КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

КРБ.СІ - 12.00.00.000 ПЗ

Група СІ-21-1

Андрій Нагірняк

2025

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій
Кафедра інформаційно-телекомунікаційних технологій та систем

Нагірняк Андрій Дмитрович
(прізвище, ім'я, по батькові)

УДК 004.67

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Розроблення автоматизованої системи обробки інформації на базі платформи ESP-32
(назва роботи)

Системна інженерія – Інтернет речей
(назва освітньої програми)

151 Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

Здобувач освітнього ступеня _____ А.Д. Нагірняк
(підпис, ініціали та прізвище здобувача)

Науковий керівник _____ Паньків Х. В. к.т.н., доц. каф. ІТТС
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

д.т.н., проф. _____ Заміховський Л. М.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2025

Івано-Франківський національний технічний університет нафти і газу

(повне найменування закладу вищої освіти)

Інститут інформаційних технологій

Кафедра інформаційно-телекомунікаційних технологій та систем

Освітній рівень бакалавр

Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології

(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТТС
д.т.н., проф Заміховський Л. М.
«05» травня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Нагірняку Андрію Дмитровичу

(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення автоматизованої системи обробки інформації на базі платформи ESP-32

керівник роботи Паньків Х.В. к.т.н., доцент кафедри ІТТС

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «05» травня 2025 року № 281/7

2. Строк подання студентом роботи 20 червня 2025 року

3. Вихідні дані до роботи технічні характеристики мікроконтролера ESP32-WROOM-32E, документація на платформу Home Assistant та ESPHome.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Вступ

1. Аналіз стану сучасних технологій та науково-технічних рішень у галузі систем обробки інформації

2. Огляд технічних рішень та компонентів для реалізації системи обробки інформації

3. Практична реалізація проєкту

Висновки

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

1. Міжблокова схема проєкту

2. Комунікаційне середовище Home Assistant

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	к.т.н., доц. каф. ІТТС Паньків Х.В.		
2	к.т.н., доц. каф. ІТТС Паньків Х.В.		
3	к.т.н., доц. каф. ІТТС Паньків Х.В.		

7. Дата видачі завдання 08.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	Вибір теми, узгодження з керівником	08.04.2025 – 09.04.2025	виконано
2	Збір матеріалів, огляд літератури	10.04.2025 – 14.04.2025	виконано
3	Огляд наукових джерел, написання 1 розділу	15.04.2025 – 29.04.2025	виконано
4	Проєктування архітектури системи, розробка структурної та функціональної схеми	30.04.2025 – 03.05.2025	виконано
5	Реалізація прототипу системи на ESP32	04.05.2025 – 06.05.2025	виконано
6	Розробка інтеграції з Home Assistant та Google TTS	07.05.2025 – 09.05.2025	виконано
7	Написання 2 розділу	10.05.2025 – 12.05.2025	виконано
8	Написання 3 розділу	13.05.2025 – 14.05.2025	виконано
9	Формування висновків, оформлення списку літератури, додатків.	19.05.2025 – 24.05.2025	виконано
10	Попереднє подання БКР на перевірку керівнику	25.05.2025 – 27.05.2025	виконано
11	Здача пояснювальної записки	20.06.2025	виконано

Студент __ _____ Нагірняк А.Д.
(підпис) (прізвище та ініціали)

Керівник роботи __ _____ Паньків Х.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Нагірняк А. Д. Розроблення автоматизованої системи обробки інформації на базі платформи ESP-32

Бакалаврська робота на здобуття освітнього ступеня бакалавр за освітньо-професійною програмною «Системна інженерія – Інтернет речей», спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології». Івано-Франківський національний університет нафти і газу. Івано-Франківськ, 2025.

У дослідженні проаналізовано існуючі підходи до побудови автоматизованих систем обробки інформації в середовищі Інтернету речей, зокрема з використанням платформ ESP32 та Home Assistant.

У роботі розроблено метод і реалізовано програмно-апаратну систему, що забезпечує виявлення подій за допомогою сенсорів руху, їх обробку на локальному сервері Home Assistant і озвучення подій засобами Google Text-to-Speech. Запропоноване рішення функціонує без залежності від хмарних сервісів, що підвищує його автономність, надійність та конфіденційність.

У межах проєкту створено повноцінну систему автоматизації, яку можна застосовувати для побутового або навчального використання, з можливістю масштабування та інтеграції додаткових сенсорів.

Ключові слова: ESP32, HOME ASSISTANT, АВТОМАТИЗАЦІЯ, СЕНСОР РУХУ, GOOGLE TEXT-TO-SPEECH, IOT, ОБРОБКА ІНФОРМАЦІЇ.

ANNOTATION

Nahirnyak A. D. Development of an automated information processing system based on the ESP-32 platform

Bachelor's thesis for the degree of Bachelor of Science in the educational and professional program “System Engineering – Internet of Things,” specialty 151 “Automation and Computer-Integrated Technologies.” Ivano-Frankivsk National University of Oil and Gas. Ivano-Frankivsk, 2025.

The study analyzes existing approaches to building automated information processing systems in the Internet of Things environment, in particular using the ESP32 and Home Assistant platforms.

The work develops a method and implements a software and hardware system that detects events using motion sensors, processes them on a local Home Assistant server, and announces events using Google Text-to-Speech. The proposed solution functions independently of cloud services, which increases its autonomy, reliability, and confidentiality.

Within the framework of the project, a full-fledged automation system has been created that can be used for domestic or educational purposes, with the possibility of scaling and integrating additional sensors.

Keywords: ESP32, HOME ASSISTANT, AUTOMATION, MOTION SENSOR, GOOGLE TEXT-TO-SPEECH, IOT, INFORMATION PROCESSING

РЕФЕРАТ

Розрахунково-пояснювальна записка: 73 сторінок, 30 рисунків, 4 таблиці, 29 посилань.

Метою даної роботи є розробка та реалізація автоматизованої системи обробки інформації на базі мікроконтролера ESP32 з інтеграцією до відкритої платформи Home Assistant та сервісу Google Text-to-Speech. Розроблена система призначена для виявлення подій, обробки сигналу у локальному середовищі та подальшого голосового сповіщення користувача.

Об’єкт дослідження – процес автоматизованої обробки інформації; предмет – апаратно-програмні засоби IoT-систем.

У першому розділі кваліфікаційної роботи проаналізовано основні підходи до побудови систем обробки інформації систем. Проведено порівняння відкритих та закритих платформ.

У другому розділі досліджено апаратну та програмну архітектуру автоматизованої системи, описано структуру системи на базі Home Assistant, її взаємодію з мікроконтролером за допомогою ESPHome.

У третьому розділі проведено практичне тестування роботи системи в умовах, наближених до реального середовища. Перевірено працездатність системи, стабільність автоматизацій, візуалізацію станів сенсорів та коректність генерації голосових повідомлень.

Розроблена система показала доцільність застосування відкритої платформи на базі ESP32 для реалізації задач автоматизації обробки інформації в умовах локального середовища. Вона є недорогим, адаптивним і масштабованим рішенням, придатним для використання у розумних будинках, навчальних проєктах або експериментальних IoT-сценаріях.

ESP32, HOME ASSISTANT, АВТОМАТИЗАЦІЯ, СЕНСОР РУХУ, GOOGLE TEXT-TO-SPEECH, IOT, ОБРОБКА ІНФОРМАЦІЇ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

IoT – (від. англ. Internet of Things, Інтернет речей) концепція мережі, що складається з безлічі фізичних пристроїв, які пов’язані між собою для постійного обміну та обробки інформації.

ESP32 – мікроконтролер виробництва компанії Espressif Systems, що має інтегровані Wi-Fi і Bluetooth модулі, низьке енергоспоживання та ціну.

MQTT – простий протокол передачі повідомлень, призначений для роботи у мережах з обмеженими ресурсами.

REST API – передача репрезентативного стану.

Google Text-to-Speech (TTS) – додаток для озвучення тексту.

Wi-Fi – технологія бездротової передачі даних, яка відповідає стандарту IEEE 802.11 і забезпечує з'єднання між пристроями через радіоканали.

Bluetooth – технологія бездротового зв’язку, передвизначена для обміну інформації на короткій відстані.

Tensilica Xtensa LX6 – ядро плати ESP32.

КБ – кілобайт, одиниця вимірювання обсягу даних.

UART – універсальний асинхронний приймач-передавач, що забезпечує перетворення між паралельною та послідовною передачею даних.

SPI – послідовний інтерфейс, розроблений компанією Motorola для зв'язку між мікроконтролером та периферійними пристроями.I2C – (англ. Inter-Integrated Circuit, взаємоінтегрована схема) послідовна шина даних для зв'язку інтегральних схем.

I2S – (англ. Inter-Integrated Circuit Sound) протокол передачі двоканального цифрового звуку.

PWM – метод регулювання сигналу за допомогою зміни ширини імпульсів, застосовується, зокрема, для управління потужністю.

ADC – (англ. Analog-to-digital converter, аналогово-цифровий перетворювач) пристрій, що перетворює вхідний аналоговий сигнал в дискретний код.

DAC – (англ. Digital-to-Analog Convertor, цифро-аналоговий перетворювач) пристрій, що перетворює цифровий (як правило двійкового) сигнал на аналоговий.

МК – мікроконтролер, комп’ютер виконаний у вигляді маленької мікросхеми.

Стек – різновид лінійного списку, структура даних.

ROM – тип пам’яті, вміст якої не змінюється і зберігається навіть після вимкнення живлення.

SRAM – (англ. static random access memory) статична оперативна пам'ять з довільним доступом.

RTC – (англ. Real-time clock) комп’ютерний годинник, який відстежує поточний час навіть без живлення.

802.11b/g/ – стандарт бездротового зв'язку.

AFH – технологія, яка змінює робочі частоти для уникнення перешкод у бездротовому зв’язку.CVSD – (англ. Continuous Variable Slope Delta Modulation) метод кодування звукових сигналів.

SBC – (англ. Subband Coding) метод кодування, який розбиває аудіосигнал на декілька частотних піддіапазонів і стискає кожен з них окремо.

GPIO – інтерфейс для зв'язку між компонентами комп'ютерної системи.

SD-карта – зовнішній тип пам’яті.

SPI – (англ. Serial Peripheral Interface) інтерфейс серійної передачі даних.

SDIO – (англ. Secure Digital Input/Output) інтерфейс для підключення пристроїв.

LED PWM – метод зміни яскравості світлодіодів шляхом регулювання тривалості імпульсів.

IR – технологія бездротового зв'язку, яка використовує інфрачервоні промені для передачі сигналів на короткі відстані.

TWAI – інтерфейс, сумісний із CAN-шиною, часто застосовується в автомобільних системах та реалізований в ESP32.

ВСТУП

У сучасному світі активно набирає обертів концепція Інтернету речей, яка передбачає створення розумних систем, здатних до автоматичного збору, обробки та передачі інформації. Однією з ключових частин таких систем є мікроконтролери, серед яких особливу популярність здобула платформа ESP32 завдяки своїй функціональності, низькому енергоспоживанню та підтримці бездротового зв’язку.

Водночас широкого поширення набувають платформи для розумного дому, зокрема Home Assistant, яка забезпечує гнучке середовище для інтеграції різних пристроїв, створення сценаріїв автоматизації та взаємодії з користувачем. Інтеграція з хмарними сервісами, такими як Google Text-to-Speech (TTS), дає змогу розширити функціональність систем автоматизації, зокрема шляхом голосового інформування про події та стани пристроїв.

Застосування Home Assistant як платформи управління дає змогу централізовано контролювати побутові пристрої, а інтеграція з Google Text-to-Speech дозволяє реалізувати ефективний канал зворотного зв’язку у вигляді голосових повідомлень. Такий підхід суттєво розширює функціональність систем автоматизації, підвищує зручність користування та дозволяє використовувати систему в побуті, освіті, охороні, а також у спеціалізованих галузях .

Створення і вивчення таких систем широко розповсюджено серед українських та закордонних науковців. Під час аналізу їх робіт було виділено, що вітчизняні дослідники зосереджують увагу на побудові архітектури розумних систем, використанні ESP-платформ у навчальному процесі та малих проєктах автоматизації, а закордонні — детально досліджують безпеку IoT, взаємодію з хмарними сервісами, використання MQTT, REST API, TTS у побутових та промислових умовах.

На сучасному етапі розвитку автоматизації особливо актуальними є проблеми надійної інтеграції мікроконтролерів з системами управління,

розширення каналів зворотного зв’язку з користувачем та забезпечення масштабованості. Розроблення системи, яка здатна реагувати на події, здійснювати обробку інформації та озвучувати повідомлення в реальному часі, є актуальною задачею в умовах розвитку Індустрії 4,0.

Метою даної бакалаврської кваліфікаційної роботи є розроблення автоматизованої системи обробки інформації на базі ESP32 з інтеграцією в платформу Home Assistant та використанням сервісу Google TTS для озвучення повідомлень.

Для досягнення мети потрібно вирішити такі завдання:

1. Провести аналіз технічних можливостей ESP32.
2. Розробити архітектуру системи та програмне забезпечення для взаємодії між МК, датчиками та виконавчими механізмами.
3. Реалізувати автоматизацію приладу.
4. Здійснити тестування системи та оцінити її функціональність.

Об’єктом дослідження є процес автоматизованого збору, обробки та озвучення інформації в розумному середовищі. Предметом дослідження - програмно-апаратні рішення на базі ESP32 з використанням Home Assistant та інтегрованих додатків цієї платформи.

Окремі результати та компоненти були апробовані у процесі проєктування, тестування прототипу та їх можна застосувати для демонстрацій у навчальному процесі або як основу для подальших досліджень, також можуть бути застосовані у системах розумного дому, побутової автоматизації, освітніх проєктах або адаптовані для промислових рішень малого масштабу.

1 АНАЛІЗ СТАНУ СУЧАСНИХ ТЕХНОЛОГІЙ ТЕХНІЧНИХ РІШЕНЬ У ГАЛУЗІ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ

1.1 Значення методів обробки інформації

В сучасному світі обробка інформації, обмін нею та її захист є пріоритетними напрямками розвитку інформаційного суспільства. Інформація (від лат. informatio — роз'яснення) — відомості, які передають усним, писемним та іншими шляхами за допомогою умовних сигналів і технічних засобів [1]. Обробка інформації охоплює широкий спектр дій — від первинного збору даних із фізичних сенсорів до складних алгоритмів аналізу, прийняття рішень і керування виконавчими механізмами. У добу цифрової трансформації потреба в автоматизованих інформаційних системах стрімко зростає, що зумовлює актуальність досліджень у цій сфері.

Бурхливий розвиток інформаційних технологій забезпечує розвиток методів та засобів для збору, зберігання, обробки та поширення інформації. Особливе місце в сучасних інформаційних технологіях набуває концепція Інтернет речей, основна ідея якої - можливість приєднати будь-який об'єкт до інформаційної мережі, обробити інформацію та обмінюватися нею [2].

Велика кількість дослідників у своїх роботах описують ті чи інші аспекти отримання, передання, обробки різних типів інформації. Наприклад, американський математик Клод Шенон, засновник теорії інформації ввів поняття ентропії як міри невизначеності та заклав основи цифрової комунікації. У своїй праці він визначив ентропію як міру невизначеності або непередбачуваності джерела повідомлень. Це поняття дозволяє кількісно оцінити обсяг інформації, що міститься в повідомленні, та визначити ефективність його кодування [3].

Шеннон також описав модель комунікаційної системи, що складається з п'яти основних компонентів: джерела інформації, передавача, каналу зв'язку, приймача та одержувача. Ця модель стала фундаментом для подальшого розвитку теорії інформації та телекомунікацій.[3].

Також важливий внесок для подальшого розвитку теорії інформації та телекомунікації зробили Гаррі Найквіс, встановивши залежність максимальної швидкості передачі інформації від ширини смуги пропускання каналу та кількості рівнів сигналу [4], Ральф Гартл, робота якого заклала основи для математичного визначення інформації [5].

Інші дослідники, які вивчали системи обробки інформації на базі платформи ESP32 у своїх роботах наголошували на ефективних способах обміну даними як от peer-to-peer [6], або ж створення систем точного вимірювання часу [7]. Науковці, які описували принцип TTS в галузі синтезу мовлення демонструють високу ефективність відкритих програмних інструментів, які можуть бути адаптовані для використання у вбудованих системах на базі ESP32. Наприклад, у роботі японських дослідників представлено інструментарій ESPnet2-TTS, який підтримує широкий спектр сучасних TTS-моделей (Tacotron 2, FastSpeech, Transformer TTS) і дозволяє інтегрувати їх у реальні прикладні проєкти [8]. Інше дослідження [9] — ESPnet-TTS, — об’єднує моделі синтезу мови в єдину платформу, що спрощує інтеграцію голосових інтерфейсів у системах автоматизації та підвищує відтворюваність результатів. Для пристроїв з обмеженими обчислювальними ресурсами, таких як ESP32, перспективною є модель LightSpeech, розроблена із застосуванням нейромережевого архітектурного пошуку (NAS), яка характеризується високою швидкістю інференції при низькому споживанні пам’яті.

Аналіз доступної науково-технічної літератури свідчить, що хоча тематика створення систем на основі ESP32 і є доволі розвиненою, інтеграція Google TTS застосовується переважно фрагментарно — для поодиноких команд, без повної адаптації до розширених сценаріїв автоматизації.

Крім того, дослідники здебільшого зосереджуються на окремих елементах — або тільки на апаратній частині, або лише на конфігурації, без поєднання всіх етапів в цілісну систему з розпізнаванням подій.

1.2 Системи автоматизованої обробки подій на ринку

Побудова автоматизованої системи обробки інформації потребує урахування наявних технічних та програмних рішень, які вже використовуються в суміжних галузях і можуть бути частково або повністю адаптовані для реалізації цільових функцій. До таких рішень належать як комерційні системи з фіксованим функціоналом, так і відкриті платформи, орієнтовані на розширюване середовище та локальну обробку даних.

