вибір протоколу передачі даних повинен враховувати специфіку завдання, характеристики мережевої інфраструктури, обмеження пристроїв за енергоспоживанням та обчислювальними ресурсами, а також вимоги до швидкості, надійності та безпеки передачі інформації. Сучасні розподілені системи часто використовують комбінацію різних протоколів на різних рівнях архітектури для досягнення оптимального балансу між функціональністю та ефективністю.

MQTT **(Message Queuing Telemetry Transport) є одним з найпопулярніших протоколів** для систем Інтернету речей, розроблений спеціально для ефективної передачі даних у мережах з обмеженою пропускною здатністю та ненадійними з'єднаннями. Протокол був створений у 1999 році компанією IBM для моніторингу нафтопроводів через супутниковий зв'язок, а у 2013 році став відкритим стандартом OASIS [12].

Основою архітектури MQTT є модель publish-subscribe, яка принципово відрізняється від традиційної клієнт-серверної моделі. У цій моделі існує три основні компоненти: publisher (публікатор), subscriber (підписник) та broker (брокер-посередник). Публікатори відправляють повідомлення до брокера з певною темою (topic), а підписники отримують повідомлення, підписавшись на цікаві їм теми. Така архітектура забезпечує слабке зв'язування (loose coupling) між відправниками та отримувачами даних, що є критично важливим для масштабованих розподілених систем [12].

Брокер MQTT виступає центральним вузлом комунікації, який приймає всі повідомлення від публікаторів та розповсюджує їх відповідним підписникам. Найпопулярнішими реалізаціями MQTT-брокерів є Mosquitto (відкритий проект Eclipse Foundation), HiveMQ (комерційне рішення з високою продуктивністю), EMQX (масштабований брокер для великих систем) та VerneMQ (розподілений брокер на Erlang). Кожен з цих брокерів має свої особливості щодо продуктивності, масштабованості та додаткових функцій, але всі вони підтримують базовий стандарт MQTT [13].

Система тем у MQTT організована ієрархічно з використанням слешів як роздільників, що нагадує файлову систему Unix. Наприклад, тема може виглядати набути наступного прикладу: "home/livingroom/temperature" або "factory/building1/floor2/machine5/status". Така структура дозволяє логічно організувати потоки даних та забезпечує гнучкість при підписці на групи пов'язаних тем. MQTT підтримує спеціальні символи підстановки для підписки на множину тем одночасно. Символ "+" (плюс) замінює один рівень ієрархії, наприклад, підписка "home/+/temperature" отримуватиме дані з усіх кімнат будинку. Символ "#" (решітка) замінює всі наступні рівні ієрархії і повинен бути останнім у темі, наприклад, "factory/building1/#" підписується на всі теми у першому будинку фабрики незалежно від їх глибини вкладеності.

Одною з ключових особливостей MQTT є підтримка трьох рівнів якості обслуговування (Quality of Service, QoS), які визначають гарантії доставки повідомлень. Вибір відповідного рівня QoS дозволяє балансувати між надійністю передачі та використанням мережевих ресурсів.

QoS 0, також відомий як "at most once" (щонайбільше один раз), є найпростішим режимом без підтвердження доставки. Повідомлення відправляється один раз без очікування підтвердження від отримувача, що робить цей режим найшвидшим та найменш ресурсомістким. Однак існує ризик втрати повідомлення при проблемах з мережею. Цей рівень підходить для даних, які швидко застарівають, наприклад, поточних показників датчиків, що передаються з високою частотою [13].

QoS 1, або "at least once" (щонайменше один раз), гарантує доставку повідомлення принаймні один раз через механізм підтвердження. Відправник зберігає повідомлення до отримання підтвердження PUBACK від брокера, і повторює відправку при відсутності підтвердження протягом певного часу. Це може призвести до дублювання повідомлень, тому отримувач повинен бути готовим обробляти дублікати. Цей рівень є золотою серединою для більшості IoT-застосувань, забезпечуючи баланс між надійністю та ефективністю.

QoS 2, відомий як "exactly once" (точно один раз), надає найвищі гарантії доставки через складний чотирьохетапний механізм підтвердження (PUBLISH, PUBREC, PUBREL, PUBCOMP). Це гарантує, що кожне повідомлення буде доставлено отримувачу рівно один раз без дублювання або втрати. Однак така надійність досягається за рахунок суттєвого збільшення мережевого трафіку та затримок. QoS 2 рекомендується використовувати лише для критичних команд або фінансових транзакцій, де дублювання абсолютно неприйнятне.

MQTT підтримує концепцію постійних сесій (persistent sessions), які зберігають стан підключення клієнта навіть після його відключення. При встановленні з'єднання клієнт може вказати прапорець Clean Session як false, що вказує брокеру зберігати інформацію про підписки клієнта та повідомлення, які надійшли під час його відсутності. Це особливо корисно для пристроїв з періодичним підключенням або нестабільним з'єднанням, оскільки після відновлення зв'язку пристрій отримає всі пропущені повідомлення відповідно до їх рівня QoS [13].

Retained messages (збережені повідомлення) є іншою важливою функцією MQTT, яка дозволяє брокеру зберігати останнє повідомлення за певною темою. Коли новий підписник підписується на тему, він негайно отримує останнє збережене повідомлення, якщо воно існує, не чекаючи на наступну публікацію. Це корисно для статусних повідомлень або конфігураційних даних, які повинні бути доступні відразу після підключення пристрою. Наприклад, стан перемикача (увімкнено/вимкнено) може публікуватися як retained message, щоб нові підписники одразу знали поточний стан без очікування зміни.

Механізм Last Will and Testament (останнє заповіт та завіт) є унікальною особливістю MQTT, яка дозволяє пристроям визначити повідомлення, яке буде автоматично опубліковано брокером у разі несподіваного відключення пристрою. При встановленні з'єднання клієнт може вказати will topic, will message та will QoS. Якщо брокер виявляє, що клієнт відключився без коректного завершення сесії (наприклад, через втрату живлення або мережеве відключення), він автоматично публікує will message у вказану тему [13].

Цей механізм критично важливий для моніторингу доступності пристроїв у розподіленій системі. Наприклад, датчик може встановити will message як "offline" для теми "sensors/temperature1/status", і якщо датчик раптово втратить з'єднання, інші компоненти системи одразу дізнаються про це та зможуть відреагувати відповідно. Типові застосування включають моніторинг здоров'я системи, автоматичне переключення на резервні пристрої та сповіщення адміністраторів про проблеми.

У 2019 році було випущено MQTT версії 5.0, яка внесла суттєві покращення порівняно з попередньою версією 3.1.1. Нова версія додала user properties, які дозволяють передавати довільні метадані разом з повідомленнями без впливу на основне payload. Це корисно для передачі контекстної інформації, timestamps, ідентифікаторів кореляції або будь-яких інших атрибутів [14].

Request-response pattern був формалізований у версії 5.0 через спеціальні властивості Response Topic та Correlation Data, що спрощує реалізацію синхронної комунікації поверх асинхронного MQTT. Shared subscriptions дозволяють розподіляти навантаження між множиною підписників однієї теми, що важливо для масштабування систем обробки повідомлень. Topic aliases дозволяють замінити довгі імена тем короткими числовими ідентифікаторами для економії пропускної здатності при частій передачі в одну тему [14].

