КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультету радіофізики, електроніки та комп'ютерних систем
Кафедра комп'ютерної інженерії

Програмно-апаратний комплекс автоматизації "розумного дому" на основі Інтернету речей

Дипломна робота магістра
студента 2 року навчання
Спеціальність: 123 «Комп'ютерна інженерія»
Олексія ЯРОША

Науковий керівник
Євген СЛЮСАР,
асистент кафедри комп'ютерної інженерії

Рецензент
Віктор ВОЛОХОВ
кандидат фіз.-мат. наук
доцент кафедри теорії та технології програмування
факультету комп'ютерних наук та кібернетики

До захисту допускаю: Завідувач кафедрою
Юрій БОЙКО

Ухвалено на засіданні кафедри "____"_____2022 р., протокол № ____

Київ 2022

РЕФЕРАТ

Обсяг роботи 55 сторінки, 21 ілюстрація, 3 схеми, 13 джерел посилання.

РОЗУМНИЙ ДІМ, АВТОМАТИЗАЦІЯ, ІНТЕРНЕТ РЕЧЕЙ, ДИСТАНЦІЙНЕ КЕРУВАННЯ, HOME ASSISTANT, ВЕБ ІНТЕРФЕЙС, ТРИГЕР.

Робота містить методичні вказівки щодо вибору платформи, її встановлення, налаштування, та створення базових сценаріїв автоматизації. Розглянуто процес інтеграції різних пристроїв, в тому числі власних DYI розробок.

Метою роботи є створення самодостатньої автоматизованої системи керування IoT пристроями, незалежно від їх архітектури, типу та виробника, яка дозволить створити єдиний інтерфейс керування, та об’єднати їх в спільних сценаріях автоматизації.

Методи дослідження включають як теоретичні: аналіз ринку, класифікація рішень, порівняння та вибір; так і емпіричні: ручне встановлення платформи та налаштування сумісної роботи з іншими пристроями. Інструментами розробки є безкоштовне програмне забезпечення VMWare Workstation, мова серіалізації даних YAML.

В роботі проведено дослідження ринку, аналіз та класифікація доступних рішень, та вибір найбільш оптимального з них. Було розгорнуто платформу Home Assistant декількома способами та розглянуто різні принципи налаштування. Було інтегровано в систему IoT пристрої, в тому числі DYI, та створено базовий сценарій автоматизації.

ЗМІСТ

ВСТУП

Неможливо уявити сучасний світ без використання технологій, які б полегшували життя людям. Одним з напрямків є технологія розумного дому. Вона дозволяє автоматизувати багато аспектів, що не лише збереже додатковий час, а й зменшить використання енергоресурсів, підвищить безпеку в будинку, і дозволить власникам сконцентруватися на власних справах.

Технологія зародилася в далеких 1960-х роках, і сьогодні є дуже популярним напрямком, оскільки тенденції розвитку суспільства спрямовані на автоматизацію рутинних процесів та економію енергоресурсів. На жаль, навіть через пів століття вона залишається недоступною для багатьох через технічну складність або високу загальну вартість. Натомість завдяки безкоштовним рішенням стало можливим самостійне впровадження технології без глибоких знань в області комп’ютерної інженерії.

Через боротьбу корпорацій на ринку IoT за лідерство, кількість та різноманіття доступних рішень лише поглиблює проблему вибору необхідної платформи. Методичні вказівки в цій роботі спрямовані на аналіз доступних рішень, їх класифікацію та порівняння між собою.

Метою роботи є створення рішення, яке об’єднало б всі IoT пристрої, незалежно від типу, архітектури чи виробника, в єдину систему, яка дозволить їм працювати разом.

Дана робота може бути розглянута як набір методичних вказівок щодо повного циклу впровадження: від вибору системи і платформи для неї, до реалізацію базового сценарію та інтеграції власних DYI пристроїв.

Розділ 1. Концепції та архітектура платформи "Розумного дому". Аналіз та класифікація доступних рішень.

Розумний дім — це технологія, яка спрямована на підвищення комфорту людини, забезпечення ефективного використання ресурсів та контроль за безпекою будинку. Зазвичай розумний будинок містить багато інших функцій, такі як віддалене керування приладами та пристроями з будь-якої точки, де є підключення до Інтернету, за допомогою мобільного або іншого пристрою; автоматичне керування освітлення базуючись на присутності людини в кімнаті, і т.д. Пристрої в розумному домі з'єднані між собою через мережу Інтернет, або за допомогою інших технологій передачі даних, що дозволяє користувачеві дистанційно керувати такими функціями, як контроль доступу до дому, температура, освітлення та різні мультимедійні технології.

1.1 Загальна концепція. Принцип роботи та необхідні складові

Розумний дім означає використання технічних систем, автоматизованих процесів та дистанційно керованих пристроїв у квартирах та будинках. Основна мета функцій – покращити якість життя та комфорт у домі. Іншими цілями є підвищення безпеки та ефективніше використання енергії завдяки підключеним пристроям з дистанційним керуванням Розумний дім реалізує наступні функції:

- Дозволяє власникам будинків дистанційно керувати приладами, термостатами, освітленням та іншими пристроями за допомогою смартфона або планшета через Інтернет.
- Розумні будинки можна налаштувати за допомогою бездротових або провідних систем.
- Технологія «Розумний будинок» забезпечує власникам будинків зручність та економію коштів.

Деякі системи домашньої автоматизації сповіщають власника будинку, якщо в будинку виявлено будь-який рух, коли власник відсутній, а інші

можуть викликати органи безпеки — поліцію або пожежну службу — у разі наявності необхідних ознак ситуації.

Загалом не існує єдиного загальновизнаного принципу роботи розумного дому, оскільки технологія залежить від бюджету, компанії та типу автоматизації. Різні підходи пропонують різні концепції керування розумним домом, як от використання панелі центрального керування, повна інтеграція функцій керування в смартфон, смарт-телевізор, або ж взагалі відмова від керування і побудова технології цілком навколо нейронних мереж, які будуть автоматично підлаштовуватися під стиль життя мешканців. Часто ці підходи використовуються паралельно, і доповнюють одна одну.

Панель центрального керування - це одна з систем, яка керує всіма пристроями у домі. Це дозволяє з одного пристрою керувати освітленням, термостатом, кондиціонером, та іншою розумною технікою. Цей тип домашньої автоматизації найбільш популярний серед ринкових рішень як для приватних будинків, так і для офісів. Зазвичай системи центрального керування запускаються через настінний термінал (планшет), як той, який зазвичай використовується для систем безпеки будинку. Перевагами такого підходу є доступ до всіх аспектів будинку з однієї головної системи, яка дозволяє керувати всім, починаючи від душової кабіни і закінчуючи мережею домашньої безпеки.

Натомість недоліками як правило є вартість, оскільки вони вимагають професійного встановлення. Інша проблема такого підходу – обмеженість інтеграції нових пристроїв. Тобто вибір розумної техніки (наприклад пральна машина) буде відбуватися лише в рамках пристроїв які працюють конкретно з цією системою.

Підхід розумного дому в смартфоні – це використання додатків для керування різними аспектами через хмарні технології. Вони є важливою частиною інтернету речей, і стали дуже популярними за останні кілька років. Більшість розумних пристроїв на основі додатків працюють, підключаючись

до домашньої мережі через Wi-Fi. Ці пристрої підключаються до зовнішнього сервера, до якого потім надається доступ через програми на смартфоні. Це дозволяє керувати пристроями та отримувати дані з сенсорів в домі з будь-якого місця. Деякі технології домашньої автоматизації працюють через Bluetooth, і в такому випадку необхідно перебувати поблизу цих пристроїв для керування ними.

Перевагами такого підходу є доступність розумних будинків для широких верств населення, що дозволяє автоматизувати все власними руками, тому що ці пристрої доступні, прості в налаштуванні, використанні та оновленні. Використання такого підходу вважається досить бюджетним і універсальним, що дає змогу реалізовувати різні ідеї та функції.

Натомість до недоліків варто віднести те, що більшість рішень вимагає кількох різних додатків для керування, що є не зовсім зручно. Також робота через віддалені сервери призводить до збільшення часу відгуку, і залежності від якості мережі інтернет. Якщо інтернет з’єднання зникне, розумний дім фактично стане некерованим. Існують варіанти локального керування, але зазвичай вони не закладені виробником у стандартні функції додатку, і вимагають технічних знань для їх активації.

1.2 Технології передачі даних та інструкцій. Робота мережі.

Першим галузевим стандартом для домашньої автоматизації був X10. Це дротовий стандарт, який використовує звичайну домашню електричну мережу для передачі закодованих цифрових даних. За допомогою радіочастотного імпульсу частотою 120 кГц та тривалістю 1 мс керуюючий сигнал надсилається в момент переходу змінного струму через нульове значення. За цей час передається один біт даних. Приймач формує вікно очікування тривалістю 200 мкс. Таким чином за наявності імпульсу сигнал інтерпритується як – логічна «1», а відсутність означає логічний «0». Варто згадати також технологію безпровідної передачі даних на частоті 433 Мгц, яка використовується сьогодні в безпровідному варіанті стандарту X10 [1]. 433 є

частиною радіочастотного спектру ISM, тобто це спектр загального призначення, який можна не ліцензувати. Зазвичай на цій частоті часто працюють зовнішні пристрої, такі як шлагбауми та гаражні ворота. Його перевагою є дуже низька ціна пристроїв, але недоліками є погана пропускна здатність і відсутність систем безпеки зв'язку. Крім того, неможливо створити сітку пристроїв, що може бути проблемою з великою площею нерухомості - кожен пристрій повинен знаходитися в прямому діапазоні, щоб мати можливість отримувати команди.

Інший стандарт бездротового зв'язку який часто використовуються в системах розумного будинку – це Zigbee. Працює в діапазоні 2,4 ГГц, для передачі даних використовується шифрування, а пристрої можна об’єднати в мережу. Крім того обладнання, яке працює по протоколу zigbee, сьогодні дуже широко поширене і тому має невелику вартість.

Загалом мережа Zigbee містить три різні типи пристроїв: координатор, маршрутизатори та кінцеві пристрої. Ці пристрої працюють один з одним для доставки інструкцій від координатора до кінцевих пристроїв. Координатор використовує канал з найменшою кількістю перешкод для передачі даних, призначає унікальний ідентифікатор мережі та унікальні адреси кожному пристрою в мережі, ініціює та передає команди. Маршрутизатори знаходяться між координатором і кінцевими пристроями і відповідають за доставку повідомлень. Кінцеві пристрої в свою чергу контролюють дуже малу кількість інформації а також часто переходять у режим очікування, що значно зменшує рівень енергоспоживання, і дозволяє працювати на одному заряді батареї більше року [2].

Недоліком протоколу zigbee є менший радіус дії та потенційні перешкоди для інших пристроїв, які використовують ту ж частоту 2,4 ГГц, тобто Bluetooth та Wi-Fi. Також для встановлення кінцевих пристроїв необхідний хоча б один координатор, який буде надсилати керуючі сигнали.

WiFi мережі, в свою чергу, широко поширені і знайомі більшості. Їх робота максимально автоматизована і не вимагає глибоких технічних знань. До того ж WiFi впровадженні в більшості домогосподарств, і для впровадження технологій розумного дому буде лише розширенням наявної інфраструктури, а не створенням з нуля. Також варто відзначити гнучкість налаштувань і контролю за мережею.

Натомість пристрої які працюють по WiFi споживають набагато більше енергії, ніж інші, а також за великої кількості пристроїв завантаженість каналів буде значно більшою, що може спровокувати падіння пропускної здатності для всіх інших клієнтів.

Також є варіант роботи розумних пристроїв за допомогою Bluetooth, але зазвичай він використовується лише для портативних колонок, портативних пристроїв керування та хабів. Перевагами роботи з Bluetooth дуже низьке енергоспоживання, що завдячує технології BLE (Bluetooth Low Energy). Проте для роботи з такими пристроями необхідно постійно активне Bluetooth з'єднання, або ж спеціальний хаб, який буде працювати у ролі контролера.

І ще один поширений стандарт зв'язку для розумного дому це так звана технологія z-wave, яка працює на частоті 868,42 МГц. Перевагами є більший радіус дії (порівняно з zigbee), низьке споживання енергії (порівняно з wifi), зашифроване з'єднання та можливість підключення пристроїв до мережі. Зазвичай z-wave використовується в ринкових рішеннях компаній, які займаються повною інтеграцією розумного будинку, оскільки для коректної роботи необхідне спеціальне обладнання та правильне налаштування. Також такі пристрої дещо дорожчі за конкурентів.

1.3 Архітектура мережі. Принцип побудови системи.

Хоч архітектура мережі залежить від конкретної платформи, але в загальному випадку все можна звести до певних стандартних наборів:

- Хаб-контролер, який є головним мозком всього розумного будинку, і відповідає за доступність інших пристроїв в мережі, керування ними та збір показників. В ролі такого хабу може виступати як спеціальний пристрій, звичайний комп’ютер і віртуальна машина, так і віддалений сервер;
- Датчики та сенсори, функція яких лише збір та надсилання показників, даних та ін. Вони встановлюють з’єднання з основним контролером, і зазвичай використовуються для контролю за мікрокліматом та в автоматизаціях;
- Пристрої, що виконують певні дії, як от реле, перемикачі, електроролети та ін. Їх функція полягає в автоматичній зміні параметрів в будинку, таких як освітленість, температура, та робота різних побутових пристроїв. Головна задача цих пристроїв в мінімізації ручного керування;
- Розумна побутова техніка – окремий розділ пристроїв, які інтегруються в існуючу архітектуру, і також з’єднуються з головним хабом для надсилання статусу, та обміну керуючими повідомленнями;
- Пристрої, які використовуються для віддаленого керування. Це можуть бути як і інтегровані панелі керування, так і мобільні пристрої, смартфони, телевізори, або також голосове керування.