Системи автоматизації широко використовують датчики руху і присутності для подій – наприклад, вмикання світла, сигналізацію чи оголошення. Такі системи можуть бути як комерційними «під ключ», так і побудованими на відкритому програмному забезпеченні. Вони відрізняються рівнем автономності (хмарні чи локальні), відкритістю (відкритий вихідний код чи зашиті логіки), вартістю, наявністю озвучення та сумісністю з платформами Home Assistant, Alexa, Google тощо. Нижче наведено огляд основних рішень.

Home Assistant – популярна відкрита система локальної автоматизації. Вона встановлюється на власному сервері/Raspberry Pi та не залежить від хмари [10]. Підтримує тисячі інтеграцій з сенсорами руху та присутності. Має вбудований механізм TTS для голосових сповіщень. Сумісна з Alexa та Google Assistant. Вартість ПО безкоштовна, логіка повністю відкрита, що дає високу гнучкість.

OpenHAB – відкритий багатоплатформовий «домашній хаб», що працює локально [11]. Підтримує понад 400 протоколів і тисячі пристроїв. Може керуватися правилами з подіями за часом чи сигналом. Як і HA, не потребує хмари. Є інтеграції для Alexa, Google Assistant, HomeKit тощо. Підтримує озвучення. Система відкрита, що забезпечує змінюваність логіки, але вимагає настройки [11].

Node-RED – візуальний конструктор логіки на основі потоків даних. [12]. Node-RED є відкритим і дуже поширеним в IoT [12]. Через безліч конекторів він може обробляти події від датчиків руху/присутності, а потім запускати сповіщення чи озвучення. Підтримка TTS забезпечується додатковими нодами. Сумісний з будь-якими сенсорами та сервісами, здебільшого потребує джерела подій.

DIY-апаратна система (Arduino/Pi) – прості рішення на Arduino/ESP32 з PIR-датчиками, ультразвуковими або радарними сенсорами. Наприклад, проєкти Arduino+OpenHAB з PIR-модулем показують, як бюджетно виявляти вхід у кімнату [13]. Такі датчики живляться від Arduino і передають сигнал на центральний хаб через MQTT або GPIO. В якості кращої альтернативи з'явились дешеві модулі mmWave – вони «бачать» навіть статичну людину.

Frigate NVR (відкрита система відеоспостереження) – який виконує детектування людей/об'єктів на відеопотоках локально за допомогою нейромереж [14]. Усі відеодані не передаються в хмару. Frigate інтегрується з Home Assistant, OpenHAB, Node-RED (через MQTT) і забезпечує датчики «присутність/об'єкт» в інтерфейсі автоматизації [14]. Це дозволяє будувати системи безпеки чи присутності на базі камер. Frigate можна встановити на Raspberry Pi/Jetson/NAS з підтримкою Coral/RTX. Система повністю відкрита і безкоштовна, дає великі можливості налаштування зон сповіщення. На практиці Frigate відіграє роль «очей» дому, автоматично видаючи події «людина на порозі» і передає їх іншим системам.

Amazon Echo (4-го покоління) – комерційна платформа розумного дому від Amazon із власним голосовим помічником Alexa. Echo 4th Gen оснащений ультразвуковим «сонаром» для виявлення присутності – пристрій може визначати людину у кімнаті через звук [15]. Базова логіка Alexa закрита і «прив'язана» до екосистеми Amazon. Однак існують навички/інтеграції для передачі подій в інші системи (Home Assistant, IFTTT). Пристрій коштує приблизно $100–150. Він має вбудоване озвучення і може проговорювати повідомлення як TTS. Сумісний лише з екосистемою Alexa/SmartThings (через хмару). Автономність низька потрібне інтернет-з'єднання, хоча базові алгоритми можуть бути локальними.

Google Nest – платформа Google з голосовим помічником. Розумні колонки та дисплеї (Nest Hub Max) можуть використовувати технологію Soli (радар) для обробки жестів та виявлення присутності людини, а також вмикати рутину Ambient (LED під будильник тощо). Вони повністю прив'язані до хмари Google. Пристрої підтримують озвучення через Google Home. Сумісність – з іншими пристроями Google Home, розумними вимикачами тощо. Ціна та логіка – як у Alexa.

Apple HomeKit – екосистема Apple з Siri. Пристрої HomePod і Apple TV можуть слугувати хабами HomeKit. Присутність визначається, наприклад, через геозони на iPhone чи присутність у мережі. Siri може виводити голосові повідомлення на HomePod. Система зашифрована/закрита, сумісна лише з HomeKit-сенсорами і аксесуарами.

Xiaomi Mi Home – китайська комерційна екосистема. Має широкий спектр датчиків: дешеві PIR-датчики, Zigbee-датчики руху, температури, датчики відкриття дверей тощо. Наприклад, Aqara FP1 – «космічний» датчик присутності на основі mmWave-радару [16]. Він «бачить» навіть статичні об'єкти і передається через Zigbee. Вартість близько $50 [16]. Такі пристрої інтегруються в Mi Home та можуть працювати з Home Assistant через Zigbee2MQTT або ZHA [16]. Відкритість – низька.

Philips Hue Motion / Eve Motion – Zigbee/Bluetooth-датчики руху для розумного освітлення. Працюють через власний хаб або HomeKit. Вартість одного датчика ≈ $40, живляться батареєю. Забезпечують прості події «детектовано рух/ніч». OpenHAB/HA з ними працюють через Hue Hub. Голосових функцій немає.

Sonoff SNZB-06P – доступний Zigbee-датчик присутності (5.8 ГГц радар), виробник Itead/Sonoff [17]. Виявляє рух, присутність людини і має датчик світла для уникнення помилкових спрацьовувань. Працює від постійного живлення, тому виступає як Zigbee-роутер, зміцнює мережу [17]. Ціна ≈ $15. Сенсор закритий потрібен Zigbee-хаб – Sonoff NSPanel/Bridge або сторонній Zigbee USB. Його можна інтегрувати в Home Assistant через ZHA, Zigbee2MQTT. TTS-самостійного немає, але події може обробляти Alexa чи HA.

Meross MS600– новий 24 ГГц mmWave-радарний сенсор присутності. Працює по стандарту Matter і сумісний з Apple HomeKit, Alexa, Google Home та локальними системами через Mattershop [18]. Має живлення 5 В через кабель. Виявляє невеликі рухи до 12 м дальності з високою точністю без «фантомних» спрацьовувань [18]. Інтегрується безпосередньо у домашні системи: у Matter-мережі створюються сутності «Presence» і «Light» для автоматизації [18]. Завдяки Matter відкрита для налаштування логіки в будь-якій сумісній системі (наприклад, Home Assistant Yellow або коробки Matter Bridge). На відміну від камер, не порушує приватність. Вартість близько $35.

Узагальнене порівняння проаналізованих систем за критеріями наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика систем автоматизованої обробки подій

Система	Автономність	Відкритість	Приблизна вартість	TTS (голосові сповіщення)	Сумісність
Home Assistant	висока (локальний сервер)	повністю відкрита	безкоштовне ПЗ	так	Alexa, Google, HomeKit тощо
OpenHAB	висока (локально)	повністю відкрита	безкоштовне ПЗ	так	Alexa, Google, HomeKit тощо
Node-RED	висока (локально)	відкрита	безкоштовно	так	будь-які
Frigate NVR	висока (локально)	повністю відкрита	безкоштовне ПЗ	ні	Home Assistant, OpenHAB, MQTT
Amazon Echo	низька (хмарна залежність)	закрита	≈$100–150	так	Alexa-екосистема
Google Nest	низька (хмарна)	закрита	≈$100–200	так	Google Home-екосистема
Apple HomeKit	середня (локація+хмара)	закрита	~$100–200	так	HomeKit-сумісні сенсори
Xiaomi Mi Home	середня (локальний хаб)	переважно закрита	дешеві (~$5–15 за датчик, FP1 $50)	ні (	Mi Home, Home Assistant (Zigbee)
Philips Hue	середня (локальний хаб)	закрита	~$40 за датчик	ні	Hue Bridge, HomeKit, Alexa, Google
Sonoff SNZB-06P	висока (локально)	закрита	≈$15	ні	Zigbee (Echo Plus, ZBBridge), HA
Meross MS600	висока (локально)	відкритий	≈$35	ні	Matter (HomeKit, Alexa, Google)
DIY Arduino/Pi + датчики	висока (локально)	повністю відкрита	дуже низька (кіт з модулями ≪$50)	частково	будь-які через MQTT/REST

З результату досліджень можна провести наступний висновок попри різноманіття доступних систем автоматизованого виявлення подій, жодна з них не забезпечує одночасно повної відкритості, локальної обробки, підтримки динамічного звукового повідомлення та гнучкої персоналізації без залучення сторонніх сервісів. Комерційні продукти мають високий рівень апаратної інтеграції, зручний інтерфейс користувача та потужну підтримку бренду, проте значною мірою обмежені у функціональності через закритість екосистем або залежність від хмарної інфраструктури.

Відкриті рішення, мають ряд переваг серед яких: повна відкритість програмного коду, локальна обробка даних, підтримка стандартів і протоколів, можливість використання безкоштовного програмного забезпечення, масштабованість. Особливої уваги заслуговують відкриті апаратно-програмні платформи на базі ESP32, які дають змогу реалізувати повністю автономну, масштабовану та адаптовану до потреб користувача систему. У поєднанні з Home Assistant такі рішення дозволяють інтегрувати різноманітні сенсори, реалізовувати локальні сценарії автоматизації, використовувати текстово-голосові повідомлення та забезпечити високий ступінь захисту персональних даних без передачі інформації на зовнішні сервери.

Таким чином, порівняльний аналіз доводить доцільність використання відкритих IoT-рішень у навчальних, побутових та експериментальних системах, де критичними є незалежність, прозорість логіки, адаптивність і можливість швидкої модернізації.

2 ОГЛЯД ТЕХНІЧНИХ РІШЕНЬ ТА КОМПОНЕНТІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ ОБРОБКИ ІНФОРМАЦІЇ

2.1 Загальна характеристика вибраної платформи

З метою забезпечення ефективної взаємодії користувача з фізичним середовищем за допомогою вбудованих засобів керування у межах даної роботи здійснюється розроблення автоматизованої системи обробки інформації на базі мікроконтролера ESP32-WROOM-32E. Актуальність створення такої системи зумовлена необхідністю розробки універсального та енергоефективного рішення, здатного здійснювати збір, обробку, передавання інформації та відповідне реагування на події в реальному часі. У системі передбачено застосування платформи Home Assistant для побудови локальної інфраструктури автоматизації, а також використання хмарного сервісу Google Text-to-Speech для генерації голосових повідомлень.

Функціонування системи передбачає передачу отриманої інформації з датчиків на ESP32. Обробка подій та сценаріїв виконується на рівні локального сервера Home Assistant, що дозволяє зменшити залежність від хмарних інфраструктур, скоротити час реакції та підвищити надійність.

Таким чином, система реалізує повний цикл обробки інформації: від первинного зчитування даних до їх візуалізації, збереження, озвучування та ініціації дій. Розробка ґрунтується на використанні апаратних та програмних рішень з відкритим кодом, що сприяє гнучкості, доступності та можливості подальшого вдосконалення проєкту.

Центральне місце в архітектурі таких систем займають вбудовані апаратно-програмні платформи, які забезпечують збирання, обробку, передачу та подальше використання інформації. Однією з найбільш популярних у цій сфері є платформа ESP32, розроблена компанією Espressif Systems.

Перейдемо до детальнішого аналізу цієї платформи. ESP32 має 32-бітний процесор з подвійним ядром Tensilica Xtensa LX6 тактова частота якого може сягати до 240 МГц. Модуль пам'яті місткістю до 520 КБ, можливість підключення додаткової зовнішньої пам'яті. Плата наділена різними інтерфейсами введення, виведення: UART, SPI, I2C, I2S для звукових додатків; PWM, ADC, DAC — для керування аналоговими й цифровими пристроями. Опираючись на інформацію з інтернет-джерела виробника, МК здатний надійно функціонувати в промислових умовах з діапазоном робочих температур від –40°C до +125°C. Завдяки розширеним схемам калібрування, може динамічно усувати дефекти зовнішніх схем і адаптуватися до змін.

Він має високу можливість бути інтегрованим із вбудованими перемикачами антени, радіочастотним балансиром, підсилювачем потужності, малошумним приймальним підсилювачем, фільтрами та модулями керування живленням.

Чіп розроблений спеціально для мобільних пристроїв, переносної електроніки та додатків IoT, забезпечує наднизьке енергоспоживання за допомогою комбінації кількох типів власного програмного забезпечення. ESP32 може працювати як повноцінна автономна система або як підлеглий пристрій для головного МК, зменшуючи накладні витрати стека зв'язку на головному прикладному процесорі [19].

Найпопулярнішою серед усіх плат є ESP32 DevKitC V3, що зображена на рисунку 2.1. Основна технічна характеристика представлена у таблиці 2.1. Платформа для розробки, має 38 пінів з яких 31 служать для підключення давачів, приладів тощо.

Рисунок 2.1 – Плата ESP32 DevKitC V3

Таблиця 2.1 – Технічні характеристики ESP32 DevKitC V3

Процесор і вбудована пам'ять	ESP32-D0WD-V3 або ESP32-D0WDR2-V3,вбудований, двоядерний 32-розрядний мікропроцесор Xtensa LX6 до 240 МГц; 448 KB ROM,520 KB SRAM,16 KB SRAM в RTC
Протоколи зв'язку	Wi-Fi - 802.11b/g/, швидкість передачі даних до 150 Мбіт/с, Діапазон центральної частоти робочого каналу: 2412 ~ 2484 МГц ; Bluetooth V4.2 BR/EDR та Bluetooth LE специфікація AFH CVSD та SBC
Периферія	26 GPIO, 5 обв'язувальних GPIO, SD – карта, UART, SPI, SDIO, I2C, LED PWM, PWM, I2S, IR, лічильник імпульсів, ємнісний сенсорний датчик, ADC, DAC, TWAI сумісний з ISO 11898-1
Інтегровані компоненти на модулі	Кристалічний генератор 40 МГц, 4/8/16 МБ SPI флешпам'яті,ESP32-D0WDR2-V3 також забезпечує 2 МБ PSRAM, вбудована антена на друкованій платі
Умови експлуатації	Джерело живлення: 3,0~3,6V,робоча температура 40 ~ 85 °C.

Блок діаграма зображена на рисунку 2.2.

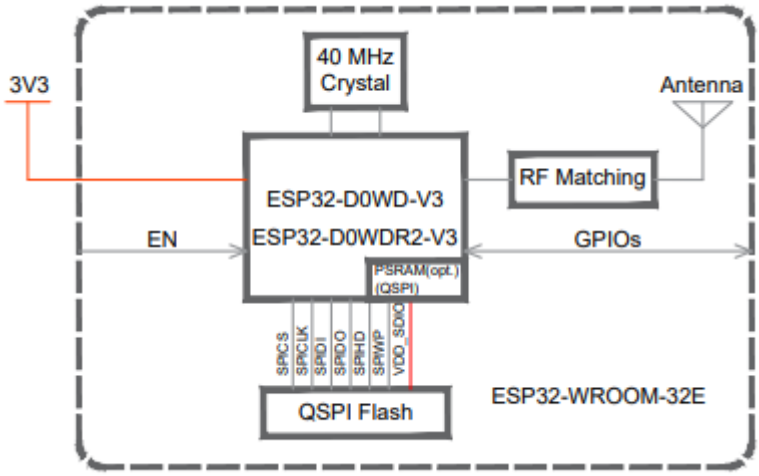

Рисунок 2.2 – Блок діаграма ESP32-WROOM-32E

На схемі, що зображена на рисунку 2.3 представлено розташування контактів на модулі.

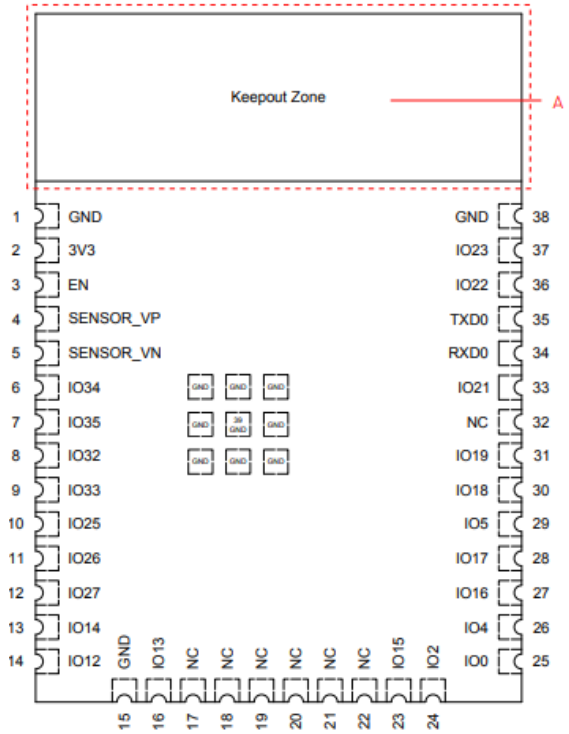

Рисунок 2.3 – Позначення контактів

Нижче в таблиці 2.2 представлений опис кожного контакту ESP32

Таблиця 2.2 – Опис контактів [20]

Назва	**№**	**Тип**	**Функція**
GND	1	P	Земля
3V3	2	P	Живлення
EN	3	I	Ввімкнення та вимкнення мікросхеми
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	I GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6,
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7,
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Земля
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
NC	17	-	Примітка 1
NC	18	-	Примітка 1
NC	19	-	Примітка 1

Продовження таблиці 2.2

NC	20	-	Примітка 1
NC	21	-	Примітка 1
NC	22	-	Примітка 1
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13,HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16[2]	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Земля

P – живлення ; I – вхід; O – вихід.