Механізм flow control через receive maximum дозволяє клієнту обмежити кількість неопрацьованих повідомлень QoS 1 та QoS 2, запобігаючи перевантаженню повільних клієнтів. Server-sent disconnect з кодом причини та описом помилки полегшує діагностику проблем з підключенням. Enhanced authentication підтримує складніші схеми автентифікації, включаючи challenge-response методи.

Стандартний MQTT за замовчуванням не шифрує дані, що робить їх вразливими до перехоплення. Тому для продакшн-систем критично важливо використовувати MQTT over TLS (також відомий як MQTTS), який працює на порту 8883 замість стандартного 1883. TLS забезпечує шифрування каналу зв'язку та автентифікацію сервера через сертифікати.
Автентифікація клієнтів може здійснюватися через username та password, що підтримується всіма брокерами. Для підвищення безпеки рекомендується використовувати клієнтські сертифікати (mutual TLS), коли не тільки сервер, а й клієнт повинен підтвердити свою ідентичність через сертифікат. Це особливо важливо для критичних промислових систем або пристроїв, які контролюють фізичні процеси.
Авторизація визначає, які теми клієнт може читати або публікувати, та зазвичай реалізується через Access Control Lists (ACL). Багато брокерів підтримують динамічну авторизацію через зовнішні системи, такі як бази даних або LDAP, що дозволяє централізовано управляти правами доступу для великої кількості пристроїв. Деякі брокери також підтримують payload шифрування на рівні додатку для додаткового захисту чутливих даних [15].
Основними перевагами MQTT є легкість та ефективність протоколу, що дозволяє використовувати його на пристроях з обмеженими ресурсами. Мінімальний розмір пакету MQTT може бути всього 2 байти, що критично важливо при передачі даних через мережі з обмеженою пропускною здатністю або високою вартістю трафіку. Модель publish-subscribe забезпечує відокремлення відправників від отримувачів, що спрощує масштабування системи та додавання нових компонентів без модифікації існуючих.
Гнучкість рівнів QoS дозволяє оптимізувати баланс між надійністю та ефективністю для кожного типу даних окремо. Вбудовані механізми для обробки нестабільних з'єднань, такі як persistent sessions та last will, роблять MQTT ідеальним для мобільних пристроїв та пристроїв з періодичним підключенням. Широка підтримка MQTT у різних платформах, мовах програмування та IoT-фреймворках робить його де-факто стандартом для IoT-комунікацій [15].
Серед недоліків можна відзначити необхідність центрального брокера, який стає single point of failure для всієї системи. Хоча існують рішення для кластеризації та резервування брокерів, вони додають складності до архітектури системи. MQTT не підходить для передачі великих файлів або потокового відео через накладні витрати на підтвердження та відсутність механізмів сегментації. Відсутність стандартизації форматів payload вимагає додаткової угоди між публікаторами та підписниками щодо структури даних, хоча на практиці часто використовується JSON.
HTTP (Hypertext Transfer Protocol) є фундаментальним протоколом World Wide Web та широко використовується у IoT-системах завдяки своїй універсальності, простоті та повсюдній підтримці. На відміну від MQTT з його асинхронною моделлю publish-subscribe, HTTP базується на традиційній синхронній моделі запит-відповідь, де клієнт ініціює комунікацію через запит до сервера, а сервер відповідає даними або статусом виконання операції [16].
REST (Representational State Transfer) є архітектурним стилем для розробки веб-сервісів, який визначає набір принципів та обмежень для створення масштабованих розподілених систем. RESTful API базується на використанні стандартних HTTP методів для виконання операцій над ресурсами, які ідентифікуються через URL. Основні HTTP методи включають GET для читання даних, POST для створення нових ресурсів, PUT або PATCH для оновлення існуючих ресурсів, та DELETE для їх видалення. У контексті IoT-систем RESTful API часто використовується для взаємодії з пристроями, де кожен датчик або актуатор представляється як ресурс з унікальним URL. Така модель є інтуїтивно зрозумілою для розробників веб-додатків та добре документованою [16, 17].
Важливим принципом REST є stateless комунікація, коли **кожен** запит містить всю необхідну **інформацію для його обробки, а сервер не зберігає** стан клієнтської сесії між запитами. **Це спрощує масштабування серверної частини та підвищує** надійність системи, оскільки відсутність стану означає відсутність проблем з його синхронізацією або втратою. Автентифікація при stateless підході зазвичай реалізується через токени (наприклад, JWT), які клієнт включає в кожен запит.
Традиційний HTTP/1.1 має ряд обмежень, які особливо помітні в IoT-сценаріях з множиною одночасних з'єднань. Протокол HTTP/2, стандартизований у 2015 році, вніс суттєві покращення через мультиплексування запитів, коли множина запитів та відповідей можуть передаватися одночасно через одне TCP-з'єднання. Це усуває проблему head-of-line blocking та зменшує накладні витрати на встановлення з'єднань.
Server push у HTTP/2 дозволяє серверу проактивно відправляти дані клієнту без явного запиту, що може бути корисним для сповіщень від IoT-пристроїв. Header compression через HPACK алгоритм зменшує розмір метаданих запитів та відповідей, що важливо при обмеженій пропускній здатності. Binary framing замість текстового формату HTTP/1.1 робить протокол більш ефективним для парсингу та передачі [17].
HTTP/3, що базується на протоколі QUIC замість TCP, обіцяє ще більші покращення через усунення head-of-line blocking на транспортному рівні та швидше встановлення з'єднань. Однак станом на сьогодні підтримка HTTP/2 та особливо HTTP/3 у мікроконтролерах залишається обмеженою через їх складність та вимоги до ресурсів.
WebSocket є протоколом, який надає full-duplex комунікацію через одне TCP-з'єднання, що встановлюється через HTTP upgrade запит. На відміну від традиційного HTTP з його моделлю запит-відповідь, WebSocket дозволяє серверу відправляти дані клієнту в будь-який момент без попереднього запиту. Це робить WebSocket привабливим для IoT-застосувань, які потребують real-time оновлення даних [17].
Після встановлення WebSocket з'єднання через initial HTTP handshake, комунікація відбувається через binary або text frames з мінімальними накладними витратами. Це забезпечує низьку латентність та ефективне використання ресурсів для довготривалих з'єднань. WebSocket особливо корисний для dashboard-додатків, які відображають дані з множини датчиків у реальному часі, або для систем віддаленого управління, де команди повинні передаватися миттєво [15, 16, 17].
Однак WebSocket вимагає постійного відкритого з'єднання, що може бути проблематичним для пристроїв з обмеженою енергією або нестабільним інтернет-з'єднанням. Для таких випадків MQTT часто є кращим вибором. WebSocket також підтримує роботу через проксі-сервери та firewall, що використовують стандартні HTTP порти, що полегшує його впровадження в корпоративних мережах.
При використанні HTTP для IoT-комунікацій критично важливим є вибір формату представлення даних. JSON (JavaScript Object Notation) є найпопулярнішим форматом завдяки своїй читабельності, простоті парсингу та широкій підтримці в різних мовах програмування. JSON ідеально підходить для передачі структурованих даних від датчиків, конфігураційних параметрів або команд управління. Однак JSON є текстовим форматом з відносно високими накладними витратами через повторення імен полів у кожному об'єкті [17].
XML (eXtensible Markup Language) був популярним у минулому, особливо в корпоративних системах, але сьогодні рідко використовується в IoT через свою verbose природу та складність парсингу на обмежених пристроях. MessagePack є binary форматом, що забезпечує компактнішу серіалізацію порівняно з JSON при збереженні схожої структури даних. Protocol Buffers (Protobuf) від Google забезпечує ще більш ефективну серіалізацію через використання схеми, яка компілюється в код для швидкого кодування та декодування [17, 18].
CBOR (Concise Binary Object Representation) є binary форматом, оптимізованим для IoT-застосувань, який пропонує компактність бінарних форматів з гнучкістю JSON. CBOR підтримує різні типи даних, включаючи integers, floats, strings, arrays, maps, та навіть теги для спеціальних типів даних. Для систем з критичними обмеженнями пропускної здатності можливе використання власних binary протоколів, хоча це вимагає

