Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Приладобудівний факультет
Кафедра автоматизації та систем неруйнівного контролю

«На правах рукопису»
УДК 681.5

«До захисту допущено»
В.о. завідувача кафедри

_____ Юрій КИРИЧУК
(підпис)
“___”_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Комп'ютерно - інтегровані технології проектування приладів»
зі спеціальності 151 Автоматизація та комп'ютерно - інтегровані технології

на тему: Інтелектуальна система моніторингу мікроклімату та безпеки будинку

Виконав: студент 6 курсу, групи ПМ-01мп
(шифр групи)
_____Олійник Владислав Сергійович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник: ____к.т.н. доцент Гришанова І. А. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант: Розробка стартап-проєкту д.е.н., проф. Бояринова К.О. _____
(назва розділу) (науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент: __професор, д.т.н., Володарський Є.Т._____ _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2021 року

ВІДОМІСТЬ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на магістерську дисертацію	1	
2	А4	МД.ПЗ	Пояснювальна записка	104	
3	А1	МД.ГД.01	Алгоритми програм	2	
4	А1	МД.ГД.02 (01…02)	Схеми	2	
5	А1	МД.ГД.03	Складальне креслення	1	
6	А1	МД.ГД.04.(01..02)	Креслення датчиків	2	
7	А1	МД.ГД.05	Презентаційний аркуш	1	
Загальна кількість графічних документів - 8 арк.ф. А1					

				МД.ВМД		
	ПІБ	Підп.	Дата			
Розробн.	Олійник			Відомість магістерської дисертації	Лист	Листів
Керівн.	Гришанова				1	1
Конс.					КПІ імені Ігоря Сікорського каф. ПБ гр. ПМ – 01мп	
Н/контр.						
Зав.каф.	Киричук					

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Приладобудівний факультет
Кафедра автоматизації та систем неруйнівного контролю

Рівень вищої освіти – другий (магістерський)

Спеціальність – 151 Автоматизація та комп'ютерно-інтегровані технології

Освітньо-професійна програма Комп'ютерно-інтегровані технології проектування приладів

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Юрій КИРИЧУК
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Олійника Владислава Сергійовича
(прізвище, ім'я, по батькові)

1. Тема дисертації

Інтелектуальна система моніторингу мікроклімату та безпеки будинку

науковий керівник дисертації Гришанова Ірина Аркадіївна, к.т.н. доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «03» листопада 2021р. № 3664-с

2. Строк подання студентом дисертації «10» грудня 2021р.

3. Перелік завдань, які потрібно розробити

Огляд та аналіз матеріалів за темою магістерської дисертації. Розробка архітектури проєкту. Розробка програмного забезпечення. Розробка прототипу інтелектуальної системи. Розробка конструкторської документації _____
_____

4. Перелік графічного (ілюстративного) матеріалу

Алгоритми програм 2 арк. ф. А1, схеми 2 арк. ф. А1, складальне креслення 1 арк. ф. А1, креслення датчиків 2 арк. ф. А1, презентаційний аркуш 1 арк. ф. А1._____
_____
_____

5. Орієнтовний перелік публікацій

Доповідь на науково-технічній конференції._____
_____
_____

6. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розробка СТАРТАП-проєкту	д.е.н., професор Бояринова К.О.		

7. Дата видачі завдання 01.10.2021

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1.	Огляд та аналіз матеріалів за темою магістерської дисертації	01.10.2021-07.10.2021	
2.	Розробка архітектури проєкту	07.10.2021-15.10.2021	
3.	Вибір датчиків та мікропроцесорів	15.10.2021-22.10.2021	
4.	Вибір програмних технологій	22.10.2021-29.10.2021	
5.	Розробка програмного забезпечення	29.10.2021-05.11.2021	
6.	Розробка прототипу інтелектуальної системи	12.11.2021-19.11.2021	
7.	Розробка конструкторської документації	19.11.2021-26.11.2021	
8.	Розробка стартап-проєкту	26.11.2021-03.12.2021	
9.	Підготовка пояснювальної записки	03.12.2021-10.12.2021	

Студент _____ Олійник В. С.
(підпис)

Науковий керівник дисертації _____ Гришанова І. А.
(підпис)

* Консультантом не може бути зазначено наукового керівника

АНОТАЦІЯ

Магістерська дисертація складається із вступу, трьох розділів, загального висновку, списку використаної літератури та додатків. Дисертація містить: 37 рисунків (схеми, фото та ін.), 37 таблиць, 17 посилань та 20 сторінок додатків з програмним кодом. Загальний обсяг роботи – 104 сторінок.

Метою роботи є створення прототипу системи моніторингу мікроклімату та безпеки будинку.

У роботі аналізуються основні показники мікроклімату та різновиди систем безпеки будинку. Проводиться вибір оптимальних компонентів системи та протоколу передачі даних. Розроблена децентралізована архітектура системи, алгоритм роботи компонентів системи, схема бази даних та програмне забезпечення.

Ключові слова: ESP32, MQTT, Python, датчик, мікроконтролер.

ABSTRACT

The master's thesis consists of an introduction, three chapters, a general conclusion, a list of references and appendices. The master's thesis contains: 37 drawings (diagrams, photos, etc.), 37 tables, 17 references and 20 pages of appendices with program code. The total volume of work is 104 pages.

The aim of the thesis is to create a prototype of the microclimate monitoring system and home security system.

The main indicators of microclimate and types of home security systems are analyzed in the thesis. The optimal components of the system and data transmission protocol are selected. Decentralized system architecture, algorithm of system components operation, database scheme and software have been developed.

Keywords: ESP32, MQTT, Python, sensor, microcontroller.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І СКОРОЧЕНЬ

АЦП – аналого-цифровий перетворювач

ЦАП – цифро-аналоговий перетворювач

I2C – Inter-Integrated Circuit (Міжінтегральна схема)

UART – Universal Asynchronous Receiver-Transmitter (Універсальний асинхронний приймач-передавач)

CAN – Controller Area Network (Мережа контролера)

SPI – Serial Peripheral Interface (Послідовний периферійний інтерфейс)

I2S – Inter-IC Sound

RMII – Reduced Media Independent Interface (Зменшений та незалежний від середовища передачі інтерфейс)

PWM – pulse-width modulation (Широтно-імпульсна модуляція)

IDE – Integrated Development Environment (Інтегроване середовище розробки)

BLE – Bluetooth Low Energy

HTTP – HyperText Transfer Protocol (Протокол передачі гіпертексту)

MQTT – message queuing telemetry transport (телеметричний транспортний протокол із чергою повідомлень)

IoT – Internet of Things (Інтернет речей)

API – application programming interface (прикладний програмний інтерфейс)

QoS – Quality of service (якість сервісу)

Ω – Ом

ВСТУП

Останнім часом широкої популярності набувають системи розумного будинку, системи безпеки будинку та інтернет речей. Такі системи дозволяють контролювати процеси свого будинку та легко керувати технікою в домі.

На ринку існує багато різноманітних систем від відомих брендів, таких як: Xiaomi, Ajax Systems, Broadlink та ін. Ключова проблема – висока ціна. До прикладу, мінімальний стартовий комплект системи безпеки Ajax Systems коштує від 6000 грн. Тому стає актуальною ідея розробки оптимізованої системи з оптимальним ціноутворенням.

Сучасні технології дозволяють проводити моніторинг багатьох параметрів нашої життєдіяльності. Досить важливо контролювати мікроклімат будинку, оскільки він сильно впливає на самопочуття людини.

Зважаючи на описані фактори, було прийнято рішення розробки системи моніторингу мікроклімату та безпеки будинку. Така система може розв’язати питання високої ціни на смарт системи розробкою оптимізованої архітектури взаємодії між компонентами системи та використанням хмарних технологій.

1. ОГЛЯД ТА АНАЛІЗ СИСТЕМ МОНІТОРИНГУ БЕЗПЕКИ БУДИНКУ ТА ОСНОВНІ ПАРАМЕТРИ МІКРОКЛІМАТУ БУДИНКУ

1.1. Мікроклімат

Існує безліч показників, які описують комфортність житла, чи то приватний будинок, квартира, дача або тимчасова споруда. Одним із найважливіших параметрів є мікроклімат, який безпосередньо впливає на людину та її здоров'я. Зазвичай це поняття має на увазі стан середовища всередині приміщень, особлива увага при цьому приділяється повітрю, його температурі, вологості, чистоті, свіжості, рухливості тощо. До додаткових параметрів оцінки мікроклімату іноді відносять освітленість, рівень шуму та вібрацій.

Для кожного з основних показників є свої вимоги, що змінюються в залежності від призначення приміщення та сезону:

- Температура. Вважається, що оптимальна температура в житловому приміщенні повинна бути приблизно 20 °С. Якщо ж говорити більш докладно, то в літній час показник повинен бути в межах 22-25 °С, а в холодний – 20-23 °С. Взимку виняток становлять ванна кімната та дитяча, де температура повинна бути трохи вищою – 24-26 °С та 23-24 °С відповідно. При підборі оптимальної температури для житла варто враховувати регіон проживання. Для регулювання температурного режиму можуть використовуватись обігрівачі, кліматичні установки, кондиціонери та інше обладнання.

- Вологість. Оптимальна вологість у житлових приміщеннях має становити від 40 до 60%, проте деякі нормативи вказують інші показники – 30-45% у холодну пору року, 30-60% улітку. При вологості нижче 30-40% у людини можуть з'являтися неприємні відчуття, у тому числі сухість у горлі та очах, а якщо показник вищий за 60%, то у приміщенні з'являється вогкість, а за нею – цвіль. Регулювати вологість у житлі можна за допомогою побутових зволожувачів та осушувачів повітря, а також правильно влаштованої вентиляції.

•Чистота повітря. Цей параметр більш складний для вимірювань, ніж попередні два, однак він є не менш важливим. Забруднювачів повітря може бути безліч, починаючи з пилу, пилку, різних газів, що надходять зовні, закінчуючи біологічними забрудненнями, що виділяються організмом людини. Очищати повітря можна за допомогою бризерів, припливної вентиляції із системою фільтрів, побутових очисників та інших пристроїв.

•Свіжість повітря, а саме – вміст у ньому вуглекислого газу. Прийнятним рівнем CO_2 вважається 800ppm, при підвищенні цієї позначки людина може почати відчувати задуху, млявість та погіршення уваги [1].

1.2. Система безпеки будинку

1.2.1. Опис системи безпеки будинку

Система безпеки будинку – це група фізичних електронних компонентів, які разом працюють для захисту будинку. Часто система безпеки будинку складається з таких об'єктів:

•Камера безпеки: розумні камери відеоспостереження підключаються до Wi-Fi, що дає змогу віддалено транслювати кадри системи в прямому ефірі та отримувати сповіщення, коли камери виявляють рух людей. Багато камер включають інфрачервоне або кольорове нічне бачення, хмарне або локальне сховище, а також двостороннє аудіо, що дозволяє говорити з тим, хто знаходиться на камері. Деякі камери також мають інтеграцію розумної платформи, наприклад Amazon Alexa або Google Assistant.

•Датчик руху: датчики руху слід розташовувати в головному під'їзді або коридорі на першому поверсі будинку, щоб вони могли виявляти рух і сповіщати, коли система поставлена на охорону. Деякі датчики руху чутливі до домашніх тварин, тому вони не вимикаються щоразу, коли тварина проходить повз [2]. На рис. 1 датчик руху від української компанії Ajax Systems.

Рис. 1. Бездротовий датчик руху від Ajax Systems [3]

•Датчик входу: також відомі як контактні датчики, датчики входу мають дві частини: одну, яка йде на вікно або двері, а іншу – на раму. Ці датчики використовують магніти, щоб визначити, коли один з цих входів відкритий чи закритий. Якщо датчик вважає, що точка входу відкрита, він попереджає нас. Пропонується розмістити датчики входу на вікнах або дверях на першому поверсі. Більшість із них працює від батарейок, а багато з них навіть мають клейку підкладку для легкої установки [2]. На рис. 2 датчик відкриття дверей від Ajax Systems.

Рис. 2. Бездротовий датчик входу від Ajax Systems [4]

•Датчик розбиття скла: іноді замість того, щоб відчиняти вікна, зловмисники просто розбивають їх, щоб уникнути спрацювання датчиків входу. Проте датчик розбиття скла також розпізнає звук розбиття скла та сповіщає за допомогою мобільного сповіщення [2]. На рис. 3 бездротовий датчик розбиття скла, який розпізнає розбиття скла на відстані до 9 метрів [5]

Рис. 3. Бездротовий датчик розбиття скла від Ajax Systems [5]

•Сирена: сирени існують в домашніх системах безпеки як самостійно, так і як частина інших пристроїв, наприклад, базової станції. Сирени часто спрацьовують одночасно з іншими сигналами тривоги і мають на меті відлякати зловмисників або попередити сусідів.

•Клавіатура: для постановки або зняття з охорони системи безпеки зазвичай потрібен код, який вводиться на клавіатурі, яка прикріплена до стіни або розміщена на рівній поверхні.

•Брелок: Брелки дозволяють зняти з охорони або поставити систему безпеки на озброєння без використання клавіатури.

•Тривожна кнопка: якщо щось піде не так, тривожна кнопка – це простий і швидкий спосіб сповістити служби екстреної допомоги, будь то поліція, лікарня або навіть пожежна служба. Як і брелки, тривожні кнопки ніде не встановлюються, але вони завжди є під рукою на випадок надзвичайної ситуації.

•Базова станція: базові станції синхронізують усі підключені пристрої з мобільним додатком, щоб можна було отримувати сповіщення.

•Знаки на подвір'ї або наклейки на вікна: багато компаній, які надають послуги систем безпеки, також видають дворові таблички або наклейки на вікна, які дозволяють сповіщати, що у будинку є система безпеки. Часто грабіжники повертаються, якщо бачать такі знаки чи наклейки, тому їх добре мати.

•Детектори диму та вуглекислого газу: рекомендується, щоб у кожному будинку був детектор диму та чадного газу. За допомогою цього простого інструменту отримується попередження, якщо повітря в домі стає небезпечним для дихання.

1.2.2. Види систем безпеки будинку

Існує кілька основних типів систем безпеки будинку:

•DIY: З такими системами безпеки користувач сам збирає їх і сам контролює систему через відповідний мобільний додаток або у інший спосіб.

•Професійна: професійна система безпеки будинку може означати одну з двох речей: професійний монтаж, тобто спеціаліст встановлює обладнання; система з професійним моніторингом, тобто команда людей реагує на сповіщення.

•Дротова система безпеки – це система, яка підключена до існуючої електричної системи будинку.

•Бездротова: з іншого боку, бездротова система безпеки не має жодних проводів і натомість залежить від комбінації батарейок, Wi-Fi та резервного стільникового зв'язку для підключення до програми та центру моніторингу, якщо є. Бездротові системи безпеки простіше встановлювати, ніж дротові, але потрібно буде замінювати або заряджати акумулятор.

•Розумна: розумні системи безпеки підключені до Інтернету, що дозволяє користувачеві переглядати відеозаписи в прямому ефірі з мобільного додатка,

отримувати сповіщення, коли спрацьовує сигнал тривоги, і дистанційно керувати системою.

- Лише локальна: локальні системи сигналізації, на відміну від розумних систем, не підключені до Інтернету, тому немає дистанційного керування, моніторингу чи сповіщень. Навпаки, якщо спрацює будильник, ви дізнаєтесь, лише якщо ви достатньо близько, щоб почути його. Однак локальні системи сигналізації все ще можна професійно контролювати за допомогою стільникового або стаціонарного резервного копіювання [2].

2. ПРОЄКТНО-КОНСТРУКТОРСЬКИЙ РОЗДІЛ

2.1. Призначення системи та модель реалізації

Дана робота ставить за мету створення прототипу інтелектуальної системи моніторингу мікроклімату та безпеки будинку – системи, яка сповіщатиме користувача про мікрокліматичні параметри будинку чи квартири та забезпечуватиме контроль над безпекою; системи, яка не передбачатиме довгого налаштування та підключення. Ключовий параметр системи – це гнучкість. До неї можна додавати нові компоненти без втручання в уже підключені компоненти. Оновлення та підтримка програмного забезпечення відбувається без будь-яких дій зі сторони користувача.

Варто також відзначити, що така система реалізується на фоні нестачі електронних компонентів на ринку. Тому завдяки простоті та невибагливості дана система є привабливою.

Головною перевагою даної системи є те, що обробка та збереження даних відбувається у хмарі і її не потрібно встановлювати кожному користувачу індивідуально, окрім встановлення самих датчиків моніторингу. Тому для системи підходить модель реалізації SaaS.

Software as a service або SaaS – модель роботи з власниками програмного забезпечення, де програмне забезпечення виступає як послуга, а власник є провайдером, який самостійно займається обслуговуванням.

У SaaS платформі користувач отримує доступ до функціоналу програмного забезпечення через інтернет. Програмне забезпечення знаходиться на серверах розробників, які забезпечують роботу програмного забезпечення та його оновлення.

Така модель ідеально підходить для даного проєкту, так як достатньо зробити монтаж датчиків та налаштувати для роботи з хмарними сервісами.

2.2. Вибір мікроконтролерів та датчиків

2.2.1. Мікроконтролер

Мікроконтролер – важлива складова системи, так як він обробляє покази датчиків та передає їх на сервер. Для таких цілей потрібен мікроконтролер, обладнаний WiFi модулем та в той же час енергоефективний і недорогий у ціні. Проаналізувавши доступні на ринку мікроконтролери, найкращим виявився процесор ESP-32.

ESP32 — це серія мікроконтролерів типу SoC (System on a chip), що мають інтегровані контролери Wi-Fi і Bluetooth, низьке енергоспоживання і невисоку ціну [6].

Тому було вибрано плату ESP32 WROOM DevKit v1. Характеристики наведено у табл. 1. Це плата з мікроконтролером ESP32. На рис. 4. показано піни плати.

Піни живлення:

- VIN: Пін для підключення зовнішнього джерела напруги від 5 до 14 вольт.
- 3V3: Пін від стабілізатора напруги з виходом 3,3 вольта та максимальним струмом 1 А. Регулятор забезпечує живлення модуля ESP32-WROOM.
- GND: Заземлення.

Піни вводу/виводу:

- Цифрові входи/виходи: 21 пін 1–5, 12–19, 21–23, 25–27, 32 та 33. Контакти введення-виведення загального призначення. Піни можуть бути налаштовані на вхід або вихід. Логічний рівень одиниці – 3,3 В, нуля – 0 В. Максимальний струм виходу – 12 мА.
- Цифрові входи: 4 піна 34–36 та 39. Контакти введення загального призначення. Можуть бути налаштовані лише на вхід.

•ШІМ: всі піни введення-виводу. Дозволяє виводити аналогові значення у вигляді ШІМ-сигналу з розрядністю 16 біт. Максимальна кількість каналів 16.

•АЦП: 15 пінів 2, 4, 12–15, 25–27, 32–36 та 39. Дозволяє представити аналогову напругу в цифровому вигляді з розрядністю 12 біт.

•ЦАП: піни 25 (DAC1) та 26 (DAC2). Аналоговий вихід цифро-аналогового перетворювача, що дозволяє формувати 8-бітові рівні напруги. Виводи можуть використовуватись для аудіо-виходу.

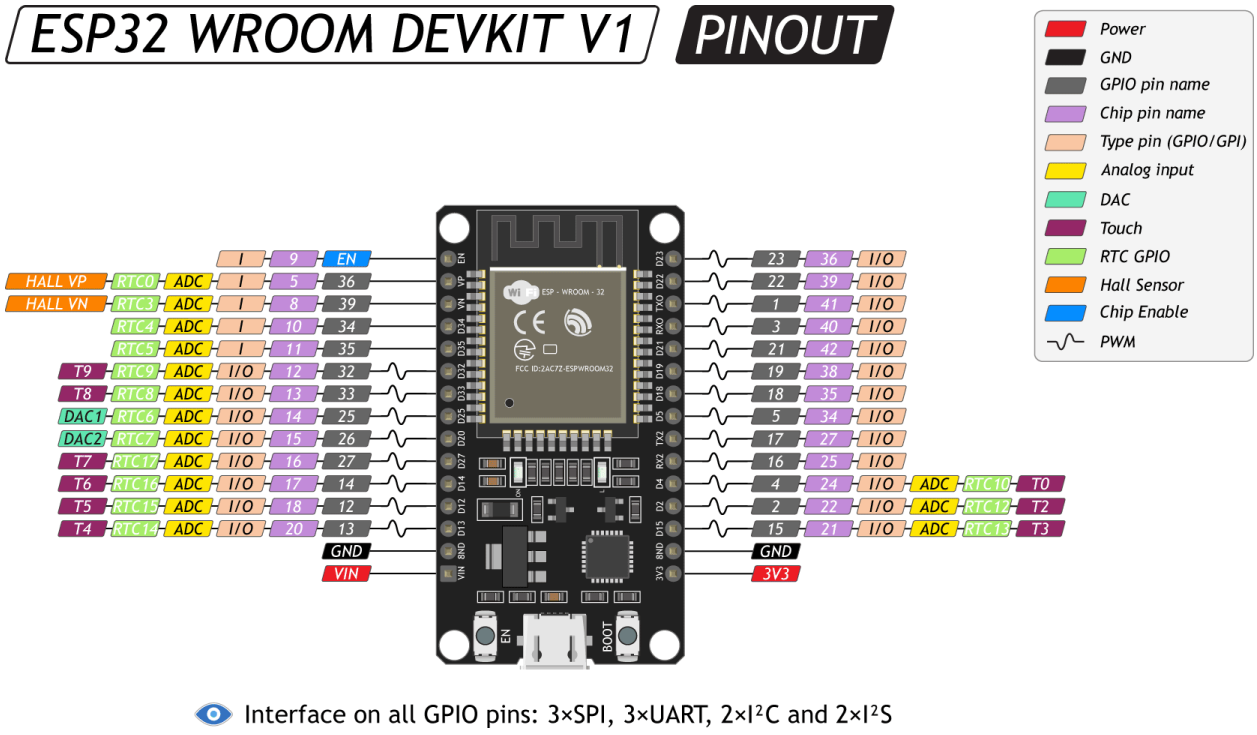

Рис. 4 - Розпіновка ESP32 WROOM DevKit v1 [7]

ESP32 можна програмувати в таких середовищах програмування:

•Arduino IDE;

•Espressif IDF;

•Micropython;

•JavaScript;

•LUA [8].

У даному проєкті використовується Micropython.

Табл. 1 - Технічні характеристики ESP32 DEVKIT V1 [8].

Кількість ядер	2 (двоядерний)
Wi-Fi	2,4 ГГц до 150 Мбіт/с
Bluetooth	BLE і застарілий Bluetooth
Архітектура	32 біти
Тактова частота	До 240 МГц
ОЗП	512 KB
Шпильки	30
Периферійні пристрої	Ємнісний сенсор, АЦП, ЦАП, I2C, UART, CAN 2.0, SPI, I2S, RMII, PWM тощо.

