Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Багатофункціональна мікроконтролерна система «домашній асистент» на базі протоколу MQTT
Назва теми

КвРКІ 200232.20.08.07 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-20-2 _____ М. В. Івахов
Підпис, Ініціали, прізвище

Керівник _____ В. В. Яцків
Підпис, дата, Ініціали, прізвище

Нормоконтролер _____ І.О. Засорнова
Підпис, дата, Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис, Ініціали, прізвище

«24» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень БАКАЛАВР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Зав. кафедри Т.О.Говорущенко

" 10 " 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Іваховому Максиму Володимировичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Багатофункціональна мікроконтролерна система «домашній асистент» на базі протоколу MQTT
Керівник проекту (роботи) Яцків В.В., д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз пристроїв розумного будинку та новітніх технологій що в них використовуються
Вибір апаратного та програмного забезпечення для створення мікроконтролерної системи розумного будинку
Програмно-апаратна реалізація мікроконтролерної системи «домашній асистент»

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Принципова схема модуля ESP32
Функціональна блокова діаграма модуля ESP32
Принципова схема модуля ESP32 з підключеним датчиком температури DHT22

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Засорнова І.О., доцент кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконано
3	Робота над розділом 1 – Аналіз пристроїв розумного будинку та новітніх технологій що в них використовуються	01.03.2024	виконано
4	Робота над розділом 2 – Вибір апаратного та програмного забезпечення для створення мікроконтролерної системи розумного будинку	01.04.2024	виконано
5	Робота над розділом 3 – Програмно-апаратна реалізація мікроконтролерної системи «домашній асистент»	29.04.2024	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконано
7	Попередній захист ВКР	26.05.2024	виконано
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ М. В. Івахов
Підпис Ініціали, прізвище

Керівник роботи _____ В. В. Яцків
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Багатофункціональна мікроконтролерна система «домашній асистент» на базі протоколу MQTT».

Автор роботи: Івахов Максим Володимирович.

Керівник роботи: Яцків Василь Васильович.

Пояснювальна записка: 68 с., 42 рисунки, 45 джерел.

Графічна частина: 3 креслення.

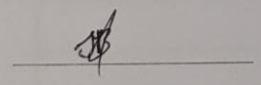
Мікроконтролер, розумний будинок, ESP, Wi-Fi, MQTT

Метою дипломної роботи є розробка програмного забезпечення для домашнього асистенту на базі протоколу звязку – MQTT.

Об'єктом дослідження – процеси функціонування системи управління розумним будинком.

Предметом дослідження – алгоритми та протоколи передачі даних в системі управління
розумним будинком.

Методами дослідження є систематичний огляд літератури для вивчення і аналізу предметної області даного дослідження та метод експериментування.



Підпис студента

30.05.2024

Дата

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 200232.20.02.08 ПЗ	Пояснювальна записка	68		
2		КвРКІ 200232.20.02.08 Е8	Лістинг коду	6		
			Графічні матеріали			
3		КвРКІ 200232.20.02.08 Е8	Принципова схема модуля ESP32	1		
4		КвРКІ 200232.20.02.08 Е8	Функціональна блокова діаграма модуля ESP32	1		
4		КвРКІ 200232.20.02.08 Е8	Принципова схема модуля ESP32 з підключеним датчиком температури DHT22	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 200232.20.02.08 ВП		
Розробив		Івахов			Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Яцків				У	1	1
Н. контр.		Засорнова				ХНУ, КІ2-20-2		
Затв.		Говорущенко		24.06				

ЗМІСТ

ВСТУП

Актуальність дослідження. Системи "розумного дому" стали популярними в сучасному суспільстві, оскільки допомагають заощадити час і гроші. Ці інноваційні розробки не лише підвищують зручність життя, а й покращують енергозбереження. Наприклад, освітлення в холодильнику працює тільки при відкритих дверцятах, що запобігає неефективному використанню електроенергії. Це особливо важливо з огляду на постійне зростання цін на енергоресурси.

Більшість сучасних пристроїв сумісні лише з контролюючими системами своїх виробників, що ускладнює інтеграцію пристроїв різних брендів в одну інфраструктуру. Універсальне рішення на базі єдиного мікроконтролера дозволить використовувати пристрої різних виробників та саморобні девайси в рамках однієї системи управління, яка підтримує єдиний протокол з периферією.

Сьогодні сфера автоматизації завдань у системах контролю розумного будинку набуває все більшої популярності. Зростає кількість людей, які встановлюють такі системи у своїх оселях для виконання повсякденних завдань, економії ресурсів та підвищення рівня безпеки. Це призводить до стрімкого розвитку ринку розумних пристроїв. Інженери постійно впроваджують нові протоколи та вдосконалюють існуючі, приділяючи велику увагу технологіям IoT.

Однак, незважаючи на значний попит, недосвідченим користувачам важко знайти комплексні рішення від одного виробника. Це змушує їх використовувати пристрої різних брендів і різне програмне забезпечення для їх управління, що значно ускладнює користувацький досвід та вимагає встановлення багатьох утиліт на персональні комп'ютери та смартфони.

Для вирішення цієї проблеми на ринку з'явилися уніфіковані глобальні системи розумного будинку, які дозволяють об'єднати різні пристрої в одну екосистему і керувати ними з одного місця. Виробники апаратного забезпечення змушені адаптувати свої продукти до вимог цих систем для забезпечення інтеграції, але більшість пристроїв поки що не мають такої можливості. Це змушує

користувачів самостійно розбиратися з процесами інтеграції або звертатися до розробників платформ для реалізації необхідного функціоналу.

У роботі представлені пропозиції щодо покращення архітектури розумних пристроїв шляхом абстрагування від систем управління та перенесення процесів інтеграції на інструменти платформ за допомогою використання стандартних протоколів та методів обміну даними. Використання стандартизованих методів і конвенцій, підтримуваних більшістю систем, дозволяє швидко інтегрувати пристрої у будь-які системи.

1 АНАЛІЗ ПРИСТРОЇВ РОЗУМНОГО БУДИНКУ ТА НОВІТНІХ ТЕХНОЛОГІЙ ЩО В НИХ ВИКОРИСТОВУЮТЬСЯ

1.1 Аналіз структурних і функціональних особливостей системи розумного будинку

Сьогодні домашня автоматизація дуже популярна. Користувачі можуть легко керувати побутовою технікою, освітленням, опаленням, розетками, мультимедіа та безпекою через ПК, планшет або смартфон. Вони можуть додавати датчики та інтелектуальні пристрої без змін в існуючій установці. Ринок розумного дому став доступним завдяки компаніям, таким як Apple (HomeKit), Samsung (SmartThings), Google (Alexa) та Siemens (KNX).

Поширення датчиків та впровадження IoT (Інтернет речей) створили потребу в домашній мережі для їх використання. Тому була створена модель, показана на рисунку 1.1. Існують різні варіанти встановлення: провідні, бездротові або комбіновані. Популярні протоколи включають Bluetooth, Wi-Fi, GSM, ZigBee, ZWave, KNX та менш популярні: EnOcean, Insteon, LowPAN, Thread, DASH, LoRA та Sigfox.

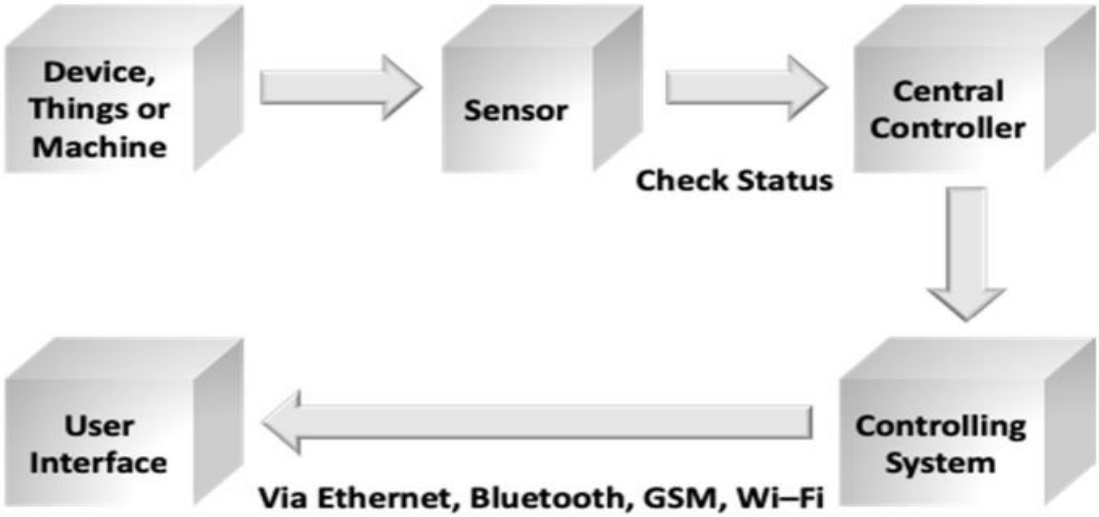


Рисунок 1.1 - Internet of Things технології для домашньої автоматизації

На рисунку представлено IoT технології для домашньої автоматизації. Основними елементами є Things or Machine - девайс чи пристрій, Sensor – датчик, Check status - перевірка статусу, Central Controller - центральный контроллер, Controlling System - система контролю, протокол передачі данних - Via Ethernet, Bluetooth, GSM, Wi-Fi та інтерфейс користувача User Interface.

Різноманітність варіантів ускладнює вибір, оскільки кожна технологія має свої переваги та недоліки. Іноді в одному будинку використовуються кілька технологій, що змушує користувачів використовувати декілька контролерів або додатків для управління.

1.2 Провідні технології

Для обміну даними між пристроями необхідно визначити протокол, який встановлює стандарти, політики, процедури та формати для передачі даних. Деякі протоколи спеціально розроблені для домашньої автоматизації, тоді як інші походять з існуючих комунікаційних і мережевих протоколів, таких як Wi-Fi, Bluetooth, GSM. Крім того, є протоколи, що використовуються в IoT (Інтернеті речей), які також можуть бути застосовані в системах домашньої автоматизації. Щороку з'являються нові протоколи або їх варіанти, а деякі, добре зарекомендувавши себе протягом багатьох років, з часом отримують численні поліпшення. У наступних розділах подано короткий огляд найпопулярніших протоколів.

BACnet. BACnet – це протокол для автоматизації будівель і мереж управління. Його вперше розробив комітет, сформований Американським товариством інженерів з опалення, охолодження та кондиціювання повітря (ASHRAE), з метою створення інтероперабельних систем домашньої автоматизації. Специфікація протоколу складається з трьох основних частин:

1. Представлення обладнання автоматизації будівель у стандартному вигляді.

2. Визначення форматів повідомлень, які можуть бути відправлені в комп'ютерній мережі для моніторингу.
3. Контроль обладнання і визначення типів локальних мереж (LAN), які можуть використовуватися для зв'язку BACnet.

Загалом існує 35 типів повідомлень (сервісів), розділених на п'ять класів: доступ і управління властивостями об'єктів, сигналізації та події, завантаження та скачування файлів, управління віддаленими пристроями, функції віртуального терміналу [2].

LonWorks. Local Operation Networks – це платформа, розроблена компанією Echelon Corporation у 1991 році для підтримки керуючих додатків. Кожен пристрій включає нейронний чіп, трансивер і прикладну електроніку. Мета цієї технології – поділ пристроїв або систем на групи інтелектуальних елементів (вузлів), пов’язаних засобами передачі даних (кручена пара, Ethernet), створюючи розумну мережу. LonWorks використовує модель OSI, надаючи послуги на всіх семи рівнях [3].

KNX (KONNEX). KNX – це відкритий стандарт для будинків і будівель, що використовує різні середовища передачі даних, такі як кручена пара (TP), лінія електропередач (PL), радіочастота (RF) і Ethernet. Він поєднує та розширює три старих стандарти шини: EIB, EHS і BatiBUS. KNX дозволяє з’єднувати різнорідні шинні пристрої незалежно від середовища, використовуючи модель OSI з об’єднанням трьох верхніх рівнів в один [5].

Ethernet. Ethernet – одна з найпопулярніших мережевих технологій, що використовується у комп’ютерних мережах, VoIP, автоматизації будинків, IoT та мобільному зв’язку. Вона характеризується високою швидкістю, мультимедійністю, стандартизованістю (IEEE 802.3), сумісністю (plug-and-play) та зворотною сумісністю. Ethernet використовує протокол TCP/IP, який забезпечує можливість підключення до 2^32 пристроїв і підтримує передачу енергії через ту ж саму середу [6].

X10**.** X10 – це протокол для дистанційного керування пристроями через лінію електропередач у будинку. Він використовує унікальні ідентифікатори для передачі команд між передавачами та приймачами. Стандарт X10 визначає 256 адрес, 16 юніт кодів і 16 секторів. Його основна проблема – низька швидкість передачі даних, через що він використовується для управління освітленням, приладами та датчиками безпеки [7].

Insteon**.** Insteon – система, розроблена для заміни стандарту X10 з використанням бездротової інтеграції систем електромереж. Вона дозволяє пристроям співпрацювати незалежно від підключення до ліній електропередач або радіочастот. Insteon сумісний з пристроями X10 і підтримує подвійну сітчасту топологію для обміну повідомленнями. Він може автоматично підключати пристрої через універсальні повідомлення і використовує ту ж саму логіку, що й KNX та Ethernet, з п’ятьма рівнями OSI [8].

1.3 Безпровідні технології

Bluetooth**.** Bluetooth-технологія персональних мереж (PAN) для домашньої автоматизації має значну перевагу завдяки вбудованій підтримці в кожному смартфоні. Смартфони можуть виступати клієнтською стороною, тоді як серверна сторона потребує використання мікроконтролера, інтегрованого в домашній шлюз (HAN - Home Area Network). Клієнт і сервер обмінюються даними через протокол Bluetooth, що забезпечує бездротову передачу даних для додатків автоматизації. Для забезпечення безпеки використовується захист паролем, щоб лише аутентифіковані користувачі мали доступ до пристроїв.

З введенням стандарту Bluetooth 4.0 з'явилася нова реалізація BLE (Bluetooth Low Energy), яка дозволяє використовувати багато додатків для домашньої автоматизації завдяки низькому енергоспоживанню без втрат ефективності. BLE використовується багатьма розробниками IoT (інтернету речей), оскільки не потребує додаткових шлюзів. Однак, Bluetooth Smart, як його ще називають,

підтримував лише однорангові та зірочні топології зв'язку, що призвело до необхідності впровадження мережевої топології Mesh Network.

BLE працює в діапазоні ISM 2,4 ГГц, використовуючи частоти від 2402 МГц до 2480 МГц. Цей спектр розділений на кілька каналів зв'язку з простором 2 МГц між ними, що дозволяє мати 40 каналів. Три канали використовуються для уникнення перешкод з Wi-Fi мережею, залишаючи 37 каналів для з'єднання. Зв'язок між пристроями може бути оповіщувальним або орієнтованим на з'єднання. У першому режимі пристрій розсилає інформацію через канали сповіщення, а у другому – синхронізується з іншими пристроями через 37 каналів.

Bluetooth має кілька рівнів, що не повністю відповідають структурі моделі OSI. Другий рівень (Baseband) включає частину перших двох рівнів OSI, третій рівень Bluetooth охоплює деякі функції другого та третього рівнів OSI, а четвертий рівень – прикладний та рівень представлення, об'єднуючи всі функції чотирьох верхніх шарів OSI [10].

Wi-Fi**.** Wi-Fi-технологія, незважаючи на схожість з Bluetooth, має багато відмінностей. Вона працює на частоті 2,4 ГГц, використовуючи TCP/IP протокол для зв'язку та має схожу структуру OSI. Фізичний шар Wi-Fi використовує повітря як середовище передачі, а не мідну чи оптоволоконну кабелі.

Wi-Fi використовує переваги існуючого домашнього підключення до Інтернету, надаючи локальне бездротове з'єднання. Більшість мережевих пристроїв підтримують TCP/IP протокол, що дозволяє використовувати IPv6 для присвоєння унікальної IP-адреси кожному пристрою. Ця технологія підтримує взаємодію з іншими технологіями автоматизації будинку, забезпечуючи гнучкість та мобільність кінцевого користувача. Сервери використовують Wi-Fi для зв'язку з локальною мережею, а модулі апаратного інтерфейсу підключаються через TCP/IP або власний протокол.

Технології TCP/IP забезпечують функціональність на перших трьох рівнях OSI, забезпечуючи прямий зв'язок між пристроями та високу швидкість передачі даних. Universal Plug and Play (UPnP) дозволяє пристроям безперешкодно

з'єднуватися між собою, інтегруючи різні пристрої виробників. UPnP використовує веб-сервіси, такі як IP, TCP, UDP, HTTP, SOAP та XML, щоб забезпечити прозору мережу з автоматичним виявленням пристроїв.

UPnP пристрої мають дві ролі: керовані пристрої (КП) та контрольні точки (КТ). КП надають послуги, а КТ виявляють і управляють пристроями. Коли пристрій додається до мережі, він отримує IP-адресу та передає свої послуги на КТ. Взаємодія з протоколом UPnP базується на шести кроках: адресація, виявлення, опис, керування, підписка та презентація.

Домашній маршрутизатор повинен підтримувати UPnP для роботи сервісу. Архітектура UPnP включає стандартні протоколи керування пристроями (DCP), забезпечуючи зв'язок та взаємодію електронних пристроїв. DCP розроблені для аудіо- та відеосистем, принтерів, камер, систем доступу, інтернет-шлюзів та автоматизації будинків [10].

Zigbee. Zigbee – це стандарт бездротових мереж для домашньої автоматизації, який забезпечує низьку швидкість передачі даних та енергоефективність. Він складається з мережевих координаторів, маршрутизаторів та кінцевих пристроїв. Zigbee працює в діапазонах частот 2,4 ГГц, 915 МГц та 868 МГц. Швидкість передачі даних може бути 250 Кбіт/с (16 каналів, 2,4 ГГц), 40 Кбіт/с (10 каналів, 915 МГц) та 20 Кбіт/с (1 канал, 868 МГц).

Мережеві координатори реєструють і підтримують продуктивність пристроїв, маршрутизатори забезпечують зв'язок між пристроями, а кінцеві пристрої передають дані. Zigbee широко використовується для керування освітленням, безпекою та енергоефективністю [12].

Z-wave. Z-wave – це бездротова технологія, що забезпечує підключення різних пристроїв у домашній мережі. Вона складається з фізичного шару, контролю доступу, мережевого та прикладного рівнів. Z-wave використовує централізовані таблиці маршрутизації для обчислення маршрутів та запису їх у повідомлення.

Кожен пристрій Z-wave ідентифікується 4-байтовим основним ідентифікатором, який призначається контролером після створення пари.

Ідентифікатор вузла – це значення одного байта, присвоєне пристрою контролером у процесі сполучення.

Прикладний рівень Z-wave визначається досить чітко, щоб забезпечити сумісність різних контролерів, датчиків та виконавчих механізмів. Кожен пристрій не повинен брати участь у кожній транзакції на цьому рівні. Для підключення пристроїв використовується механізм сполучення, а для забезпечення безпеки повідомлення шифруються перед відправкою [13].

1.4 Висновки

У першому розділі було проведено аналіз доступних технологій, що використовуються в теперішніх системах і архітектурах розумного дому. На основі цього аналізу був складений перелік найпопулярніших і найпоширеніших протоколів передачі даних та методів, які лежать в основі систем управління та менеджменту розумних будинків.

Результати аналізу показують, що через велику кількість технологій спостерігається активна тенденція розвитку технологій розумного дому, що, в свою чергу, призводить до зростання запитів користувачів. На данний час з’являються нові технології та продовжують вдосконалюватися існуючі, тому користувачі стикаються з завданням вибору найбільш оптимального рішення для задоволення всіх своїх потреб у домашній чи промисловій автоматизації.

На основі проведеного аналізу в роботі було прийнято рішення використовувати бездротову технологію Wi-Fi через її простоту, поширеність та доступність у пристроях розумного дому, оскільки більшість із них оснащені вбудованими Wi-Fi компонентами.

2 ВИБІР АПАРАТНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СТВОРЕННЯ МІКРОКОНТРОЛЕРНОЇ СИСТЕМИ РОЗУМНОГО БУДИНКУ

2.1 Існуючі системи для управління розумним будинком

З огляду на велику кількість систем управління розумним будинком, важливим завданням роботи є аналіз найбільш популярних технологій. Для відповідності всім сучасним потребам користувачів пристрої мають бути максимально гнучкими для інтеграції з більшістю сучасних платформ автоматизації та відповідати актуальним вимогам користувача.

Усі системи управління можна розділити на дві категорії:

– вендорське програмне забезпечення з закритою кодовою базою, протоколами передачі, власними алгоритмами роботи та фіксованою або стандартною платою за інтеграцію пристрою;

– open source проекти, які здобули велику популярність у середовищі домашніх автоматизаторів. Вони мають відкриту кодову базу, здатні до масштабування та впровадження нових протоколів і технологій, а також є абсолютно безкоштовними.

У цьому розділі будуть розглянуті найпопулярніші Open source проекти які використовуються для управління розумним будинком, досліджено їх основні технології та можливості, а також виділено ряд переваг і недоліків у їх роботі.

Програмне забезпечення з відкритим кодом має значну перевагу, а саме, багато з них є 100% безкоштовним. Тому його простіше використовувати ентузіастам Інтернету речей по всьому світу.

2.2 Open source проекти

OpenHAB. OpenHAB (скорочення від Open Home Automation Bus) є одним із найпопулярніших інструментів для домашньої автоматизації серед ентузіастів із

відкритим вихідним кодом. Цей інструмент має велику спільноту користувачів і підтримує численні пристрої та інтеграції. Написаний на Java, openHAB є сумісним з більшістю основних операційних систем і добре працює навіть на Raspberry Pi. Завдяки підтримці сотень пристроїв, openHAB спрощує для розробників додавання власних модулів до системи.

OpenHAB також пропонує мобільні додатки для iOS і Android для управління пристроями, а також інструменти для створення індивідуального користувальницького інтерфейсу для вашої домашньої системи (рисунок 2.1).

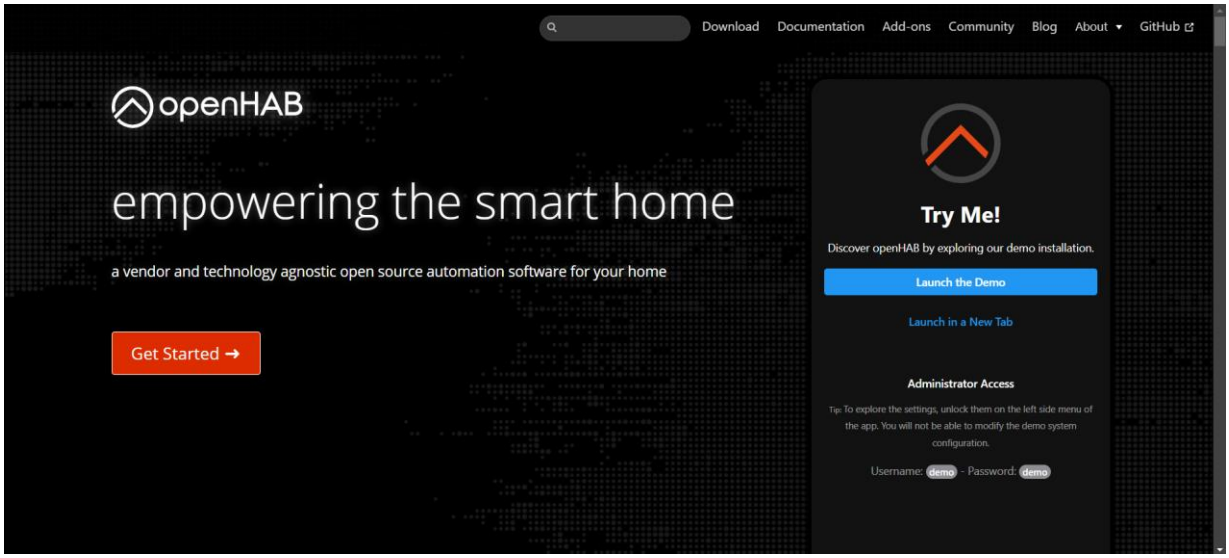


Рисунок 2.1 - Платформа OpenHAB для домашньої автоматизації

OpenHAB - це система домашньої автоматизації з відкритим вихідним кодом, яка має потужну підтримку спільноти. Архітектура з підтримкою плагінів дозволяє розробникам легко додавати нові пристрої та інтегрувати нові сервіси. Система також включає зручний REST-API, індивідуальні дизайни з тригерами на основі часу і подій, службу повідомлень та управління на основі VoiceUI.