більше зусиль на розробку та підтримку.
Використання HTTPS (HTTP over TLS) є абсолютно необхідним для продакшн IoT-систем, оскільки забезпечує шифрування даних та автентифікацію сервера. Сучасні браузери та операційні системи навіть відмовляються працювати з незахищеним HTTP для багатьох типів контенту. ESP32 та ESP8266 підтримують HTTPS через бібліотеки WiFiClientSecure або HTTPClient з TLS, хоча це вимагає додаткових ресурсів пам'яті для сертифікатів та криптографічних операцій.
Автентифікація IoT-пристроїв при використанні HTTP може здійснюватися різними методами. Basic Authentication передає username та password у заголовку Authorization, але вразливий без HTTPS. Bearer Tokens, особливо JWT (JSON Web Tokens), є популярним методом для stateless автентифікації, де токен містить закодовану інформацію про користувача та його права доступу. API Keys є простим методом, коли кожен пристрій має унікальний ключ, який включається в запити [13, 15].
OAuth 2.0 є складнішим, але більш безпечним протоколом авторизації, який широко використовується для інтеграції з хмарними сервісами. Mutual TLS (mTLS) забезпечує найвищий рівень безпеки через взаємну автентифікацію клієнта та сервера з використанням сертифікатів, що робить його ідеальним для критичних промислових систем, де безпека є пріоритетом. Certificate pinning може використовуватися для захисту від атак типу man-in-the-middle через перевірку конкретного сертифікату сервера [19].
Основною перевагою HTTP є його універсальність та повсюдна підтримка. Практично будь-який пристрій з мережевим підключенням може працювати з HTTP, а інструменти для розробки, тестування та моніторингу HTTP API є зрілими та широко доступними. Синхронна природа запит-відповідь робить HTTP простим для розуміння та налагодження, особливо для розробників без спеціалізованого досвіду в IoT. RESTful архітектура забезпечує чітку структуру API, що полегшує документування та використання.
HTTP добре підходить для випадків, коли пристрій періодично відправляє дані або виконує команди на вимогу. Це типово для систем моніторингу, де датчики опитуються з певною періодичністю, або для систем управління, де команди відправляються користувачем через веб-інтерфейс. HTTP також природно інтегрується з веб-технологіями, що спрощує створення веб-інтерфейсів для управління IoT-пристроями.
Однак HTTP має суттєві недоліки для багатьох IoT-сценаріїв. Кожен запит вимагає встановлення з'єднання, передачі заголовків та очікування відповіді, що створює значні накладні витрати, особливо для малих повідомлень. Для пристрою, який відправляє кілька байт даних кожні кілька хвилин, накладні витрати HTTP можуть перевищувати корисні дані у десятки разів. Відсутність вбудованої підтримки асинхронних сповіщень вимагає використання polling або додаткових технологій типу WebSocket [19].
Енергоспоживання є критичною проблемою для батарейних пристроїв, оскільки кожен HTTP запит вимагає активації Wi-Fi модуля, встановлення TCP-з'єднання, виконання TLS handshake, передачі даних та очікування відповіді. Це може займати секунди та споживати суттєву енергію порівняно з легковаговими протоколами. HTTP також не має вбудованих механізмів для обробки втрати з'єднання або гарантованої доставки повідомлень, що вимагає реалізації логіки повторних спроб на рівні додатку [19].
CoAP був розроблений IETF спеціально для обмежених пристроїв та мереж як альтернатива HTTP, оптимізована для IoT. Протокол використовує UDP замість TCP, що зменшує накладні витрати на встановлення з'єднання та робить його більш ефективним для невеликих повідомлень. CoAP підтримує RESTful архітектуру, схожу на HTTP, з методами GET, POST, PUT, DELETE, що полегшує міграцію для розробників, знайомих з HTTP [17, 18, 19].
CoAP підтримує observe pattern, який дозволяє клієнту підписатися на ресурс та отримувати автоматичні оновлення при його зміні, схоже на MQTT publish-subscribe. Block-wise transfers дозволяють передавати великі ресурси фрагментами для пристроїв з обмеженою пам'яттю. Resource discovery через well-known URIs (.well-known/core) дозволяє автоматично виявляти доступні ресурси на пристрої. DTLS (Datagram TLS) забезпечує шифрування та автентифікацію для CoAP аналогічно TLS для HTTP.
Незважаючи на свої переваги, CoAP має обмежену підтримку порівняно з HTTP та MQTT. Багато хмарних платформ та IoT-сервісів не підтримують CoAP нативно, що вимагає використання проксі або шлюзів для інтеграції. UDP nature CoAP може створювати проблеми з проходженням через NAT та firewall, які часто налаштовані на роботу з TCP-трафіком. Складність DTLS та обмежена підтримка в бібліотеках для мікроконтролерів також стримують широке впровадження CoAP [19].
Вибір протоколу передачі даних для конкретної системи збору інформації повинен базуватися на комплексному аналізі вимог до системи, характеристик мережевого середовища, обмежень пристроїв та сценаріїв застосування. У наступному розділі буде детально розглянуто платформу ESPHome та її підхід до організації комунікації в розподілених IoT-системах.