2.2.2. Датчик температури та вологості

При виборі датчику температури та вологості було проаналізовано ринок і знайдено недорогі та оптимальні датчики DHT11, DHT21, DHT22. Ці датчики попередньо відкалібровані і не вимагають додаткових електронних компонентів. Однією з найбільших переваг цих датчиків – це те, що температура та вологість вимірюються з точністю до десятої. Єдиним недоліком цих датчиків є те, що отримувати нові покази можна лише з інтервалом у кілька секунд, але для даного проєкту дана частота знімання показів цілком підходить.

Порівняння характеристик датчиків наведено в таблиці 2. Варто зазначити, що датчик DHT21 повністю ідентичний датчику DHT22 за винятком наявності захисного корпусу, що дозволяє використовувати його на вулиці, де цей корпус захистить його від пилу, бруду і дощу.

Табл. 2 - Порівняння датчиків

	DHT11	DHT21/DHT22
Робоча напруга	Від 3 В до 5 В	Від 3 В до 5 В

Максимальний робочий струм	2.5 мА	2.5 мА
Діапазон вологості	20-80% / ± 5%	0-100% / ± 2%
Температурний діапазон	0-50°C / ± 2°C	Від -40 до 80°C / ± 0.5°C
Частота дискретизації	1 Гц (читання з інтервалом в секунду)	0.5 Гц (читання з інтервалом в 2 секунди)
Ключові переваги	Низька вартість	Більша точність вимірювання

Хоча DHT21 та DHT22 більш точні та працюють в більшому діапазоні температури та вологості є три властивості згідно яким DHT11 виглядає привабливо. Він менш дорогий, менший за розміром і має вищу частоту дискретизації. Частота дискретизації DHT11 становить 1 Гц, тобто одне зчитування щосекунди, тоді як частота дискретизації DHT21 та DHT22 становить 0,5 Гц, тобто одне зчитування кожні дві секунди. Так як точність вимірювань для даного проєкту важливіша, то більш підходять датчики DHT21 та DHT22. Порівнявши їх, було вибрано датчик DHT21 (рис. 5), так як він має ті ж характеристики та високу точність вимірювання як і DHT22, але він більш універсальний оскільки може використовуватись як у будинку, так і на вулиці.

Рис. 5 - Датчик вологості та температури DHT21

DHT21 складається з двох елементів: компонент для визначення вологості та термістор який використовується для визначення температури.

Компонент для визначення вологості (рис. 6) має два електроди з вологоутримуючою підкладкою між ними. Іони вивільняються підкладкою у міру того, як нею поглинається водяна пара, що, у свою чергу, збільшує провідність між верхнім електродом та нижнім електродом. Зміна опору між двома електродами пропорційна відносній вологості повітря. Вища відносна вологість зменшує опір між електродами, а нижча відносна вологість збільшує опір між електродами.

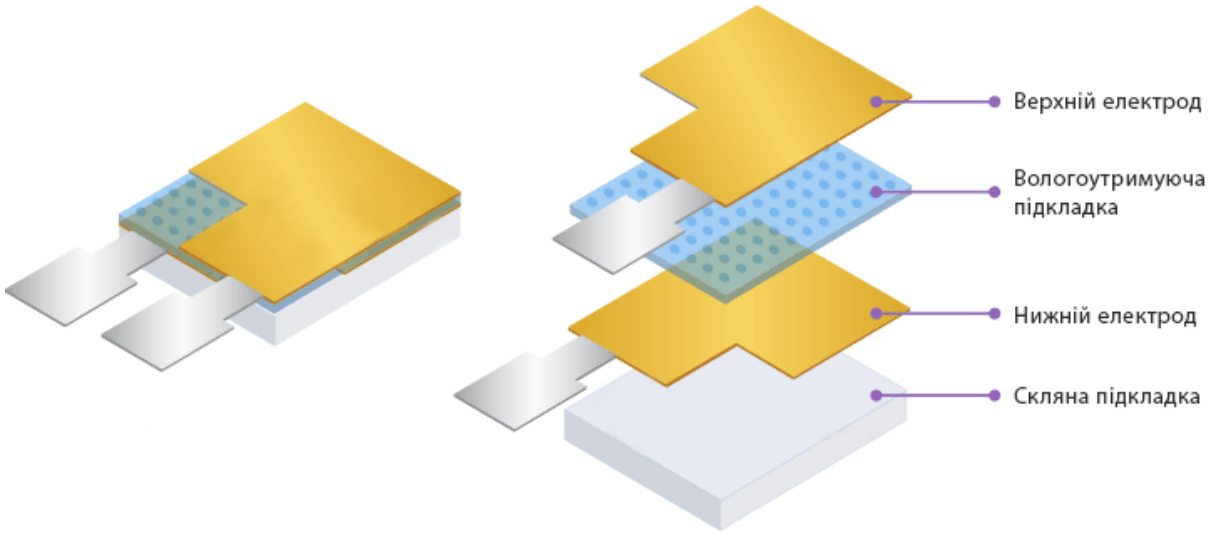

Рис. 6 - Внутрішня структура компонента вимірювання вологості

Для вимірювання температури використовується термістор NTC (Negative Temperature Coefficient).

Терморезистор, термістор — напівпровідниковий резистор, активний електричний опір якого залежить від температури.

Розрізняють терморезистори з негативним (NTC-термістори, від англ. «Negative Temperature Coefficient») і позитивним (PTC-термістори, від англ. «Positive Temperature Coefficient») температурним коефіцієнтом опору (ТКО) [9].

NTC-термістори зменшують свій опір із підвищенням температури. Також вони є найбільш часто використовуваним типом датчиків температури,

оскільки вони можуть використовуватися практично в будь-якому типі обладнання. NTC-термістори мають негативне співвідношення електричного опору та температури (R/T). Відносно великий негативний відгук NTC-термістора означає, що навіть невеликі зміни температури можуть спричинити значні зміни їх електричного опору. Це робить їх ідеальними для точного вимірювання та контролю температури.

NTC-термістори зазвичай характеризуються опором основи при кімнатній температурі, тобто 25 ° C, оскільки це забезпечує зручну контрольну точку. Іншою важливою характеристикою термістора є його коефіцієнт B. B – це константа матеріалу, з якого він виготовлений. Вона описує градієнт резистивної кривої (R/T) у певному температурному діапазоні між двома температурними точками. Кожен матеріал термістора матиме різну константу матеріалу і, отже, різну криву залежності опору від температури.

Таким чином, значення B визначатиме резистивне значення термісторів у першій температурі або базовій точці (яка зазвичай становить 25 ° C), що називається T_1 , і значення резистивного опору термісторів у другій температурній точці, наприклад 100 ° C, яка називається T_2 .

Тому значення B визначатиме константу матеріалу термісторів між діапазоном T_1 і T_2 . Це $B_{T1/T2}$ або $B_{25/100}$ із типовими значеннями NTC термістора β, заданими десь між приблизно 3000 і приблизно 5000.

Однак слід зазначити, що обидві температурні точки T_1 і T_2 обчислюються в одиницях температури Кельвіна, де $0\ ^0$ C = 273,15 Кельвіна. Таким чином, значення 25 ° C дорівнює 25 ° + 273,15 = 298,15K, а 100 ° C дорівнює 100 ° + 273,15 = 373,15K.

Таким чином, знаючи значення B конкретного термістора, можна створити таблицю залежності температури від опору, щоб побудувати відповідний графік за допомогою рівняння (1):

$$B_{(T_1/T_2)} = \frac{T_2 \cdot T_1}{T_2 - T_1} \cdot \ln(\frac{R_1}{R_2}), \quad (1)$$

де

T_1 — перша температурна точка в Кельвінах

T_2 — друга температурна точка в Кельвінах

R_1 - опір термісторів при температурі T_1 в Ом

R_2 - опір термісторів при температурі T_2 в Ом

Для прикладу можна обчислити NTC-термістор 10 кОм значення B якого дорівнює 3455 між температурним діапазоном 25 ° C і 100 ° C. Обчислимо його резистивне значення при 25 ° C і при 100 ° C.

Варто зазначити, що обидві температурні точки T_1 і T_2 обчислюються в одиницях температури Кельвіна, де 0 ^{0} C = 273,15 Кельвіна. Тому, щоб перевести температурну шкалу з градусів Цельсія в градуси Кельвіна, потрібно додати математичну константу 273,15

$$B = \frac{(100 + 273{,}15) \cdot (25 + 273{,}15)}{(100 + 273{,}15) - (25 + 273{,}15)} \cdot \ln(\frac{10000}{R_2})$$

$$3455 = \frac{111254{,}6725}{75} \cdot \ln(\frac{10000}{R_2})$$

$$3455 = 1483{,}4 \cdot \ln(\frac{10000}{R_2})$$

$$e^{(\frac{3455}{1483{,}4})} = \frac{10000}{R_2}$$

$$R_2 = (\frac{10000}{e^{(2{,}33)}_2}) = 973\Omega$$

Визначивши R_2, можна побудувати наступний графік (рис. 7)

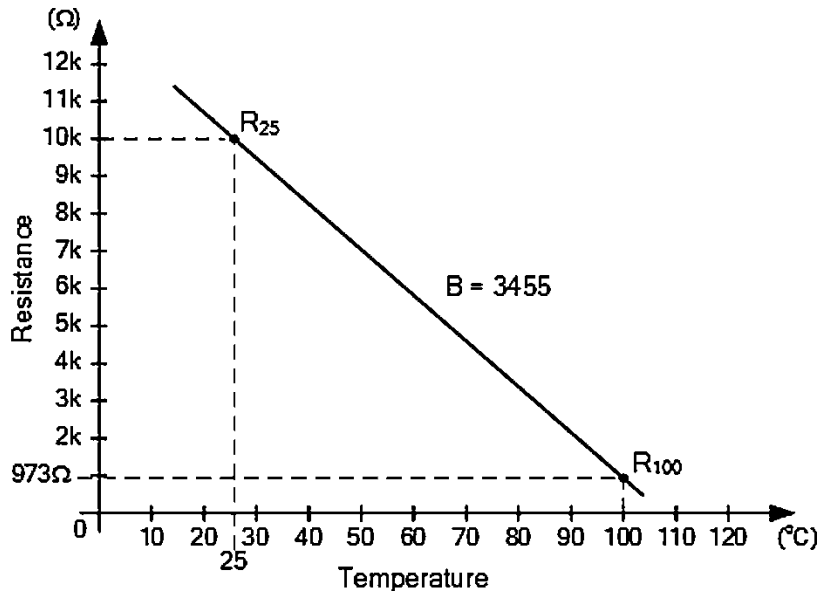

Рис. 7 - Характеристична крива з двома значеннями

Варто зауважити, що в цьому прикладі було знайдено лише дві точки, але зазвичай термістори змінюють свій опір по експоненті зі зміною температури, тому їх характеристична крива є нелінійною. Чим більше температурних точок обчислено, тим точнішою буде крива.

У таблиці 3 наведено обчислені значення опорів при заданих температурах і на основі цих даних можна побудувати більш точну криву.

Табл. 3 - Значення опорів при заданих температурах

Температура, ° С	Опір, Ом
10	18476
20	12185
25	10000
30	8260
40	5740
50	4080
60	2960
70	2188
80	1645

90	1257
100	973
110	765
120	608

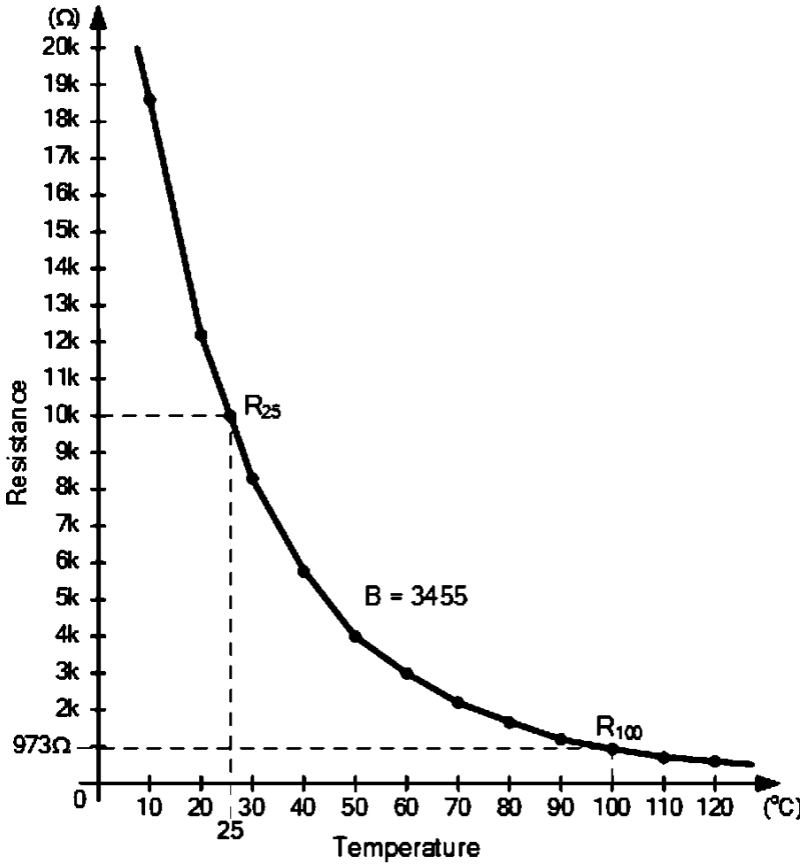

Рис. 8 - Характеристична крива NTC-термістора

Наведена на рис. 8 характеристична крива підтверджує принцип негативного температурного коефіцієнта (NTC), тобто його опір зменшується з підвищенням температури.

2.2.3. Датчик дверей

Для детектування відкриття дверей було вибрано датчик МС-38 (рис. 9). Датчик МС-38 представляє собою геркон та магніт. Електричне коло замикається, коли поруч з герконом знаходиться магніт – двері закриті. Коли магніт знаходиться далеко від геркона, то двері відкриті – ланцюг розімкнутий. Принцип роботи датчика наведено на рис. 10.

Рис. 9 - Датчик дверей МС-38

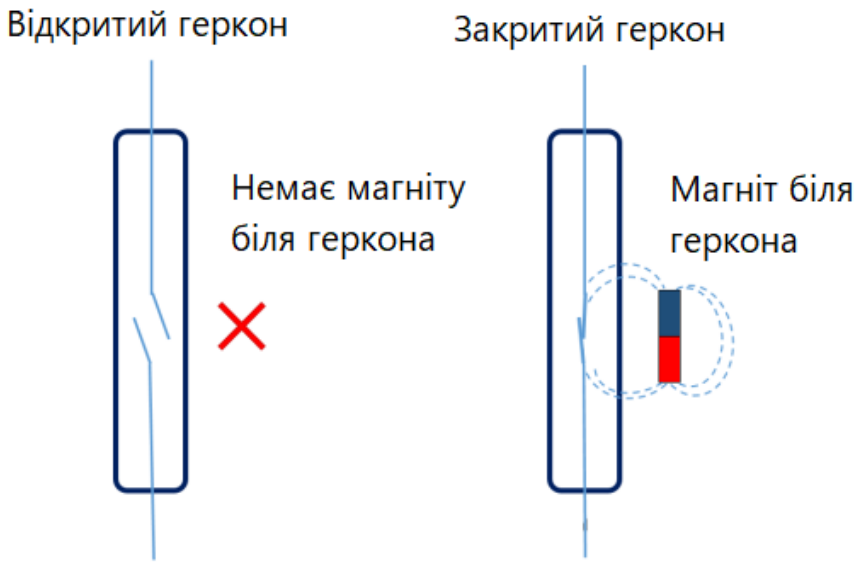

Рис. 10 - Принцип роботи датчика МС-38

Характеристики:

- Номінальний струм: 100 мА
- Номінальна напруга: 200 В постійного струму
- Робоча відстань: 15 - 25 мм
- Номінальна потужність: 3 Вт
- Розмір: 28х15х0,9 см

- Вихід геркона: нормально замкнутий (геркон і магніт разом, коли геркон закритий) [10].

2.3. Вибір протоколу передачі даних

Для передачі даних зазвичай використовують протокол HTTP. Він є популярним та широко використовуваним протоколом. Але для систем IoT дуже важливою є швидкість передачі інформації та енергоефективність. Тому зараз все частіше у проектах розумного будинку та інтернету речей використовують протокол MQTT.

Згідно тестів MQTT на 22% більш енергоефективніший і на 15% швидше. І це не залежить від типу підключення 3G чи Ethernet [11]. Для даного проєкту енергоефективність та швидкість є найважливішими парметрами тому було прийняте рішення використовувати для передачі даних MQTT протокол.

MQTT – це клієнт-серверний транспортний протокол обміну повідомленнями. Він легкий, відкритий, простий і розроблений таким чином, щоб його можна було легко реалізувати. Ці характеристики роблять його ідеальним для використання в багатьох ситуаціях, включаючи обмежені середовища в контексті інтернету речей, де потрібна висока пропускна здатність мережі [12].

Основні особливості протоколу MQTT:

- Асинхронний протокол
- Компактні повідомлення
- Робота в умовах нестабільного зв'язку на лінії передачі даних
- Підтримка кількох рівнів якості обслуговування (QoS)
- Легкая інтеграція нових пристроїв

Обмін повідомленнями у протоколі MQTT здійснюється між клієнтом (client), який може бути видавцем (publisher) або підписником (subscriber) повідомлень, та брокером (broker) повідомлень.

Видавець відправляє дані на брокер MQTT, вказуючи в повідомленні певну тему, топік. У кожного повідомлення є топік. Приклад топіку:

/room/dht/temperature. Підписники можуть отримувати різні дані від багатьох видавців залежно від підписки на відповідні топіки.

Пристрої MQTT використовують певні типи повідомлень для взаємодії з брокером, наведені нижче основні:

- Connect – встановити з'єднання з брокером
- Disconnect – розірвати з'єднання з брокером
- Publish – опублікувати дані в топік на брокері
- Subscribe – підписатися на топік на брокері
- Unsubscribe – відписатися від топіка [13]

На рис. 11 наведена схема взаємодії між видавцем, брокером та підписником.

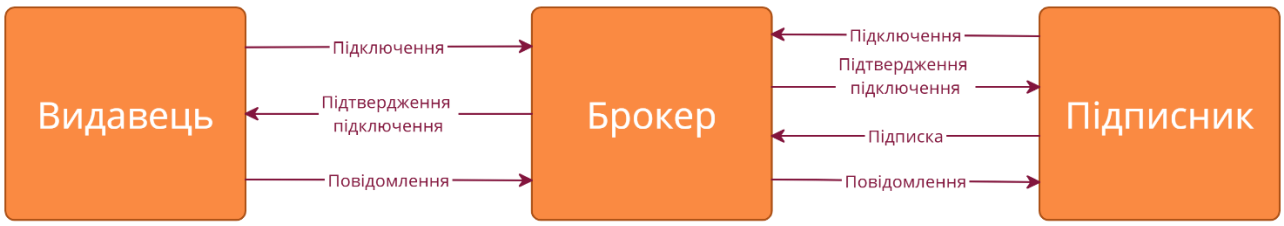

Рис. 11 - Схема взаємодії між видавцем, брокером та підписником

Спочатку видавець та підписник підключаються до брокера, далі вони отримують підтвердження підключення та встановлюють з'єднання. Після встановлення з'єднання, підписник підписується на конкретні топіки. Далі брокер буде пересилати всі повідомлення із вибраним топіком від видавця до підписника.

2.4. Структура системи та принцип роботи

На відміну від класичних систем розумного будинку та систем безпеки будинку, де є центральний контролер, через який компоненти взамодіють, дана система бездротова та децентралізована. Датчики не залежать один від одного. Кожен датчик під'єднаний до мікроконтролера, який у свою чергу передає покази датчика на сервер через WiFi. Тому датчик та мікроконтролер являє собою автономний компонент системи. Уся інфраструктура знаходиться на хмарних серверах, що робить систему більш ефективною та швидко

налаштовуваною. Такий підхід дозволяє спростити монтаж системи, так як потрібно тільки налаштувати підключення до хмарних сервісів та розташувати датчики по будинку.

Варто відмітити основні переваги та недоліки хмарної інфраструктури:

Переваги:

- Гнучкість. Часто хмарні сервіси мають вже готові середовища для прискорення налаштування програмного забезпечення.
- Масштабованість. Хмарні сервіси можуть адаптуватись під трафік. Мають умовно необмежені ресурси для підтримки сервісів. Традиційна інфраструктура з власним сервером не зможе розширитись і доведеться використовувати лише ті потужності, що доступні.
- Витрати на обслуговування. З хмарними сервісами можна платити лише за ті потужності, що використовуються. Власний сервер – при недостатньому навантаженні буде простоювати. Також для власної інфраструктури доведеться наймати системних адміністраторів, які будуть слідкувати за нею.

Недоліки:

- Безпека. Так як дані знаходяться у хмарі, то при недостатній захищеності системи ними можуть заволодіти хакери.
- Стабільність. Якщо у постачальника хмарних послуг з’являться проблеми з системою (DDoS-атака, перебої електропостачання), то це позначиться і на вашій системі. Але зараз постачальники хмарних послуг дуже серйозно ставляться до цього і таке майже ніколи не трапляється.