[1]Виводи GPIO6 - GPIO11 мікросхеми ESP32-D0WD-V3/ESP32D0WDR2-V3 підключені до SPI флешпам'яті інтегрованого в модуль і не виведені назовні.

[2]У варіантах модулів, які мають вбудовану QSPI PSRAM, тобто, які використовують ESP32-D0WDR2-V3, IO16 підключений до вбудованої PSRAM підключений до вбудованої PSRAM і не може бути використаний для інших функцій.

Також на сайті виробника представлений зручний інтерактивний список, за допомогою якого можна підібрати відповідну модель ESP32, що буде ідеально підходити під ваші потреби [21].

Мікроконтролер ESP32 підтримує широкий спектр програмних засобів, що забезпечують гнучкість у створенні систем обробки, передачі та аналізу інформації. Завдяки відкритій екосистемі, платформа дозволяє використовувати як високорівневі фреймворки, так і низькорівневі бібліотеки для роботи з периферією, зв’язком та логікою керування.

Існує безліч середовищ для розробки (рис 2.3) серед яких Arduino IDE, ESPHome, ESP-IDF, PlatformIO, інтерпретатор MicroPython . Для передачі даних у системах на базі ESP32 широко застосовуються такі програмні протоколи:

1. MQTT — легкий брокерний протокол для публікації/підписки, що забезпечує передачу даних у реальному часі;
2. HTTP/HTTPS — класичні вебзапити до REST-серверів;
3. WebSocket — для постійного двостороннього з'єднання;
4. CoAP, Modbus, TCP/UDP — у спеціалізованих або промислових системах.

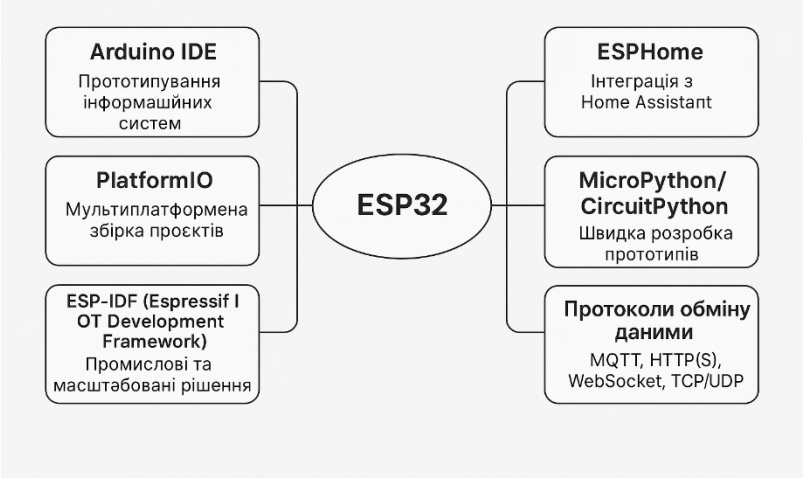

Рисунок 2.3 – Програмні засоби для програмування ESP32

ESP32 підтримує цілу низку гнучких програмних засобів, які дозволяють реалізовувати як прості, так і високонавантажені системи обробки інформації. Завдяки поєднанню потужного фреймворку ESP-IDF, зручного Arduino IDE, універсального ESPHome та мов високого рівня (Python), розробники можуть створювати рішення, що відповідають сучасним вимогам до IoT, автоматизації та смартсистем.

2.2 Структура та функціональна модель системи

Автоматизована система обробки інформації, що проєктується у межах даної роботи, реалізується на базі мікроконтролера ESP32-WROOM-32E і передбачає функціональну взаємодію між апаратними та програмними компонентами з метою забезпечення зчитування, передавання, обробки та виводу інформації у зручному для користувача вигляді. Функціональна схема проєкту представлена на рисунку 2.4.

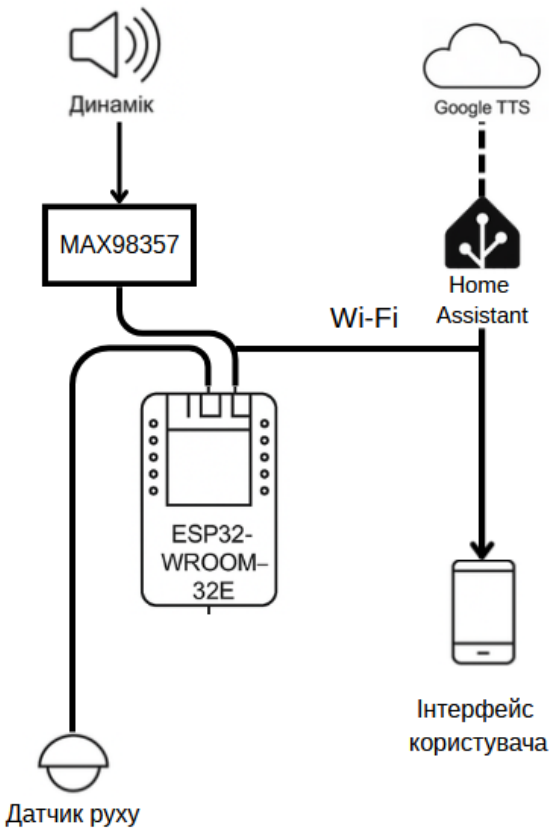

Рисунок 2.4 – Функціональна схема проєкту

На представленій схемі у ролі обчислювальної машини виступає ESP32, операційна система Home Assistant – це локальний сервер на базі Linux 2,6 / 3.x / 4.x (64-bit), виконавчий елемент – динаміки, людино-машинний інтерфейс – вебдодаток Home Assistant, датчик руху служить тригером для запуску обробки отриманої інформації (виявлення руху).

2.3 Апаратні компоненти системи

Для реалізації автоматизованої системи обробки інформації було використано низку апаратних компонентів, кожен з яких виконує чітко визначену функцію в межах загальної архітектури. Вибір елементної бази здійснювався з урахуванням критерію сумісності, енергоефективності, надійності та простоти інтеграції в систему на базі ESP32.

Центральним елементом обробки інформації є мікроконтролер ESP32-WROOM-32E, який забезпечує зчитування сигналів з периферійних пристроїв, локальну обробку подій та бездротову передачу даних до серверної частини системи. Завдяки наявності вбудованого Wi-Fi-модуля, ESP32 виконує функцію мережевого вузла, що працює у режимі публікації подій у середовище Home Assistant.

Для виявлення руху в контрольованому середовищі використовується інфрачервоний сенсор руху, що генерує цифровий сигнал при зміні інфрачервоного випромінювання, характерного для руху людини. Цей сенсор підключається до одного з цифрових входів ESP32 і слугує тригером для запуску логіки обробки подій у системі.

Голосова індикація подій реалізується за допомогою аудіопідсилювача MAX98357A (рис 2.4), що приймає цифровий аудіосигнал у форматі I2S з мікроконтролера або локального сервера, перетворюючи його на аналоговий сигнал. Підсилений аудіосигнал подається на пасивний динамік потужністю 3 Вт, який виконує відтворення згенерованих повідомлень.

Живлення системи забезпечується через USB-інтерфейс або стабілізатор 5 В, що є достатнім для підтримки роботи мікроконтролера та підключених модулів у стаціонарних або портативних умовах. Загалом апаратна архітектура системи забезпечує всі необхідні функції — від зчитування даних із середовища до голосового інформування користувача.

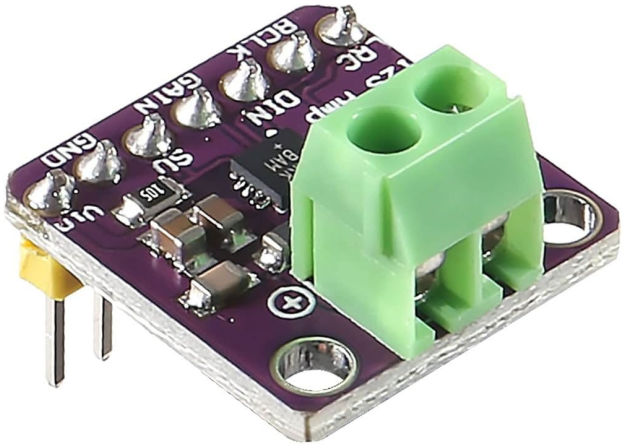

Рисунок 2.4 – Підсилювач MAX98357A

Кожен компонент системи має чітке функціональне призначення та інтегрується у загальну структуру через відповідні інтерфейси.

2.4 Огляд Home Assistant

Home Assistant являє собою програмне забезпечення з відкритим кодом, основним призначенням якої є автоматизація домашніх систем. Воно функціонує як універсальна платформа для об’єднання різноманітних пристроїв Інтернету речей. Основна увага приділяється локальному управлінню системою та захисту персональних даних користувача. Керування системою відбувається через зручний вебінтерфейс або за допомогою мобільних і десктопних застосунків доступних для Android, iOS і macOS. Крім того, підтримується голосове управління через таких асистентів, як Google Assistant, Amazon Alexa, Siri та вбудований Assist. Платформа підтримує взаємодію з широким спектром пристроїв і сервісів, які використовують стандартні відкриті протоколи — серед них MQTT, Wi-Fi, Bluetooth, Zigbee, Z-Wave, Thread Matter тощо.

Таке підключення забезпечується через відкриті API або MQTT-інтерфейси, що дозволяє налаштовувати інтеграцію як локально, так і віддалено через Інтернет. Продукт є безкоштовним і незалежним від хмарних технологій, має понад 2000 інтеграцій та потужні інструменти для автоматизації.

Як правило, Home Assistant встановлюється на окремий пристрій і функціонує за моделлю клієнт-сервер (рис 2.5).

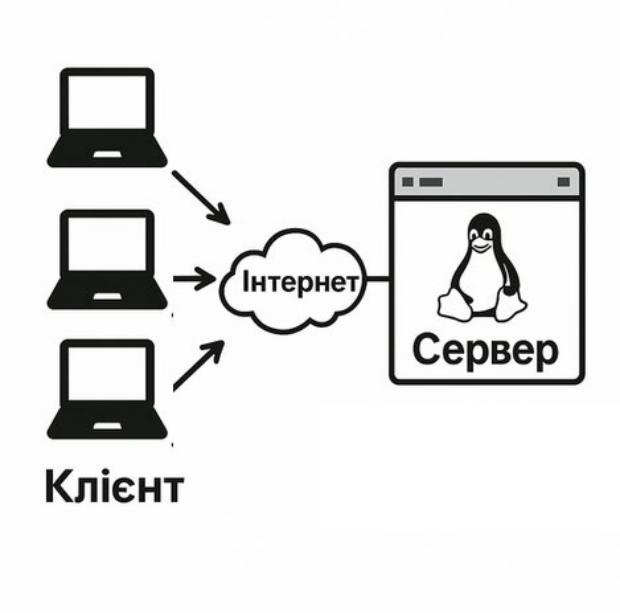

Рисунок 2.5 – Модель клієнт-сервер

Ця архітектура є найпопулярнішою концепцією у створенні розподілених мережних застосунків [22]. У ролі сервера виступає віртуальна машина, створена засобами програмного забезпечення VirtualBox.

Home Assistant використовує мову високого рівня Python, з динамічною типізацією та автоматичним керуванням пам’яттю. Вона підтримує кілька парадигм програмування, включно з об’єктноорієнтованим, імперативним, функціональним та процедурним стилем. Завдяки цьому Python забезпечує високу читабельність коду, що значно полегшує розробку та супровід програмного забезпечення. Серед характерних особливостей варто зазначити:

1. Простий і зрозумілий синтаксис, що наближений до природної мови.
2. Велика стандартна бібліотека, яка забезпечує підтримку широкого кола завдань — від обробки файлів до роботи з мережею.

3. Модульність, що дозволяє створювати масштабовані системи на основі багаторазового використання коду.

Для конфігурації системи у Home Assistant використовується мова опису даних YAML. Завдяки зрозумілому для людини синтаксису YAML-файли легко редагуються вручну. Мова YAML є ключовим інструментом конфігурації в системі Home Assistant. Вона забезпечує зрозумілий спосіб опису логіки розумного дому, дозволяючи користувачам реалізовувати складні сценарії взаємодії пристроїв та автоматизації дій. Хоча синтаксична чутливість YAML може бути викликом, її потужність і гнучкість виправдовують широке застосування у сфері побутової автоматизації.

У сучасних системах автоматизації, зокрема у платформі Home Assistant, важливим компонентом є здатність до розширення функціональності відповідно до зростаючих потреб користувача. Одним з інструментів, який забезпечує таку гнучкість, є магазин доповнень (рис. 2.5) — спеціальний модуль, що дозволяє інтегрувати сторонні або офіційно підтримувані сервіси до системи без складної ручної конфігурації. Магазин доповнень функціонує як репозиторій мікросервісів, що працюють у вигляді контейнерів та інтегруються безпосередньо в Home Assistant. За кілька кліків користувач може додати нові можливості — від локального MQTT-брокера до інструментів резервного копіювання, мережевого моніторингу або голосових сервісів.

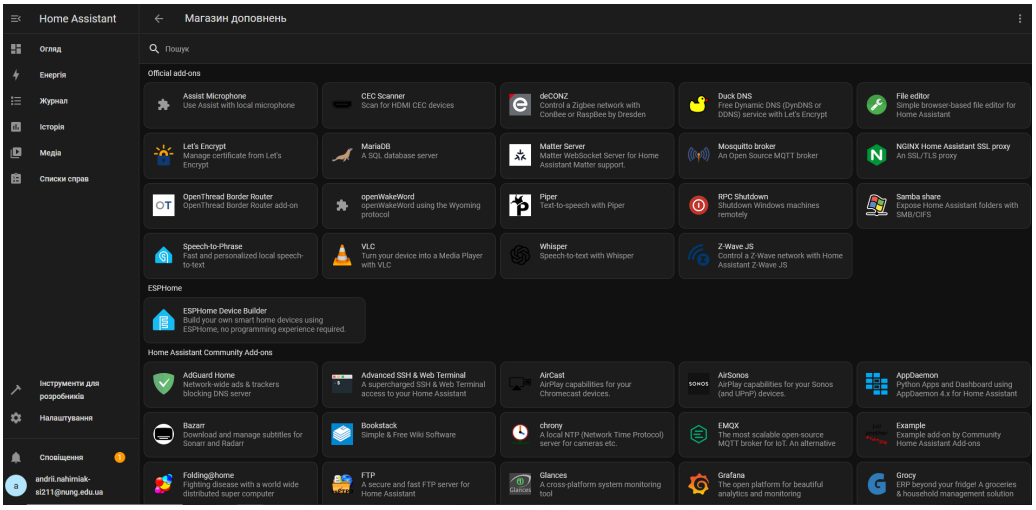

Рисунок 2.5 – Додатки у магазині доповнень

Доповнення дозволяють розширити можливості базової системи автоматизації без додаткових знань з адміністрування Linux або контейнеризації. Це надзвичайно важливо для побудови розумного середовища в домашніх умовах, де очікується простота використання, безпека та надійність. Завдяки магазину доповнень Home Assistant реалізує архітектуру модульного розширення. Кожне доповнення має власну вебконсоль або API, налаштовується через простий YAML-файл або графічний інтерфейс.

Не варто і забувати про безпеку саме тому усі доповнення запускаються в ізольованому середовищі. Офіційні доповнення проходять перевірку на відповідність стандартам спільноти Home Assistant. Для додаткових репозиторіїв користувач несе відповідальність самостійно, однак система попереджає про потенційні ризики при їх інсталяції.

Магазин доповнень у Home Assistant є ключовим елементом архітектури платформи, який забезпечує адаптивність, масштабованість і функціональну гнучкість системи автоматизації. Завдяки йому Home Assistant перестає бути лише інструментом збору даних і перетворюється на універсальний центр керування розумним середовищем**,** придатний як для побутового, так і для наукового використання.

2.4.1 ESP Home

Для конфігурації мікроконтролера ESP32 використовується доповнення ESPHome (рис. 2.6) — високорівневе середовище створення прошивок на основі YAML-конфігурацій, відкрите середовище для автоматизованого створення прошивок для мікроконтролерів родини ESP8266/ESP32, орієнтоване на спрощення інтеграції пристроїв у локальні системи автоматизації, зокрема до Home Assistant [23].

Прошивка, згенерована ESPHome, завантажується в мікроконтролер, після чого пристрій може передавати дані, реагувати на події, взаємодіяти з іншими вузлами мережі чи сервером автоматизації. Можна виділити такі основні переваги:

1. Підтримка GPIO (ввід/вивід).
2. Зчитування даних з цифрових/аналогових сенсорів.
3. Робота з протоколами I2C, SPI, UAR.
4. Управління Wi-Fi, Bluetooth та мережевими службами.
5. Підтримка локальних автоматизацій (умови, тригери, дії).
6. Безпровідне оновлення прошивки.
7. Автоматичне розпізнавання та підключення до Home Assistant.