## 1.4 Огляд платформи ESPHome

ESPHome є відкритою системою для конфігурування та управління ESP8266 та ESP32 пристроями, яка радикально спрощує процес створення IoT-рішень через використання декларативного підходу до програмування. На відміну від традиційних методів розробки, де розробник пише код на C++ або Python, ESPHome дозволяє описувати бажану функціональність пристрою у YAML-файлах, після чого автоматично генерує оптимізований C++ код, компілює його та завантажує на мікроконтролер. Такий підхід значно знижує поріг входу для створення IoT-пристроїв та дозволяє зосередитися на функціональності системи, а не на деталях імплементації низького рівня [20].
Платформа була створена Отто Вінтером у 2018 році як альтернатива існуючим рішенням для домашньої автоматизації та швидко здобула популярність завдяки простоті використання, потужній функціональності та тісній інтеграції з Home Assistant. У 2021 році проект ESPHome офіційно увійшов до складу екосистеми Home Assistant, що забезпечило йому стабільну підтримку та активний розвиток. Сьогодні ESPHome використовується тисячами ентузіастів та професіоналів по всьому світу для створення різноманітних IoT-пристроїв від простих датчиків температури до складних систем управління освітленням та безпеки.
Архітектура ESPHome побудована на принципі розділення конфігурації та реалізації, що є класичним підходом у сучасній розробці програмного забезпечення. Користувач створює конфігураційний файл у форматі YAML, який описує апаратну конфігурацію пристрою, підключені компоненти, мережеві налаштування та бажану поведінку системи. Цей файл є повністю декларативним, тобто описує що повинен робити пристрій, а не як саме це має бути реалізовано на рівні коду. Такий підхід дозволяє людям без глибоких знань програмування створювати функціональні IoT-пристрої, просто описуючи їх бажану конфігурацію [19, 20].
Процес роботи з ESPHome включає кілька етапів, які автоматизовані та приховані від користувача. Спочатку інструмент командного рядка або веб-інтерфейс ESPHome зчитує YAML-конфігурацію та валідує її на предмет синтаксичних помилок та несумісних налаштувань. Після успішної валідації ESPHome використовує систему шаблонів для генерації C++ коду на основі конфігурації, включаючи необхідні бібліотеки для роботи з датчиками, актуаторами, мережевими протоколами та іншими компонентами. Згенерований код оптимізується для апаратної платформи та компілюється за допомогою PlatformIO, створюючи бінарний файл прошивки [20].
Отриманий бінарний файл завантажується на ESP пристрій одним з двох способів: через USB-підключення при першому завантаженні або

через мережу за допомогою механізму over-the-air updates для всіх наступних оновлень. OTA-оновлення є однією з ключових переваг ESPHome, оскільки дозволяє оновлювати прошивку пристроїв без фізичного доступу до них, що критично важливо для пристроїв, встановлених у важкодоступних місцях або для систем з великою кількістю сенсорних вузлів. Після завантаження прошивки пристрій автоматично підключається до налаштованої Wi-Fi мережі та починає виконувати запрограмовану функціональність.
Внутрішня архітектура згенерованого коду базується на компонентній системі, де кожен елемент функціональності реалізований як окремий компонент з чітко визначеним інтерфейсом. ESPHome включає понад сто п'ятдесят готових компонентів для роботи з різноманітними датчиками, дисплеями, світлодіодами, реле, моторами та іншими пристроями. Кожен компонент є самодостатнім модулем, який інкапсулює логіку роботи з відповідним апаратним пристроєм, включаючи ініціалізацію, читання даних, обробку помилок та комунікацію з іншими компонентами системи. Така модульна архітектура забезпечує високу гнучкість та можливість розширення функціональності через додавання нових компонентів [20, 21].
Бібліотека компонентів ESPHome охоплює практично всі популярні датчики та пристрої, що використовуються в IoT-проектах. Для вимірювання параметрів навколишнього середовища доступні компоненти для **датчиків температури та** вологості DHT 11, DHT22, AM2320, SHT3x, BME280, BME680, DS18B20 та багатьох інших. Кожен компонент підтримує специфічні особливості відповідного датчика, включаючи різні режими точності, частоти опитування, калібрування та компенсації помилок. Для датчика BME680, наприклад, реалізована підтримка вимірювання якості повітря через обчислення індексу IAQ на основі вмісту летких органічних сполук [21].
Для промислових застосувань ESPHome підтримує роботу з аналоговими входами для вимірювання напруги та струму, датчиками тиску, витратомірами, енкодерами для визначення положення та швидкості обертання. Компоненти для роботі з шинами Modbus RTU та Modbus TCP дозволяють інтегрувати ESPHome-пристрої з промисловим обладнанням, що використовує стандартні протоколи автоматизації. Підтримка інтерфейсів RS485, CAN bus розширює можливості використання платформи в складних розподілених системах управління та моніторингу [22, 23].

Рисунок 1.3 - Головна сторінка Home Assistant

Мережева функціональність ESPHome включає не тільки базове підключення до Wi-Fi, а й розширені можливості налаштування. Підтримуються статична та динамічна IP-адресація, налаштування DNS серверів, використання hostname для доступу до пристрою в локальній мережі. Компонент captive portal автоматично створює точку доступу Wi-Fi для первинного налаштування пристрою через веб-інтерфейс, коли пристрій не може підключитися до налаштованої мережі. Це особливо корисно при масовому розгортанні пристроїв або при зміні параметрів Wi-Fi мережі [23].
Функції діагностики та моніторингу включають компоненти для відстеження внутрішнього стану пристрою: температура чіпу, вільна пам'ять, сила Wi-Fi сигналу, час роботи, причина останнього перезавантаження. Система логування дозволяє виводити діагностичну інформацію через UART, мережу або на підключений дисплей з різними рівнями деталізації від помилок до детального трасування виконання коду. Веб-сервер, вбудований у кожен ESPHome пристрій, надає простий веб-інтерфейс для перегляду стану всіх сенсорів та управління перемикачами без необхідності використання додаткових інструментів.
Нативна інтеграція з Home Assistant є однією з найсильніших сторін ESPHome, що робить його ідеальним вибором для побудови систем домашньої автоматизації. Після першого запуску ESPHome пристрою в мережі він автоматично виявляється Home Assistant через протокол mDNS та з'являється в інтерфейсі як нова інтеграція, готова до додавання одним кліком. Після підтвердження додавання всі сенсори, перемикачі, світильники та інші сутності з ESPHome пристрою автоматично створюються в Home Assistant з правильними типами, одиницями вимірювання та іконками. Це радикально спрощує процес налаштування порівняно з ручним створенням сутностей через конфігураційні файли [15, 16, 24].
Двостороння комунікація між ESPHome та Home Assistant відбувається в реальному часі через постійне TCP-з'єднання з автоматичним відновленням при розривах. Зміни стану сенсорів миттєво передаються в Home Assistant, а команди управління з Home Assistant так само швидко виконуються на пристрої. Затримка від натискання кнопки в інтерфейсі Home Assistant до спрацювання реле на ESPHome пристрої зазвичай не перевищує ста мілісекунд у локальній мережі. Така швидкість відгуку робить систему придатною для критичних застосувань типу керування освітленням або охоронних систем, де затримка повинна бути мінімальною [15, 16, 25].

Рисунок 1.4 - Вікно завантаження плагіна ESPhome

ESPHome підтримує створення власних компонентів через систему custom components, що дозволяє розширити функціональність платформи для роботи з нестандартним обладнанням або реалізації специфічної логіки. Користувач може написати клас на C++, який успадковує базові класи ESPHome компонентів, та включити його в конфігурацію через директиву esphome includes. Такі кастомні компоненти мають доступ до всієї функціональності ESPHome framework, включаючи систему подій, логування, мережеві функції, та можуть бути параметризовані через YAML конфігурацію як стандартні компоненти [16, 26].
Лямбда-функції у ESPHome дозволяють вставляти фрагменти C++ коду безпосередньо в YAML конфігурацію для реалізації простої логіки без створення повноцінних кастомних компонентів. Лямбди можуть використовуватися для обчислення значень на основі інших сенсорів, трансформації даних, реалізації складних умов в автоматизаціях, створення власних ефектів освітлення. Наприклад, лямбда може обчислювати точку роси на основі температури та вологості, конвертувати одиниці вимірювання, фільтрувати викиди значень датчиків [15, 27].