Датчики використовують для передачі даних протокол MQTT. Обґрунтування вибору даного проколу описане у розділі 2.3

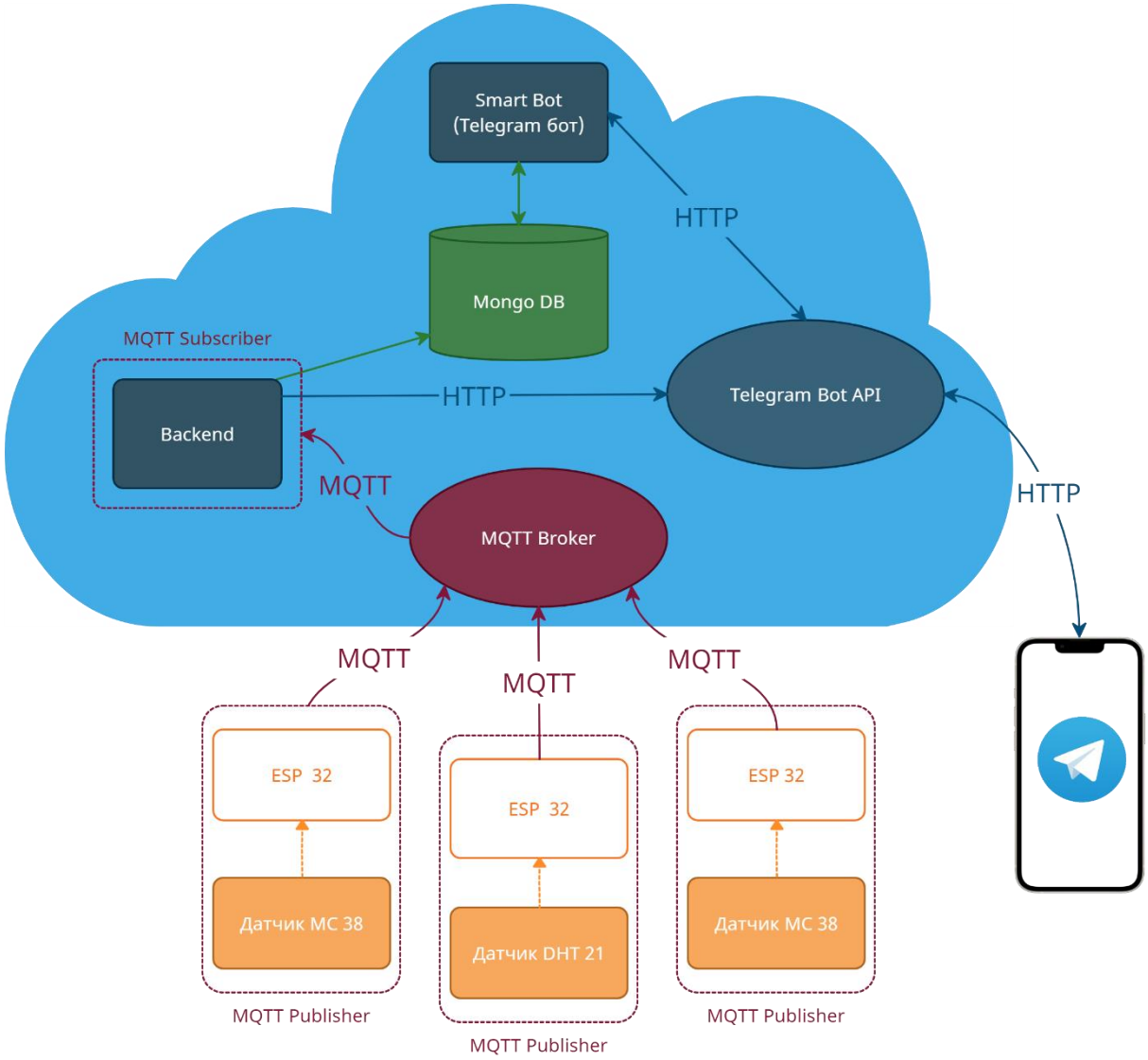

Рис. 12 - Схема взаємодії та обміну даними у системі

На рисунку 12 наведено схему взаємодії та обміну даними. Всі хмарні сервіси розташовані на рисунку у хмарці. Усі фізичні компоненти (плати з датчиками та телефон) поза хмаркою. Червоні стрілки означають, що дані передаються між елементами системи по протоколу MQTT. Сині стрілки означають, що дані передаються по протоколу HTTP. Зелена стрілка направлена до Mongo DB означає запис у базу даних Mongo DB. Зелені стрілки, що направлені у обидві сторони означають запис та читання з бази даних Mongo DB.

Мікропроцесор із датчиком виступають у цій системі MQTT видавцями (MQTT Publisher) та в безперервному режимі відправляють дані через

протокол MQTT до MQTT брокера. Далі брокер відправляє отримані дані від MQTT видавців до MQTT підписників (MQTT Subscriber).

Backend – програмна частина сервісу, яка відповідає за функціонування внутрішньої частини сервісу. Backend написаний з використанням мови програмування Python 3. Backend виступає MQTT підписником і відразу отримує передану інформацію та записує її у базу даних Mongo DB. Також коли датчик MC-38 відправляє на Backend інформацію про те, що двері відчинились/зачинились, то Backend посилає HTTP запит напряму до Telegram Bot API, який у свою чергу повідомляє користувача про це. Програмний код для Backend наведений у додатку А.

Telegram бот записує інформацію про користувача у базу даних при реєстрації. Коли користувач робить запит у Telegram боті, то бот звертається до бази даних Mongo DB і видає користувачу запитану інформацію.

2.5. Компоненти системи

Для вимірювання температури та вологості у кімнаті використовується датчик DHT21. Для детектування відкриття дверей використовується датчик MC-38. Ці датчики працюють із мікроконтролером ESP32, який у свою чергу живиться від літій-іонного аккумулятора формату 18650 LiitoKala Lii 34B із реальною емністю до 3400 мАг. У проєкті використовується два компоненти з датчиком MC-38 для детектування входу на вхідних дверях та дверях вітальні. Для моніторингу мікроклімату у вітальні використовується один компонент з датчиком DHT21.

На рис. 13 та рис. 14 представлені структурні схеми компонентів. Підключення наведено на рис. 15 та рис. 16.

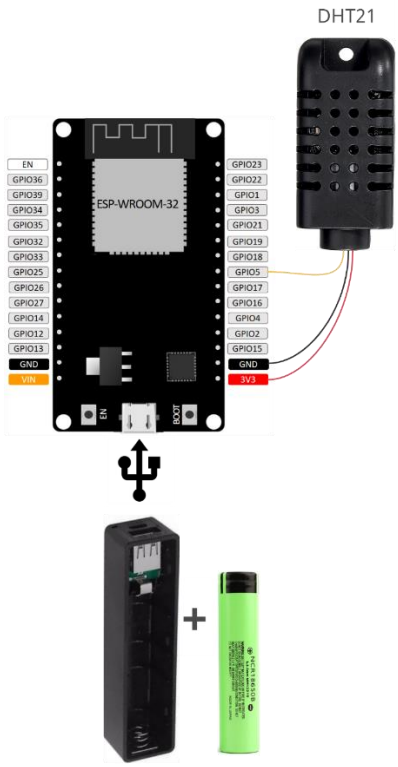

Рис. 13 - Структурна схема компонента з датчиком температури та вологості

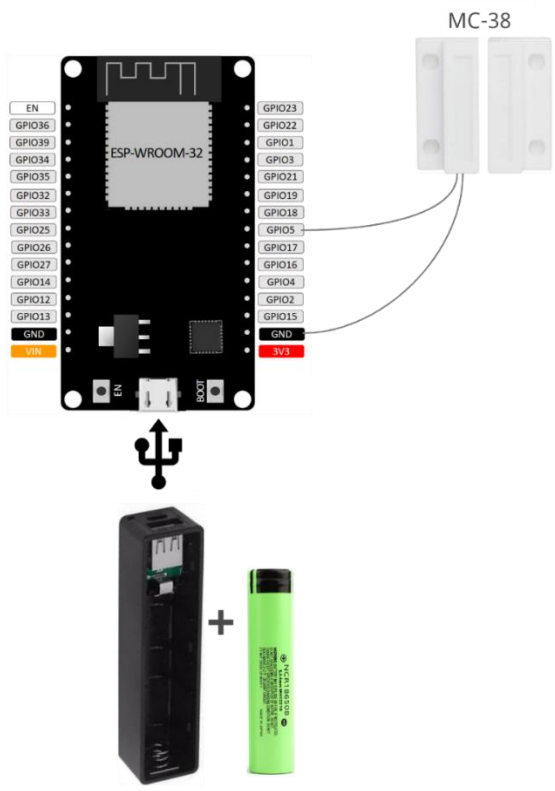

Рис. 14 - Структурна схема компонента з датчиком дверей

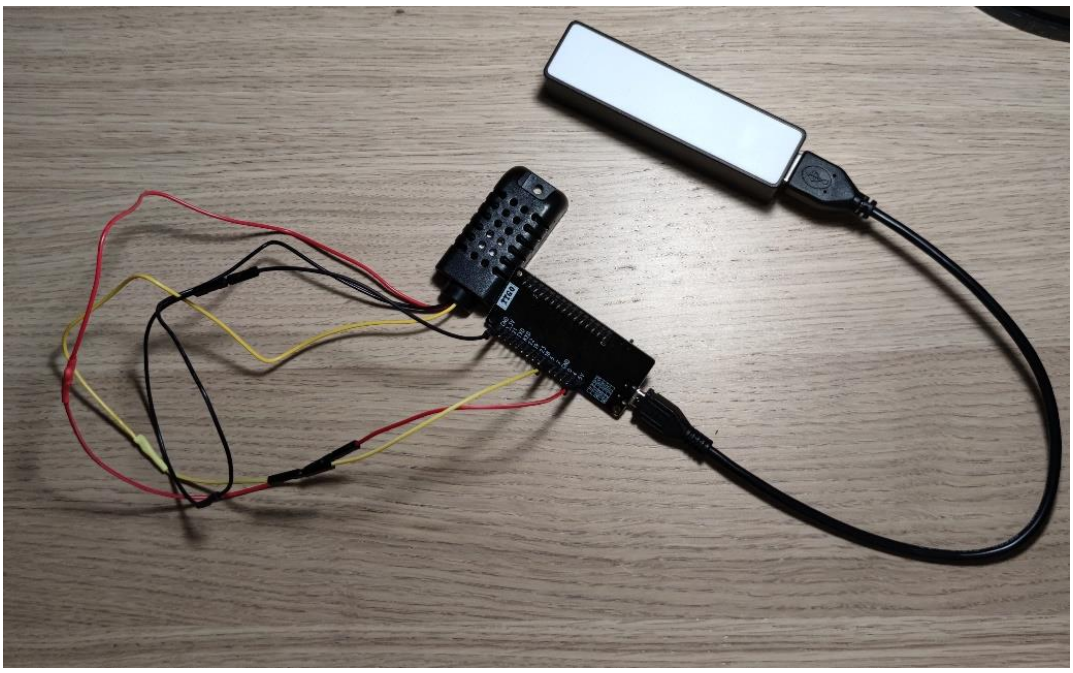

Рис. 15 - Підключення датчика DHT21

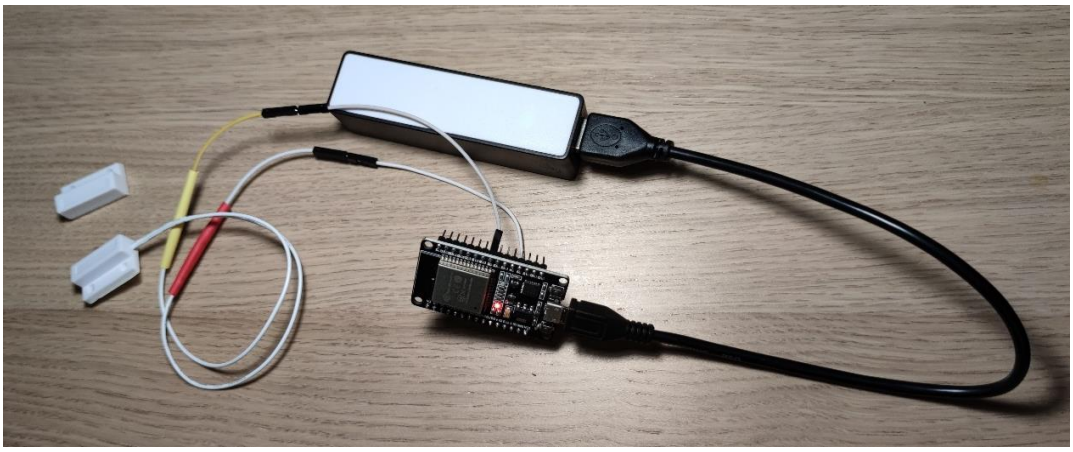

Рис. 16 - Підключення датчика MC-38

Для програмування плат використовується MicroPython. MicroPython — це ефективна реалізація мови програмування Python 3, яка включає в себе невелику підмножину стандартної бібліотеки Python і оптимізована для роботи на мікроконтролерах і в обмежених середовищах.

Програмний код наведений у додатках Б та В. Для підключення системи до MQTT брокера використовується програмний код наведений у додатку Г.

Перед початком необхідно налаштувати компоненти. Для цього потрібно вказати актуальні дані у файл config.json та завантажити його у плату. Файл config.json має наступний вигляд:

```
{
  "sensor_id": "61afa58e81e2f0c225a2748d",
  "door_topic": "door",
  "wifi": {
    "ssid": "My WiFi",
    "password": "password"
  },
  "mqtt": {
    "server": "hairdresser.cloudmqtt.com",
    "port": "1883",
    "user": "admin",
    "password": "password"
  }
}
```

Опис параметрів файлу:

- sensor_id – ідентифікатор датчика у базі даних
- door_topic або dht_topic (залежить від датчика) – MQTT топік на який відправляються дані
- wifi ssid – назва точки WiFi до якої ESP32 буде підключений
- wifi password – пароль WiFi
- mqtt server – MQTT сервер до якого ESP32 буде підключений
- mqtt port – MQTT порт який використовує сервер
- mqtt user – MQTT користувач
- mqtt password – пароль користувача MQTT

Алгоритм роботи плати з датчиком температури та вологості наведено на рис. 17. Алгоритм роботи плати з датчиком дверей на рис. 18. Алгоритм функції підключення до WiFi та функції встановлення з'єдання до MQTT брокера є спільними для двох алгоритмів та наведені на рис. 19 та рис. 20.

Робота компонентів починається з підключенням до джерела живлення. ESP32 підключається до WiFi та створює з'єднання із MQTT брокером. Компонент з датчиком температури та вологості робить вимірювання,

обробляє ці дані за допомогою вбудованої бібліотеки dht та відправляє MQTT брокеру. Вимірювання та відправка даних проводяться із інтервалом у 30 секунд. Компонент з датчиком дверей кожну секунду перевіряє стан геркона та відправляє ці дані.

