МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп'ютерної інженерії

ДО ЗАХИСТУ ДОПУЩЕНА

Зав. кафедрою _____

д.т.н., проф. Переверзєв А.В.

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

МІКРОПРОЦЕСОРНА СИСТЕМА
УПРАВЛІННЯ ЖИТЛОВИМ КОМПЛЕКСОМ.
ПРОГРАМНА ЧАСТИНА

Виконав
ст. гр. КІ-128 _____ М.С. Змієвський

Керівник
професор _____ С.О. Сабанов

Запоріжжя
2022

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Зав. кафедрою
д.т.н., професор
А.В. Переверзєв

_____

17.01.2022 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ

Студенту гр. *КІ-128*, спеціальності *123 «Комп'ютерна інженерія»*

Змієвському Максиму Сергійовичу

1.Тема: *Мікропроцесорна система управління комплексом. Програмна частина*

затверджена наказом по інституту 06.1-50 від 15 січня 2022р.

2. Термін здачі студентом закінченої роботи: 17 червня 2022 року

3. Перелік питань, що підлягають розробці:

1. Вивчити літературу, присвячену темі розробки.

2. Розглянути загальні відомості про загальну концепцію та системи розумного будинку.

3. Виконати огляд технологій Smart Home.

4. Навести приклади реалізації Smart Home, що присутні на ринку.

5. Ознайомитися з технологіями програмної реалізації систем розумного будинку та готовими рішеннями, що використовуються на практиці

6. Провести аналіз принципової схеми модуля розумного дому, для якого буде створюватися програма управління.

7. Виконати вибір інструментальних засобів для створення ПЗ комплексу.

8. Ознайомитись з середовищем розробки Arduino IDE, Visual Studio розробити алгоритм функціонування системи та реалізувати його програмно.

9. Відлагодити створену програму на діючому пристрої.

10. Оформити результати роботи у вигляді звіту.

Дата видачі завдання: 18 січня 2022 р.

Керівник бакалаврської роботи _____ С.О. Сабанов

Завдання отримав до виконання _____ М.С. Змієвський

РЕФЕРАТ

Бакалаврська дипломна робота: 77 сторінок, 22 малюнків, 20 першоджерела.

Об'єкт розробки: автоматизовані системи з мікроконтролерним управлінням.

Мета роботи: створення програмного забезпечення для керування підсистемою розумного будинку, побудованою на базі мікроконтрелера.

У бакалаврській роботі розглянуто концепцію розумних житлових комплексів, основні технології їхньої побудови та приклади використання основних типів IoT-речей. Наведено принципи роботи основних підсистем розумних будинків та виконано огляд найбільш популярних реалізацій, що представлені на ринку.

Запропоновано розробку, що використовує різні апаратні частини, які взаємодіють з сервером з використанням фреймворку Angular для візуалізації даних, Node.Js для взаємодії з базою даних, та база даних mongoDb.

ANGULAR, ARDUINO, API, IOT, MONGODB,
МІКРОКОНТРОЛЕР, РОЗУМНИЙ БУДИНОК, ФРЕЙМВОРК

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ, СИМВОЛІВ ТА ТЕРМІНІВ 6
ВСТУП 8
РОЗДІЛ 1 ОПИС ТЕХНОЛОГІЙ 11
1.1 Технології Smart Home. Приклади використання 11
1.1.1 Smart-Light рішення 11
1.1.2 Smart Entertaiment рішення 13
1.1.3 Розумна побутова техніка 15
1.1.4 Модулі та компоненти розумного дому 17
1.2 Інтелектуальні IoT платформи 20
1.3 Додатки для роботи з стандартними системами IoT 23
1.3.1 Amazon Alexa 23
1.3.2 SmartThings 24
1.3.3 Google Home 26
1.3.4 Apple Home 27
РОЗДІЛ 2 ВИБІР ЗАСОБІВ РОЗРОБКИ ТА РІШЕННЯ ЗАДАЧ ПРОЕКТУ . 30
2.1 Основні системи розумного будинку 30
2.1.1 Принципи побудови системи 34
2.1.2 Централізовані системи автоматизації ...**Ошибка! Закладка не определена.**
2.1.3 Децентралізовані системи автоматизації**Ошибка! Закладка не определена.**
2.1.4 Змішана система керування .....**Ошибка! Закладка не определена.**
2.1.5 Способи побудови комутуючого середовища 36
2.1.6 Дротові системи автоматизації .....**Ошибка! Закладка не определена.**
2.1.7 Бездротові системи автоматизації .....**Ошибка! Закладка не определена.**
2.2 Фреймворки та платформи для написання системи управління розумним будинком 38
2.2.1 ReactJs 39

2.2.2 Vue.Js 40
2.2.3 Angular 41
2.2.4 Node.Js 42

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ, СИМВОЛІВ ТА ТЕРМІНІВ

Слово/словосполучення	Скорочення	Умова використання
М		
Мегагерц	Мгц	В тексті
Г		
Гігагерц	Ггц	В тексті
О		
Операційна система	ОС	В тексті
A		
Alternating Current	AC	В тексті
C		
Customer relationship management	CRM	В тексті
D		
Direct Current	DC	В тексті
Dependency injection	DI	В тексті
Document Object Model	DOM	В тексті
E		
Electrically Erasable Programmable Read-Only	EEPROM	В тексті

Memory		
G		
Ground	GND	В тексті
I		
Internet Of Things	IoT	В тексті
iPhone Operation System	iOS	В тексті
L		
Light-emitting diode	LED	В тексті
W		
Wireless Fidelity	Wi-Fi	В тексті

ВСТУП

Стандартного визначення того, що означає термін Smart Home, не існує. Зазвичай, розумний будинок, або в ширшому розумінні, – розумний житловий комплекс, – це загальний термін для житлового середовища, в якому основні елементи життєзабезпечення, такі як освітлення, водопостачання, охоронні та кліматичні системи, а також побутові електроприлади, об'єднані в єдину мережу. Керування такою системою виконується через термінальні пристрої – смартфони, планшети чи комп’ютери.

На протязі десятиліть окремі операції – від натискання вимикача світла до відкриття дверей гаража за допомогою дистанційного керування, – були послідовно автоматизовані. Сама ж концепція бере свій початок ще в 1934 році на Всесвітній виставці в Чикаго, де було представлено «дім майбутнього». Проте за останні 80 років автоматизований дім перетворився на розумний дім завдяки Інтернету, сучасним датчикам і засобам комунікації.

Будь-яке житлове приміщення з підключенням до Інтернету вже є розумним будинком. Мобільні пристрої, ноутбуки та телевізори з підключенням до Wi-Fi дають можливість керувати певними частинами середовища, наприклад, транслювати фільм з ноутбука на екран телевізора. Однак справжнє налаштування розумного будинку містить пристрої, які взаємопов’язані, якими користувач може керувати прямо чи опосередковано. Таким чином, з поняттям розумних житлових комплексів тісно пов’язане ще одне поняття – Інтернет речей.

IoT – це багаторівнева система, що включає датчики і контролери, встановлені на вузлах і агрегатах об'єктів, засоби передачі даних, що збираються, та візуалізації цих даних, потужні аналітичні інструменти інтерпретації отриманої інформації та багато інших компонентів. Інтернет

речей – це новий етап розвитку Інтернету, що значно розширює можливості збору, аналізу та розподілу даних, які людина може перетворити на інформацію.

В контексті розумного будинку, IoT – це модулі, з яких створюються якісь додаткові внутрішні контури сучасного житла. Зокрема, у складі елементів і приладів розумного дому широко використовуються автоматичні (розумні) вимикачі, «розумні» розетки, камери з функцією відправлення та архівування інформації в «хмарі», інтелектуальні системи кондиціювання та вентиляції, тощо.

Розумні житлові комплекси працюють за принципом введення, обробки та виведення інформації, внаслідок чого вихідні параметри можуть впливати на вхідні. Введення інформації може здійснюватися різними способами подачі команд: натисканням клавіші, датчиком руху, освітлення, жестами, голосом, тощо. Обробка виконується зазвичай централізовано в блоці управління системою Smart-Home. Тут усі smart-пристрої з'єднуються за допомогою різноманітних каналів зв'язку, як кабельних, так і бездротових: WLAN, Bluetooth, ZigBee, Z-Wave або KNX.

Апаратна складова розумних житлових комплексів може бути досить різною. В простіших випадках, деякі розумні будинки можна побудувати на базі Arduino або Raspberry Pi. Складні ж комплекси базуються на потужних серверних платформах.

Обладнання, що використовується, визначає вибір програмних засобів для управління комплексом. Програмне забезпечення повинно обслуговувати три основні типи пристроїв для Smart Home:

- контролер (хаб), керуючий пристрій, що з'єднує всі елементи системи один з одним і пов'язує її із зовнішнім світом;
- сенсори (датчики), пристрої, які отримують інформацію про зовнішні та внутрішні фактори та умови;

- актуатори, пристрої, що відповідають безпосередньо за виконання команд.

Дії обладнання, які воно повинно виконати за певних зовнішніх умов або під час вступу команди, визначаються сценаріями. Їх програмування має на увазі голосове, пультове управління та контроль за допомогою цифрової апаратури. Наприклад, розмова з розумним динаміком – це спосіб безпосередньо керувати розумною екосистемою. Прикладом непрямого керування може бути випадок, коли камера розумного динаміка виявляє рух і миттєво вмикає світло, коли хтось проходить повз.

В роботі наводиться опис усіх важливих етапів створення розумного житлового комплексу:

- визначення етапів взаємодії внутрішніх складових сценарію з користувачем та між собою, а також відгук програми на нестандартні ситуації;
- підбір або створення драйверів модулів системи та відповідного інтерфейсу для кожної групи пристроїв;
- безпосереднє програмування на основі попередніх етапів.

Зважаючи на складність виконання завдання зі створення комплексної програми, було прийняте рішення проілюструвати процес програмування тільки однієї підсистеми.

Був використаний фреймворк Angular для візуалізації даних, Node.Js для взаємодії з базою даних, та база даних mongoDb.

Реалізація проекту була виконана з врахуванням можливості поширення та масштабування, додаючи до цієї системи контроль освітлення, щоб тим самим контролювати штучне освітлення приміщення, чи зробити автоматичну роботу штор на вікнах. Його можна об’єднати з іншими модулями і створити цілу екосистему оселі.

РОЗДІЛ 1
ОПИС ТЕХНОЛОГІЙ

1.1 Технології Smart Home. Приклади використання

На ринку сучасних рішень наразі представлений досить широкий спектр технологій розумного дому, від окремих пристроїв для комунальних послуг, розваг і зв’язку до комплексних мереж, створених для того, щоб об’єднати все це разом. Найголовніше, що робить цю технологію настільки важливою – це постійно зростаючий попит на рішення для Smart Home.

1.1.1 Smart-Light рішення

Розумні технології інтегрували майже всі аспекти домашнього життя, включаючи освітлення. Завдяки їм можна керувати світлом у домі з будь-якої точки світу, запрограмувати їх за розкладом і вмикати їх автоматично, коли знадобиться.

Одним із прикладів деяких найкращих рішень для освітлення розумного дому є Holî SleepCompanion [1]. Ця спеціалізована розумна лампочка розроблена для того, щоб розбудити природним чином із спеціально налаштованим синім відтінком, який знижує рівень мелатоніну у тілі. У нього є інший режим, який допомагає швидше заснути. По суті, цей продукт не тільки освітлює кімнату, але й допоможе відпочити в довгостроковій перспективі.

Ще одним інноваційним продуктом освітлення є Flux Bluetooth Smart LED (рисунок 1.1). Ця одна лампочка підтримує понад 16 мільйонів колірних градієнтів і повністю керується за допомогою програми для смартфона. Його

можна запрограмувати так, щоб він світився різними кольорами в різний час доби, і навіть можна запрограмувати реагування на музику в кімнаті.

Рис. 1.1 – Додаток Flux Bluetooth Smart LED

Sengled Boost має вбудований розширювач сигналу, який розширює діапазон Wi-Fi. Чим більше їх, тим краще буде Wi-Fi по приміщенню. Сам розширювач знаходиться під LED-освітлювачами (рисунок 1.2).

Рис. 1.2 – Лампа Sengled

1.1.2 Smart Entertaiment рішення

Оскільки бездротові технології, смартфони та Інтернет стають частиною повсякденного життя, самі наші будинки перетворюються на простір, керований комп'ютером. Це, звісно, включає розважальні пристрої. Приклад того, що доступно з точки зору розумних домашніх розважальних пристроїв, склавши список деяких з найкращих і найпростіших у використанні варіантів.