Схему побудови наведено нижче (рис. 1):

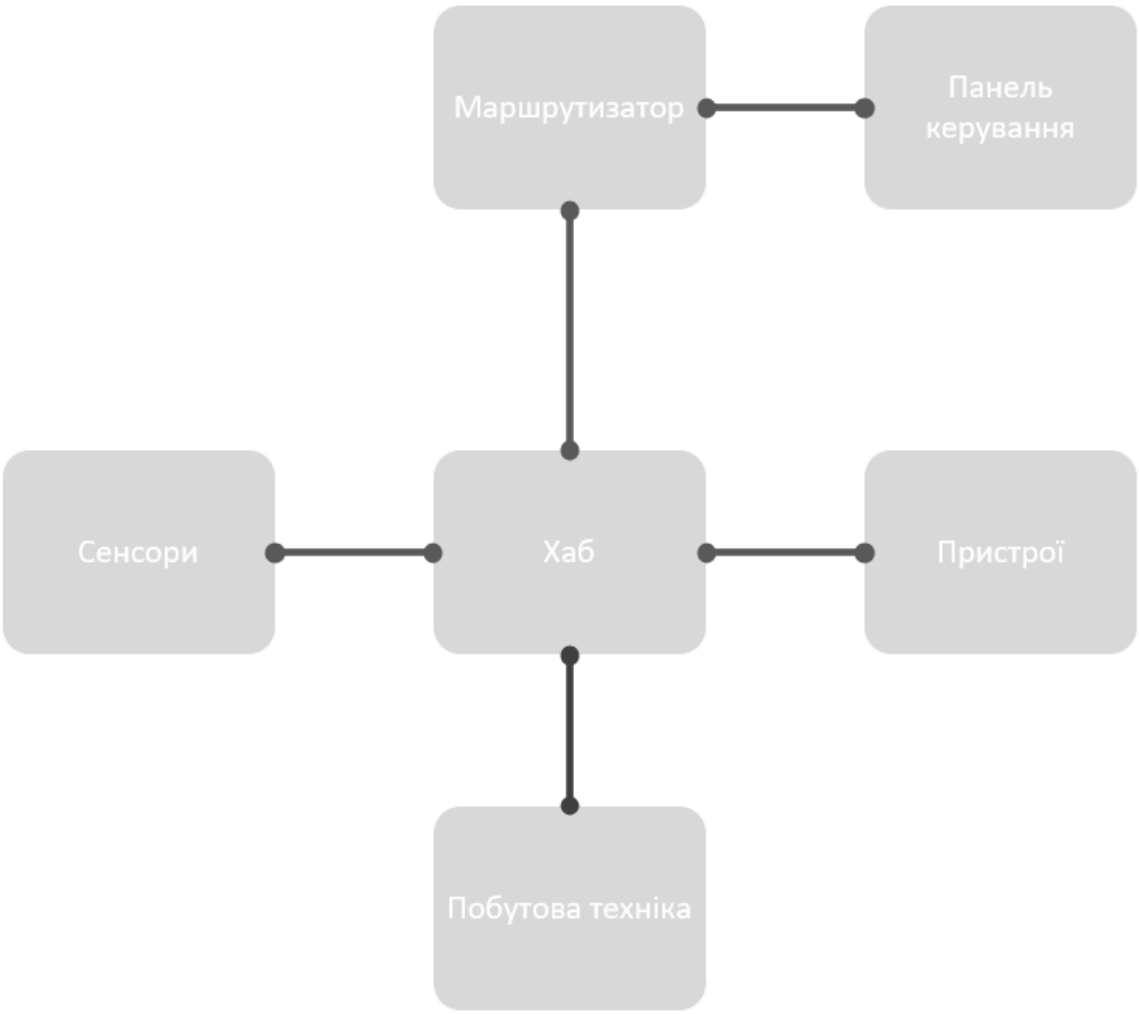

Рисунок 1

Розумний будинок реалізований на основі хмарних рішень буде мати іншу архітектуру, оскільки в ролі хаба буде виступати віддалений сервер, і всі дані будуть контролюватися ним (рис. 2):

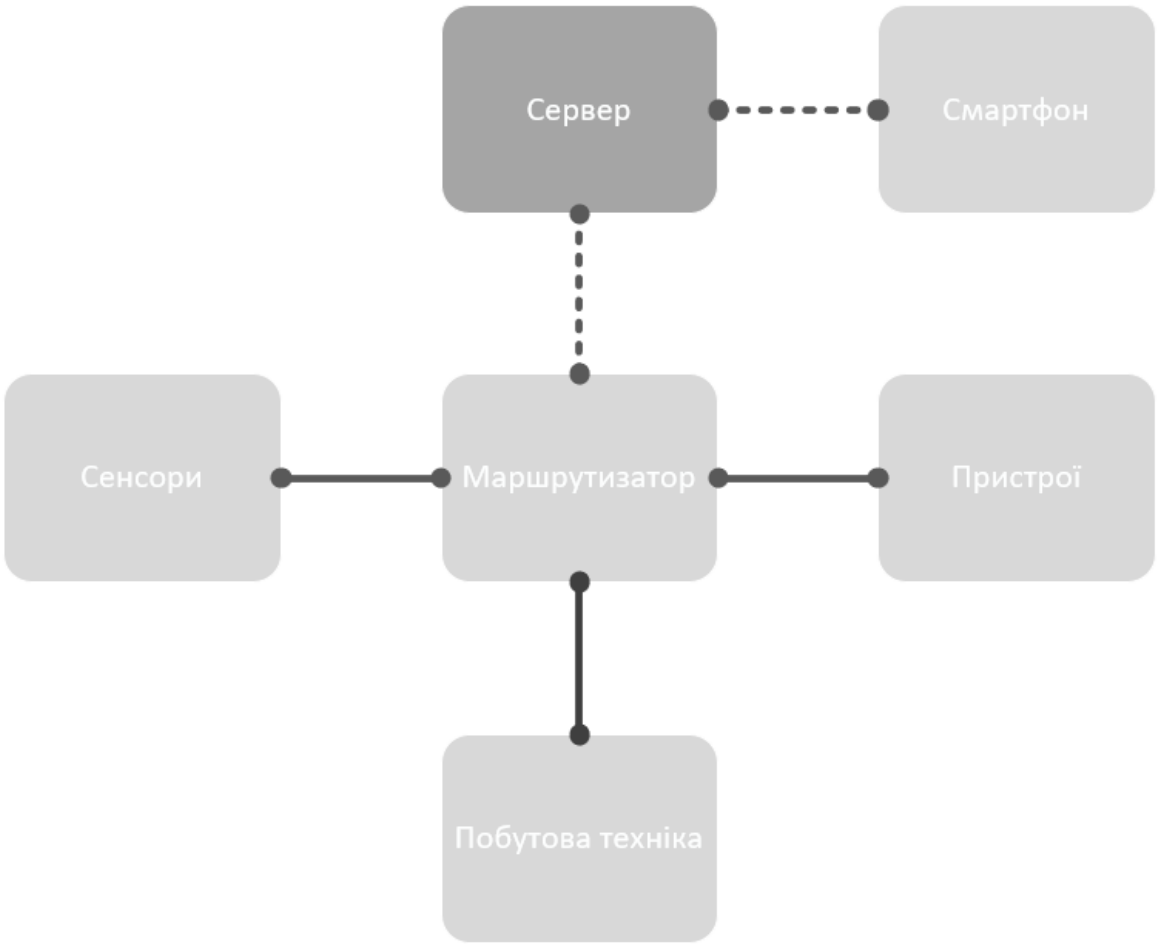

Рисунок 2

1.4 Огляд доступних платформ. Аналіз та класифікація доступних рішень.

Вибір платформи, для подальшого впровадження розумного дому, є одним з головних виборів на етапі планування, оскільки від нього залежить подальший розвиток всієї мережі, можливість інтеграції тих чи інших компонентів, залежність від конкретного виробника та ін. Окремі системи на основі різних виробників мають свої особливості та основні сфери застосування, які можуть як обмежувати функціональні можливості, так і розширювати їх.

1.4.1 Закриті системи. Платформи орієнтовані на власний продукт.

Mi Home / Aqara – два незалежних підрозділи компанії Xiaomi, які відповідають за розробку рішень для розумного будинку. Кожен з них має

широкий вибір різноманітних датчиків і пристроїв, які можуть працювати як через окремий спеціальний хаб, так через віддалені сервери. Основний протокол для взаємодії це Zigbee, Bluetooth та WiFi. Загалом всі функції та автоматизації налаштовуються через мобільний додаток, що робить екосистему закриту для власних змін та налаштувань, тобто весь функціонал обмежений лінійкою доступних пристроїв, та функцій, які були закладені виробником. Таку саму політику підтримують й інші розробники технологій розумного дому, як Tuya, Sonoff, NOUS, Ajax та ін.

HomeKit / Google Home – системно орієнтовні платформи від розробників операційних систем Apple IOS та Android відповідно, які мають повний функціонал систем розумного дому, таких як керування освітленням, мікрокліматом, безпекою та ін. Мають окремі, створені під свою конкретну платформу, пристрої, але на відміну від вище розглянутих розробників також підтримують інтеграцію пристроїв інших виробників. Це відбувається завдяки доступному API, який можна використати для надання доступу до смарт-девайсів з додатків HomeKit та Google Home. Завдяки цьому ці платформи є більш універсальним рішенням, які дозволяють створити більше можливостей автоматизації. Також ці платформи інтегровані в системи голосового керування, і дозволяють керувати розумним домом лише за допомогою голосових команд. Проте їх робота все так же базується на зв’язку з зовнішніми серверами, що робить ці екосистеми закриті для власних змін та налаштувань, а також залежною від інтернет з’єднання.

Окремо варто зазначити платформи виробників побутової техніки. До таких платформ можна віднести LG ThinQ, SmartThings, Home Connect та ін. Це високобюджетні проекти, які в першу чергу орієнтовані на власну продукцію. Системи, які працюють через WiFi, керуються зі спеціального додатку на смартфоні і дозволяють відслідковувати статус побутової техніки, та віддалено керувати нею. Всі платформи працюють подібно – через віддалені сервери, проте мають різне бачення розвитку платформ.

LG ThinQ – орієнтована лише на власну продукцію, та виключає будь-яку інтеграцію зі сторонніми виробниками. Вона підтримує взаємодію із платформами операційних систем смартфонів, тобто Google Home та HomeKit [3], що дозволяє використовувати голосових помічників для керування приладами. Проте оскільки екосистема закрита, а вся взаємодія відбувається через зовнішні сервери, весь функціонал обмежений можливостями закладеними виробником.

SmartThings – це підрозділ корпорації Samsung, який займається напрямком інтернету речей. Хоч ця платформа і є частиною Samsung, проте політика розвитку відрізняється від системи LG ThinQ: основна перевага надається власним побутовим пристроям, проте система дозволяє інтеграцію з іншими виробниками, які працюють з відкритим API SmartThings. Ці інтеграції включають широкий спектр різних сенсорів та пристроїв від різних виробників [4]. Це дозволяє створити мультиплатформу для керування багатьма функціями розумного дому.

Home Connect від Bosch є певним компромісом між двома попередніми конкурентами. З технічного боку платформа працює подібним чином: з'єднання відбувається через віддалені сервери, інтеграції працюють через відкритий API. Проте загальний розвиток відбувається в дещо іншому напрямку: Home Connect об'єднує в собі різних виробників побутової техніки як Siemens, Neff, Gaggenau, Balay, Thermador та ін. Але додатково можна оформити Home Connect Plus, який є справжньою мультиплатформою для десятків різних виробників різних пристроїв – від сенсорів до холодильників, та дверей в гараж, тощо.

1.4.2 Open-source платформи

OpenHab (Open Home Automation Bus) розвивається з 2010 року і на даний момент є повністю готовим продуктом. Сьогодні вже існує друге покоління, яке працює на основі фреймворку Eclipse SmartHome, що покращує можливості для редагування користувацького інтерфейсу. Це проект

написаний на Java із відкритим вихідним кодом, тому він розвивається завдяки спільноті, що дозволяє йому активно збільшувати кількість підтримуваних протоколів та виробників (на даний момент ця цифра 370 [5]).

Концепція OpenHab реалізує єдину шину і взаємодіє між так званими об’єктами (things), каналами (channels) і зв’язками (bindings). Це дозволяє об'єднати всі пристрої з різними протоколами в єдину мережу, абстрагуючи користувача від кожного конкретного протоколу [6]. Таким чином все управління відбувається через шлюз, в ролі якого виступає програмне забезпечення на будь-якій платформі (Linux, Windows, MacOS), віртуальній машині або ж докер контейнеру.

Особливостями платформи є скриптова мова для кофігурацій Xtend, яка є логічним послідовником Java. Для налаштування взаємодії можна скористатися або Web-інтерфейсом, або створювати конфігурації мовою Xtend. Також web-інтерфейс для адміністрування є дуже гнучким, і може бути зміненим під будь-які потреби.

MajorDoMo – концептуально схожий проект на всі інші opensource проекти, проте має деякі важливі відмінності. На сам перед цей проект створювався для пост-радянських країн, тому враховує всі реалії цього ринку, спільнота спілкується знайомою мовою, а платформа підтримує багато необхідних функцій, необхідних саме для українських користувачів.

Проте попри всі переваги ця платформа менш поширена ніж інші, тому відповідно, підтримує менше доступних функцій та виробників (на даний момент загальна кількість трохи більше 160). [7]

Загалом архітектура та принцип роботи схожий на інші платформи, тобто MajorDoMo виступає в ролі центрального серверу, який працює на локальному сервері, і збирає дані та надсилає команди контролерам, які є пропрієтарними пристроями, і в свою чергу керують кінцевими пристроями.

ioBroker позиціонує себе як інтеграційну платформу для Інтернету речей, і дозволяє інтегрувати комерційні продукти практично з усіх сфер

життя, або ж інтегрувати власні DYI рішення. В першу чергу це німецька платформа і має велику спільноту в Німеччині мову, проте документація дублюється також англійською[8].

Система має модульну структуру (рис. 3) і може бути легко масштабована для шляхом встановлення окремих адаптерів. Також ioBroker, як і конкуренти, можна встановити практично на всі апаратні платформи (Raspberry Pi, сервери, NAS, сервери з віртуалізації, настільні комп'ютери, під управлінням Linux, OSX, Windows та Docker)

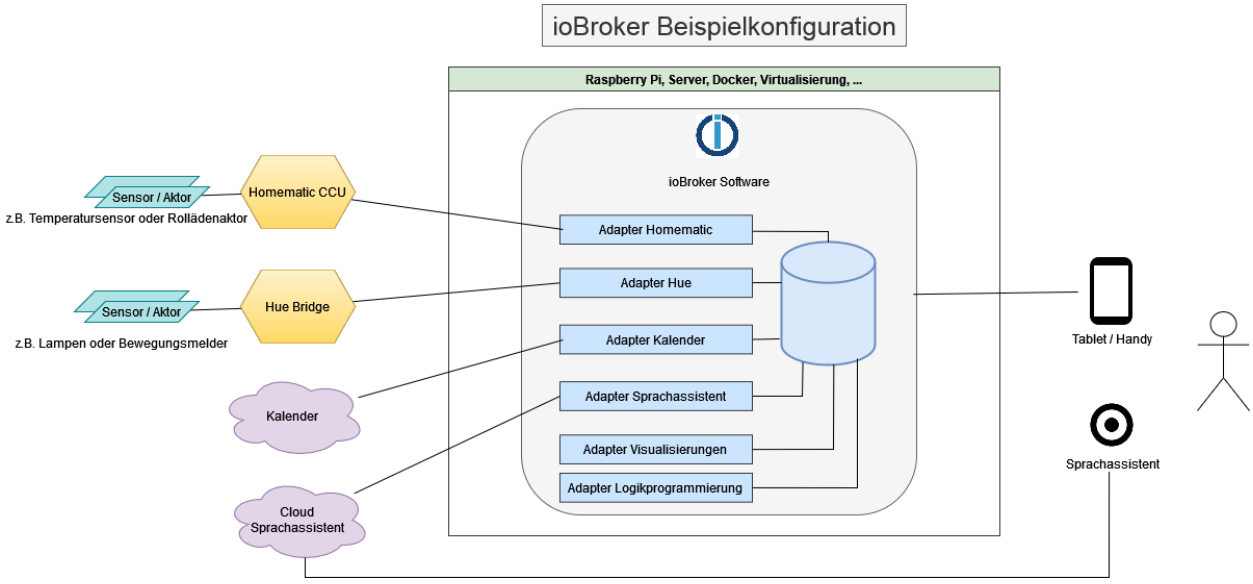

Рисунок 3

Вся робота платформи побудована на так званих адаптерах. На цьому малюнку показано, як ioBroker координує та з'єднує різні системи розумного будинку та інші служби (наприклад, календар). Адаптери, доступні в ioBroker, можуть обмінюватись даними, як ми можемо бачити на малюнку, між центральним календарем та системами розумного будинку конкретних виробників (які підключені до локальної мережі). Таким чином, саме через адаптери ioBroker може зчитувати інформацію про стан або надсилати команди, які в свою чергу можуть залежати від інших адаптерів. На даний момент ioBroker підтримує близько 400 різних адаптерів для роботи з різними

виробниками та іншими функціями [9] Також варто відмітити що система платформа написана мовою JavaScript.