OpenHAB працює на Linux, Windows і macOS, а також може бути встановлений на Raspberry Pi, Pine64 і Docker. Він пропонує мобільні додатки для Android-пристроїв і iOS для iPhone та iPad.

З OpenHAB користувач може уникнути використання хмарних сервісів для захисту конфіденційності, але при необхідності система підтримує інтеграцію з хмарою. Вона сумісна з Google Assistant, Amazon Alexa, IFTTT і Apple HomeKit. Для роботи з хмарою OpenHAB пропонує хмарну версію, яка може бути розміщена на їхньому сервері або на власному сервері користувача. Платформа підтримує понад 1500 пристроїв.

OpenHAB випускається під Eclipse Public License і написаний на Java. Вихідний код OpenHAB доступний на GitHub.

Переваги OpenHAB:

- велика спільнота користувачів;
- понад 200 інтеграцій з різними пристроями;
- відкритий API для зовнішніх розробників та DIY-користувачів;
- власний веб-інтерфейс та мобільні додатки;
- відкрита інтеграція через протокол MQTT [14].

Calaos**.** Calaos розроблений як повнофункціональна платформа для домашньої автоматизації, що включає серверний додаток, інтерфейс з сенсорним екраном, веб-додаток, власні мобільні додатки для iOS та Android, а також попередньо налаштовану операційну систему Linux. Цей проект був створений французькою компанією, тому форуми підтримки в основному ведуться французькою мовою, хоча значна частина інструкцій і документації перекладена англійською.

Calaos підтримує широкий спектр пристроїв, включаючи Wago PLC, Raspberry Pi, Zodianet's ZiBASE, Cubieboard, Squeezebox, CCTV та багато інших, з постійним додаванням нових апаратних платформ (рисунок 2.2) [15].

Рисунок 2.2 - Платформа Calaos - повнофункціональна платформа для домашньої автоматизації

Home Assistant. Home Assistant – це платформа домашньої автоматизації з відкритим вихідним кодом, призначена для простого розгортання практично на будь-якому пристрої, який підтримує Python 3, від Raspberry Pi до мережевого сховища (NAS). Вона також поставляється з Docker-контейнером для легкого розгортання на інших системах. Home Assistant інтегрується з безліччю рішень з відкритим вихідним кодом і комерційних пропозицій, дозволяючи зв'язувати, наприклад, IFTTT, погодні сервіси або ваш пристрій Amazon Echo для керування

різноманітним обладнанням, від замків до освітлення. Home Assistant розповсюджується під ліцензією MIT.

Home Assistant забезпечує велику та безшовну інтеграцію з Amazon Alexa, Google Assistant і голосовим помічником Mycroft.io з відкритим вихідним кодом (рисунок 2.3) [16].

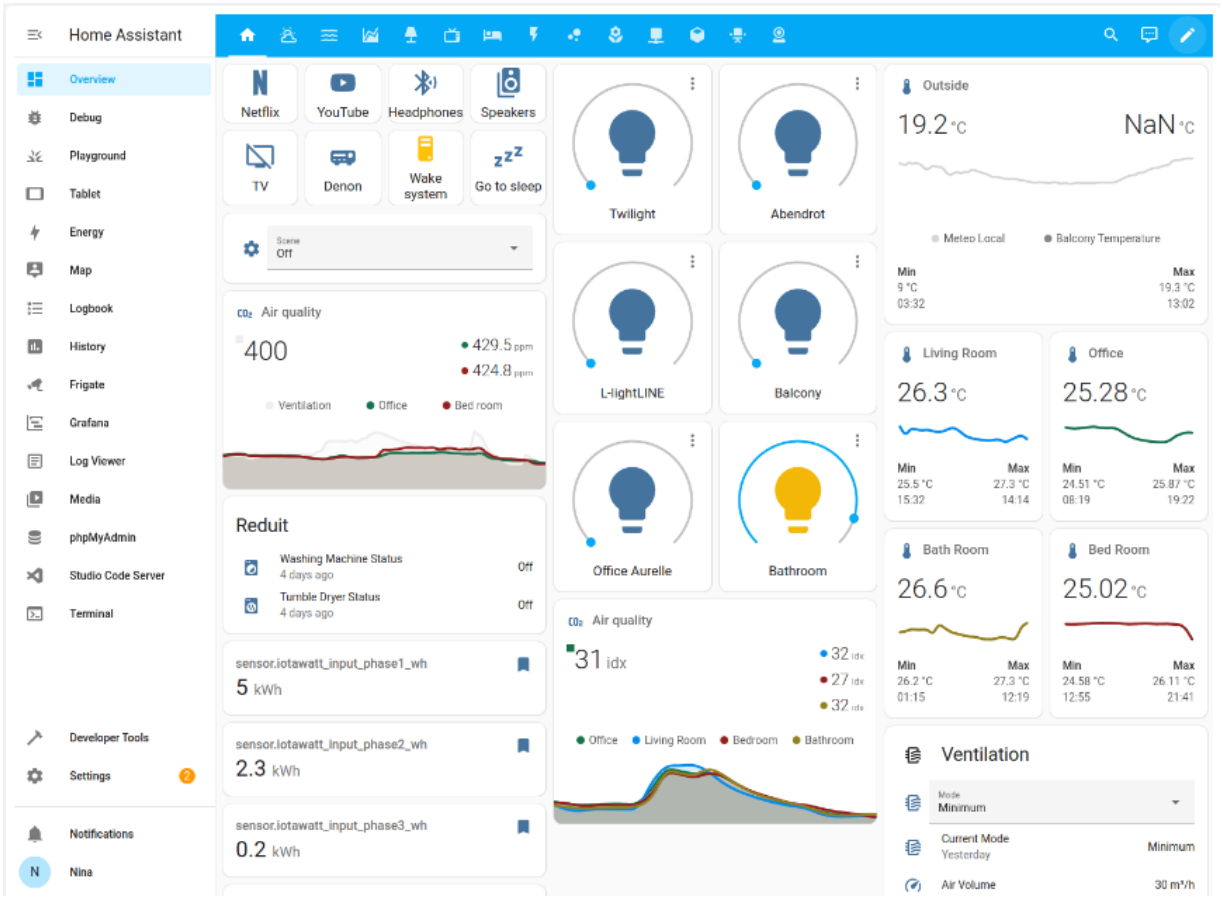


Рисунок 2.3 - Платформа Home Assistant для домашньої автоматизації з відкритим вихідним кодом

OpenMotics. OpenMotics - це система домашньої автоматизації з відкритим апаратним і програмним забезпеченням. Вона розроблена для забезпечення комплексного управління пристроями, а не для об'єднання різноманітних пристроїв від різних постачальників. Відмінною особливістю OpenMotics є фокус на

дротових рішеннях, на відміну від багатьох інших систем, орієнтованих на легку модернізацію.

Платформа OpenMotics поєднує доступне обладнання з відкритим вихідним кодом та сучасні хмарні рішення. Вона дозволяє легко налаштовувати, відстежувати та використовувати кожен аспект вашої установки за допомогою інтуїтивно зрозумілих інтерфейсів на ПК, планшетах і смартфонах. Для отримання додаткової інформації дивіться нашу повну статтю від Backend-розробника OpenMotics Фредеріка Рікбоша.

OpenMotics призначена для забезпечення комплексної системи управління пристроями, яка поєднує доступне обладнання з відкритим вихідним кодом та сучасні хмарні рішення. На відміну від багатьох інших систем, орієнтованих на легку модернізацію, OpenMotics фокусується на дротових рішеннях (рисунок 2.4). [17].

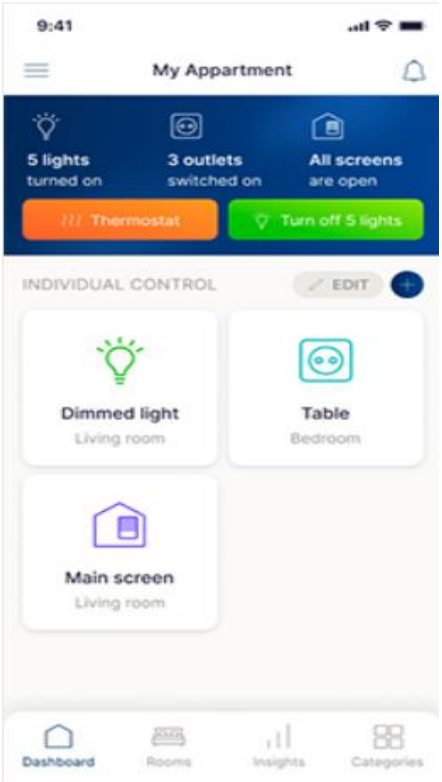


Рисунок 2.4 - Платформа Open Motics

HomeGenie. HomeGenie - це сервер домашньої автоматизації з відкритим вихідним кодом, розроблений для управління, контролю, моніторингу та автоматизації пристроїв, які підключені до Інтернету, таких як інтелектуальні пристрої. Він забезпечує безшовну інтеграцію з багатьма пристроями і службами, а також підтримує безліч протоколів і технологій керування, таких як IR / RF і UPnP / DLNA.

Розроблений на основі багатоплатформного підходу, HomeGenie може взаємодіяти з різними пристроями, такими як X10, Insteon, Z-Wave, Philips Hue, UPnP / DLNA, RFXCom, KNX, і інтегрувати їх в загальну середу автоматизації. Таким чином, навіть якщо пристрої використовують різні стандарти, всередині HomeGenie всі "модулі" можна контролювати і автоматизувати для спільної роботи. Завдяки сучасному вбудованому веб-інтерфейсу користувача, HomeGenie можна використовувати з будь-якого ПК, смартфона або планшета.

HomeGenie можна встановити на операційні системи Windows, Linux і macOS, а також підтримує Raspberry Pi. Він досить легкий у встановленні, налаштуванні та управлінні, навіть для початківців (рисунок 2.5) [18].

Рисунок 2.5 - Платформа Home Genie з відкритим вихідним кодом для домашньої автоматизації

ioBroker. ioBroker – це платформа автоматизації з відкритим вихідним кодом, написана на NodeJS. Вона працює на Windows, Linux, macOS та одноплатних комп'ютерах, таких як Raspberry Pi. Це повноцінна платформа для Інтернету речей (IoT), яка легко підтримує додавання, налаштування та управління пристроями.

ioBroker має сотні адаптерів, які забезпечують інтеграцію з сервісами, пристроями, іншими платформами, датчиками, системами безпеки та протоколами (рисунок 2.6).

Ця платформа, заснована на JavaScript, може керувати освітленням, замками, термостатами, мультимедіа, веб-камерами тощо. ioBroker знаходиться під ліцензією MIT [19].

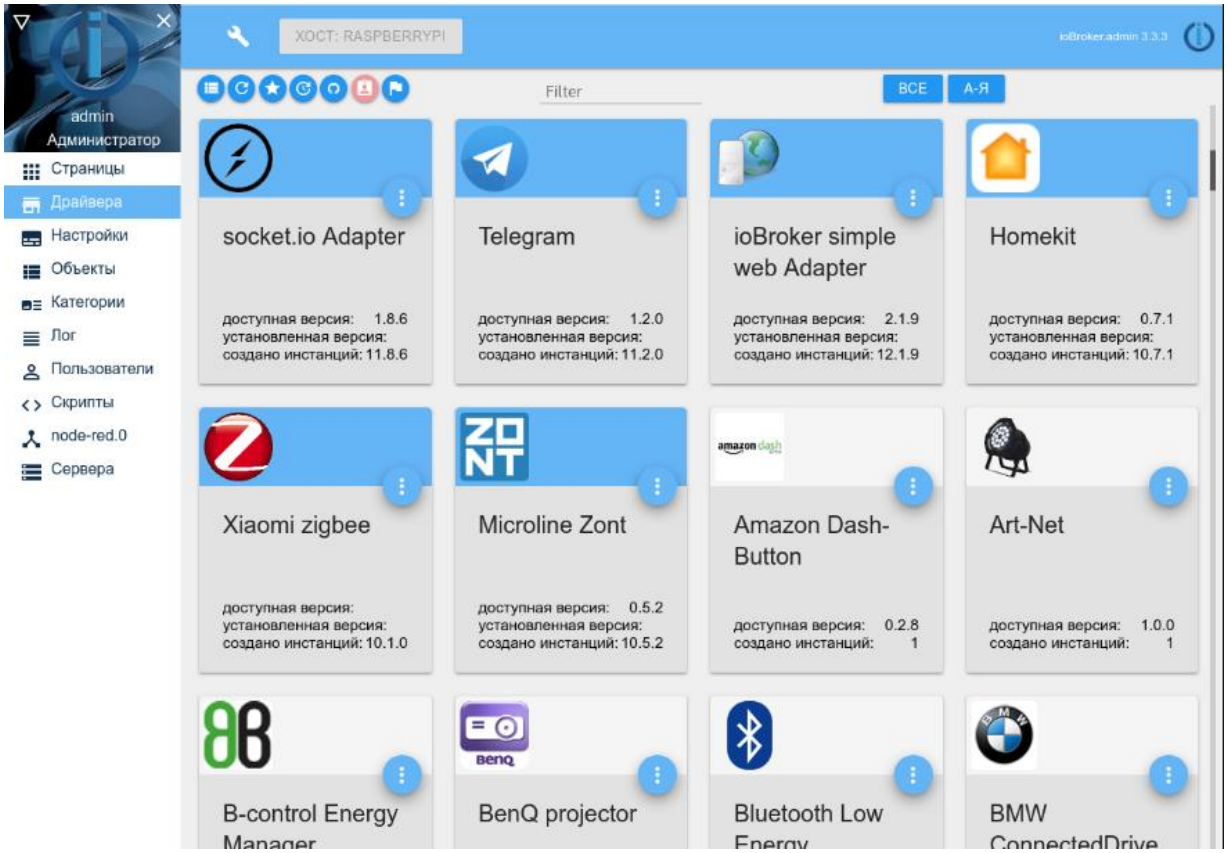


Рисунок 2.6 - Платформа ioBroker відкритим вихідним кодом для систем Windows, Linux, macOS і SBC

FHEM. FHEM – це Perl-сервер для домашньої автоматизації, що поширюється під GPL. Він автоматизує завдання в домі, такі як управління освітленням, жалюзі, опаленням, та реєстрацію подій (температура, вологість, споживання енергії). Програмою можна керувати через веб-інтерфейс, смартфон, telnet або TCP/IP (рисунок 2.7). Для роботи FHEM потрібен цілодобовий сервер (NAS, Raspberry Pi, ПК, Mac Mini тощо).

Особливості FHEM:

– підтримка багатьох протоколів для домашньої автоматизації та інших пристроїв.

– автоматичне створення пристроїв і журналів;

– запис подій у файли або базу даних;

– повідомлення зовнішніх програм при певних подіях;

– команди за розкладом;

– різні інтерфейси: текст, JSON, XML через TCP/IP, SSL або HTTP;

– модульна архітектура з понад 430 модулями [20].

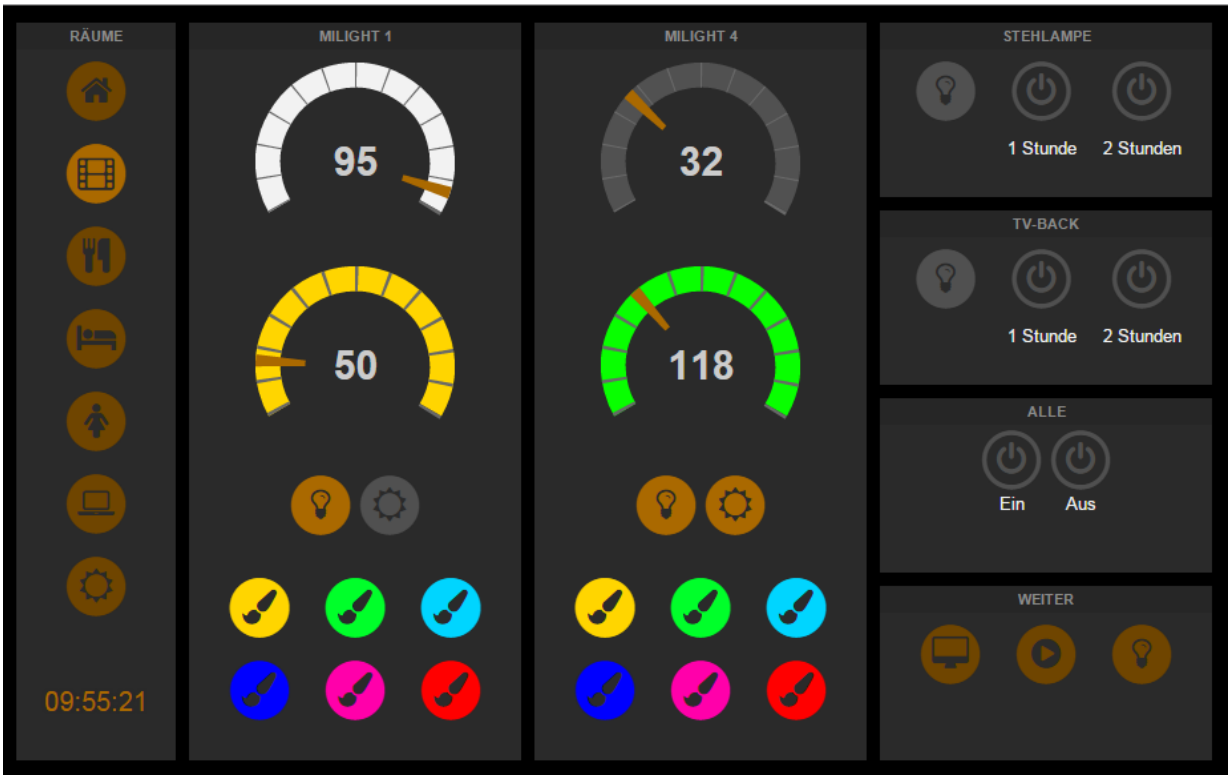


Рисунок 2.7 - fHem - Платформа для домашньої автоматизації

AGO Control. AGO Control — це система домашньої автоматизації під ліцензією GPLv3, створена для Raspberry Pi та вбудованих систем. Вона також забезпечує зв'язок з іншими системами розумного будинку, такими як LinuxMCE.

AGO Control підтримує численні інтелектуальні та мультимедійні пристрої, такі як Z-Wave, 1wire, KNX/EIB, MySensors, Chromoflex USP3, AV-ресивери Onkyo тощо. Інтерфейс наведено на рисунку 2.8.

Основні особливості:

– використання шини повідомлень AMQP Enterprise як комунікаційного сервера;

– легкий протокол, зрозумілий для людей і машин;

– сучасна та модульна архітектура;

– хмарні функції;

– визначення пристроїв у форматі YAML;

– відмінна продуктивність, зокрема на вбудованих пристроях, таких як Raspberry Pi, Sheevaplug і Guruplug;

– підтримка багатьох пристроїв і протоколів;

– підтримує Raspberry Pi і працює з Android [21].

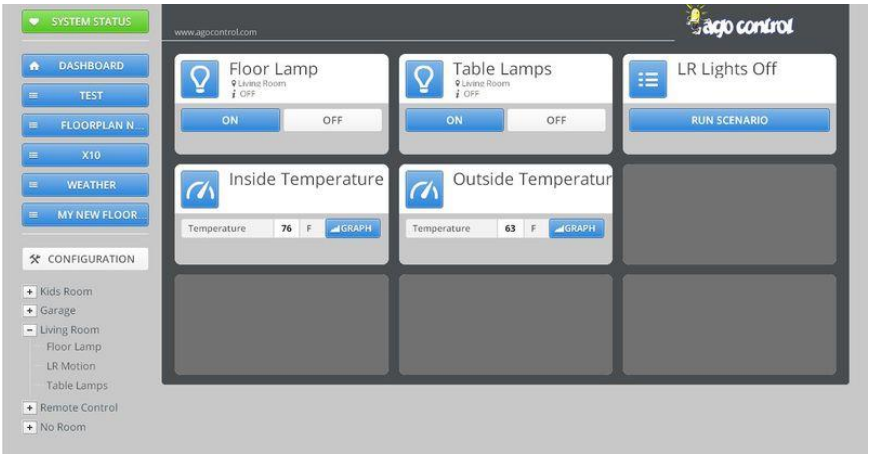


Рисунок 2.8 - Платформа AGO Control для домашньої автоматизації в вбудованих системах

Homebridge. Homebridge дозволяє інтегрувати пристрої розумного будинку, які не підтримують HomeKit, з екосистемою Apple. Існує понад 2000 плагінів Homebridge, що підтримують тисячі різних інтелектуальних аксесуарів.

Платформа в основному призначена для інтеграції пристроїв розумного дому з іншими платформами, зокрема HomeKit, що дозволяє керувати пристроями за допомогою голосового асистента Siri. Крім того, Homebridge має численні плагіни для популярних асистентів, таких як Google, Alexa, Alisa та інших (рисунок 2.9) [22].

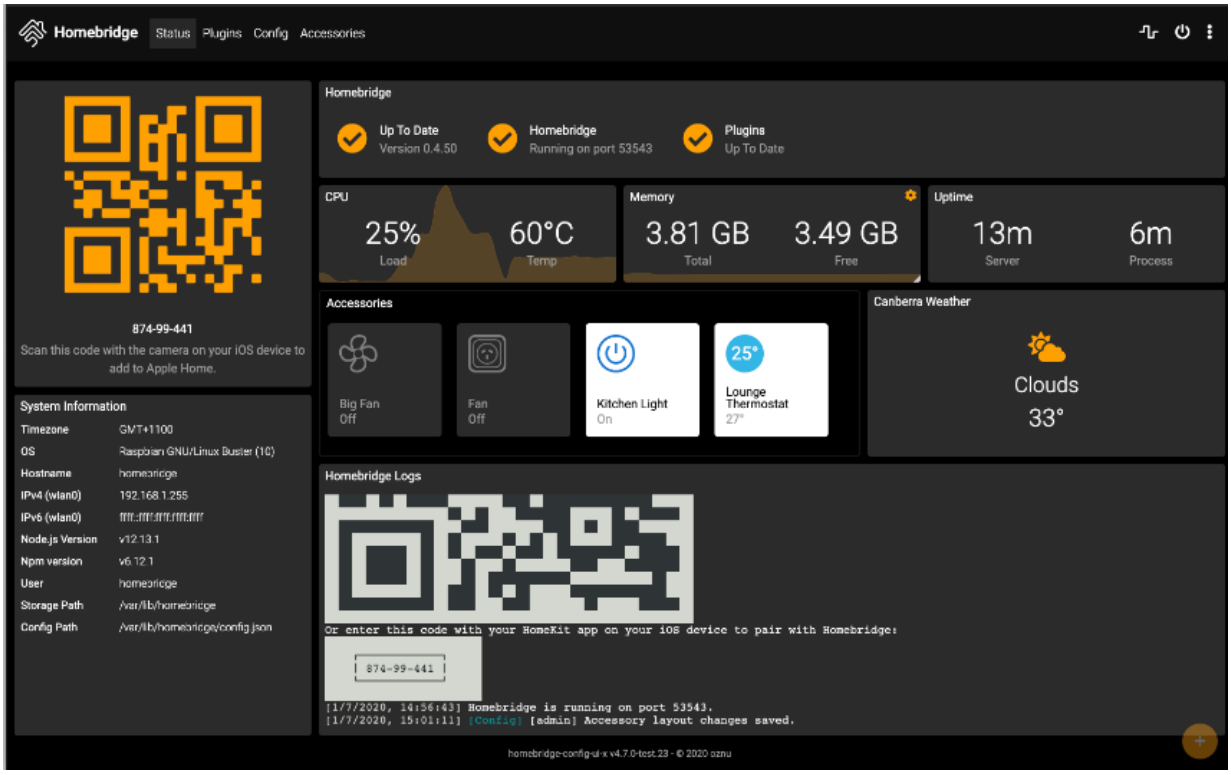


Рисунок 2.9 - Платформа Homebridge для домашньої автоматизації в вбудованих системах

Pimatic. Pimatic - це фреймворк для домашньої автоматизації, який працює на node.js. Він надає загальну платформу для управління будинком та автоматизації. Плагін мобільного інтерфейсу забезпечує зручний веб-інтерфейс з відображенням

датчиків, керуванням пристроями та налаштуванням правил. Веб-інтерфейс побудований з використанням Express і jQuery Mobile.