Коли справа доходить до розумної автоматизації в домі, розважальні пристрої, мабуть, є першим очевидним кроком, який потрібно зробити. Існуюча електроніка, ймовірно, готова до підключення, якщо її було придбано нещодавно. Завжди потрібен центр для керування ними, розумний спосіб підключення до телефону чи планшета чи домашню автоматизацію.

Logitech Harmony [2] – використовуючи безкоштовну супутню програму, пристрій є розумним центром, який дає змогу контролювати на основі смартфона до 8 домашніх розваг та інших пристроїв автоматизації. Це фізична центральна частина будь-якої розумної розважальної установки, ключовий компонент, який пов'язує все разом (рисунок 1.3).

Рис. 1.3 – Домашній концентратор Logitech

На додаток до керування тим, що відтворюється на екрані або відтворення з динаміків, він може взаємодіяти з розумними лампами, термостатами та іншими елементами першої необхідності, щоб дати контроль над усім будинком.

Amazon Echo [6] – це центр розумного дому (рисунок 1.4), створений навколо голосу. Він може не тільки відтворювати музику через вбудовані програми або потокове передавання по Bluetooth. Echo може керувати іншими розумними пристроями по всьому дому, від освітлення та вимикачів до обігрівачів або будь-чого іншого, що можна підключити.

Рис. 1.4 – Amazon Echo

Пульт NEEO Thinking Remote (рисунок 1.5). Завдяки вбудованій базі даних із понад 30 000 пристроїв, до яких він може підключатися та керувати, NEEO розроблено для спрощеного, централізованого керування всім розумним будинком. Крім цього, пульт дистанційного керування пропонує розпізнавання рук і сенсорний екран з високою роздільною здатністю, щоб персоналізоване автоматизоване керування було максимально прямим і простим. Також наявна функція SOS.

Рис. 1.5 – NEEO Thinking Remote

1.1.3 Розумна побутова техніка

Завдяки технології розумного дому техніка може стати більш зручною, безпечною та ефективною.

LG Smart ThinQ – це електрична одиночна духовка з інфрачервоним грилем (рисунок 1.6). Працюючи зі спеціалізованим додатком для смартфона, цю піч можна програмувати та керувати з будь-якої точки світу. Наприклад змінити температуру, режим та будь-які інші налаштування можна за допомогою розширеної панелі керування або смартфона[7].

Рис. 1.6 – LG Smart ThinQ

Найгірше в приготуванні їжі з духовкою – це очікування попереднього розігріву. Піч Café від GE усуває цю проблему, дозволяючи вмикати духовку, попередньо розігрівати, встановлювати таймери та змінювати температуру з будь-якого місця.

Рис. 1.7 – Café

Плита Wemo Enagbled Crock-Pot – плита від компанії Crock-Pot (рисунок 1.8), за допомогою супутнього додатка можна керувати температурою, таймерами та плануванням за допомогою свого смартфона з будь-якого місця, дозволяючи починати готувати, розігрівати вже готову їжу та планувати вечерю. Це навіть приносить спокій тим, хто турбується про те, щоб обід не готувався повільно, поки вони на роботі.

Рис. 1.8 – Wemo Enagbled Crock-Pot

1.1.4 Модулі та компоненти розумного дому

Завдяки технології розумного дому комунальні послуги можуть стати більш зручними, ефективними та безпечними.

Blossom – розумний контролер поливу керувати вашими дощувальними машинами з будь-якого місця, використовуючи розумні технології, щоб поливати ваш ландшафт якомога ефективніше (рисунок 1.9).

Рис. 1.9 – Blossom

За допомогою супутнього додатка для смартфона ви зможете контролювати використання води з будь-якої точки світу, встановлюючи різні поливи для різних присадибних зон.

Кондиціонер Aros – один з перших розумних кондиціонерів (рисунок 1.10). Звичайні віконні блоки змінного струму дають лише два варіанти: залишити повітря увімкненим на цілий день, витрачаючи електроенергію та гроші, або повернутися додому в жаркий і вологий будинок. Замість цього Aros використовує збір інформації, щоб дізнатися про моделі та звички, адаптуючи його роботу до будинку та способу життя.

Рис. 1.10 – Aros

Використовуючи в поєднанні з додатком для смартфона Wink, Aros може автоматично підтримувати потрібну температуру в потрібний час, максимізуючи економію в процесі. Крім того, ним можна керувати та відстежувати з будь-якої точки світу.

FortrezZ Z-Wave – бездротовий водяний клапан, відображений на рисунку 1.11. Завдяки цьому клапану, підключеному до водопроводу

будинку, з’являється можливість дистанційно керувати водопостачанням за допомогою системи домашньої автоматизації Z-wave[16]. Виїжджаючи з дому у відпустку, можна бути спокійним, знаючи, що з технікою все гаразд[8].

Рис. 1.11 – FortrezZ Z-Wave

Наприклад, якщо пристрій протікає, датчики затоплення можуть виявити це, автоматично перекриваючи подачу води, щоб обмежити пошкодження. Тим часом прийде сповіщення по телефону, щоб можна було розібратися в проблемі. Це ідеальне рішення для будинків для відпочинку, оренди та навіть комерційних будівель, його можна застосовувати до посудомийних машин, водонагрівачів, ванн, холодильників тощо.

1.2 Інтелектуальні IoT платформи

Internet of Things – це багаторівнева платформа, яка здійснює безпосереднє управління, автоматизацію та надання підключених пристроїв у межах Інтернет речей. Працює за допомогою хмарних, охоронних та експертних систем, підключених до апаратних пристроїв. Для початківців та розробників це готова платформа, доступна для миттєвого використання, яка працює з великою швидкістю.

Платформа IoT відіграє важливу роль у постачальниках та стартапах інтелектуальних пристроїв, які можуть використовувати її для встановлення продуктів із дистанційним керуванням, моніторингом у реальному часі, хмарними сервісами та інтеграції з використанням смартфонів та інших пристроїв.

Інтелектуальні речі використовують штучний інтелект для отримання додаткової інформації зі сховищ.

Комбінація хмарних та периферійних обчислень забезпечує переваги хмарної моделі, яка розділяє та розподіляє між підключеними пристроями масштабованим чином. Доведено, що Інтернет речей ознаменував успішне зростання DevOps.

Архітектура платформи IoT включає чотири рівні:

- речі;
- зв'язок;
- основні атрибути IoT;
- аналітика і програми.

Найвищий рівень стека IoT використовується для розробки підключених пристроїв та інтелектуальних речей. Рівень зв'язку використовується для обміну повідомленнями із підключеними пристроями. Основний рівень використовується для управління конфігурацією, OTA-сервісів та обміну повідомленнями. Вершина основного

рівня IoT – це аналітичний механізм правил та системи безпеки, які виявляють аномалії користувальницьких рішень IoT.

Деякі переваги платформ IoT: масштабованість, настроювання та безпека. Удосконалена платформа IoT забезпечує гнучку масштабованість відповідно до вимог клієнта. Як правило, розробник вимагає строгого контролю над всією інфраструктурою, тому він забезпечує прямий доступ до вихідного коду, систем інтеграції, системи розгортання, механізмів підключення та безпеки тощо. Також пропонує наскрізне шифрування потоку, включаючи дані у стані спокою, автентифікацію пристрою , управління ідентифікаторами користувачів та інфраструктура приватної хмари

Нижче наведено деякі з доступних на ринку платформ для створення програм IoT.

- Microsoft Azure IoT – одна з найпередовіших IoT-платформ і продуктів Microsoft, створена для надання сервісів, сервісів додатків, засобів обміну повідомленнями, сервісів зберігання даних і ефективних баз даних. Мета Azure IoT – забезпечити постійне обслуговування за рахунок ефективного зберігання даних та допомогти компанії швидко та точно надати рішення з доступними даними у сховищі у кожному підрозділі підприємства.
- Веб-сервіси Amazon – дозволяє користувачам використовувати хостинг та керувати послугами у кіберпросторі. Деякі організації використовують AWS для створення, розміщення, управління та організації інфраструктури компанії. Тут платформа IoT пропонує користувачам безліч переваг хмарного зберігання даних, передачі програм та управління сховищем.
- Google Cloud Platform – популярна платформа IoT, що надає базу даних із документами, хмарні обчислення та еластичну базу

даних. Це третя за величиною платформа IoT, що підтримує машинний інтелект, аналітику Google тощо.

- ThingWorx – платформа IoT, що широко використовується, придбана PTC в 2013 році. Розробники додатків використовують цю платформу для простого та ефективного виконання, аналізу складних даних та контенту та пропонують рішення для швидкого управління даними.
- Cisco IoT Cloud Connect – надає послуги з ефективним з'єднанням для передачі даних та голосу, надійним життєвим циклом SIM-карти, ефективним керуванням IP-сеансами та білінговою системою, що настроюється.
- HP Universal things – забезпечує точні вирішення всіх проблем, пов'язаних із IT. Він ефективно використовується для монетизації, ефективного збору даних, аналізу даних та правильної платформи для запуску нових мобільних додатків.
- Хмарна платформа SAP – ефективно застосовується для віддалених пристроїв, які підключені подвійним чином або до прямих хмарних сервісів. Це обладнання з компонентами для створення та управління IoT-додатком.
- Оракул Інтернет речей – встановлює зв'язок між програмним забезпеченням компанії та її політикою. Це гнучке середовище для розробки нового додатка та використання його в комерційних цілях.
- Bosch IoT Suite – розроблений у Німеччині та відрізняється надійними рішеннями та інноваційним підходом. Він доступний у середовищі з відкритим вихідним кодом.
- IBM Watson Інтернет речей – пропонує такі послуги, як керування віддаленими пристроями, передача даних без будь-яких перешкод

та хмарне сховище. Він пропонує кілька варіантів використання платформи IoT для машинного навчання та штучного інтелекту.

1.3 Додатки для роботи з стандартними системами IoT

Додаток для домашньої автоматизації може бути двох типів – однозадачний і багатозадачний. Однозадачні програми створені лише для одного конкретного пристрою, припустимо, для певної системи безпеки будинку, ці програми не допоможуть керувати будь-якою іншою системою Інтернету речей, яка може бути вдома, ці програми працюватимуть лише з системами, які сумісні з ними.

Для цих цілей є багатозадачна програма – ці програми можуть допомогти керувати всією системою IoT, яку встановили у домі. Кожним розумним домашнім пристроєм у домі можна керувати за допомогою багатозадачної програми.

Тепер, яким додатком для керування розумним будинком буде користуватися споживач, залежить від його уподобань. Деякі віддають перевагу однозадачним додаткам, думаючи, що вони забезпечать кращу безпеку та безпеку, тоді як інші віддають перевагу простоті та зручності багатозадачної програми.

1.3.1 Amazon Alexa

Amazon Alexa вважається одним з найкращих додатків для керування домом (рисунок 1.12).

Цей додаток допоможе налаштувати пристрої з підтримкою Amazon, створювати списки покупок, слухати музику. Додаток Alexa призначений не лише для пристроїв Amazon, але також існує багато інших пристроїв,

сумісних із цим додатком, наприклад Philips Hue Smart Bulbs, Wemo smart plug, Lutron Caseta, термостат Nest Learning, iRobot Roomba 690 та багато інших[14].

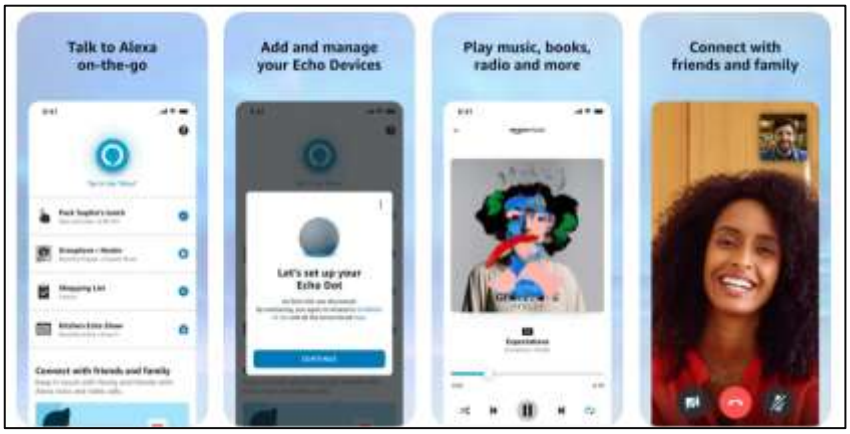


Рис. 1.12 – додаток Alexa

Налаштування програми Alexa на смартфоні є дуже простим процесом завдяки його легкому інтерфейсу розумного дому. Щоб розбудити Алексу, все, що потрібно зробити, це назвати її ім’я. Функція голосового керування дозволяє користувачам керувати всіма розумними пристроями, підключеними до Alexa, за допомогою усних команд.