Недоліком ioBroker можна назвати складну інтеграцію та налаштування. Попри складність всіх процесів платформи, ioBroker також має проблеми з документацією, що призводить до втрати часу на пошук необхідної інформації на форумах.

Home Assistant - платформа домашньої автоматизації з відкритим кодом, створена в основному спільнотою ентузіастів, проте сьогодні має окрему фінансову одиницю Nabu Casa яка займається розподілом передплатних сервісів та заробітної плати для постійних працівників.

Подібно конкурентам Home Assistant (далі “HA”) підтримує велику кількість операційних систем та платформ для роботи – від Raspberry PI, локальних комп’ютерів з ОС Windows, Linux, MacOS та ін., до платного хмарного рішення з підтримкою та налаштуванням. Також HA має для продажу стартові набори. Додатково HA реалізує чотири різні методи встановлення[10].

Вся робота HA зосереджена навколо інтеграцій, які є попередньо налаштованими розробником бібліотеками для роботи з IOT пристроями конкретних виробників. Після запуску платформи HA автоматично сканує мережу на наявність розумних пристроїв, і автоматично додає їх разом з функціями керування до веб-інтерфейсу, що значно зменшує поріг входження, та дозволяє розпочати роботу розумного дому з самого початку. Окрім веб-інтерфейсу, який можна використовувати для інтуїтивно-зрозумілого налаштування пристроїв та створення скриптів та автоматизацій, можна створювати та редагувати конфігурації вручну. Платформа написана мовою Python, а всі конфігурації створюються скриптовою мовою Yaml.

1.5 Вибір платформи. Критерії та вимоги

Вибір системи є головною задачею, оскільки саме від цього рішення залежить подальший розвиток розумного дому. Кожна система має як переваги, так і недоліки. В той час як одна з них має зрозумілу документацію, інша дозволяє підключити більшу кількість пристроїв, або ж має велику гнучкість в налаштуванні власних рішень.

Список вимог і критеріїв не є вичерпним, а також є індивідуальним для кожного випадку. Проте можна сформулювати загальні положення, які будуть зрозумілі для недосвідчених користувачів. Одним з головних критеріїв для вибору є гнучкість системи, оскільки будь-який дім з часом змінюється. Власники додають нові пристрої, реалізують нові функції і змінюють наявні. Для того щоб система не перетворилася на непотрібну функцію – вона має змінюватися разом з домом.

- Підтримка максимально доступної кількості виробників
- Проста масштабованість та легка зміна пристроїв
- Повний контроль над системою
- Можливість повністю локальної роботи
- Вбудовані системи захисту та конфіденційності
- Можливість реалізації власних DIY пристроїв
- Гнучкість панелі керування

Через критерії щодо гнучкості та повного контролю над системою всі системи закритого типу відразу були виключені з розгляду. Серед систем з відкритим кодом найбільше відповідали вимогам OpenHab та Home Assistant, проте з огляду на доступність документації, простоту реалізації а також використання знайомої мови YAML вибір був зроблений в бік Home Assistant.

Розділ 2. Побудова платформи автоматизації на основі Home Assistant

2.1 Мова серіалізації даних YAML. Синтаксис.

YAML це основна мова для конфігурацій в HA. Це досить поширена мова, яка також використовується в інших поширених проектах, таких як Docker, Ansible, та ін. За допомогою цієї мови можна виражати складні конфігурації для вирішення різних задач.

Основами синтаксису YAML є колекції блоків і відображення, що містять пари ключ-значення. Кожен елемент у колекції починається з тире «-», тоді як зіставлення мають ключ формату двокрапки «:». Це дещо схоже на хеш-таблицю або, точніше, на словник у Python. Вони також можуть бути вкладеними.

Відступ є важливою частиною визначення підпорядкованості за допомогою YAML. Речі з відступом вкладені «всередині» речей, які знаходяться на один рівень вище. Табуляції не можна використовувати для відступів. Для кожного рівня відступу передбачається використання 2 пробілів. [11]

Приклад YAML синтаксису:

```
- Worker_1:
    name: Joseph
    job: Developer
    skills:
      - python
      - perl
      - pascal
- Worker_2:
    name: Andrii
    job: Engineer
    skills:
      - BGP
      - docsis
      - ipv6
```

Також YAML документи можуть мати змінні середовища. HA Core надає підтримку значення зі змінних середовища системи за допомогою !env_var.

Якщо встановлено інший тип, тоді необхідно використовувати оператор !include:

```
example:
  password: !include PASSWORD
```

Якщо змінна середовища не встановлена, встановити значення за замовчуванням можна за зазначивши значення після функції:

```
example:
  password: !include PASSWORD default_password
```

Також є можливість отримати певні файли з головного конфігураційної директорії за допомогою синтаксису !include:

```
example:
  light: !include lights.yaml
```

Існують різні онлайн-сервіси, такі як YAML Validator, щоб перевірити правильність синтаксису YAML перед завантаженням конфігурації до HA.

2.2 Доступні платформи та операційні системи. Варіанти роботи

HA підтримує велику кількість доступних платформ а також чотири варіанти встановлення:

- Home Assistant Operating System
- Home Assistant Container
- Home Assistant Supervised
- Home Assistant Core

Кожен варіант має свої можливості та функції, які наведені в зображенні 4. Home Assistant Supervised та Core – це розширені варіанти, які встановлюються лише в окремих випадках для спеціальних цілей. Рекомендованим варіантом є Home Assistant Operating System, який оптимізований для роботи з відповідною платформою. Він поставляється

відразу з супервізером для керування основними і додатковими компонентами HA. Home Assistant Container – це автономний Linux-контейнер з HA Core, який контролюється певним менеджером контейнерів (orchestrator), таким як Docker.

	OS	Container	Core	Supervised
Automations	✅	✅	✅	✅
Dashboards	✅	✅	✅	✅
Integrations	✅	✅	✅	✅
Blueprints	✅	✅	✅	✅
Uses container	✅	✅	❌	✅
Supervisor	✅	❌	❌	✅
Add-ons	✅	❌	❌	✅
Backups	✅	✅[1]	✅[1]	✅
Managed OS	✅	❌	❌	❌

Рисунок 4

2.3 Встановлення та запуск. HA Core. HA Operation System.

Першим кроком для початку роботи є запуск та налаштування середовища (операційної системи та всіх залежних сервісів) і подальше встановлення Home Assistant. Існує декілька варіантів платформ та декілька варіантів встановлення, проте рекомендованим є встановлення повноцінної HA Operating System. Для цього є готові збірки під різні середовища: VMware Workstation, VirtualBox та KVM, які значно пришвидшують процес

встановлення. Оскільки сервер має працювати 24/7, популярним варіантом є встановлення на Raspberry PI. Також серед варіантів варто виділити стандартний спосіб встановлення на машину на платформі Linux, Windows, MacOS та інші.

2.3.1 Home Assistant Operating System

Для швидкого розгортання необхідно завантажити готовий образ .vdi, .qcow2 або .vmdk відповідно до середовища з офіційного сайту (https://www.home-assistant.io/installation/linux), і далі завантажити обраний диск в гіпервізор.

Для створення віртуальної машини необхідно вказати мінімальні ресурси:

- 2GB RAM
- 32GB Storage
- 2vCPU

Наступним кроком є вибір операційної системи «Linux» для VirtualBox та VMware. Для KVM необхідно вибрати «Generic Default». Потім вибрати завантажений образ. Далі необхідно вибрати мережевий адаптер типу «міст» (bridge), який дозволить мати доступ до локальної мережі. Якщо встановити інший адаптер, HA не буде доступний з домашньої мережі.

Останнім кроком є вибір UEFI в налаштуваннях материнської плати. Для VMware Workstation цей крок відбувається вже після створення машини.

Після всіх кроків диск буде завантажено як власну операційну систему на базі Linux з встановленими необхідними залежностями та розгорнутим докер контейнером HA. Така система повністю готова для подальшого налаштування. Налаштовувати можна як через операційну систему, так і за допомогою веб-інтерфейсу, який має бути доступний через http://homeassistant:8123

2.3.2 Home Assistant Core

Встановлення на Linux систему окремим пакетом є складнішим методом, і вимагає ручного встановлення всіх залежностей, які можуть різнитися в залежності від дистрибутиву:

- python3
- python3-dev
- python3-venv
- python3-pip
- libffi-dev
- libssl-dev
- libjpeg-dev
- zlib1g-dev
- autoconf
- build-essential
- libopenjp2-7
- libtiff5
- libturbojpeg0-dev
- tzdata

Основним дистрибутивом для HA вибрано Debian. Саме не ньому відбувається тестування. У випадку проблем з HA розробники приймають на розгляд лише випадки при використання останньої версії Debian [12].

Встановити всі залежності можна однією командою:

```
sudo apt-get install -y python3 python3-dev python3-venv python3-pip libffi-dev libssl-dev libjpeg-dev zlib1g-dev autoconf build-essential libopenjp2-7 libtiff5 libturbojpeg0-dev tzdata
```

Після цього створимо користувача homeassistant з домашньою директорією:

```
sudo useradd -rm homeassistant
sudo mkdir /srv/homeassistant
sudo chown homeassistant:homeassistant /srv/homeassistant
```

Наступний крок – це створення/активація віртуального середовища, і встановлення пакету wheel:

```
sudo -u homeassistant -H -s
cd /srv/homeassistant
python3 -m venv .
source bin/activate
python3 -m pip install wheel
```

Отримаємо повідомлення про успішне встановлення:

```
(homeassistant) homeassistant@skywalker:/srv/homeassistant$ python3 -m pip
install wheel
Collecting wheel
  Downloading wheel-0.37.1-py2.py3-none-any.whl (35 kB)
Installing collected packages: wheel
Successfully installed wheel-0.37.1
```

Останній крок – власне встановлення home assistant:

```
pip3 install homeassistant
```

Отримаємо повідомлення про встановлення HA та залежностей:

```
(homeassistant) homeassistant@skywalker:/srv/homeassistant$ pip3 install
homeassistant
...
Successfully installed MarkupSafe-2.1.1 PyJWT-2.3.0 aiohttp-3.8.1 aiosignal-
1.2.0 anyio-3.5.0 astral-2.2 async-timeout-4.0.2 atomicwrites-1.4.0 attrs-
21.2.0 awesomeversion-22.2.0 bcrypt-3.1.7 certifi-2021.10.8 cffi-1.15.0
charset-normalizer-2.0.12 ciso8601-2.2.0 cryptography-35.0.0 frozenlist-1.3.0
h11-0.12.0 homeassistant-2022.4.6 httpcore-0.14.7 httpx-0.22.0 idna-3.3
ifaddr-0.1.7 jinja2-3.1.0 multidict-6.0.2 pip-22.0.4 pycparser-2.21 python-
slugify-4.0.1 pytz-2022.1 pyyaml-6.0 requests-2.27.1 rfc3986-1.5.0 six-1.16.0
sniffio-1.2.0 text-unidecode-1.3 typing-extensions-4.2.0 urllib3-1.26.9
voluptuous-0.12.2 voluptuous-serialize-2.5.0 yarl-1.7.2
```

Тепер система готова для запуску Home Assistant. Зробити це можна за допомогою команди hass. Перший запуск може зайняти від 5 до 10 хв, оскільки будуть створені всі конфігурації за замовчуванням, а також завантажені додаткові залежності.

```
(homeassistant) homeassistant@skywalker:/srv/homeassistant$ hass
Unable to find configuration. Creating default one in
/home/homeassistant/.homeassistant
2022-04-22 04:58:27 WARNING (MainThread) [homeassistant.bootstrap] Waiting on
integrations to complete setup: default_config
2022-04-22 05:58:31 ERROR (MainThread) [homeassistant] Error doing job: Task
exception was never retrieved
```

Для підключення до веб інтерфейсу необхідно перейти за IP адресою машини на порт 8123.

2.4 Огляд функцій. Налаштування. Керування

2.4.1 Веб-інтерфейс

Рекомендований спосіб початкового налаштування – це використання веб-інтерфейсу. Спочатку система запропонує створити обліковий запис власника Home Assistant. Цей обліковий запис буде адміністратором і матиме доступ до всіх функцій системи.

Далі можна ввести назву для свого будинку та встановити своє місцезнаходження та систему одиниць. НА має можливість встановити розташування автоматично, та налаштувати часовий пояс та систему одиниць на основі цього розташування.

Після цих початкових налаштувань системи Home Assistant вже готовий до роботи, і на наступному екрані будуть доступні всі пристрої, які були автоматично виявлені у мережі. В автоматичному режимі не всі пристрої можуть бути виявлені, і деякі потребують ручного налаштування.

Першочергово необхідно ввімкнути розширений режим (рис. 5) для користувача. Це можна зробити у вкладці користувача на панелі керування, та далі знайти “Advanced mode”. Ця функція активує доступ до системних змін, зміни конфігурації а також її перевірки перед перезавантаженням системи.