## 2 МЕТОДИ РЕАЛІЗАЦІЇ ЗБОРУ ІНФОРМАЦІЇ З ОБ’ЄКТІВ

Після детального вивчення фундаментальних концепцій, характеристик та підходів у сфері збору інформації з розподілених об'єктів, доцільно перейти до практичного аналізу та обґрунтування вибору методології реалізації системи моніторингу. У цьому розділі досліджується застосування мікроконтролерних платформ як базових елементів для побудови систем збору даних, визначається їх роль та потенціал у контексті розподіленого моніторингу. Особливу увагу приділено детальному розгляду архітектури системи збору інформації, виділенню основних функціональних блоків та їх взаємодії, а також обґрунтуванню обраних методів вимірювання параметрів моніторингу.

### 2.1 Ефективність мікроконтролерів для збору інформації з розподілених об'єктів

Мікроконтролери є основою сучасних систем збору інформації з розподілених об'єктів завдяки своїй універсальності, низькій вартості, компактності та можливості автономної роботи. На відміну від повноцінних комп'ютерних систем типу одноплатних комп'ютерів Raspberry Pi, мікроконтролери споживають значно менше енергії, мають менші розміри та можуть працювати в жорсткіших умовах експлуатації. Для розподіленої системи моніторингу, де необхідно розгорнути десятки або сотні сенсорних вузлів у різних локаціях, використання мікроконтролерів є економічно виправданим та технічно оптимальним рішенням [25].
Архітектура мікроконтролерів ESP32 та ESP8266 спеціально оптимізована для IoT-застосувань. Вбудований Wi-Fi модуль дозволяє створювати бездротові сенсорні мережі без необхідності прокладання кабелів між вузлами та центральним сервером, що критично важливо для існуючих будівель або великих територій, де прокладання проводів є складним або неможливим. Підтримка протоколу 802.11 b/g/n забезпечує сумісність з будь-якою сучасною Wi-Fi інфраструктурою та дозволяє передавати дані зі швидкістю до ста п'ятдесяти мегабіт за секунду, чого більш ніж достатньо для передачі даних від датчиків навіть при високій частоті опитування [24, 25].
ESP32 додатково включає модуль Bluetooth та Bluetooth Low Energy, що розширює можливості комунікації в межах коротких відстаней. BLE особливо корисний для створення mesh-мереж, де пристрої можуть ретранслювати дані один одному для розширення покриття в умовах, де прямий Wi-Fi зв'язок з центральним шлюзом неможливий. Також BLE дозволяє створювати локальні мікромережі датчиків з дуже низьким енергоспоживанням, де периферійні BLE-датчики передають дані до центрального ESP32 вузла, який агрегує інформацію та відправляє її через Wi-Fi до сервера [26]. Така ієрархічна архітектура дозволяє оптимізувати енергоспоживання та збільшити кількість моніторингових точок без перевантаження Wi-Fi мережі.

INCLUDEPICTURE "https://devdotnet.org/wp-content/uploads/esp32-wroom-wifi-devkit-v1_pinout_36.jpg" * MERGEFORMATINET

Рисунок 2.1 - Схема контроллера ESP32

Обчислювальна потужність ESP32 з двоядерним процесором на частоті до двохсот сорока мегагерц дозволяє виконувати локальну обробку даних безпосередньо на сенсорному вузлі. Це реалізує концепцію edge computing, коли первинна обробка, фільтрація, агрегація та аналіз даних відбуваються на периферії системи, а не на центральному сервері. Наприклад, замість відправки кожного індивідуального виміру датчика температури кожні п'ять секунд, мікроконтролер може обчислювати середнє, мінімальне та максимальне значення за хвилину та відправляти тільки ці агреговані дані. Це зменшує навантаження на мережу та центральний сервер у десятки разів при збереженні корисної інформації про динаміку параметрів.
Локальна обробка даних також дозволяє реалізувати алгоритми виявлення аномалій та критичних ситуацій безпосередньо на сенсорному вузлі. Мікроконтролер може бути запрограмований на відправку сповіщень тільки при виході параметрів за встановлені межі або при виявленні швидких змін, що свідчать про аварійну ситуацію. Наприклад, датчик температури в серверній кімнаті може відправляти дані раз на хвилину в нормальному режимі, але перейти на відправку кожні п'ять секунд та генерувати тривожне сповіщення при перевищенні критичної температури. Така адаптивна поведінка зменшує загальний трафік даних при збереженні швидкої реакції на критичні події [25, 26].

## 2.2 Система моніторингу об'єктів з використанням датчиків

Мікроконтролери дають змогу під’єднувати зовнішні датчики або використовувати вбудовані аналого-цифрові перетворювачі (АЦП), що забезпечують точне перетворення аналогових сигналів, такі як напруга, темпаратура, вологість, рівент води у резервуарі, освітлення, тощо у цифрову форму.
Ефективність системи збору інформації з розподілених об'єктів значною мірою визначається правильним вибором та організацією датчиків, які забезпечують первинне перетворення фізичних величин у електричні сигнали для подальшої обробки мікроконтролером. Сучасний ринок пропонує величезну різноманітність датчиків для вимірювання практично будь-яких параметрів навколишнього середовища, стану обладнання або технологічних процесів. Побудова ефективної системи моніторингу вимагає комплексного підходу до вибору типів датчиків, їх розміщення, підключення та обробки отриманих даних [24, 25, 26].
Датчики температури є одними з найпоширеніших у системах моніторингу завдяки універсальності цього параметру для оцінки стану різноманітних об'єктів. Для вимірювання температури повітря широко використовуються інтегровані цифрові датчики типу DHT22 (рисунок 2.1), які поєднують вимірювання температури та вологості в одному корпусі. забезпечує діапазон вимірювання температури від мінус сорока до плюс вісімдесяти градусів Цельсія з точністю плюс-мінус нуль цілих п'ять десятих градуса та вологості від нуля до ста відсотків з точністю плюс-мінус два-п'ять відсотків. Комунікація з мікроконтролером відбувається через простий однопровідний інтерфейс, що спрощує підключення та дозволяє використовувати довгі з'єднувальні проводи до п'ятдесяти метрів [26, 27, 28].

Рисунок 2.1 - Датчик DHT22

Для вимірювання температури рідин, поверхонь або ґрунту оптимальним вибором є датчики DS18B20 (рисунок 2.2) на базі цифрового термометра з інтерфейсом 1-Wire. Основною перевагою цих датчиків є можливість підключення множини датчиків до одного GPIO виходу мікроконтролера через паралельне з'єднання на шині 1-Wire. Кожен датчик має унікальний 64-бітний ідентифікатор, що дозволяє мікроконтролеру індивідуально адресувати та зчитувати покази кожного датчика. Датчики випускаються в різних корпусах, включаючи водонепроникні зонди в нержавіючому корпусі для занурення в рідини або вбудовування в ґрунт. Діапазон вимірювання від мінус п'ятдесяти п'яти до плюс ста двадцяти п'яти градусів Цельсія з програмованою роздільною здатністю від дев'яти до дванадцяти біт робить DS18B20 універсальним рішенням для багатьох застосувань [27, 28].