Однією з найкращих характеристик домашньої автоматизації Alexa є альтернативна система керування. Ті, хто боїться використовувати систему голосового керування через безліч невдач , які траплялися з часом, можуть скористатися альтернативним інтерфейсом Alexa. Це служить бонусним фактором успіху Alexa.

1.3.2 SmartThings

SmatThings дозволяє користувачеві контролювати всі аспекти будинку з підтримкою IoT. А з центром SmartThings ви також отримуєте додаток SmartThings (рисунок 1.13).

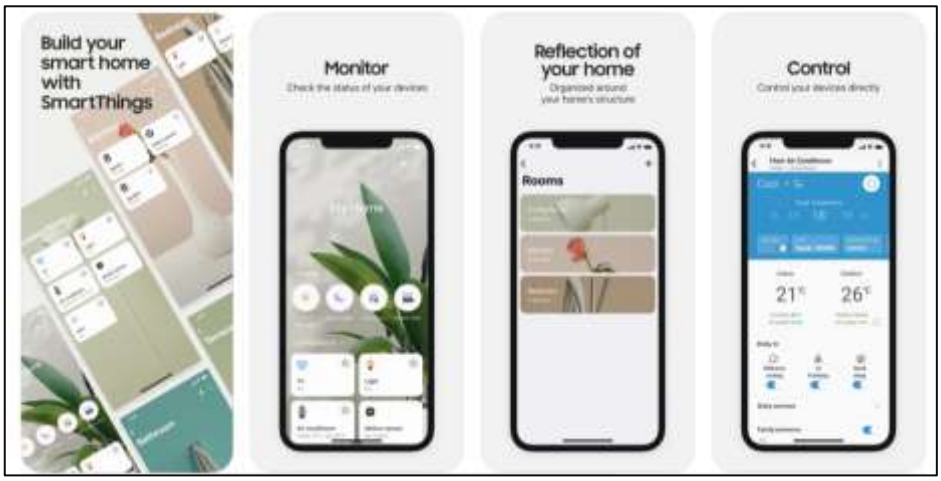


Рис. 1.13 – додаток SmartThings

Додаток SmartThings може допомогти користувачам отримати доступ до кількох пристроїв швидше, ніж будь-коли. Завдяки цьому можна керувати домашніми пристроями IoT, такими як холодильник, пральна та сушильна машина, кондиціонер для посудомийної машини тощо.

Однією з головних ключових функцій цього додатка є можливість дистанційно керувати та перевіряти стан пристроїв, групувати кілька пристроїв разом, щоб керувати ними одночасно, налаштовувати параметри пристрою, отримувати сповіщення про різні пристрої.

Багато пристроїв розумного дому сьогодні підтримуються пристроями Samsung SmartThings. Ці пристрої включають розумний термостат Ecobee, камеру безпеки Netgear Arlo без проводів Pro, систему безпеки будинку Yale Assure Lock, а після 2020 року також пристрої Google Nest . Це означає, що якщо вдома є концентратор Samsung SmatThings, а на телефоні програма Samsung Smart Home, не доведеться використовувати кілька програм для різних пристроїв.

Успіх програми SmartThings залежить від багатофункціональності та ефективного дизайну нової та покращеної програми. Однак, будучи повністю підключеним до хмари, він не надсилає важливі сповіщення у разі збою.

1.3.3 Google Home

Google Home – додаток від технологічної компанії Google (рисунок 1.14).

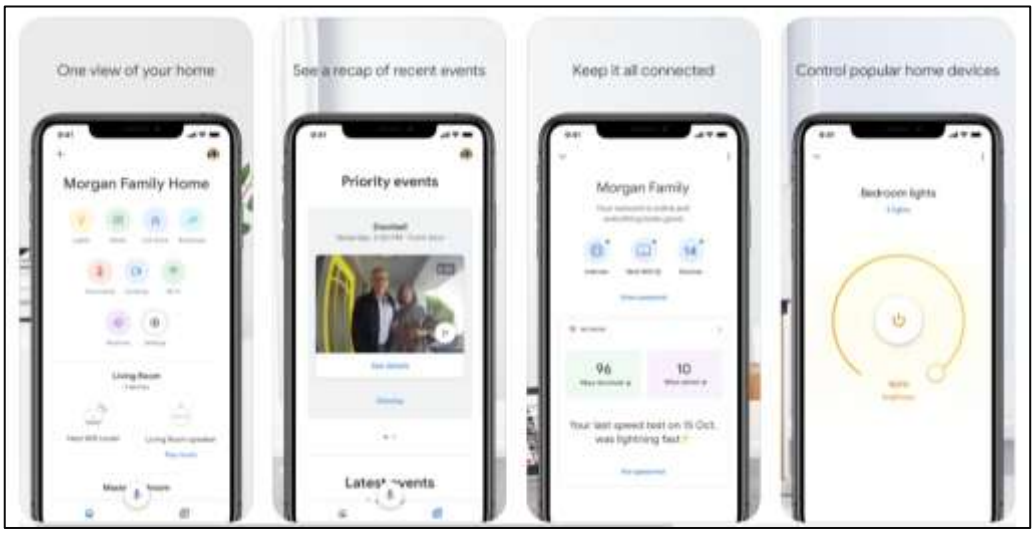


Рис. 1.14 – додаток Google Home

Популярна як програма для розумного дому для Android, так і на платформі iOS, програма Home дозволяє налаштувати пристрої Google Home і Chromecast, а також багато інших підключених домашніх пристроїв, як-от світильники, камери та багато іншого. Пристроями сторонніх розробників, які сумісні з програмами Google Home, є термостат Nest, Philips Hue, Samsung SmartThings, Honeywell Home, WeMo тощо.

Програма Google Home не лише для Android для розумного дому. Додаток Google Home дає змогу керувати, організовувати й керувати сумісними лампами й колонками, а також керувати ними лише одним або

двома натисканням. Додаток надасть ярлики для більшості дій, таких як увімкнення освітлення, перевірка погоди, отримання найцікавіших новин та інші дії, і все це за допомогою простої команди. Ця легкість та ефективність програми з часом персоналізуються. Завдяки своїй універсальності програма Google Home є однією з найкращих програм для автоматизації розумного дому на ринку зараз[13].

1.3.4 Apple Home

Домашній комплект Apple, ймовірно, є однією з найповніших систем автоматизації розумного дому на ринку, а додаток для домашньої автоматизації iOS – ідеальний компаньйон для нього. Він призначений для роботи як з пристроями розумного дому Apple, так і з іншими пристроями розумного дому.

Додаток Apple Home Kit можна використовувати будь-де, будь то iPhone, iPad, MacBook. Додаток оснащено інформаційною панеллю розумного дому, яка полегшує роботу з кожним аспектом розумного будинку, на рисунку 1.15 зображено основні екрани роботи з цим додатком[15].

Рис. 1.15 – додаток Apple Home Kit

Додаток дозволяє користувачам створювати «сцени», що дозволяє їм виконувати кілька дій лише одним натисканням на своєму смартфоні. Так само, як і вихід з дому: виходячи з дому, все, що потрібно зробити, це один раз натиснути на свій смартфон, щоб переконатися, що штори по всьому будинку закриті, світло вимкнено, а термостат налаштований на енергоефективний. На даний момент існує більше сотні брендів з пристроями, які підтримуються HomeKit.

Додатковими факторами успіху цієї програми є безпека та ефективність. Додавання нових пристроїв можливе прямо з головного екрана, що позбавляє від необхідності переміщатися по програмі. Крім того, додаток Apple Home Kit також є найбезпечнішим додатком для розумного дому, що забезпечує повну безпеку для більшості користувачів.

1.4 Висновки за першим розділом

На ринку сучасних рішень наразі представлений досить широкий спектр технологій розумного дому, що обумовлено постійно зростаючим попитом на рішення для Smart Home.

Базовою технологією розумних житлових комплексів можна вважати Internet of Things – багаторівневу платформу, яка здійснює безпосереднє управління, автоматизацію та надання підключених пристроїв у межах Інтернета речей. Серед найбільш популярних рішень можна відмітити Holî SleepCompanion, Flux Bluetooth, Smart LED Sengled, Boost Logitech, Harmony, Amazon Echo, LG Smart ThinQ, FortrezZ Z-Wave та ін.

В якості інтерфейсу користувача можуть виступати застосування двох типів – однозадачний і багатозадачний. Яким саме додатком для керування розумним будинком буде користуватися споживач, залежить від його уподобань. Деякі віддають перевагу однозадачним додаткам, думаючи, що вони забезпечать кращу безпеку та безпеку, тоді як інші віддають перевагу простоті та зручності багатозадачної програми. Типові представники подібного ПЗ є, наприклад, Amazon Alexa, SmartThings, Google Home, Apple Home та інші.

РОЗДІЛ 2
ВИБІР ЗАСОБІВ РОЗРОБКИ ТА РІШЕННЯ ЗАДАЧ ПРОЕКТУ

2.1 Основні системи розумного будинку

Кожна з систем розумного дому заснована на певній технології, яка використовується для обміну даними між усіма системними модулями комплексу.

Ці різні технології відрізняються за продуктивністю, масштабом, швидкістю передачі даних тощо. Деякі з цих технологій наведено нижче.

Interbus – це шина даних датчика / приводу. Він характеризується кільцевою топологією на основі інтерфейсу RS485, що приносить деякі переваги, а саме: можливість передачі даних всередині системи на довжину до 13 км і спрощення пошуку несправностей у критичних ситуаціях. Недоліком цієї системи є те, що при виході з ладу одного пристрою всі інші також виходять з ладу. Пристрої на шині не повинні мати адреси через топологію кільця, положення всередині кільця достатньо для ідентифікації. Швидкість передачі даних по цій шині становить до 500 кбіт/с.

P-NET: це тип шини master-slave. Він використовує інтерфейс RS485, а також можна використовувати інтерфейс TP або RS232, але зі зниженою швидкістю передачі даних. Максимальна довжина такої шини – 1200 м, а швидкість передачі даних – 76,8 кбіт/с.;

Profibus використовує інтерфейс RS485 або оптичний кабель як середовище передачі даних. Існує три варіанти цієї технології: Profibus-FMS (специфікація повідомлень fieldbus), Profibus-DP (децентралізовані периферійні пристрої) і Profibus-PA (автоматизація процесів), які охоплюють широкий спектр можливих застосувань і можуть співпрацювати один з одним.

CAN (Controller Area Network): спеціалізується на автомобільних додатках завдяки високій швидкості передачі даних 1 Мбіт/с і короткій довжині передачі даних 40 метрів. Під час передачі даних він використовує метод CSMA / CA (множинний доступ із визначенням несучого зв'язку з уникненням зіткнень).

LonWorks – це відкрите мережеве рішення для автоматизації та управління мережами, розроблене американською компанією Echelon. Він розроблений таким чином, щоб його можна було використовувати в централізованих контролерах автоматизації будівель, а також у децентралізованих компонентах управління будівлею. LONWORKS – це стандартизована система шин (ANSI/CEA-709.1-B і EN ISO/IEC 14908), яка дозволяє інтелектуальним пристроям спілкуватися один з одним через локальну мережу керування. LON означає локальну операційну мережу;

BACnet (Мережа автоматизації та управління будівель) – це стандартизований протокол передачі даних, розроблений Американським товариством інженерів з опалення, охолодження та кондиціонування повітря (ASHRAE) для використання в автоматизації будівель, щоб дозволити пристроям і системам обмінюватися інформацією[18]. BACnet використовується в багатьох системах автоматизації будівель по всьому світу і отримав міжнародний стандарт ISO 16484-5 у 2003 році. BACnet розвинувся через потребу в стандартизованому протоколі передачі даних, який дозволив би різноманітну автоматизацію та контроль компонентів у будівлі для зв'язку один з одним, забезпечення сумісності та незалежності виробника.

Z-Wave – це бездротова система нового покоління, спочатку розроблена компанією Zensys в Данії, яка дозволяє всім електронним елементам спілкуватися як один з одним, так і з користувачем бездротовим способом. Після перших установок у 2005 році було створено Z-Wave Alliance як консорціум компаній, які виготовляють підключені пристрої, керовані через програми на смартфонах, планшетах або комп'ютерах,

використовуючи технологію бездротової сітчастої мережі Z-Wave. Воно використовує радіосигнал з частотою 868.4МГц і максимальна випромінювана потужність близько 20 – 30 мВт. При цьому він використовує прості, надійні радіохвилі малої потужності, які легко проходять крізь стіни. Z-Wave використовує топологію мережі Mesh, тому вона може поширюватися набагато довше, ніж радіохвилі окремого пристрою. Система дуже проста і легка як в установці, так і в експлуатації, і практично не вимагає багато знань і часу для встановлення та початку експлуатації. Ця система використовується особливо для будинків з рішеннями «зроби сам».