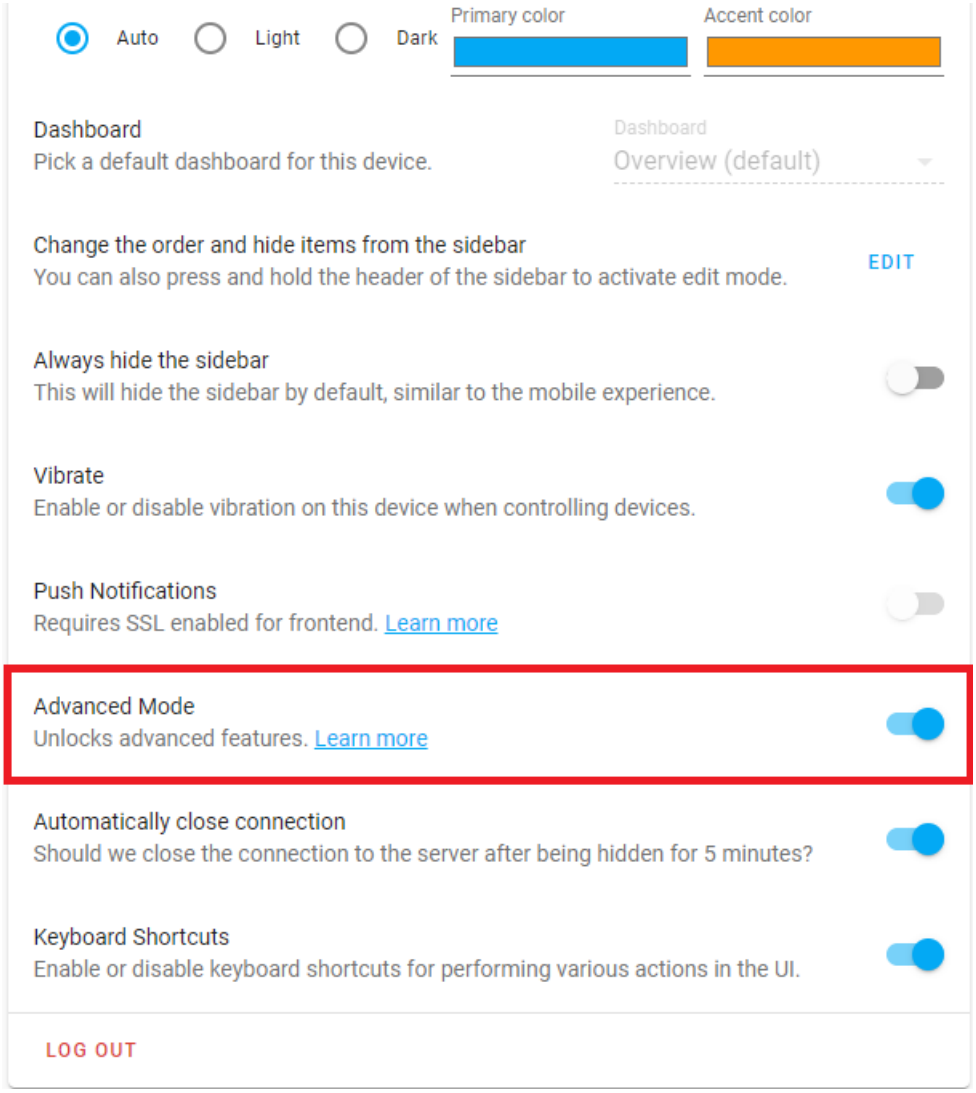

Рисунок 5

Всі налаштування, зміни та інтеграції виконуються у вкладці “Configuration” на лівій бічній панелі (рис. 6). В ній міститься загалом 10 різних розділів для налаштування різних частин системи.

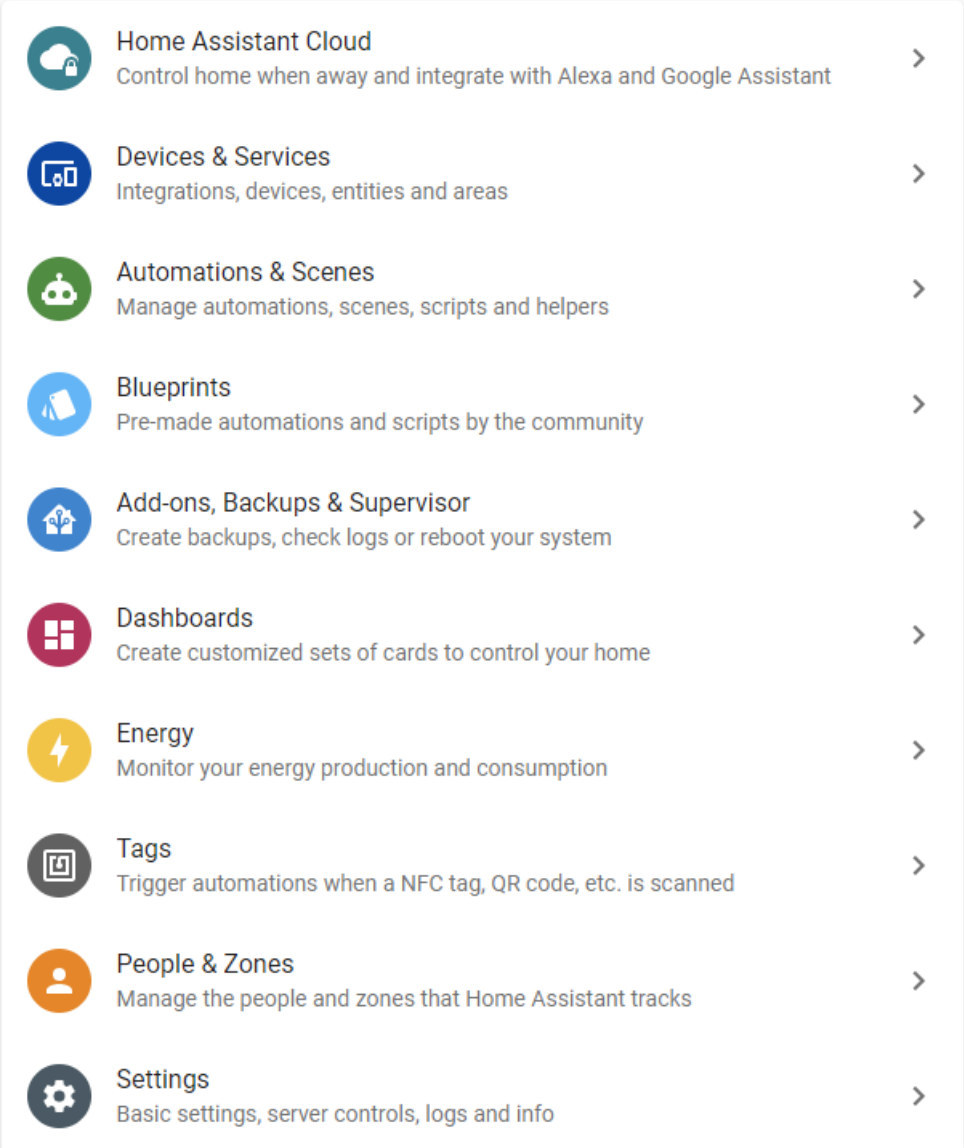

Рисунок 6

- Home Assistant Cloud

Розділ для налаштування хмарних сервісів. Має сенс у випадку передплати хмарного рішення Nabu Casa.

- Devices & Services

Один з основних розділів, які часто використовуються. Цей розділ має чотири частини:

- Integration - використовується для додавання нових пристроїв

- Devices – надає детальну інформацію про пристрої і дозволяє редагувати різні складові, як от вмикати/вимикати сутності, додавати їх на панель приладів і тому подібне;
- Entities – має подібний функціонал до розділу “Devices”, але надає доступ до всіх сутностей відразу, що дозволяє пошуком швидко змінити налаштування конкретного сенсору;
- Areas – розділ для створення та зміни локацій. Можна розділити житло на логічні частини, наприклад на кімнати, і застосовувати для них групові правила

- Automation & Scenes

Теж один з основних розділів, який дозволяє налаштовувати всі можливі дії, які мають бути виконані при певних умовах. Містить чотири підрозділи: Automation, Scenes, Scripts та Helpers

- Blueprints

Розділ, який містить всі завантажені шаблони поведінки. Цей розділ дозволяє створювати, а також завантажувати готові шаблони поведінки створені спільнотою. Завдяки шаблонам можна швидко та легко знайти необхідну конфігурацію для певного пристрою, або ж наприклад завантажити автоматизації, скрипти та ін.;

- Dashboards

Розділ для створення та налаштування додаткових панелей. Може бути корисним у випадку коли необхідно логічно розділити менеджмент. Наприклад можливо створити панель доступну лише для адміністратора, з якої можна керувати обмеженнями, або діями, до яких не мусить мати доступ звичайний користувач;

- Energy

Розділ, який містить всю інформацію по енерговитратам, або витратам газу. Для коректної роботи вимагає спеціальних “розумних” газових або

електролічильників, які збирають та надсилають інформацію. Також сюди можна додати контроль за сонячними панелями, станом акумуляторів, або контролювати енергоспоживання окремих пристроїв

- Tags

Розділ для роботи з NFC картами. Дозволяє створювати картки, які ппри скануванні запускають певний алгоритм дій. Для роботи необхідний зчитувач карток.

- People & Zones

Розділ для створення користувачів. Дозволяє налаштовувати права для кожного користувача, прив’язувати особисті девайси (наприклад смартфон), для відстеження локації, і подальшого запуску алгоритму дій в залежності від зони перебування.

- Settings

Теж один з головних розділів, який поділяється на чотири частини:

- General – налаштування назви, часової зони, URL адреси та мережі
- Server Controls – після зміни конфігурації дозволяє зробити валідацію конфігурації, перезавантажити сервер або лише певну частину (скрипти, автоматизації, та ін.)
- Logs та Info містять відповідно історію про всі події та інформацію про систему, включаючи всі встановленні інтеграції з посиланням на документацію та відомі баги.

2.4.2 Home Assistant Operation System

Home Assistant Operating System (раніше HassOS) — це операційна система на базі Linux, оптимізована для роботи Home Assistant та всіх залежностей. Вона використовує Docker як двигун, який з завантаженням системи розгортає Home Assistant Supervisor як контейнер. У свою чергу Home Assistant Supervisor використовує механізми Docker-у для керування Home Assistant Core і окремими контейнерами з додатками. HA Operation System

створений за допомогою Buildroot, і не базується на певних дистрибутивах (наприклад Debian), а фактично є власним вузькоспеціалізованим дистрибутивом. Він в першу чергу орієнтований на одноплатні пристрої, як-от Raspberry Pi або ODROID, але також підтримує системи x86-64 з UEFI.

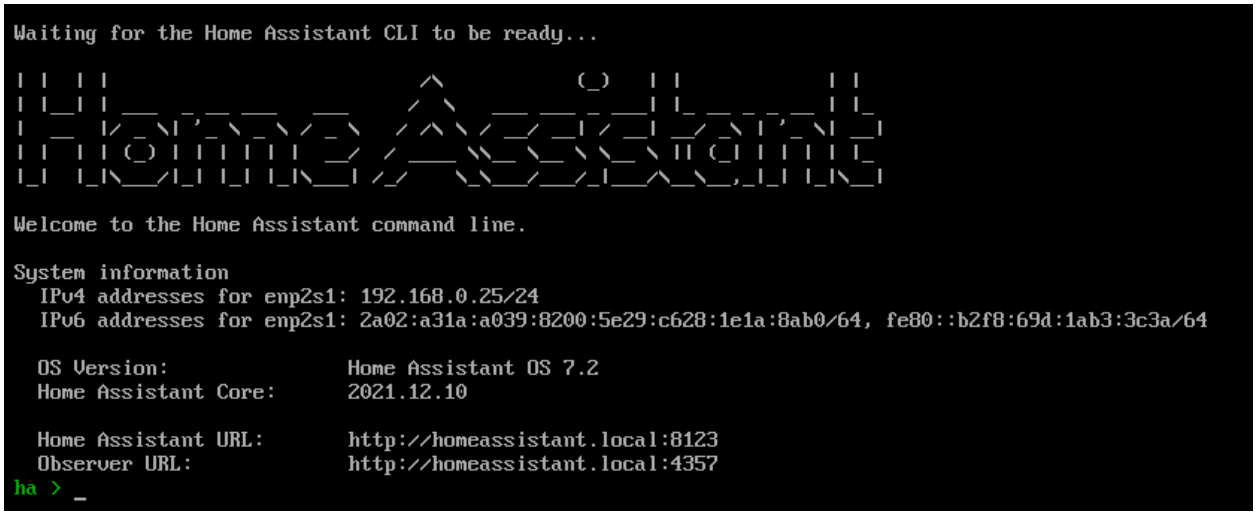

Рисунок 7

CLI для роботи в HAOS – це модифікований Almquist shell, який пітдримує певні заготовлені команди для керування сервісами, перевірки їх статусу, і тому подібне (рис. 7 та 8):

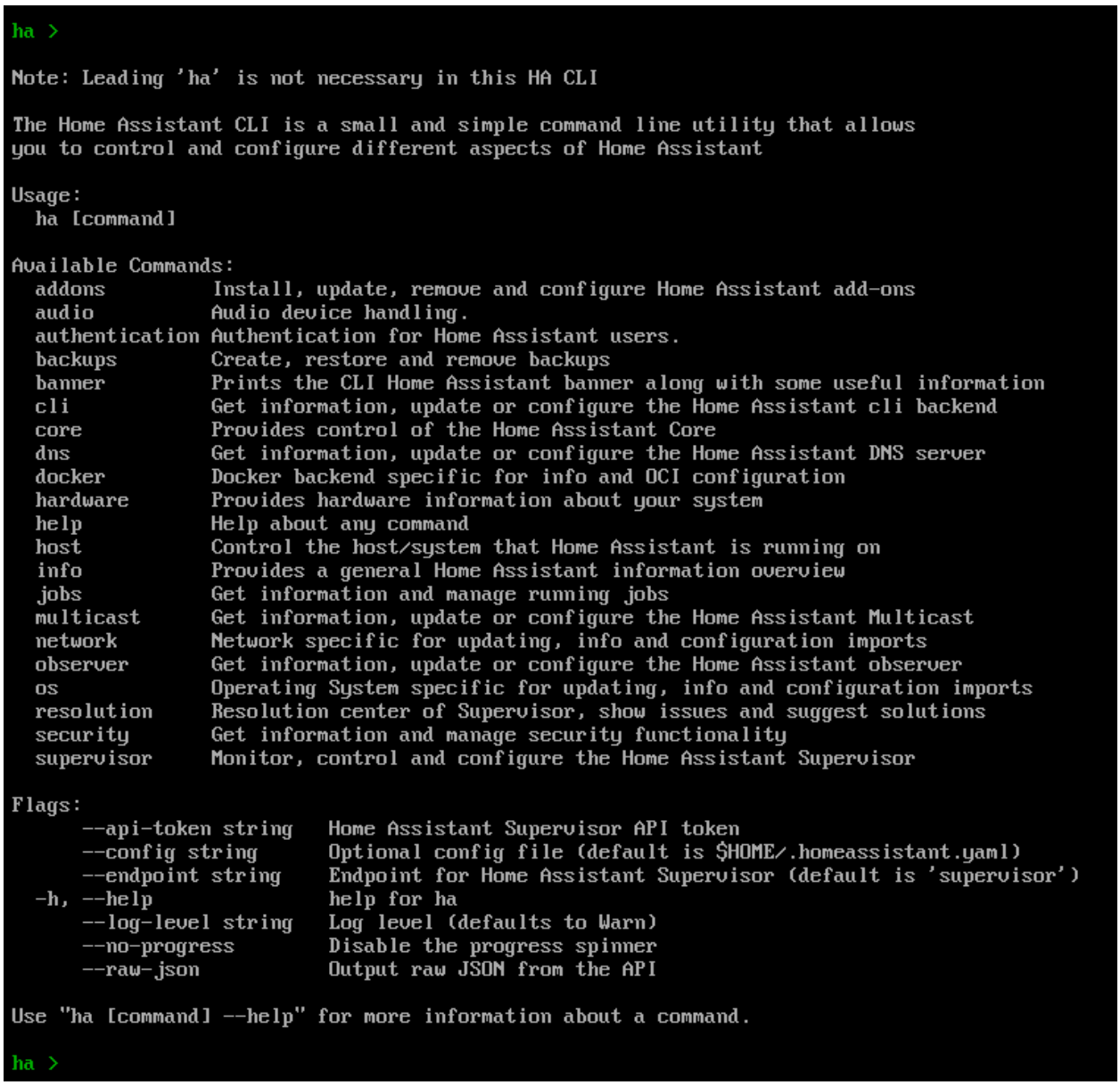

Рисунок 8

З цього середовища можна керувати будь-якими HA процесами. CLI дозволяє оновити систему до певної конкретної версії. Наприклад якщо наявна версія HA Core погано працює з обладнанням, можна повернутися до попередньої версії за допомогою команди core update:

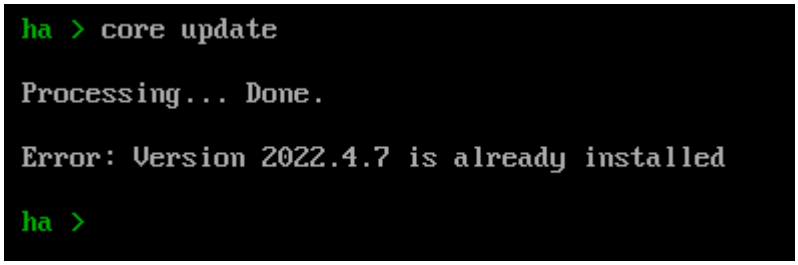

Рисунок 9

Можна виконати перевірку конфігурації перед перезавантаженням ядра.