Рисунок 2.2 - Датчик DS18B20

Датчики освітленості дозволяють системі моніторингу адаптувати роботу залежно від природного освітлення або контролювати роботу систем штучного освітлення. Прості фоторезистори забезпечують аналоговий вихід, пропорційний інтенсивності світла, але мають нелінійну характеристику та залежність від температури. Більш точні вимірювання забезпечують спеціалізовані датчики освітленості типу BH1750, які виводять цифрові дані в люксах через інтерфейс I2C. Датчик BH1750 автоматично адаптує час інтеграції залежно від рівня освітленості, забезпечуючи діапазон вимірювання від одного до шістдесяти п'яти тисяч п'ятисот тридцяти п'яти люксів з роздільною здатністю один люкс.

Спектральна чутливість датчика близька до чутливості людського ока, що робить покази придатними для систем керування освітленням орієнтованих на комфорт людини [25, 28].

INCLUDEPICTURE "https://arduino.ua/products_pictures/medium_datchik-osveshhennosti-cifrovoi-1.jpg" * MERGEFORMATINET

Рисунок 2.3 - Датчик BH1750

Датчики якості повітря набувають все більшої популярності для моніторингу житлових та офісних приміщень, особливо після підвищення уваги до якості повітря у контексті пандемії. Датчики CO2 типу MH-Z19 (рисунок 2.4) або SCD30 вимірюють концентрацію вуглекислого газу методом недисперсійної інфрачервоної спектроскопії, забезпечуючи точність вимірювання у межах плюс-мінус тридцять-п'ятдесят ppm. Підвищена концентрація CO2 понад тисячу ppm вказує на недостатню вентиляцію приміщення та може призводити до зниження когнітивних функцій, втоми, головного болю. Системи моніторингу можуть автоматично активувати вентиляцію або сигналізувати про необхідність провітрювання при перевищенні порогових значень концентрації CO2.

Рисунок 2.4 - Датчики CO2 MH-Z19

Датчики руху та присутності забезпечують можливість автоматизації систем освітлення, опалення, вентиляції залежно від присутності людей у приміщенні. Пасивні інфрачервоні датчики PIR детектують рух теплих об'єктів у полі зору датчика через зміну інфрачервоного випромінювання. Ці датчики є недорогими, споживають мало енергії, мають широкий кут огляду, але не можуть детектувати нерухомого об'єкта та схильні до помилкових спрацювань від змін температури або руху тварин. Мікрохвильові датчики радіолокаційного типу детектують рух через ефект Допплера відбитих радіохвиль та можуть працювати через неметалічні перешкоди типу стін або меблів. Комбінація PIR та мікрохвильового датчиків у двотехнологічних сенсорах знижує ймовірність помилкових спрацювань через вимогу одночасної детекції обома методами [24, 28].
Датчики струму та напруги дозволяють моніторити споживання електроенергії обладнанням для оптимізації енергоспоживання, виявлення несправностей, розподілу витрат. Безконтактні датчики струму на базі ефекту Холла або трансформаторів струму встановлюються на проводи без розриву електричного кола, вимірюючи магнітне поле навколо провідника. Датчики напруги можуть використовувати резистивні дільники для зменшення вимірюваної напруги до рівня, придатного для АЦП мікроконтролера, або ізольовані вимірювальні модулі для гальванічної розв'язки та безпеки [28].

2.3 Використання ОС Home assistant та ESPhome

Home Assistant є відкритою операційною системою для домашньої автоматизації, яка забезпечує централізоване управління, автоматизацію та моніторинг розподілених IoT-пристроїв у реальному часі. На відміну від простих систем збору даних, що лише агрегують інформацію від датчиків, Home Assistant надає повноцінну платформу для створення інтелектуальних систем, здатних автоматично реагувати на зміни стану пристроїв, виконувати складні сценарії автоматизації, візуалізувати дані через налаштовувані dashboard, інтегруватися з сотнями різних пристроїв та сервісів [15, 16].
Home Assistant та ESPHome - обидва інструменти з можливістю впровадження та налаштування систем, таких як розумний будинок.
Home Assistant - це система керування розумним будинком з відкритим програмним кодом, яка об’єднує різноманітні пристрої та сервіси в одну централізовану екосистему. Платформа має гнучкі налаштування та дає змогу реалізовувати складні сценарії автоматизації [15, 16].
ESPHome - це засіб для розроблення широкого спектра пристроїв і сенсорів на базі доступних Wi-Fi мікроконтролерів ESP8266 та ESP32 без необхідності глибоких знань програмування. Він використовує конфігураційні файли у форматі YAML для генерації індивідуального прошивання, яке встановлюється на ESP-пристрій. Усі компоненти, визначені в конфігурації ESPHome, автоматично інтегруються та відображаються в інтерфейсі Home Assistant.
Home Assistant надає розширені можливості для кастомізації та використання додаткових модулів. Платформа дозволяє створювати автоматизовані сценарії керування пристроями залежно від заданих умов і подій, а також налаштовувати систему сповіщень і інтегрувати сторонні сервіси, зокрема Google Assistant та Alexa [15, 16].
Можливості їх використання вже було вказано у попередніх розділах, від простих систем до розумного будинку, контроль в аграрному горподарстві тощо.
Процес встановлення Home Assistant є доволі простим, однак важливо дотримуватися офіційних інструкцій, наведених на вебсайті розробників. Платформа підтримує інсталяцію на різних середовищах, зокрема на Raspberry Pi, у контейнері Docker або у вигляді віртуальної машини.
Наприклад пропонується кілька видів встановлення Home Assistant [15]:
1. Home Assistant Operating System - спеціалізована мінімальна **операційна система, оптимізована для роботи Home** Assistant, яка містить Supervisor **для** керування системою та **додатковими компонентами.** Це рекомендований варіант встановлення;
**2. Home Assistant** Container - запуск Home **Assistant Core** у контейнеризованому середовищі, наприклад із використанням Docker;
3. Home Assistant Supervised - ручне встановлення з використанням Supervisor;
4. Home Assistant Core - інсталяція шляхом ручного розгортання у віртуальному середовищі Python.
На рисунку 2.5 наведено можливі методи завантаження в залежності яку платформу ти пристрій використовувати.

Рисунок 2.5 - Метод завантаження

Ключовою особливістю Home Assistant є можливість об’єднання пристроїв від різних виробників в одну систему. Платформа підтримує велику кількість інтеграцій із різноманітними брендами та типами обладнання, зокрема смарт-освітленням, термостатами, камерами відеоспостереження, датчиками руху, розумними розетками та іншими пристроями. Завдяки цьому користувач може побудувати єдину централізовану систему керування всіма компонентами розумного будинку [15, 16, 28].
Отже, після детального аналізу можливих варіантів реалізації систем та з урахуванням переваг і недоліків розглянутих методів, можна визначити оптимальний підхід для побудови системи збору інформації. Вибір слід робити з урахуванням таких критеріїв, як простота інтеграції різних пристроїв, масштабованість системи, можливість налаштування автоматизацій та надійність роботи.
У межах даної роботи як експериментальне рішення обрано поєднання Home Assistant і ESPHome із використанням мікроконтролера ESP32.