Pimatic має архітектуру, готову до плагінів, що дозволяє розробникам додавати пристрої, служби та протоколи. Основна увага в цій структурі приділяється гнучкості: її можна використовувати досить швидко і легко. Завдяки вбудованим функціям ви можете автоматизувати завдання, підключивши домашні пристрої та додавши умовні правила (рисунок 2.10). Підтримує Raspberry Pi [23].

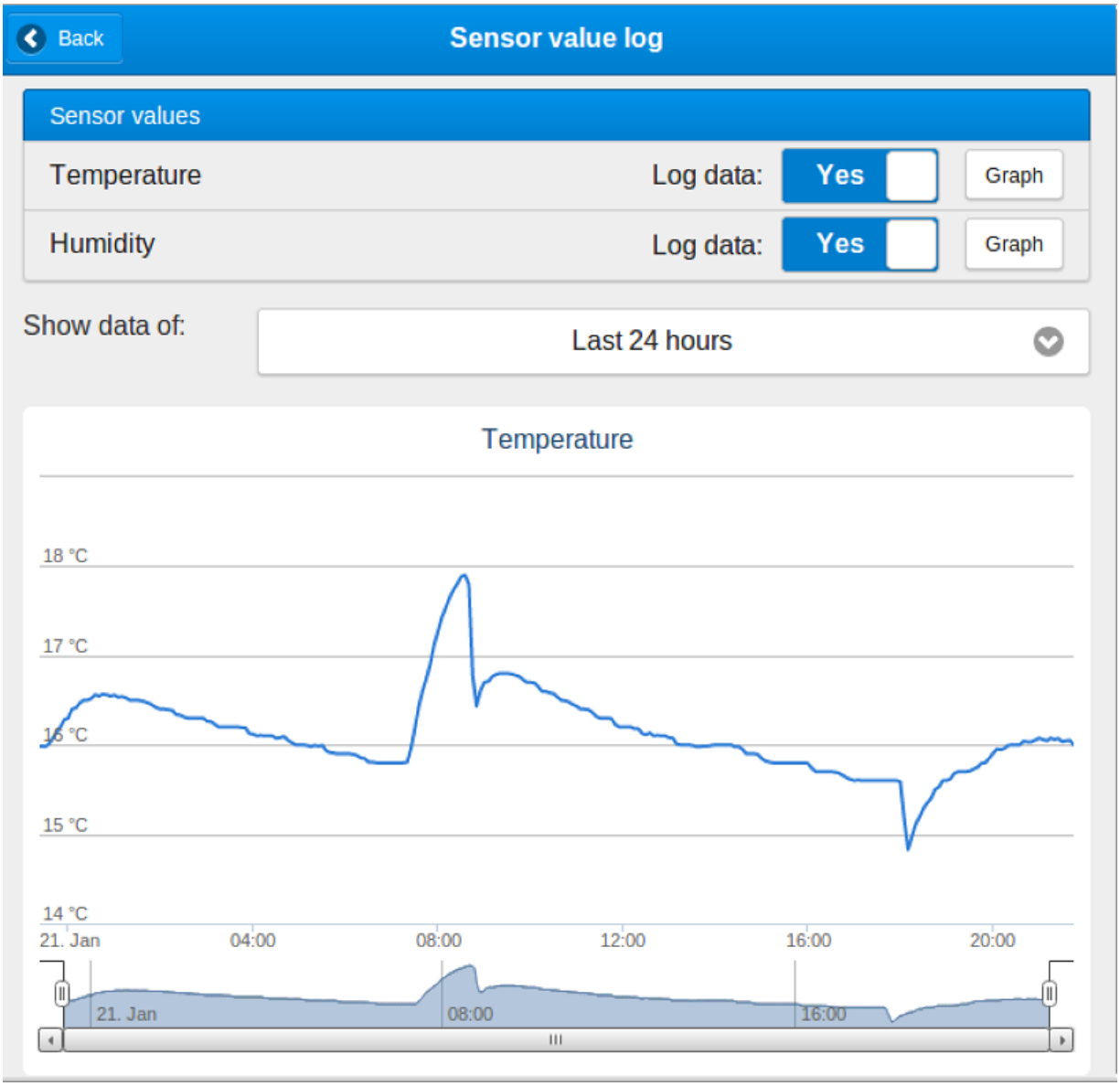


Рисунок 2.10 - Платформа Pimatic як фреймворк для домашньої автоматизації

MyController. MyController - це платформа домашньої автоматизації та Інтернету речей з відкритим вихідним кодом. Початково він був розроблений для підтримки проекту MySensors. Платформа розроблена для роботи на пристроях з обмеженими ресурсами, і оскільки вона побудована на Java, вона працює на одноплатних комп'ютерах з операційними системами Windows, Linux, macOS і на одноплатних комп'ютерах, таких як Raspberry Pi, Pine64, Rock64, Orange Pi.

MyController підтримує багато мереж, шлюзів, протоколів та пристроїв (рисунок 2.11). Це одна з сумісних систем домашньої автоматизації з MySensors у цьому списку. Підтримує декілька шлюзів з різними протоколами: Serial, Ethernet і MQTT [24].

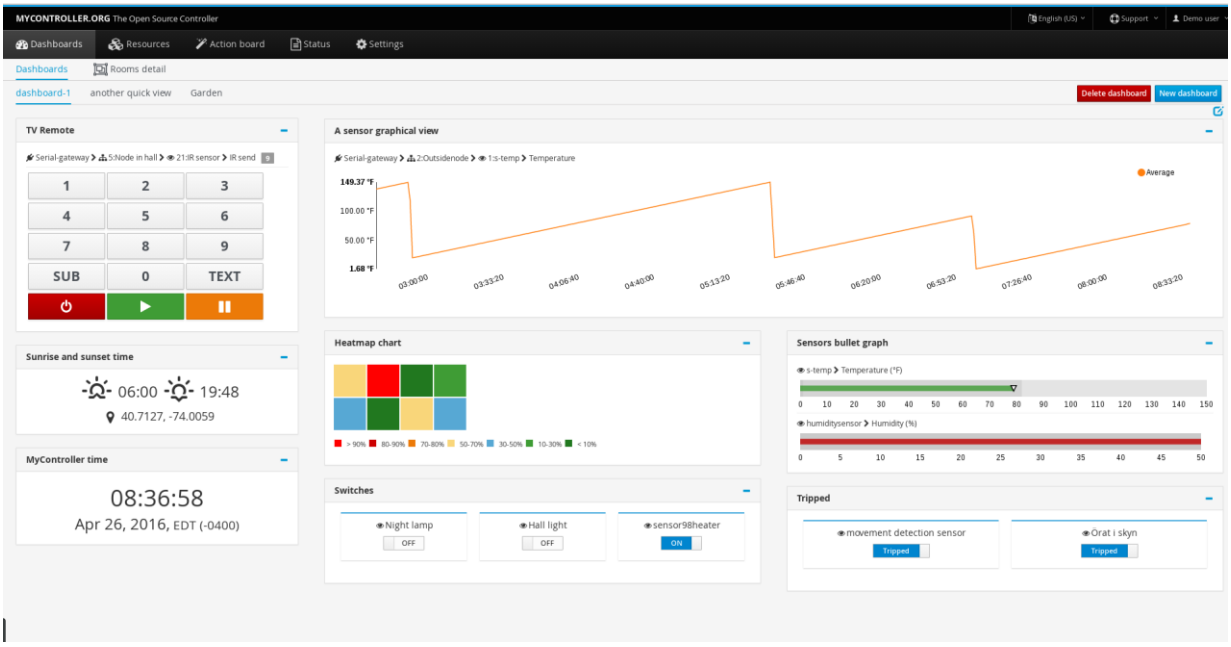


Рисунок 2.11 - Платформа MyController для домашньої автоматизації та Інтернету речей з відкритим вихідним кодом

Node-RED. Node-RED - це інструмент програмування для Інтернету речей, що спрощує об'єднання апаратних пристроїв, API та онлайн-сервісів у нові та цікаві способи. Його редактор потоків, доступний у браузері, дозволяє з легкістю з'єднувати різні компоненти (рисунок 2.12). Після створення потоків їх можна

легко розгорнути для виконання одним кліком. Також можна створювати власні функції JavaScript та зберігати їх у вбудованій бібліотеці для подальшого використання. Node-RED базується на Node.js і використовує його неблокуючу модель подій, що робить його ідеальним для роботи на різних пристроях, включаючи Raspberry Pi та в хмарі. З більш ніж 225 000 модулів у репозиторії пакетів Node можна легко розширити функціональність платформи, додавши нові компоненти. Потоки, створені в Node-RED, зберігаються у форматі JSON, що спрощує їх обмін з іншими користувачами [25].

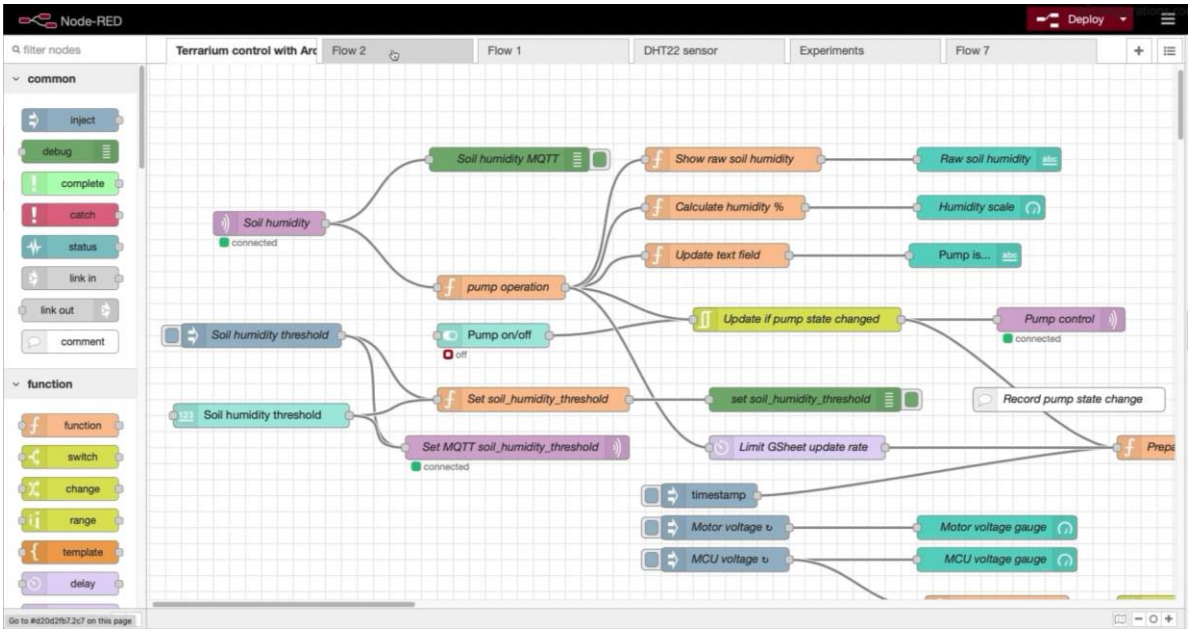


Рисунок 2.12 - Платформа Node-RED − інструмент програмування для Інтернету речей

2.3 Визначення та вибір методу для інтеграції з системами управління

Після аналізу багатьох популярних систем управління було зроблено висновок щодо можливих способів їх інтеграції.

Найпоширенішим, пропонованим виробниками платформ автоматизації, є розробка власних модулів або розширень на рівні програмного забезпечення. Цей підхід, хоча й має свої переваги, такі як стабільність роботи і використання ресурсів

платформи, також має недоліки. Серед них обмеженість у виборі та нездатність задовольнити потреби користувачів у різноманітності та універсальності.

Другим за популярністю способом є розробка підтримки відкритих API платформ на рівні програмного забезпечення (рисунок 2.13). Аналіз показав, що більшість платформ мають відкриті API, але кожна система має свої власні особливості та підходи до роботи з ними, що може створювати складнощі у використанні.

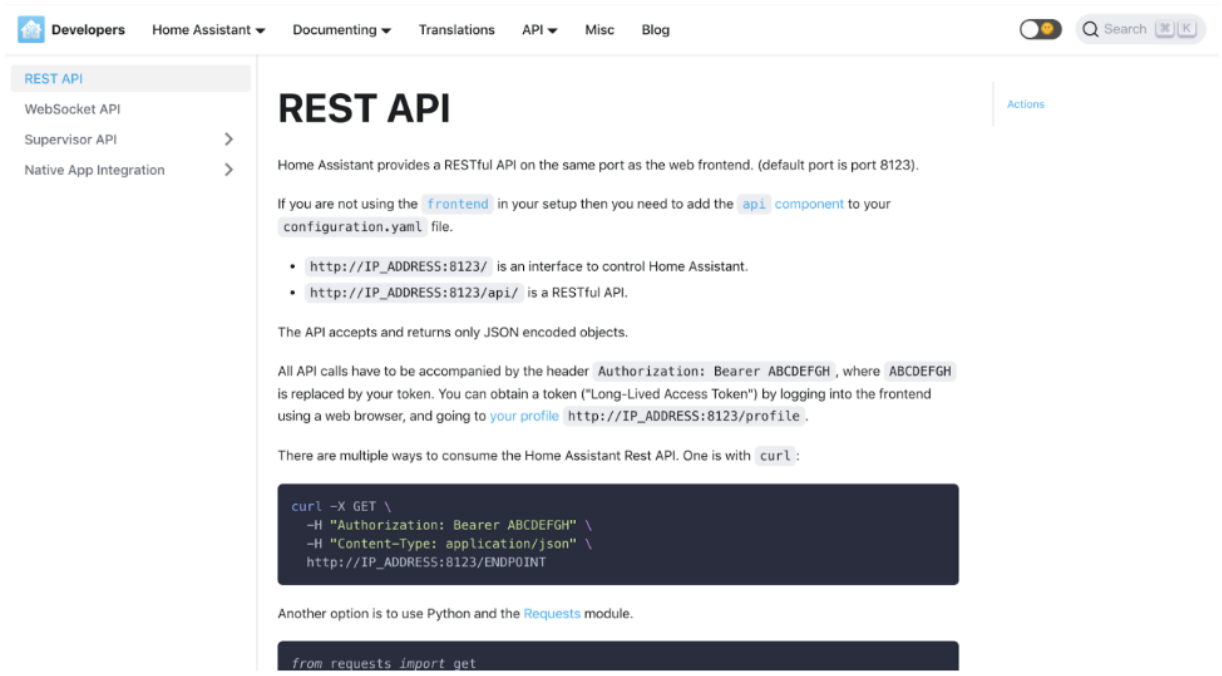


Рисунок 2.13 - Документація відкритого API Home Assistant

Третім способом є використання вже існуючих модулів або компонентів, розроблених сторонніми розробниками, як до прикладу в платформі OpenHab (рисунок 2.14). Цей підхід досить популярний, оскільки у більшості платформ є активна спільнота розробників, які створюють зовнішні модулі.

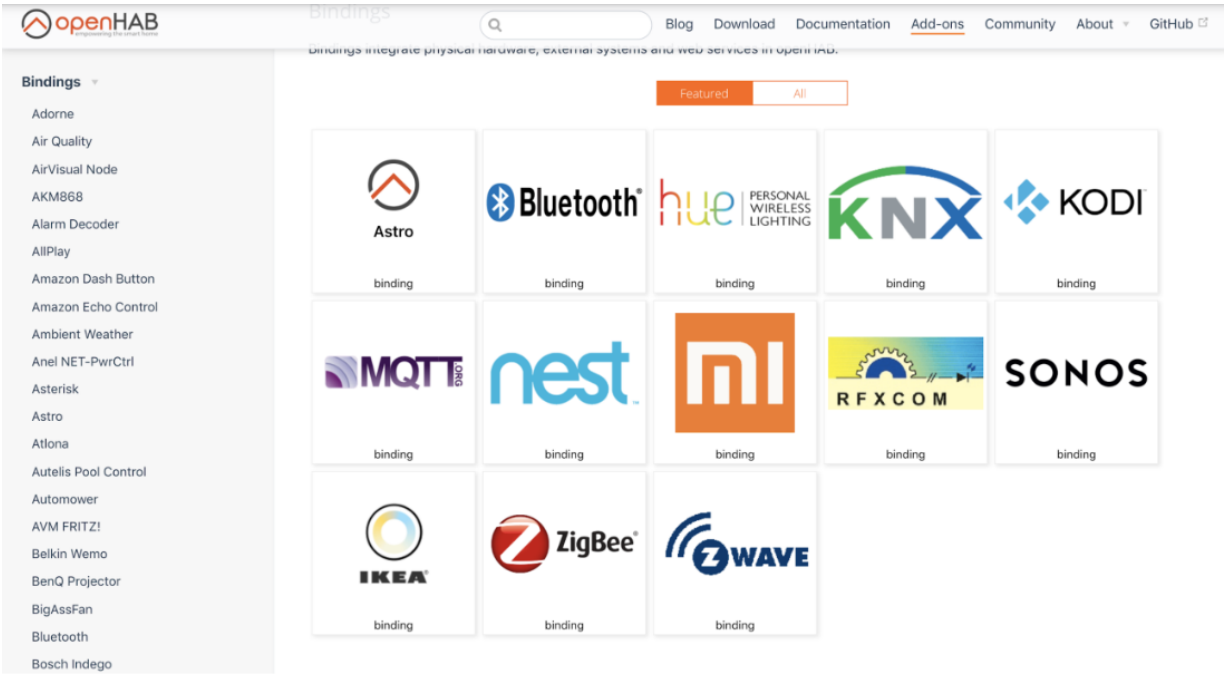


Рисунок 2.14 - Доступні модулі в платформі OpenHab

Аналіз показав, що MQTT брокер є одним з найпопулярніших модулів у багатьох платформах. З цими висновками було прийнято рішення використовувати третій підхід для інтеграції з сучасними платформами через вбудовані MQTT модулі. Це дозволить уніфіковано інтегруватися з платформами, що підтримують MQTT, а таких більшість, та спростить розробку та налаштування системи управління розумним будинком.

2.4 Вибір протоколу передачі даних

MQTT (Message Queuing Telemetry Transport) - це технологія, яка спочатку використовувалася для створення з'єднань у супутниковій мережі. Завдяки своєму легкому протоколу вона знизила пропускну здатність і енергоспоживання.

Ефективність використання ресурсів MQTT відіграла ключову роль у системі, що працює через супутниковий зв'язок. Пізніше цей протокол став популярним у різних областях, включаючи Інтернет речей (IoT), завдяки своїй доступності та низьким вимогам.

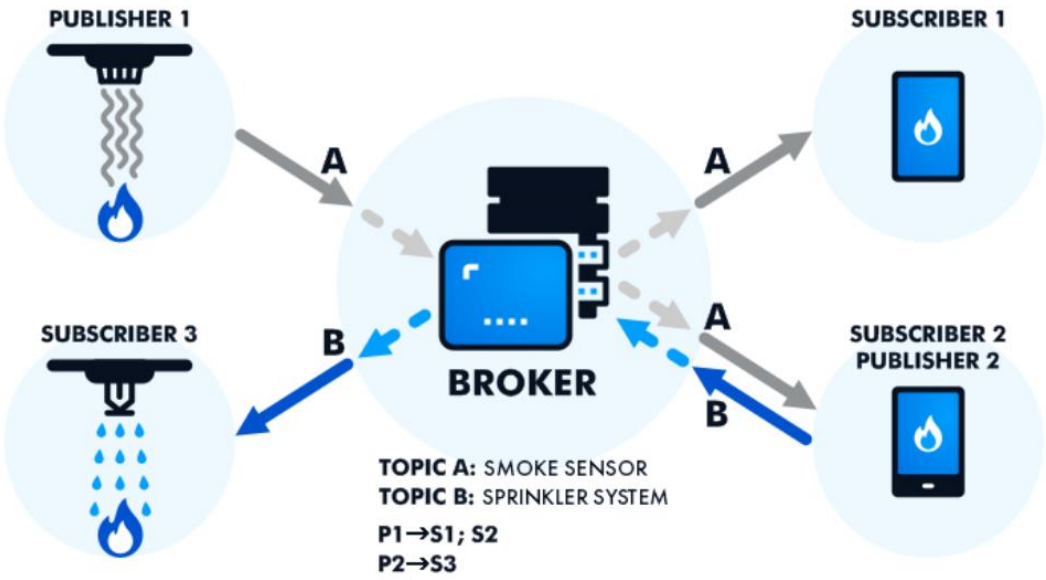


Рисунок 2.15 - Схема роботи MQTT

Принцип роботи MQTT (рисунок 2.15) полягає у використанні шаблону публікації-підписки. Він включає такі компоненти:

- сервер або брокер, який спілкується з клієнтами через інтернет-з'єднання або локальну мережу;
- публікатор, що створює повідомлення і публікує їх у певній темі;
- підписник, що отримує повідомлення, які відносяться до підписаних тем.

Публікатори і підписники можуть обмінюватися ролями, а кількість їх необмежена, залежно від потужності сервера. Кожний клієнт має ідентифікатор, який його ідентифікує на сервері.

Теми можуть містити різні підтеми, утворюючи ієрархію. Це дозволяє публікаторам зв'язуватися з підписниками на різних рівнях тем.

MQTT передає повідомлення пакетами, які складаються з фіксованого заголовка, теми змінної та корисного навантаження (рисунок 2.16). Цей протокол має високу масштабованість, низькі системні вимоги та простоту використання, що робить його популярним у IoT та інших областях [32].