Enocean – технологія, заснована на ефективному використанні найменших змін у навколишньому середовищі для отримання необхідної енергії, яка потім перетворюється в електрику, яка потім служить джерелом енергії для передачі радіочастотних сигналів від датчиків до приводів. Усі компоненти Enocean працюють без батарейок і призначені для роботи без технічного обслуговування. Найпростішим прикладом продукту, заснованого на РЧ-протоколі, є вимикач освітлення без батарейок і бездротового зв’язку. Натискаючи на такий перемикач, в нього вкладається стільки енергії, що перемикач виробляє стільки електроенергії, скільки йому потрібно для відправки слабкого радіочастотного сигналу на актуатор, щоб увімкнути світло. Або, повертаючи ручку на вікні, виробляється стільки енергії, що ручка посилає радіочастотний сигнал до клапана на радіаторі, який повідомляє йому, що вікно відкрите і що клапан повинен закритися. Система вимагає меншого часу підключення, тому що не потрібна проводка між вимикачем і світлом.

Gould Modicon, підтримується більшістю програмованих логічних контролерів. Він спирається на протокол Master/Slave. Він дуже простий і зручний у використанні, тому його використовують як виробники контролерів, так і виробники будівельного монтажного обладнання. Він став

дуже популярним, оскільки є безкоштовним. Він обмежений через простий обмін даними і тому не використовується для більш складних потреб.

X10 – це протокол для зв'язку між електронними пристроями, що використовуються для домашньої автоматизації[19]. В основному він використовує проводку лінії електропередачі для сигналізації та керування. Також визначено транспортний протокол на основі бездротового радіо. Він був розроблений в 1975 році для дистанційного керування побутовою технікою. Це була перша технологія домашньої автоматизації, яка досі доступна по всьому світу. На жаль, сьогодні він застарів, оскільки дуже повільний (близько 20 біт/с). Через невеликий набір команд і погану надійність він сьогодні не дуже поширений в Європі.

Специфікація ZigBee описує інфраструктуру та послуги, доступні для програм, що працюють на платформі ZigBee. Він визначає протокол зв'язку високого рівня, який використовує малопотужні цифрові радіосигнали на основі стандарту IEEE 802.15.4 для потреб бездротових мереж (медичні пристрої, димові та охоронні сигналізації, автоматизація будівель). Сама технологія простіша і дешевша, ніж з іншими бездротовими мережами, такими як Bluetooth. Найпотужніший вузол ZigBee містить лише 10% програмного забезпечення, що міститься в типовій Bluetooth або звичайній бездротовій мережі, а найпростіший лише 2%. На даний момент між ZigBee та комісією BACnet тривають домовленості щодо встановлення з’єднання між дротовим та бездротовим відкритим протоколом[17].

DALI (DALIa, DALIb) – це протокол цифрового зв'язку, розроблений спеціально для схем освітлення. DALI дуже підходить для створення сцен і отримання відгуків про несправні лампи. Тому він дуже підходить в поєднанні з автоматизацією будівель, де потрібен дистанційний моніторинг та звіти про обслуговування. DALI був представлений у 1999 році виробниками електронних баластів, які хотіли стандартизований цифровий протокол керування електронними баластами. Він зроблений так, що його

дуже легко встановити і налаштувати. Всі приводи, контролери та датчики з'єднані один з одним одним кабелем зв'язку. Система DALI складається з електронних баластів, перемикачів, датчиків, інтерфейсів керування та контролерів 1-10 В. DALI – це не система автоматизації будівель, а лише система управління освітленням.

DMX 512/1990 є стандартом цифрової передачі даних для диммерів і контролерів, що працюють у режимі постійного струму. Він може керувати до 512 каналами. Дані передаються пакетами. Кожен пакет оновлює всі вбудовані пристрої. Кожен пакет складається з до 513 кадрів, які позначають початок і кінець кожного пакета. Ми не отримуємо доступ до пристроїв безпосередньо, але інформація, яку ми надсилаємо на пристрій, визначається в конкретній структурі кожного пакета. Стандарт DMX широко використовується в театрах, дискотеках і клубах, оскільки багато спеціальних ламп, стробоскопів, протитуманних апаратів та інших пристроїв реалізують його.

Konnex – технологія була стандартизована та визнана в усьому світі європейськими правилами (EN50090), китайськими (GB/T 20965), американськими правилами (ANSI/ASHRAE 135) та світовими нормативами (ISO/IEC 14543-3)) у світі не існує лише одного виробника, який виробляє елементи KNX. Натомість сьогодні в асоціації Konnex зареєстровано понад 419 виробників. Разом ці виробники виробляють понад 7000 різних елементів KNX. Це кількість виробників забезпечує 100-відсоткове забезпечення елементами KNX протягом усього терміну експлуатації будинку.

2.1.1 Принципи побудови системи

Існує 3 підходи до організації управління системами автоматизації: централізовані, децентралізовані та змішані системи[20].

Суть централізованого розумного будинку полягає в тому, що йде програмування лише одного центрального логічного модуля. Зазвичай це вільно програмований контролер, в який записується заздалегідь спеціально створена під об'єкт програма, основі якої йде управління виконавчими пристроями та інженерними системами. Це дозволяє використовувати широкий вибір обладнання та складні сценарії.

Переваги:

- єдиний інтерфейс керування;
- створення складних сценаріїв, прив'язаних до часу доби;
- станом мешканця, температурі, місячному циклі;
- Простота початкового налаштування. Усі маніпуляції виконуються лише на центральному контролері.

Недоліки:

- потрібне обов'язкове програмування системи;
- залежність системи від центрального контролера;
- потребує наявність резервного обладнання.

У децентралізованих системах автоматизації кожен виконавчий пристрій містить у собі мікропроцесор з енергонезалежною пам'яттю. Цим пояснюється надійність таких систем. При виході з експлуатації одного пристрою вся система працює справно, крім приладів, підключених до цього пристрою.

Переваги: висока надійність.

Недоліки:

- висока вартість обладнання;
- складне первісне програмування обладнання;
- складність інтеграції коїться з іншими апаратними рішеннями;
- часто має закритий протокол керування.

Система, що найчастіше зустрічається, на сьогоднішній день – це система змішаного управління. У ролі центрального управління стоїть

контролер, але управляючі модулі мають убудовані функції управління. Таким чином, при виході з ладу центрального контролера всі життєво важливі системи переводять на ручне управління.

Переваги:

- висока надійність;
- відносно не висока вартість обладнання;
- легкість у початковому налаштуванні;
- не потребує резервного обладнання.

2.1.2 Способи побудови комутуючого середовища

Для того, щоб контролер міг передавати сигнали управління датчикам і виконавчим пристроям, потрібно вибрати яким чином буде здійснено спілкування пристроїв між собою. Є два основних типи передачі сигналу – провідний та бездротовий.

Дротові системи автоматизації – це коли всі керуючі пристрої – датчики, вимикачі, пристрої керування кліматом, керуючі панелі зв'язуються єдиною провідною інформаційною шиною, якою йдуть сигнали до виконавчих пристроїв, розташованих в електричному щиті. Як провідної інформаційної шини використовуються спеціальні кабелі, а в окремих випадках звичайна кручена пара. Дротова система має свої переваги і недоліки.

Переваги:

- висока швидкість відгуку;
- великий вибір дизайну елементів, що управляють;
- відносна простота інтеграції з іншими системами;
- передача керуючих сигналів великі відстані.

Недоліки:

- проектування на стадії ремонту;

- потрібне прокладання великої кількості проводів;
- для встановлення та обслуговування системи потрібна висока кваліфікація інсталятора.

У бездротових системах автоматизації сигнал від керуючих пристроїв до виконавчих йде по радіоканалу. Це дозволяє скоротити кількість дротів, а також час на інсталяцію системи. Ці системи можна монтувати на об'єкти з готовим ремонтом із класичною проводкою. Бездротовий вимикач може бути радіопередавачем, який зв'язується з усіма іншими вимикачами.

Переваги:

- можна встановлювати в квартири та будинки з уже готовим ремонтом із класичною проводкою;
- невелика кількість дротів.

Недоліки:

- погана перешкодозахисність;
- потрібний витратний матеріал у вигляді батарейок;
- складна інтеграція з іншими системами;
- низька безпека передачі інформаційних повідомлень;
- складності реєстрації деяких радіочастот;
- складність передачі радіосигналу через товсті перегородки та великі відстані.

Усі системи можуть мати відкритий і закритий протокол управління. Відкритий протокол дозволяє користувачеві самому розробляти програмне забезпечення та нове обладнання. Є можливість повної модифікації всієї системи. Закритий протокол не дозволяє змінювати або вбудоване програмне забезпечення модифікацію, але за це користувач може розраховувати на хорошу техпідтримку і нижчу вартість.

2.2 Фреймворки та платформи для написання системи управління розумним будинком

Фреймворк – це платформа для розробки програмних додатків. Він забезпечує основу, на якій розробники програмного забезпечення можуть створювати програми для певної платформи. Наприклад, фреймворк може включати попередньо визначені класи та функції, які можна використовувати для обробки введених даних , керування апаратними пристроями та взаємодії з системним програмним забезпеченням. Це спрощує процес розробки, оскільки програмістам не потрібно винаходити велосипед щоразу, коли вони розробляють нову програму.

Фреймворк схожий на інтерфейс прикладного програмування (API), хоча технічно фреймворк включає API. Він також може включати бібліотеки коду, компілятор та інші програми, що використовуються в процесі розробки програмного забезпечення.

Існує кілька різних типів програмних фреймворків. Популярні приклади включають ActiveX і .NET для розробки Windows, Cocoa для Mac OS X, Cocoa Touch для iOS, Android Application Framework для Android, Angular для веб-додатків. Комплекти для розробки програмного забезпечення (SDK) доступні для кожної з цих платформ і включають інструменти програмування, розроблені спеціально для відповідної платформи. Наприклад, програмне забезпечення для розробки Xcode від Apple включає пакет SDK для Mac OS X, призначений для написання та компіляції програм для фреймворку Cocoa.

У багатьох випадках програмна платформа підтримується операційною системою. Наприклад, програма, написана для Android Application Framework, працюватиме на пристрої Android, не вимагаючи встановлення інших додаткових файлів. Однак для запуску деяких програм потрібна спеціальна структура. Наприклад, для програми Windows може знадобитися

Microsoft .NET Framework 4.0, який інстальовано не на всіх машинах Windows (особливо на комп'ютерах зі старими версіями Windows). У цьому випадку для запуску програми необхідно встановити пакет інсталятора Microsoft .NET Framework 4.

2.2.1 ReactJs

Створена в 2013 році Facebook, React є найбільш використовуваною бібліотекою відповідно до опитування State of JavaScript 2021. React технічно не є фреймворком; це бібліотека для компонентів інтерфейсу користувача, але в розмовній мові вона розглядається як фреймворк[9].

React.js забезпечує швидкий інтерфейс для інтерактивних програм, які ефективно використовують дані, миттєво застосовуючи зміни до елементів, а не оновлюючи всі випадки одночасно, як це роблять інші бібліотеки. Він також підтримує поетапне використання віртуальної DOM (об'єктна модель документа) для швидкого оновлення вмісту веб-сторінки. React також використовує JSX, інтерфейс домену, створений тією ж командою.

Особливості:

- Компоненти React: React ділить сторінку на кілька компонентів. Кожен компонент є частиною інтерфейсу користувача і має власну логіку та дизайн, що робить їх доступнішими та швидшими для повторного використання.
- Віртуальний DOM: об'єктна модель віртуального документа представляє дані в деревоподібній структурі. Він також поділяє дані на модулі, що містять вузол для кожного незалежного елемента інтерфейсу користувача, присутній в документі. Це

гарантує, що коли перезавантажується сторінка, вона оновлює лише її частину, а не весь веб-сайт.

- JSX: JavaScript XML – це синтаксис, подібний до HTML, який обробляється у виклики JavaScript, що дозволяє розробникам вбудовувати об'єкти JavaScript в елементи HTML.
- Декларативний інтерфейс користувача: синтаксис у стилі HTML JSX дозволяє контролювати потік і стан у програмі, вирішуючи, як має виглядати компонент.

2.2.2 Vue.Js

Vue був створений у 2014 році Еваном Ю., колишнім співробітником Google. Він має на меті стати фреймворком, який поєднує найкращі можливості Angular і React. Vue зберігає синтаксис шаблону Angular і прив’язування даних React, реквізити та підхід на основі компонентів[11].

Vue дозволяє розробникам із знаннями HTML, CSS і JavaScript створювати SPA і кросплатформні програми корпоративного рівня та інтегрувати їх у нові або вже існуючі проекти за допомогою HTML або JSX.