Такий підхід дозволяє створити централізовану систему управління розподіленими пристроями, забезпечує підтримку великої кількості протоколів і API, а також дає змогу швидко налаштовувати автоматизації та сповіщення. Використання ESP32 у поєднанні з ESPHome спрощує створення та інтеграцію різноманітних датчиків і виконавчих пристроїв без необхідності глибоких знань програмування.
Крім того, такий підхід є економічно ефективним і гнучким: нові пристрої можна легко додавати до системи, змінювати сценарії роботи та інтегрувати сторонні сервіси, такі як голосові помічники або хмарні платформи. Це робить обране рішення придатним як для лабораторних експериментів і навчальних проектів, так і для реального використання у побуті або на виробництві [28].

# 3 ТЕХНІЧНА ТА ПРОГРАМНА ЧАСТИНА

## 3.1 Контроллер ESP32 LOLIN32

У межах даного проекту було використано мікроконтроллер ESP32 LOLIN32 (рисунок 3.1). Це компактна плата розробки на базі мікроконтролера ESP32, яку випускає компанія LOLIN (Wemos). Вона популярна у проєктах IoT, автоматизації та навчанні, зокрема добре підходить для роботи з ESPHome і Home Assistant [16, 29].

Рисунок 3.1 - ESP32 LOLIN32

Основні характеристики:
1. мікроконтролер: ESP32 (Xtensa Dual-Core, до 240 МГц);
2. бездротові інтерфейси: Wi-Fi 802.11 b/g/n, Bluetooth 4.2;
3. Flash-пам’ять: зазвичай 4 МБ;
4. живлення: через USB або 5V / 3.3V пін;
5. USB-UART: CP2104 або CH340.
Даний девайс є сучасною та універсальною платформою для розроблення IoT-систем і систем збору інформації з розподілених об’єктів. Завдяки наявності вбудованих модулів Wi-Fi та Bluetooth, високій обчислювальній потужності, широкому набору інтерфейсів і сумісності з популярними середовищами розробки, зокрема ESPHome, дана плата забезпечує ефективне підключення датчиків і керування виконавчими пристроями. Компактні розміри, низька вартість і надійність роботи роблять ESP32 LOLIN32 доцільним вибором для навчальних, дослідницьких і практичних проєктів у сфері автоматизації та моніторингу речей, враховуючи його схему (Додаток Б).

Рисунок 3.2 - Батарея

У вікні проекту «Edit» прописується код для контроллера (Додаток А). Після збереження змін відбувається підключення девайсу до комп’ютера (рисунок 3.3).
Скрипти у ESPHome є іменованими послідовностями дій, які можуть викликатися з автоматизацій, через HTTP API або з інших скриптів. Скрипти підтримують параметри, локальні змінні, умовні оператори, цикли, затримки, паралельне та послідовне виконання. Це дозволяє створювати складні сценарії поведінки, які можуть повторно використовуватися в різних контекстах. Наприклад, скрипт може реалізувати послідовність увімкнення-вимкнення кількох пристроїв з затримками, що викликається при натисканні кнопки або за розкладом [15, 30].

## 3.2 Проект у ESPhome

Рисунок 3.3 - Підключення контроллера

Програма повідомляє про успішне завершення підключення пристрою до комп’ютера (рисунок 3.4).
Інтервальні компоненти дозволяють виконувати дії через певні проміжки часу, що корисно для періодичного опитування датчиків, відправки статистики, виконання профілактичних операцій. Компонент time забезпечує синхронізацію часу через NTP сервери та дозволяє створювати автоматизації на основі часу доби, дня тижня, дати. Можна налаштувати виконання дій на схід та захід сонця з урахуванням географічних координат пристрою, що корисно для систем керування освітленням. Підтримка часових зон та автоматичного переходу на літній час забезпечує коректну роботу в різних регіонах [16, 31].

Рисунок 3.4 - Успішне підключення

Система шаблонів у ESPHome дозволяє створювати віртуальні сенсори, що обчислюють свої значення на основі інших сенсорів або зовнішніх даних. Template sensor може агрегувати дані з кількох фізичних датчиків, виконувати математичні обчислення, застосовувати фільтри згладжування або усереднення. Template binary sensor може комбінувати стани кількох бінарних сенсорів через логічні операції. Template switch створює віртуальний перемикач, дії якого визначаються лямбда-функціями, що дозволяє реалізувати складну логіку управління без прив'язки до конкретного апаратного виходу. Щоб пристрій працював із Home assistant його потрібно інтегрувати (рисунку 3.5) [15, 31].

Рисунок 3.5 - Індеграція з Home Assistant

Після наявності подачі живлення відповідно буде показано повідомлення у Home Assistant (рисунок 3.6).

Рисунок 3.6 - Присутня подача 220 V

Аналогічно після відключення від енергії буде зворотнє повідомлення про відсутність потоку електрики (рисунок 3.7).

Рисунок 3.7 - Відсутня подача 220 V

На останок було проведено налаштування автоматизації для отримання повідомлення про відтсутність електрики у мережі. Якщо статус змінюється з підключено на відключено то думаємо що мережа відключилася (рисунок 3.8).

Рисунок 3.8 - Налаштування автоматизації

Отже у рамках практичної реалізації системи збору інформації з розподілених об'єктів з використанням ESPHome було розроблено та впроваджено експериментальний пристрій для моніторингу стану електропостачання на базі мікроконтролера ESP32 LOLIN32 з автономним живленням від літій-іонної батареї [16, 32, 33].
Для детекції наявності або відсутності основного електропостачання було реалізовано моніторинг напруги живлення через аналоговий вхід ESP32, який контролює напругу на вході USB або зовнішнього адаптера живлення. Коли напруга присутня, бінарний сенсор знаходиться в стані "увімкнено", а при зникненні напруги перемикається в стан "вимкнено", що генерує подію зміни стану.

ВИСНОВКИ

У дипломній роботі проведено комплексне дослідження, спрямоване на розроблення системи збору інформації з розподілених об'єктів з використанням платформи ESPHome. В результаті виконаної роботи отримано наступні результати та зроблено такі висновки. Проведено детальний аналіз сучасних систем збору інформації з розподілених об'єктів, який показав, що існуючі рішення охоплюють широкий спектр застосувань від промислових SCADA-систем до домашніх систем автоматизації. Встановлено, що традиційні промислові системи характеризуються високою вартістю впровадження, складністю масштабування та залежністю від propriet арних рішень, що обмежує їх використання для невеликих та середніх застосувань. Системи розумного будинку на базі відкритих платформ демонструють значно більшу гнучкість та економічну ефективність при достатній функціональності для більшості практичних завдань.
У рамках практичної реалізації системи збору інформації з розподілених об'єктів з використанням ESPHome було розроблено та впроваджено експериментальний пристрій для моніторингу стану електропостачання на базі мікроконтролера ESP32 LOLIN32 з автономним живленням від літій-іонної батареї. Цей пристрій демонструє ключові принципи побудови розподілених систем збору інформації та підтверджує практичну застосовність обраної платформи ESPHome для вирішення реальних задач моніторингу.