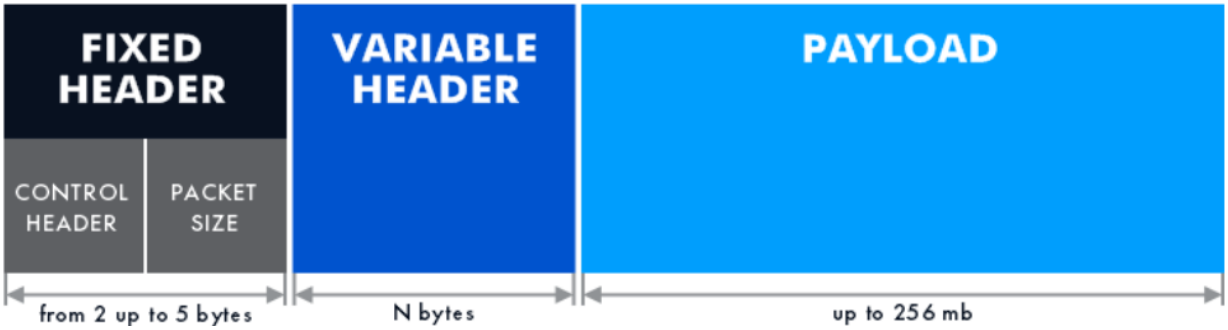


Рисунок 2.16 - Структура MQTT повідомлення

Використання в IoT

Протокол MQTT ідеально підходить для застосувань в Інтернеті речей (IoT), оскільки він забезпечує надійну передачу даних навіть у нестабільних мережних умовах. Він пропонує три рівні якості обслуговування (QoS), які визначаються важливістю та стабільністю з'єднання:

– QoS 0: Публікатор надсилає повідомлення без гарантії доставки. Використовується для некритичних даних і стабільних з'єднань;

– QoS 1: Публікатор надсилає повідомлення і отримує підтвердження про їхню доставку. Використовується для критичних даних і нестабільних з'єднань;

– QoS 2: Публікатор надсилає повідомлення з гарантованою доставкою один раз. Використовується для критичних даних і нестабільних з'єднань, гарантуючи їхню унікальність.

MQTT ідеально підходить для IoT, де пристрої обмінюються даними між собою. Цей легкий протокол забезпечує швидку відповідь та ефективну взаємодію пристроїв незалежно від їх кількості.

Однією з привабливих особливостей MQTT є мінімальні накладні витрати, що дозволяє ефективно передавати дані з низькою пропускною здатністю та зменшує навантаження на ЦП і оперативну пам'ять. Доступні бібліотеки MQTT з відкритим вихідним кодом та публічні брокери спрощують розробку та прискорюють процес впровадження.

Для розгортання MQTT для пристроїв IoT потрібна бібліотека та брокер, який може бути розміщений як локально, так і у хмарі.

Безпека. При використанні протоколу MQTT в системі IoT важливо забезпечити безпеку передачі даних. Хоча безпека MQTT ґрунтується на протоколах TLS/SSL, аутентифікація може бути досить слабкою, що робить його менш надійним у порівнянні з іншими протоколами.

Для покращення безпеки MQTT можна використовувати різні механізми на різних рівнях системи:

– мережевий рівень: Встановлення брандмауера для захисту брокера на мережевому рівні може допомогти у попередженні несанкціонованого доступу;

– рівень застосунку: Використання алгоритмів шифрування на рівні додатку забезпечить безпеку MQTT. Також можна впровадити наскрізне шифрування для захисту конфіденційних повідомлень між клієнтами;

– ідентифікація клієнтів: Використання токенів доступу для ідентифікації клієнтів допоможе запобігти несанкціонованому доступу до системи.

Для поліпшення аутентифікації може знадобитися зміна протоколу та брокера. Оскільки стандартний брокер не дозволяє такі зміни, може бути необхідно розгорнути власний брокер та налаштувати його відповідно до потреб безпеки [28].

IoT додатки, що використовують MQTT. Враховуючи гнучкість та ефективність MQTT, його використання у IoT є вибором багатьох розробників та компаній. Цей протокол дозволяє ефективно обмінюватися даними між пристроями, забезпечуючи надійну доставку навіть у нестабільних умовах мережі, що робить його особливо популярним у сфері IoT і тому використовується в багатьох відомих платформах та сервісах.

IBM Watson IoT Platform: Вона використовує MQTT як основний протокол зв'язку для обміну даними між пристроями IoT та хмарною інфраструктурою IBM.

Amazon Web Services (AWS) IoT: AWS IoT Core має власний брокер повідомлень на основі MQTT і підтримує різні рівні QoS для забезпечення надійності доставки повідомлень.

Microsoft Azure IoT Hub: Цей сервіс також надає можливість створювати додатки IoT з використанням протоколу MQTT.

Крім великих хмарних платформ, багато інших компаній, таких як McAfee, Red Hat, Cisco і IECC, також використовують MQTT для різних завдань в сфері IoT.

Перспективи протоколу

Зростання інтересу до протоколу MQTT, яке зафіксоване за допомогою Google Trends, свідчить про значний розвиток та популярність цього протоколу в сфері Інтернету речей (IoT) протягом 2015-2019 років (рисунок 2.17). Така тенденція пояснюється швидким розширенням ринку IoT та зростаючим попитом на прості та ефективні засоби комунікації між пристроями.

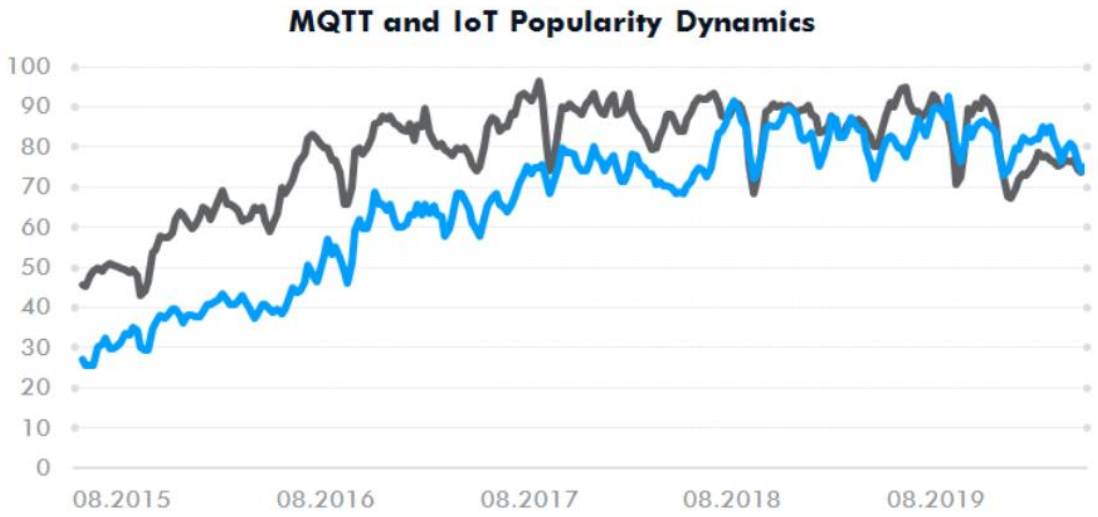


Рисунок 2.17 - Графік популярності протоколу

Протокол MQTT володіє рядом переваг, які підтримують і зберігають інтерес до нього у сфері IoT. Його проста архітектура, легкість реалізації, надійна доставка повідомлень та висока масштабованість роблять його важливим інструментом для забезпечення зв'язку між пристроями IoT.

Остання версія протоколу, MQTT 5.0, випущена організацією OASIS у березні 2019 року. У цій версії додані нові функції, такі як закінчення терміну дії повідомлення, загальна підписка та псевдонім теми. Ці зміни спрямовані на оптимізацію продуктивності MQTT та роблять його ще більш привабливим для розробників IoT, дозволяючи їм зручно та ефективно використовувати цей протокол у своїх проектах [32].

Модель публікації / підписки. У моделі публікації / підписки, використованій у протоколі MQTT, клієнти поділені на дві групи: видавці, які надсилають дані, і передплатники, які отримують ці дані. Видавці і передплатники не мають прямого контакту один з одним і не знають про існування інших. Брокер MQTT діє як посередник, який направляє повідомлення від видавців до всіх передплатників, які підписалися на відповідні теми (рисунок 2.18).

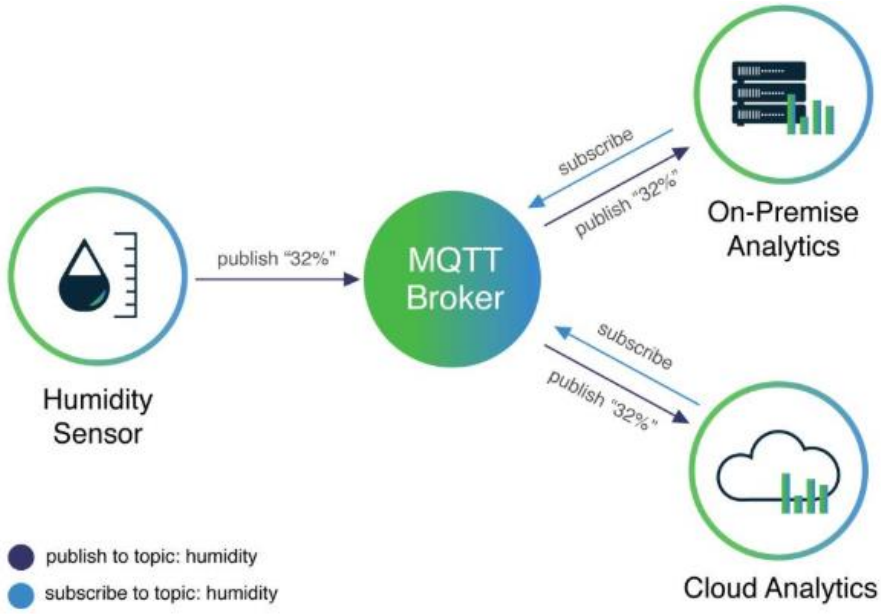


Рисунок 2.18 - Модель публікації-підписки

Теми MQTT. Теми MQTT служать як ієрархічні рядки UTF-8, які використовуються для передачі повідомлень від видавців до передплатників. Кожен рівень в темі відокремлюється косою рискою, і кожне повідомлення має включати тему. Для отримання опублікованого повідомлення об'єкт повинен підписатися на ту ж тему, що і видавець. Брокер MQTT відправляє отримане повідомлення тільки тим клієнтам, які підписалися на відповідну тему.

Хоча MQTT не є єдиним протоколом в області мережевих комунікацій та передачі даних, він має кілька ключових переваг, включаючи надійність, швидкий час відгуку, підтримку необмеженої кількості пристроїв і можливість публікувати / підписуватися на повідомлення, що ідеально підходить для спілкування "багато

до багатьох". Ці переваги роблять MQTT популярним вибором для IoT-пристроїв, де він використовується як частина стека протоколів.

Однак MQTT не визначає структуру та зміст повідомлень, і контролюючий об'єкт повинен бути налаштований для взаємодії з пристроєм IoT. В цьому контексті стандартизовані протоколи, такі як Homie, можуть сприяти автоматичному виявленню, налаштуванню та використанню пристроїв та служб на основі MQTT.

2.5 Вибір апаратного забезпечення

Обрання відповідного апаратного забезпечення є важливим кроком у проектуванні архітектури програмного забезпечення для системи розумного дому. Плати мікроконтролерів є популярним вибором для таких проектів, оскільки вони представляють собою невеликі комп'ютери, здатні керувати різними пристроями та виконувати необхідні завдання. Основні компоненти плати мікроконтролера включають центральний процесор (ЦП), пам'ять та периферійні пристрої введення/виводу.

Мікроконтролери застосовуються в широкому спектрі пристроїв, від автомобільних систем керування до побутової техніки та інших вбудованих систем. Вони також популярні серед інженерів, ентузіастів і програмістів для DIY-проектів і навчальних цілей.

Під час аналізу було вибрано декілька найпопулярніших мікроконтролерів, які відповідають вимогам до апаратного забезпечення для проектування системи розумного дому. Такий підхід дозволяє забезпечити сумісність та ефективну роботу програмного забезпечення з обраною апаратурою.

Arduino Microcontroller Board. Плата Arduino Uno R3 є однією з найпопулярніших плат мікроконтролерів для проектів розумного дому. Вона була випущена компанією Arduino в 2011 році і базується на чіпі ATmega328P. Ця плата має різні введено-виведено (I/O) контакти, які дозволяють підключати її до різних

сенсорів, актуаторів та інших пристроїв. Крім того, до неї підключені різноманітні порти, включаючи USB-порт для підключення до комп'ютера або ноутбука (рисунок 2.19).

Arduino Uno R3 доступна за доступною ціною і має відкритий вихідний код, що робить її популярним вибором для проектів IoT. Ця плата також легко підключається до різних онлайн-бібліотек і ресурсів, що дозволяє розробникам отримати доступ до багатьох корисних матеріалів для реалізації своїх проектів. Існує багато різновидів комплектацій Arduino Uno R3 з різними технічними характеристиками та ціновими пропозиціями, що дозволяє вибрати оптимальний варіант для конкретного проекту [33].

Рисунок 2.19 - Arduino Uno R3

Arduino Pro Mini 328. Arduino Pro Mini 328 - це ще один популярний варіант мікроконтролера від Arduino. Ця міні-плата призначена для застосувань, де важлива компактність та обмежені ресурси живлення, оскільки вона працює при напрузі до 5 вольт. Arduino Pro Mini 328 оснащена 16-мегагерцовим генератором тактових імпульсів. У порівнянні з іншими моделями Arduino, на цій платі немає вбудованих роз'ємів і портів, що означає, що можливо доведеться самостійно паяти з'єднання (рисунок 2.20). Однак ця плата стане вигідним вибором, особливо якщо

ви працюєте з обмеженим бюджетом, і вам потрібен компактний і надійний мікроконтролер для свого проекту [34].

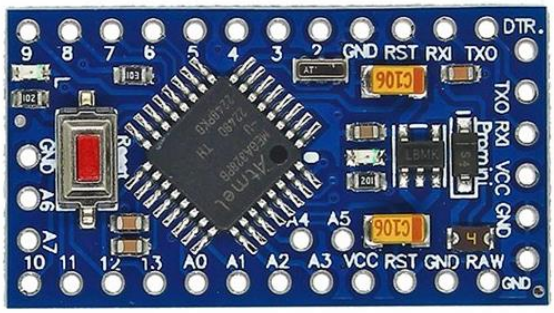


Рисунок 2.20 - Arduino Pro Mini 328

ESP32**.** Плата мікроконтролера ESP32 є комбінованим модулем Bluetooth і Wi-Fi на одній платі (2,4 ГГц) з низьким енергоспоживанням (рисунок 2.21). Вона вважається найкращим вибором для додатків, де потрібна найкраща продуктивність RF. Плата ESP32 широко використовується для проектів DIY, таких як розумний будинок і проекти на основі Інтернету речей. Ця плата забезпечує зручний і потужний інструментарій для розробки різноманітних проектів з підключенням до мережі Wi-Fi та Bluetooth [35].

Рисунок 2.21 - ESP32

ESP8266. ESP8266 - це мікроконтролер з невеликими розмірами, що підтримує IoT. Він має дуже низьку вартість (приблизно 3 долари США) і може бути використаний для проектів розумного будинку та Інтернету речей. Ця дошка також може бути використана для створення особистих помічників, таких як Cortana або Alexa. ESP8266 має 128 КБ оперативної пам'яті і 4 МБ флеш-пам'яті (рисунок 2.22). Однак головна перевага ESP8266 полягає в можливості створення власної мережі для підключення інших пристроїв [36].

Рисунок 2.22 - ESP8266

Raspberry Pi 4. Raspberry Pi 4, випущена в 2019 році, є найшвидшою платою мікроконтролера на сьогоднішній день. Завдяки наявності 4 ГБ оперативної пам'яті ви можете створювати потужні і просунуті електронні проекти. Raspberry Pi 4 може надавати струм до 1,2 А для USB-пристроїв. Ця плата мікроконтролера доступна у різних варіантах оперативної пам'яті від 1 ГБ до 4 ГБ. Додаткові функції включають вбудовану бездротову локальну мережу, Bluetooth 5.0, два порти USB 2.0 і USB 3.0, два порти Micro HDMI і порт Gigabit Ethernet (рисунок 2.23). Raspberry Pi 4 стала популярним вибором для широкого кола проектів, які вимагають високої продуктивності та можливостей підключення різних пристроїв [37].

Рисунок 2.23 - Raspberry Pi 4

Raspberry Pi Zero W. Raspberry Pi Zero W - це розширена версія мікроконтролера Pi Zero з додатковим модулем WLAN для підключення по Bluetooth. Вона зберігає всі функції оригінального Pi Zero, але має додаткові можливості підключення. Pi Zero W працює на одноядерному процесорі з тактовою частотою 1 ГГц і має 512 МБ оперативної пам'яті (рисунок 2.24). Це ідеальний вибір для створення власних проектів Інтернету речей (IoT), оскільки він є економічним і коштує близько 10 доларів США [38].

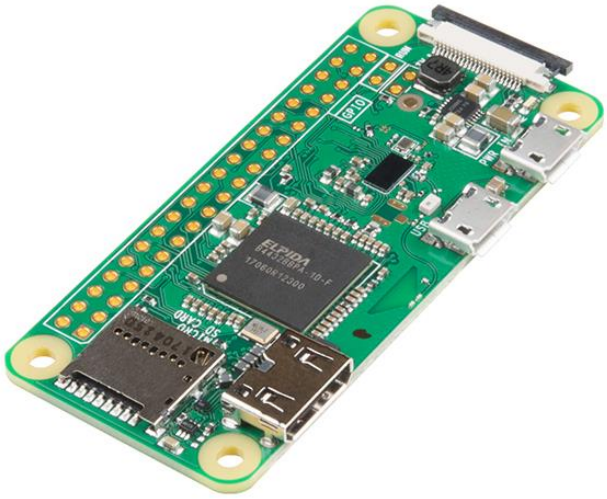


Рисунок 2.24 - Raspberry Pi Zero W

За поданим списком можна зробити висновок, що існує безліч можливостей для створення пристроїв розумного дому на різних апаратних платформах. Всі описані плати для програмування можуть служити основою для створення різноманітних пристроїв розумного дому, хоча не всі вони відповідають вимогам.

Після аналізу було вибрано мікроконтролери ESP з кількох причин:

- низька вартість мікроконтролера;
- різноманітність моделей та технічних характеристик;
- вбудований Wi-Fi модуль;
- доступність документації, інструкцій та приладів;
- зручність використання;
- компактні габарити.

Крім цього, важливим аспектом є те, що більшість сучасних Wi-Fi пристроїв розумного дому базуються на мікроконтролерах ESP. Це дозволяє використовувати розроблене програмне забезпечення на готових пристроях без необхідності розробки власних плат або корпусів.

2.6 Вибір інструментіу для створення ПЗ

Оскільки апаратне забезпечення обрано з родини ESP, вибір інструментів програмування має враховувати архітектуру і можливості мікроконтролера. На сьогоднішній день існує багато інструментів для програмування, проте потрібно обрати найбільш доцільний та сумісний з ESP фреймворк, що здатний виконати всі поставлені вимоги та завдання.

Основною вимогою до інструменту, на якому буде розроблятися програмне забезпечення, є низький поріг входу користувачів і можливість самостійної розробки без необхідності вивчення складних мов програмування.

Для відповіді на ці вимоги був проведений огляд існуючих інструментів, в результаті чого було визначено ряд їх переваг та недоліків.

MicroPython**.** MicroPython - це одна з програмних платформ для мікроконтролерів ESP32 і ESP8266, яка базується на мові програмування Python 3. Вона включає невелику частину стандартної бібліотеки Python і спеціально оптимізована для роботи на мікроконтролерах ESP.

ESP32 і ESP8266 - це мікроконтролери з підтримкою Wi-Fi, на яких можна встановити MicroPython. Вона надає можливість використовувати Python на низькорівневому рівні, що дозволяє керувати різноманітними електронними проектами.

MicroPython має розширений функціонал, включаючи інтерактивний режим, обробку виключень та інші корисні можливості. Для роботи потрібно небагато ресурсів, 256 КБ кодового простору і 16 КБ ОЗУ [39].

CircuitPython**.** CircuitPython базується на Python і легко використовується з платами ESP32 і ESP8266. Якщо ви вже знайомі з Python, ви зможете легко застосувати свої знання до CircuitPython. Навіть якщо ви новачок, почати роботу з CircuitPython дуже просто.

Це через те, що CircuitPython спрощений у використанні: вам знадобиться лише плата ESP32 або ESP8266, USB-кабель і комп'ютер з USB-підключенням. Також для роботи з CircuitPython можна використовувати Mu - простий текстовий редактор, який підтримується на платформах Windows, Mac і Linux [40].

Arduino IDE. Arduino - це дуже потужна та легка у програмуванні платформа. Вона має велике всесвітнє співтовариство, яке дозволяє ділитися знаннями та отримувати відповіді від багатьох ентузіастів. Arduino поставляється зі своїм власним програмним забезпеченням, яке сумісне з операційними системами Windows, Linux та Mac. Ви також можете програмувати онлайн. Якщо ви знаєте мови програмування C і C++, це дозволить вам легше розуміти синтаксис.

Проект Arduino був започаткований у 2003 році як ініціатива студентів Інституту дизайну взаємодії у місті Івреа, Італія. Мета проекту - надати доступну та просту можливість для початківців і фахівців у створенні пристроїв, що взаємодіють з навколишнім середовищем за допомогою датчиків та виконавчих

механізмів. Типові приклади проектів для початківців-любителів включають прості роботи, термостати та датчики руху.

Для початку роботи з Arduino IDE необхідно встановити підтримку для плат ESP8266 та ESP32, щоб завантажувати та перевіряти коди [41].

NodeMCU. NodeMCU - це прошивка, розроблена на мові Lua для мікроконтролерів ESP32 і ESP8266 WiFi SOC від Espressif. Вона використовує файлову систему SPIFFS на базі вбудованої флеш-пам'яті. Спочатку прошивка була розроблена як додатковий проект для популярних модулів розробки NodeMCU на основі ESP8266. Однак зараз проект підтримується спільнотою, і NodeMCU можна використовувати на будь-якому модулі ESP [36].

Mongoose OS. Mongoose OS - це середовище розробки прошивки, призначене для використання з мікроконтролерами ESP32 і ESP8266. Основними компонентами операційної системи Mongoose є інструмент mos, який забезпечує можливості управління пристроєм і створення прошивки, а також набір інструментів для збірки. Останній включає в себе образ докера, що містить SDK виробника обладнання разом з вихідними кодами Mongoose OS. За допомогою інструмента mos можна зібрати прошивку, використовуючи файл mos.yml у поточному каталозі, і викликати образ докера для збірки або локально, або віддалено [42].

Espruino. Espruino - це набір мікроконтролерних пристроїв, які мають вбудований інтерпретатор JavaScript. Найменшим з них є Espruino Pico. Оригінальний інтерпретатор JavaScript, використовуваний в Espruino, спрямований на швидку розробку на пристроях з обмеженими процесорними ресурсами. Існують версії Espruino для різних платформ, включаючи ESP8266 і Raspberry Pi. Також для Espruino доступний широкий вибір готових модулів для підключення різноманітного периферійного обладнання, яке сумісне з екосистемою Arduino, разом із відповідними інструкціями [43].

2.7 Висновки

В другому розділі було проведено аналіз сучасних платформ та систем управління для розумного будинку з метою визначення вимог до майбутньої архітектури програмного забезпечення. На основі цього аналізу було сформульовано перелік функціональних та нефункціональних вимог, враховуючи сучасні потреби користувачів і їхні повсякденні завдання.

Для забезпечення простої і гнучкої інтеграції з сучасними платформами було обрано використання MQTT як протоколу спілкування з брокером. Це дозволить підключати пристрої до більшості існуючих систем управління або до стандартного MQTT брокера і керувати ними з веб-або мобільних клієнтів. Для стандартизації та самодекларативного опису пристроїв було обрано Homie як базову MQTT конвенцію.