Особливості:

- Анімовані переходи: Vue надає різні способи застосування ефектів переходу до елементів HTML під час додавання або оновлення їх із DOM. Є можливість інтегрувати сторонні бібліотеки анімації для ще більшої інтерактивності.
- Шаблони: Vue надає шаблони HTML, які пов’язують DOM з даними екземпляра Vue. Він компілює шаблони у віртуальний DOM у вигляді HTML, який можна проаналізувати та відтворити всіма браузерами.
- Маршрутизація Vue: маршрутизація дозволяє користувачам перемикатися між сторінками, не оновлюючи сторінку, що робить навігацію простішою та швидшою.

– Директиви: директиви – це інструкції, які прив’язують себе до властивості екземпляра Vue. Вони дозволяють VueJS маніпулювати візуальною частиною програми, щоб змінити те, що користувач бачить перед нею.

2.2.3 Angular

Angular – це платформа та фреймворк для створення односторінкових клієнтських програм за допомогою HTML і TypeScript. Angular написаний на TypeScript. Він реалізує основні та додаткові функції у вигляді набору бібліотек TypeScript, які імпортуєте у свої програми[10].

Архітектура Angular спирається на певні фундаментальні концепції. Основними будівельними блоками фреймворка Angular є компоненти Angular, які організовані в NgModules. NgModules збирає пов'язаний код у функціональні набори; Angular додаток визначається набором NgModules. Додаток завжди має принаймні кореневий модуль, який дозволяє завантажувати, і зазвичай має набагато більше модулів функцій.

Компоненти визначають перегляди, які є наборами елементів екрану, які Angular може вибирати та змінювати відповідно до логіки та даних програми.

Компоненти використовують служби, які надають певну функціональність, не пов’язану безпосередньо з представленнями. Постачальники послуг можуть бути введені в компоненти як залежності, що робить код модульним, багаторазовим та ефективним.

Модулі, компоненти та служби – це класи, які використовують декоратори. Ці декоратори позначають свій тип і надають метадані, які повідомляють Angular, як їх використовувати.

Метадані для класу компонента пов’язують його з шаблоном, який визначає подання. Шаблон поєднує звичайний HTML з директивами Angular і розміткою прив’язки, які дозволяють Angular змінювати HTML перед відображенням його для відображення.

Метадані для класу служби надають інформацію, необхідну Angular, щоб зробити її доступною для компонентів за допомогою ін'єкції залежностей (DI).

Компоненти програми зазвичай визначають багато представлень, розташованих ієрархічно. Angular надає Router послугу, яка допоможе визначити шляхи навігації між представленнями. Маршрутизатор надає складні навігаційні можливості в браузері.

Це робить його найкрайщим варіантом для створення та розгортання CRM-систем.

Angular поєднує в собі декларативні шаблони, впровадження залежностей, надійні наскрізні інструменти та інтегрований набір найкращих практик для вирішення проблем розробки.

Особливості:

- Двостороннє прив’язування даних, що представляє рівень моделі, тому якщо змінити модель, користувачі зможуть автоматично бачити зміни в моделі перегляду, що скорочує час розробки.
- Впровадження залежності. Ця парадигма програмування дозволяє класам, компонентам і модулям працювати взаємозалежно, зберігаючи узгодженість коду, зменшуючи частоту змін класу.
- Архітектура MVC. Цей тип архітектури ізолює логіку програми від рівня UI, підтримуючи розділення проблем і заощаджуючи час на кодування.

2.2.4 Node.Js

Node.Js – програмна платформа, заснована на двигуні V8, що перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення. Node.js представляє цікавий перетин фронтенду та бекенд-технологій. Побудований на JavaScript, який є мовою веб-скриптів на стороні клієнта, Node.js розширює можливості JavaScript для роботи на серверній частині архітектури веб-додатків, а також для архітектури без сервера.

Node.js робить це, виконуючи у власному середовищі виконання на сервері. Гарною аналогією для проведення є порівняння між Node.js і Java, яка також запускає програми в межах власного середовища виконання. Середовище виконання Node.js розроблено, щоб бути легким і ефективним, з неблокуючим введенням-виводом і менеджером пакетів, щоб зробити створення програм в екосистемі Node.js ще простіше.

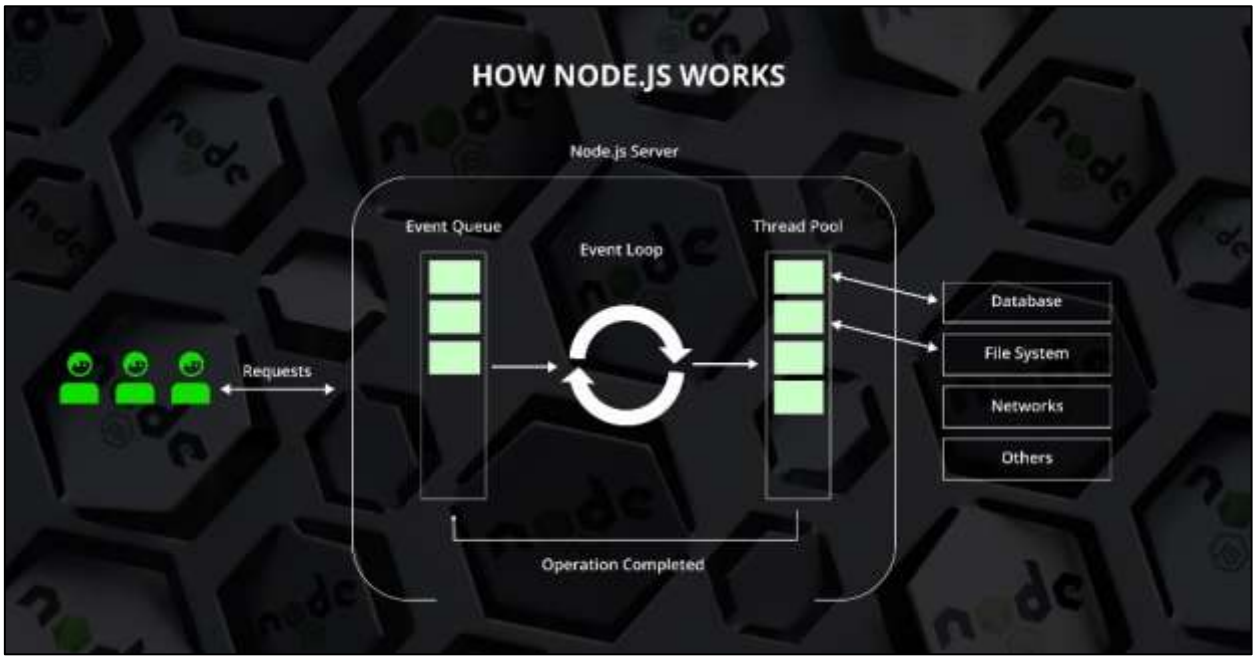


Рис. 2.1 – Схема роботи Node.Js

Менеджер пакетів для Node.js називається npm. Його мета – служити індексом бібліотек, створених спільнотою розробників Node.js, які легко надаються та імпортуються іншими проектами. Ці пакети надають корисні

рішення для поширених функцій і коду, які спрощують створення нових проектів і покращення старих.

Node.js є застосовним рішенням для багатьох різних типів випадків використання. Як фреймворк на стороні сервера, Node.js підходить для додатків у внутрішній частині технологічного стеку.

Розуміння розмірів продукту є важливим для вибору правильної технології для його створення. Гнучка й ефективна природа Node.js дозволяє створювати невеликі, швидкі та масштабовані програми. Додатки в режимі реального часу, такі як миттєві повідомлення та інструменти для співпраці, є одним із прикладів цього. Це, у поєднанні з можливостями швидкої синхронізації Node.js, також робить його корисним для програм на основі подій. Прикладами є програми, які використовують WebSockets і WebRTC.

Завдяки своїй гнучкості Node.js дуже добре підходить для створення як безсерверних, так і мікросервісних програм. Ці стилі дизайну дуже популярні як для економії ресурсів, так і для ефективного керування життєвими циклами додатків.

Безсерверна архітектура популярна завдяки своїй здатності заощаджувати обчислювальні витрати, вимагаючи лише ресурсів, необхідних для роботи програми без додаткових витрат. Ці програми мають легку вагу. Це робить інтеграцію Node.js із безсерверною архітектурою чудовою комбінацією. Існують пакети npm для безсерверного проектування, і створення безсерверної програми Node.js добре працює із загальною архітектурою мікросервісів у випадках використання C2C та B2C, коли навантаження на сервер нестабільна.

Node.js дуже добре обробляє одночасні з'єднання. Оскільки IoT побудований на багатьох пристроях, які надсилають невеликі повідомлення, які потрібно швидко обробляти, Node.js є хорошим бекендом для таких видів додатків, забезпечуючи безсерверну архітектуру та підтримку зв’язку в реальному часі[12].

2.3 Висновки за другим розділом

Додатки для розумних будинків можуть бути побудовані на базі різноманітних рішень. Додатки для систем розумних будинків побудованих на Angular та Node.Js досить популярні у світі, через свою доступність, та те що обидва ці фреймворки побудовані на базі JavaScript. Багато професійних систем побудовані саме на цих платформах. Такий вибір виникає через велику кількість документації та навчального матеріалу.

Протягом тривалого часу Angular розглядався професійними розробниками як інтерфейсний інструмент через його універсальність і здатність поєднувати бізнес-логіку та елементи інтерфейсу. Він залишається однією з найпопулярніших технологій розробки програмного забезпечення, що пропонує різноманітні функції, які допомагають масштабувати, оптимізувати та прискорювати платформи в Інтернеті. Поряд з окремими технологіями мобільної розробки, він також використовується для створення інтерфейсної частини кросплатформених додатків.

Завдяки такому зв’язку як Angular + Node.Js, з’явився стек під назвою MEAN – це безкоштовний стек програмного забезпечення JavaScript з відкритим вихідним кодом для створення динамічних веб-сайтів і веб-додатків [3].

Оскільки всі компоненти стеку MEAN підтримують програми, написані на JavaScript, програми MEAN можуть бути написані однією мовою як для серверного, так і для клієнтського середовища виконання.

Незважаючи на те, що їх часто порівнюють безпосередньо з іншими популярними стеками веб-розробки, такими як стек LAMP , компоненти стека MEAN є більш високими, включаючи рівень презентації веб-додатків і не включають рівень операційної системи.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ КЕРУВАННЯ КОНТРОЛЕМ ДОСТУПУ

3.1 Вибір мікроконтролеру

Для практичної реалізації проекту було обрано платформу Arduino. Серед широкої лінійки мікроконтролерів було прийнято рішення у користь Arduino UNO. Для створення більш бюджетного проекту було обрано Atmega328 Maker UNO (рисунок 3.1). Цей мікроконтролер є повним аналогом Arduino UNO.

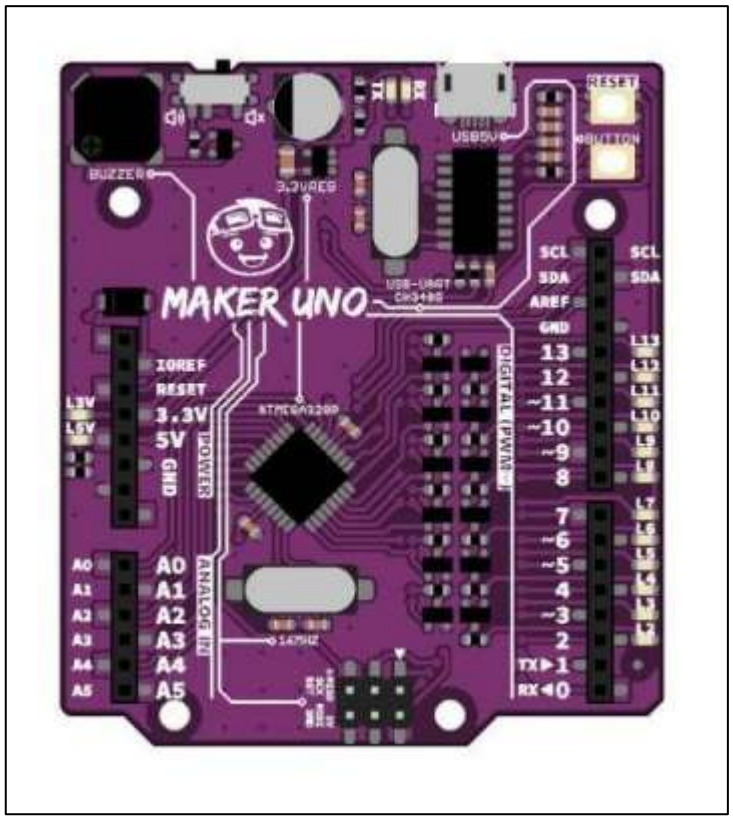


Рис. 3.1 – Atmega328 Maker UNO

Він сумісний з Arduino UNO R3, тому Maker UNO можна запрограмувати через Arduino IDE. Він сумісний з усіма прикладами коду та бібліотеками для Arduino UNO, що не доставить труднощів у розробці.