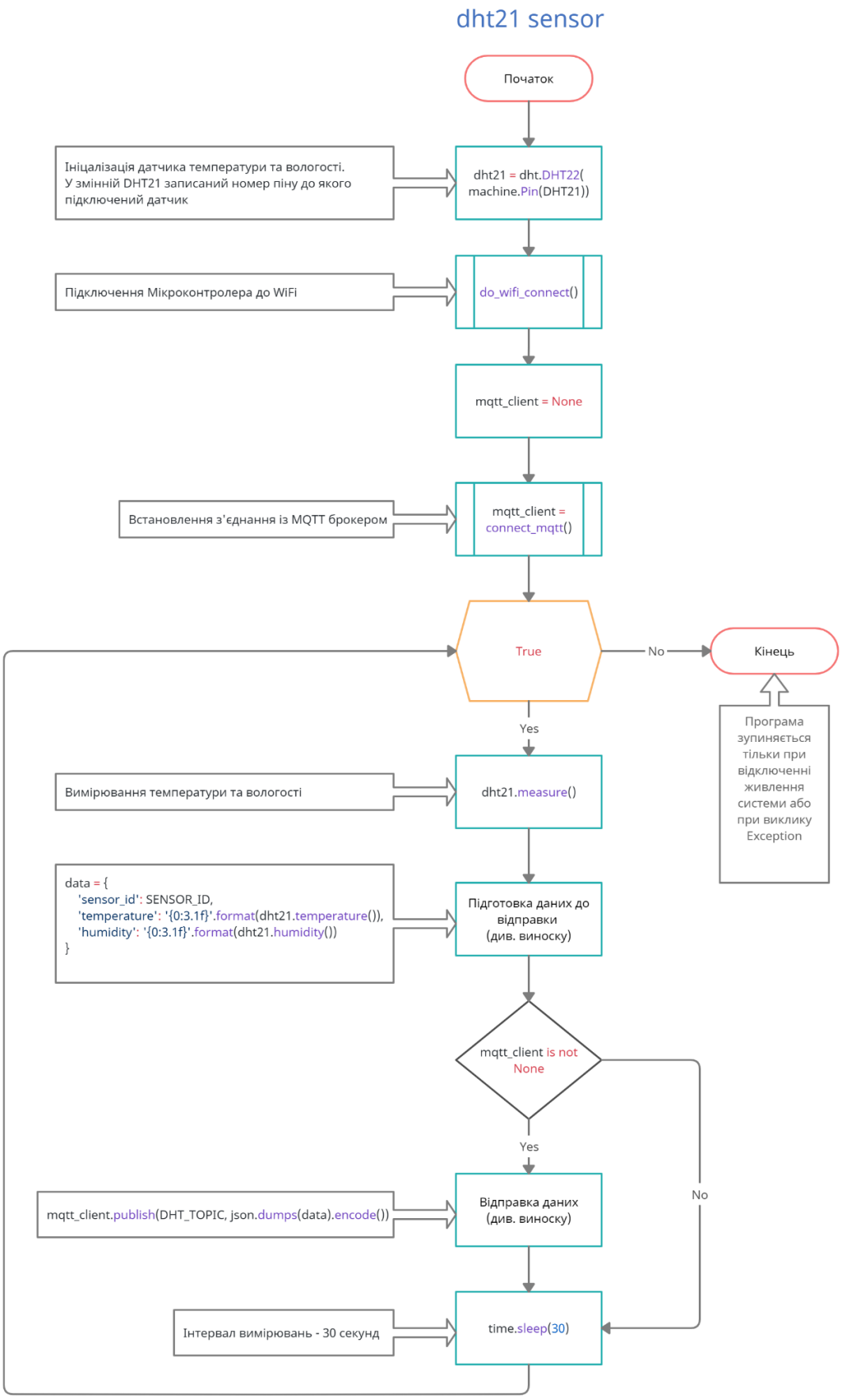

Рис. 17 - Алгоритм роботи датчика температури та вологості

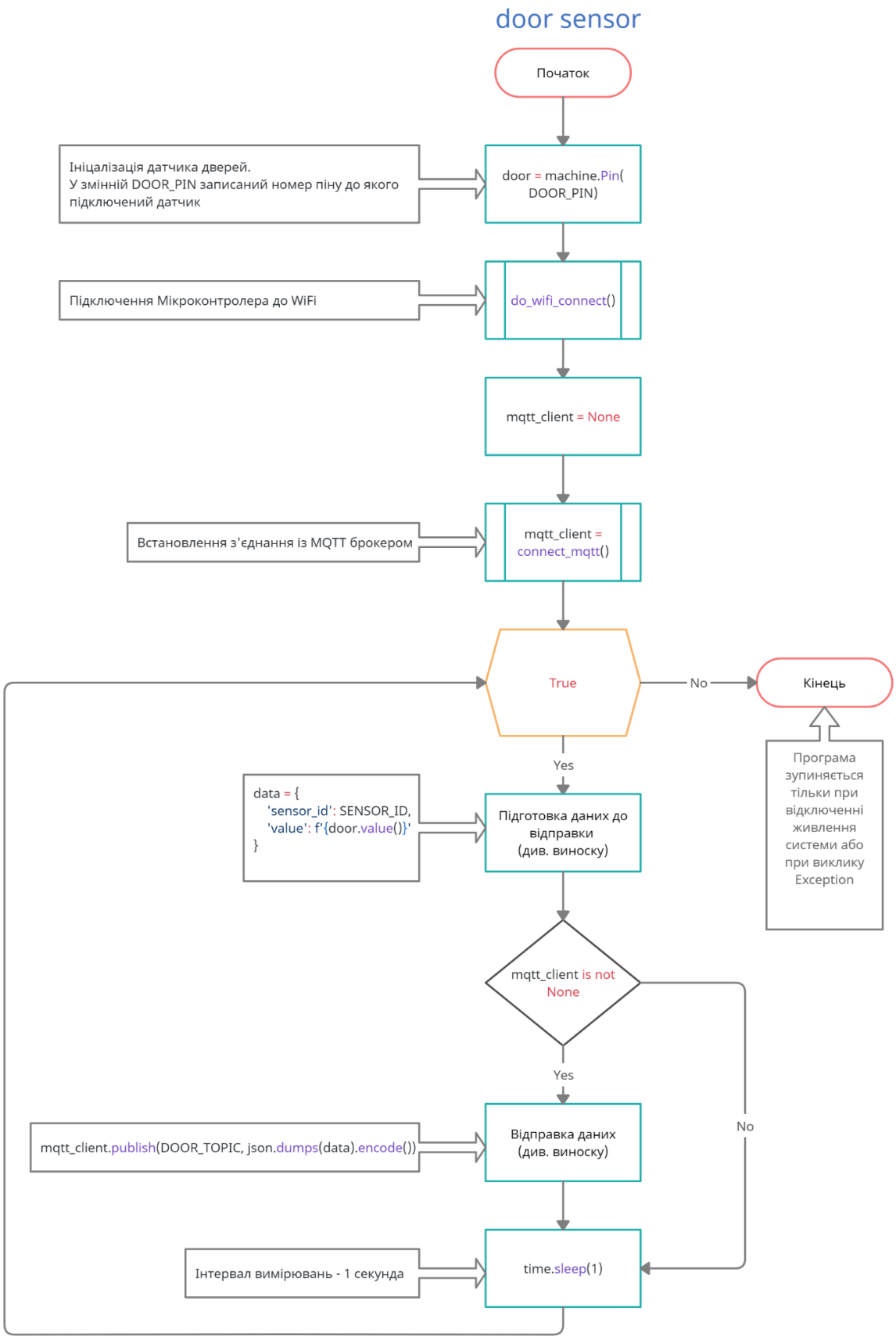

Рис. 18 - Алгоритм роботи датчика дверей

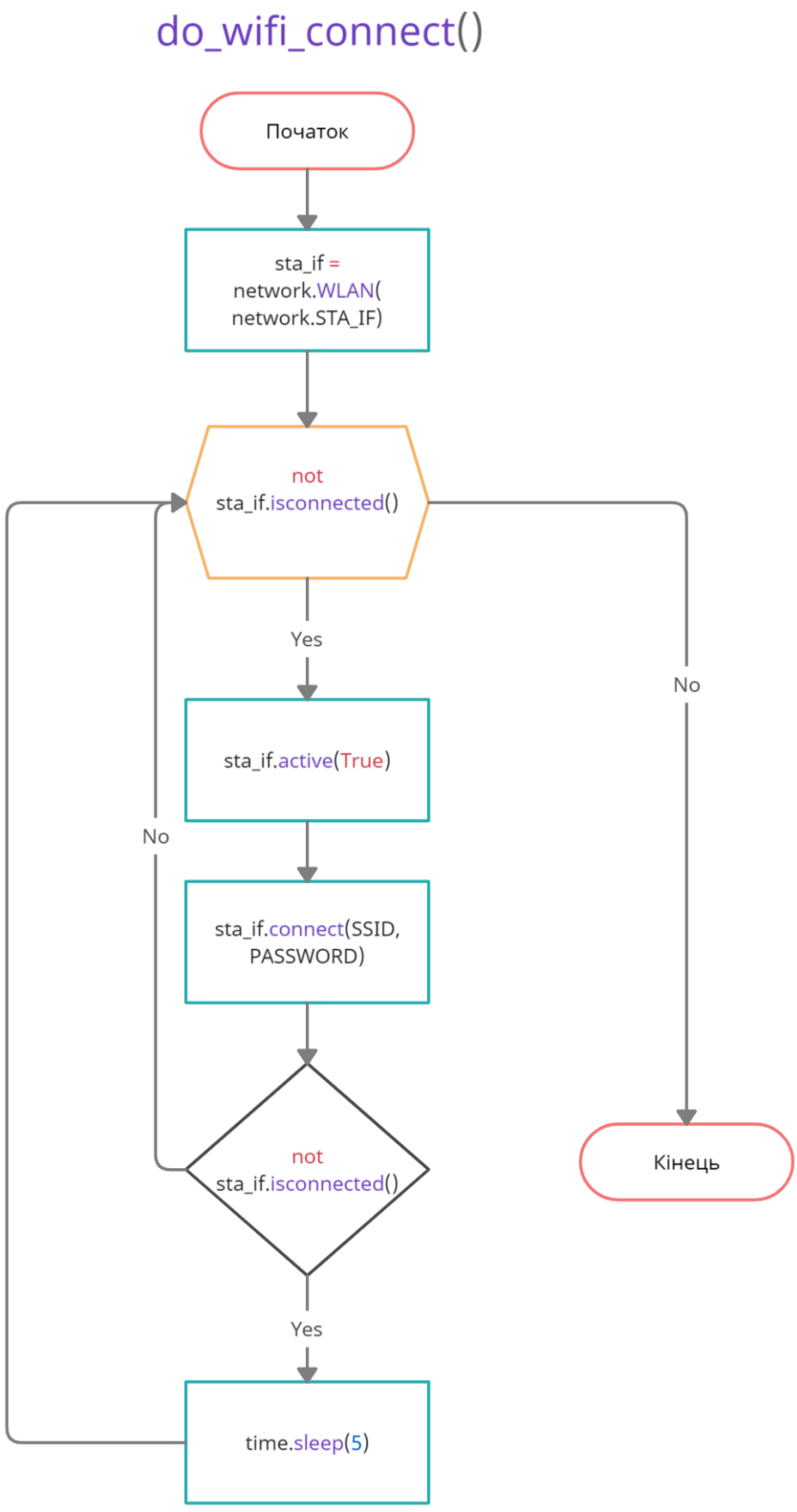

Рис. 19 - Алгоритм функції підключення до WiFi

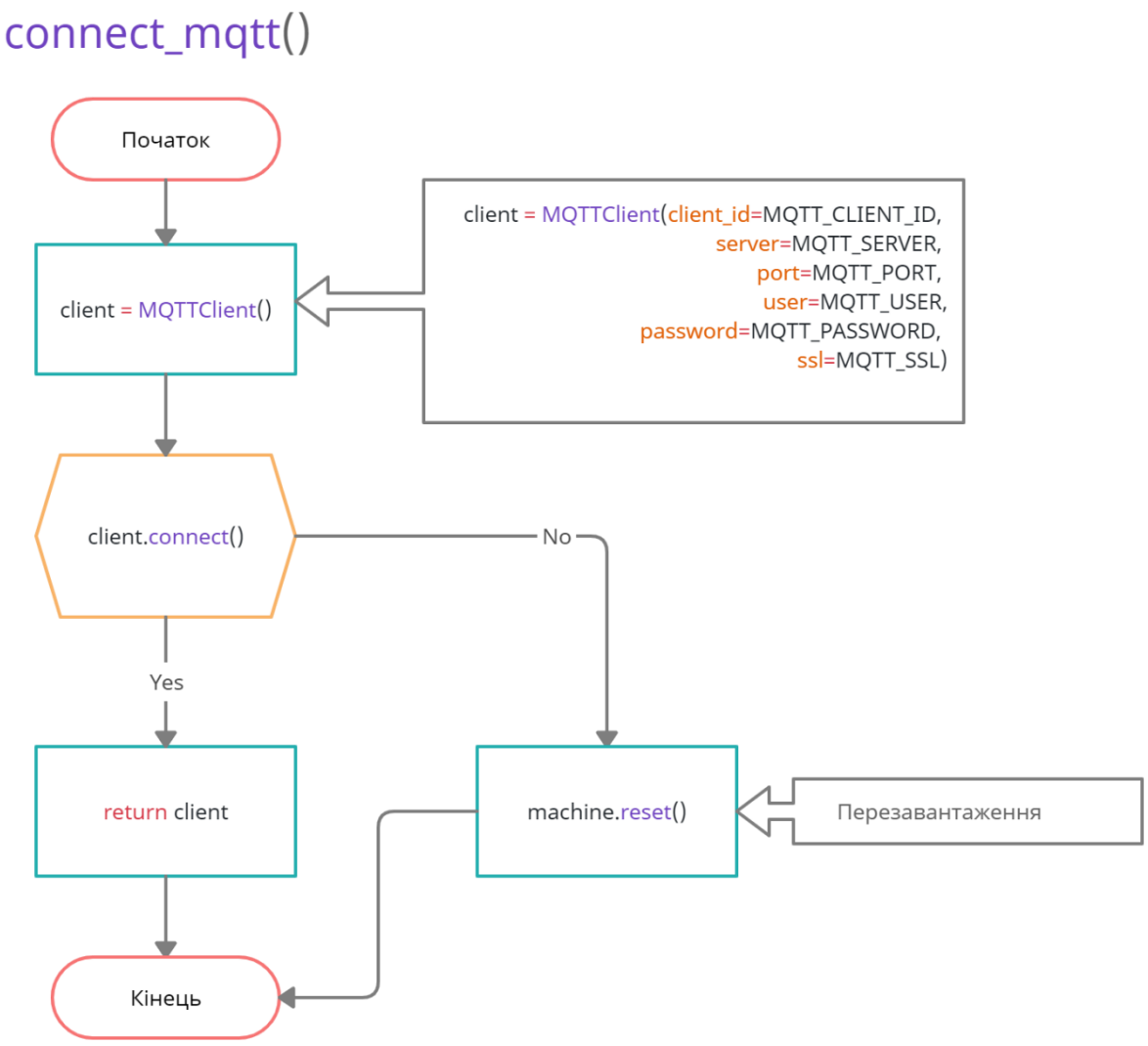

Рис. 20 - Алгоритм функції з’єднання із MQTT брокером

Монтаж датчиків. Магніт кріпиться до дверей, геркон кріпиться до рами дверей (рис. 21 та рис. 22). Датчик температури та вологості краще розміщувати над робочим місцем. Для надійності датчики краще кріпити саморізами (рис. 23), але щоб не робити отвори можна скористатись двохсторонніми клейкими смужками. Плату та елементи живлення потрібно вкласти в будь-який корпус та прикріпити до стіни.

Рис. 21. Спосіб монтажу датчика МС-38

Рис. 22. Демонстрація відчинених дверей із закріпленим датчиком

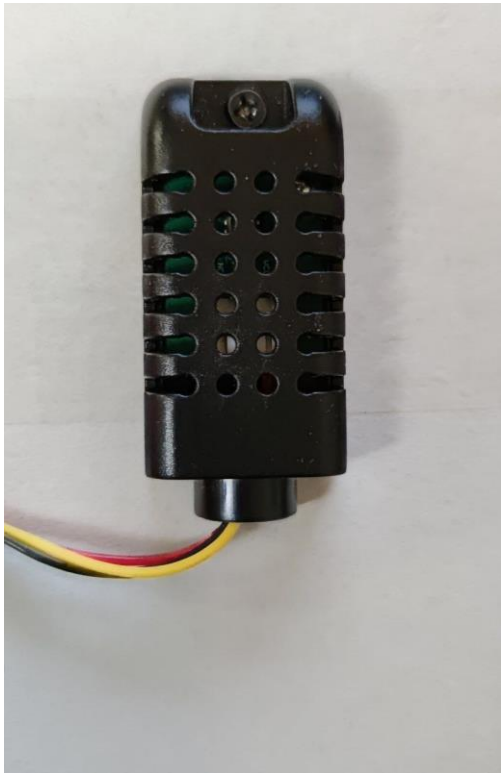

Рис. 23. Спосіб кріплення датчика DHT21

2.6. База даних

Для збереження даних використовується база даних MongoDB. MongoDB – це кросплатформна документоорієнтована база даних, яка забезпечує високу продуктивність, доступність і легку масштабованість. MongoDB працює за концепцією колекції та документа.

Колекція – це група документів MongoDB. Вона є еквівалентом таблиці реляційної СУБД. Колекції не застосовують схему. Документи в колекції можуть мати різні поля. Як правило, усі документи в колекції мають схоже або споріднене призначення.

Документ – це набір пар ключ-значення. Документи мають динамічну схему. Динамічна схема означає, що документи в одній колекції не повинні мати однаковий набір полів або структуру, а загальні поля в документах колекції можуть містити різні типи даних. [14]

Переваги MongoDB над реляційними СУБД:

- MongoDB – це база даних документів, в якій одна колекція містить різні документи. Кількість полів, вміст і розмір документа можуть відрізнятися від одного документа до іншого.
- Явна структура окремих об'єктів.
- Відсутність складних зв’язків.
- Можливість глибокого запиту. MongoDB підтримує динамічні запити до документів, використовуючи мову запитів на основі документів, яка настільки ж потужна, як і SQL.
- Простота масштабування.
- Перетворення/відображення об’єктів програми в об’єкти бази даних не потрібні.

Навіщо використовувати MongoDB?

- Документно-орієнтоване сховище — дані зберігаються в JSON документах.
- Індекс будь-якого атрибута
- Тиражування та висока доступність
- Автоматичне розділення
- Розширені запити

Де використовується MongoDB?

- Big Data
- Управління контентом
- Мобільна та соціальна інфраструктура
- Управління даними користувача [15]

Для зберігання даних було створено 5 документів Mongo DB:

1. User – документ для збереження даних користувача. Поля документа та їх опис наведено у табл. 4

Табл. 4 - документ User

Ключ	Тип поля у Mongo DB	Опис
id	Object ID	Ідентифікатор запису
tg_id	Integer	Числове поле для Telegram ідентифікатора користувача
tg_first_name	String	Строкове поле для імені користувача у Telegram
tg_last_name	String	Строкове поле для прізвища користувача у Telegram
tg_username	String	Строкове поле для псевдоніму користувача у Telegram
date_created	Date	Поле дати створення запису у базі даних
date_modified	Date	Поле дати оновлення запису у базі даних

2. Door Sensor – документ, у якому зберігається інформація та стан датчика дверей. Поля документа та їх опис наведено у табл. 5

Табл. 5 - документ Door Sensor

Ключ	Тип поля у Mongo DB	Опис
id	Object ID	Ідентифікатор запису
name	String	Строкове поле із назв
user	Object (User)	Посилання на запис з користувачем.

value	Int	Числове поле для запису стану датчика
date_created	Date	Поле дати створення запису у базі даних
date_modified	Date	Поле дати оновлення запису у базі даних

3. DHT – документ, у якому зберігається інформація про датчик температури та вологості. Поля документа та їх опис наведено у табл. 6

Табл. 6 - документ DHT

Ключ	Тип поля у Mongo DB	Опис
id	Object ID	Ідентифікатор запису
name	String	Строкове поле із назвою
user	Object (User)	Посилання на запис з користувачем.
date_created	Date	Поле дати створення запису у базі даних
date_modified	Date	Поле дати оновлення запису у базі даних

4. Temperature Value – документ, зв’язаний із документом DHT, у якому зберігаються покази датчика температури. Поля документа та їх опис наведено у табл. 7.

5. Humidity Value – документ, зв’язаний із документом DHT, у якому зберігаються покази датчика температури. Поля документа та їх опис наведено у табл. 7.

Табл. 7 - документ Temperature Value та Humidity Value

Ключ	Тип поля у Mongo DB	Опис
id	Object ID	Ідентифікатор запису
sensor	Object (User)	Посилання на запис з датчиком.
value	Double	Числове поле для запису показів датчика
date_created	Date	Поле дати створення запису у базі даних

Схема бази даних наведена рис. 24.

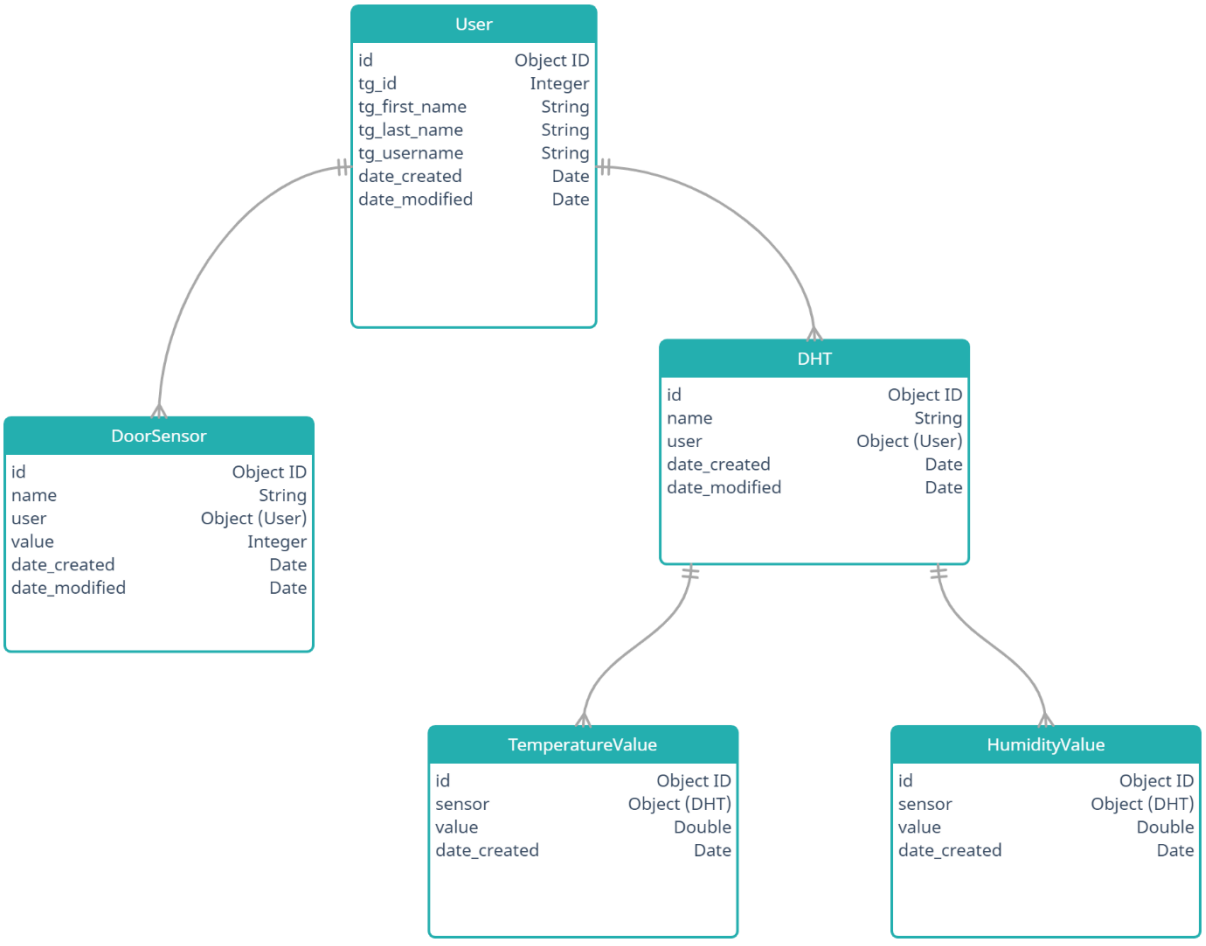

Рис. 24 - Схема бази даних

2.7. Розгортання системи на хмарних серверах

Для повноцінної та постійної роботи системи потрібно розгорнути сервіси на хмарних серверах. А саме backend, базу даних, Telegram бота та MQTT брокера.

Для MQTT брокера було вибрано хмарний сервіс CloudMQTT. CloudMQTT – це сервіс із серверами, які керуються завдяки Mosquitto у хмарі. Mosquitto – це брокер повідомлень з відкритим вихідним кодом, який реалізує протокол MQTT. У сервісу великий вибір тарифних планів, тому, використовуючи цей сервіс проблем із масштабуванням не буде. Для реалізації даного проєкту використовується тарифний план $5/міс. Він включає в себе підтримку 25 з'єднань одночасно та швидкість 20 Кбіт в секунду.

База даних Mongo DB розгорнута з використанням хмарного сервісу MongoDB Atlas. MongoDB Atlas — це хмарна служба баз даних, безпосередньо від розробників самої MongoDB. Atlas спрощує розгортання та керування базами даних, пропонуючи універсальність, необхідну для створення стійких і продуктивних глобальних додатків. Даний сервіс пропонує багато тарифних планів, але також присутній безоплатний тариф для невеликих проєктів. Тому для прототипу системи використовується саме такий тариф.

Для розгортання Backend частини проєкту та Telegram бота було вибрано сервіс Heroku. Heroku – хмарна платформа для швидкого розгортання проєкту. Сервіс було вибрано тому, що він дає можливість розгорнути проєкт за декілька хвилин без необхідності довгого налаштування. Достатньо лише створити файл з назвою Procfile, який визначає команди, що виконує програма під час запуску. Procfile для Telegram бота наведений на рис. 25

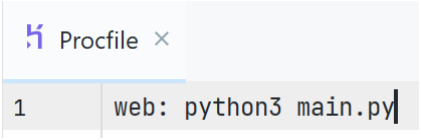

Рис. 25 - Procfile для Telegram бота

«web» це тип процесу, а «python3 main.py» запуск файлу main.py за допомогою третьої версії мови програмування Python.

Для Backend частини проєкту Procfile (рис. 26) має інший тип процесу «worker». Різниця між web та worker у тому що web процес може отримувати зовнішній HTTP трафік. Це необхідно для того щоб обробляти HTTP запити від Telegram Bot API.

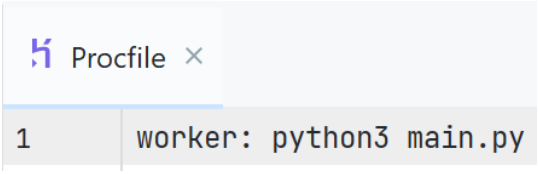

Рис. 26 - Procfile для Backend частини проєкту

2.8. Взаємодія користувача із Telegram ботом

Для управління системою та отримання даних використовується мессенджер Telegram. Telegram – це хмарний сервіс для обміну повідомленнями. Використання даного месенджеру є вдалим рішенням, так як він є поширеним і користувачам не потрібно встановлювати додаткове програмне забезпечення. У Telegram є можливість створювати ботів. Боти — це сторонні програми, які працюють усередині месенджера Telegram. Користувачі можуть взаємодіяти з ботами, надсилаючи їм повідомлення, команди та вбудовані запити. Боти керуються за допомогою HTTP запитів до Telegram Bot API. Тому, використовуючи Telegram бота, можна взаємодіяти із користувачами месенджеру.

Telegram бот (рис. 27) написаний мовою програмування Python з використанням фреймворку aiogram. Aiogram — це досить простий і повністю асинхронний фреймворк для Telegram Bot API, написаний на Python з використанням вбудованої в Python бібліотеки asyncio та асинхронного фреймворка aiohttp. Використання aiogram робить ботів швидшими та простішими. Програмний код Telegram бота наведений у додатку Д.

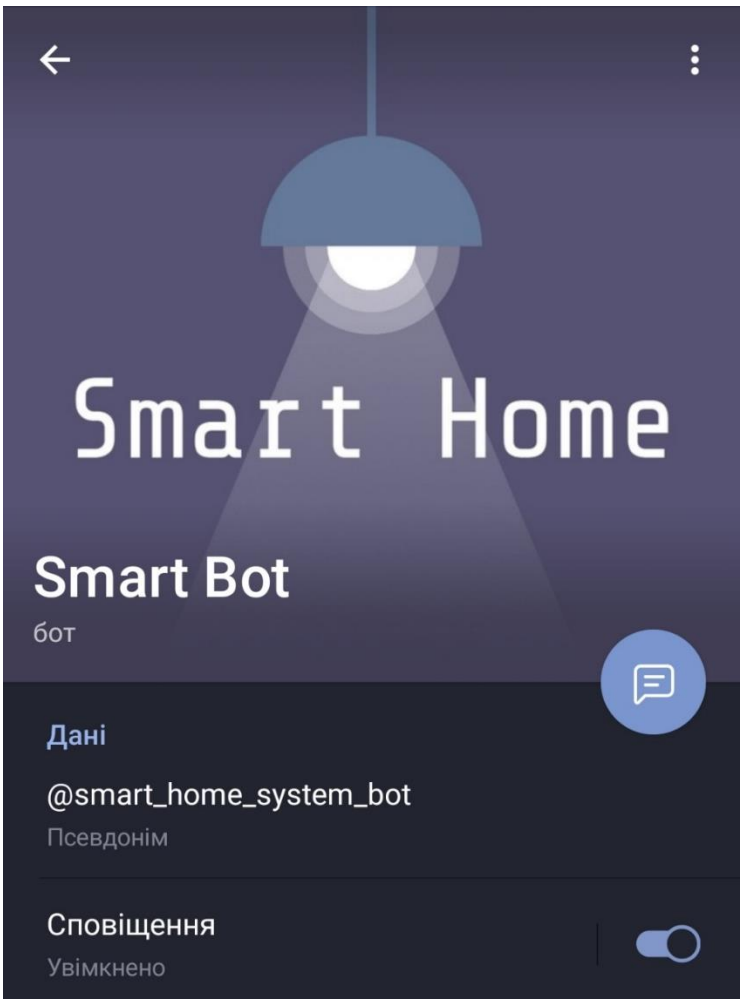

Рис. 27 - Сторінка бота у месенджері Telegram

Початок роботи з ботом. При введенні команди /start, бот вітає користувача, автоматично реєструючи його та записуючи всі потрібні дані у базу даних Mongo DB. Відразу після реєстрації для нас відкривається доступ до клавіатури бота (рис. 28), яка спрощує взаємодію з ним.