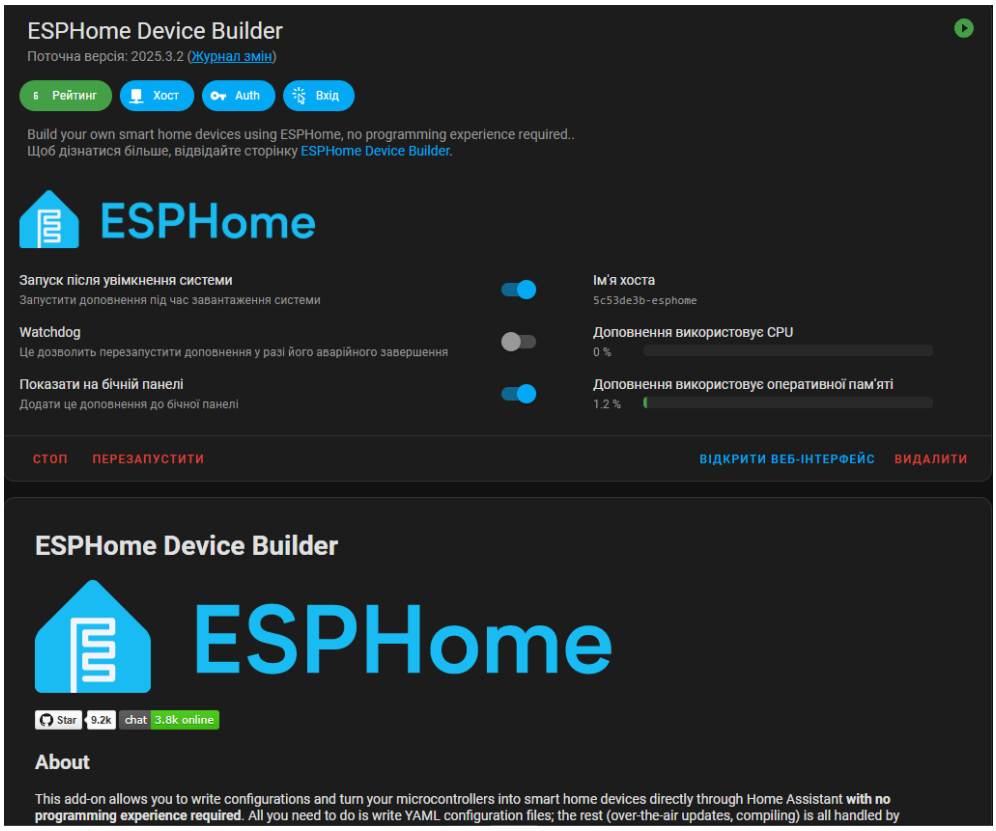

Рисунок 2.6 – Сторінка доповнення ESPHome в Home Assistant

Цей інструмент дозволить конфігурувати мережеві параметри, інтегрувати пристрої у систему Home Assistant без підключення MQTT, реалізувати обробку подій, налаштувати зчитування сигналів від датчика руху.

Крім того, ESPHome дозволяє реалізовувати умовну логіку реагування, наприклад, лише у певний час доби або за сукупністю факторів, що є особливо цінним у системах збору та обробки інформації.

Таким чином, ESPHome є потужним інструментом для розробки прошивок у системах автоматизації на базі ESP32, який поєднує простоту

використання, відкритість, активну підтримку спільноти та високий рівень функціональності. У межах реалізованої системи обробки інформації ESPHome забезпечує стабільну інтеграцію сенсорного модуля з сервером, дозволяючи розробити повноцінну систему збору, аналізу та оповіщення з мінімальними затратами часу.

2.4.2 Google Text-to-speech

Наступним додатком, що був використаний у цій роботі є Google Text-to-Speech – це хмарний сервіс, який дозволяє перетворювати текстову інформацію на природне мовлення. Рішення підтримує понад 40 мов і діалектів, включаючи українську, та пропонує різноманітні типи голосів — стандартні та нейронні. Завдяки цьому система звучить максимально наближено до живого мовлення. Фактично, Google TTS дозволяє наділити "голосом" будь-який інформаційний пристрій, надаючи користувачеві більше контексту та зручності.

Інтеграція сервісу Google TTS з Home Assistant реалізується через просту конфігурацію в configuration.yaml:

```
tts:
  - platform: google_translate
    language: 'uk'
```

Подальше озвучення здійснюється викликом сервісу tts.google_translate_say, який автоматично генерує аудіофайл і надсилає його на вказаний динамік, підключений до Home Assistant. Однак, для функціонування сервісу потрібен інтернет, оскільки синтез виконується через сторонні сервера.

Google Text-to-Speech є потужним інструментом, що дозволяє реалізувати голосовий канал взаємодії між системою та користувачем. Його застосування в проєктах автоматизації, таких як система на базі ESP32 з Home Assistant, забезпечує інтуїтивну, доступну й адаптивну форму подання інформації, роблячи технічні системи ближчими до людини. Простота інтеграції, висока якість синтезу та багатомовність роблять Google TTS одним із найкращих рішень у сфері голосових технологій на побутовому рівні.

2.5 Рішення для розгортання Home Assistant

Існує декілька основних методів встановлення Home Assistant. Найлегший з них придбання Home Assistant Green (рис. 2.7) – це інтелектуальний блок управління, що містить в собі чотиядерний процесор Arm Cortex-A55, 32 ГБ місця для збереження та 4 ГБ RAM, він повністю готовий до роботи й на ньому вже містяться готові інструменти, для його функціонування лишень потрібно вставити кабель живлення та підключити пристрій до мережі інтернет [24]. Пристрій може бути чудовим рішенням для створення системи «Розумний дім», адже містить всі необхідні інструменти в компактному вигляді. Основним недоліком є його вартість, що сягає 99 доларів на офіційному сайті, що не підходить для малобюджетних проєктів.

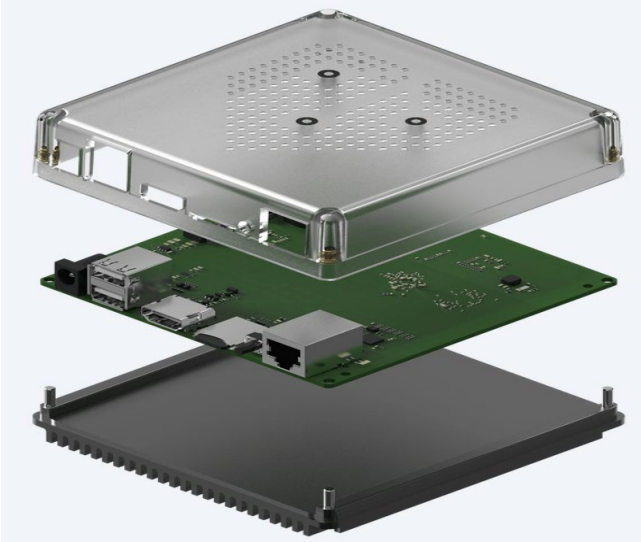

Рисунок 2.7 – пристрій Home Assistant Green

Більш складним є розгортання Home Assistant на платформі Rasberry Pi, окрім самого мінікомп'ютера знадобиться також micro SD-карта місткістю не менше 32 ГБ та з'єднання з інтернетом через Ethernet кабель або Wi-Fi і кабель живлення [25]. Вимоги до кабелю живлення відрізняються залежно від моделі, для Raspberry Pi 4 і Raspberry Pi 400 рекомендують обирати блок живлення 3A 15Вт USB-C, для Raspberry Pi 5 5A 27 Вт USB-C [26]. Схожі вимоги мають і Odroid платформи (рис. 2.8).

Рисунок 2.8 – ODROID-XU4

У підсумку було обрано розгортання Home Assistant на віртуальній машині, створеній за допомогою програмного забезпечення VirtualBox. Серед усіх запропонованих варіантів VirtualBox має ряд переваг які представлені у таблиці 2.3.

Таблиця 2.3 – Порівняння VirtualBox, VMware Workstation, Hyper-V

Характеристика	**VirtualBox**	**VMware Workstation**	**Hyper-V**
Платформа	Windows, macOS, Linux	Windows, Linux	Тільки Windows (Pro, Enterprise)
Ціна	Безкоштовно	Безкоштовна і платна Pro версія	Вбудовано безкоштовно у Windows
Простота використання	Зручний інтерфейс	Зручний інтерфейс	Складний інтерфейс
Продуктивність	Середня	Висока	Висока
Підтримка ОС-гостей	Широка	Широка	Обмежена (немає macOS)
Інтеграція з хостом	Висока	Висока	Висока з Windows

Продовження таблиці 2.3

Збереження стану VM	Так	Так	Так
Снапшоти	Так	Тільки у Pro версії	Так
Підтримка графіки/3D	Обмежена	Висока у Pro версії	Обмежена

Перш за все, VirtualBox — це безкоштовне програмне забезпечення з інтуїтивно зрозумілим інтерфейсом і підтримкою встановлення різних гостьових операційних систем, що робить його цілком придатним для виконання поставленого завдання.

2.5.1 Інсталяція Home Assistant за допомогою VirtualBox

Встановлення поділяється на два етапи:

1. Створення віртуальної машини з операційною системою Linux 2,6 / 3.x / 4.x (64-bit).
2. Інсталяція ядра Home Assistant.

Розглянемо перший та другий етап детальніше. Спершу потрібно завантажити відповідне зображення ОС, яке можна знайти на офіційному сайті Home Assistant у розділі інсталяції [27]. У програмі Virtual Box створюємо нову віртуальну машину та вказуємо тип Linux і версію Linux 2,6 / 3.x / 4.x (64-розрядна). У розділі «Обладнання» потрібно обрати обсяг пам’яті та кількість ЦП.

Мінімум 2 ГБ оперативної пам'яті, 32 ГБ пам'яті сховища та 2 ядра. Важливим є увімкнення опції EFI якщо його не ввімкнено, HAOS не завантажиться. У розділі «Жорсткий диск» потрібно обрати «Використовувати наявний віртуальний жорсткий диск», і обрати раніше завантажений файл VDI. У розділі «Мережа» потрібно налаштувати мережевий адаптер «Bridged Adapter».

На рисунку 2.9 зображено кінцевий вигляд інстальованої ОС.

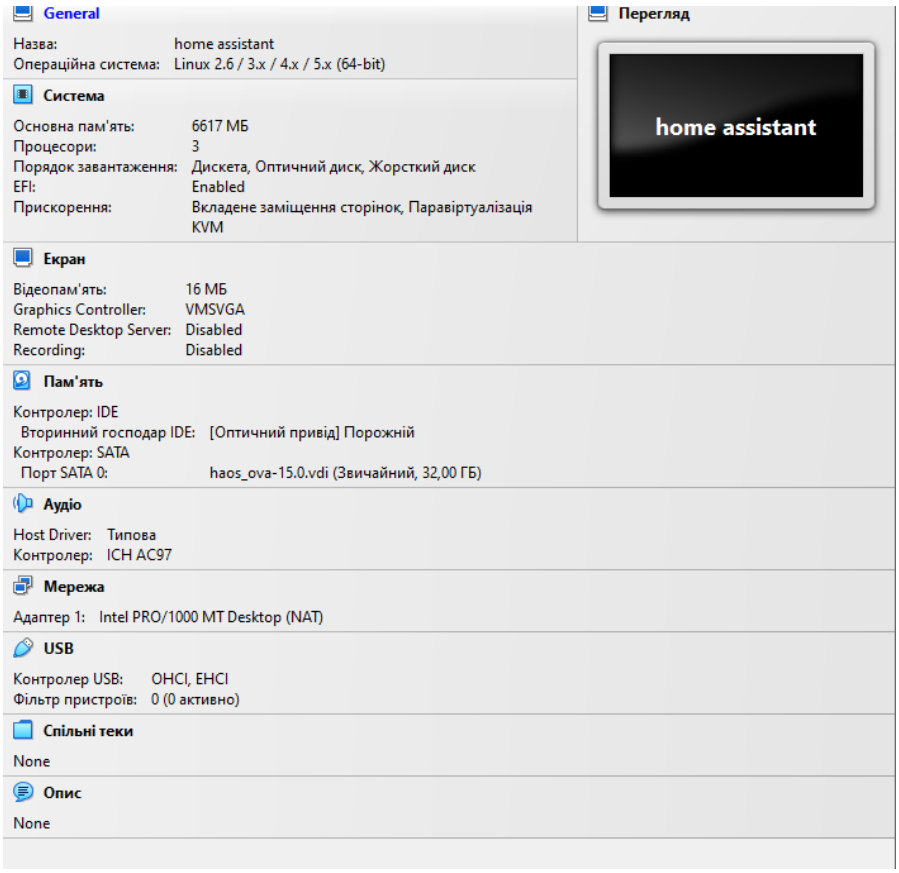

Рисунок 2.9 – Вікно з інстальованою Linux у середовищі VirtualBox

Після запуску віртуальної машини відбудеться процес встановлення HAOS, у результаті ми отримаємо адресу WEB-сторінки (рис 2.10), увівши її в рядку браузера ми перейдемо до створеного середовища Home Assistant (рис. 2.11) в якому можна продовжувати роботу. Також через консоль можна керувати нею.

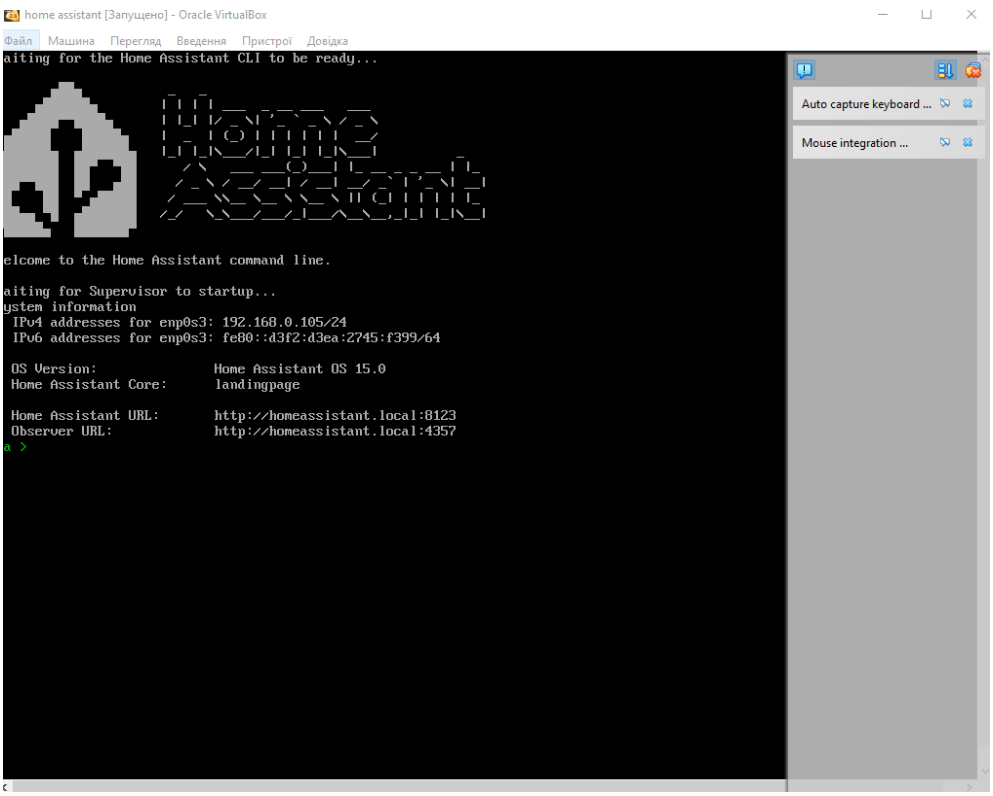

Рисунок 2.10 – Запущений сервер Home Assistant

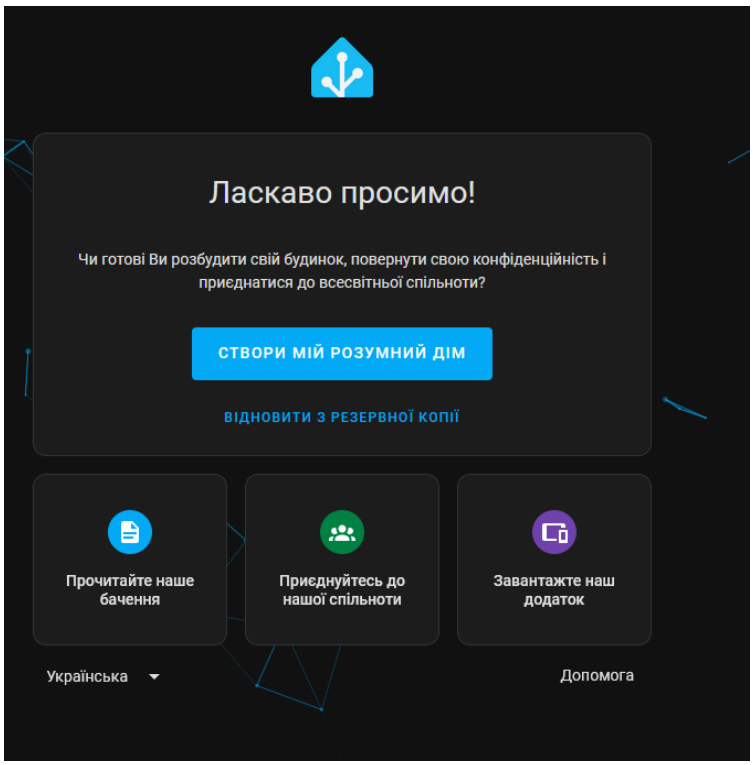

Рисунок 2.11 – Вітальне вікно Home Assistant

2.6 Методи забезпечення надійності та захисту обробки інформації

У сучасних IoT-системах одним із базових принципів захисту є використання багаторівневої автентифікації, яка поєднує класичні паролі з одноразовими ключами, токенами або біометричними параметрами. Це дозволяє значно підвищити стійкість до атак типу "brute-force" і забезпечити контроль над доступом до критичних компонентів системи [28].

Забезпечення надійної та безпечної обробки інформації є ключовим чинником ефективного функціонування будь-якої автоматизованої системи. У системах обробки інформації, особливо побудованих на мікроконтролерах, важливо передбачати механізми резервного збереження стану системи, наприклад, у flash-пам’яті. У разі збоїв це дозволяє відновити останній відомий стан та уникнути повторного запуску сценаріїв [28].