ESP8266 (рисунок 3.2) – мікроконтролер китайського виробника Espressif з інтерфейсом Wi-Fi.

Рис. 3.2 – ESP8266

Крім Wi-Fi мікроконтролер відрізняється можливістю виконувати програми з зовнішньої флеш-пам'яті з інтерфейсом SPI. Дана плата використовується для програмування, контролю, управління та підключення до програмного забезпечення, що виконується на комп'ютері.

3.2 Вибір фреймворку та бібліотек для написання веб-додатку

Для написання веб-додатку було обрано фреймворк Angular, цей фреймворк є найкращим рішенням для створення закритих додатків.

Бібліотекі які використовувалися:

- Chart.js
- Bootstrap
- RxJs

Chart.js – це безкоштовна бібліотека JavaScript з відкритим вихідним кодом для візуалізації даних , яка підтримує вісім типів діаграм:

- стовпчаста;
- лінія;
- область;
- пиріг(пончик);
- бульбашка;
- радар;
- полярна;
- розсіяна.

Створена лондонським веб-розробником Ніком Дауні в 2013 році, тепер бібліотека підтримується спільнотою і є другою за популярністю бібліотекою діаграм JavaScript на GitHub за кількістю зірочок після D3.js, яка вважається значно легшою у використанні, хоча менше настроюється, ніж D3.js. Chart.js відтворюється на полотні HTML5 і широко розглядається як одна з найкращих бібліотек візуалізації даних.

Bootstrap – це безкоштовна платформа веб-розробки з відкритим вихідним кодом. Він призначений для полегшення процесу веб-розробки адаптивних веб-сайтів, орієнтованих на мобільні пристрої, надаючи набір синтаксису для дизайну шаблонів. Іншими словами, Bootstrap допомагає веб-розробникам швидше створювати веб-сайти, оскільки їм не потрібно турбуватися про основні команди та функції. Він складається зі скриптів на основі HTML, CSS та JS для різних функцій і компонентів, пов’язаних із веб-дизайном.

Основною метою Bootstrap є створення адаптивних веб-сайтів, орієнтованих на мобільні пристрої. Це забезпечує оптимальну роботу всіх елементів інтерфейсу веб-сайту на всіх розмірах екрана.

Bootstrap доступний у двох варіантах: попередньо скомпільований і заснований на версії вихідного коду.

Bootstrap із вихідним кодом дозволяє отримати доступ до порту Sass . Це означає, що він створює власну таблицю стилів, яка імпортує Bootstrap, що дозволяє змінювати та розширювати інструмент за потреби.

Також можна встановити Bootstrap за допомогою менеджера пакетів – інструмента, який керує та оновлює фреймворки, бібліотеки та активи.

Деякі з найпопулярніших менеджерів пакетів включають npm, Composer і Bower. Npm керує залежностями на стороні сервера, тоді як Composer зосереджується на інтерфейсі.

Деякі з компонентів інтерфейсу Bootstrap включають панелі навігації, системи сіток, каруселі зображень і кнопки.

Щоб зменшити час завантаження сторінки сайту, Bootstrap мінімізує файли CSS і JavaScript. Крім того, Bootstrap підтримує узгодженість синтаксису між веб-сайтами та розробниками, що ідеально підходить для командних проектів.

Bootstrap поставляється з попередньо визначеною системою сіток, що позбавляє від створення системи з нуля. Система сіток складається з рядків і стовпців, що дозволяє створити сітку всередині існуючої замість того, щоб вводити медіа-запити у файлі CSS.

Крім того, система сіток Bootstrap робить процес введення даних більш простим. Він містить багато медіа-запитів, що дозволяє визначити індивідуальні точки зупинки кожного стовпця на основі потреб веб-проекту.

RxJS (Reactive Extensions for JavaScript) – це бібліотека для реактивного програмування з використанням Observable, що полегшує створення асинхронного коду або коду на основі зворотного виклику. Реактивне

програмування – це парадигма асинхронного програмування, що стосується потоків даних і поширення змін.

RxJS забезпечує реалізацію Observableтипу, яка потрібна до тих пір, поки тип не стане частиною мови і поки браузери не підтримають його.

Бібліотека також надає допоміжні функції для створення спостережуваних і роботи з ними. Ці допоміжні функції можна використовувати для:

- перетворення існуючого коду для асинхронних операцій у спостережувані;
- ітерація значень у потоці;
- відображення значень до різних типів;
- фільтрація потоків;
- створення кількох потоків.

3.3 Алгоритм роботи програми

Arduino Uno підключений до ESP8266 у відношенні провідний-підпорядкований, надсилаючи дані до API. API зберігає дані в базі даних, а додаток Angular робить запит через протокол HTTP, щоб отримати дані та відобразити їх у вигляді діаграми або таблиці.

API було розроблено на Node.Js. Підключено до бази даних mongoDb:

```
const mongoose = require("mongoose");
mongoose.connect("mongodb+srv://localhost:8080:test@smar
 thome.mongodb.net/smarthome", { useNewUrlParser: true
})
    .then(console.log("Connected to mongoDB..."))
    .catch(() => {
        console.log("Connection to mongoDB refused...");
    })
mongoose.set("useCreateIndex", true);
module.exports = mongoose;
```

Для сигналізації було розроблено 5 запитів:

```
const Alarm = require('../models/alarm');
const _ = require('lodash');

exports.controller = {
```

Отримання статусу сигналізації:

```
async getStatus(req, res) {
    const status = await Alarm.findOne();
    return res.status(200).send({ alarm: status });
},
```

Створення тривоги:

```
async createAlarm(req, res) {
    const alarm = await Alarm.find();

    if (alarm.length === 0) {
        const newAlarm = new Alarm({
            status: 0
        })
        await
    newAlarm.save().then(res.status(200).send("Alarm
    created"));
    } else {
        res.send("Alarm already exists");
    }
},
```

Включення сигналізації:

```
async enableAlarm(req, res, next) {
    const alarm = await Alarm.findOne();

    if (!alarm) return next();

    alarm.status = 1;
    await alarm.save();

    return res.status(200).send("Alarm enabled");
},
```

Виключення сигналізації:

```
async disableAlarm(req, res, next) {
    const alarm = await Alarm.findOne();

    if (!alarm) return next();

    alarm.status = 0;
    await alarm.save();

    return res.status(200).send("Alarm disabled");
},
```

Отримання коротко статусу для esp:

```
    async getShortStatus(req, res, next) {
        const alarm = await Alarm.findOne();
        if (!alarm) return next();

        let shortStatus = "0";
        if (alarm.status) shortStatus = "1";

        return res.send(shortStatus);
    }
}
```

Для даних про температуру було розроблено 5 запитів:

```
const Data = require('../models/data');
const _ = require('lodash');
exports.controller = {
    // :id -> yyyy/mm/dd
    async findByDay(req, res, next) {
        let currentDay =
  _.lowerCase(req.params.currentDay);
        const data = [];

        await Data.find((err, records) => {
            if (!records) return next();

            records.forEach((record) => {
                const date = _.lowerCase(record.date);
                if (date === currentDay) {
                    data.push(record);
                }
            });
        });
        res.send({ data: data });
    },
```

Отримання усього, що є в базі даних:

```
    async findAll(req, res) {
        const data = await Data.find().sort({ createdAt:
   'desc' });
        return res.status(200).send({ data: data });
    },
```

Видалити всі дані з бази даних:

```
    async removeAll(req, res, next) {
        Data.deleteMany((err) => {
            if (err) next();
            else  return  res.status(200).send("Deleted
  all records");
        });
    },
```

Створити новий запис в базі даних:

```
    async create(req, res) {
        const date = new Date().toISOString().slice(0,
      10);
        const newData = await new Data({
            temperature: req.body.temperature,
            humidity: req.body.humidity,
            date: date
        }).save();

        return res.status(201).send({ data: newData });
    },
```

Видалити дані за день:

```
    async removeByDay(req, res, next) {
        let currentDay =
    _.lowerCase(req.params.currentDay);

        await Data.find((err, records) => {
            if (!records) return next();

            records.forEach((record) => {
                const date = _.lowerCase(record.date);
                if (date === currentDay) {
                    Data.deleteMany({ date: record.date
     }, (err) => {
                        if (err) return next();
                    });
                }
            });
            res.send("Records posted on " +
      req.params.currentDay + " deleted");
        });
    }
}
```

Для звернення до API потрібна маршрутизація, яка виглядає таким чином:

```
const express = require("express");
const router = express.Router();
const alarmController =
require('../controllers/alarmController');
const catchAsync =
  require('../middlewares/errors').catchAsync;
const jwtAuth = require('../middlewares/auth');

router.get('/',
 catchAsync(alarmController.controller.getStatus));
router.get('/short',
catchAsync(alarmController.controller.getShortStatus));
// if no alarm create one
```

```
router.post('/create',                jwtAuth.jwtAuth,
catchAsync(alarmController.controller.createAlarm));
router.post('/enable',                jwtAuth.jwtAuth,
catchAsync(alarmController.controller.enableAlarm));
router.post('/disable',               jwtAuth.jwtAuth,
catchAsync(alarmController.controller.disableAlarm));
module.exports               =               router;
const express = require("express");
const router = express.Router();
const                 dataController                =
require('../controllers/dataController');
const                   catchAsync                  =
require('../middlewares/errors').catchAsync;
const jwtAuth = require('../middlewares/auth');
router.get('/',
catchAsync(dataController.controller.findAll));
router.post('/',
jwtAuth.jwtAuth,catchAsync(dataController.controller.cr
eate));
router.delete('/',jwtAuth.jwtAuth,
catchAsync(dataController.controller.removeAll));
//yyyy-mm-dd
router.get('/:currentDay',
catchAsync(dataController.controller.findByDay));
router.delete('/:currentDay',         jwtAuth.jwtAuth,
catchAsync(dataController.controller.removeByDay));
module.exports = router;
```

Веб-додаток був написан на Agular 13. Він використовує протокол HTTP для зв'язку з API. Додаток робить запит GET, щоб отримати дані та зберегти їх.

У проекті постійно використовується Observable – це послідовність подій у часі.

Сервіс який повертає Observable з даними:

```
getWether(): Observable<WeatherModule> {
    return this._http.get<WeatherModule>(this.apiDate);
  }
  getAlarm(): Observable<boolean> {
    return this._http.get<boolean>(this.apiGetAlarm);
  }
```

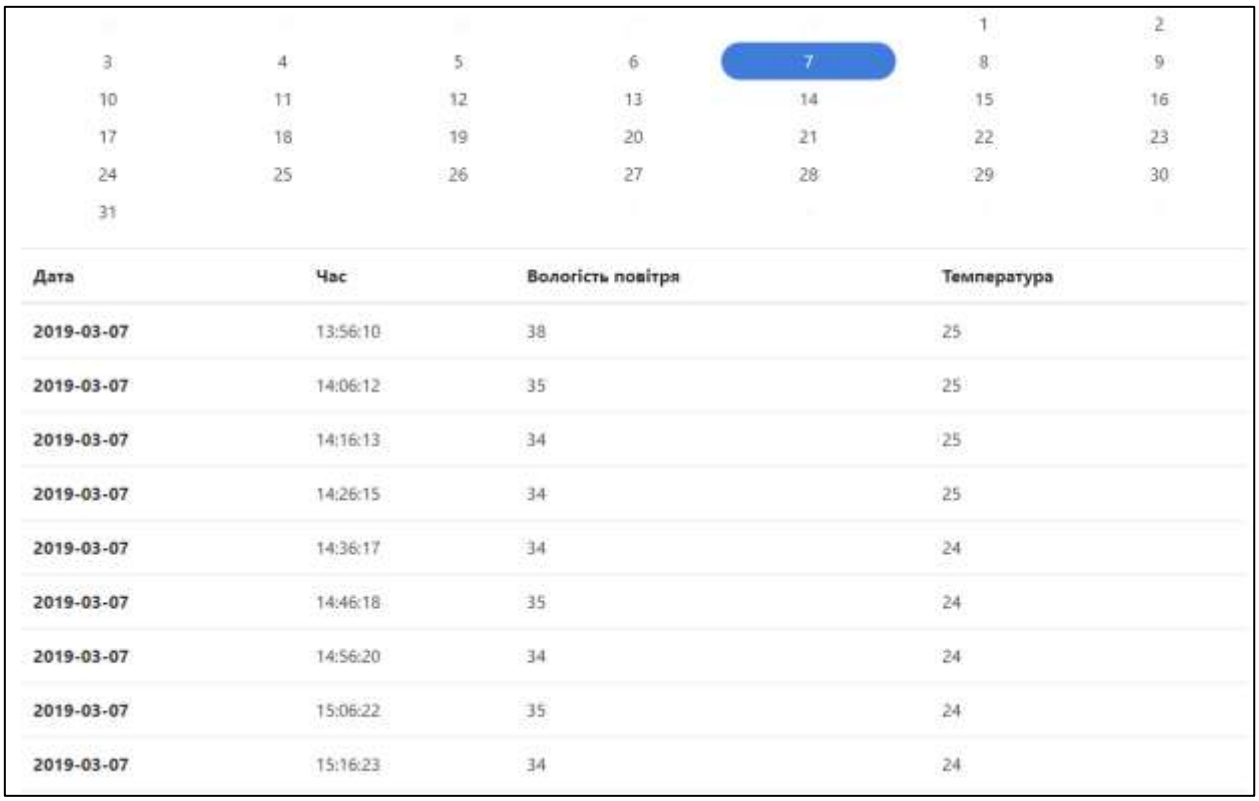
Компонент, який підписується на Observable, дані про температуру:

```
weather: WeatherModule[] = [];

  constructor(private _weather: WetherService) {
  };
```

```
        ngOnInit(): void {
          this.get();
        }

        get(): void{
          this._weather.getWether().subscribe((res:
WeatherModule[]) => {
            this.weather = res
          });
        }
```