ПЕРЕЛІК ДЖЕРЕЛ


1. **Tanenbaum, A. S., & Wetherall, D. J.** (2021). Computer Networks (6th ed.). Pearson Education.
2. **Олифер, В. Г., & Олифер, Н.** А. (2016). **Компьютерные сети. Принципы, технологии, протоколы (5-е** изд.). **Питер.**
3. Pureswaran, V., & Lougee, R. (2015). **The Internet of Things: Making Sense of the Next Mega-Trend.** IBM Institute for Business Value.
4. Шеремет, О. І., & Садовий, К. В. (2019). Інтернет речей: технології та застосування. Київ: КПІ ім. Ігоря Сікорського.
5. Maksimović, M. (2017). The Role of Green Internet of Things (G-IoT) in Transforming Cities into Smart and Sustainable Environments. Springer.
6. Гриценко, В. І., & Бодянський, Є. В. (2020). Інтелектуальні системи інтернету речей для промислових застосувань. Кібернетика та обчислювальна техніка, 2(200), 5-18.
7. Петренко, А. І., & Ладанюк, А. П. (2018). Розподілені системи збору та обробки даних в промислових мережах. Автоматизація технологічних і бізнес-процесів, 10(3), 4-11.
8. Kononenko, O., & Kuchuk, H. (2019). Development of a method for reducing the power consumption of IoT devices. Eastern-European Journal of Enterprise Technologies, 5(9), 50-57.
9. Schwartz, M., & Olivier, B. (2016). Building the Internet of Things with Arduino and ESP8266. IEEE Potentials, 35(5), 19-23.
10. Колпакова, Т. О., & Ловейкін, А. В. (2021). Архітектура розподілених систем моніторингу на базі ESP32. Вісник НТУУ "КПІ". Серія Приладобудування, (61), 42-50.
11. Мокін, Б. І., & Слободянюк, О. В. (2019). Застосування мікроконтролерів ESP для побудови IoT-систем. У Тези доповідей XII Міжнародної науково-практичної конференції "Сучасні інформаційні технології" (с. 125-127). Київ.
12. Кулик, А. Я., & Лупенко, С. А. (2020). Протоколи передачі даних у розподілених IoT-системах. У Матеріали XV Міжнародної конференції "Контроль і управління в складних системах" (с. 89-92). Вінниця.
13. Bianchi, V., Bassoli, M., & De Munari, I. (2018). IoT solutions for crop monitoring and irrigation automation. In 2018 IEEE International Conference on Environmental Engineering (pp. 1-5). IEEE.
14. **Gubbi, J., Buyya,** R., & Palaniswami, **M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems,** 29(7), 1645-1660.
15. ESPHome. (2023). ESPHome Documentation. Отримано з https://esphome.io/
16. Home Assistant. (2023). Home Assistant Documentation. Отримано з https://www.home-assistant.io/docs/
17. Espressif Systems. (2022). ESP32 Technical Reference Manual (Version 4.6). Shanghai: Espressif Systems.
18. Espressif Systems. (2021). ESP8266 Technical Reference Manual (Version 1.7). Shanghai: Espressif Systems.
19. MQTT.org. (2023). MQTT Version 5.0 Specification. OASIS Standard.
20. Повхан, І. Ф. (2019). Методи та засоби побудови розподілених інформаційних систем реального часу (Автореф. дис. д-ра техн. наук). Тернопільський національний технічний університет, Тернопіль.
21. Кучук, Г. А. (2018). Метод мінімізації затримки передачі даних в мережі підтримки хмарного сервісу (Дис. канд. техн. наук). Харківський національний університет радіоелектроніки, Харків.
22. Баклан, І. В., & Таран, В. М. (2020). Порівняльний аналіз платформ для розробки IoT-додатків. Вісник Хмельницького національного університету, 3(285), 202-207.
23. Савченко, Є. А., & Романенко, В. Д. (2019). Безпека даних у розподілених IoT-системах. Захист інформації, 21(3), 156-163.
24. Maier, A., Sharp, A., & Vagapov, Y. (2017). **Comparative analysis and practical implementation of the ESP32 microcontroller module for** IoT applications. Internet Technologies and Applications, 47-51.
25. Rodali, R. K., & **Soratkal, S.** R. ( 2016). **MQTT based home automation system using ESP8266.** Humanitarian Technology Conference, 1-5.
26. IEEE Std 802.11-2020. (2020). **IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems**

**Local and Metropolitan Area Networks-Specific Requirements** Part 11 Wireless LAN Medium **Access Control (MAC) and Physical Layer (PHY) Specifications.**
27. ISO/IEC 20922:2016. (2016). Information technology - Message Queuing Telemetry Transport (MQTT) v3.1.1.
28. ДСТУ ISO/IEC 27001:2015. (2016). Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги. Київ: ДП "УкрНДНЦ".
29. Петров, В. В., & Волков, О. Є. (2021). Енергоефективність бездротових сенсорних мереж на базі ESP32. Вісник Національного технічного університету "ХПІ", (1), 78-85.
30. Іванов, Ю. Б., & Приходько, С. Б. (2020). Методи інтеграції розподілених систем збору даних. Системні дослідження та інформаційні технології, (2), 95-104.
31. Glover, B., & Bhatt, H. (2020). RFID Essentials: Theory and Practice (2nd ed.). O'Reilly Media.
32. **Solaković, A., & Hadžialić, M. (2018). Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues. Computer Networks, 144, 17-39.**
33. Лісовець, О. О., & Шаповалова, О. А. (2019). Аналіз протоколів передачі даних для IoT-пристроїв. Системи обробки інформації, (3), 67-73.

Додатки

Додаток А

```
esphome:
  name: iot
  friendly_name: IoT
esp32:
  board: esp32dev
  framework:
    type: esp-idf
# Enable logging
logger:
# Enable Home Assistant API
api:
  encryption:
    key: "eK63wt0YlgXpZ/KePxey3LBCNhvU0n+bPChXx7WQEKI="
ota: - platform: esphome
  password: "**********"
wifi: ssid: !secret wifi_ssid password: !secret wifi_password # Enable fallback hotspot (captive portal) in case wifi connection fails ap: ssid: "Iot Fallback Hotspot"
    password: "**********"
captive_portal:
    web_server:
  port: 80
esp32_ble_tracker:
  scan_parameters:
    interval: 1100ms
    window: 1100ms
    active: true
bluetooth_proxy:
  active: true
binary_sensor:
platform: gpio
    pin: GPIO 22
    id: utility_ac
    name: Utility AC
    publish_initial_state: True
```

Додаток Б

)
(
Арк.
Зм.
Підпис
Дата
No докум.
М
Р.А
КС
м
-
0
1
1
.00.00.000 ПЗ
Арк.
1
)

**БІБЛІОГРАФІЧНА ДОВІДКА**

**Тема магістерської роботи:** «Розроблення системи збору інформації з розподілених об'єктів з використання ESPhome»
Обсяг пояснювальної записки: 56 аркуш.
Кількість рисунків: 18 шт.
Кількість таблиць: 2 шт.
"____"_____2025 р.
Підпис студента