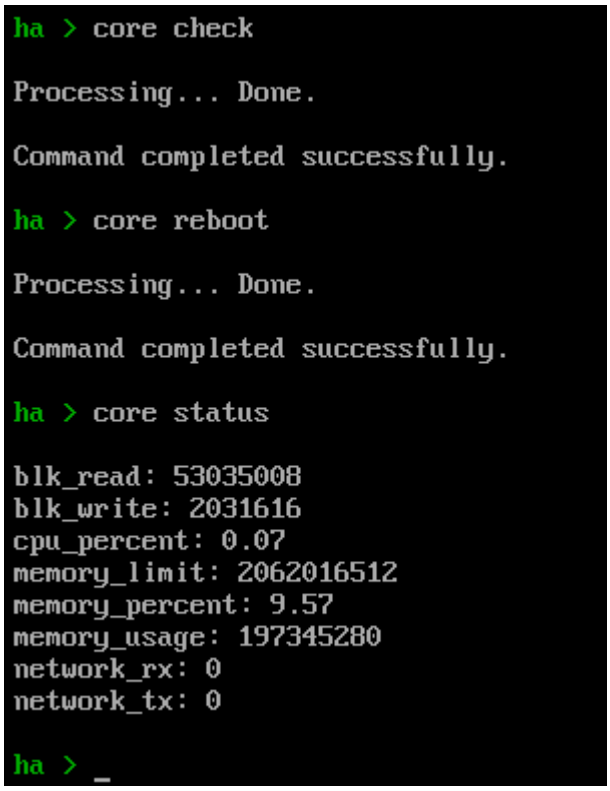

Рисунок 10

Також корисною функцією є створення резервних копій за допомогою backup new:

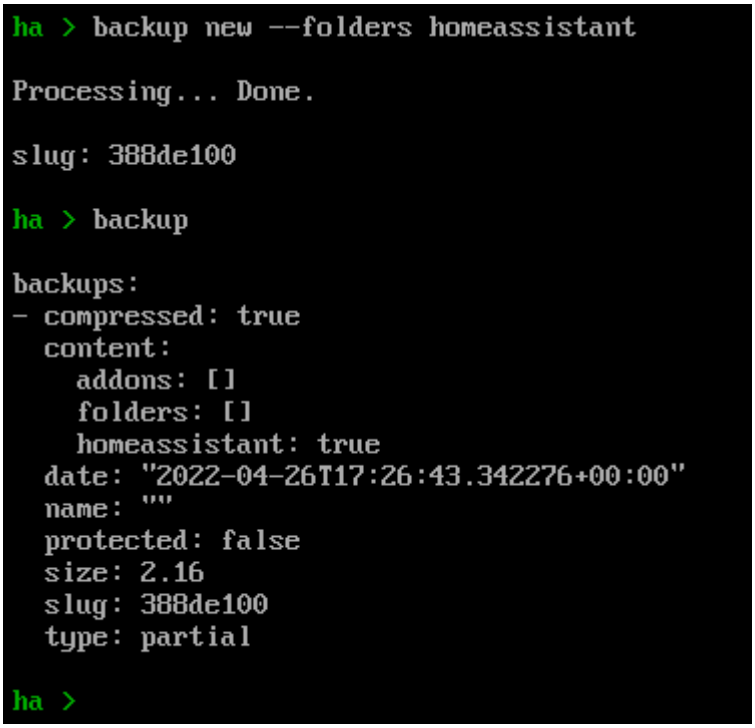

Рисунок 11

CLI не підтримує стандартні команди bash, тому для ручного редагування конфігурацій необхідно потрапити в linux bash, і за допомогою текстового редактору vi змінити файли. Потрапити в bash можна за допомогою команди login. Всі файли знаходяться в директорії /mnt/data/supervisor/homeassistant/

Рисунок 12

2.4.3 Home Assistant Core

HA Core – це просто Python пакет без менеджмент оболонки, тому насправді жодного специфічного способу керування системою немає. Для зміни налаштувань необхідно вручну редагувати конфігураційні файли або змінювати встановленні пакети. Всі зміни рекомендовано робити з віртуального Python середовища. Для його запуску необхідно спочатку перейти до користувача homeassistant:

```
sudo -u homeassistant -H -s
```

А потім запустити середовище наступною командою:

```
source /srv/homeassistant/bin/activate
```

Директорія з HA Core знаходится в “/srv/homassistant/bin/”, а всі конфігураційні фали в домашній директорії “/home/homeassistant/”

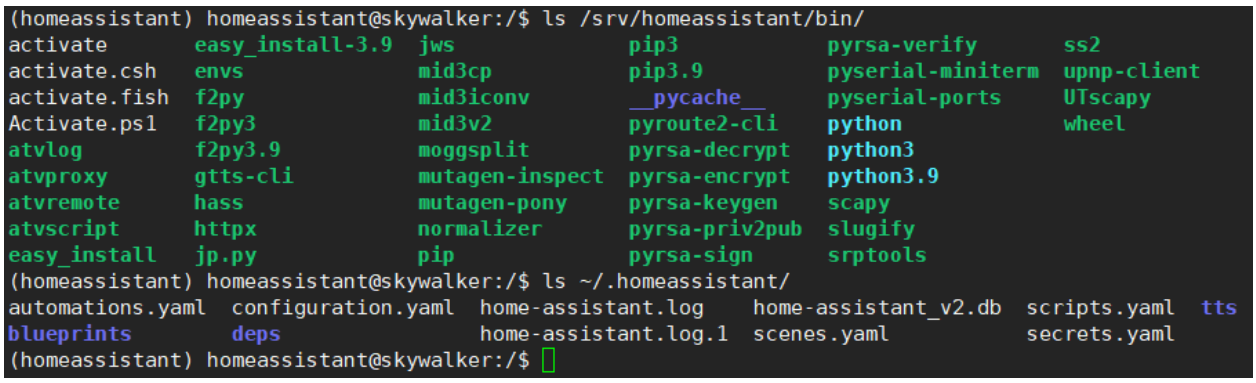

Рисунок 13

Типовими задачами при роботі з Linux середовища є оновлення системи. За допомогою наступної команди можна оновити систему до останньої стабільної збірки

```
pip3 install --upgrade homeassistant
```

Можна натомість оновитися до конкретної версії, наприклад якщо інтегровані пристрої не працюють з останнім оновленням. Це можна зробити за допомогою наступної команди:

```
pip3 install homeassistant==2022.4.7
```

Окремо варто відмітити опцію script пакету hass, які дозволяють виконувати корисні опції:

```
(homeassistant) homeassistant@skywalker:/$ hass --script
Please specify a script to run.
Available scripts: ensure_config, benchmark, auth, macos, check_config
```

Ensure_config: скрипт, який дозволяє виконати початкову ініціалізацію всіх конфігураційних файлів, і додатково перевірити, чи всі необхідні для роботи файли наявні. Це, в разі потреби, дозволяє скинути всі налаштування до налаштувань за замовчуванням. Приклад:

```
(homeassistant) homeassistant@skywalker:/$ mv
~/.homeassistant/configuration.yaml
~/.homeassistant/configuration.yaml.backup
(homeassistant) homeassistant@skywalker:/$ hass --script ensure_config
Unable to find configuration. Creating default one in
/home/homeassistant/.homeassistant
Configuration file: True
(homeassistant) homeassistant@skywalker:/$
```

Benchmark: скрипт, який запускає цикл певної операції, і виводить час відгуку системи. Ця опція може бути корисна для розрахунку швидкодії системи, або для звичайного порівняння різних платформ. Виглядає це наступним чином:

```
(homeassistant) homeassistant@skywalker:/$ hass --script benchmark
valid_entity_id
Using event loop: _UnixSelectorEventLoop
Benchmark valid_entity_id done in 0.3853781030047685s
Benchmark valid_entity_id done in 0.39940059700165875s
Benchmark valid_entity_id done in 0.40058962200419046s
```

Auth: дуже корисний скрипт для роботи з обліковими записами користувачів. Це власне той самий скрипт, який використовує команда authentication з HA OS. Всі можливі дії, які можна виконати з обліковими записами (такі як зміна паролю, додавання або видалення акаунту, та ін.) – необхідно робити за допомогою цього скрипту. Доступні опції:

```
(homeassistant) homeassistant@skywalker:/$ hass --script auth
usage: hass [-h] [--script {auth}] [-c CONFIG]
{list,add,validate,change_password} ...
hass: error: the following arguments are required: func
(homeassistant) homeassistant@skywalker:/$ hass --script auth list
simpson

Total users: 1
(homeassistant) homeassistant@skywalker:/$
```

Macos: це скрипт, який призначений лише для операційної системи MacOS, і дозволяє швидко створити процес, який буде запускати Home Assistant при завантаженні системи. Доступні опції такі як install, uninstall та restart.

Check config: корисний скрипт, який дозволяє перевірити конфігураційний файл на наявність синтаксичних та структурних помилок. Дозволяє перевірити інші файли, а також файли, які використовуються та не використовуються. Наприклад:

```
(homeassistant) homeassistant@skywalker:/$ hass --script check_config --files
--info
Testing configuration at /home/homeassistant/.homeassistant
yaml files (used / not used)
- /home/homeassistant/.homeassistant/automations.yaml
```

```
-
/home/homeassistant/.homeassistant/blueprints/automation/homeassistant/motion
_light.yaml
-
/home/homeassistant/.homeassistant/blueprints/automation/homeassistant/notify
_leaving_zone.yaml
-
/home/homeassistant/.homeassistant/blueprints/script/homeassistant/confirmabl
e_notification.yaml
- /home/homeassistant/.homeassistant/configuration.yaml
- /home/homeassistant/.homeassistant/scenes.yaml
- /home/homeassistant/.homeassistant/scripts.yaml
- /home/homeassistant/.homeassistant/secrets.yaml
Successful config (all)
  homeassistant:
    customize: ?
    customize_domain: ?
    customize_glob: ?
  tts:
    - platform: google_translate
      cache: True
      cache_dir: tts
      language: en
      time_memory: 300
  automation:
  script:
  scene:
(homeassistant) homeassistant@skywalker:/$
```

2.5 Інтеграція пристроїв. Вбудовані сутності. Сенсори

2.5.1 Системні сенсори

Home Assistant, як власна платформа, може керувати сама собою, та бути частиною будь-яких сценаріїв. Це дозволяє як налаштувати моніторинг за платформою, щоб отримувати прямі сповіщення у разі порушення роботи, або перевантаження, так і виконувати різні адміністративні дії на основі скриптів.

Системні сенсори – це універсальні сенсори, які однакові для будь-яких систем, і є чудовим прикладом для демонстрації роботи інтеграцій. Їх можна використовувати незалежно від пристроїв та сенсорів в системі, оскільки зазвичай набір всіх пристроїв для кожної окремої системи унікальний.

Для додавання моніторингу за системою необхідно використати спеціальний сенсор systemmonitor. Всі інтеграції працюють однаково як для Core системи, так і для Operation System, і всі зміни відбуваються у файлі

configuration.yaml, тому надалі їх розгляд буде уніфіковано. Після додавання до конфігурації файл має виглядати наступним чином:

```
root@skywalker:/home/homeassistant/.homeassistant# cat configuration.yaml

# Loads default set of integrations. Do not remove.
default_config:

# Text to speech
tts:
  - platform: google_translate

automation: !include automations.yaml
script: !include scripts.yaml
scene: !include scenes.yaml

sensor:
  - platform: systemmonitor
    resources:
      - type: processor_use
      - type: load_1m
      - type: load_5m
      - type: load_15m
      - type: memory_use
      - type: memory_use_percent
      - type: swap_use
      - type: swap_use_percent
      - type: disk_use
      - type: disk_use_percent
      - type: last_boot
root@skywalker:/home/homeassistant/.homeassistant#
```

Далі необхідно перевірити конфігурацію та повторно запустити систему (для HA CORE):

```
(homeassistant) homeassistant@skywalker:/srv/homeassistant/bin$ hass --script
check_config
Testing configuration at /home/homeassistant/.homeassistant
INFO:homeassistant.util.package:Attempting install of psutil==5.8.0
(homeassistant) homeassistant@skywalker:/srv/homeassistant/bin$ hass
```

Також можливо перевірити та запустити систему з веб-інтерфейсу:

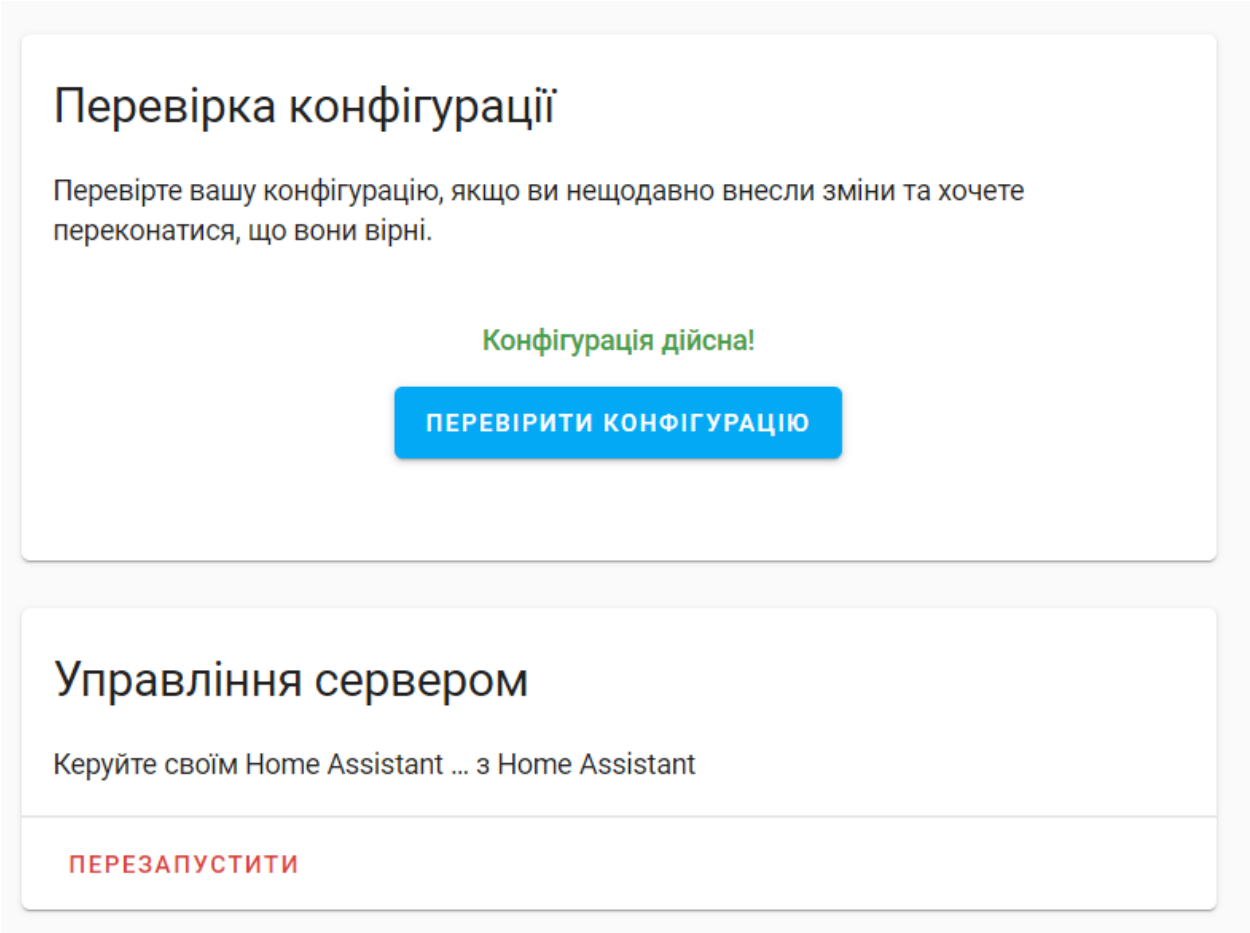

Рисунок 14

Якщо інтеграція додана правильно, на панелі автоматично з’являться всі додані сутності:

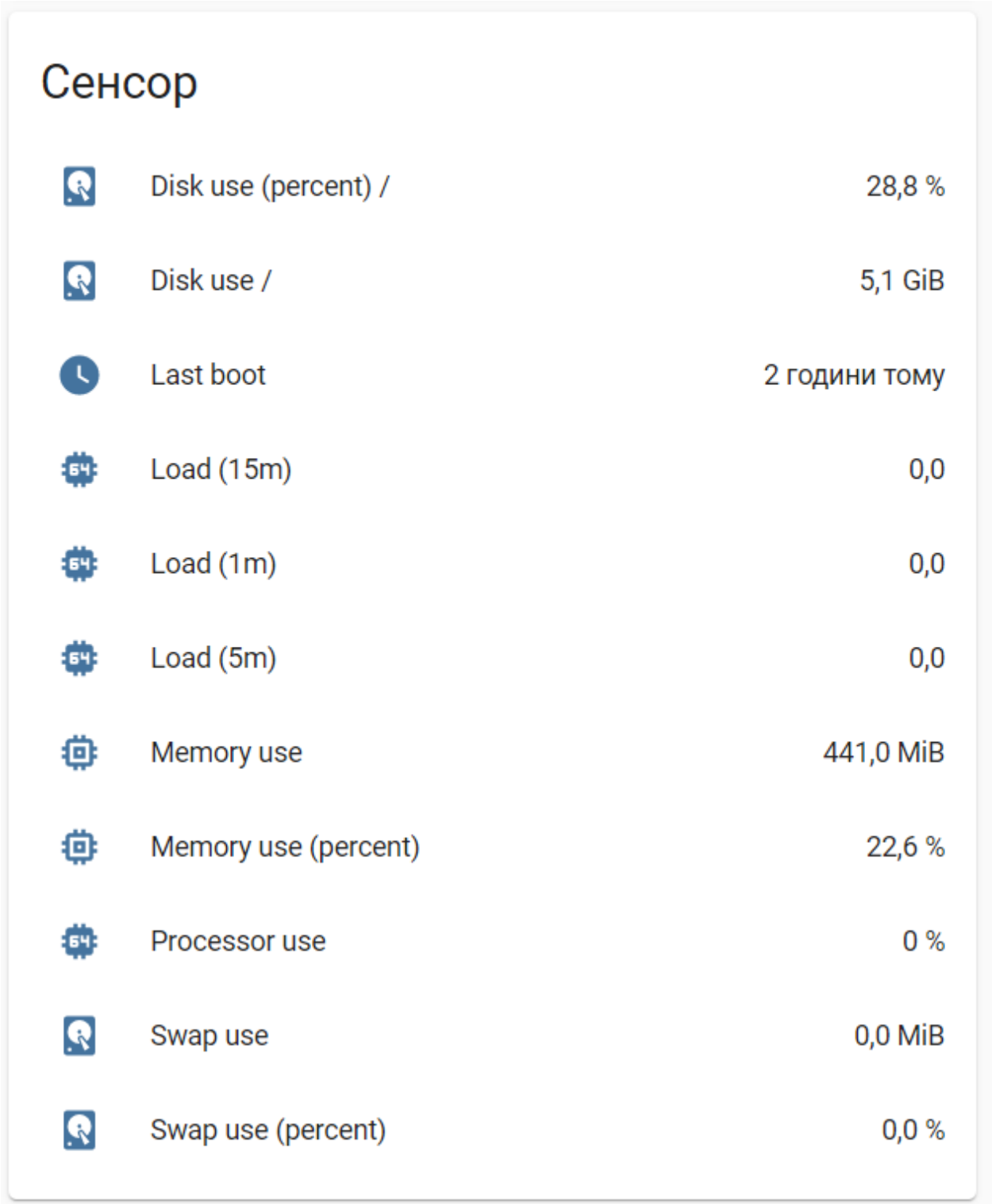

Рисунок 15

В залежності від платформи не всі сенсори можуть зчитувати інформації. Якщо наприклад НА працює на віртуальній машині такі значення як температура процесора і подібні можуть не працювати.

2.5.2 Сторонні інтеграції. Android додаток.

Дуже корисною функцією є додавання певних особистих речей (наприклад смартфон), які дозволять як відслідковувати місце перебування, так і контролювати наприклад заряд акумулятора, і сповіщати в разі необхідності про низький рівень заряду.

Ця інтеграція є за замовчуванням, тому жодних змін в configuration.yaml не треба.

```
root@skywalker:/home/homeassistant/.homeassistant# cat configuration.yaml

# Loads default set of integrations. Do not remove.
default_config:
```

Проте якщо default інтеграції були вимкнені, зв'язок зі смартфоном можна встановити за допомогою mobile_app. Для її роботи необхідно завантажити додаток Home Assistant, і слідувати інструкціям для налаштування. Після синхронізації з додатком Home Assistant автоматично створить інтеграцію:

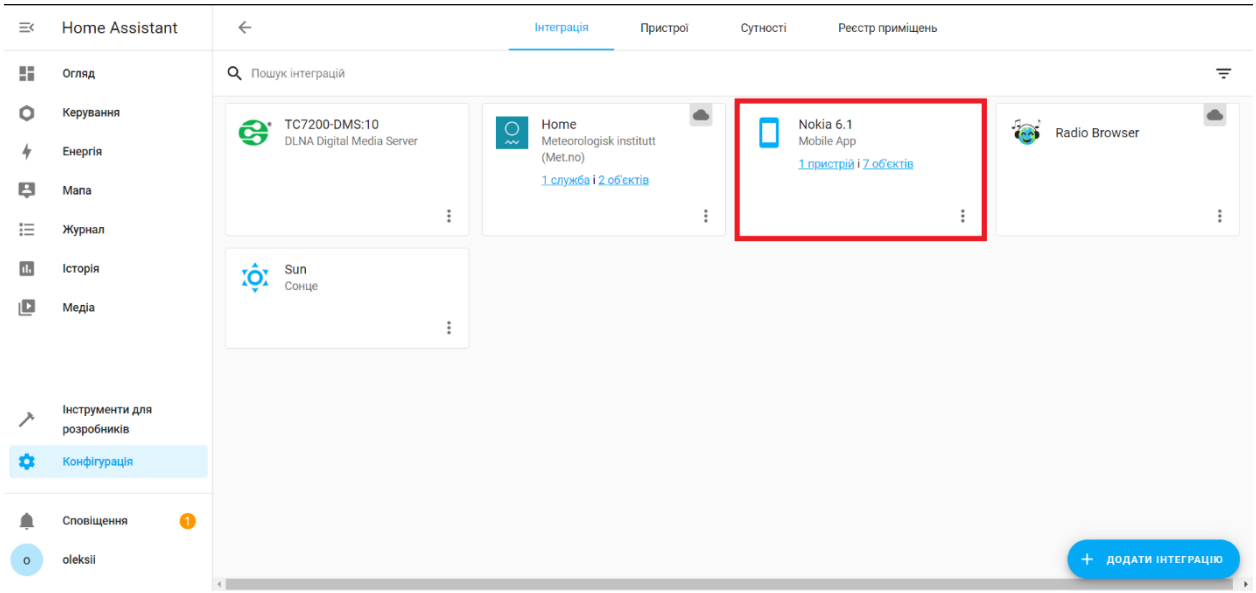

Рисунок 16

Тут будуть доступні всі дані з сенсорів пристрою. Додаткові збір даних можна ввімкнути в додатку. Це дозволить отримувати дуже розширений спектр інформації: від обсягу пам'яті – до даних мобільного оператора.

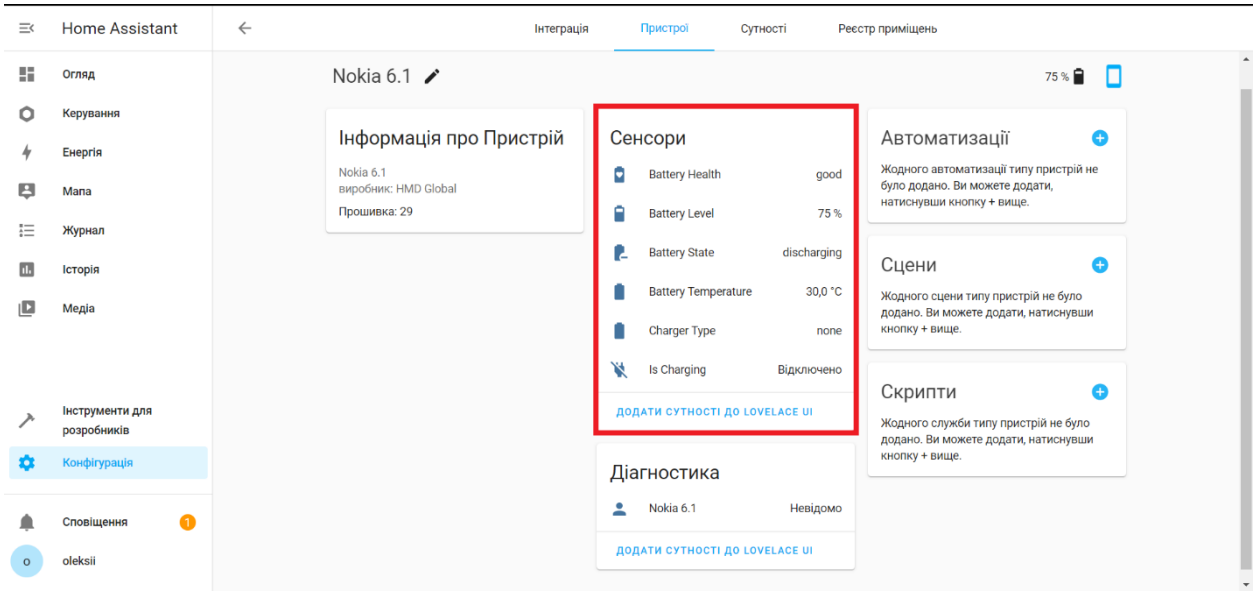

Рисунок 17

Home Assistant в автоматичному режимі виявляє багато пристроїв і служб, доступних у локальній мережі. Більшість інтеграцій можна повністю налаштувати через веб-інтерфейс, однак деякі застарілі або комплексні інтеграції доведеться налаштувати вручну за допомогою YAML конфігурацій.

Розділ 3 Реалізація системи "Розумний дім". Сценарії роботи.

3.1 Принцип автоматизації. Сутності та стани. Частини сценарію.