Крім того, було проведено аналіз сучасних мікроконтролерів та їх можливостей, і вибір було зупинено на мікроконтролерах сімейства ESP через їх доступність, варіативність, низьку вартість та необхідну технічну складову для вирішення поставлених задач.

Останнім етапом було проведено аналіз можливих інструментів для програмування мікроконтролерів ESP, і було обрано Espruino для реалізації проекту. з кількох причин:

1. Низький поріг входу: Espruino надає найбільш доступний шлях для недосвідчених користувачів.
2. Широкий вибір бібліотек: Більшість існуючих бібліотек, основних для Arduino, вже інтегровані в інфраструктуру Espruino або легко переносяться, що спрощує розробку.
3. Швидка розробка ПЗ: Espruino дозволяє найшвидше отримувати працююче програмне забезпечення на мікроконтролері.
4. Готові бібліотеки: Існує велика кількість готових бібліотек для різних пристроїв та датчиків.

5. Зручний веб-інтерфейс: Інтерфейс Espruino дуже зручний у використанні, що полегшує налаштування та взаємодію з пристроєм.

6. Сумісність з ESP: Espruino повністю сумісний з більшістю мікроконтролерів сімейства ESP, що робить його привабливим вибором для різних проектів.

Наприкінці третього розділу було повністю сформульовано ряд вимог, необхідних для початку розробки власної архітектури, та визначено всі технічні можливості та бар'єри, з якими можна зіткнутися в процесі розробки.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ МІКРОКОНТРОЛЕРНОЇ СИСТЕМИ «ДОМАШНІЙ АСИСТЕНТ»

3.1 Реалізація автоматичної збірки програмного забезпечення на основі конфігураційного файлу

Після налаштування середовища розробки першочерговим завданням стало робота з конфігураційним файлом: його розбиття та передача необхідних даних компілятору для створення потрібних бінарних файлів, які потім завантажуватимуться на пристрій.

Для опису конфігураційного файлу було обрано популярний і зрозумілий формат JSON. Спочатку був створений формат файлу і розділений на основні компоненти.

Файл конфігурації було умовно розбито на кілька вкладених об'єктів (рисунок 3.1). Весь основний контент було винесено в об'єкт "extensions" для майбутнього масштабування. Основний об'єкт складається з кількох додаткових:

- *modules* – масив підключених модулів з бібліотеки; такий формат опису необхідний для оптимізації використання пам'яті при створенні бінарних файлів ПЗ, що дозволяє завантажити тільки ті модулі, які точно використовуються в пристрої, а всі інші будуть проігноровані;
- *controller* – атрибут, який вказує, який тип мікроконтролера буде використовуватись на пристрої; цей атрибут є обов'язковим, оскільки на його основі запускається спеціалізований процес збирання ПЗ під час завантаження;
- *mapping* – масив з описом співвідношення конкретних виходів мікроконтролера до модуля, який буде з ним взаємодіяти (далі буде детально описано, яким чином формується цей масив для кожного з існуючих модулів).

Рисунок 3.1 - Розбиття конфігураційного файлу

Другим основним об'єктом конфігураційного файлу є deviceConfig. Тут відбувається формування структури пристрою в брокері згідно з конвенцією Homie.

Згідно з конвенцією, кожен пристрій представлений у вигляді набору дочірніх сутностей — нод. Кожна нода має свій набір сенсорів, опцій та телеметрії. Ці поняття є абстрактними і служать для уніфікованого підходу до відображення пристрою в брокері. Крім того, у самого пристрою можуть бути опції та телеметрії, але сенсори — це атрибути лише нод. На основі цього конфігураційного файлу буде формуватися стан пристрою та його топіки в MQTT (рисунок 3.2, рисунок 3.3).

На рисунку показано приклад об'єкта deviceConfig, що описує пристрій, який складається з декількох телеметрій (одна з яких — IP-адреса пристрою), однієї стандартної опції та кількох нод. Відображення сенсорів, опцій та телеметрії як для пристроїв, так і для нод однакове, що свідчить про універсальність та єдиний підхід.

Кожна нода має кілька обов'язкових атрибутів:

- *id* — унікальний ідентифікатор ноди згідно з вимогами конвенції;
- *state* — статус ноди за замовчуванням; цей атрибут є нефункціональним, але згідно з конвенцією статус має бути одним із списку (ready, init, lost, disconnected) і описувати її стан;
- *type* — опціональний атрибут ноди, який може оброблятись в системах управління в їх бізнес-логіці;

– *sensors* — масив об'єктів сутностей сенсора, де може бути описана будь-яка комбінація сенсорів. Аналогічно в нодах можуть бути масиви об'єктів для options і telemetry, а уніфікований підхід дозволяє легко змінювати місце розташування цих сутностей.

Подібним чином потрібно описувати стан сенсорів, опцій та телеметрії і розміщувати їх у потрібних місцях пристрою (в ноді чи в самому пристрої). На (рисунок 3.3) представлено список обов'язкових атрибутів сенсорів:

– *id* — унікальний ідентифікатор сенсора, опції чи телеметрії згідно з вимогами конвенції;

– *datatype* — ідентифікатор типу даних сутності згідно з конвенцією; може бути одним із списку (string, integer, float, boolean, color, enum); цей атрибут призначений для використання в бізнес-логіці системи управління і ідентифікує, які дані можуть бути передані або прийняті в топік даного сенсора;

– *retained* — атрибут, який описує тип повідомлень, що будуть відправлятися в брокер для топіків цього сенсора;

– *unit* — атрибут, що описує статичну одиницю вимірювання значення даного сенсора;

– *settable* — атрибут, який описує можливість зміни стану в топіках цього сенсора (наприклад, для датчика температури цей атрибут має бути false, оскільки відредагувати значення температури може лише сам мікроконтролер, а для аналогового виходу — true, оскільки сенсори такого типу будуть регулюватися користувачем);

– *name* — атрибут, що описує статичну назву сенсора; валідація цього атрибуту відрізняється від id, що дає можливість описати в ньому зручнішу назву для ідентифікації.

Рисунок 3.2 - Приклад об’єкта з описом стану пристрою згідно конвенції

Рисунок 3.3 - Приклад об’єкта з описом стану пристрою згідно конвенції

В об'єкті *mapping* виконується співвідношення модулів до конкретних сенсорів, опцій чи телеметрій з об'єкта deviceConfig. У кожної ноди, сенсора, опції та телеметрії є унікальний в межах пристрою ідентифікатор, який дозволяє сформувати повний топік для нього, що дає можливість передати його в модуль для передачі даних. Таким чином, масив mapping складається з об'єктів, що описують співвідношення модуля до топіка сутності. У кожного об'єкта є набір обов'язкових атрибутів:

- *pin* — атрибут, що описує фізичний вихід на пристрої, з яким він взаємодіє з датчиком чи керованим пристроєм;
- *type* — атрибут, що описує тип модуля з бібліотеки підтримуваних модулів;

– *topic* — атрибут, який описує співвідношення з сенсором, опцією чи телеметрією конкретної ноди у форматі node-id/sensor-id;

– *options* — атрибут, в якому опціонально вказуються додаткові характеристики для роботи модуля; цей об'єкт відрізняється для кожного модуля і є унікальним в межах кожного.

На рисунку 3.4 зображено приклад об'єкта mapping з варіацією декількох модулів.

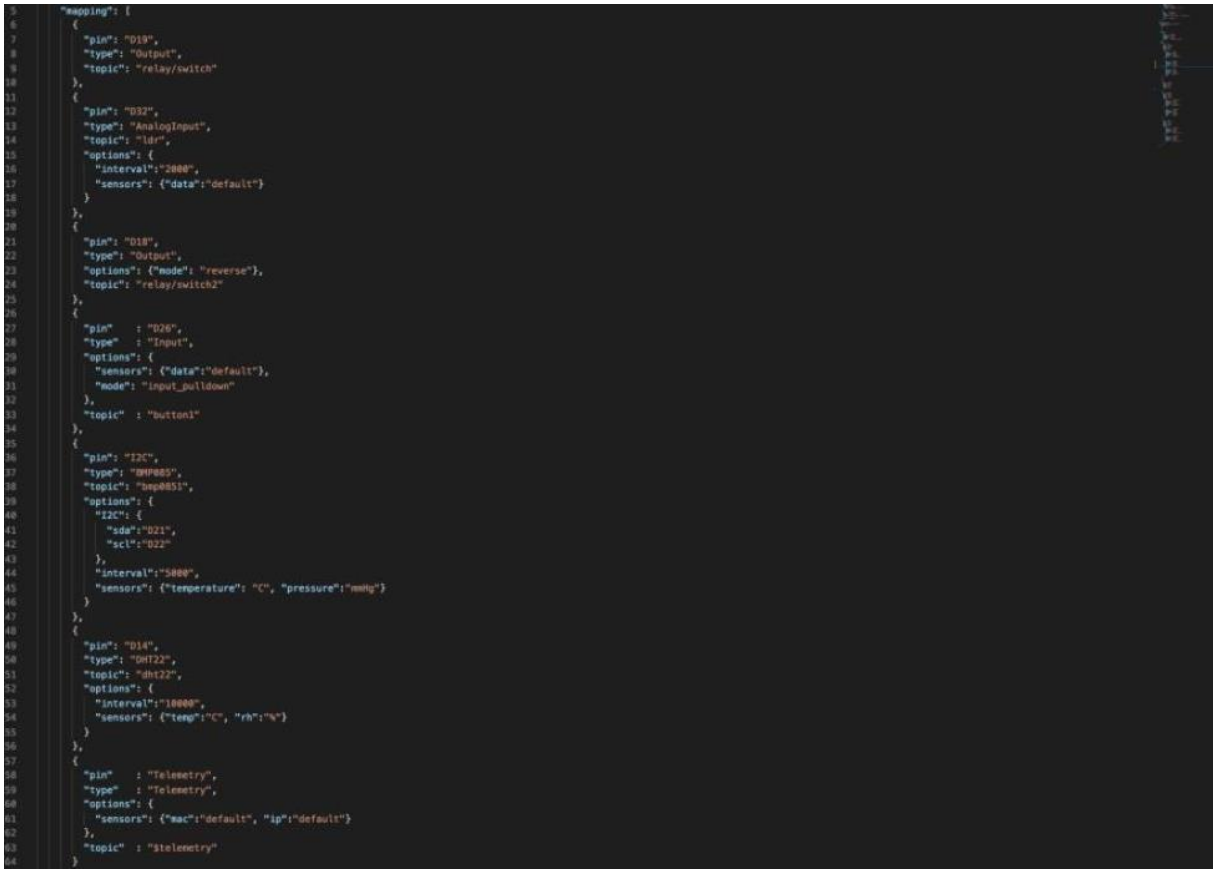


Рисунок 3.4 - Приклад об’єкта з описом співвідношення модулів до сенсорів

Після створення структури конфігураційного файлу в основному виконуючому файлі була написана функція парсингу конфігурації. Ця функція збирає необхідні дані з інших файлів на основі описаної конфігурації та формує структуру топіків, які потрібно відправити в брокер і на які слід підписатися.

Конфігураційний файл піддається мініфікації та завантажується у вбудовану пам'ять пристрою. Функція звертається до цього файлу під час кожного

завантаження пристрою. Крім того, функція відправляє в брокер необхідні за конвенцією топіки з потрібними значеннями за замовчуванням (рисунок 3.5).

```
prefix = rTopic + '/' + credentials.mqttName.replace(/:/g, '-') + '/';
const pinValueTopic = prefix + '$options/values';
const loggerTopic = prefix + '$options/logger';
const heartbeatTopic = prefix + '$heartbeat';
const publish = (t, v, o) => mqtt.publish(t, v, o || {retain: true, qos: 1});
mqtt.on('connected', function() {
    logger('MQTT_CONNECTED', { isRec });
    if (isRec) E.reboot(); // HARDCODE REBOOT ON RECONNECT

    setTimeout(() => {
        // DEVICE INITIALIZATION
        publish(prefix + '$homie', '3.0.1');
        publish(prefix + '$fw/name', '2smart');
        publish(prefix + '$fw/version', '0.9');
        publish(prefix + '$state', 'ready');
        publish(prefix + '$implementation', '0.9');
        publish(prefix + '$mac', telemetry.mac);
        publish(prefix + '$localip', telemetry.ip);

        publish(pinValueTopic, global.defaultPinValue); // publish default pin option value
        publish(loggerTopic, global.mqttLogger); // publish logger state value

        logger('PUBLISH_DEVICE');

        global.confLen = +fs.read('config_length');

        for(var j = 1; j <= global.confLen; j++) {
            global.pub = fs.readJSON('config' + j);

            if (!global.pub) break;

            global.pub.p.forEach((t, idx) => {
                if (idx % 2) return;
                publish(prefix + t, global.pub.p[idx + 1])
            });

            delete global.pub;
        }

        delete global.confLen;
    }, 1000);

    setTimeout(() => {
        logger('START_READING_CONFIG');
        global.config = fs.readJSON('config');

        var modules = fs.read('modules').split(',');
        modules.forEach(module => {
            global["modules"][module] = (new Function('exports',fs.read(module)+'return exports'))({});
        });
        options.telemetry = telemetry;
        global.config.nodes.forEach(node => {
            global["nodeMap"][node.topic] = {};
            global["nodeMap"][node.topic]['config'] = node;
            logger('NODE_CONFIG', { node }, false);
            if (typeof node.options != 'undefined' &&  typeof node.options.pinMap != 'undefined'){
            } else {
                publish(prefix + node.topic, global.defaultPinValue);
            }
            options.moduleOpts = typeof node.options === 'object' ? node.options : false;
            if (global["modules"][node.type]) {
            }
        });

        logger('END_READING_CONFIG');
    }, 2000);
```

Рисунок 3.5 - Функція зчитування та обробки даних

3.2 Створення бази для роботи з сенсорами

Для забезпечення повноцінного функціонування програмного забезпечення необхідно підключити відповідні модулі для роботи з фізичними датчиками та пристроями. Існує багато підходів до зчитування даних і відправки команд на пристрої, причому кожен зовнішній пристрій часто вимагає індивідуального підходу.

Щоб зменшити витрати пам'яті та ресурсів мікроконтролера, було вирішено винести бізнес-логіку роботи з кожним типом пристрою в окремі модулі й

підключати їх лише за необхідності, тобто коли вони зазначені у файлі конфігурації.

У процесі розробки проєкту було створено модулі для найпопулярніших сторонніх типів датчиків і шин, а також передбачена можливість створення нових модулів користувачами чи розробниками.

Для правильної роботи з модулями, файл із бізнес-логікою має бути розміщений в окремій папці згідно з документацією і названий відповідно до імені, вказаного у файлі конфігурації.

Output. Найпростішим прикладом є модуль для роботи з керованими виходами мікроконтролера. Гарним прикладом є ввімкнення світлодіода, підключеного до певного контакту мікроконтролера, або ввімкнення/вимкнення реле. Цей модуль було названо output, а бізнес-логіку для роботи з ним поміщено у файл *Output.js* (рисунок 3.6).

Приклад роботи з цифровими пристроями (output) було взято зі стандартної бібліотеки Espruino. Модуль дозволяє надсилати на цифровий вихід керуючі команди 1 або 0, ввімкнення або вимкнення відповідного виходу. Стан цифрового пристрою відправляється в брокер у топіки з типом даних *boolean*, де 1 відповідає значенню *true*, а 0 — *false*.

Модуль підтримує додатковий режим реверсу, що описується в розширених налаштуваннях конфігураційного файлу та переводить режим роботи в інвертований. У цьому режимі значенню 1 відповідає *false*, а 0 — *true*.

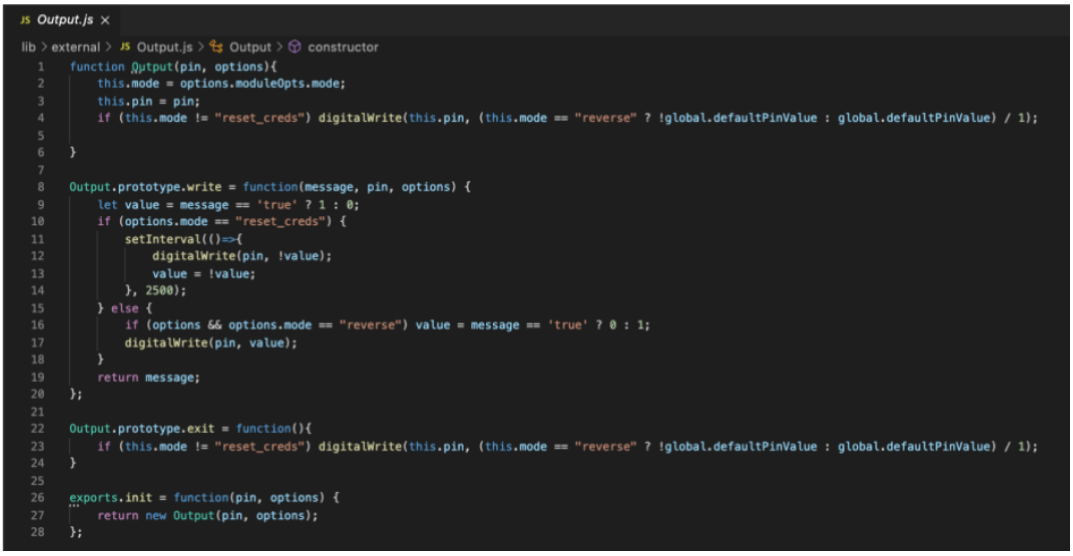


Рисунок 3.6 - Модуль Output

Input. Наступним базовим модулем був *Input.* Це модуль, який використовується для прийому сигналів з пристроїв або датчиків, що працюють у режимах *input*, *input_pullup*, або *input_pulldown* (отримання сигналу з пристрою на контролер).

Режими роботи модуля:

- *input_pull_X* — методи змінюють стан один раз при тривалому або нетривалому сигналі;
- *input_pullup* — пристрої працюють в інверсному режимі (Цифровий вхід з внутрішнім підтягуючим резистором ~ 40 кОм);
- *input_pulldown* — пристрої працюють у звичайному режимі (Цифровий вхід з внутрішнім стягуючим резистором ~ 40 кОм);
- *input* — цифровий вхід для постійного відстеження стану.

На рисунку 3.7 зображений код, що забезпечує роботу з сенсорами цього типу.

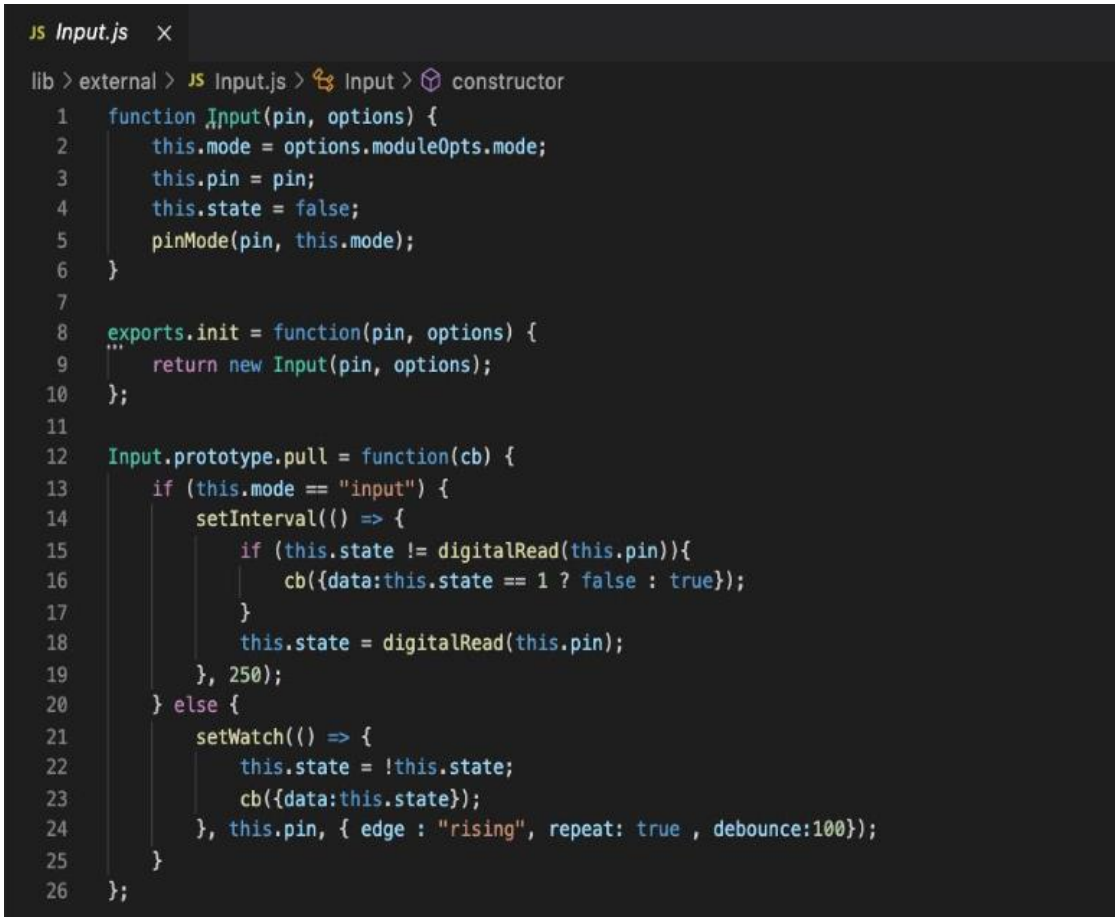


Рисунок 3.7 - Модуль Input

AnalogInput**.** *AnalogInput* - це модуль, призначений для отримання сигналу від пристроїв або датчиків, які працюють у режимі input та надсилають аналогові значення до контролера.

У модулі *AnalogInput* реалізована функціональність reverse, яка дозволяє використовувати порт у зворотньому режимі. Для цього необхідно вказати *"options"*: *{"mode": "reverse"}* (рисунок 3.8).

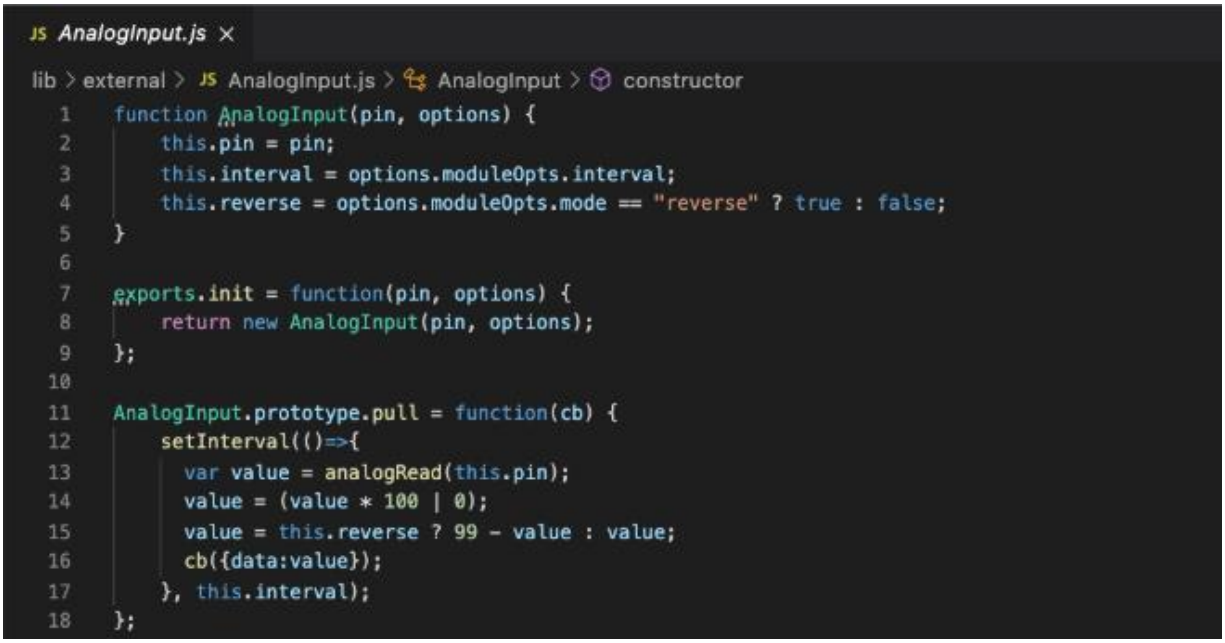


Рисунок 3.8 - Модуль AnalogInput

Telemetry. *Telemetry* - це додатковий стандартний модуль, який дозволяє передавати мак-адресу та IP-адресу пристрою до брокера у локальній мережі. Цей модуль створено для зручності користувача та швидкого доступу до параметрів пристрою.