У розробленій системі, яка базується на мікроконтролері ESP32 та платформі Home Assistant, передбачено комплекс заходів (рис 2.12), що спрямовані на мінімізацію ризиків втрати даних, хибних спрацювань та несанкціонованого доступу до ресурсів системи. Однією з важливих умов досягнення надійності є зменшення впливу перешкод та випадкових збурень, які можуть спричинити хибні спрацювання сенсорів, зокрема інфрачервоного датчика руху. У системі реалізовано програмну фільтрацію таких подій шляхом використання затримок обробки сигналу, обмеження частоти повторного виклику сценаріїв, а також перевірки тривалості активного стану сенсора. Це дозволяє зменшити кількість хибно ініційованих реакцій та підвищити правдивість інформаційного потоку.

Особливу увагу в проєкті приділено безпеці передавання даних. Використовується захищені канали зв’язку з використанням протоколу SSL/TLS, що гарантує цілісність та конфіденційність інформації.

Доступ до керування системою обмежується механізмами автентифікації в інтерфейсі Home Assistant, що унеможливлює втручання з боку сторонніх осіб. Інформація про всі події в системі реєструється у журналах подій Home Assistant.

У процесі експлуатації важливим є також забезпечення простоти оновлення та відновлення системи після збоїв. Для цього ESPHome підтримує бездротове оновлення прошивки, що дозволяє вносити зміни до конфігурації без фізичного доступу до пристрою. Home Assistant, у свою чергу, підтримує автоматичний перезапуск сервісів та створення резервних копій (рис 2.13), що спрощує процес відновлення у разі пошкодження або втрати даних.

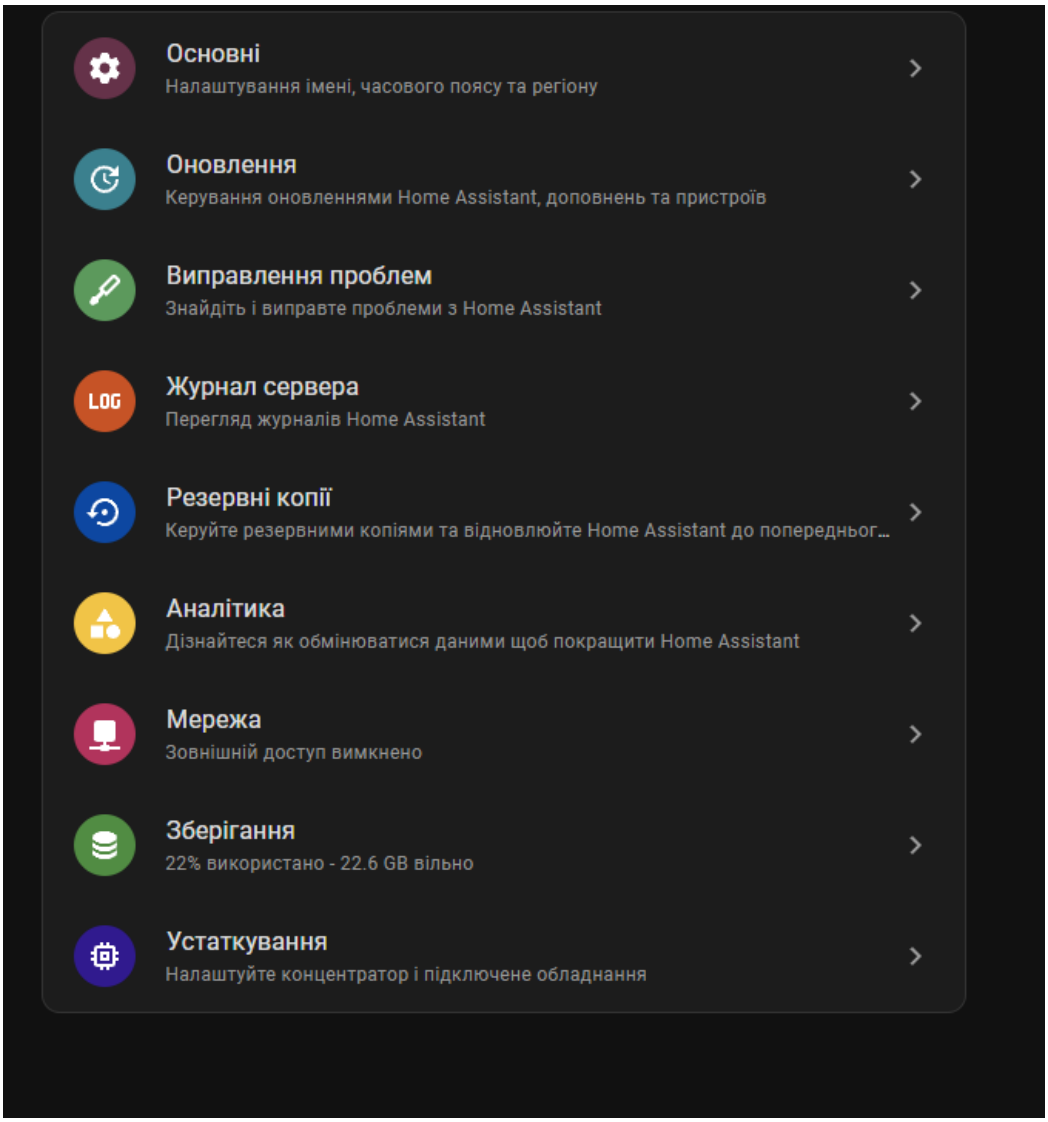

Рисунок 2.12 – Засоби налаштування і безпеки в Home Assistant

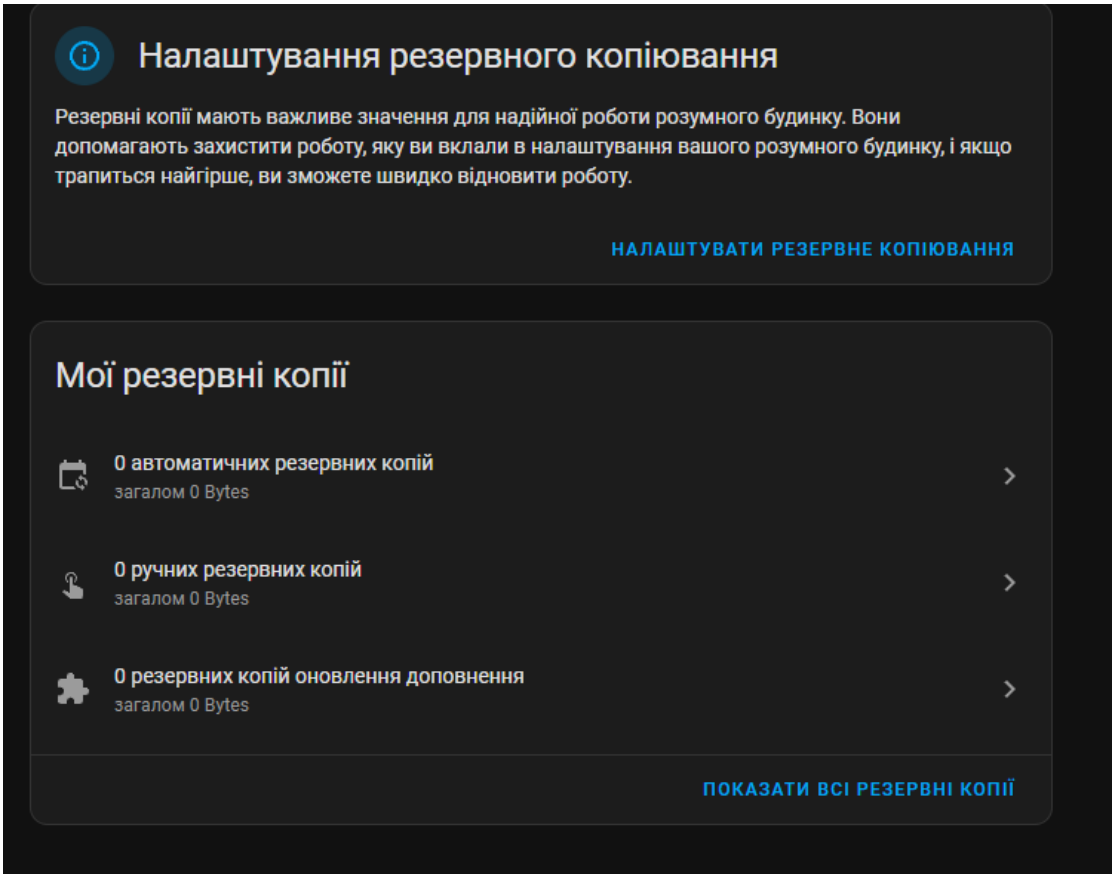

Рисунок 2.13 – Можливість резервного копіювання

У даному розділі були розглянуті основні компоненти та інструменти, які були використані при проєктуванні автоматизованої системи обробки інформації з використанням мікроконтролера ESP32-WROOM-32E та програмної платформи Home Assistant. Розроблена система орієнтована на виявлення подій (зокрема, руху), подальшу передачу інформації, її обробку згідно з заданими умовами та інформування користувача через голосовий інтерфейс на базі Google Text-to-Speech. У межах розділу сформовано функціональну модель системи, розроблено структурну схему та описано логіку взаємодії апаратних і програмних компонентів. Здійснено обґрунтований вибір мікроконтролера, сенсорного модуля, аудіопідсилювача MAX98357, а також необхідного програмного забезпечення, включно з ESPHome, Google TTS та Home Assistant. Таким чином, розроблена система є функціонально завершеною, модульною та масштабованою.

Вона відповідає сучасним вимогам до побудови інтелектуальних інформаційних систем побутового або навчального призначення та може бути розширена для роботи з додатковими сенсорами або інтегрована в ширші інфраструктури розумного середовища.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1 Підключення ESP-32

Для роботи з ESP-32, спершу потрібно встановити доповнення «ESPHome» , його можна знайти в магазині доповнень, після на бічній панелі з'явиться вкладка ESPHome Builder – це місце де можна додавати нові пристрої. Для цього ми натиснемо на кнопку «New Device» і перейдемо до підключення нашої плати (рис. 3.1), вводимо ім'я й обираємо тип пристрою (рис. 3.2) в моєму випадку опція «Esp32». Тепер я можу встановити конфігурацію на пристрій. Роблячи це вперше, знадобиться кабель, але після того, як пристрій буде встановлено і налаштовано я зможу керувати ним бездротовим шляхом. Також у наступному вікні з'явився унікальний ключ шифрування, він знадобиться, щоб додати пристрій в Home Assistant.

Щоб додати пристрій до ESPHome, його потрібно підключити до комп'ютера за допомогою USB-кабелю.

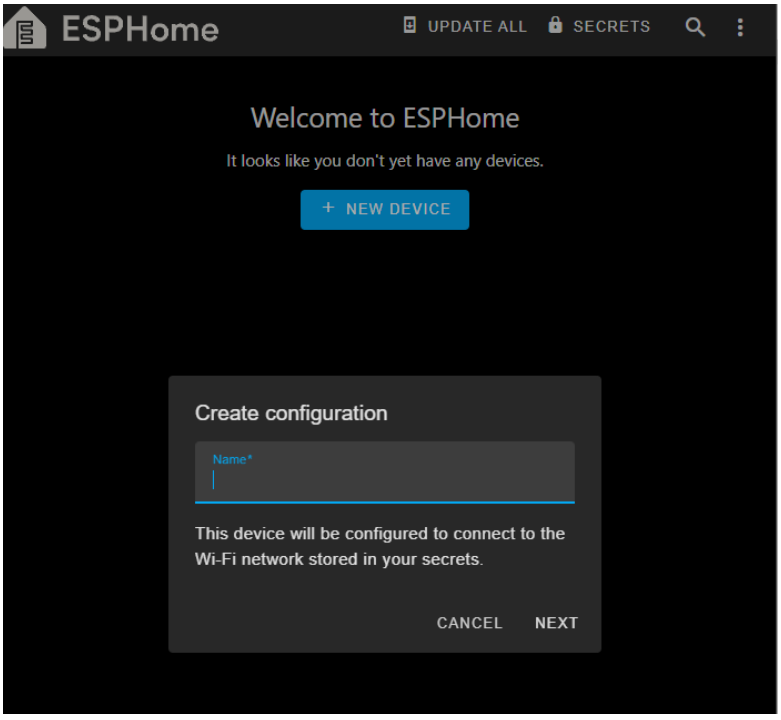

Рисунок 3.1 – Створення конфігурації

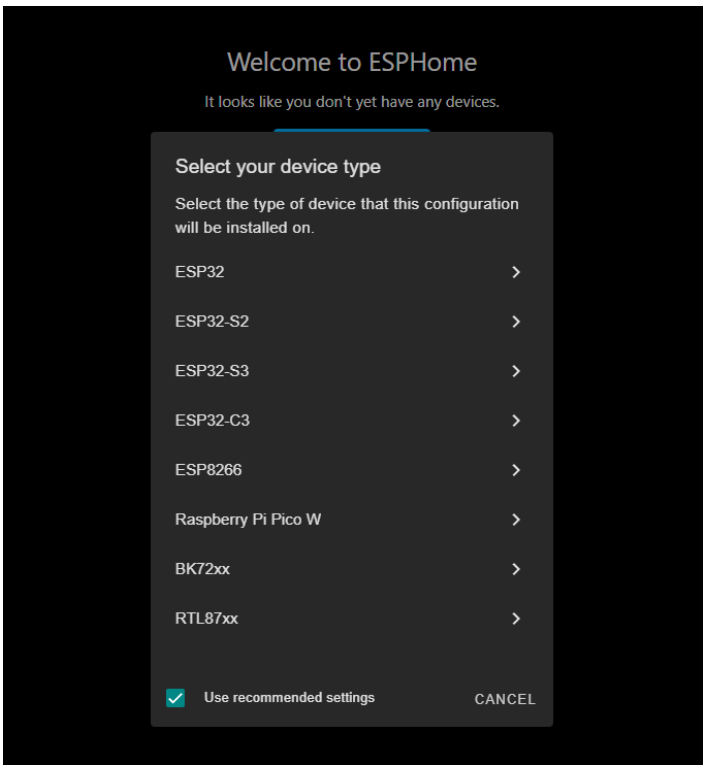

Рисунок 3.2 – Вибір плати

Після цих дій пристрій з'явиться на головній сторінці ESPHome (рис. 3.3), проте тепер потрібно завантажити файл конфігурації «esp32.yaml» на плату, для цього натискаємо на три крапочки та з випадаючого меню обираємо «Install». Далі потрібно обрати яким чином код буде завантажений на нього, є декілька варіантів:

1. Бездротовим способом, потребує, щоб пристрій був онлайн.
2. Підключенням через комп'ютер, потребує, щоб пристрій був з'єднаний з комп'ютером через USB-кабель.
3. Через комп'ютер на якому запущено ESPHome, для пристроїв, які підключені до сервера через USB-кабель.
4. Завантаження вручну, через ESPHome Web.

Обираємо другий варіант і очікуємо коли наш файл буде готовим для завантаження, після завантажуємо його і тиснемо на посилання, яке переведе нас на сторінку представлену на рисунку 3.4.

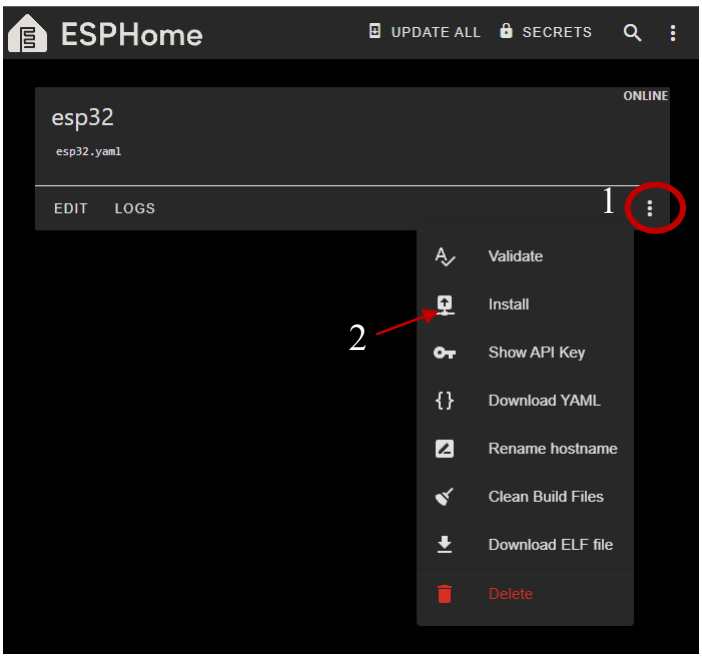

Рисунок 3.3 – Доданий пристрій

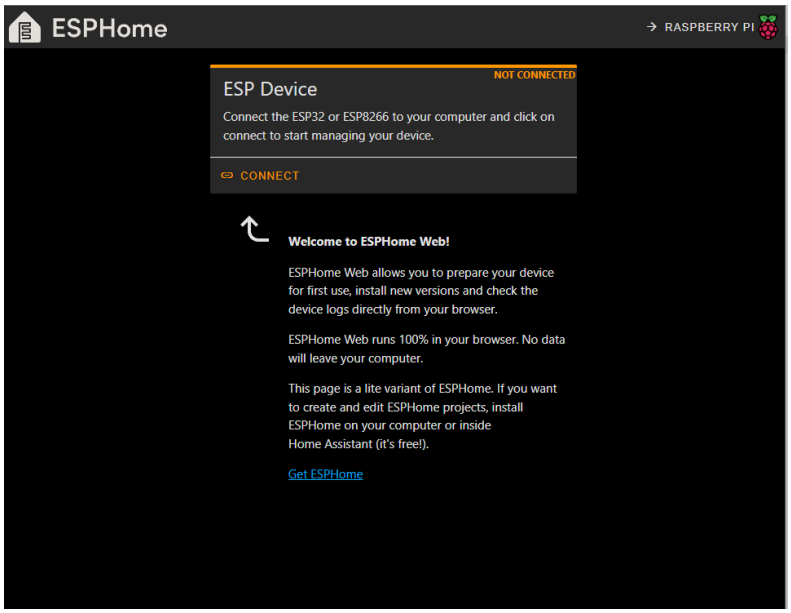

Рисунок 3. 4 – ESPHome Web

Тиснемо на «Connect» і вибираємо COM порт у який підключена плата, далі «Install» (рис 3.5) і обираємо файл, який завантажили раніше. Процес інсталяції займає близько двох хвилин, після успішного завершення наш пристрій перейде в режим «онлайн» (рис 3.6) і буде готовий до використання.