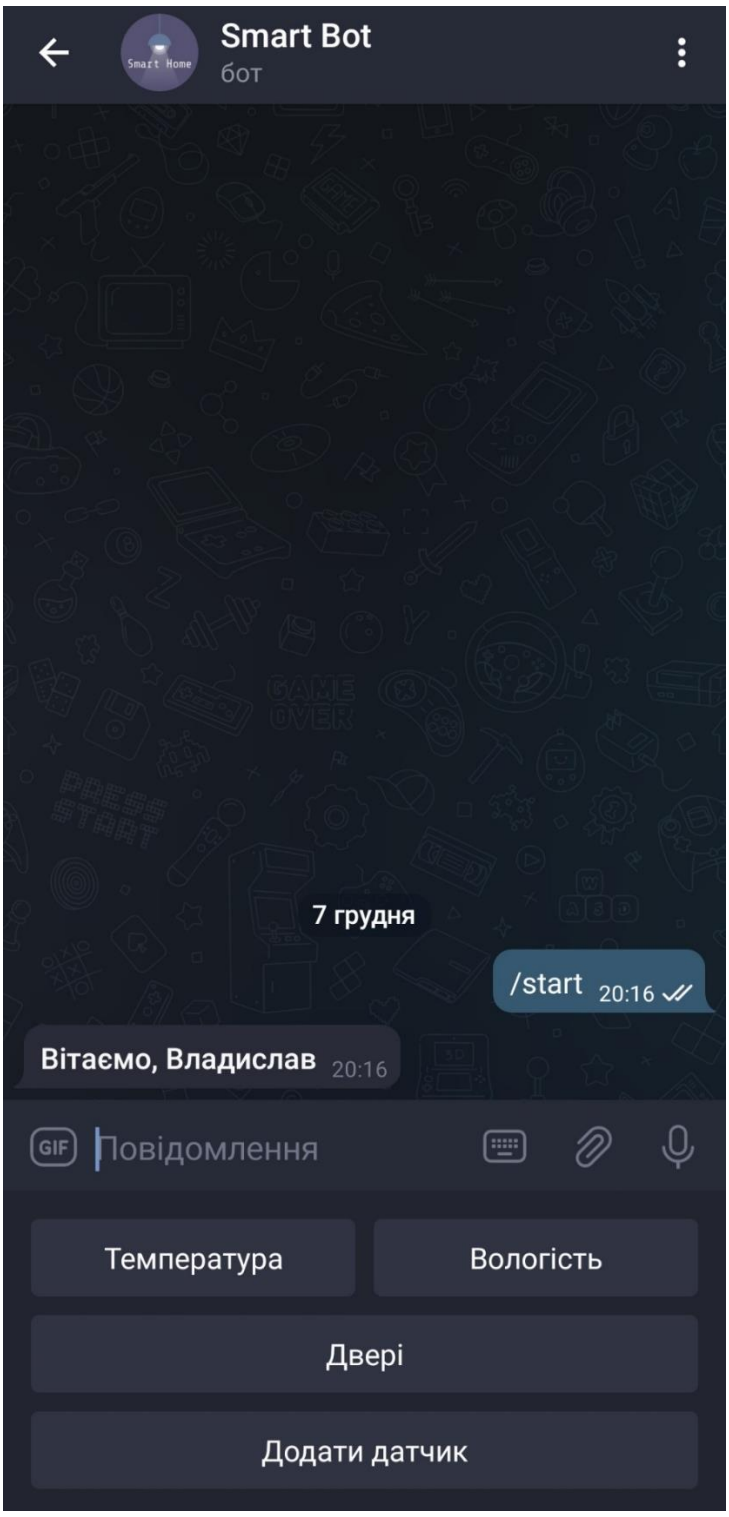

Рис. 28. Початок роботи з ботом

Натискаючи на кнопки «Температура», «Вологість» та «Двері» отримуються покази датчиків, але так як вони ще не додані у користувача, то відправляється повідомлення «Датчиків не знайдено» (рис. 29).

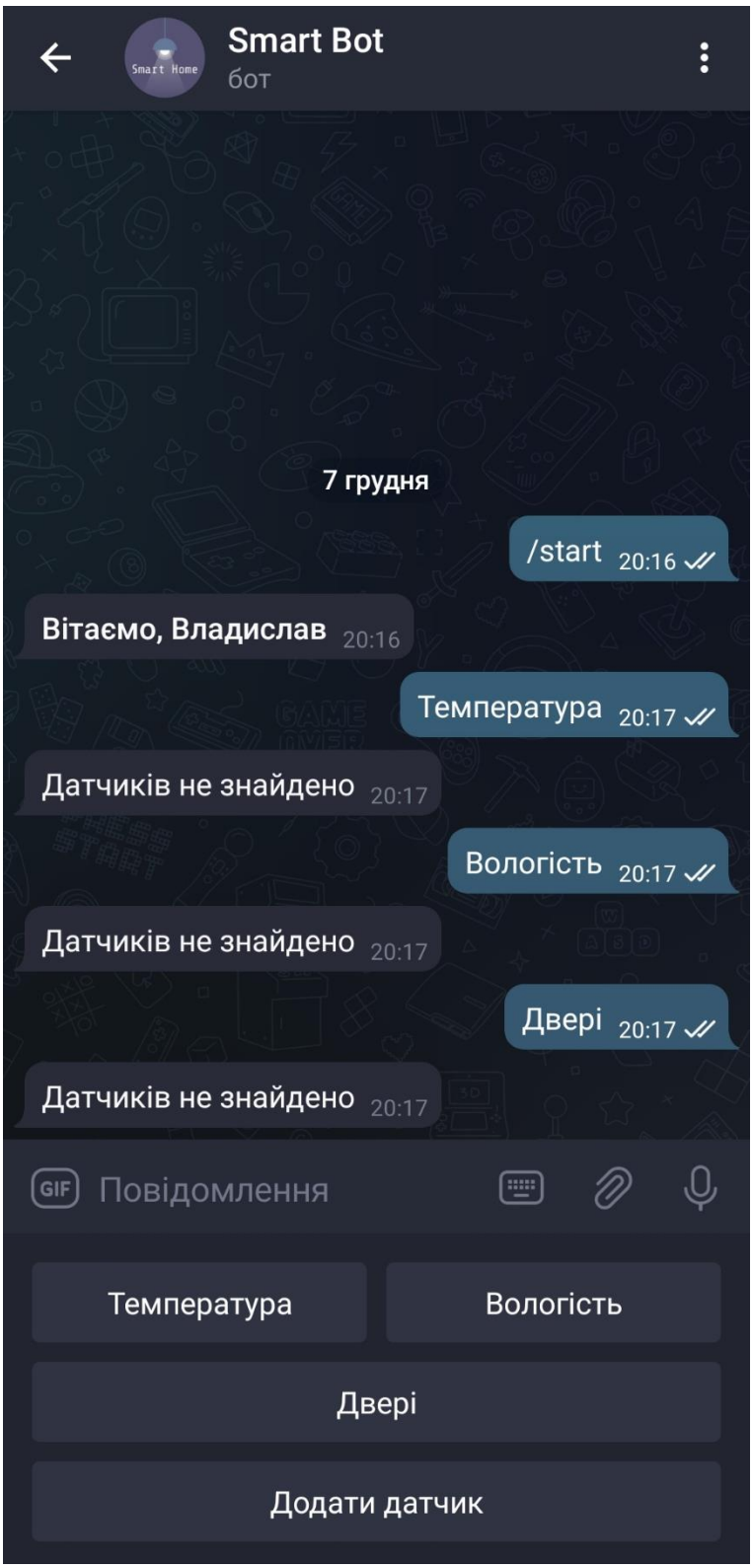

Рис. 29 - Повідомлення про відсутність датчиків

Тому датчики необхідно додати. Для цього треба натиснути кнопку «Додати датчик», вибрати його тип (рис. 30) та вказати назву (рис. 31).

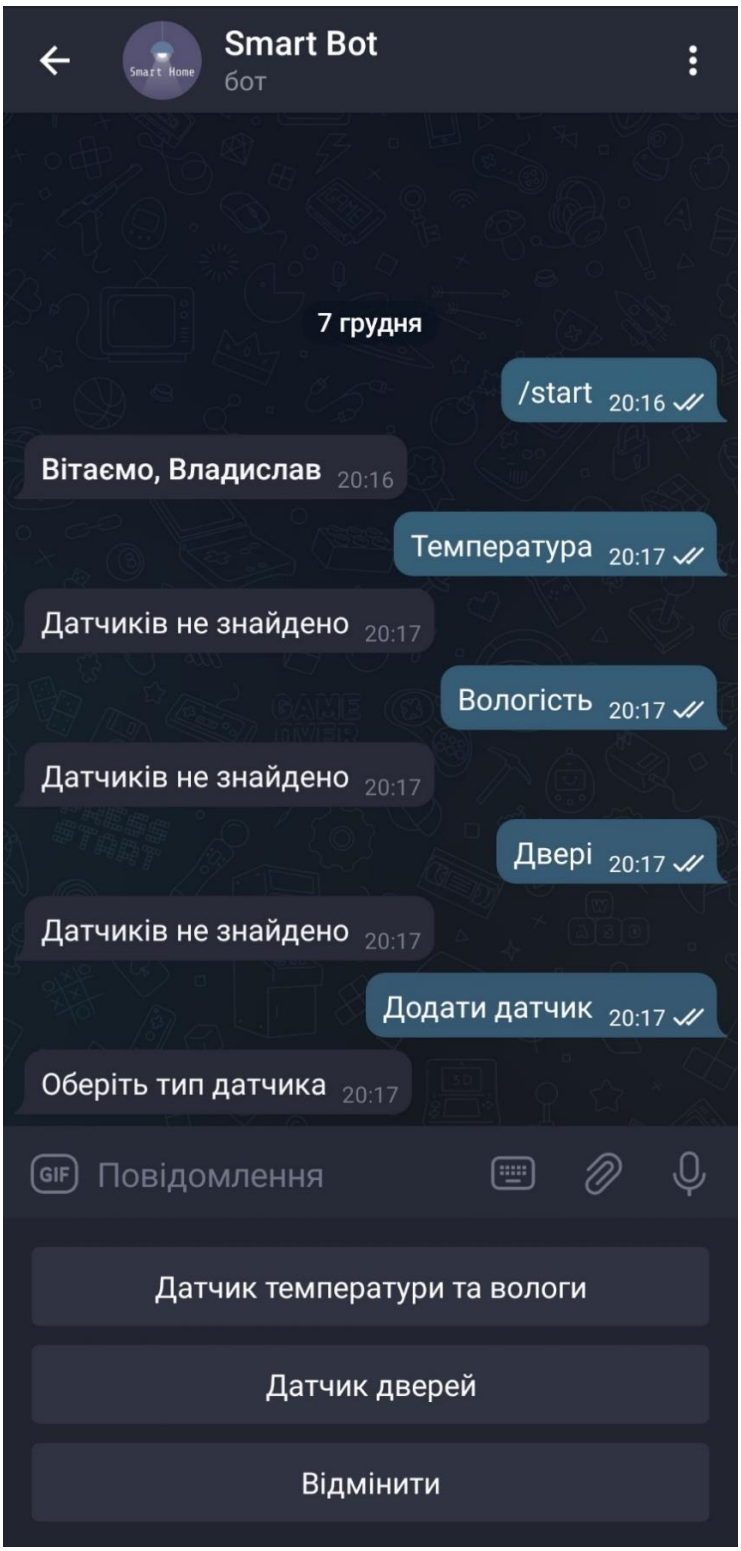

Рис. 30 - Вибір типу датчика

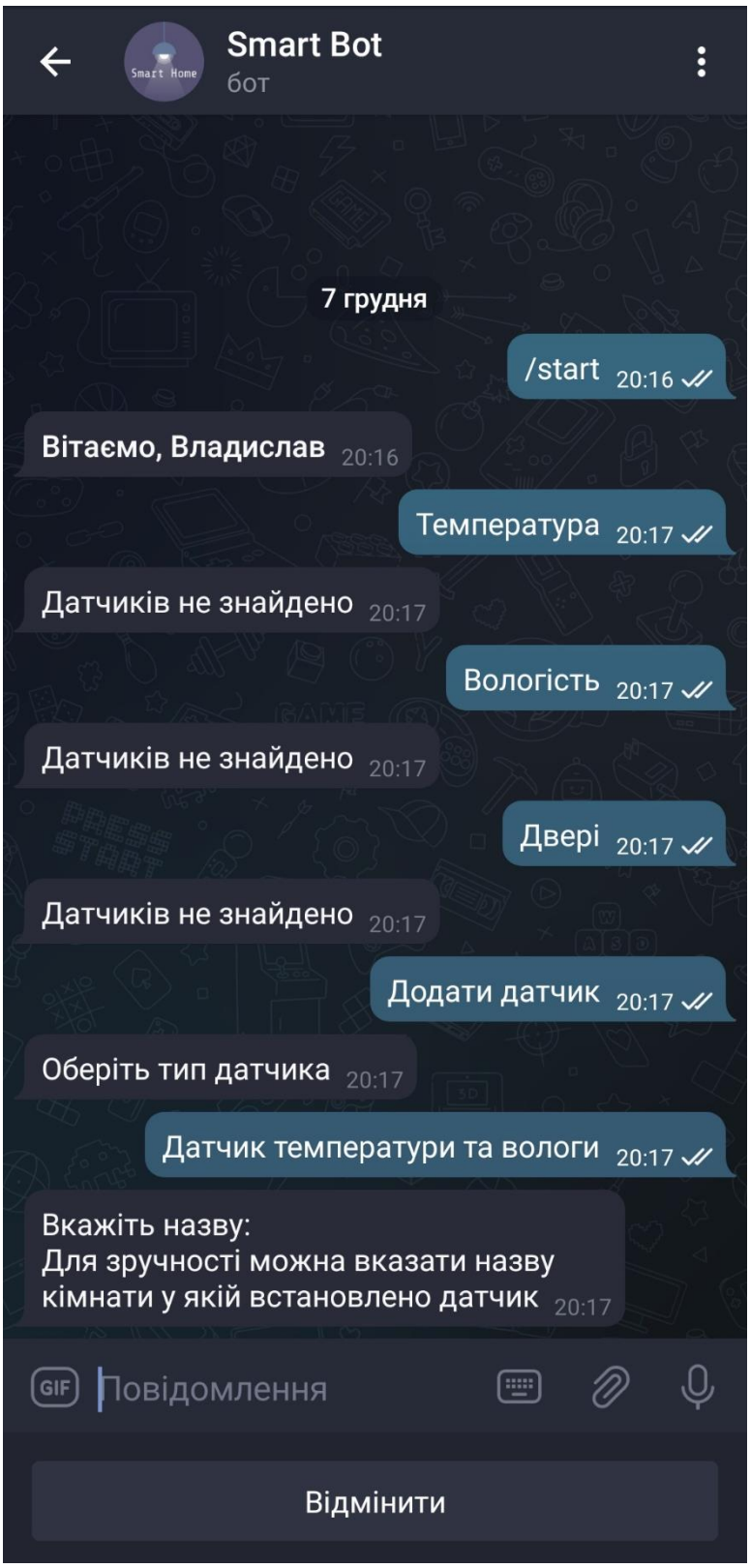

Рис. 31 - Вказування назви датчика

У результаті бот відправляє повідомлення про успішне додавання датчика (рис. 32) у систему і відправляє користувачу ідентифікатор датчика, який потрібно вказати у файлі config.json при налаштуванні датчика. За таким самим принципом додаються датчики дверей (рис. 33 та рис. 34).

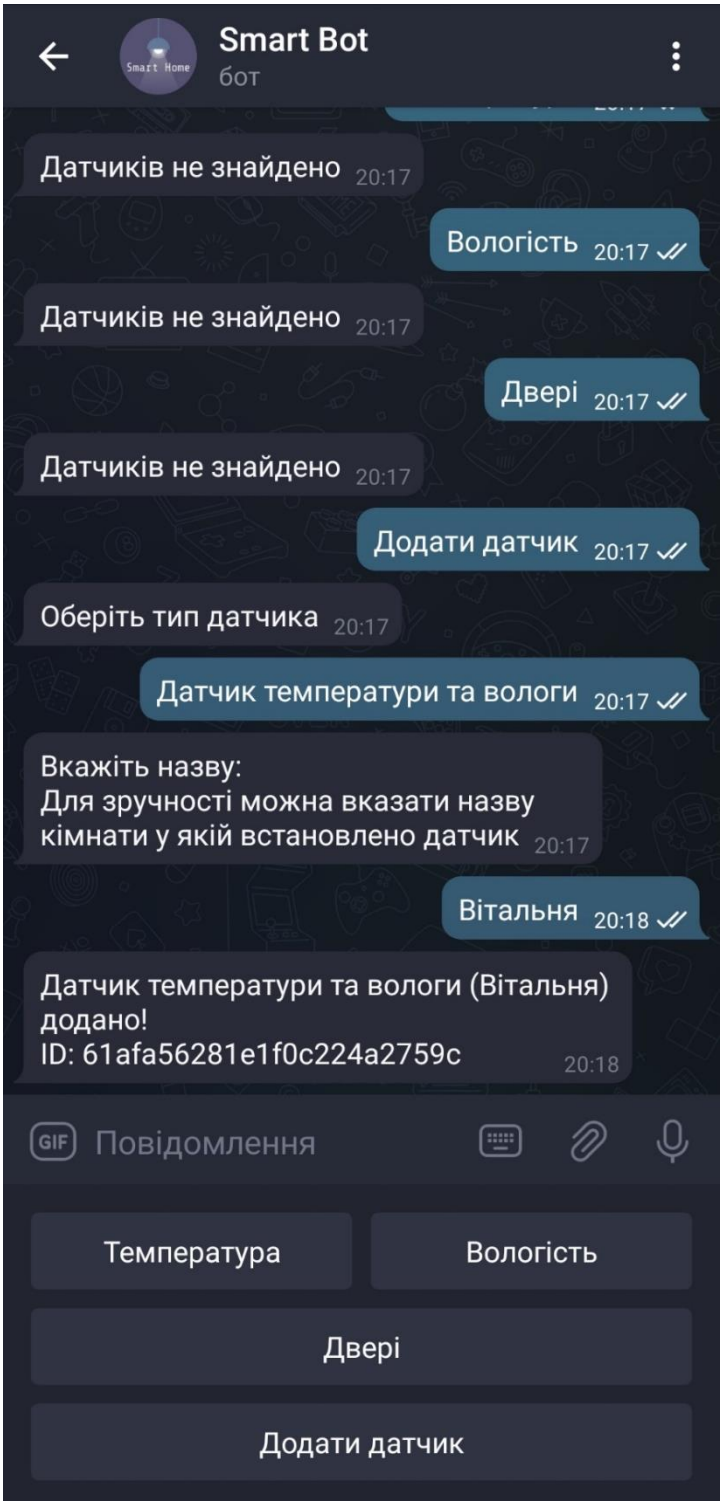

Рис. 32 - Результат додавання датчика

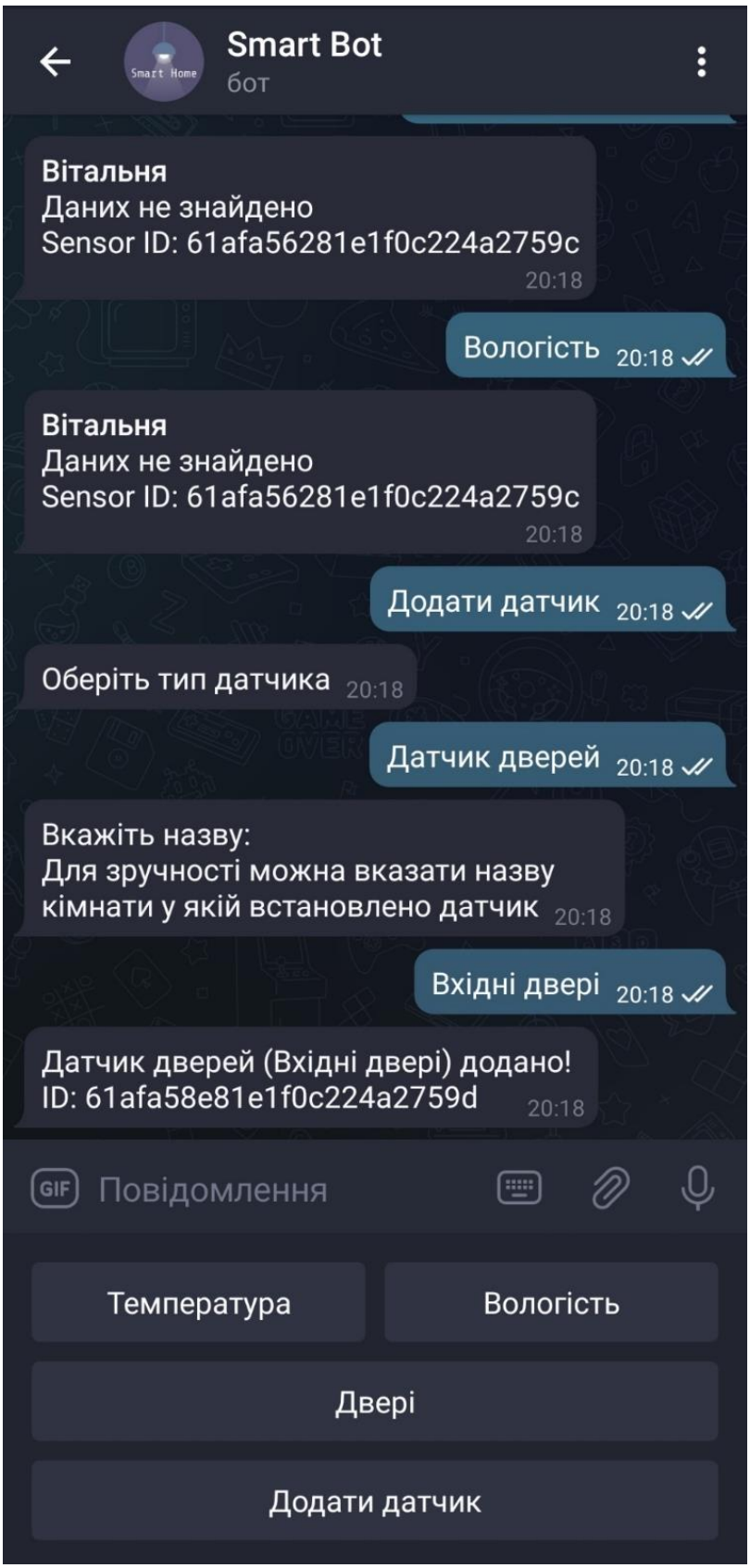

Рис. 33 - Додавання датчика вхідних дверей

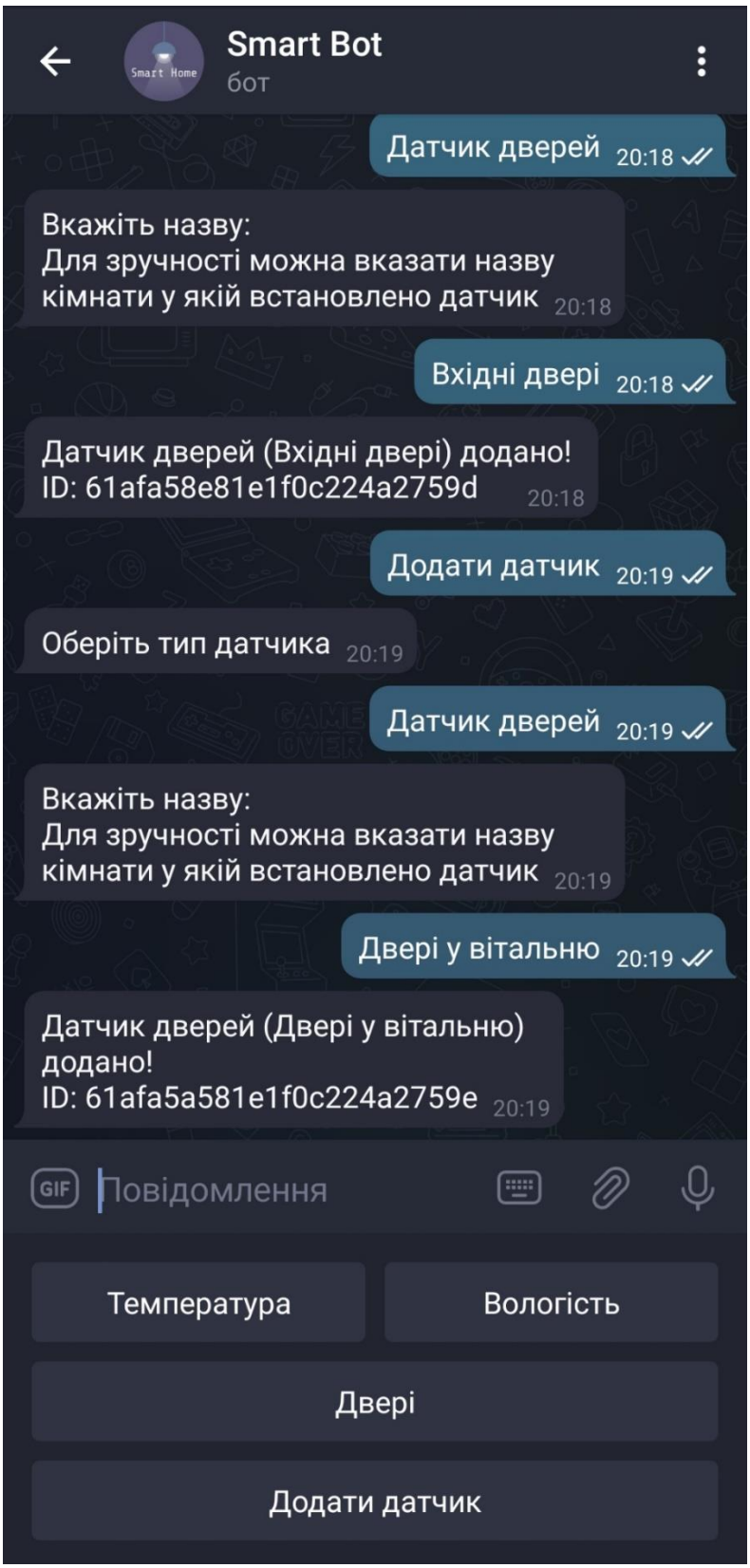

Рис. 34 - Додавання датчика дверей у вітальні

Після додавання датчиків та їх налаштування, користувач може отримувати покази датчиків (рис. 35 та рис. 36).