Виконується запит до API в методі життєвого циклу Angular компонента ngOnInit(Ініціалізація компонента) зберігається температура. Потім, вибравши дату, компонент фільтрує вхідні дані по цій даті та поміщає їх в таблицю за допомогою інтерполяції рядка.

```
<th scope="row">{{weather.date}}</th>
<td>{{weather.time}}</td>
<td>{{weather.humidity}}</td>
<td>{{weather.temperature}}</td>
```



					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

Дата	Час	Вологість повітря	Температура
2019-03-07	13:56:10	38	25
2019-03-07	14:06:12	35	25
2019-03-07	14:16:13	34	25
2019-03-07	14:26:15	34	25
2019-03-07	14:36:17	34	24
2019-03-07	14:46:18	35	24
2019-03-07	14:56:20	34	24
2019-03-07	15:06:22	35	24
2019-03-07	15:16:23	34	24

Рис. 3.3 – Таблиця з даними про температуру та вологість повітря

За допомогою бібліотки ChartJs є діаграма яка показує почасово зміни вологості повітря та температури.

Використовуються ті ж самі дані, що й для таблиці. Для того щоб використовувати ChartJs необхідно ініціалізувати canvas у шаблоні, та налаштування передати за допомогою інтерполяції.

```
canvas id="canvas">{{ chart }}</canvas>

chart = new Chart('canvas', {
type: 'line',
data: {
  labels: this.times,
  datasets: [
    {
      data: this.temps,
      borderColor: "#3cba9f",
      fill: false,
      label: "Temperature"
    },
    {
      data: this.humis,
      borderColor: "#ffcc00",
      fill: false,
      label: "Humidity"
    },
  ]},
options: {
  legend: {
    display: true
  },
  scales: {
    xAxes: [{
      display: true
    }],
    yAxes: [{
      display: true
    }],
  }
}});
```

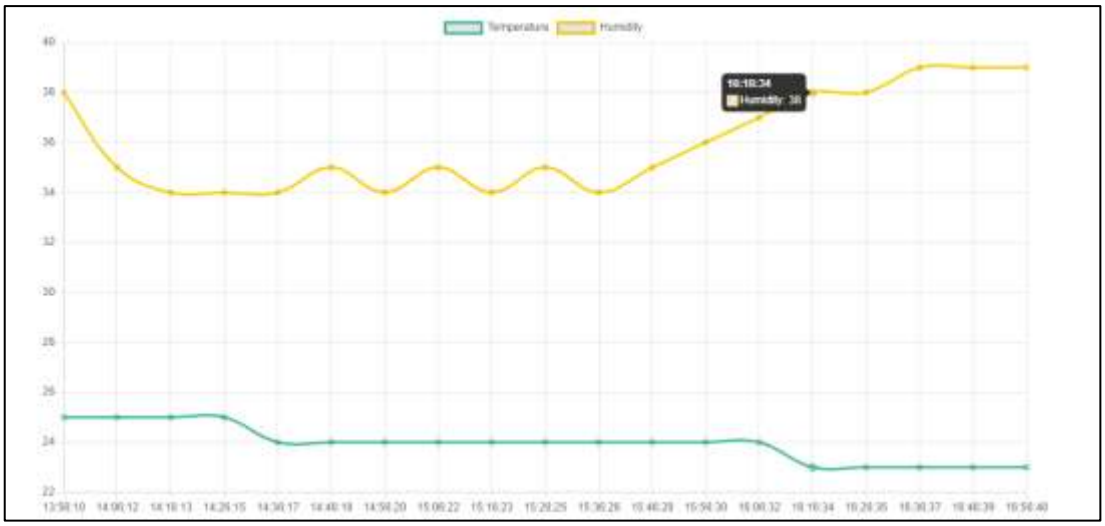


Рис. 3.4 – Діаграма з даними про температуру та вологість повітря

Також було створено керування сигналізацією. Щоб поставити сигнал тривоги, користувач повинен увійти.

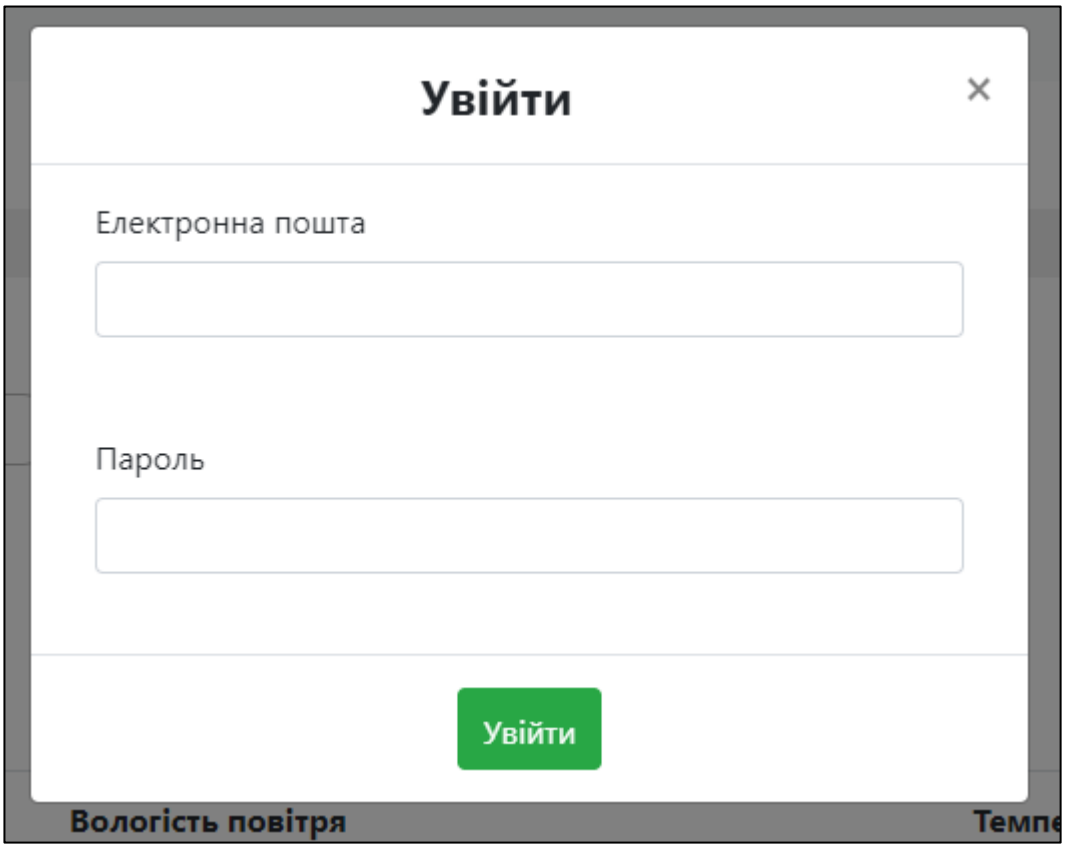


Рис. 3.5 – Вхід до додатка

При натисканні кнопки «Увійти» виконується підпис на Observable який повертає сервіс:

```
login(email: string, password: string):
Observable<{token: string}> {
    return this._http.post<{token:
string}>(this.apiLogin, { email, password });
  }
Підпис на Observable:
login(): void {
    this._wetherService.login("test@test",
"test").subscribe(res => {
      localStorage.setItem("token", res.token);
    });
    if (localStorage.getItem("token")) {
      this.isLogged = true;
    }
  }
```

Після входу API повертає токен, який використовується для запиту PUT та оновлення статусу тривоги, токен зберігається до LocalStorage. LocalStorage – локальне сховище браузера. Далі отриманий токен поміщається в заголовок запиту.

```
setAlarmOn() {
    const httpOptions = {headers: new HttpHeaders({
      'Content-type':'application/json',
      'Authorization':'Bearer '+
localStorage.getItem("token")
    })
  };
    return this._http.post<any>(this.apiAlarmEnable,
null,httpOptions);
  }

  setAlarmOff() {
    const httpOptions = {headers: new HttpHeaders({
      'Content-type':'application/json',
      'Authorization':'Bearer '+
localStorage.getItem("token")
    })
  };
    return
this._http.post<any>(this.apiAlarmDisable, null,
httpOptions)
  }
```

Після входу у шапці додатка з’являється додаткова кнопка «Увімкнути сигналізацію»

При натисканні виконується підпис на Observable та при успіху статус зберігається до змінної isAlarmSet.

В залежності від поточного значення змінної isAlarmSet, визначається на який Observable підписуватись. Також використовується оператор RxJs switchMap – це оператор в основному є комбінацією двох операторів – switchAll і map. Частина дозволяє зіставляти значення з джерела вищого порядку, яке можна спостерігати, у внутрішній Observable потоку. Частина перемикача працює так switchAll: вона підписується на останній наданий внутрішній Observable, випромінювану Observable вищого порядку, відмовляючись від будь-якого раніше підписаного внутрішнього Observable.

switchMap має лише одну активну підписку за раз, з якої значення передаються спостерігачеві. Після того, як спостережуваний вищого порядку видає нове значення, switchMap виконує функцію, щоб отримати новий внутрішній спостережуваний потік, і перемикає потоки. Він скасовує підписку на поточний потік і підписується на новий внутрішній спостережуваний. Який повертає Observable з запитом статусу тривоги.

```
  setAlarm(): void {
      const setAlarm = this.isAlarmSet ? 'setAlarmOff()'
: 'setAlarmOn()';
      this._wetherService[setAlarm]
      .pipe(switchMap(item       =>       {       return
this._wetherService.getAlarm()}))
      .subscribe(res => {
        this.isAlarmSet = res['alarm'].status;
        localStorage.setItem("alarmStatus",
this.isAlarmSet.toString());
      });
    }
```

В залежності від змінної відображається статус в шапці додатка. Або червоного кольору з текстом «Вимкнути сигналізацію», або зеленого кольору з текстом «Увімкнути сигналізацію».

```
<a class="nav-link" (click)="setAlarm()" *ngIf="!isAlarmSet"
style="color: green">Увімнкути сигналізацію</a>
          <a      class="nav-link"      (click)="setAlarm()"
*ngIf="isAlarmSet"         style="color:         red">Вимкнути
сигналізацію</a>
```

Рис. 3.6 – Шапка з змінною isAlarmSet

Налаштування приладів прикладних до Arduino(Slave):

Створюємо змінні які відносяться до відповідних контактів мікроконтролера.

```
#define DHT11_PIN 12
#define TRIG 6
#define ECHO 4
#define PIEZO 8
#define BUTTON 3
#define                    YELLOW_LED                    2


bool alarmArmed;
bool objectDetected;
void setup()
{
```

Задаємо параметри роботи змінних.

```
  pinMode(PIEZO, OUTPUT);
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  pinMode(BUTTON, INPUT);
  attachInterrupt(digitalPinToInterrupt(3),        NULL,
   RISING);

  alarmArmed = false;
  objectDetected = false;
  Serial.begin(9600);
  Wire.begin(3);
  dht.setup(DHT11_PIN);
  Wire.onReceive(receiveHandler);
  Wire.onRequest(requestHandler);
}
```

```
double temperature;
double humidity;
char response[11];

unsigned long nowTime = 0;
unsigned long measuresTime = 0;
unsigned long lookForObjectTime = 0;


void loop()
{
    if(pulseIn(BUTTON, HIGH) > 0)
  {
    objectDetected = false;
  }
```