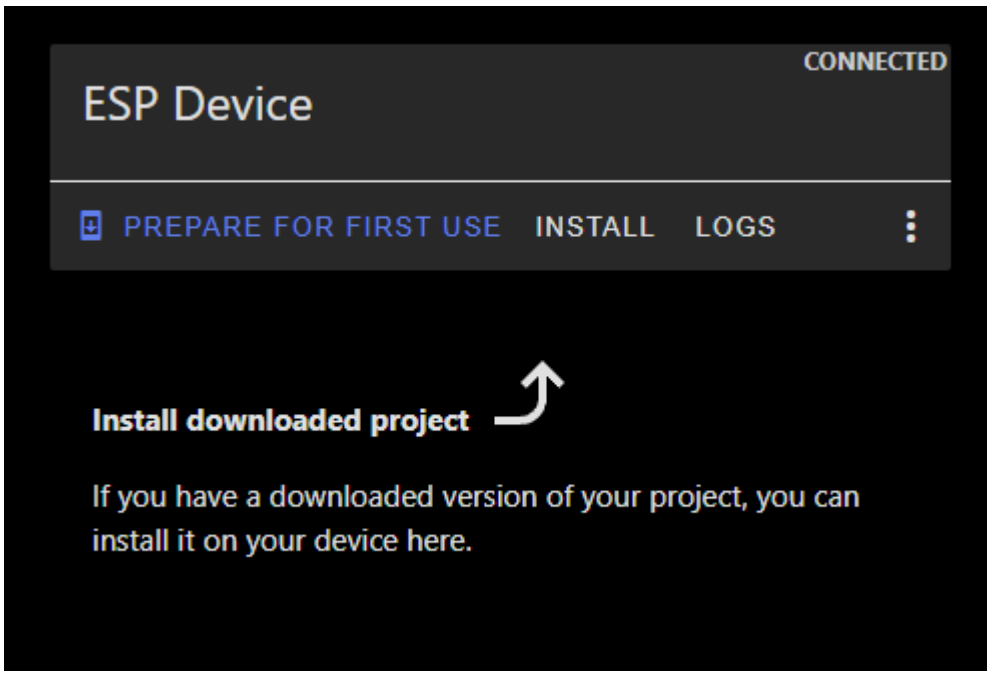

Рисунок 3.5 – Інсталяція проєкту

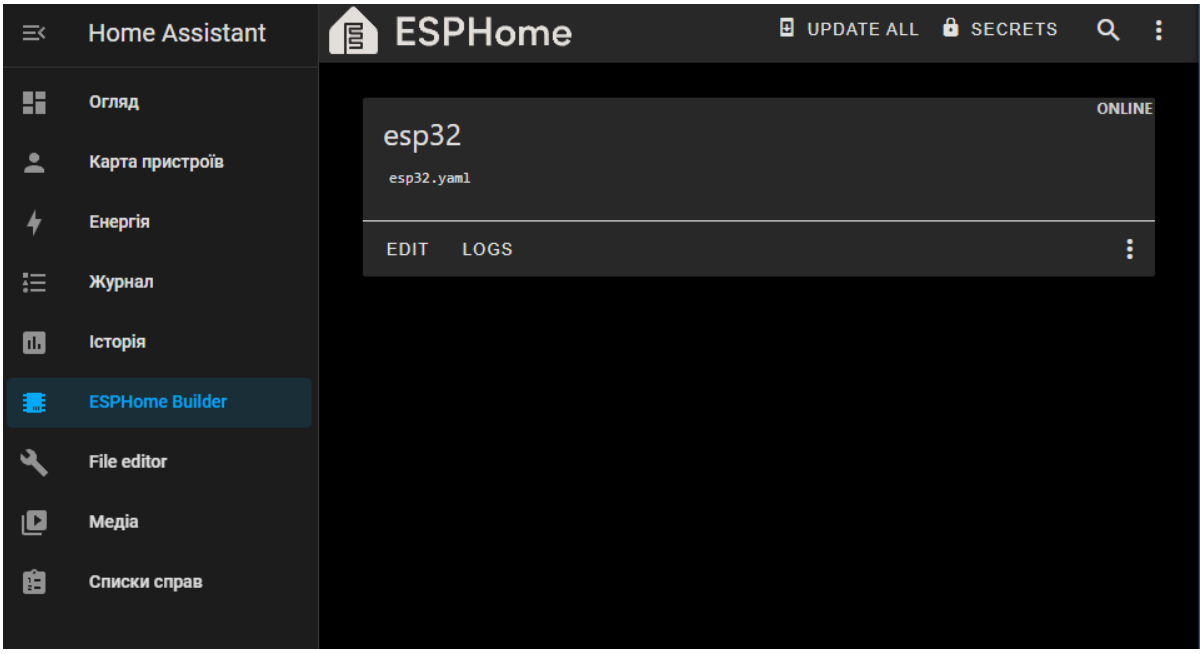

Рисунок 3.6 – Оновлений статус пристрою

3.2 Підключення нових пристроїв

Тепер додамо нові пристрої. У yaml конфігурації міститься код:

1) Фрагмент з заголовком та інформацією про пристрій:

esphome:

```
  name: esp32
  friendly_name: esp32
```

2) Налаштування плати ESP32:

```
esp32:
  board: esp32dev
  framework:
    type: esp-idf
```

3) Підтримання сенсорного введення:

```
esp32_touch:
```

4) Інтеграція з Home Assistant через API:

```
api:
  encryption:
    key: "0V54mKh5gfLixaKcm7CVGv4HSgeNRiV1lAbtBJ1jblA="
```

5) OTA – оновлення по Wi-Fi, дозволяє прошивати плату по Wi-Fi

```
ota:
  platform: esphome
  password: "0ad258943e6b15d7e55cf3ceac868a5e"
```

6) Налаштування мережі:

```
wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password
```

7) Встановлення режиму fallback, якщо Wi-Fi недоступний, ESP створює свою точку доступу:

```
 ap:
    ssid: "Esp32 Fallback Hotspot"
    password: "TRiQq53byDle"
```

8) Налаштування сенсора руху:

```
  - platform: gpio
    pin: 13
```

```
    name: "PIR Sensor"
    device_class: motion
```

9) Налаштування сенсорних входів :

```
binary_sensor:
  - platform: esp32_touch
    name: "Touch Sensor T0"
    pin: GPIO4
    threshold: 700
  - platform: esp32_touch
    name: "Touch Sensor T1"
    pin: GPIO15
    threshold: 700
  - platform: esp32_touch
    name: "Touch Sensor T2"
    pin: GPIO32
    threshold: 700
  - platform: esp32_touch
    name: "Touch Sensor T3"
    pin: GPIO27
    threshold: 700
  - platform: esp32_touch
    name: "Touch Sensor T4"
    pin: GPIO14
    threshold: 700
  - platform: esp32_touch
    name: "Touch Sensor T5"
    pin: GPIO12
    threshold: 700
```

10) Додання медіа програвача:

```
speaker:
  - platform: i2s_audio
    id: va_speaker
    i2s_audio_id: i2s_out
    dac_type: external
    i2s_dout_pin: GPIO22
    channel: mono
    bits_per_sample: 16bit
    sample_rate: 16000
media_player:
  - platform: speaker
    id: va_media_player
    name: "ESP I2S Speaker"
    announcement_pipeline:
      speaker: va_speaker
      format: WAV
    codec_support_enabled: false
    buffer_size: 6000
```

Після завантаження коду, в Home Assistant у розділі «Пристрої та сервіси», у вкладці ESPHome з’явиться пристрій «ESP32» та сутності (рис 3.7).

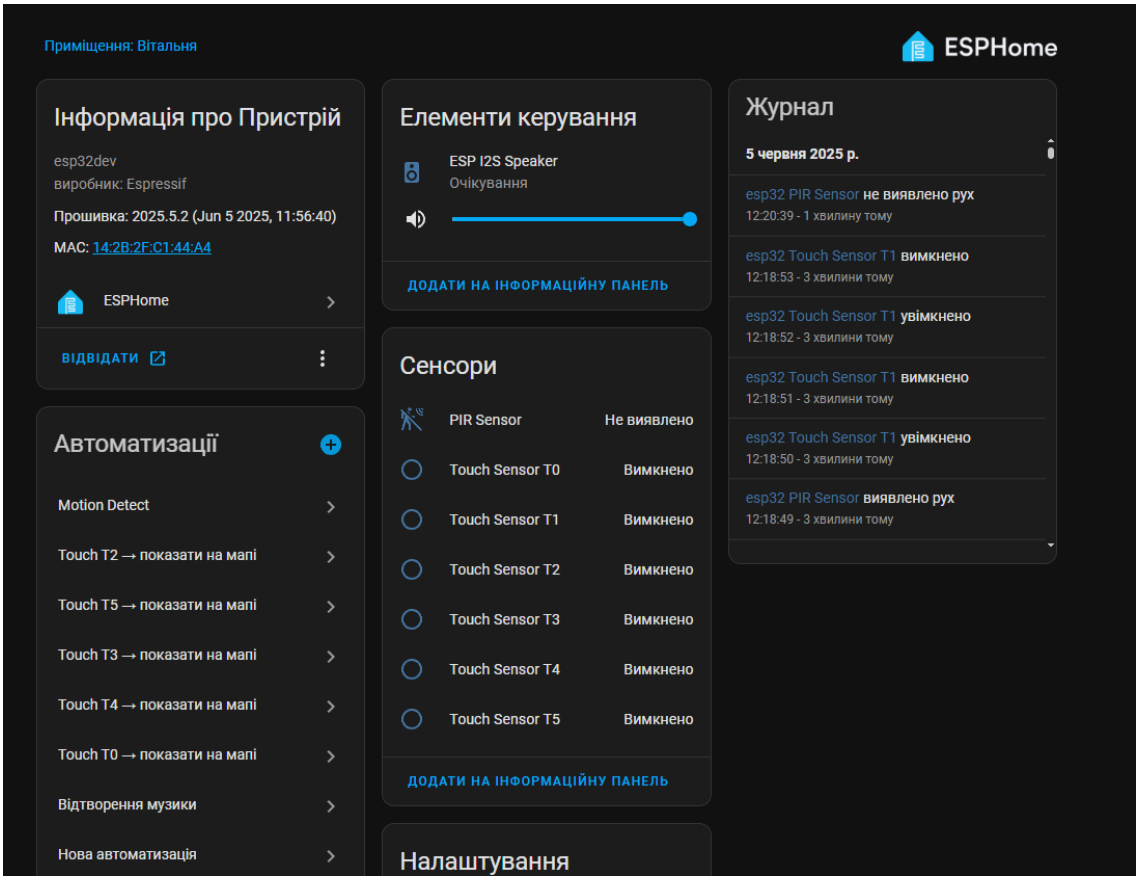

Рисунок 3.7 – Список сутностей у ESPHome

3.3 Створення карти пристроїв

Для забезпечення ефективного тестування та подальшого впровадження автоматизованої системи обробки інформації було здійснено створення карти пристроїв, яка візуалізує їхнє фізичне розташування. У Home Assistant, карта є одним із типів інформаційних панелей, тому її створення починається з додання панелі у вкладці «Налаштування»>«Інформаційні панелі» (рис 3.8), після чого вона з’явиться на панелі.

Налаштування карти пристроїв можливе за допомогою графічного інтерфейсу та редактору коду. У створеній панелі додано картку з картою та графік активності сенсорів. Оскільки сенсори не мають GPS-трекера для їх відображення на карті потрібно створити власні модифіковані сутності device.tracker у configuration.yaml.

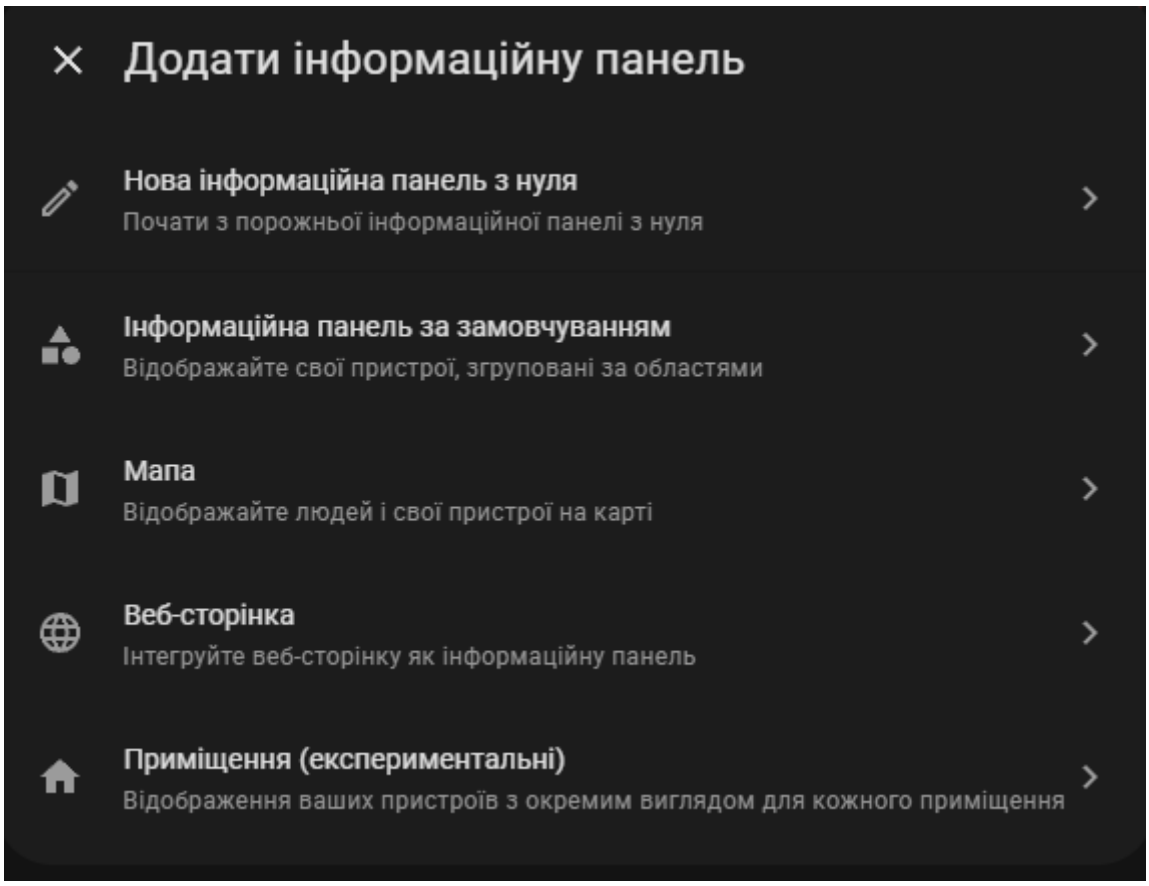

Рисунок 3.8 – Вікно з вибором типа панелі

Файл configuration.yaml містить код:

1) Основна конфігурація:

```
homeassistant:
  name: Home
  latitude: 48.916889
  longitude: 24.714500
  unit_system: metric
  time_zone: Europe/Kyiv
```

2) Змінення відображення сутності в інтерфейсі Home Assistant:

```
customize:
    device_tracker.touch_sensors_tracker:
      friendly_name: "touch sensor"
icon: mdi:gesture-tap
```

3) Додання MQTT-трекера, що дозволяє вручну або автоматично оновлювати координати:

```
mqtt:
  device_tracker:
    - platform: mqtt
      devices:
        pir_sensor_tracker: 'location/pir_sensor'
    - name: touch_sensors_tracker
      state_topic: "home/sensor_tracker/state"
      json_attributes_topic: "home/sensor_tracker/attributes"
      payload_home: "home"
      payload_not_home: "not_home"
```

4) Створення локації для кожного сенсора:

```
default_config:
zone:
  - name: "Touch Sensor T0"
    latitude: 48.922476
    longitude: 24.710187
    radius: 10
    icon: mdi:gesture-tap
  - name: "Touch Sensor T1"
    latitude: 48.917397
    longitude: 24.708608
    radius: 10
    icon: mdi:gesture-tap
  - name: "Touch Sensor T2"
    latitude: 48.919061
    longitude: 24.714439
    radius: 10
```

```
    icon: mdi:gesture-tap
  - name: "Touch Sensor T3"
    latitude: 48.921274
    longitude: 24.707161
    radius: 10
    icon: mdi:gesture-tap
  - name: "Touch Sensor T4"
    latitude: 48.918172
    longitude: 24.708958
    radius: 10
    icon: mdi:gesture-tap
  - name: "Touch Sensor T5"
    latitude: 48.919714
    longitude: 24.705178
    radius: 10
    icon: mdi:gesture-tap
  - name: "PIR Sensor"
    latitude: 48.9232
    longitude: 24.7117
    radius: 10
    icon: mdi:motion-sensor
```

5) Підключення файлу автоматизації:

```
automation: !include automations.yaml
```

6) Налаштування мови Text-to-Speech:

```
tts:
  - platform: google_translate
    language: 'uk'
```