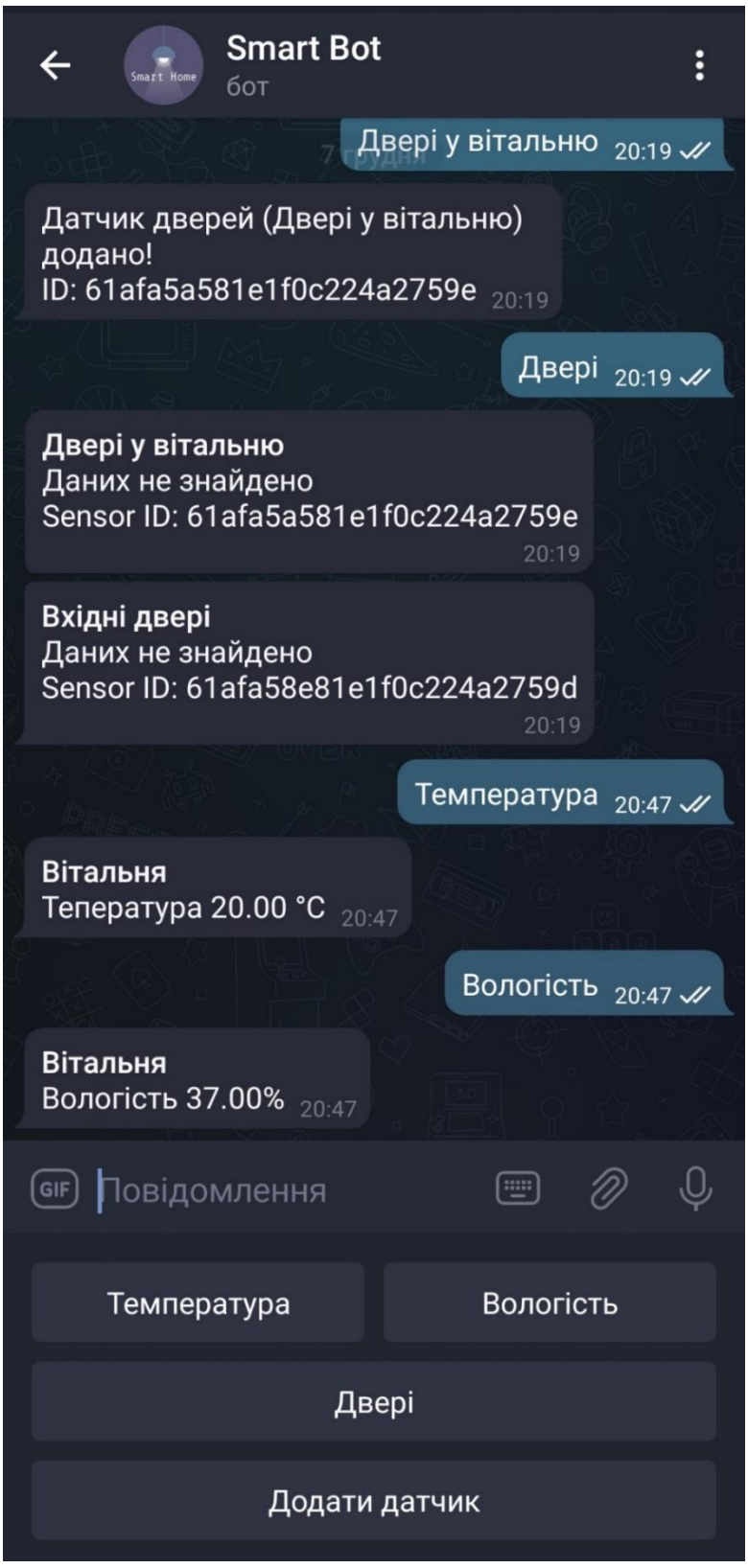

Рис. 35 - Значення датчика температури та вологості

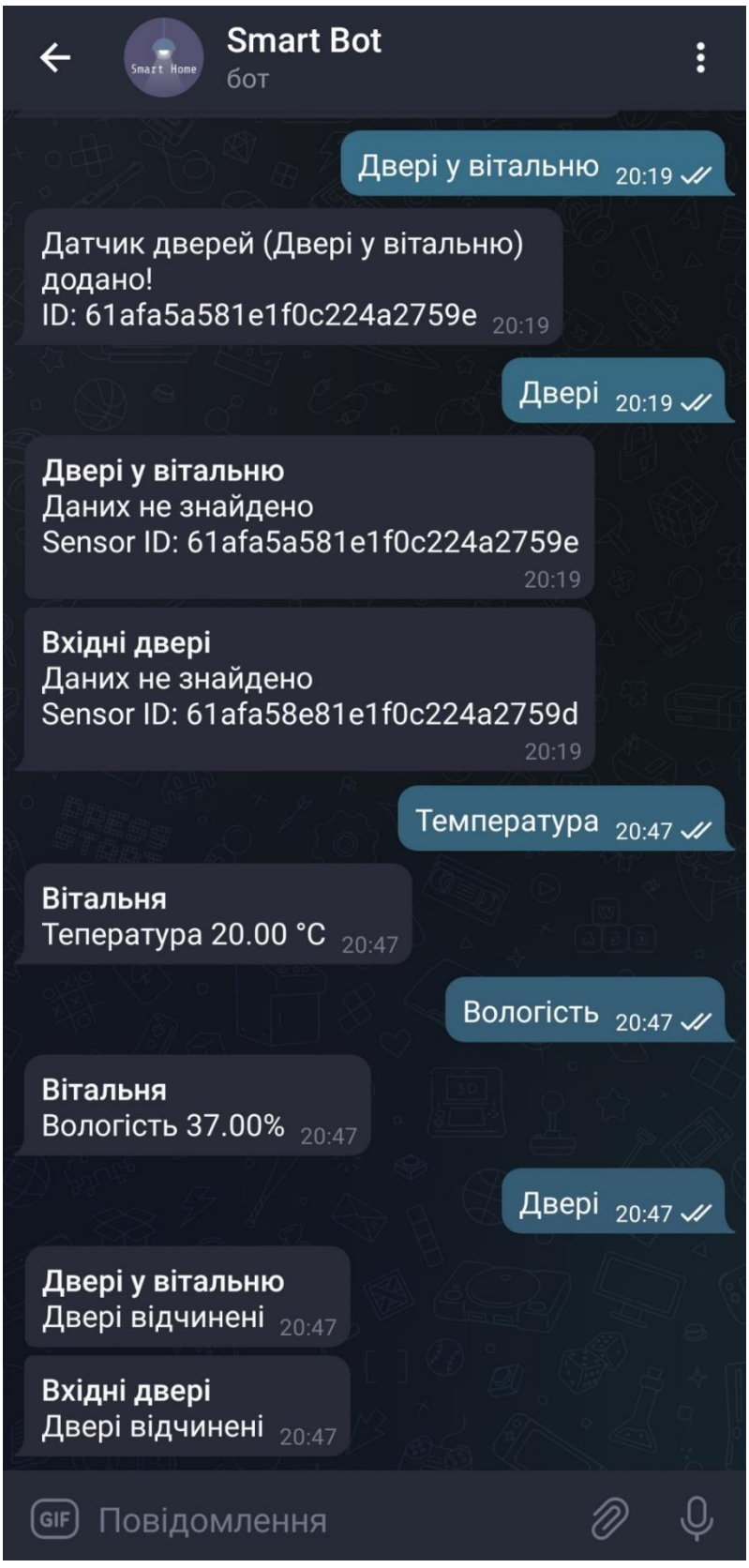

Рис. 36 - Отримання інформації з датчиків дверей

Коли двері будуть відчинятися та зачинятися, бот буде повідомляти користувача про такі дії (рис. 37)

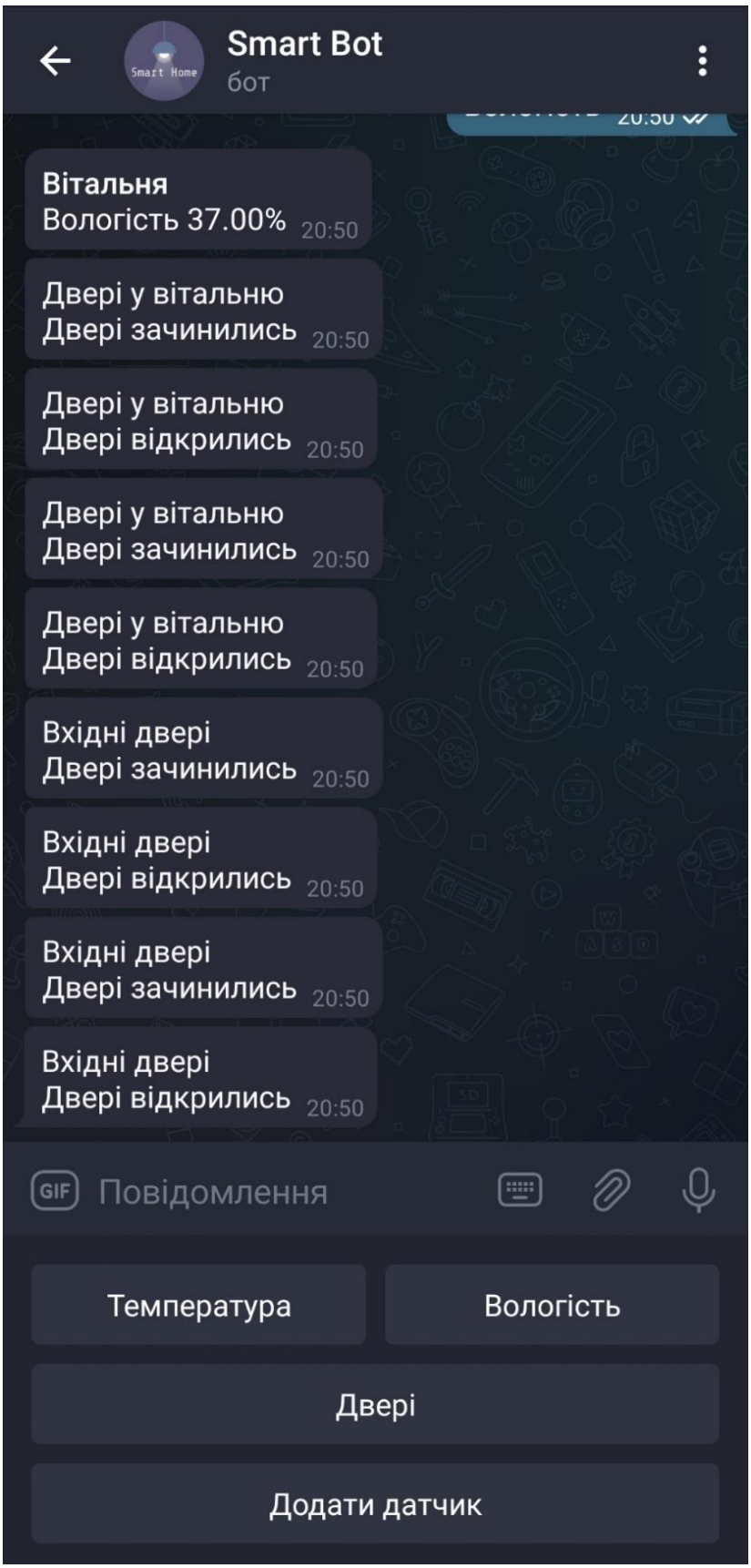

Рис. 37 - Повідомлення про відкривання/закривання дверей

3. РОЗРОБКА СТАРТАП-ПРОЄКТУ

У цьому розділі буде проведено аналіз стартап проєкту «Інтелектуальна система моніторингу мікроклімату та безпеки будинку». Для аналізу використаний алгоритм наведений в [17].

3.1. Опис ідеї проєкту

Зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів описано у таблиці 8.

Таблиця 8. Опис ідеї стартап-проєкту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Інтелектуальна система моніторингу мікроклімату та безпеки будинку	1. Охоронні системи	Простота налаштування, автономність компонентів, зменшення витрат на обслуговування та підтримки системи.
	2. Системи розумного будинку	

Запропонована система моніторингу мікроклімату та безпеки будинку забезпечить простоту налаштування, автономність компонентів системи, що в свою чергу робить її більш стійкою та дозволяє зменшити витрати на обслуговування та підтримку. У таблиці 1 приведені основні напрямки застосування запропонованої системи. Основними споживачами є люди які хочуть обладнати свій будинок системою моніторингу мікроклімату та в той же час контролювати безпеку будинку.

У таблиці 9 наведена інформаційна карта стартап проєкту.

Таблиця 9. Інформаційна картка стартап-проєкту

Назва проєкту	Інтелектуальна система моніторингу мікроклімату та безпеки будинку
Автор	Олійник В. С.
Анотація	Система, завдяки своїй простоті, дозволяє зберегти кошти клієнта та самостійно налаштувати її.
Термін реалізації	1 рік

Необхідні ресурси	Людські, фінансові.
Опис проблеми, яку вирішує стартап-проєкт	Поточні системи моніторингу мікроклімату та безпеки будинків є надто дорогими та складними у самостійному налаштуванні.
Ціль	Зробити системи моніторингу доступними для більшої кількості населення.
Очікуваний результат	Реалізація системи за допомогою якої можна зручно моніторити та контролювати мікроклімат та безпеку будинку

Аналіз потенційних слабких, нейтральних та сильних техніко-економічних характеристик ідеї порівняно із пропозиціями конкурентів є підґрунтям для формування його конкурентоспроможності у таблиці 10.

W – слабка сторона

N – нейтральна сторона

S – сильна сторона

Таблиця 10. Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*				*W*	*N*	*S*
		Мій проєкт	*Ajax Systems*	*BroadLink*	*Fibaro*			
1	Простота налаштування та встановлення (від 1 до 10, де 1 - складно, 10 - просто)	10	10	8	7			+
2	Якість підтримки продукту (від 1 до 10, де 1 – погана якість, 10 – хороша якість)	7	10	9	9	+		
3	Широкий вибір компонентів системи (від 1 до 10, де 1 –	3	10	9	8	+		

	вузький вибір, 10 – широкий вибір)							
4	Надійність системи	Висока	Висока	Висока	Висока		+	
5	Вартість	Низька	Висока	Середня	Середня			+

У порівнянні із головними конкурентами, перевагою даної системи є простота налаштування та низька вартість. Недоліками є нижча якість підтримки продукту та вузький вибір компонентів системи.

3.2. Технологічний аудит ідеї проєкту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проєкту (табл. 11).

Таблиця 11. Технологічна здійсненність ідеї проєкту

№ п/п	*Ідея проєкту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Розробка архітектури взаємодії системи	Огляд та аналіз існуючих реалізацій систем	Наявні	Доступні
		Розробка оптимальної архітектури системи		
		Тестування розробленої архітектури системи		
2	Розробка програмного забезпечення	Вибір програмних технологій	Наявні	Доступні
		Безпосередньо розробка програмного забезпечення		
Обрана технологія реалізації ідеї проєкту: Методика реалізації ідеї проєкту спирається на розробці архітектури взаємодії системи та розробці програмного забезпечення.				

За результатами аналізу таблиці можна зробити висновок, що реалізація ідеї можлива так як всі технології у наявності та доступності.

3.3. Аналіз ринкових можливостей запуску стартап-проєкту

Проведемо аналіз ринкових загроз, які можуть перешкодити реалізації проєкту, та ринкових можливостей, які можна використати під час ринкового впровадження проєкту.

Проаналізуємо попит: наявність попиту, обсяг, динаміка розвитку ринку (табл. 12).

Таблиця 12. Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	5
2	Загальний обсяг продаж, грн/ум.од	5000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Високий рівень конкуренції
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	30%

За результатами таблиці можна зробити висновок, що вихід на ринок є привабливим. Високий процент рентабельності та відсутність серйозних обмежень для входу дає змогу швидко вийти в прибуток.

Визначимо потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (табл. 13).

Таблиця 13. Характеристика потенційних клієнтів стартап-проєкту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Наявність на ринку недорогих системи моніторингу мікроклімату та безпеки будинку	Малий бізнес, люди з невеликим бюджетом	Особливості експлуатації	Простота у налаштуванні, невисока вартість та надійність системи

Формування ринку визначається потребою в невисокій вартості продукту, надійності та простоті у налаштуванні. Цільовою аудиторією є малий бізнес та люди з невеликим бюджетом.

У таблиці 14 наведено фактори ринкового середовища, що перешкоджають ринковому впровадженню проєкту.

Таблиця 14. Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Конкуренція	Ширший асортимент продуктів у конкурентів	Розширення власного асортименту, підвищення якості власних продуктів
2	Економічна нестабільність	Економічний стан країни, де виробляється продукт, мала купівельна спроможність населення.	Спрощення комплектації системи. Залишаються тільки основні датчики для контролю та моніторингу. Оптимізація витрат на розробку.

3	Якість	Зниження якості елементів, що закупаються	Пошук нових постачальників
4	Технології	Поява нових технологій та можливостей у конкурентів	Удосконалення продукції та проведення R&D діяльності
5	Відсутність попиту	Ймовірність того що клієнти не будуть купувати продукт	Маркетингова діяльність, пошук нових клієнтів та шляхів збуту

Основна загроза бізнесу – висока конкуренція. У конкурентів ширший асортимент продуктів, тому потрібно якнайшвидше розширювати власний асортимент та підвищувати якість власних продуктів. Також важливим моментом є попит на продукт. Для вирішення цього фактору потрібно проводити активні маркетингові кампанії.

У таблиці 15 наведено фактори ринкового середовища, що сприяють ринковому впровадженню проєкту.

Таблиця 15. Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Збільшення попиту	Корисність системи	Збільшення кількості систем, що випускаються. Залучення нових інвестицій
2	Зростання рівня доходу населення	Збільшення середнього доходу	Удосконалення продукту та збільшення ціни
3	Технологічний розвиток. Впровадження нових технологій	Модернізація елементної бази	Вкладання коштів в нові технології, розробка додаткових функцій
4	Зацікавленість продуктом за кордоном	Вихід на міжнародний ринок	Відкриття відділів продажу за кордоном та збільшення оборотів виробництва

Головним фактором можливостей є збільшення попиту на продукт, що призведе до збільшення кількості клієнтів, а також кількості нових інвестицій. Зацікавленість продуктом за кордоном дає змогу вийти компанії на міжнародний ринок. Технологічний розвиток дозволить провести модернізацію та удосконалення системи.

Наступним кроком проводиться аналіз пропозиції, а саме визначити загальні риси конкуренції на ринку (табл. 16).

Таблиця 16. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Олігополія	Існує декілька конкурентів на ринку	Підвищення якості продукту
2. Міжнародна	Існує багато компаній на міжнародному ринку.	Підвищення якості продукту та оптимізація витрат на виготовлення
3. Внутрішньогалузева	Конкуренція між підприємствами в межах однієї галузі	Маркетингова діяльність для залучення клієнтів
4. Товарно-видова	Конкуренція товарів одного виду	Створення модифікацій та додавання нових компонентів системи
5. Цінова	Використання ціни для приваблення клієнтів	Пошук постачальників, які пропонують вигідніші ціни на компоненти.
6. Марочна	Товар, що пропонують конкуренти є подібним	Маркетингова діяльність; покращення якості продукту; створення власної торгової марки

Аналіз конкуренції показав, що запропонована система демонструє високу конкурентоспроможність. Тип конкуренції – олігополія. Рівень конкурентної боротьби – міжнародна з внутрішньогалузевою ознакою. Конкуренція за видами товарів – товарно-видова, а за характером конкурентних переваг – цінова.

Аналіз умов конкуренції в галузі за моделлю 5 сил М. Портера навдено у табл. 17.

Таблиця 17. Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Складові аналізу	*Ajax Systems, BroadLink, Fibaro*	Компанії, що які мають широкий асортимент	Велика кількість постачальників	Малий бізнес та клієнти з обмеженим бюджетом	Впровадження власного виробництва компонентів
Висновки	Конкуренція існує, але вона прийнятна так як немає монополістів	Розширення власного асортименту та вихід на ринок з привабливішою ціною.	Постачальник не диктує умови на ринку	Залежність від клієнтів висока, клієнт обирає рішення, яке найкраще йому підходить	Можливі зміни вектору розвитку індустрії, що потребуватимуть інших рішень

За результатами аналізу таблиці 17 можна відмітити, що на ринку конкуренція існує, але вона прийнятна так як немає монополістів.

У таблиці 18 наведено обґрунтування факторів конкурентоспроможності, перелік факторів конкурентоспроможності, що базується на основі аналізу конкуренції, а також із урахуванням характеристик ідеї проєкту, вимог споживачів до товару та факторів маркетингового середовища.

Таблиця 18. Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)*
1	Якість	Збільшена якісь завдяки використанню останніх технологій
2	Надійність	Збільшена надійності в порівнянні з конкурентами
3	Швидкість налаштування	Не потребує спеціаліста для встановлення та налаштування
4	Зручність використання	Система використовує актуальний месенджер для сповіщення та моніторингу
5	Ціна	Ціна нижча ніж у конкурентів

Підвищена надійність, якість та хороша цінова політика дає можливість вийти на ринок, та отримати частку покупців для яких ці параметри є важливими.

За визначеними факторами конкурентоспроможності (табл. 18) проводиться порівняльний аналіз сильних та слабких сторін стартап-проєкту (табл. 19).