Перевіряємо статус сигналізації на увімкнено чи вимкнено, в залежності від змінної alarmArmed включаємо діод, або вимикаємо.

```
if(alarmArmed)
{
  digitalWrite(YELLOW_LED, HIGH);
}
else
{
  digitalWrite(YELLOW_LED, LOW);
}

nowTime = millis();
```

З затримкою перевіряємо температуру та вологість повітря, якщо вони більше 0 тоді записуємо до масиву

```
if(nowTime - measuresTime >= 2100UL)
{
  measuresTime = nowTime;
  temperature = dht.getTemperature();
  humidity = dht.getHumidity();
  if(temperature > 0 && humidity > 0)
  {
    String responseTmp = "";
    responseTmp += (String)temperature;
    responseTmp += "|";
    responseTmp += (String)humidity;
    for(int i = 0; i < 11; i++)
    {
      response[i] = responseTmp[i];
    }
  }
}
```

З затримкою перевіряємо статус сигналізації, а саме дистанцію до об’єкту в 7 метрів.

```
 if(nowTime - lookForObjectTime >=500UL)
  {
    lookForObjectTime = nowTime;

    if(alarmArmed && !objectDetected)
    {
      objectDetected = lookForObject();
    }
  }
  if(objectDetected)
  {
        tone(8, 1000, 300);
        delay(100);
        tone(8, 500, 300);
        delay(100);
        tone(8, 1000, 300);
        delay(100);
        tone(8, 500, 300);
        delay(100);
  }

}

bool lookForObject()
{
  long _time, distance;

  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  _time = pulseIn(ECHO, HIGH);
  distance = _time / 58;
  Serial.println(distance);
  if(distance <= 7)
  {
    return true;
  }
  else
  {
    return false;
  }
}
```

Повертаємо значення у MASTER.

```
void requestHandler()
{
  Wire.write(response);
}

void receiveHandler()
{
  int value = Wire.read();
  if(value == 1)
  {
    alarmArmed = true;
  }
  else
  {
    alarmArmed = false;
  }
}
```

Налаштування Arduino(Master):

```
bool alarmArmed;
const char* ssid = "";
const char* password = "";
char dataTempHum[11];
LiquidCrystal_I2C lcd(0x27,16,2);

byte thermometer[8] =
{
    B00100,
    B01010,
    B01010,
    B01110,
    B01110,
    B11111,
    B11111,
    B01110
};
byte droplet[8] =
{
    B00100,
    B00100,
    B01010,
    B01010,
    B10001,
    B10001,
    B10001,
    B01110,
};
```

Задаємо параметри роботи змінних.

```
void setup () {
```

```
    pinMode(D7,INPUT);
    Wire.begin(D2,D1);

  lcd.init();
  lcd.createChar(1,thermometer);
  lcd.createChar(2,droplet);

  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("INITIALIZATION");
  delay(500);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Network:");
  lcd.setCursor(0, 1);
  lcd.print(ssid);
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {

    delay(1000);
    lcd.clear();
    lcd.print("Connecting..");
  }
  lcd.clear();

}

unsigned long nowTime = 0;
unsigned long checkAlarmTime = 0;
unsigned long checkMeasurementsTime = 0;
unsigned long sendDataTime = 0;
String response = "";

void loop()
Входимо до свого аккаунту:
{
  if(pulseIn(D7, HIGH) > 0)
  {
    String token = loginPOSTrequestApi();
    alarmArmed = setAlarmInAPI(token, alarmArmed);
  }
  nowTime = millis();
```

З певною затримкою отримуємо дані для виводу:

```
  if(nowTime - checkMeasurementsTime  >= 3000UL)
  {
    response="";
```

```
        checkMeasurementsTime = nowTime;
        Wire.requestFrom(3, 11);
         while(Wire.available())
         {
            char c = Wire.read();
            response += c;
         }
            printTempHumOnLCD(response);
            printAlarmStatus(alarmArmed);

      }
      if(nowTime - sendDataTime >= 1200000UL)
      {
          sendDataTime = nowTime;
          String token = loginPOSTrequestApi();
          sendValuesPOSTrequestApi(token, response);
      }

      if(nowTime - checkAlarmTime >= 300UL)
      {
        checkAlarmTime = nowTime;
        alarmArmed = setAlarm();
      }
    }
```

Генеруємо об’єкт для відправки даних.

```
    String parseValuesToJSON(String rawData)
    {
      String JSON = "";

      JSON += "{";
      JSON += "\"temperature\":";
      JSON += rawData[0];
      JSON += rawData[1];
      JSON += ",";
      JSON += "\"humidity\":";
      JSON += rawData[6];
      JSON += rawData[7];
      JSON += "}";
      return JSON;
    }
```

Метод для відправки даних до API

```
    void   sendValuesPOSTrequestApi(String   token,   String
dataToPost)
    {
      HTTPClient http;
      String postData = parseValuesToJSON(dataToPost);

      http.begin("http://localhost:8080/api/data");
      http.addHeader("Content-Type", "application/json");
```

```
      http.addHeader("Authorization", "Bearer " + token);
      int httpCode = http.POST(postData);
      Serial.println(httpCode);

      http.end();
    }
```

Метод для входу до свого користувача

```
    String loginPOSTrequestApi()
    {
       HTTPClient http;
      String postData;
      String login = "test@test";
      String password = "test";

      postData = "email=" + login + "&password=" + password;

      http.begin("http://localhost:8080/api/auth/login");
      http.addHeader("Content-Type",     "application/x-www-
form-urlencoded");

      int httpCode = http.POST(postData);
      String payload = http.getString();
      http.end();  //Close connection

      int payloadLength = payload.length();
      String token = payload.substring(10,181);
      return token;
    }
    void printTempHumOnLCD(String response)
    {
      String temperature, humidity, alarm;
      for(int i = 0; i < 5; i++)
      {
        temperature += response[i];
      }

      for(int i = 6; i < 11; i++)
      {
        humidity += response[i];
      }
      lcd.setCursor(0, 0);
      lcd.clear();
      lcd.write(1);
      lcd.print(temperature);
      lcd.print((char)223);
      lcd.print("C|");
      lcd.setCursor(0, 1);
      lcd.write(2);
      lcd.print(humidity);
      lcd.print("% |");
```

```
      lcd.noBlink();
    }
```

Метод для включення або виключення сигналізації.

```
    bool setAlarmInAPI(String token, bool alarmStatus)
    {
      if(alarmStatus)
      {
        HTTPClient http;


http.begin("http://localhost:8080/api/alarm/disable");
        http.addHeader("Content-Type", "application/json");
        http.addHeader("Authorization", "Bearer " + token);
        int httpCode = http.POST("");
        Serial.println(httpCode);
        http.end();
        }
      else
      {
        HTTPClient http;


http.begin("http://localhost:8080/api/alarm/enable");
        http.addHeader("Content-Type", "application/json");
        http.addHeader("Authorization", "Bearer " + token);
        int httpCode = http.POST("");
        Serial.println(httpCode);
        http.end();
      }

      return !alarmStatus;
    }

    int requestGETapiAlarm()
    {
      String payload;
      if (WiFi.status() == WL_CONNECTED) {
        HTTPClient http;

        http.begin("http://localhost:8080/api/alarm/short");
        int httpCode = http.GET();
        if (httpCode > 0) {

          payload = http.getString();
          Serial.println(payload);
        }

        http.end();   //Close connection
        return payload.toInt();
```

```
  }
}

int setAlarm()
{
  int alarmStatus = requestGETapiAlarm();
  Wire.beginTransmission(3);
  Wire.write(alarmStatus);
  Wire.endTransmission();
  return alarmStatus;
}

void printAlarmStatus(int alarmStatus)
{
  lcd.setCursor(9, 0);
  lcd.print("ALARM:");
  lcd.setCursor(9,1);
  if(alarmStatus == 1)
  {
    lcd.print("Сигналізація працює");
  }
  else
  {
    lcd.print("Сигналізація ввимкнена");
  }
}
```

ВИСНОВКИ

На даний момент існує багато різноманітних додатків для систем «Розумний дім, але переважна більшість з них передбачає зв’язок лише з їх апаратним забезпеченням, або велику ціну.

Огляд предметної області показав, що розробити додаток для управління системою «Розумний дім», можна за допомогою поширеного стеку MEAN, але цей стек не виключає використання інших фреймворків для написання фронтенду або бекенду.

Відмінність запропонованої розробки від стандартних рішень полягає в тому, що для налаштування такої системи можна використовувати різні апаратні частини, які будуть взаємодіяти з сервером та надавати данні одного й того ж виду (інтерфейсу). Також ця система надає змогу використовувати багато однакових датчиків у різних приміщеннях, де єдиним затратним джерелом буде лише апаратна частина.

Був використаний фреймворк Angular для візуалізації даних, Node.Js для взаємодії з базою даних, та база даних mongoDb.

Реалізація проекту була виконана з врахуванням можливості поширення та масштабування, додаючи до цієї системи контроль освітлення, щоб тим самим контролювати штучне освітлення приміщення, чи зробити автоматичну роботу штор на вікнах. Його можна об’єднати з іншими модулями і створити цілу екосистему оселі.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is home automation and how do I get started [Електронний ресурс] / Режим доступу: www. URL: https://www.networkworld.com/article/2874914/what-is-home-automation-and-how-do-i-get-started.html – 23.05.2022 р.
2. Design and Realization of Smart Home Terminal Applications Based on IOT Technology [Електронний ресурс] / Режим доступу: www. URL: https://www.researchgate.net/publication/283109456_Design_and_Realization_of_Smart_Home_Terminal_Applications_Based_on_IOT_Technology – 23.05.2022 р.
3. What is MEAN [Електронний ресурс] / Режим доступу: www. URL: https://en.wikipedia.org/wiki/MEAN_(solution_stack) – 23.05.2022 р.
4. О платформе Arduino [Електронний ресурс] / Режим доступу: www. URL: https://doc.arduino.ua/ru/about/ – 23.05.2022 р.
5. Arduino Reference [Електронний ресурс] / Режим доступу: www. URL: https://www.arduino.cc/reference/en/ – 23.05.2022 р.
6. What is Smart Home Technology [Електронний ресурс] / Режим доступу: www. URL: https://www.enginess.io/insights/what-is-smart-home-technology – 23.05.2022 р.
7. Best Smart Home Devices [Електронний ресурс] / Режим доступу: www. URL: https://www.pcmag.com/picks/the-best-smart-home-devices – 23.05.2022 р.
8. smart home or building (home automation or domotics) [Електронний ресурс] / Режим доступу: www. URL: https://www.techtarget.com/iotagenda/definition/smart-home-or-building – 23.05.2022 р.

9. React A JavaScript library for building user interfaces [Електронний ресурс] / Режим доступу: www. URL: https://reactjs.org/ – 23.05.2022 р.
10. Angular [Електронний ресурс] / Режим доступу: www. URL: https://angular.io/guide/template-reference-variables – 23.05.2022 р.
11. VueJS progressive JS Framework [Електронний ресурс] / Режим доступу: www. URL: https://vuejs.org/ – 23.05.2022 р.
12. NodeJs [Електронний ресурс] / Режим доступу: www. URL: https://nodejs.org/en/about/ – 23.05.2022 р.
13. Google Home Application [Електронний ресурс] / Режим доступу: www. URL: https://support.google.com/chromecast/answer/7071794?hl=ru&co=GENIE.Platform%3DAndroid – 23.05.2022 р.
14. Amazon Alexa [Електронний ресурс] / Режим доступу: www. URL: https://ru.wikipedia.org/wiki/Amazon_Alexa – 23.05.2022 р.
15. IOS Home [Електронний ресурс] / Режим доступу: www. URL: https://www.apple.com/ua/ios/home/ – 23.05.2022 р.
16. Z-Wave [Електронний ресурс] / Режим доступу: www. URL: https://en.wikipedia.org/wiki/Z-Wave – 23.05.2022 р.
17. Zigbee [Електронний ресурс] / Режим доступу: www. URL: https://en.wikipedia.org/wiki/Zigbee – 23.05.2022 р.
18. BACnet [Електронний ресурс] / Режим доступу: www. URL: https://en.wikipedia.org/wiki/BACnet – 23.05.2022 р.
19. X-10 Industry standart [Електронний ресурс] / Режим доступу: www. URL: https://en.wikipedia.org/wiki/X10_(industry_standard) – 23.05.2022 р.
20. Smart home platform supporting decentralized adaptive automation control [Електронний ресурс] / Режим доступу: www. URL: https://dl.acm.org/doi/10.1145/3341105.3373925 – 23.05.2022 р.

Додаток А. Код застосування