У automations.yaml міститься код, який при активації сенсора показує його місце розташування на карті і після затримки в 1 хвилину зникає з карти, якщо сенсор не було активовано повторно:

```
- alias: "Touch T0 → показати на мапі"
  trigger:
    - platform: state
      entity_id: binary_sensor.touch_0
      to: "on"
  action:
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t0_tracker
        gps: [48.922476, 24.710187]
        attributes:
          icon: mdi:gesture-tap
    - delay: "00:01:00"
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t0_tracker
        gps: [0, 0]
        attributes:
          icon: mdi:gesture-tap
  mode: restart
- alias: "Touch T1 → показати на мапі"
  trigger:
    - platform: state
      entity_id: binary_sensor.touch_1
      to: "on"
  action:
```

```
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t1_tracker
        gps: [48.917397, 24.708608]
        attributes:
          icon: mdi:gesture-tap
    - delay: "00:01:00"
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t1_tracker
        gps: [0, 0]
        attributes:
          icon: mdi:gesture-tap
  mode: restart
- alias: "Touch T2 → показати на мапі"
  trigger:
    - platform: state
      entity_id: binary_sensor.touch_2
      to: "on"
  action:
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t2_tracker
        gps: [48.919061, 24.714439]
        attributes:
          icon: mdi:gesture-tap
    - delay: "00:01:00"
    - service: device_tracker.see
      data:
```

```
      dev_id: touch_sensor_t2_tracker
      gps: [0, 0]
      attributes:
        icon: mdi:gesture-tap
  mode: restart
- alias: "Touch T3 → показати на мапі"
  trigger:
    - platform: state
      entity_id: binary_sensor.touch_3
      to: "on"
  action:
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t3_tracker
        gps: [48.921274, 24.707161]
        attributes:
          icon: mdi:gesture-tap
    - delay: "00:01:00"
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t3_tracker
        gps: [0, 0]
        attributes:
          icon: mdi:gesture-tap
  mode: restart
- alias: "Touch T4 → показати на мапі"
  trigger:
    - platform: state
      entity_id: binary_sensor.touch_4
```

```
    to: "on"
  action:
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t4_tracker
        gps: [48.918172, 24.708958]
        attributes:
          icon: mdi:gesture-tap
    - delay: "00:01:00"
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t4_tracker
        gps: [0, 0]
        attributes:
          icon: mdi:gesture-tap
  mode: restart
- alias: "Touch T5 → показати на мапі"
  trigger:
    - platform: state
      entity_id: binary_sensor.touch_5
      to: "on"
  action:
    - service: device_tracker.see
      data:
        dev_id: touch_sensor_t5_tracker
        gps: [48.919714, 24.705178]
        attributes:
          icon: mdi:gesture-tap
    - delay: "00:01:00"
```

```
  - service: device_tracker.see
    data:
      dev_id: touch_sensor_t5_tracker
      gps: [0, 0]
      attributes:
        icon: mdi:gesture-tap
  mode: restart
```

Також у файлі реалізовано автоматизацію для сенсора руху:

```
alias: PIR → показати на мапі
  triggers:
  - entity_id:
    - binary_sensor.esp32_pir_sensor
    from: 'off'
    to: 'on'
    trigger: state
  actions:
  - data:
      dev_id: pir_sensor_tracker
      gps:
      - 48.9232
      - 24.7117
      attributes:
        friendly_name: PIR Sensor
        icon: mdi:motion-sensor
    action: device_tracker.see
  - delay: 00:01:00
  - data:
      dev_id: pir_sensor_tracker
      location_name: not_home
```

action: device_tracker.see

mode: single

Автоматизація відтворення mp3 файлу та озвучення Google TTS

alias: Відтворення музики

description: "

1) Тригери, які запускають автоматизацію:

triggers:

- trigger: state

entity_id:

- binary_sensor.esp32_touch_sensor_t1

to: 'on'

from: 'off'

2) Умови автоматизація виконається тільки якщо PIR-сенсор руху також активний:

conditions:

- condition: state

entity_id: binary_sensor.esp32_pir_sensor

state: 'on'

3) Дії, що виконуються умовний блок, що дозволяє виконувати різні сценарії залежно від станів:

actions:

- choose:

4) Варіант перший, якщо активний Touch Sensor T0, запускається аудіофайл ssstik.io_1745662354387.mp3, який зберігається у теці media/local :

- conditions:

- condition: state

entity_id: binary_sensor.esp32_touch_sensor_t0

state: 'on'

sequence:

```
    - action: media_player.play_media
      target:
        entity_id: media_player.esp32_esp_i2s_speaker
      data:
        media_content_id:media-
source://media_source/local/ssstik.io_1745662354387.mp3
        media_content_type: audio/mpeg
      metadata:
        title: ssstik.io_1745662354387.mp3
        thumbnail:
        media_class: music
        children_media_class:
        navigateIds:
        - {}
        - media_content_type: app
          media_content_id: media-source://media_source
```

5) Варіант другий, якщо є рух, то голосовим повідомленням проговорюється: «Виявлено рух» через той самий медіаплеєр.

```
  - conditions:
      - condition: state
        entity_id: binary_sensor.esp32_pir_sensor
        state: 'on'
      sequence:
      - action: tts.google_translate_say
        data:
          cache: false
          entity_id: media_player.esp32_esp_i2s_speaker
          message: Виявлено рух
          language: uk
```

mode: single

3.2.1 Графік спрацювання пристроїв

Для візуалізації активності сенсорів системи у реальному часі та відображення історії подій, що надходили з пристроїв, у середовищі Home Assistant було реалізовано спеціалізовану графічну компоненту на основі картки mini-graph-card. Цей елемент є кастомним та використовується для виведення графіків зміни станів бінарних сенсорів із високою роздільною здатністю.

До графіка було додано наступні пристрої:

- шість сенсорних каналів T0-T5 модуля ESP32;
- інфрачервоний датчик руху PIR .

Графік дозволяє відстежувати момент спрацювання кожного сенсора, оцінювати частоту активності, виявляти аномалії або часові закономірності.

Конфігурація графіка в середовищі Home Assistant реалізується за допомогою наступного YAML-коду:

```
type: custom:mini-graph-card
name: Історія сенсорів
entities:
  - entity: binary_sensor.esp32_touch_sensor_t0
    name: T0
  - entity: binary_sensor.esp32_touch_sensor_t1
    name: T1
  - entity: binary_sensor.esp32_touch_sensor_t2
    name: T2
  - entity: binary_sensor.esp32_touch_sensor_t3
    name: T3
  - entity: binary_sensor.esp32_touch_sensor_t4
    name: T4
  - entity: binary_sensor.esp32_touch_sensor_t5
```

```
    name: T5
  - entity: binary_sensor.esp32_pir_sensor
    name: PIR
 state_map:
  - value: "on"
    label: "1"
    color: green
  - value: "off"
    label: "0"
    color: red
 hours_to_show: 0.5
 points_per_hour: 60
 show:
  legend: true
  labels: true
  icon: false
```

У цій конфігурації:

- entities: вказує перелік сенсорів, які відображаються на графіку;
- state_map: визначає, як інтерпретуються логічні стани on/off (1 — активний, 0 — неактивний);
- hours_to_show: визначає період, за який будуються дані (30 хвилин);
- points_per_hour: налаштовує частоту відображення точок, забезпечуючи деталізацію;
- show: дозволяє вмикати або вимикати легенду, іконки, підписи тощо.

Графік оновлюється динамічно та інтерактивно у вебінтерфейсі Home Assistant, дозволяючи в режимі реального часу оцінити поведінку користувача, якість роботи сенсорів та логіку запуску подій у системі.

3.3 Тестування роботи системи

Після завершення розробки апаратної та програмної частини системи було проведено етап функціонального тестування з метою перевірки коректності її роботи. Перевірка охоалюввала наступні етапи:

1. Поточні стани сутностей відображаються в двох станах off – коли вимкнені, on – коли ввімкнені (рис 3.9).
2. Розроблені автоматизації працюють справно (рис 3.10).
3. На карті відображається місцеперебування сенсора при його активації і автоматично зникає через 1 хвилину (рис 3.11).
4. На графіку візуалізується історія спрацювань сенсорів (рис 3.12).
5. Відбувається відтворення голосового повідомлення та mp3 файлу (рис. 3.13).
6. Усі файли компілюються без помилок (рис 3.14).

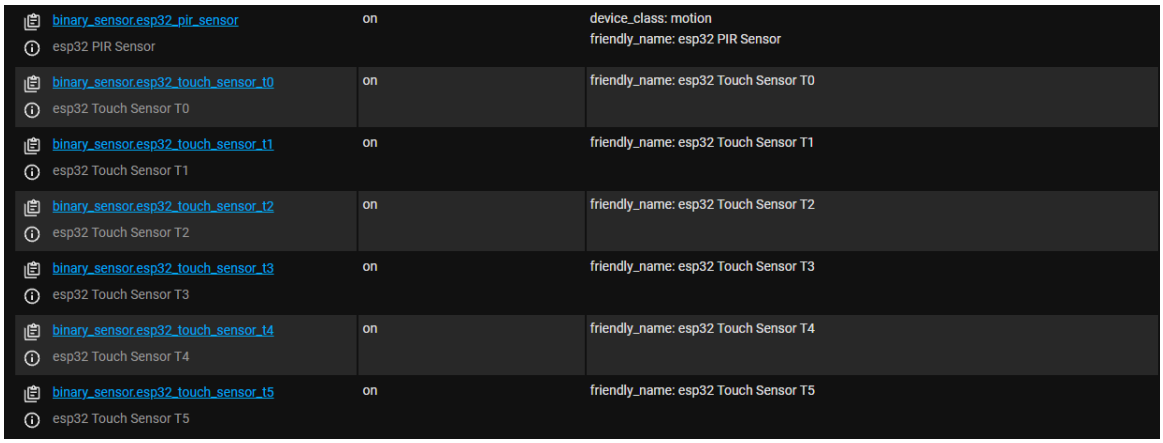

Рисунок 3.9 – Вікно розробника з станами сутностей системи

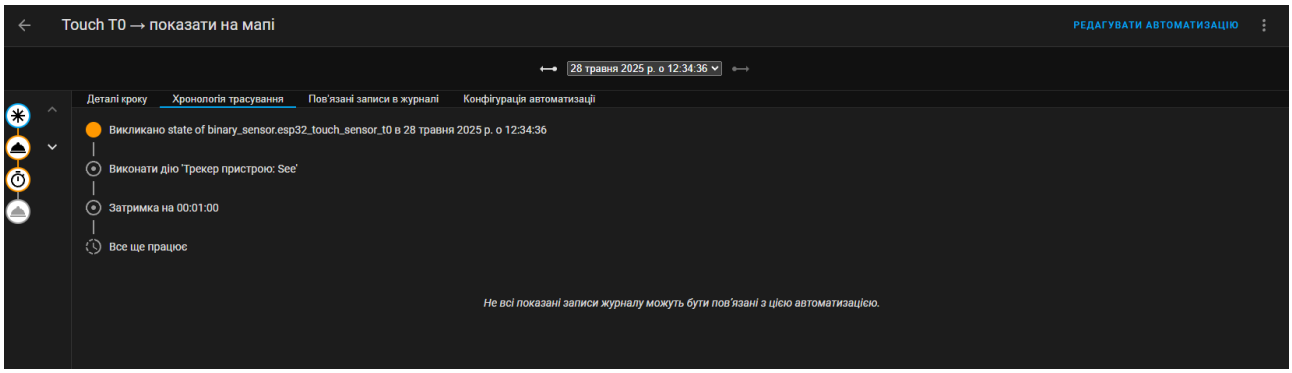

Рисунок 3.10 – Приклад трасування автоматизації «Touch T0 – показати на мапі»