Вся автоматизація в HA будується за єдиним простим принципом: існує тригер і дія. Також можна додати умову. Один з базових прикладів автоматизації – це ввімкнення освітлення після повернення власника додому ввечері. Цей сценарій поділяється на наступні три частини:

1. Коли власник повертається додому – це тригер
2. Якщо сонце за часом вже зайшло – це умова
3. Вмикання світла – це дія

Тригер описує події, які повинні запустити одну або декілька дій. У цьому випадку це персональний пристрій, яка передає дані щодо місця перебування на основі GPS, або ж можна використовувати знайдені WiFi мережі. Цей стан можна спостерігати за допомогою раніше налаштованого смартфону. Це сутність device_tracker.nokia_6_1

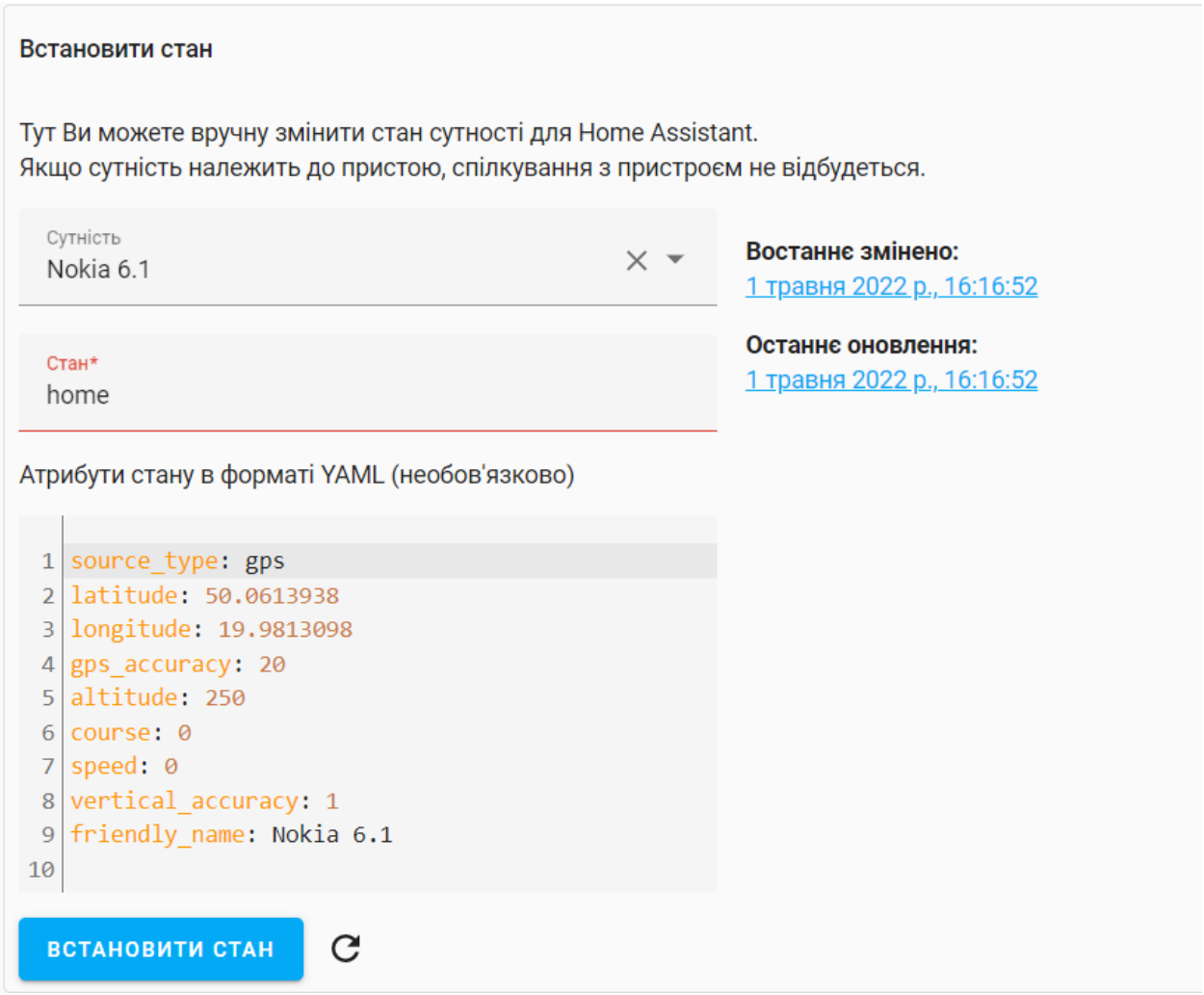

Рисунок 18

Умова — це додаткова не обов’язкова частина, яка обмежую роботу правила лише у конкретних випадках. Після спрацювання тригеру умова буде перевірена, і якщо стан задовільняє умову або умови – тоді буде виконана дія. Умовою може бути будь-що, як от поточний час, стан пристроїв, місцеперебування людей та інші. Для контролю за Сонцем в НА використовується сутність Sun (Сонце) – це вбудована сутність. Вона має різні стани, такі як above_horizon та below_horizon, а також атрибути:

- next_dawn
- next_dusk
- next_midnight
- next_noon
- next_rising

- next_setting
- elevation
- azimuth
- rising

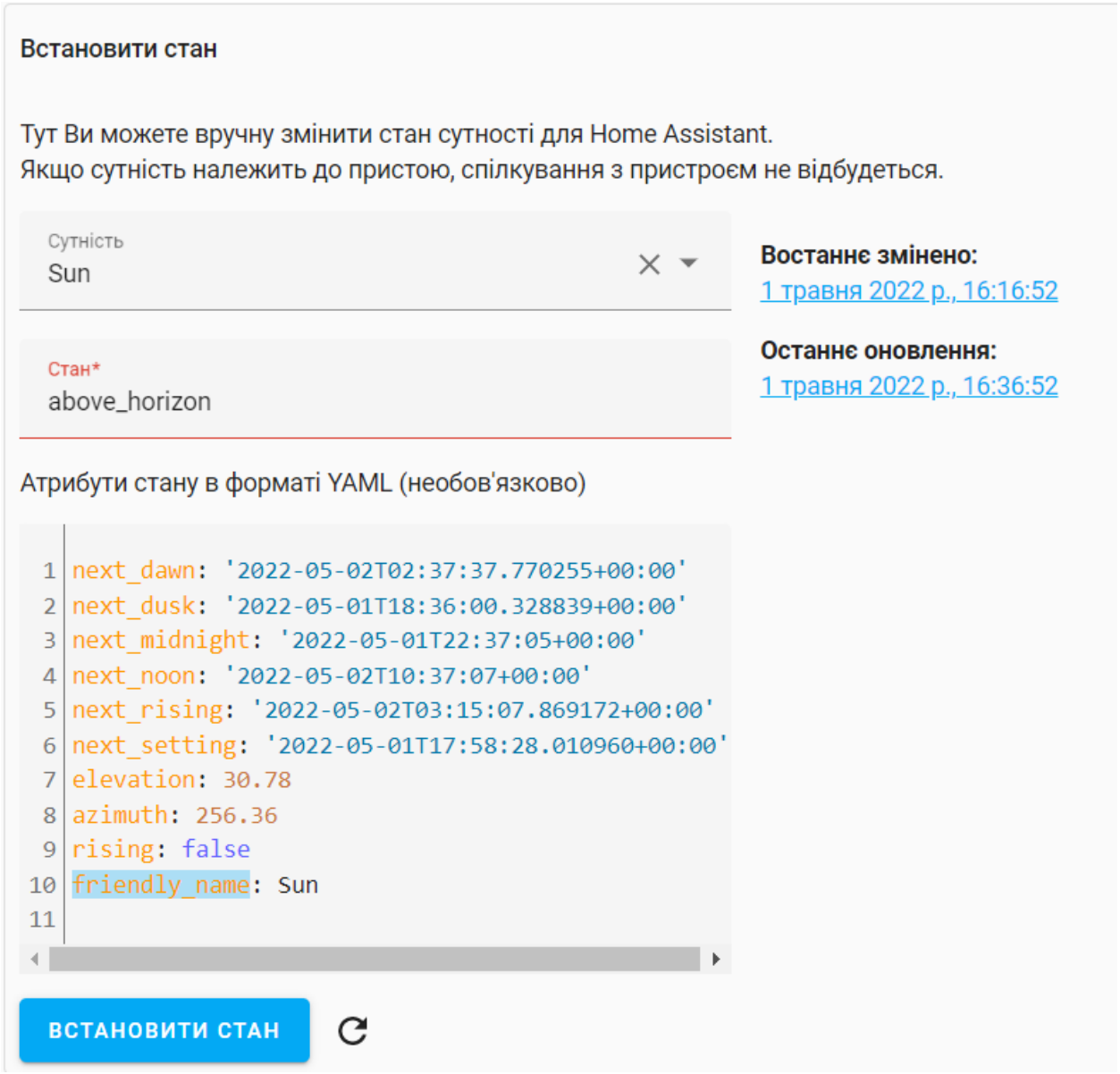

Рисунок 19

Для відслідковування заходу Сонця необхідно використовувати sun.next_setting.

Третя частина — це дія, яка буде виконана, коли спрацює правило і виконуються всі умови. Це може бути будь-що, наприклад

ввімкнення/вимкнення світла, зміна температури, активація сцени, надсилання сповіщення.

Підсумовуючи – вся автоматизація в НА будується на основі сутностей та їх станів. Інформація про поточні сутності доступна в інструментах розробника (нижня ліва частина бічної панелі в інтерфейсі).

3.2 MQTT Broker. Віртуальні пристрої.

MQTT (MQ Telemetry Transport) — це протокол для зв'язку IOT пристроїв типу «пристрій-пристрій», який працює через TCP/IP протоколи. Це забезпечує простий спосіб для встановлення з'єднань за допомогою так званих підписок. Це не обов'язкова частина для роботи розумного дому, але досить корисна в багатьох аспектах.

Для інтеграції MQTT в НА необхідно підключити брокер, який може бути як приватний, що забезпечую повну приватність даних, так і публічний. Для швидкого налаштування можна створити інтеграцію в меню інтеграцій.

Налаштувати власний брокер можна використовуючи одну й ту саму машину, на якій працює НА, або використати окрему. Варто зазначити що деякі брокери не підтримуються (ActiveMQ MQTT та RabbitMQ MQTT), а рекомендованим є Mosquitto MQTT. Він також має публічний брокер, який не варто використовувати для роботи локальних пристроїв, проте він чудово підходить для тестування та налаштування автоматизаціїї. Це найпростіший спосіб який немає конфіденційності, оскільки всі повідомлення є загальнодоступними.

Щоб скористатися загальнодоступним брокером Mosquito необхідно налаштувати інтеграцію MQTT для підключення до test.mosquitto.org через порт 1183 або 8883.

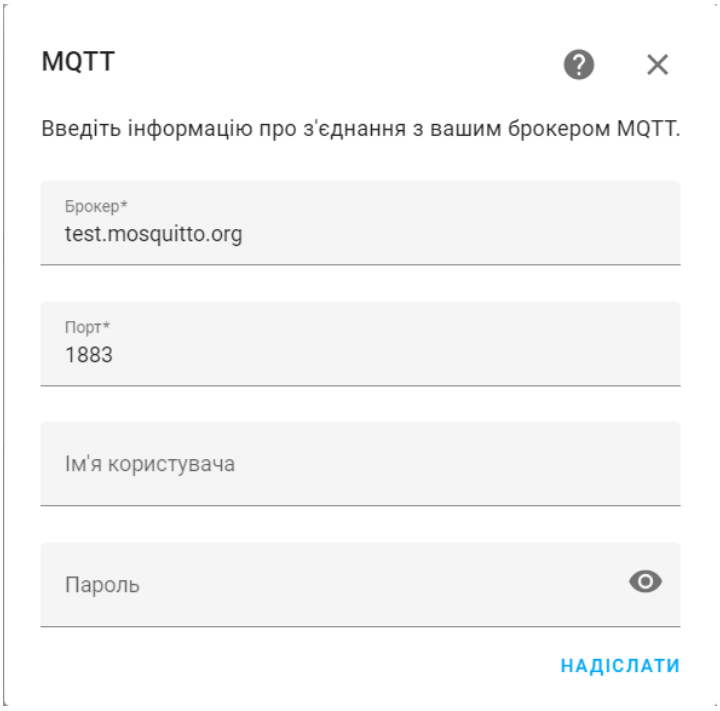

Рисунок 20

Робота через публічний брокер дозволяє інтегрувати велику кількість різних пристроїв, які можна використати в тестуванні.

3.3 Базовий сценарій

Створення автоматизації доступне двома способами – через веб-інтерфейс та ручним редагування конфігураційних файлів. При роботі через інтерфейс зміни все рівно зберігається у форматі YAML в automations.yaml. Цей файл редагується автоматично, тому його краще не редагувати вручну.

Використовуючи попередньо налаштовані сутності, а також джерело освітлення з загальнодоступного MQTT брокера, ми можемо реалізувати базовий сценарій для автоматичного вмикання світла.

Правильним способом буде записати автоматизації безпосередньо в configuration.yaml або створити інший файл. Для цього необхідно додати блок автоматизації до configuration.yaml:

```
alias: Автоматичне ввімкнення світла
description: ''
trigger:
  - platform: device
    device_id: 32081b0eadb496446c105783c4daf60b
    domain: device_tracker
```

```
    entity_id: device_tracker.nokia_6_1
    type: enters
    zone: zone.home
condition:
  - condition: state
    entity_id: sun.sun
    state: below_horizon
action:
  - service: light.turn_on
    data: {}
    target:
      entity_id: light.all_light
mode: single
```

У випадку ручного створення автоматизації після перезавантаження системи інформація з’явиться у вкладці автоматизацій, але редагування буде заблоковане. Контроль, перевірку стану та пробний запуск можна виконати саме з цього вікна. Тестовий запуск допоможе перевірити чи всі дії працюють коректно. Також тут доступна вся інформація про історію спрацювань і тому подібне:

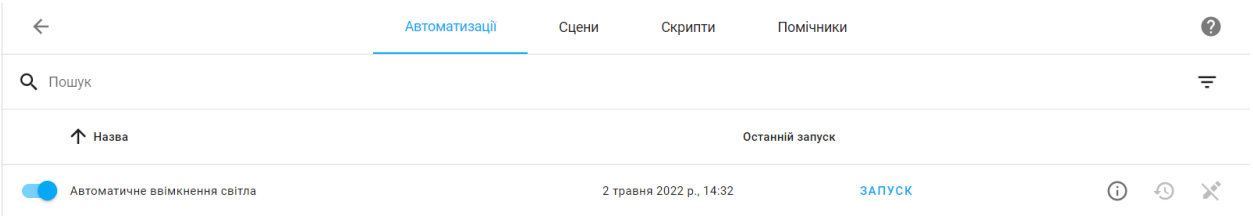

Рисунок 21

Після спрацювання тригеру, та у разі виконання умов світло буде автоматично ввімкнене.