Таблиця 19. Порівняльний аналіз сильних та слабких сторін

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Якість	17				+			
2	Надійність	17				+			
3	Швидкість налаштування	18					+		
4	Зручність використання	18							+
5	Ціна	19							+

Останнім етапом ринкового аналізу можливостей впровадження проєкту є складання SWOT-аналізу (табл. 20) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 19).

Таблиця 20. SWOT-аналіз стартап-проєкту

Сильні сторони: Ціна Оптимальна якість Зручність налаштування Надійність	Слабкі сторони: Вузький асортимент компонентів Поява нових технологій Відсутність торгової марки
Можливості: Збільшення попиту Покращення системи Вихід на міжнародний ринок	Загрози: Конкуренція Збільшення ціни у постачальників Відсутність попиту

За результатами SWOT-аналізу було визначено, що за рахунок сильних сторін продукт має високу конкурентоспроможність на ринку.

Таблиця 21. Альтернативи ринкового впровадження стартап-проєкту

№ п/п	*Альтернатива ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Збільшення ринку збуту завдяки маркетингу	Висока ймовірність	1-2 роки
2	Демпінг ціни	Висока ймовірність, оскільки є можливість переманити потенційних покупців у конкурентів	6 місяців
3	Вихід на міжнародний ринок	Середня ймовірність	2-3 роки

Серед можливих альтернатив більш надійним є варіант демпінгу ціни, так як є можливість переманити потенційних покупців у конкурентів.

3.4. Розроблення ринкової стратегії та маркетингової стратегії проєкту

Розробимо ринкову стратегію, що передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 22).

Таблиця 22. Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Державні підприємства	Середня	Низький	Низька	Середня
2	Приватні підприємства	Середня	Середній	Середня	Середня
3	Індивідуальні користувачі	Висока	Високий	Висока	Середня
Обрані цільові групи: Приватні підприємства та Індивідуальні споживачі					

Формування базової стратегії розвитку для роботи в обраних сегментах ринку (табл. 23).

Таблиця 23. Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проєкту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
1	Демпінг ціни	Стратегія диференційованог	Ціна Якість	Стратегія диференціації

		о маркетингу	Зручність налаштування Надійність	

У якості базової стратегії розвитку було прийнято рішення використовувати стратегію диференціації. У якості стратегії охоплення ринку – стратегію диференційованого маркетингу. Визначення базової стратегії конкурентної поведінки описано в табл. 24.

Таблиця 24. Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проєкт «першопроходцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
1	Проєкт не є першопроходцем	Буде шукати нових споживачів, адже користувачі у конкурентів не будуть замінювати вже працюючі системи.	Деякі функції є основними у будь-яких систем моніторингу мікроклімату та безпеки	Стратегія заняття конкурентної ніші.

Оскільки продукт не є першим на ринку було прийнято рішення притримуватись стратегії зайняття конкурентної ніші. Так як продукт має вигіднішу ціну ця стратегія є оптимальною. Базуючись на вимогах споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див.

табл. 12), а також в залежності від обраної базової стратегії розвитку (табл. 23) та стратегії конкурентної поведінки (табл. 24) розробляється стратегія позиціонування (табл. 25) [17].

Таблиця 25. Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проєкту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)*
1	Простота налаштування	Стратегія диференціації	Самостійне налаштування системи	Швидке та просте налаштування системи
2	Надійність системи	Стратегія диференціації	Якісний та надійний продукт	Якість, гарантія надійності.
3	Ціна	Стратегія диференціації	Доступна цінова політика	Низька ціна

Виконання підрозділу дало можливість визначити стратегію позиціювання стартап-проєкту на ринку, вибір асоціацій, які мають сформувати комплексну позицію власного проєкту.

Таблиця 26. Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
1.	Простота використання	Зменшення часу для ознайомлення з роботою системи	Забезпечується за допомогою якісного та продуманого програмного забезпечення
2.	Невисока ціна	Якість	Якісний продукт за невеликою ціною

Виділено дві основні переваги перед конкурентами, на основі яких буде створено рекламні продукти для поширення серед потенційних покупців.

Опис трьох рівнів моделі товару описано в табл. 27.

Таблиця 27. Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
I. Товар за задумом	Система моніторингу мікроклімату та безпеки будинку. Спрощене налаштування, надійність, невисока ціна системи.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	1. Економічні	Нм	Вр
	2. Призначення	Нм	Тх
	3. Надійність	М	Тх
	4. Технологічні	М	Тх
	5. Безпеки	М	Тх
	Якість: Сертифікати відповідності ДСТУ		
	Пакування: відповідно до стандартів		
	Марка: «Smart Security»		
III. Товар із підкріпленням	До продажу: покращення якості системи		
	Після продажу: розробка нових компонентів системи		
Захист від копіювання: патент			

Продукт повинен мати патент для захисту від копіювання. Важливо відмітити відсутність прямого доступу до основного програмного коду за рахунок надання його як сервісу, що унеможливлює копіювання

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (табл. 28).

Таблиця 28. Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення*

				ціни на товар/послугу
	3000-5000 грн	5 000-10 000 грн	10000-15000 грн	2200-3500 грн

За результатами визначення меж встановлення ціни було визначено, що оптимальна ціна є від 2 200 грн до 3 500 грн, що робить продукт привабливим серед конкурентів, які пропонують системи починаючи від 5 000 грн.

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 29)

Таблиця 29. Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1	Продаж	Встановлення контакту з покупцем. Продаж системи замовнику.	Канал нульового рівня	Пряма

Основним каналом збуту є продаж систем. Системою збуту – пряма із каналом нульового рівня, тобто без залучення посередників між виробником та споживачем.

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 30) [17].

Таблиця 30. Концепція маркетингових комунікацій

№ п/п	
Специфіка поведінки цільових клієнтів	Необхідність у моніторингу мікроклімату та безпеки будинку

Канали комунікацій, якими користуються цільові клієнти	Інтернет, виставки, конференції, рекламні статті
Ключові позиції, обрані для позиціонування	Моніторинг мікроклімату. Безпека будинку.
Завдання рекламного повідомлення	Формування впізнаваності марки. Демонстрування переваг над конкурентами
Концепція рекламного звернення	Простота використання та налаштування, надійність, швидкість.

Розглянемо організаційний план стартап-проєкту. Опис команди стартапу представлено в таблиці 31.

Таблиця 31. Початкові вкладення у стартап проєкт

Стадія стартапу	*Завдання*	*Члени команди*	*Освіта*	*Досвід роботи*	*Спеціалізовані знання*	*Витрати, грн*
Передпосівна	Розробка ідеї без розробленого механізму її реалізації	Олійник Владислав;	Інженер-програміст;	4 роки	Знання у сфері розробки ПО ;	-
Посівна	Дослідити ринок, розробити початкову дорожню карту проєкту, провести пошук інвесторів	-//-	-//-	-//-	-//-	10000
Прототипування	Створення архітектури проєкту;	Олійник Владислав; Шевченко Дмитро	Інженер-програміст; Програміст	4 роки; 7 років	Знання у сфері розробки ПО;	50000

	Створення програмного забезпечення				Знання у сфері архітектури та розробки ПО	
Закрита бета-версія	Створити готовий, життєздатний продукти. Протестувати продукт на невеликій закритій групі споживачів	Костенко Анастасія	Маркетолог	2 роки	Знання у сфері маркетингу	30000
Ведення бізнесу	Вивести продукт на широку аудиторію, почати продажі. Отримати відгуки щодо продукту, можливі незначні доопрацювання	-//-	-//-	-//-	-//-	250000

Отже, загальному команда складається з 4 людей:

1. Олійник Владислав, засновник, розробник програмного забезпечення;
2. Шевченко Дмитро, інженер-технолог, архітектор програмного забезпечення, займається проєктуванням та прийняттям ключових рішень щодо внутрішнього устрою програмної системи;

3. Костенко Анастасія, маркетолог, займається маркетинговою діяльністю та рекламою.

Складемо календарний план-графік підготовки стартапу (табл\ 32).

Таблиця 32. Календарний план-графік підготовки стартапу

Стадія стартапу	*Період запуску (за місяцями з початку підготовки проєкту)*					*Вартість стадії, грн*
	1	2	3	4	5	
Передпосівна	01.01.22 01.02.22					-
Посівна		01.02.22 01.03.22				10000
Прототипування			01.03.22 01.06.22			50000
Закрита бета-версія				01.06.22 01.08.22		30000
Ведення бізнесу					01.08.22 01.01.23	250000
Разом						340000

Отже, з календарного плану-графіку підготоввки стартапу (табл. 25) можна зробити висновок, що на реалізацію проєкту знадобиться 1 рік та 340 тис. грн. Найдорожчим етапом є введення бізнесу через його довготривалість.

Розглянемо виробничий план стартап-проєкту. Сировина, матеріали та комплектуючі наведені в таблиці 26:

Таблиця 33. Матеріали та їх постачальники

Найменування видів сировини, матеріалів та комплектуючих виробів	*Постачальники*	*Ціна за одиницю, грн*	*Кількість*
1. Плата з мікропроцесором ESP32	Espressif Systems	250	3
2. Датчик температури та вологості DHT21	AOSONG	150	1

3. Датчик дверей MC-38	SYNACORP	30	2
4. Пластиковий корпус	Епіцентр	50	3
Разом		1110	

Отже, з таблиці 33 можна зробити висновок, що для виготовлення однієї одиниці продукту необхідно 1110 грн.

Потреба в промислово-виробничому персоналі розлянута в таблиці 34.

Таблиця 34. Потреба в промислово-виробничому персоналі

Посада	*Чисельність*	*Витрати, тис. грн*
Інженер-програміст	1	20
Архітектор програмного забезпечення	1	25
Маркетолог	1	18
Працівник з досвідом роботи складання електроніки	3	15
Разом	6	108

З таблиці 34 можемо зробити висновок, що необхідно 6 людей промислово-виробничого персоналу та загальні витрати в кількості 108 тис. грн. на місяць для надання заробітної плати для персоналу.

Розглянемо зведений план витрат на запуск виробництва продукції (табл. 29).

Таблиця 35. Зведений план витрат на запуск виробництва продукції

Найменування	*Пояснення*	*Вартість тис. грн.*
Витрати на придбання обладнання та устаткування	Витрати на придбання обладнання	50
Сировина, основні матеріали	Вартість сировини та матеріалів для забезпечення технологічного процесу	30
Комплектуючі	Витрати на комплектуючі продукту	150

Паливо та електроенергія на технологічні цілі	Витрати на електроенергію, а також на паливо, необхідні для запуску проєктної потужності виробництва	15
Оплата праці промислово-виробничого персоналу	Витрати на заробітну плату та соціальні відрахування	108
Освоєння та запуск виробництва	Витрати на пусконалагоджувальні роботи, запуск виробництва	50
Разом:		403

Отже, зведений план витрат на запуск виробництва продукції показує, що для реалізації та випуску продукції необхідно 403 тис. грн.

Розглянемо фінансову модель стартапу. Визначимо початкові вкладення для розвитку стартап-проєкту (табл. 36).

Таблиця 36. Початкові вкладення у стартап проєкт

Види витрат	*Вартість*
НДДКР	20000
Захист прав на об'єкти інтелектуальної власності	10000
Закупівля сировини та матеріалів	180000
Створення прототипу, досліди	150000
Оренда приміщення	354000
Просування	150000
Витрати на команду	1296000
Замовлення послуг, менторство	20000
Закупівля обладнання	50000
Створення сайту стартапу	25000
Орієнтована собівартість першого продукту	1110
РАЗОМ	2266110
Витрати, що бере на себе стартапер	120000
Необхідні інвестиції для запуску стартапу та виробництва першої партії	2146110

За результатами таблиці 36 можна зробимо висновок, що інвестиції необхідні для старту проєкту складають 568110 грн, а витрати, що бере на себе стартапер складають 120000 грн.

Планові фінансово-економічні показники проєкту наведені в таблиці 37.

Таблиця 37. Планові фінансово-економічні показники

	Показник	**Періоди (по місяцях)**												**Всього за рік**
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	
1	Обсяг виробництва продукції в натуральних показниках	10	20	30	40	60	80	90	100	150	200	200	300	1280
2	Собівартість одиниці продукції, тис. грн.	1,1	1,1	1,1	1	1	1	1	0,9	0,9	0,9	0,9	0,9	11,8
3	Собівартість виробництва продукції, тис. грн. *(3 = 1 · 2)*	11	22	33	40	60	80	90	90	135	180	180	270	1191
4	Обсяг реалізації продукції в натуральних показниках	5	10	23	25	45	60	70	90	145	150	180	230	1033
5	Ціна реалізації продукції без ПДВ, тис. грн.	2,5	2,5	2,5	2,5	2,5	2,5	2,5	2,5	2,5	3	3	3	31,5
6	Виручка від реалізації продукції без	12,5	25	57,5	62,5	112,5	150	175	225	362,2	450	540	690	2866

	ПДВ, тис. грн. *(6 = 4 · 5)*												2,5
7	Податок на додану вартість (ПДВ), тис. грн. (7=6/5)	5	10	23	25	45	60	70	90	145	150	180	230	1033
8	Валовий прибуток *(8 = 6 – 3)*	1,5	3	24,5	22,5	52,2	70	85	135	227,5	270	360	420	1671,5
9	Податок на прибуток (9=8*0,20 (20 %))	0,3	0,6	4,9	4,5	10,5	14	17	27	45,5	54	72	84	334,3
10	Чистий прибуток *(10 = 8 – 9)*	1,2	2,4	19,6	18	42	56	68	108	182	216	288	336	1337,2

Згідно даних таблиці 37, можна зробити висновок, що при виробництві 1280 одиниць продукту на рік виручка від реалізації продукції (без ПДВ) складе 2 862 500 грн. Чистий прибуток складе 1 337 тис. грн.

Окупність інвестицій:

$$ROI = \frac{ANP}{I} \cdot 100\% = \frac{1337,2}{2146,11} \cdot 100\% = 62,3\%$$

3.5. Висновок

Результатом роботи над стартап-проєктом є сформована ідея реалізації системи моніторингу мікроклімату та безпеки будинку. Проведений технологічний аудит ідеї показав, що всі технології для реалізації наявні та доступні.

Проаналізувавши ринкові можливості можна сказати, що є можливість ринкової комерціалізації проєкту так як ринок зростає та має високий процент рентабельності.

Аналіз конкуренції показав що стартап демонструє високу конкурентоспроможність незважаючи на високий рівень конкуренції. Приваблива ціна та швидкість налаштування системи є ключовими факторами конкурентоспроможності.

Оскільки продукт не є «першопроходцем» на ринку було прийнято рішення притримуватись стратегії зайняття конкурентної ніші. Так як продукт має вигіднішу ціну ця стратегія є оптимальною. Також компанія буде шукати нових користувачі, адже користувачі, що обслуговуються у конкурентів не будуть змінювати вже працюючі системи, бо це дорого.

Серед можливих альтернатив ринкового впровадження більш підходящим є варіант демпінгу ціни, тобто продажу системи по заниженим цінам, для входу та закріплення на ринку.

Для початку роботи над стартапом потрібна команда. Було визначено, що потрібно 3 спеціалісти, саме архітектор програмного забезпечення, розробник програмного забезпечення та спеціаліст по маркетинговій діяльності. Маркетингова діяльність дуже важлива для даного проєкту так як на ринку існує висока конкуренція тому для успішного старту необхідно провести ряд маркетингових заходів для збільшення впізнаваності торгової марки для потенційних клієнтів.

Аналіз фінансової моделі стартапу показав, що на його реалізацію потрібно близько 2,3 млн. грн., а також один рік часу. При цьому чистий прибуток складе близько 1,3 млн.

Спираючись на аналіз проведений у цьому розділі можна впевнено сказати, що подальша імплементація проєкту є доцільною.

ЗАГАЛЬНІ ВИСНОВКИ

У даній роботі було проведено огляд та аналіз систем безпеки будинку, визначено головні параметри мікроклімату та розроблено робочий прототип інтелектуальної системи моніторингу мікроклімату та безпеки будинку.

Використання хмарних технологій дозволило зменшити витрати на налаштування та обслуговування системи, так як усі складові цієї системи, крім датчиків, винесено у хмарні сервіси.

У процесі розробки проєкту було визначено модель реалізації (SaaS) програмного забезпечення, виконано підбір датчиків та мікроконтролерів для реалізації прототипу. У якості мікроконтролера вибрано ESP32. Датчик температури та вологості – DHT21 та датчики дверей – MC-38.

Було запропоновано та реалізовано децентралізовану архітектуру взаємодії компонентів у системі.

Проаналізовано протоколи передачі даних та, опираючись на дані тестування швидкості та енергоефективності, вибрано оптимальний протокол – MQTT.

Розроблено алгоритми роботи компонентів системи та програмне забезпечення для датчика температури та вологості, датчиків дверей, Telegram бота та backend частини проєкту. Створено схему бази даних.

Розгорнуто програмне забезпечення на хмарних сервісах. Telegram бот та backend частину системи розгорнуто на хмарному сервісі Heroku. Налаштовано базу даних Mongo DB у хмарі та вибрано оптимальний хмарний MQTT брокер.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Микроклимат в доме [Електронний ресурс] – Режим доступу до ресурсу: https://www.promstok.com/articles/raznoe/mikroklimat-v-dome-osnovnye-parametry-i-sposoby-kontrolya/ (дата звернення: 5.12.2021)

2. What Is A Home Security System and How Does It Work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.security.org/home-security-systems/what-is-a-home-security-system/ (дата звернення: 5.12.2021)

3. MotionProtect [Електронний ресурс] – Режим доступу до ресурсу: https://ajax.systems/ru-ua/products/motionprotect/ (дата звернення: 5.12.2021)

4. DoorProtect [Електронний ресурс] – Режим доступу до ресурсу: https://ajax.systems/ru-ua/products/doorprotect/ (дата звернення: 5.12.2021)

5. GlassProtect [Електронний ресурс] – Режим доступу до ресурсу: https://ajax.systems/ru-ua/products/glassprotect/ (дата звернення: 5.12.2021)

6. ESP32 [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ESP32 (дата звернення: 5.12.2021)

7. ESP32 [Електронний ресурс] – Режим доступу до ресурсу: http://developer.alexanderklimov.ru/arduino/esp32/img/esp32-7.png (дата звернення: 5.12.2021)

8. Getting Started with the ESP32 Development Board [Електронний ресурс] – Режим доступу до ресурсу: https://randomnerdtutorials.com/getting-started-with-esp32/ (дата звернення: 5.12.2021)

9. Терморезистор [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/ Терморезистор (дата звернення: 5.12.2021)

10. Datasheet MC-38 [Електронний ресурс] – Режим доступу до ресурсу: https://blogmasterwalkershop.com.br/arquivos/datasheet/Datasheet%20MC-38.pdf (дата звернення: 5.12.2021)

11. HTTP vs MQTT performance tests [Електронний ресурс] – Режим доступу до ресурсу: https://flespi.com/blog/http-vs-mqtt-performance-tests (дата звернення: 5.12.2021)

12. Introducing the MQTT Protocol [Електронний ресурс] – Режим доступу до ресурсу: https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/ (дата звернення: 5.12.2021)

13. Что такое MQTT и для чего он нужен в IIoT? Описание протокола MQTT [Електронний ресурс] – Режим доступу до ресурсу: https://ipc2u.ru/articles/prostye-resheniya/chto-takoe-mqtt/ (дата звернення: 5.12.2021)

14. MongoDB - Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/mongodb/mongodb_overview.htm (дата звернення: 5.12.2021)

15. MongoDB - Advantages [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/mongodb/mongodb_advantages.htm (дата звернення: 5.12.2021)

16. Олійник В. С. Система моніторингу будинку з використанням хмарних

технологій // Ефективність та автоматизація інженерних рішень у приладобудуванні. – 2021. – С. 167-170.

17. Розроблення стартап-проєкту [Електронний ресурс]: Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.