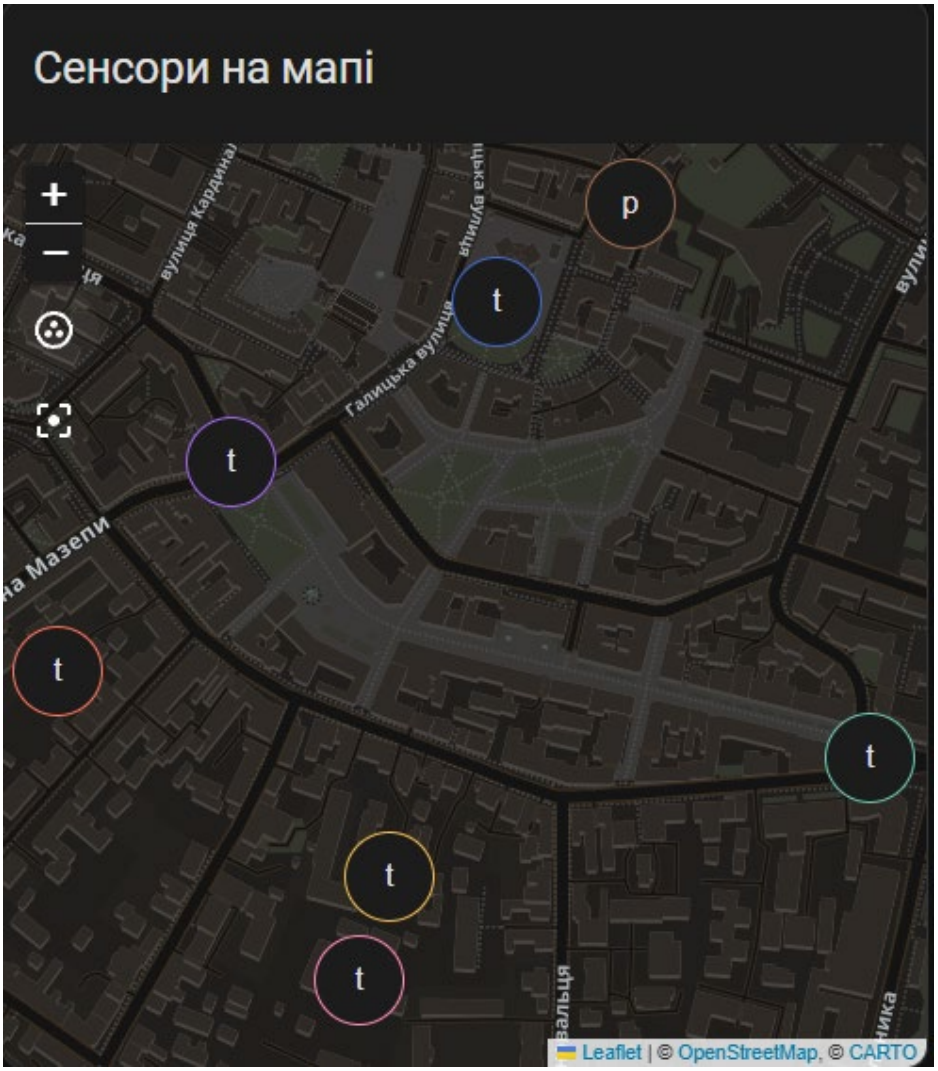

Рисунок 3.11 – Карта з місцеперебуванням сенсорів

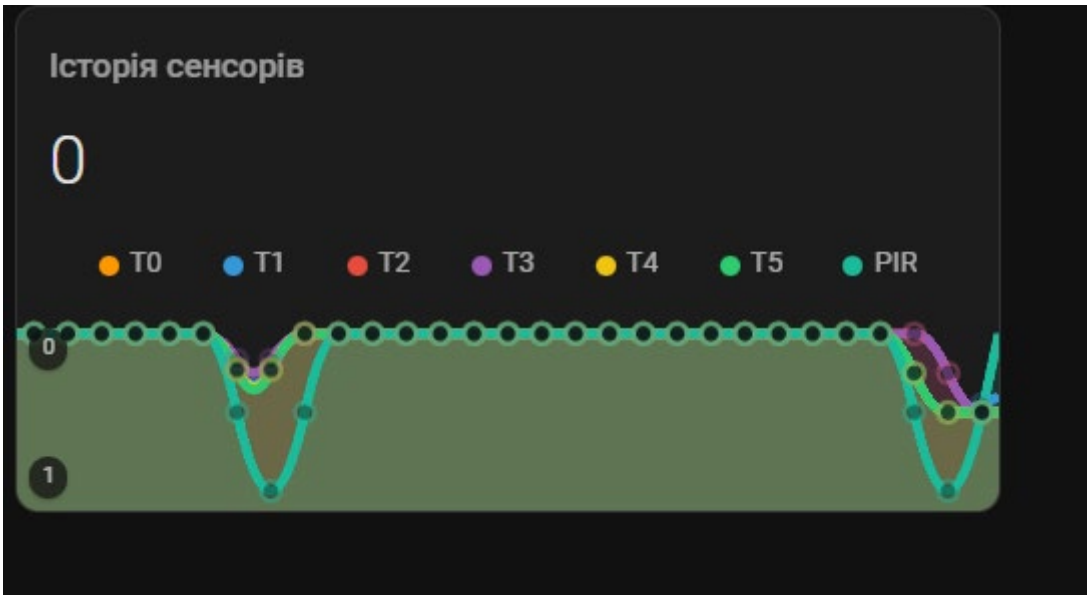

Рисунок 3.12 – Графік активації сенсорів

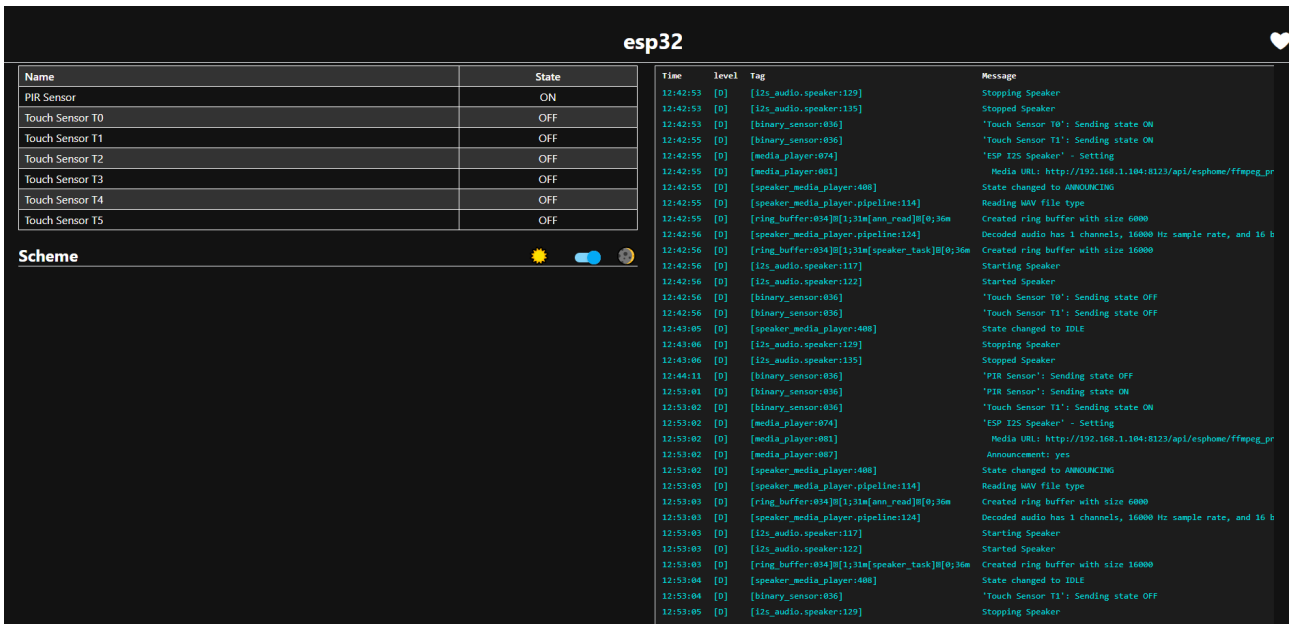

Рисунок 3.13 – Вебсторінка esp32 з усією історією

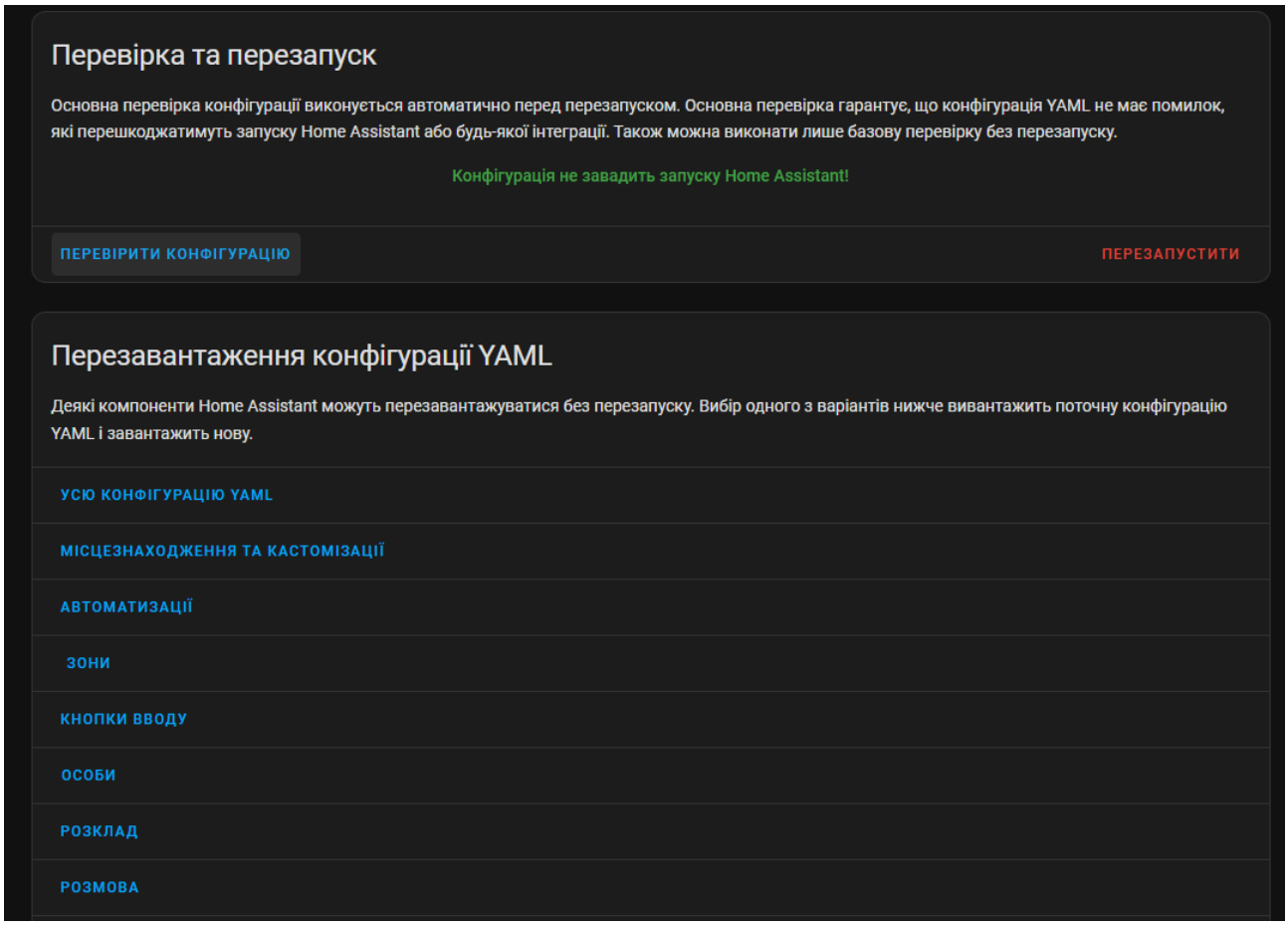

Рисунок 3.14 – Перевірка конфігурації засобами Home Assistant

Проведене тестування підтвердило функціональну готовність розробленої системи до експлуатації відповідно до поставлених вимог. У результаті перевірки було встановлено, що всі сутності коректно відображають свій стан у середовищі Home Assistant, розроблені автоматизації виконуються згідно з логікою, а події від сенсорів адекватно візуалізуються в реальному часі.

Зокрема, система успішно реагує на зміну стану вхідних сигналів, виводячи їх у двостановому режимі "on/off", що свідчить про правильне налаштування сенсорів. Автоматизації виконуються стабільно, з чіткою відповідністю запрограмованим сценаріям. При активації сенсора подія миттєво відображається на карті та автоматично зникає через заданий інтервал часу, що підтверджує коректну реалізацію часової логіки.

Окремо варто відзначити візуалізацію історії активацій сенсорів, що дозволяє аналізувати поведінку системи у динаміці та, за потреби, оптимізувати сценарії. Усі конфігураційні файли було успішно зкомпільовано без помилок, що засвідчує цілісність і коректність програмного забезпечення.

Таким чином, за результатами тестування встановлено, що розроблена система є працездатною, стабільною та готовою до подальшого розгортання або масштабування. Виявлених критичних помилок не зафіксовано, що дозволяє рекомендувати систему для подальшого використання в реальних умовах.

ВИСНОВКИ

У межах виконання бакалаврської кваліфікаційної роботи було розроблено та досліджено автоматизовану систему обробки інформації на базі мікроконтролера ESP32 з інтеграцією до платформи Home Assistant та хмарного сервісу Google Text-to-Speech.

На основі проведеного аналізу сучасних апаратно-програмних засобів та огляду ринку аналогічних рішень було обґрунтовано доцільність використання відкритої платформи ESP32 для реалізації автоматизованих систем, орієнтованих на локальну обробку даних і розширену автоматизацію. Було визначено, що поєднання ESP32 з Home Assistant забезпечує високу гнучкість, масштабованість, незалежність від хмарних інфраструктур та можливість реалізації користувацької логіки з мінімальними ресурсами.

У результаті роботи було:

- Проведено системний аналіз науково-технічної літератури та методів обробки інформації.
- Розроблено функціональну структуру автоматизованої системи та виконано технічне моделювання взаємодії компонентів.
- Впроваджено інтеграцію сенсорів та виконавчих пристроїв у середовищі Home Assistant.
- Забезпечено звукове оповіщення про події через Google Text-to-Speech.
- Реалізовано сценарії автоматизації з використанням ESPHome та сенсорного керування.
- Проведено тестування системи з оцінкою точності спрацювань, стабільності передачі даних, надійності роботи прошивки та відображення інформації у візуальних компонентах Home Assistant.

Запропонована система має практичне значення: вона може бути використана у побутових умовах, в навчальних демонстраційних цілях, а також як основа для створення складніших IoT-рішень. Використання відкритих технологій дозволяє розширювати функціонал системи шляхом додавання нових датчиків, сервісів або автоматизацій, а локальна обробка підвищує рівень конфіденційності.

Очікуваний ефект від реалізації полягає у підвищенні рівня автоматизації, зменшенні залежності від хмарних сервісів, економії ресурсів при збереженні функціональності, а також покращенні користувацького досвіду шляхом голосового інформування. Подальші дослідження можуть бути зосереджені на інтеграції інших хмарних сервісі, підтримці додаткових сенсорів та створенні адаптивної системи навчання поведінки користувача на основі аналізу подій.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Погорілко В. Ф. Інформація [Електронний ресурс] / В. Ф. Погорілко, Т. Я. Любива // Енциклопедія Сучасної України. – Режим доступу: https://esu.com.ua/article-12485

2. Сучасні інформаційні системи та технології: матеріали III Всеукраїнської науково-практичної інтернет-конференції студентів, аспірантів та молодих вчених за тематикою: «Сучасні комп'ютерні системи та мережі в управлінні», 30 листопада 2020 р., м. Херсон / за ред. Г.О. Райко. – Херсон: ФОП Вишемирський В. С., 2020. – 312 с. – ISBN 978-617-7783-98-4. – [Електронне видання].

3.Шеннон, К. Е. Математична теорія зв'язку [Електронний ресурс] / Клод Е. Шеннон // Bell System Technical Journal. – 1948. – Т. 27, № 3. – С. 379–423. – Режим доступу: https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf

4. Certain Factors Affecting Telegraph Speed" / H. Nyquist // Bell System Technical Journal. – 1924. – Vol. 3, No. 2. – P. 324–346.

5. "Transmission of Information" / R. V. L. Hartley // Bell System Technical Journal. – 1928. – Vol. 7, No. 3. – P. 535–563.

6.Литовченко О. Створення системи комунікації peer-to-peer на базі мікроконтролерів ESP32 для домашньої автоматизованої мережі SmartHome [Електронний ресурс] / О. Литовченко // Науково-технічна бібліотека КПІ ім. Ігоря Сікорського. – Режим доступу: https://repository.kpi.ua/handle/123456789/98765,

7. Barral, V., Campos, O., Dominguez-Bolano, T., & Escudero, C. J. (2022). Fine time measurement for the Internet of Things: A practical approach using ESP32. ResearchGate.

8. Hayashi, T., Yamamoto, R., Inoue, K., Watanabe, S., Toda, T., Takeda, K., Zhang, Y. ESPnet2-TTS: Extending the Edge of TTS Research [Електронний ресурс] //

arXiv preprint. – 2021. – № arXiv:2110.07840. – 18 с. – Режим доступу: https://arxiv.org/pdf/2110.07840

9. Hayashi, T., Wang, Y., Takamichi, S., Uenoyama, K., Zhang, Y., Watanabe, S., Toda, T. ESPnet-TTS: Unified, Reproducible, and Integratable Open Source End-to-End Text-to-Speech Toolkit [Електронний ресурс] // arXiv preprint. – 2019. – № arXiv:1910.10909. – 8 с. – Режим доступу: https://arxiv.org/pdf/1910.10909

10. Home Assistant [Електронний ресурс] // Home Assistant. – Режим доступу: https://www.home-assistant.io/#:~:text=Awaken%20your%20home.

11. openHAB [Електронний ресурс] // openHAB. – Режим доступу: https://www.openhab.org/

12. Contributors to Wikimedia projects. Node-RED - Wikipedia [Електронний ресурс] / Contributors to Wikimedia projects // Wikipedia, the free encyclopedia. – Режим доступу: https://en.wikipedia.org/wiki/Node-RED#:~:text=Node,3

13. Instructables. Uber Home Automation W/ Arduino & Pi [Електронний ресурс] / Instructables // Instructables. – Режим доступу: https://www.instructables.com/Uber-Home-Automation-w-Arduino-Pi/#:~:text=This%20Arduino%20sensor%20can%20be,your%20OpenHAB%20smart%20phone%20app

14. Frigate NVR [Електронний ресурс] // Frigate NVR. – Режим доступу: https://frigate.video/

15.Amazon**.** Alexa and Alexa Device Terms of Use [Електронний ресурс] // Amazon Help. – Режим доступу: https://www.amazon.com/gp/help/customer/display.html?nodeId=GSR22RYDWS3KBUYW

16. Aqara FP1 Human Presence Sensor Review [Електронний ресурс] // SmartHomeScene. – Режим доступу: https://smarthomescene.com/reviews/aqara-human-presence-sensor-fp1-review/#:~:text=handles%20motion%20and%20presence%20with,electromagnetic%20waves,%20called%20millimetre%20waves

17.SONOFF Zigbee Human Presence Sensor | SNZB-06P [Електронний ресурс] // ITEAD STUDIO OFFICIAL. – Режим доступу: https://itead.cc/product/sonoff-zigbee-human-presence-sensor/?srsltid=AfmBOorLOu_v4vbJGPpjx8cHqepyIjdrpG_vEVEXo6c9AUfo9nDMnXbb#:~:text=

18. Meross Matter Smart Presence Sensor, MS600 [Електронний ресурс] // Meross Official Store. – Режим доступу: https://shop.meross.com/products/smart-presence-sensor-ms600?srsltid=AfmBOoqS-0q5YusDKRnupH6APiSuWKY7Ca_ZKnM_3htKUbJblU1Tmb5Q#:~:text=*%20Multi,it%20doesn’t%20record%20images%20or

19. ESP32 Wi-Fi & Bluetooth SoC | Espressif Systems [Електронний ресурс] // Wireless SoCs, Software, Cloud and AIoT Solutions | Espressif Systems. – Режим доступу: https://www.espressif.com/en/products/socs/esp32

20. User Guide for ESP32-DevKitM-1 [Електронний ресурс]. – Режим доступу: https://docs.espressif.com/projects/esp-idf/en/v4.3/esp32/hw-reference/esp32/user-guide-devkitm-1.html –

21. ESP Product Selector [Електронний ресурс] // ESP Product Selector. – Режим доступу: https://products.espressif.com/#/product-selector?language=en&names=.

22. Учасники проектів Вікімедіа. Клієнт-серверна архітектура – Вікіпедія [Електронний ресурс] / Учасники проектів Вікімедіа // Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура.

23. ESPHome – ESPHome [Електронний ресурс] // ESPHome. – Режим доступу: https://esphome.io/

24. Home Assistant Green [Електронний ресурс] // Home Assistant. – Режим доступу: https://www.home-assistant.io/green/

25. Raspberry Pi [Електронний ресурс] // Home Assistant. – Режим доступу: https://www.home-assistant.io/installation/raspberrypi

26. Raspberry Pi Documentation [Електронний ресурс] // Raspberry Pi Foundation. — Режим доступу: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#power-supply

27.Windows [Електронний ресурс] // Home Assistant. – Режим доступу: https://www.home-assistant.io/installation/windows

28. Sadeeq M. A., Zeebaree S. R. M., Jacksi K. Security in Internet of Things: A Review [Електронний ресурс] / M. A. Sadeeq, S. R. M. Zeebaree, K. Jacksi // ResearchGate. – 2022. – Режим доступу: https://www.researchgate.net/publication/363879308_Security_in_Internet_of_Things_A_Review

29. Review of Embedded Systems Security [Електронний ресурс] / A. Kumar, M. Verma, A. Sharma // ResearchGate. – 2020. – Режим доступу: https://www.researchgate.net/publication/346657646_Review_of_embedded_systems_security

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: «Розроблення автоматизованої системи обробки інформації на базі платформи ESP-32»

Обсяг пояснювальної записки в аркушах – 73.

Перелік графічного матеріалу:

КРБ.СІ-12.00.001 Е1 Комунікаційне середовище Home Assistant. Схема структурна (аркушів – 1).

КРБ.СІ–12.00.00.001 Е2 Міжблокова схема проєкту (аркушів – 1). Схема функціональна.

Дата закінчення бакалаврської роботи « 20 » червня 2025р.

Студент Нагірняк А. Д

					КРБ.СІ-12.00.001 Е1			
Змн.	Лист	№ докум.	Підпис	Дата		Літ.	Маса	Масштаб
Розроб.		Нагірняк А.Д.			*Комунікаційне середовище Home Assistant* *Схема структурна*			
Перевір.		Паньків Х. В.				Арк. 1	Аркушів 1	
Т. Контр.								
Реценз.								
Н. Контр.		Возний А. В.				ІФНТУНГ СІ-21-1		
Затверд.		Заміховський Л. М.						

КРБ.СІ-02.00.001 Е1

КРБ.СІ-12.00.002 Е2

Змн.	Лист	№ докум.	Підпис	Дата	*Міжблокова схема проєкту. Схема функціональна*	Літ.	Маса	Масштаб
Розроб.		Нагірняк А. Д.						
Перевір.		Паньків Х. В.				Арк. 1	Аркушів 1	
Т. Контр.								
Реценз.								
Н. Контр.		Возний А. В.				ІФНТУНГ СІ-21-1		
Затверд.		Заміховський Л. М.						