Для налаштування модуля на конкретні порти у mapping необхідно вказати поле type ім'ям модуля, а ключі поля sensors - назвами полів об'єкту, які повертаються з модуля, наприклад, *{"Mac": "default", "ip": "default"}*, а значення - величиною одиниці виміру (рисунок 3.9). Функція для отримання мак- та IP-адреси є стандартною у Espruino.

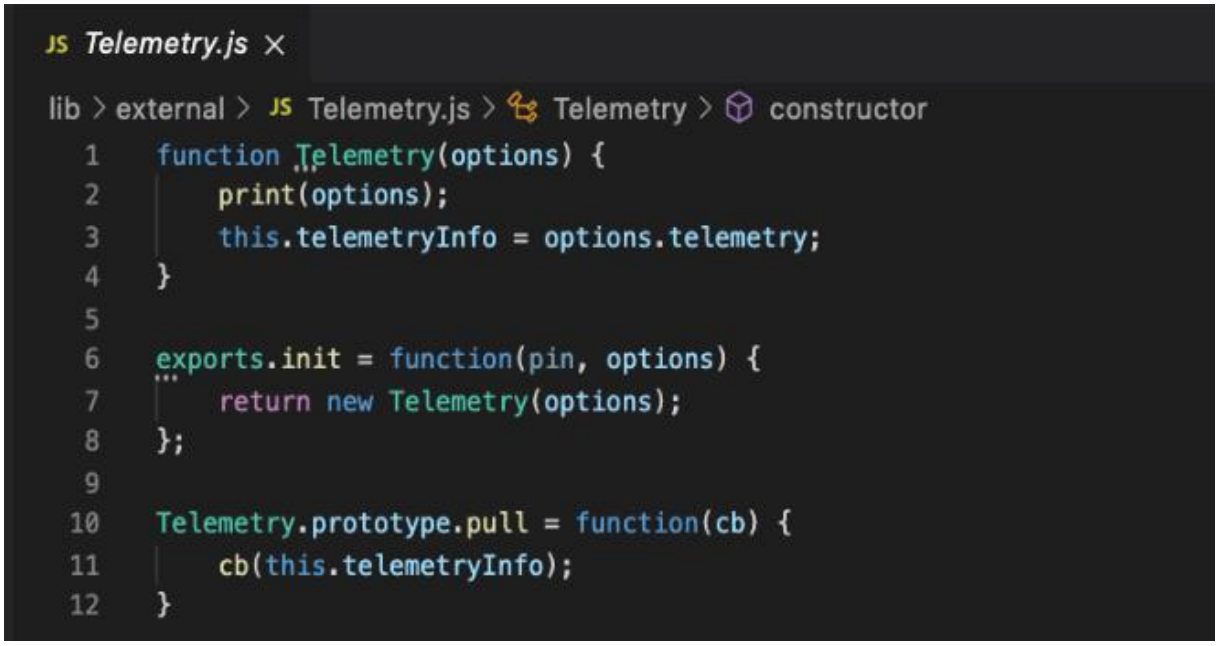


Рисунок 3.9 - Модуль Telemetry

DHT22. Модуль DHT22 призначений для зчитування значень з датчиків температури і вологості типу DHT2x / AM230x / RHT0x. Він реалізований специфічно під ці типи сенсорів, не так універсально, як попередні.

Для налаштування модуля на певні порти в *mapping* необхідно вказати поле *type* ім'ям модуля. В поле *options.interval* слід вказати значення інтервалу зчитування інформації з датчиків DHT, де значення задається в мілісекундах.

Доступні значення для температури та вологості: *"C"* - для Цельсія, *"K"* - для Кельвіна, а також *"%"* - для відсоткового відображення вологості.

Цей модуль має додаткову логіку роботи з пристроєм (рисунок 3.10).

DHT22.js

lib > external > DHT22.js > DHT22 > constructor

```
function DHT22(pin, options) {
  this.pin = pin;
  this.interval = options.moduleOpts.interval;
  this.sensors = {};
  for (sensor in options.moduleOpts.sensors){
    this.sensors[sensor] = {};
    switch(options.moduleOpts.sensors[sensor]) {
      case "K":
        this.sensors[sensor].divider = 1;
        this.sensors[sensor].offset = 273.15;
        break;
      case "C":
        this.sensors[sensor].divider = 1;
        break;
      case "%":
        this.sensors[sensor].divider = 1;
        break;
    }
  }
}

DHT22.prototype.read = function (cb, n) {
  if (!n) n=10;
  var d = "";
  var ht = this;
  digitalWrite(ht.pin, 0);
  pinMode(ht.pin,"output"); // force pin state to output
  // start watching for state change
  this.watch = setWatch(function(t) {
    d+=0|(t.time-t.lastTime>0.00005);
  }, ht.pin, {edge:'falling',repeat:true} );
  // raise pulse after 1ms
  setTimeout(function() {pinMode(ht.pin,'input_pullup');pinMode(ht.pin);},1);
  // stop looking after 50ms
  setTimeout(function() {
    if(ht.watch){ ht.watch = clearWatch(ht.watch); }
    var cks =
        parseInt(d.substr(2,8),2)+
        parseInt(d.substr(10,8),2)+
        parseInt(d.substr(18,8),2)+
        parseInt(d.substr(26,8),2);
    if (cks&&((cks&0xFF)==parseInt(d.substr(34,8),2))) {
      cb({
        rh : parseInt(d.substr(2,16),2)*0.1,
        temp : parseInt(d.substr(19,15),2)*0.2*(0.5-d[18]) + (ht.sensors.temp.offset || 0)
      });
    } else {
      if (n>1) setTimeout(function() {ht.read(cb,--n);},500);
      else cb({temp:"-", rh:"-"});
    }
  }, 50);
};

exports.init = function(pin, options) {
    return new DHT22(pin, options);
};

DHT22.prototype.pull = function(cb) {
    setInterval(()=>{
        this.read(cb);
    }, this.interval);
};
```

Рисунок 3.10 - Модуль DHT22

Додаткові модулі. Додаткові модулі розроблено для взаємодії з різними типами сенсорів та пристроями, а також передбачено можливість створення нових модулів користувачами з експертним рівнем знань.

Короткий опис кожного з цих додаткових модулів:

– BMP085: Зчитує значення температури і атмосферного тиску з цифрових барометрів BMP085 / BMP180 через інтерфейс I2C;

– Register: Дозволяє працювати з пристроями, що контролюються зсувним регістром;

– UltraSonicSensor: Вимірює відстань до перешкоди за допомогою ультразвукового сонара HC-SR04;

– MoveSensor: Визначає рух у просторі за допомогою піроелектричного датчика руху HC-SR50;

– CCS811: Робота з датчиком якості повітря в приміщеннях; визначає рівні TVOC і eCO2 за допомогою цифрового газового датчика по засобу інтерфейсу I2C;

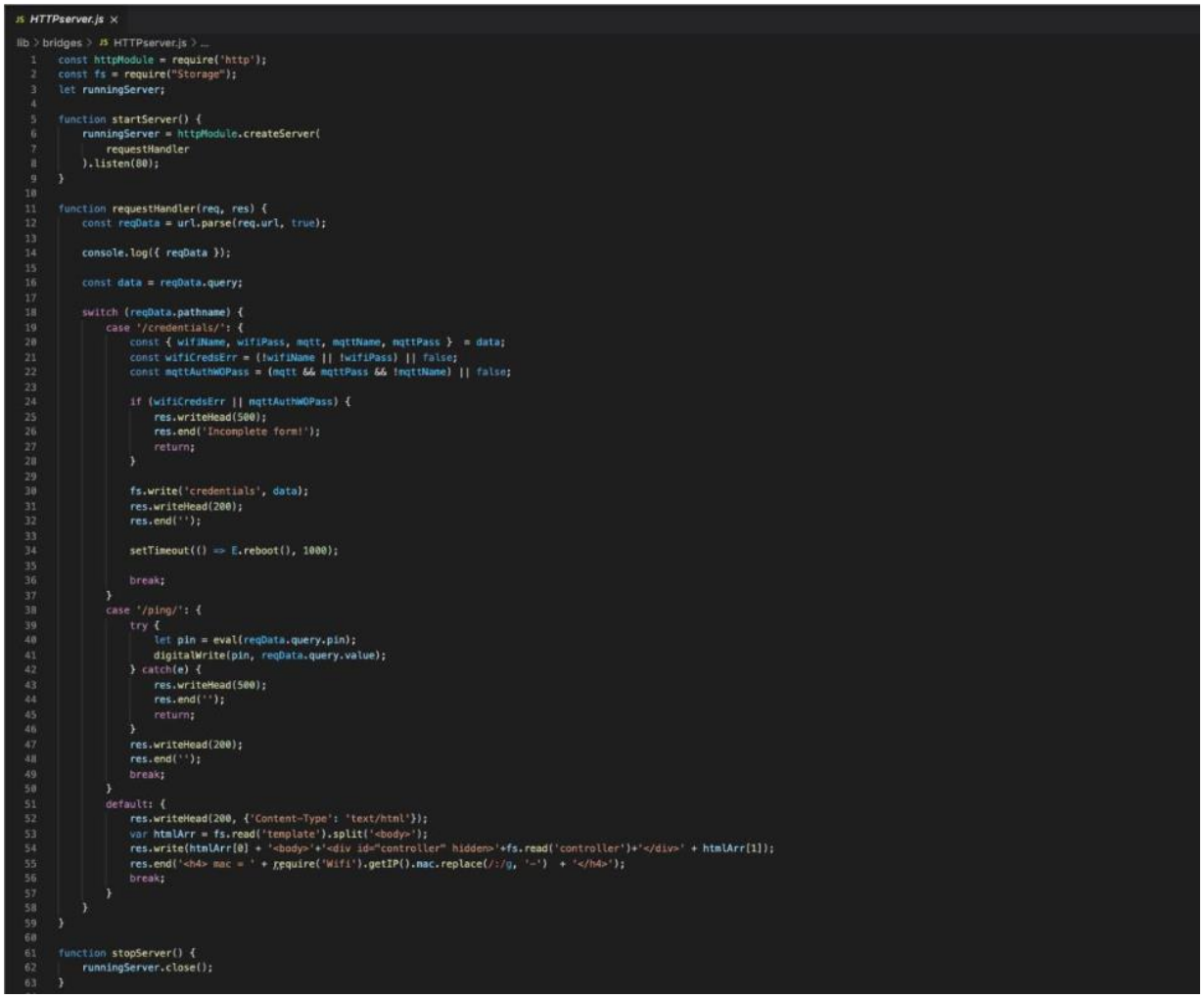
– DS18B20: Зчитує значення з датчика температури DS18B20.

Після розробки кожного модулю була створена функція для парсингу модулів, які потрібно підключити, та їх включення в фінальну збірку. Код кожного модулю зберігається на мікроконтролері і виконується при виклику з основної функції.

3.3 Створення інтерфейсу для підключення до локальної мережі

Після того, як пристрої були програмно налаштовані, потрібно передати їм параметри для підключення до мережі та адресу брокера. У початковій версії програмного забезпечення ці параметри зберігалися у файлі конфігурації, але з часом стало очевидно, що такий підхід непрактичний і вимагає постійного перепрограмування при навіть найменших змінах.

Для забезпечення зручності користування пристроями було вирішено створити постійний веб-сервер, який прийматиме REST-запити з необхідними параметрами (рисунок 3.11). Першим кроком було програмне налаштування серверної частини, яка автоматично запускалася на мікроконтролері після його завантаження.



```
lib > bridges > JS HTTPserver.js > ...
const httpModule = require('http');
const fs = require("Storage");
let runningServer;

function startServer() {
    runningServer = httpModule.createServer(
        requestHandler
    ).listen(80);
}

function requestHandler(req, res) {
    const reqData = url.parse(req.url, true);

    console.log({ reqData });

    const data = reqData.query;

    switch (reqData.pathname) {
        case '/credentials/': {
            const { wifiName, wifiPass, mqtt, mqttName, mqttPass } = data;
            const wifiCredsErr = (!wifiName || !wifiPass) || false;
            const mqttAuthWOPass = (mqtt && mqttPass && !mqttName) || false;

            if (wifiCredsErr || mqttAuthWOPass) {
                res.writeHead(500);
                res.end('Incomplete form!');
                return;
            }

            fs.write('credentials', data);
            res.writeHead(200);
            res.end('');

            setTimeout(() => E.reboot(), 1000);

            break;
        }
        case '/ping/': {
            try {
                let pin = eval(reqData.query.pin);
                digitalWrite(pin, reqData.query.value);
            } catch(e) {
                res.writeHead(500);
                res.end('');
                return;
            }
            res.writeHead(200);
            res.end('');
            break;
        }
        default: {
            res.writeHead(200, {'Content-Type': 'text/html'});
            var htmlArr = fs.read('template').split('<body>');
            res.write(htmlArr[0] + '<body>'+'<div id="controller" hidden>'+fs.read('controller')+'</div>' + htmlArr[1]);
            res.end('<h4> mac = ' + require('Wifi').getIP().mac.replace(/:/g, '-')  + '</h4>');
            break;
        }
    }
}

function stopServer() {
    runningServer.close();
}
```

Рисунок 3.11 - Веб сервер для прийому запитів

Сервер приймає наступні необхідні параметри:

– назву локальної мережі;

– пароль для підключення до локальної мережі;

– адрес MQTT брокеру;

– логін для підлючення до MQTT брокеру;

– пароль для підлючення до MQTT брокеру.

За замовчуванням, пристрій перевіряє наявність параметрів у файловій системі. Якщо параметри відсутні, пристрій автоматично переходить у режим роздачі точки доступу за замовчуванням. У цьому режимі користувач може підключитись до мережі і надіслати необхідні параметри через REST-запит. Після

отримання параметрів, пристрій записує їх у файлову систему, автоматично перезавантажується і підключається до вказаної мережі та брокера.

Для очищення отриманих параметрів було розроблено окремий системний модуль, який видаляє параметри з файлової системи після взаємодії з кнопкою, що з'єднана з потрібним контактом мікроконтролера.

Для зміни параметрів підключення пристрою до мережі можна використовувати перехід в режим точки доступу шляхом довгого замикання *GPIO0* (для ESP32 / ESP8266 це кнопка boot або flash). Для цього слід підключити модуль *Output* в *extensions.modules*, а потім вказати в *mapping* режим модуля *Output* в *reset_creds.* (рисунок 3.12).

Рисунок 3.12 - Приклад налаштування модуля зміни параметрів підключення

Отже, можна ініціювати зміну параметрів шляхом фізичного замикання кнопки на відповідній ніжці пристрою або через брокер, надіславши в потрібний топік значення true.

Наступним кроком була розробка базового веб-інтерфейсу для передачі параметрів доступу. Для цього було створено просту HTML-сторінку з формою, що містить необхідні поля для введення кожного з параметрів (рисунок 3.13).

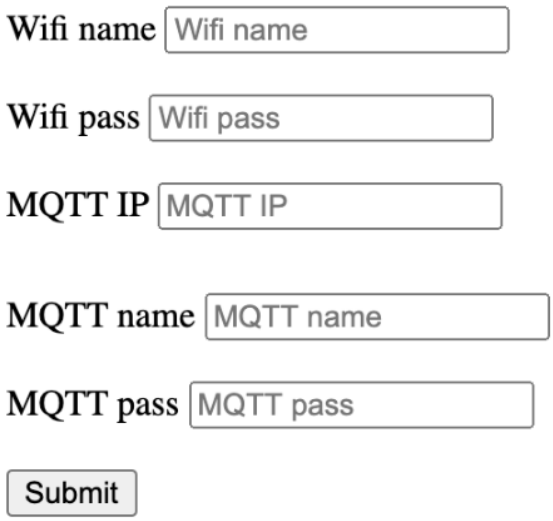


Рисунок 3.13 – Базова веб сторінка

Таким чином, за замовчуванням ця веб-сторінка буде доступна за IP-адресою веб-сервера 192.168.4.1, коли пристрій перебуває в режимі точки доступу. Після підключення пристрою до локальної мережі буде можливою адресація цієї сторінки за IP-адресою самого пристрою в локальній мережі.

3.4 Тестування програмного забезпечення

Для тестування програмного забезпечення використовувалась стандартна реалізація MQTT брокера на основі відкритого коду від компанії EMQX. Брокер був налаштований на стандартному порту 1883 і розміщувався на домашньому персональному комп’ютері. Таким чином, пристрої з встановленим на них програмним забезпеченням можна було підключати до брокера за IP-адресою ПК та портом брокера.

Для розмежування параметрів доступу у інтерфейсі брокера була налаштована ACL (Access Control List), що включала базовий набір логінів та

паролів. Це дозволяло надавати доступ конкретному пристрою до відповідних областей топіків у брокері.

Отже, знавши параметри для підключення до локальної мережі, адресу брокера, а також логін та пароль, можна було передавати отримані дані через веб-інтерфейс.

На рисунку 3.14 зображено стан пристрою в брокері після підключення. Для моніторингу стану пристрою використовувалась стандартна утиліта MQTT Explorer, яка по суті є звичайним клієнтом брокера, але підключалась до нього з правами адміністратора, що надавало можливість переглядати та керувати всіма підключеними пристроями.

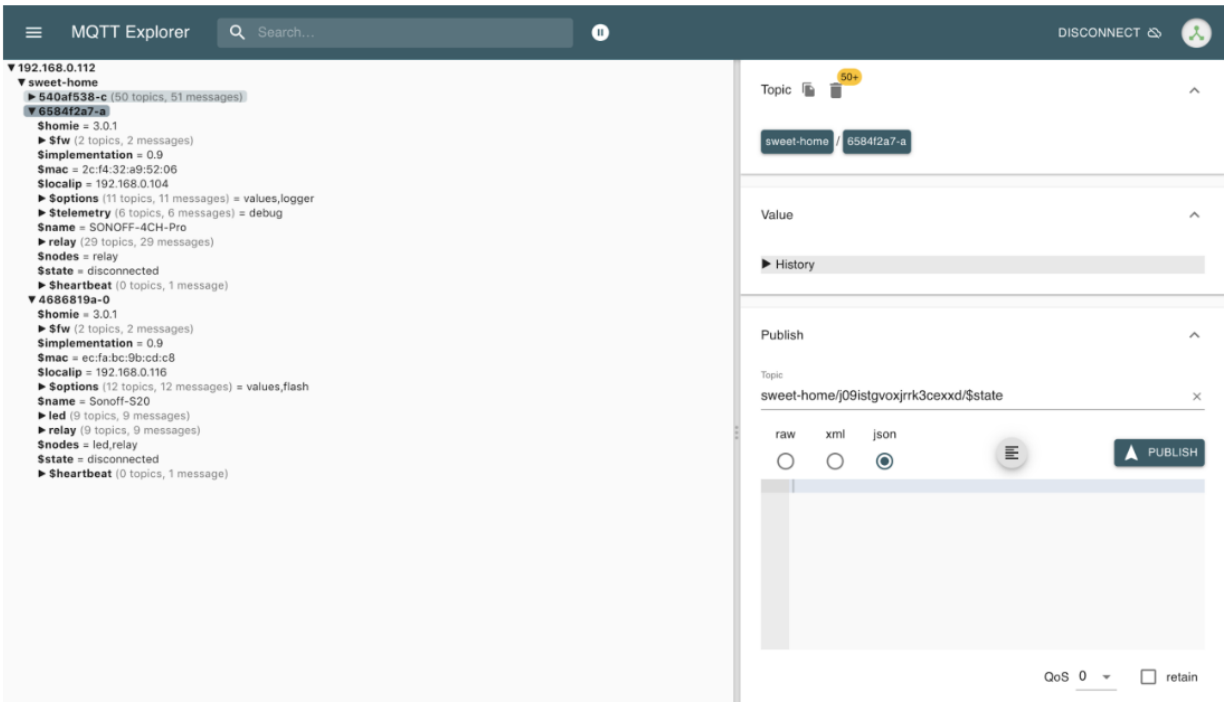


Рисунок 4.14 - Стан пристроїв в брокері

Для тестування всіх компонентів програмного забезпечення можна придбати різноманітні сенсори та датчики, а прототипи реальних пристроїв можна скласти на макетних платах за допомогою доступних інструкцій у відкритих джерелах.

Можна провести тестування наступних сценаріїв:

- увімкнення LED підсвітки та світлодіодів на макетних платах;

- отримання стану натискання стандартних кнопок на макетній платі;
- зчитування показників з сенсорів температури, тиску та вологості;
- вимірювання рівня освітлення за допомогою фоторезисторів;
- керування кількома виходами через здвигові регістри;
- вимірювання відстані до перешкод за допомогою відповідних датчиків;
- оцінка якості повітря та рівнів летючих речовин після калібрування відповідного сенсору.

Ці тестування можна виконати з використанням макетної плати та мікроконтролера ESP32. Отримані дані передаються через брокер MQTT, і їх можна переглядати за допомогою утиліти MQTT Explorer.

Наступним кроком є проведення тестування програмного забезпечення на реальних промислових пристроях. Для цього можна придбати низку пристроїв, що базуються на мікроконтролерах ESP, які можна легко встановити та використовувати з програмним забезпеченням.

Серед цих пристроїв є:

- Sonoff S20 (однопортова розетка);
- Sonoff Basic (однопортове реле);
- Sonoff 4CH (чотирьохпортове реле);
- Sonoff Dual (двопортове реле);
- Blitz Relay (однопортове реле);
- Blitz SHP2 (однопортовий перехідник в розетку).

На цих шести пристроях можна провести тестування програмного забезпечення у реальних користувацьких сценаріях. Різноманітні пристрої (лампи, обігрівачі, пристрої для контролю вологості тощо) можна підключити до них. Крім того, пристрій Sonoff Dual має вбудований вихід для підключення аналогових датчиків температури, до якого можна підключити один з таких сенсорів, і значення температури буде замірятися в реальному часі.

Пристрої на макетних платах можуть бути протестовані, але не у довгостроковому режимі, тоді як пристрої від реальних виробників були піддані

випробуванням у тривалих і стресових умовах. Це включало реакцію на електроперерви, стабільну роботу під час втрати зв'язку в локальній мережі та автоматичне підключення після відновлення зв'язку з ПК та брокером.

Тестувальні скрипти дозволили виконати включення та вимикання пристроїв згідно заданого графіку, що дозволило відтворити реальні умови експлуатації, наприклад, автоматичне включення освітлення для домашніх тварин.

Після успішного тестування за допомогою стандартних інструментів та скриптів, було проведено низку тестів для перевірки можливості інтеграції з іншими платформами. Однією з обраних платформ стала HomeBridge, яка надала можливість голосового керування та управління через веб-інтерфейс та мобільний додаток Home для iOS.