Для перевірки роботи можна не чекати спрацювання тригеру, або досягнення всіх умов, оскільки стан сутностей можна змінити вручну у вікні розробника. Таким чином після створення певних сценаріїв можна відразу перевірити їх відгук на зміну стану, і визначити недоліки. У цьому конкретному випадку для перевірки роботи можемо змінити стани сутностой:

- Стан для device_tracker.nokia_6_1 встановимо в not_home
- Стан для sun встановимо в belowe_horizon

Після оновлення локації на смартфоні автоматизація спрацює і автоматично ввімкне світло

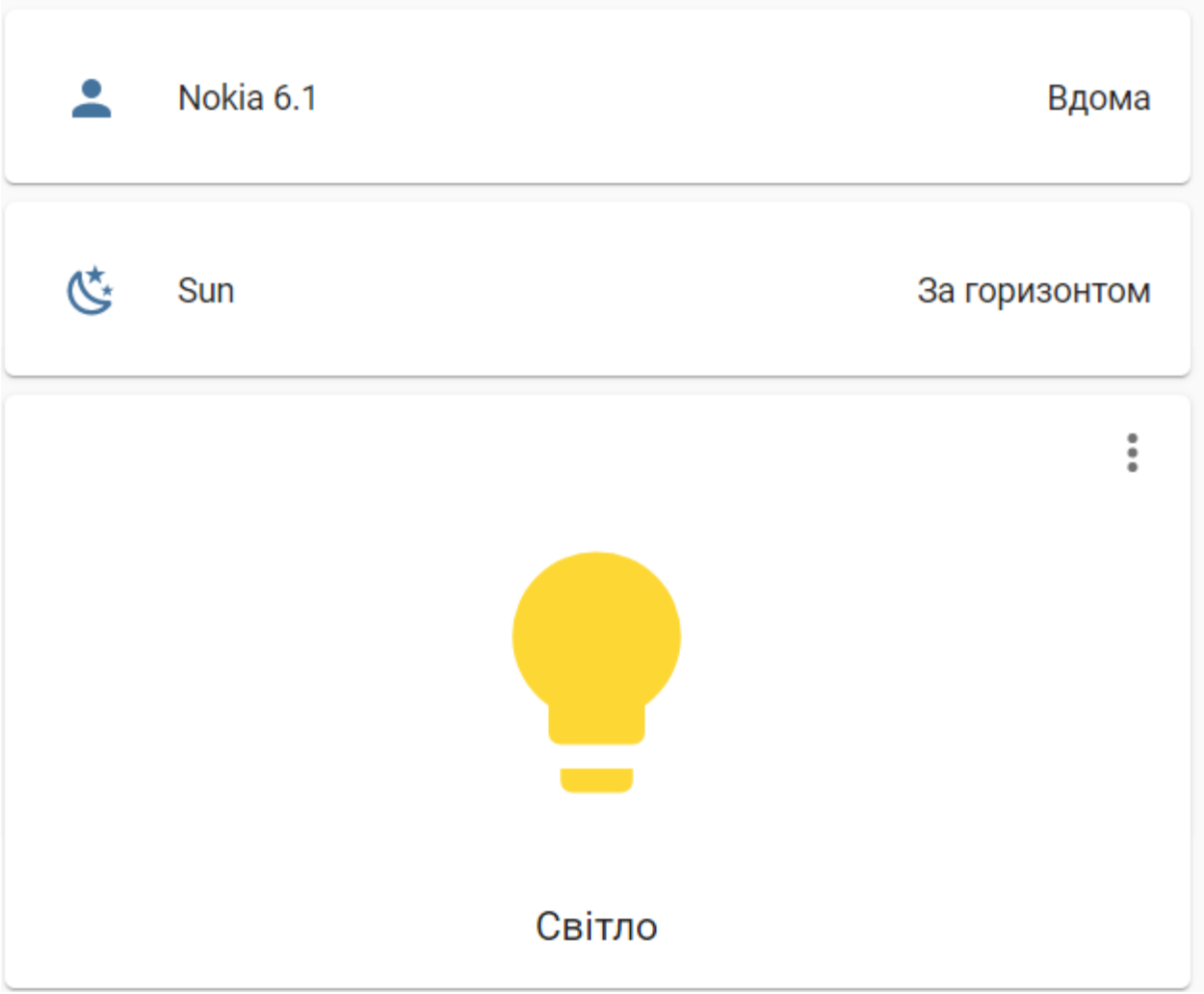

Рисунок 22

3.4 Додатки та додаткові можливості. ESPHome. DIY пристрої.

Home Assistant має змогу інтегрувати різні пристрої зроблені власноруч на основі різних апаратних платформ, як от Arduino або ESP8266. За допомогою додатків (адонів) користувач може створювати та оновлювати конфігурацію. Різні додатки надають різні можливості, а також мають свої особливості. До таких додатків відносять aRest, MySensors, Raspberry Pi, ESPHome та ін

ESPHome може бути встановлений у вікні Налаштувань, опція Додатки (Add-ons, Backups & Supervisor). Цей додаток дозволяє керувати та програмувати мікроконтролери на базі ESP8266 і ESP32 безпосередньо через Home Assistant без досвіду програмування. Він повністю інтегрований в HA, і

дозволяє створювати файли конфігурації мовою YAML, які будуть автоматично перетворені в необхідну прошивку відповідно до обраної платформи.

Найскладнішою частиною додавання нового пристрою в ESPHome є перше завантаження прошивки. Для встановлення потрібно щоб пристрій ESP був підключений спеціальним кабелем до комп'ютера, проте в наступні рази оновлення можна встановити бездротовим способом [13].

Після додавання пристрою до ESPHome, та оновлення конфігурації на ньому Home Assistant автоматично знайде та запропонує інтегрувати його. Кожен доданий пристрій буде окремою одиницею, і матиме всі додані сутності, такі як дані сенсорів, час роботи або потужність сигналу (рис. 23).

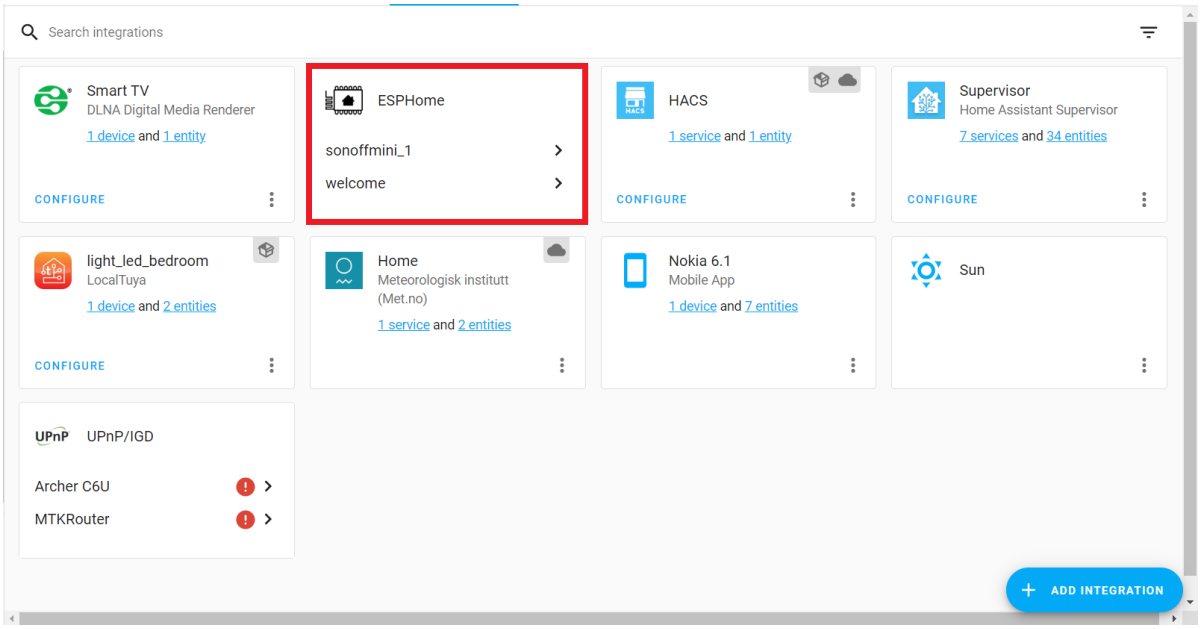

Рисунок 23

Налаштування кожної платформи має свої особливості завдяки різній будові, оскільки кожен пристрій має різну кількість портів з різними порядковими номерами.

Простий приклад конфігурації для бінарного сенсору на базі ESP01 буде виглядати наступним чином:

```
esphome:
  name: welcome
```

```
esp8266:
  board: esp01_1m

# Enable logging
logger:

# Enable Home Assistant API
api:

ota:
  password: "6bbe14914bc1c536f7c79920aebe6a32"

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

  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:
    ssid: "Welcome Fallback Hotspot"
    password: "urlOJdsKfE8n"

captive_portal:

binary_sensor:
  - platform: gpio
    pin:
      number: RX
    name: Hall motion
    device_class: motion
```

До плати підключено звичайний бінарний сенсор руху, який має два стани, і у разі спрацювання надсилає сигнал до HA. Сенсор має три стани:

- On
- Off
- Unavailable

Після інтеграції пристрій автоматично з’явиться на панелі. Він буде мати сконфігуровані атрибути, назву та тип відповідно до створеного файлу в ESPHome:

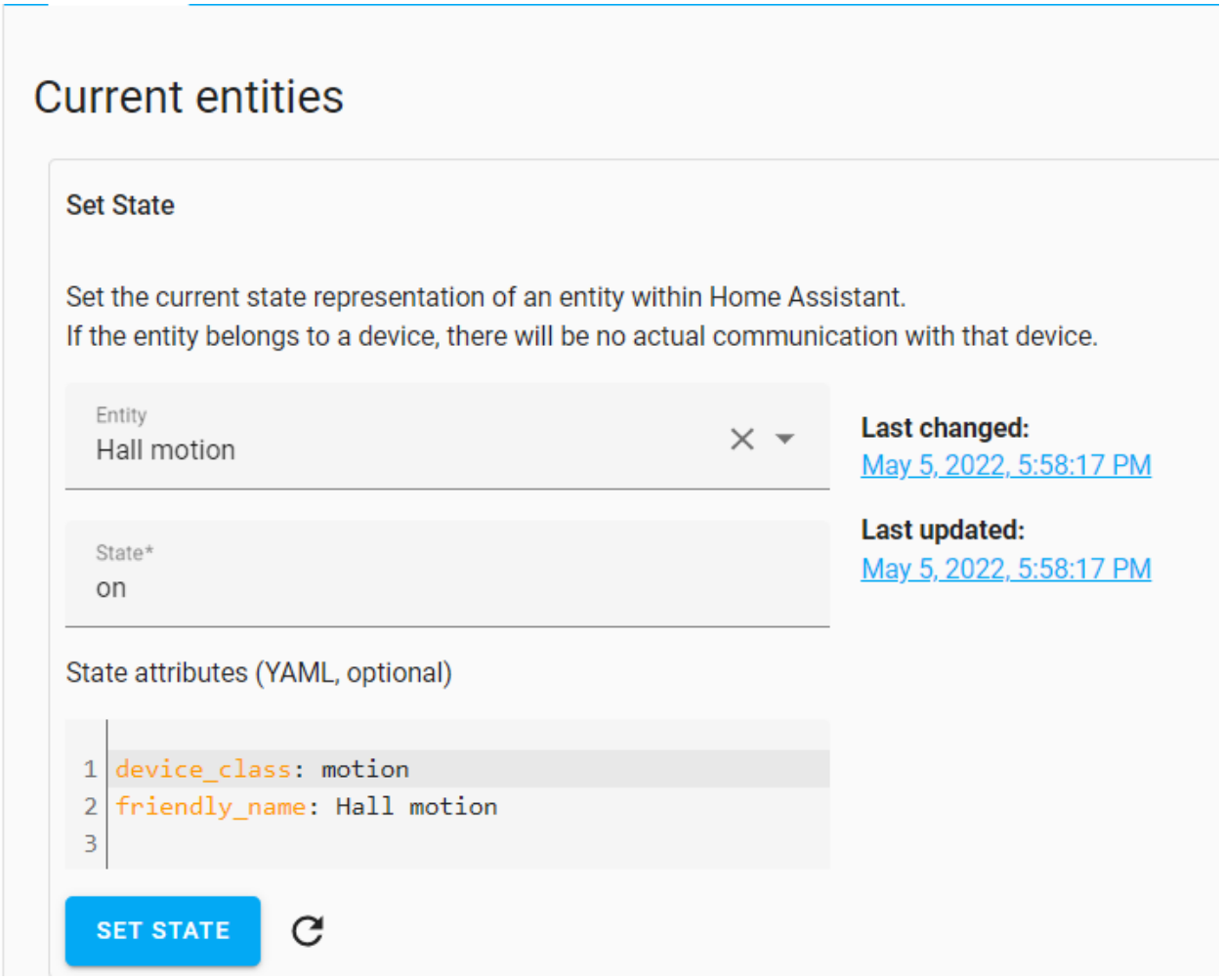
Current entities
Set State
Set the current state representation of an entity within Home Assistant.
If the entity belongs to a device, there will be no actual communication with that device.
Entity
Hall motion
Last changed:
May 5, 2022, 5:58:17 PM
State*
on
Last updated:
May 5, 2022, 5:58:17 PM
State attributes (YAML, optional)
1 device_class: motion
2 friendly_name: Hall motion
3
SET STATE

Рисунок 24

ВИСНОВКИ

У результаті виконання роботи було проведено аналіз доступних рішень, обрано найбільш оптимальне серед інших, та створено методичні вказівки до покрокового встановлення, налаштування та автоматизації.

Дана робота доводить, що обрана система повністю відповідає вимогам щодо сумісності з іншими виробниками, та не залежить від типу або апаратної платформи пристроїв. Це дозволяє в повній мірі реалізувати будь-які необхідні автоматизації за допомогою DIY пристроїв, як створених власноруч, так і іншими студентами.

Хоч платформа і є орієнтованою на роботу в приватних господартсвах, проте система може бути використана для автоматизації процесів всередині університету, або структурних підрозділів.

У даній роботі було налаштовано систему Home Assistant, як платформу для автоматизації IoT пристроїв. Було створено базовий сценарій на основі геолокації смартфону для ввімкнення світла, а також створено, налаштовано та інтегровано сенсор руху на апаратній платформі ESP-01.

Ця робота, та технологія «розумний дім» загалом, має доцільність для подальшого дослідження, оскільки система має потенціал для розвитку. Home Assistant, окрім розглянутих в роботі сценаріїв автоматизації, може бути застосована для контролю доступу, моніторингу безпеки, сповіщення власників, та ін.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. X10 Knowledge Base / База знань протоколу X10 [Електронний ресурс]: https://kbase.x10.com/wiki/Main_Page
2. Zigbee specification / Протокол роботи Zigbee [Електронний ресурс]: https://csa-iot.org/all-solutions/zigbee/
3. LG ThinQ with Voice Assistant / Голосовий помічник для LG ThinQ [Електронний ресурс]: https://www.lg.com/ca_en/support/lgthinq-with-voice-assistant
4. Samsung Smart Things supported devices / Підтримувані пристрої Samsung Smart Things [Електронний ресурс]: https://www.samsung.com/ua/apps/smartthings
5. OpenHub Add-on Reference / Доступні додатки OpenHub [Електронний ресурс]: https://www.openhab.org/addons
6. OpenHub Concept / Концепт роботи OpenHub [Електронний ресурс]: https://www.openhab.org/docs/concepts
7. MajorDomo доступні виробники [Електронний ресурс]: https://connect.smartliving.ru/components.html
8. IoBroker connects different smart home systems / Під'єднання різних платформ розумного дому до IoBroker [Електронний ресурс]: https://www.iobroker.net/#en/documentation
9. Architecture. Servers. Hardware requirements / Архітектура. Сервери. Мінімальні вимоги. [Електронний ресурс]: https://www.iobroker.net/#en/documentation/basics/README.md
10. Home Assistant Installation / Встановлення Home Assistant [Електронний ресурс]: https://www.home-assistant.io/installation
11. Home Assistant YAML [Електронний ресурс]: https://www.home-assistant.io/docs/configuration/yaml
12. Installation method Home Assistant Supervised / Метод встановлення для HA Supervised [Електронний ресурс]: https://github.com/home-assistant/supervised-installer

13. Getting Started with the ESPHome Command Line / Початок роботи з командним рядком ESPHome [Електронний ресурс]: https://www.esphome.io/guides/getting_started_command_line.html