Для експерименту платформа HomeBridge була розгорнута на домашньому ПК згідно з офіційними інструкціями та підключена до раніше налаштованого брокера.

Для налаштування було встановлено декілька плагінів для роботи з брокером та двома голосовими асистентами (Siri та Google) (рисунок 3.15).

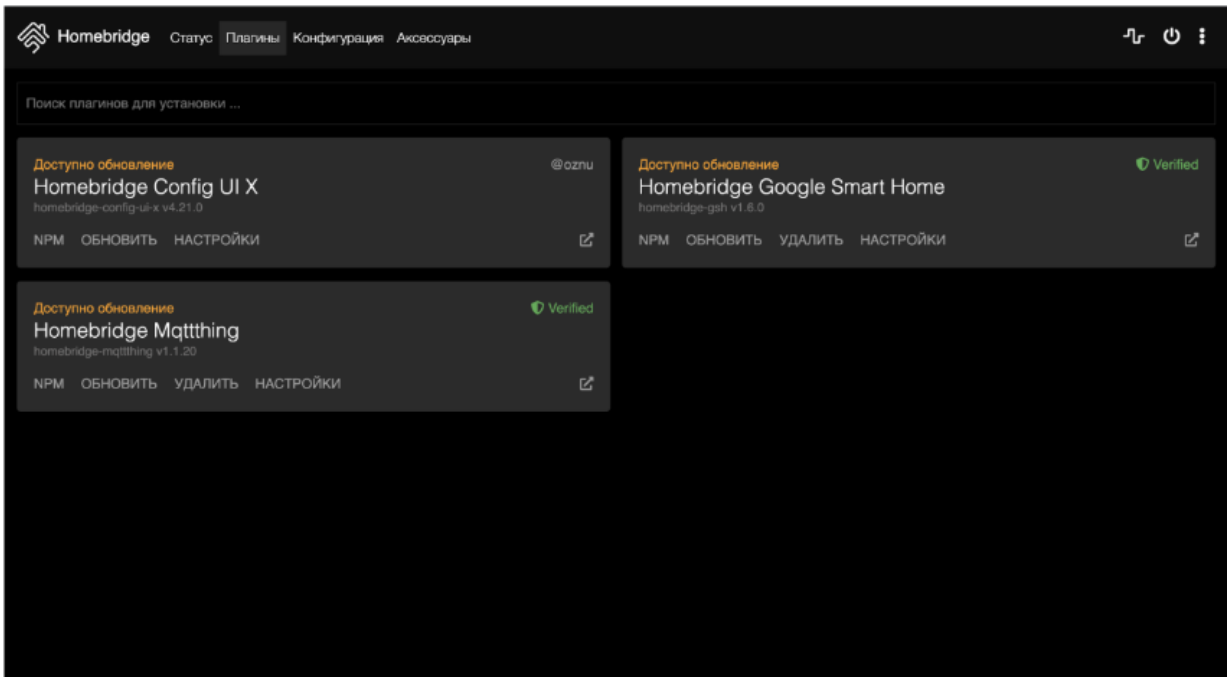


Рисунок 3.15 - Встановлення необхідних плагінів

Після налаштування плагінів було сконфігуровано файл налаштувань для роботи з топіками реального пристрою. Цей процес підкреслив можливість абстрагування від конкретних платформ та надання різним платформам можливості працювати з топіками у вигляді абстракцій.

Конфігураційний файл містить опис аксесуарів, які потім відображаються в веб-інтерфейсі та взаємодіють з голосовими асистентами. Документація HomeBridge містить опис всіх доступних аксесуарів, а для тестування було обрано стандартний перемикач для керування навантаженням.

Після збереження файлу конфігурації, в веб-інтерфейсі HomeBridge з'явився компонент для взаємодії з аксесуаром та відображення його стану та логів (рисунок 3.17).

Наступним кроком було налаштування голосових асистентів Siri та Google за допомогою відповідних плагінів, які були встановлені на попередньому етапі (рисунок 3.16).

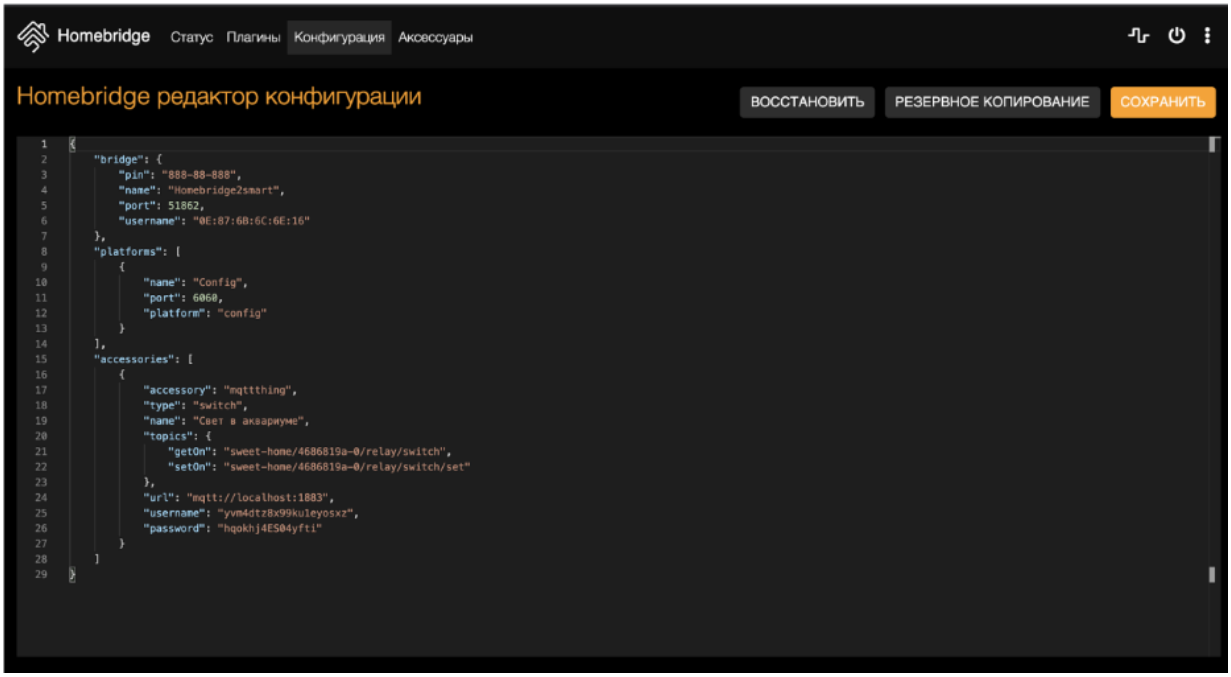


Рисунок 3.16 - Файл конфігурації в HomeBridge

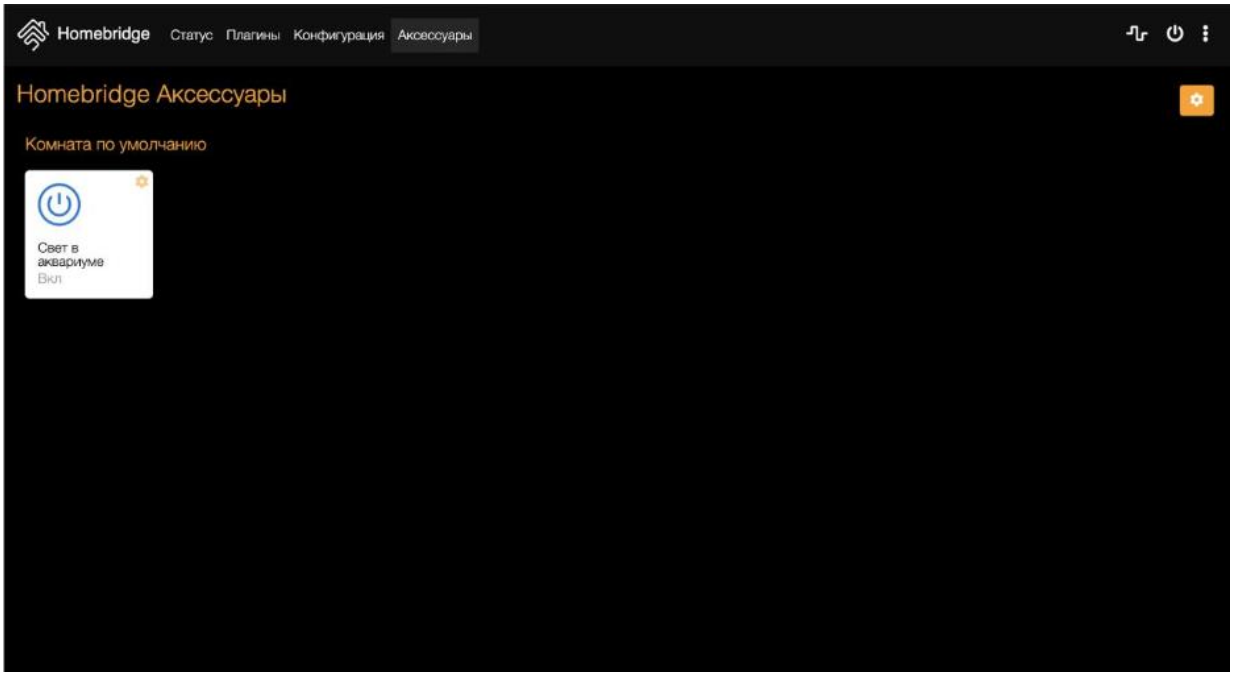


Рисунок 3.17 - Компонент для взаємодії з пристроєм

Для налаштування інтеграції з голосовими асистентами досить просто слідувати інструкціям та встановити відповідні додатки на мобільний телефон.

Після виконання цих кроків, що займають не більше трьох хвилин, пристрій, яким керує розроблене програмне забезпечення, став доступний в додатку Home для iOS та Google Home. Це дозволило вмикати та вимикати світло для домашніх улюбленців за допомогою голосових команд. Такий простий інтерфейс став можливим завдяки використанню голосових асистентів (рисунок 3.18).

За таким же принципом можна легко додавати багато інших аксесуарів, доступних в HomeBridge, таких як сенсори температури, руху, освітлення, датчики відкриття вікон та дверей, і сенсори якості повітря. Голосові асистенти взаємодіють із цими аксесуарами належним чином і відповідають на запити користувача [45].

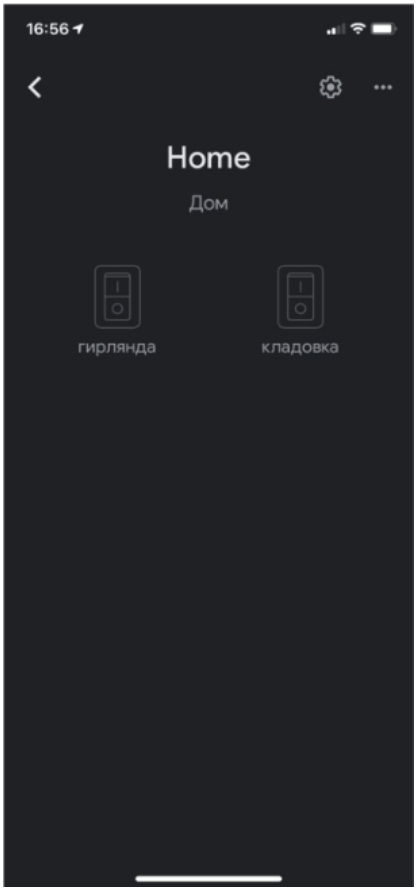


Рисунок 3.18 - Інтеграція з голосовим асистентом

3.5 Висновки

У третьому розділі дослідження була розроблена архітектура програмного забезпечення для пристроїв розумного будинку. В процесі розробки було вирішено наступні завдання:

1. Розроблено бібліотеку модулів для взаємодії з популярними датчиками та пристроями.

2. Впроваджено механізм для створення нових модулів користувачами.

3. Створено інструмент командного рядка (CLI) для встановлення та оновлення програмного забезпечення на пристроях.

4. Реалізовано підтримку спеціалізованої IoT конвенції.

5. Розроблено веб-сервер та інтерфейс для підключення пристроїв до локальної мережі.

6. Створено бібліотеку для конфігураційних файлів для тестування модулів у різних комбінаціях.

7. Інтегровано з платформою HomeBridge.

8. Налаштовано голосове керування за допомогою асистентів Siri та Google.

На основі результатів тестування було зроблено висновок про працездатність програмного забезпечення та відповідність його всім поставленим задачам. Існуючі архітектурні рішення були покращені, включаючи введення технології MQTT для абстрагування пристроїв від платформи, розробку механізмів універсального підключення до будь-яких брокерів, можливість збору конфігурацій з простого файлу, а також створення розширюваної бібліотеки модулів.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було визначено, що пристрої на базі мікроконтролерів ESP становлять значну частину ринку сучасних розумних пристроїв, що працюють за технологією Wi-Fi. В роботі розроблено програмне забезпечення, яке розв'язує проблеми, що існують у наявних аналогах.

У процесі розробки проведено аналіз сучасних технологій, які використовуються у смарт-пристроях, та вибрали оптимальні для вирішення наших завдань. Проаналізовано багато платформ для інтеграції пристроїв в єдину екосистему та розробили стратегію для універсалізації підходу до роботи з більшістю з них. В результаті аналізу обрано протокол MQTT для передачі даних між пристроями та системою, що дозволило абстрагуватися від системи управління.

Після вибору технологій та концепції обрано мікроконтролер ESP як базу для розробки пристроїв. Запропонована архітектура є універсальною, гнучкою та масштабованою, що дозволяє навіть недосвідченим користувачам інтегрувати свої пристрої в будь-яку систему розумного будинку за допомогою стандартних методів інтеграції.

В роботі інтегровано пристрої з різними системами контролю розумного будинку та платформами автоматизації, а також проведено стрес-тестування в реальних умовах використання.

В результаті розроблено проект для створення програмного забезпечення для великої кількості пристроїв з можливістю інтеграції в сучасні системи автоматизації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Smart home and appliances: State of the art. URL: https://publications.jrc.ec.europa.eu/repository/bitstream/JRC113988/kjna29750enn_1.pdf (дата звернення: 15.05.2024).
2. Шлюз BACnet і маршрутизатор BACnet в IoT- DusunIoT. URL: https://www.dusuniot.com/uk/blog/what-is-a-bacnet-gateway-and-bacnet-router-in-iot-application/ (дата звернення: 20.05.2024).
3. fieldbusbook | Посібник по промисловим мережам. URL: https://pupenasan.github.io/fieldbusbook/2010/2_5_5.html (дата звернення: 20.05.2024).
4. Розумний дім: програмне забезпечення для автоматизації з відкритим кодом URL: https://www.linuxadictos.com/uk/%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D1%96%D1%8F-%D1%80%D0%BE%D0%B7%D1%83%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE-%D0%B1%D1%83%D0%B4%D0%B8%D0%BD%D0%BA%D1%83-%D0%B7-%D0%B2%D1%96%D0%B4%D0%BA%D1%80%D0%B8%D1%82%D0%B8%D0%BC-%D0%BA%D0%BE%D0%B4%D0%BE%D0%BC.html (дата звернення: 20.05.2024).
5. KNX - що це таке? - Безпека та відеоспостереження. URL: https://oxorona.com/knx/ (дата звернення: 20.05.2024).
6. Все про Ethernet. URL: https://medium.com/@mariiaradzhapova/%D0%B2%D1%81%D0%B5-%D0%BF%D1%80%D0%BE-ethernet-a1abcf429108 (дата звернення: 20.05.2024).
7. Бездротова сенсорна мережа. URL: https://uk.wikipedia.org/wiki/%D0%91%D0%B5%D0%B7%D0%B4%D1%80%D0%BE%D1%82%D0%BE%D0%B2%D0%B0_%D1%81%D0%B5%D0%BD%D1%81%D

0%BE%D1%80%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0 (дата звернення: 20.05.2024).

8. Insteon. URL: https://en.wikipedia.org/wiki/Insteon (дата звернення: 20.05.2024).

9. Домашня автоматизація за допомогою Інтернету речей URL: https://www.dusuniot.com/uk/blog/home-automation-using-iot/ (дата звернення: 20.05.2024).

10. Протоколи звязку для розумного дому - Wise Home URL: https://wisehome.com.ua/ua/scho-take-protokol-dlya-rozumnogo-budinku-6-populyarnih-vidiv/ (дата звернення: 20.05.2024).

11. D.M. Han, J.H. Lim Design and implementation of smart home energy management systems based on zigbee.IEEE Transactions on Consumer Electronics,, 56 (3), pp. 1417-1425 (2010) (дата звернення: 20.05.2024).

12. Бездротові стандарти. Що таке ZigBee? URL: https://www.sea.com.ua/ua/besprovodnye-komponenty/news/bezdrotovi-standarti-so-take-zigbee/ (дата звернення: 20.05.2024).

13. Опис технології Z-Wave. Статті URL: https://z-wave.com.ua/ua/a63077-opisanie-tehnologii-wave.html (дата звернення: 20.05.2024).

14. OpenHAB проти Домашнього помічника URL: https://dusuniot.com/uk/blog/openhab-vs-home-assistant-which-one-is-right-for-you/ (дата звернення: 22.05.2024).

15. Calaos, Open Source Home Automation URL: https://calaos.fr/en/ (дата звернення: 22.05.2024).

16. Home Assistant 101. Посібник для початківців URL: https://dou.ua/forums/topic/38947/ (дата звернення: 22.05.2024).

17. Smart living: improve your living comfort with home URL: https://renson.net/gd-gb/products/smart-living (дата звернення: 22.05.2024).

18. HomeGenie, the programmable automation intelligence URL: https://github.com/genielabs/HomeGenie (дата звернення: 22.05.2024).

19. ioBroker Smarthome URL: https://www.iobroker.net/ (дата звернення: 22.05.2024).

20. Home of FHEM URL: https://fhem.de/ (дата звернення: 25.05.2024).

21. ago control – open source home automation system URL: https://www.mysensors.org/controller/agocontrol (дата звернення: 25.05.2024).

22. Посібник Homebridge: Як зробити все HomeKit-сумісним URL: https://hitech.mediadoma.com/uk/posibnik-homebridge-jak-zrobiti-vse-homekit-sumisnim/ (дата звернення: 25.05.2024).

23. pimatic/pimatic: A home automation server and framework URL: https://github.com/pimatic/pimatic (дата звернення: 26.05.2024).

24. MyController URL: https://mycontroller.org/docs/overview/ (дата звернення: 26.05.2024).

25. Лек 2. Основи Node-RED - pupenasan.github.io URL: https://pupenasan.github.io/ProgIngContrSystems/%D0%9B%D0%B5%D0%BA%D1%86/2_nodered.html (дата звернення: 26.05.2024).

26. Home Assistant and MQTT: 4 Things You Could Build URL: https://emqx.medium.com/home-assistant-and-mqtt-4-things-you-could-build-61193fc03f73 (дата звернення: 02.06.2024).

27. Тестування навантаження з MQTT URL: https://dou.ua/forums/topic/33420/ (дата звернення: 02.06.2024).

28. Lightweight MQTT Benchmark Tool written in Erlang URL: https://github.com/emqx/emqtt-bench (дата звернення: 02.06.2024).

29. paho-mqtt 2.1.0 URL: https://pypi.org/project/paho-mqtt/#simple (дата звернення: 02.06.2024).

30. Lee, S., Kim, H., Hong, D. K., & Ju, H. . Correlation analysis of MQTT loss and delay according to QoS level. In 2013 IEEE The International Conference on Information Networking (ICOIN) (pp. 714-717) (2013, January). (дата звернення: 02.06.2024).

31. Lampkin V et al. Building smarter planet solutions with MQTT and IBM WebSphere MQ telemetry IBM, ITSO (2012) (дата звернення: 05.06.2024).

32. Що таке MQTT і для чого він потрібний URL: https://highload.today/uk/shho-take-mqtt-i-dlya-chogo-vin-potribnij/ (дата звернення: 05.06.2024).

33. Arduino Uno Microcontroller Board URL: https://www.mouser.lu/new/arduino/arduino-uno/?_gl=1*fubnrp*_gcl_au*NDQxMTM5OTgwLjE3MTg5MTc2MTI.*_ga*NTgxMDgwMTM5LjE3MTg5MTc2MTU.*_ga_15W4STQT4T*MTcxODkxNzYxNS4xLjAuMTcxODkxNzYxOS41Ni4wLjA. (дата звернення: 08.06.2024).

34. Arduino Pro Mini ATMEL M328P-CN 5В 16МГц URL: https://arduino.ua/prod2624-arduino-pro-mini-5v-16mgc-atmega328p-mu#:~:text=%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80%20Arduino%20Pro%20Mini%20(ATMEL,%D0%9C%D1%96%D0%BA%D1%80%D0%BE%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80%3A%20ATmega328P%20(%D0%BA%D0%BB%D0%BE%D0%BD) (дата звернення: 08.06.2024).

35. Мікроконтролер ESP32 URL: https://itmaster.biz.ua/directory/microcontrollers/esp32.html (дата звернення: 08.06.2024).

36. ESP8266 Архіви - Безпека та відеоспостереження URL: https://oxorona.com/tag/esp8266/#:~:text=ESP8266%20%E2%80%93%20%D1%86%D0%B5%20%D0%B5%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D1%96%D1%87%D0%BD%D0%B8%D0%B9%20%D1%96%20%D0%B2%D0%B8%D1%81%D0%BE%D0%BA%D0%BE%D1%96%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9,%D0%B2%D0%B5%D0%BB%D0%B8%D0%BA%D0%BE%D1%97%20%D0%BF%D0%BE%D0%BF%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%96%20%D0%B7%D0%B0%D0%B2%D0%B4%D1%8F%D0%BA%D0%B8%20%D0

%BD%D0%B8%D0%B7%D1%8C%D0%BA%D1%96%D0%B9%20%D1%86%D1%96%D0%BD%D1%96.&text=%D0%9A%D1%80%D1%96%D0%BC%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D0%B0%20Wi%2DFi%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80,bit%20%D0%90%D0%A6%D0%9F%2C%20SPI%20%D1%82%D0%B0%20UART. (дата звернення: 08.06.2024).

37. Що таке мікрокомп'ютери? Розповідаємо на прикладі Raspberry Pi URL: https://e-server.com.ua/uk/poradi/shho-take-mikrokompiuteri-rozpovidajemo-na-prikladi-raspberry-pi (дата звернення: 08.06.2024).

38. Raspberry Pi Zero 2 W - Безпека та відеоспостереження URL: https://oxorona.com/raspberry-pi-zero-2-w/ (дата звернення: 08.06.2024).

39. Чи можна програмувати esp 32 на Python? URL: https://arduino.ua/ru/art239-chi-mojna-programyvati-esp-32-na-python (дата звернення: 14.06.2024).

40. CIRCUITPYTHON ТА MICROPYTHON ДЛЯ MEOWBIT URL: https://www.inforum.in.ua/conferences/24/77/558 (дата звернення: 14.06.2024).

41. About Arduino URL: https://www.arduino.cc/en/about (дата звернення: 14.06.2024).

42. Mongoose OS - an IoT firmware development framework URL: https://mongoose-os.com/mos.html (дата звернення: 14.06.2024).

43. Espruino Pico URL: https://www.espruino.com/Pico (дата звернення: 14.06.2024).

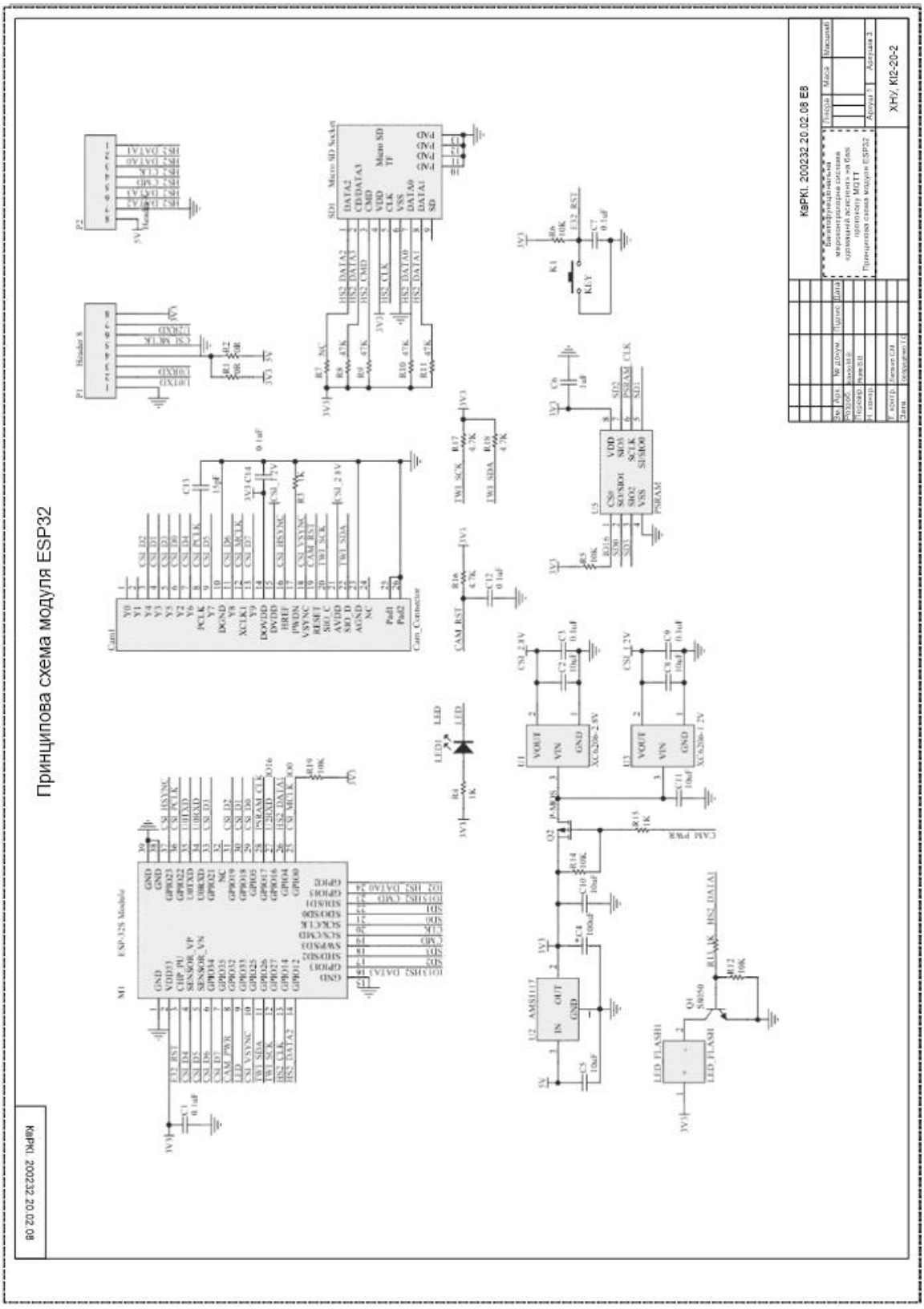
44. O. Vermesan and P. Friess (Eds.). Digitizing the Industry - Internet of Things Connecting the Physical, Digital and Virtual Worlds, ISBN: 978-87-93379- 82-4, River Publishers, Gistrup, (2016) (дата звернення: 18.06.2024).

45. Kumar, S. Ubiquitous smart home system using android application. arXiv preprint arXiv:1402.2114 (2014). (дата звернення: 18.06.2024).

Додаток А

(обов'язковий)

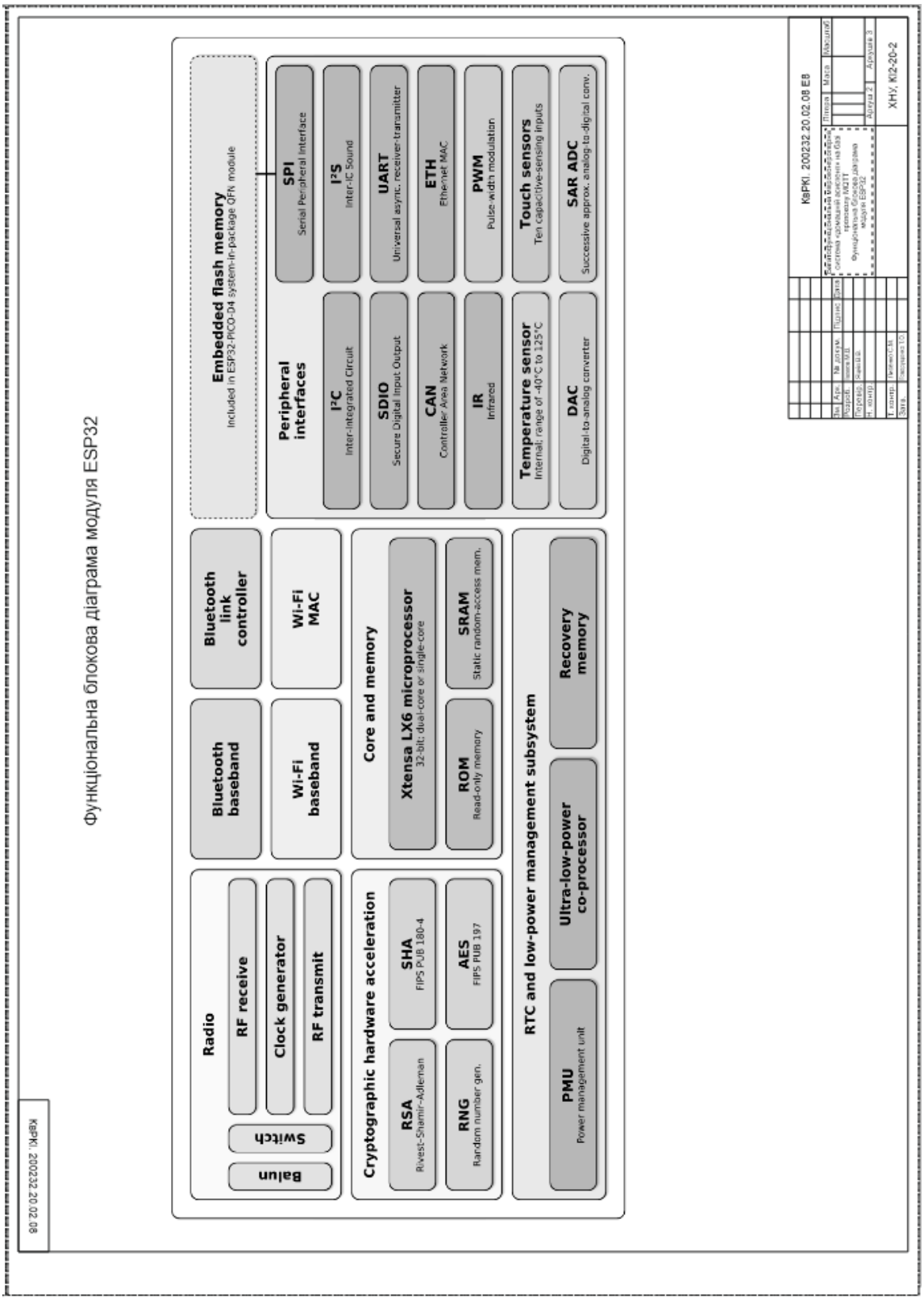
Копія креслення «Принципова схема модуля ESP32»



Додаток Б

(обов'язковий)

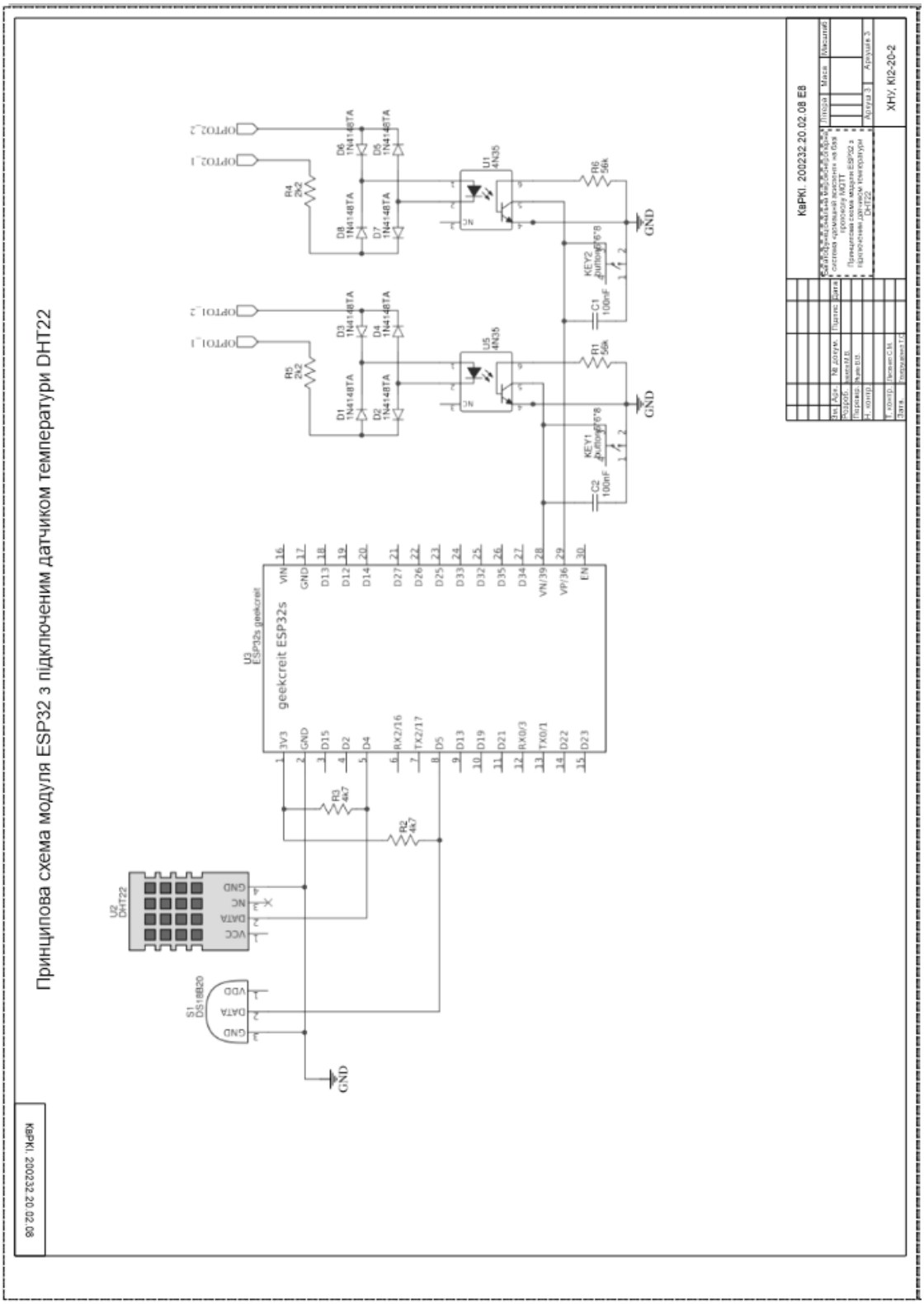
Копія креслення «Функціональна блокова діаграма модуля ESP32»



Додаток В

(обов'язковий)

Копія креслення «Принципова схема модуля ESP32 з підключеним датчиком температури DHT22»



Додаток Г

Лістинг коду

Output

```
function Output(pin, options){
	this.mode options.moduleOpts.mode;
	this.pin = pin;
	if (this.mode != "reset_creds") digitalWrite(this.pin, (this.mode == "reverse" ? !global.defauttPinVa1ue : global. defaultPinValue) / 1) ;
}
Output. prototype.write function(message, pin, options){
	let value = massage == 'true' ? 1 : 0;
	if (options.mode == "reset_creds") {
		setlnterval( ()=>{
			digitatWrite(pin, !value);
			value = !value;
			}, 2500);
	}else {
			 if (options && options.mode == "reverse") value = massage == 'true' ? 0 : 1;
		digitalWrite(pin, value);
	}
	return message;
};
Output. prototype.exit = function(){
	  if (this.mode != "reset_creds") digitalWrite(this.pin, (this.mode == "reverse" ? !gtobat.defauItPinVatue : global.defaultPinValue) / 1);
}
exports.init = function pin, options) {
	return new Output (pin, options);
};
```

Input

```
function Input (pin, options) {
	this.mode
	options.moduleOpts.mode;
	this.pin = pin;
	this.state = false;
	pinMode(pin, this.mode);
}
exports.init function (pin, options) {
	return new Input (pin, options);
};
```

```
Input.prototype. pull = function(cb) {
        if (this.mode == "input") {
                setInterval(() => {
                        if (this.state = digitalRead(this.pin)){
                        cb({data: this.state == 1 ? false: true});
                        }
                        this.state = digitalRead(this.pin);
                }, 250);
        } else {
                setWatch(() => {
                        this.state = !this.state;
                        cb({data: this.state});
                }, this.pin, edge "rising", repeat: true, debounce:100});
            }
};
```

AnalogInput

```
function AnalogInput(pin, options) {
        this.pin = pin;
        this.interval = options.moduleOpts.interval;
        this.reverse = options.moduleOpts.mode == "reverse" ? true : false;
}
exports.init = function (pin, options) {
        return new AnalogInput(pin, options);
};
AnalogInput.prototype.pull function(cb) {
        setInterval(()=>{
                var value = analogRead(this.pin);
                value = (value * 100 | 0);
                value = this.reverse ? 99 value: value;
                cb({data:value});
        }, this.interval);
};
```

Telemetry

```
function Telemetry(options) {
        print(options);
        this.telemetryInfo = options.telemetry;
}
exports.init = function(pin, options) {
        return new Telemetry(options);
};
```

```
Telemetry.prototype.pull = function(cb) {
        cb(this.telemetryInfo);
}


        DHT22

function DHT22 (pin, options) {
        this.pin = pin;
        this.interval = options.moduleOpts.interval;
        this.sensors = {};
        for (sensor in options.moduleOpts.sensors) {
                this.sensors [sensor] = {};
                switch(options.moduleOpts.sensors [sensor]) {
                        case "K":
                                this.sensors [sensor].divider = 1;
                                this.sensors [sensor).offset = 273.15;
                                break;
                        case "C":
                                this.sensors [sensor].divider = 1; break; case "":
                                this.sensors [sensor].divider = 1; break;
                }
        }
}
DHT22.prototype. read function (cb, n) {
        if (!n) n=10;
        var d = "";
        var ht = this;
        digitalWrite(ht.pin, 0);
        pinMode(ht.pin,"output"); // force pin state to output
        // start watching for state change
        this.watch = setWatch(function(t) {
                d+=0|(t.time-t.lastTime>0.00005);
        }, ht.pin, {edge: 'falling', repeat: true});
        // raise pulse after 1ms
        setTimeout(function() {pinMode(ht.pin, 'input_pullup'); pinMode(ht.pin);},1); //
stop looking after 50ms
        setTimeout(function() {
                if(ht.watch) {ht.watch = clearWatch(ht.watch); }
                var cks =
                        parseInt(d.substr(2,8),2)+
                        parseInt(d.substr(10,8), 2)+
                        parseInt(d.substr(18,8), 2)+
                        parseInt(d.substr(26,8), 2);
                if (cks&& ((cks&0xFF)==parseInt(d.substr(34,8),2))) {
```

```
                cb({
                rh : parseInt(d.substr(2,16), 2)*0.1,
                temp = parseInt(d.substr(19,15), 2)*0.2*(0.5-d [18]) +
        (ht.sensors.temp.offset || 0) });
        } else {
        if (n>1) setTimeout(function() {ht.read(cb,--n); },500);
        else cb({temp:"-", rh:"-"});
        }
    }, 50);
};
exports.init = function(pin, options) {
    return new DHT22(pin, options);
};
DHT22.prototype.pull=function(cb) {
    setInterval(()=>{
        this.read(cb);
    }, this.interval);
};
```

Скрипт для встановлення Espruino

```
#!/bin/bash
set -e
exit_if_error() {
    local exit_code=$1
    shift
    [[ $exit_code ]] &&
        ((exit_code != 0)) && {
        printf '\nERROR: %s\n' "$@" >&2
        exit "$exit_code"
    }
}
ping -w3 -c1 google.com &>/dev/null && echo "internet connection established" ||
exit_if_error 1 "Need to establish internet connection"
DEVICE=$1
DEVICE_NAME=$2
DEVICES_LIST=`ls ../device-samples -1 | sed -e 's/\.json$//'`
if [ $# -ne 2 ]; then-
${DEVICES_LIST}"
Fi


[`ls $DEVICE` = $DEVICE ] && echo "device found" || exit_if_error 1 "Need add device
port (example: /dev/ttyUSB0) and device name ${DEVICES_LIST}"
```

```
sudo chown $USER $DEVICE;
if [[ $DEVICES_LIST = $DEVICE_NAME ]]; then-
${PROPOSED_DEVICES_LIST}"
        fi
else-
${DEVICES_LIST}"
fi
if [[ $BOARD_CONFIG = "ESP32.json" ]]; then-
elif [[ $BOARD_CONFIG="SONOFF_ESP8266_4MB.json" ]]; then
        dockerCommands="
                pip install esptool && \
                esptool.py --port $DEVICE --baud 115200 \
                write_flash --flash_freq 40m -flash_mode dout --flash_size 1MB \
                0x0000../fixtures/esp8266-sonoff/boot_v1.6.bin 0x1000
        ../fixtures/esp8266-sonoff/espruino_esp8266_user1.bin\
                0xFC000 ../fixtures/esp8266-sonoff/esp_init_data_default.bin
        0xFE000../fixtures/esp8266-sonoff/blank.bin
        "
elif [[ $BOARD_CONFIG= "ESP8266_4MB.json" ]]; then
fi
docker run \-
read VAR
[`ls $DEVICE` = $DEVICE ] && echo "device found" || exit_if_error 1 "Need add device
port (example: /dev/ttyUSB0) and device name"
docker run \--
echo ""
echo "Reconnect device!"
```

Веб сервер

```
const httpModule= require('http');
const fs = require("Storage");
let runningServer;
function startServer() {
        runningServer = httpModule.createServer(
                requestHandler
        ).listen(80);
}
function requestHandler(req, res) {
        const reqData = url.parse(req.url, true);
        console.log({ reqData });
        const data = reqData.query;

        switch (reqData.pathname) {
```

```
            case '/credentials/': {
                const {wifiName, wifiPass, mqtt, mqttName, mqttPass} = data; const
                wifiCredsErr= (!wifiName || !wifiPass) || false;
            const mqttAuthWOPass = (mqtt && mqttPass && !mqttName) || false;
            if (wifiCredsErr || mqttAuthWOPass) {
                res.writeHead (500);
                res.end('Incomplete form!');
                return;
            }
            fs.write('credentials', data);
            res.writeHead (200);
            res.end('');
            setTimeout(() => E. reboot(), 1000);
            break;

        }
        case '/ping/': {
            try {
                let pin = eval(reqData.query.pin);
                digitalWrite(pin, reqData.query.value);
            } catch(e) {
                res.writeHead (500);
                res.end('');
                return;
            }
            res.writeHead (200);
            res.end('');
            break;
        }
        default: {
            res.writeHead (200, {'Content-Type': 'text/html'});
            var htmlArr = fs.read('template').split('<body>');
            res.write(htmlArr[0] + '<body>'+'<div id="controller"
            hidden>'+fs.read('controller')+'</div>' + htmlArr[1]);
            res.end('<h4> mac = require('Wifi').getIP().mac.replace(/:/g,'-') + '</h4>');
            break;
        }
}
}
function stopServer() {
runningServer.close();
}
```

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016382103

Дата перевірки:
22.06.2024 09:34:34 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
22.06.2024 09:35:37 EEST

ID користувача:
100005591

Назва документа: Івахов_Багатофункціональна мікроконтролерна система «домашній асистент» на базі прот...

Кількість сторінок: 78 Кількість слів: 11530 Кількість символів: 86107 Розмір файлу: 6.12 MB ID файлу: 1016191743

17.2%
Схожість

Найбільша схожість: 9.49% з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/39331/1/OliinykKV_magistr.pdf)

17% Джерела з Інтернету | 686 ..... Сторінка 80

7.07% Джерела з Бібліотеки | 221 ..... Сторінка 94

0.9% Цитат

Цитати | 5 ..... Сторінка 95

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

	Документ		Сумарний збіг по Базі Даних	
ID: 132160 Назва: БКР Багатофункціональна мікроконтролерна система «домашній асистент» на базі протоколу MQTT Додано в БД: 2024-06-22 Автора: М. В. Івахов Керівники: В. В. Яцків Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	75757	612	655 (1%)	7 (1%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Івахов Максим Володимирович

Тема: Багатофункціональна мікроконтролерна система «домашній асистент» на базі протоколу MQTT

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___68___

1. Короткий зміст роботи та прийнятих рішень: Метою дипломної роботи є розробка програмного забезпечення для домашнього асистенту на базі протоколу звязку – MQTT.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі було проведено аналіз доступних технологій, що використовуються в теперішніх системах і архітектурах розумного дому. На основі цього аналізу був складений перелік найпопулярніших і найпоширеніших протоколів передачі даних та методів, які лежать в основі систем управління та менеджменту розумних будинків та виконано постановку задачі дослідження. В другому розділі було проведено аналіз сучасних платформ та систем управління для розумного будинку з метою визначення вимог до майбутньої архітектури програмного забезпечення. На основі цього аналізу було сформульовано перелік функціональних та нефункціональних вимог, враховуючи сучасні потреби користувачів і їхні повсякденні завдання. було проведено аналіз можливих інструментів для програмування мікроконтролерів ESP, і було обрано Espruino для реалізації проекту. У третьому розділі дослідження була розроблена архітектура програмного забезпечення для пристроїв розумного будинку. В процесі розробки було вирішено наступні завдання: розроблено бібліотеку модулів для взаємодії з популярними датчиками та пристроями; впроваджено механізм для

створення нових модулів користувачами; створено інструмент командного рядка (CLI) для встановлення та оновлення програмного забезпечення на пристроях; реалізовано підтримку спеціалізованої IoT конвенції; розроблено веб-сервер та інтерфейс для підключення пристроїв до локальної мережі; створено бібліотеку для конфігураційних файлів для тестування модулів у різних комбінаціях; інтегровано з платформою HomeBridge; налаштовано голосове керування за допомогою асистентів Siri та Google.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатня увага тестуванню програмного забезпечення у реальних користувацьких сценаріях.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

к.т.н., доцент каф. АКІТтаР Корецька Л.О.

"21" 06 2024 р. _____(підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Івахова Максима Володимировича
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-20-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22 квітня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Багатофункціональна мікроконтролерна система «домашній асистент» на базі протоколу MQTT

Автор: Івахов Максим Володимирович

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Яцків Василь Васильович, д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з 100 - 300 джерелами на один фрагмент речення;
4) в якості запозичень в окремих місцях системою зафіксовано послідовності чотирьохрозряних двійкових кодів, які є вхідними даними до великої кількості задач і не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;
5) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 17,2% і адресується до 686 першоджерел, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В. М. Яцків

Гарант ОП _____ С.М. Лисенко

Завідувач кафедри КІС _____ Т. О. Говорущенко