Додаток А

Backend

Лістинг файлу models.py

```
import datetime
from utils.constants import MONGO_DB_USER, MONGO_DB_PASS

from mongoengine import *

MONGO_DB_CONNECTION_STRING = f'mongodb+srv://' \
                             f'{MONGO_DB_USER}:{MONGO_DB_PASS}' \

f'@cluster.vkvzs.mongodb.net/SmartHome?retryWrites=true&w=majority'
connect(host=MONGO_DB_CONNECTION_STRING)


class User(Document):
    tg_id = IntField(unique=True)
    tg_first_name = StringField()
    tg_last_name = StringField(required=False)
    tg_username = StringField(required=False)
    date_created = DateTimeField()
    date_modified = DateTimeField()

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        self.date_modified = datetime.datetime.now()
        return super().save(*args, **kwargs)

    @property
    def tg_full_name(self):
        return f"{self.tg_first_name} {self.tg_last_name}" if self.tg_last_name else
self.tg_first_name


class DHT(Document):
    name = StringField()
    user = ReferenceField(User)
    date_created = DateTimeField()
    date_modified = DateTimeField()

    meta = {
        'ordering': ['-date_created']
    }

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        self.date_modified = datetime.datetime.now()
        return super().save(*args, **kwargs)


class TemperatureValue(Document):
    sensor = ReferenceField(DHT)
    value = DecimalField(precision=2)
    date_created = DateTimeField()
```

```
    meta = {
        'ordering': ['-date_created']
    }

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        return super().save(*args, **kwargs)


class HumidityValue(Document):
    sensor = ReferenceField(DHT)
    value = DecimalField(precision=2)
    date_created = DateTimeField()

    meta = {
        'ordering': ['-date_created']
    }

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        return super().save(*args, **kwargs)


class DoorSensor(Document):
    name = StringField()
    user = ReferenceField(User)
    value = IntField()
    date_created = DateTimeField()
    date_modified = DateTimeField()

    meta = {
        'ordering': ['-date_created']
    }

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        self.date_modified = datetime.datetime.now()
        return super().save(*args, **kwargs)
```

Лістинг файлу utils/constants.py

```
from dotenv import load_dotenv
import os

load_dotenv()

TELEGRAM_BOT_TOKEN = os.getenv('TELEGRAM_BOT_TOKEN')

MONGO_DB_USER = os.getenv('MONGO_DB_USER')
MONGO_DB_PASS = os.getenv('MONGO_DB_PASS')

MQTT_SERVER = os.getenv('MQTT_SERVER')
MQTT_PORT = int(os.getenv('MQTT_PORT'))
MQTT_USER = os.getenv('MQTT_USER')
MQTT_PASSWORD = os.getenv('MQTT_PASSWORD')
```

```
DHT_TOPIC = 'dht'
DOOR_TOPIC = 'door'
```

Лістинг файлу utils/misc.py

```
import requests


class TelegramBot:
    def __init__(self, token):
        self.token = token
        self.url = "https://api.telegram.org/bot" + self.token

    def send_message(self, chat_id, text, parse_mode=None,
disable_notification=False, reply_markup=None):
        params = {
            "chat_id": chat_id,
            "text": text,
            "parse_mode": parse_mode,
            "disable_notification": disable_notification,
            "reply_markup": reply_markup
        }
        return requests.get(self.url + "/sendMessage", params=params)
```

Лістинг файлу main.py

```
import paho.mqtt.client as mqtt
import json

from models import DHT, TemperatureValue, HumidityValue, DoorSensor
from utils.constants import DHT_TOPIC, DOOR_TOPIC, TELEGRAM_BOT_TOKEN, MQTT_USER,
MQTT_PASSWORD, MQTT_SERVER, MQTT_PORT
from utils.misc import TelegramBot

tg_bot = TelegramBot(token=TELEGRAM_BOT_TOKEN)


def on_connect(client, obj, flags, rc):
    print("rc: " + str(rc))


def on_message(client, obj, msg):
    # print(msg.topic + " " + str(msg.qos) + " " + str(msg.payload))
    payload = json.loads(msg.payload.decode())
    print(payload)

    if msg.topic == DHT_TOPIC:
        dht = DHT.objects.get(id=payload['sensor_id'])
        temp = TemperatureValue(sensor=dht, value=float(payload['temperature']))
        hum = HumidityValue(sensor=dht, value=float(payload['humidity']))
        temp.save()
        hum.save()
    elif msg.topic == DOOR_TOPIC:
        new_value = int(payload['value'])
        door = DoorSensor.objects.get(id=payload['sensor_id'])
        if door:
            if door.value != new_value:
                door.value = new_value
                door.save()

                if bool(new_value):
```

```
                    tg_bot.send_message(chat_id=door.user.tg_id,
text=f'{door.name}\n'
                                                                    f'Двері
відкрились')
                else:
                    tg_bot.send_message(chat_id=door.user.tg_id,
text=f'{door.name}\n'
                                                                    f'Двері
зачинились')
        else:
            tg_bot.send_message(chat_id=door.user.tg_id, text='Ви не додали датчик
дверей')


def on_publish(client, obj, mid):
    print("mid: " + str(mid))


def on_subscribe(client, obj, mid, granted_qos):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))


def on_log(client, obj, level, string):
    print(string)


mqtt_cl = mqtt.Client()
mqtt_cl.username_pw_set(MQTT_USER, MQTT_PASSWORD)
mqtt_cl.on_message = on_message
mqtt_cl.on_connect = on_connect
mqtt_cl.on_publish = on_publish
mqtt_cl.on_subscribe = on_subscribe

mqtt_cl.connect(MQTT_SERVER, MQTT_PORT, 60)

mqtt_cl.subscribe(DHT_TOPIC)
mqtt_cl.subscribe(DOOR_TOPIC)

mqtt_cl.loop_forever()
```

Додаток Б

DHT21 Sensor

Лістинг файлу main.py

```
import dht
import time
import machine
import ubinascii
import json

from mqtt import MQTTClient

DHT21 = 5

f = open('config.json')
config = json.load(f)

SENSOR_ID = config['sensor_id']
DHT_TOPIC = config['dht_topic'].encode()

# WiFi
SSID = config['wifi']['ssid']
PASSWORD = config['wifi']['password']

# MQTT
MQTT_CLIENT_ID = ubinascii.hexlify(machine.unique_id())
MQTT_SERVER = config['mqtt']['server']
MQTT_PORT = config['mqtt']['port']
MQTT_USER = config['mqtt']['user']
MQTT_PASSWORD = config['mqtt']['password']
MQTT_SSL = True

f.close()


def connect_mqtt():
    client = MQTTClient(client_id=MQTT_CLIENT_ID,
                        server=MQTT_SERVER,
                        port=MQTT_PORT,
                        user=MQTT_USER,
                        password=MQTT_PASSWORD,
                        ssl=MQTT_SSL)
    client.connect()
    print("MQTT Client ID" + MQTT_CLIENT_ID.decode())
    print('Connected to %s MQTT broker' % MQTT_SERVER)
    return client


def restart_and_reconnect_mqtt():
    print('Failed to connect to MQTT broker. Reconnecting...')
    time.sleep(10)
    machine.reset()


def do_wifi_connect():
    import network
    sta_if = network.WLAN(network.STA_IF)
    while not sta_if.isconnected():
```

```
            print('Connecting to network...')
            sta_if.active(True)
            sta_if.connect(SSID, PASSWORD)
            if not sta_if.isconnected():
                print('Connection error. Trying to reconnect in 5 sec')
                time.sleep(5)

        print('Connected!')
        print('Network config:', sta_if.ifconfig())


    if __name__ == '__main__':
        dht21 = dht.DHT22(machine.Pin(DHT21))
        do_wifi_connect()

        mqtt_client = None

        try:
            mqtt_client = connect_mqtt()
        except OSError as e:
            restart_and_reconnect_mqtt()

        while True:
            dht21.measure()

            data = {
                'sensor_id': SENSOR_ID,
                'temperature': '{0:3.1f}'.format(dht21.temperature()),
                'humidity': '{0:3.1f}'.format(dht21.humidity())
            }

            if mqtt_client:
                mqtt_client.publish(DHT_TOPIC, json.dumps(data).encode())

            time.sleep(30)
```

Додаток В

Door Sensor

Лістинг файлу main.py

```
import time
import machine
import ubinascii
import json

from mqtt import MQTTClient

DOOR_PIN = 5

f = open('config.json')
config = json.load(f)

SENSOR_ID = config['sensor_id']
DOOR_TOPIC = config['door_topic'].encode()

# WiFi
SSID = config['wifi']['ssid']
PASSWORD = config['wifi']['password']

# MQTT
MQTT_CLIENT_ID = ubinascii.hexlify(machine.unique_id())
MQTT_SERVER = config['mqtt']['server']
MQTT_PORT = config['mqtt']['port']
MQTT_USER = config['mqtt']['user']
MQTT_PASSWORD = config['mqtt']['password']
MQTT_SSL = True

f.close()


def connect_mqtt():
    client = MQTTClient(client_id=MQTT_CLIENT_ID,
                        server=MQTT_SERVER,
                        port=MQTT_PORT,
                        user=MQTT_USER,
                        password=MQTT_PASSWORD,
                        ssl=MQTT_SSL)
    client.connect()
    print("MQTT Client ID" + MQTT_CLIENT_ID.decode())
    print('Connected to %s MQTT broker' % MQTT_SERVER)
    return client


def restart_and_reconnect_mqtt():
    print('Failed to connect to MQTT broker. Reconnecting...')
    time.sleep(10)
    machine.reset()


def do_wifi_connect():
    import network
    sta_if = network.WLAN(network.STA_IF)
    while not sta_if.isconnected():
        print('Connecting to network...')
```

```
        sta_if.active(True)
        sta_if.connect(SSID, PASSWORD)
        if not sta_if.isconnected():
            print('Connection error. Trying to reconnect in 5 sec')
            time.sleep(5)


if __name__ == '__main__':
    door = machine.Pin(DOOR_PIN)
    do_wifi_connect()

    mqtt_client = None

    try:
        mqtt_client = connect_mqtt()
    except OSError as e:
        restart_and_reconnect_mqtt()

    while True:
        data = {
            'sensor_id': SENSOR_ID,
            'value': f'{door.value()}'
        }

        if mqtt_client:
            mqtt_client.publish(DOOR_TOPIC, json.dumps(data).encode())

        time.sleep(1)
```

Додаток Г

MQTT

Лістинг файлу mqtt.py

```
try:
    import usocket as socket
except:
    import socket
import ustruct as struct
from ubinascii import hexlify


class MQTTException(Exception):
    pass


class MQTTClient:

    def __init__(self, client_id, server, port=0, user=None, password=None,
keepalive=0,
                 ssl=False, ssl_params={}):
        if port == 0:
            port = 8883 if ssl else 1883
        self.client_id = client_id
        self.sock = None
        self.server = server
        self.port = port
        self.ssl = ssl
        self.ssl_params = ssl_params
        self.pid = 0
        self.cb = None
        self.user = user
        self.pswd = password
        self.keepalive = keepalive
        self.lw_topic = None
        self.lw_msg = None
        self.lw_qos = 0
        self.lw_retain = False

    def _send_str(self, s):
        self.sock.write(struct.pack("!H", len(s)))
        self.sock.write(s)

    def _recv_len(self):
        n = 0
        sh = 0
        while 1:
            b = self.sock.read(1)[0]
            n |= (b & 0x7f) << sh
            if not b & 0x80:
                return n
            sh += 7

    def set_callback(self, f):
        self.cb = f

    def set_last_will(self, topic, msg, retain=False, qos=0):
        assert 0 <= qos <= 2
```

```
        assert topic
        self.lw_topic = topic
        self.lw_msg = msg
        self.lw_qos = qos
        self.lw_retain = retain

    def connect(self, clean_session=True):
        self.sock = socket.socket()
        addr = socket.getaddrinfo(self.server, self.port)[0][-1]
        self.sock.connect(addr)
        if self.ssl:
            import ussl
            self.sock = ussl.wrap_socket(self.sock, **self.ssl_params)
        premsg = bytearray(b"\x10\0\0\0\0\0")
        msg = bytearray(b"\x04MQTT\x04\x02\0\0")

        sz = 10 + 2 + len(self.client_id)
        msg[6] = clean_session << 1
        if self.user is not None:
            sz += 2 + len(self.user) + 2 + len(self.pswd)
            msg[6] |= 0xC0
        if self.keepalive:
            assert self.keepalive < 65536
            msg[7] |= self.keepalive >> 8
            msg[8] |= self.keepalive & 0x00FF
        if self.lw_topic:
            sz += 2 + len(self.lw_topic) + 2 + len(self.lw_msg)
            msg[6] |= 0x4 | (self.lw_qos & 0x1) << 3 | (self.lw_qos & 0x2) << 3
            msg[6] |= self.lw_retain << 5

        i = 1
        while sz > 0x7f:
            premsg[i] = (sz & 0x7f) | 0x80
            sz >>= 7
            i += 1
        premsg[i] = sz

        self.sock.write(premsg, i + 2)
        self.sock.write(msg)
        # print(hex(len(msg)), hexlify(msg, ":"))
        self._send_str(self.client_id)
        if self.lw_topic:
            self._send_str(self.lw_topic)
            self._send_str(self.lw_msg)
        if self.user is not None:
            self._send_str(self.user)
            self._send_str(self.pswd)
        resp = self.sock.read(4)
        assert resp[0] == 0x20 and resp[1] == 0x02
        if resp[3] != 0:
            raise MQTTException(resp[3])
        return resp[2] & 1

    def disconnect(self):
        self.sock.write(b"\xe0\0")
        self.sock.close()

    def ping(self):
        self.sock.write(b"\xc0\0")

    def publish(self, topic, msg, retain=False, qos=0):
```

```
        pkt = bytearray(b"\x30\0\0\0")
        pkt[0] |= qos << 1 | retain
        sz = 2 + len(topic) + len(msg)
        if qos > 0:
            sz += 2
        assert sz < 2097152
        i = 1
        while sz > 0x7f:
            pkt[i] = (sz & 0x7f) | 0x80
            sz >>= 7
            i += 1
        pkt[i] = sz
        # print(hex(len(pkt)), hexlify(pkt, ":"))
        self.sock.write(pkt, i + 1)
        self._send_str(topic)
        if qos > 0:
            self.pid += 1
            pid = self.pid
            struct.pack_into("!H", pkt, 0, pid)
            self.sock.write(pkt, 2)
        self.sock.write(msg)
        if qos == 1:
            while 1:
                op = self.wait_msg()
                if op == 0x40:
                    sz = self.sock.read(1)
                    assert sz == b"\x02"
                    rcv_pid = self.sock.read(2)
                    rcv_pid = rcv_pid[0] << 8 | rcv_pid[1]
                    if pid == rcv_pid:
                        return
        elif qos == 2:
            assert 0

    def subscribe(self, topic, qos=0):
        assert self.cb is not None, "Subscribe callback is not set"
        pkt = bytearray(b"\x82\0\0\0")
        self.pid += 1
        struct.pack_into("!BH", pkt, 1, 2 + 2 + len(topic) + 1, self.pid)
        # print(hex(len(pkt)), hexlify(pkt, ":"))
        self.sock.write(pkt)
        self._send_str(topic)
        self.sock.write(qos.to_bytes(1, "little"))
        while 1:
            op = self.wait_msg()
            if op == 0x90:
                resp = self.sock.read(4)
                # print(resp)
                assert resp[1] == pkt[2] and resp[2] == pkt[3]
                if resp[3] == 0x80:
                    raise MQTTException(resp[3])
                return

    # Wait for a single incoming MQTT message and process it.
    # Subscribed messages are delivered to a callback previously
    # set by .set_callback() method. Other (internal) MQTT
    # messages processed internally.
    def wait_msg(self):
        res = self.sock.read(1)
        self.sock.setblocking(True)
        if res is None:
```

```
            return None
        if res == b"":
            raise OSError(-1)
        if res == b"\xd0":  # PINGRESP
            sz = self.sock.read(1)[0]
            assert sz == 0
            return None
        op = res[0]
        if op & 0xf0 != 0x30:
            return op
        sz = self._recv_len()
        topic_len = self.sock.read(2)
        topic_len = (topic_len[0] << 8) | topic_len[1]
        topic = self.sock.read(topic_len)
        sz -= topic_len + 2
        if op & 6:
            pid = self.sock.read(2)
            pid = pid[0] << 8 | pid[1]
            sz -= 2
        msg = self.sock.read(sz)
        self.cb(topic, msg)
        if op & 6 == 2:
            pkt = bytearray(b"\x40\x02\0\0")
            struct.pack_into("!H", pkt, 2, pid)
            self.sock.write(pkt)
        elif op & 6 == 4:
            assert 0

    # Checks whether a pending message from server is available.
    # If not, returns immediately with None. Otherwise, does
    # the same processing as wait_msg.
    def check_msg(self):
        self.sock.setblocking(False)
        return self.wait_msg()
```

Додаток Д

Telegram Bot

Лістинг файлу models.py

```
import datetime
from utils.constants import MONGO_DB_USER, MONGO_DB_PASS

from mongoengine import *

MONGO_DB_CONNECTION_STRING = f'mongodb+srv://' \
                             f'{MONGO_DB_USER}:{MONGO_DB_PASS}' \

f'@cluster.vkvzs.mongodb.net/SmartHome?retryWrites=true&w=majority'
connect(host=MONGO_DB_CONNECTION_STRING)


class User(Document):
    tg_id = IntField(unique=True)
    tg_first_name = StringField()
    tg_last_name = StringField(required=False)
    tg_username = StringField(required=False)
    date_created = DateTimeField()
    date_modified = DateTimeField()

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        self.date_modified = datetime.datetime.now()
        return super().save(*args, **kwargs)

    @property
    def tg_full_name(self):
        return f"{self.tg_first_name} {self.tg_last_name}" if self.tg_last_name else
self.tg_first_name


class DHT(Document):
    name = StringField()
    user = ReferenceField(User)
    date_created = DateTimeField()
    date_modified = DateTimeField()

    meta = {
        'ordering': ['-date_created']
    }

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        self.date_modified = datetime.datetime.now()
        return super().save(*args, **kwargs)


class TemperatureValue(Document):
    sensor = ReferenceField(DHT)
    value = DecimalField(precision=2)
    date_created = DateTimeField()
```

```
    meta = {
        'ordering': ['-date_created']
    }

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        return super().save(*args, **kwargs)


class HumidityValue(Document):
    sensor = ReferenceField(DHT)
    value = DecimalField(precision=2)
    date_created = DateTimeField()

    meta = {
        'ordering': ['-date_created']
    }

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        return super().save(*args, **kwargs)


class DoorSensor(Document):
    name = StringField()
    user = ReferenceField(User)
    value = IntField()
    date_created = DateTimeField()
    date_modified = DateTimeField()

    meta = {
        'ordering': ['-date_created']
    }

    def save(self, *args, **kwargs):
        if not self.date_created:
            self.date_created = datetime.datetime.now()
        self.date_modified = datetime.datetime.now()
        return super().save(*args, **kwargs)
```

Лістинг файлу utils/constants.py

```
from dotenv import load_dotenv
import os

load_dotenv()

API_TOKEN = os.environ.get('API_TOKEN')
PRODUCTION = os.getenv('PRODUCTION')

PROJECT_NAME = 'smart-home-system-bot'
WEBAPP_HOST = '0.0.0.0'
WEBAPP_PORT = os.getenv('PORT')

WEBHOOK_PATH = '/'
WEBHOOK_HOST = f'https://{PROJECT_NAME}.herokuapp.com'
WEBHOOK_URL = f'{WEBHOOK_HOST}{WEBHOOK_PATH}'
```

```
MONGO_DB_USER = os.getenv('MONGO_DB_USER')
MONGO_DB_PASS = os.getenv('MONGO_DB_PASS')

MQTT_SERVER = os.getenv('MQTT_SERVER')
MQTT_PORT = os.getenv('MQTT_PORT')
MQTT_USER = os.getenv('MQTT_USER')
MQTT_PASSWORD = os.getenv('MQTT_PASSWORD')
```

Лістинг файлу utils/filters.py

```
from aiogram import types
from aiogram.dispatcher.filters import BoundFilter

from models import User
from utils import keyborards


class RegisteredFilter(BoundFilter):
    key = 'is_registered'

    def __init__(self, is_registered):
        self.is_registered = is_registered

    async def check(self, message: types.Message):
        try:
            await message.answer_chat_action('typing')
        except AttributeError:
            pass
        user = User.objects(tg_id=message.from_user.id).first()
        if not bool(user):
            await message.answer(f"Ви не зареєстовані. Натисніть почати для
реєстрації",
                                 reply_markup=keyborards.start_keyboard)
        return bool(user)
```

Лістинг файлу utils/keyboards.py

```
from aiogram import types


def make_reply_row_keyboard(buttons):
    keyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
    for row in buttons:
        keyboard.row(*row)
    return keyboard


start_button_aliases = ['почати!', 'start']
temp_button_aliases = ['температура', 'temperature']
hum_button_aliases = ['вологість', 'humidity']
door_button_aliases = ['двері', 'door']
cancel_button_aliases = ['відмінити', 'cancel']
add_sensor_button_aliases = ['додати датчик', 'add sensor']

dht_button = "Датчик температури та вологи"
door_button = "Датчик дверей"
cancel_button = "Відмінити"
allowed_sensors = [dht_button.lower(), door_button.lower()]
```

```
start_row_buttons = [('Почати!',)]
main_row_buttons = [("Температура", "Вологість"), ("Двері",), ("Додати датчик",)]
sensors_row_buttons = [(dht_button,), (door_button,), (cancel_button,)]
cancel_row_buttons = [(cancel_button,)]

start_keyboard = make_reply_row_keyboard(start_row_buttons)
main_keyboard = make_reply_row_keyboard(main_row_buttons)
sensors_keyboard = make_reply_row_keyboard(sensors_row_buttons)
cancel_keyboard = make_reply_row_keyboard(cancel_row_buttons)
```

Лістинг файлу utils/misc.py

```
from aiogram.dispatcher.filters.state import StatesGroup, State

from models import User, DHT, DoorSensor
from utils import keyborards
from utils.constants import PRODUCTION


def is_production():
    return bool(int(PRODUCTION))


def create_or_update_user(tg_id, tg_first_name, tg_last_name, tg_username):
    user = User.objects(tg_id=tg_id).first()
    if not user:
        user = User(tg_id=tg_id, tg_first_name=tg_first_name,
tg_last_name=tg_last_name, tg_username=tg_username)
        user.save()
    return user


def add_sensor(data: dict, user: User):
    if data['type'] == keyborards.dht_button:
        dht = DHT(name=data['name'], user=user)
        dht.save()
        return dht

    if data['type'] == keyborards.door_button:
        door = DoorSensor(name=data['name'], user=user)
        door.save()
        return door


class SensorAddForm(StatesGroup):
    type = State()
    name = State()
```

Лістинг файлу handlers/add_sensor.py

```
import logging

from aiogram.dispatcher import FSMContext
from aiogram import types, Dispatcher

from utils.filters import RegisteredFilter
from models import User

from utils import keyborards
```

```
from utils.misc import SensorAddForm, add_sensor


async def add_sensor_handler(message: types.Message):
    await SensorAddForm.type.set()
    await message.answer("Оберіть тип датчика",
                         reply_markup=keyborards.sensors_keyboard,
                         parse_mode=types.ParseMode.HTML)


async def process_sensor_type(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['type'] = message.text

    await SensorAddForm.next()
    await message.answer("Вкажіть назву:\n"
                         "Для зручності можна вказати назву кімнати у якій
встановлено датчик",
                         reply_markup=keyborards.cancel_keyboard)


async def process_sensor_name(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['name'] = message.text
        user = User.objects.get(tg_id=message.from_user.id)
        sensor = add_sensor(data, user)
        await message.answer(f"{data['type']} ({data['name']}) додано!\n"
                             f"ID: {sensor.id}",
                             reply_markup=keyborards.main_keyboard,
                             parse_mode=types.ParseMode.HTML)
    await state.finish()


async def process_sensor_type_invalid(message: types.Message):
    return await message.reply("Невірний тип датчика. Виберіть тип із клавіатури.")


async def cancel_handler(message: types.Message, state: FSMContext):
    current_state = await state.get_state()
    if current_state is None:
        await message.answer('Вітаю!', reply_markup=keyborards.main_keyboard)
        return

    logging.info('Cancelling state %r', current_state)
    await state.finish()
    await message.answer('Відмінено', reply_markup=keyborards.main_keyboard)


def register_add_sensor_handlers(dp: Dispatcher):
    dp.register_message_handler(cancel_handler, commands='cancel', state='*')
    dp.register_message_handler(cancel_handler,
                                lambda message: message.text.lower() in
keyborards.cancel_button_aliases,
                                state='*')
    dp.register_message_handler(add_sensor_handler,
                                lambda message: message.text.lower() in
keyborards.add_sensor_button_aliases,
                                RegisteredFilter(is_registered=True))
    dp.register_message_handler(process_sensor_type,
                                RegisteredFilter(is_registered=True),
                                state=SensorAddForm.type)
```

```
    dp.register_message_handler(process_sensor_type_invalid,
                                lambda message: message.text.lower() not in
keyborards.allowed_sensors and
                                                message.text.lower() not in
keyborards.cancel_button_aliases,
                                RegisteredFilter(is_registered=True),
                                state=SensorAddForm.type)
    dp.register_message_handler(process_sensor_name,
                                RegisteredFilter(is_registered=True),
                                state=SensorAddForm.name)
```

Лістинг файлу handlers/main.py

```
from aiogram import Dispatcher
from aiogram.dispatcher.filters import CommandStart
from aiogram import types

from utils.filters import RegisteredFilter
from models import TemperatureValue, HumidityValue, DoorSensor, DHT, User

from utils import keyborards
from utils.misc import create_or_update_user


async def bot_start(message: types.Message):
    user = create_or_update_user(tg_id=message.from_user.id,
tg_first_name=message.from_user.first_name,
                                 tg_last_name=message.from_user.last_name,
tg_username=message.from_user.username)
    await message.answer(f"<b>Вітаємо, {user.tg_full_name}</b>",
                         reply_markup=keyborards.main_keyboard,
                         parse_mode=types.ParseMode.HTML)


async def get_temperature_handler(message: types.Message):
    user = User.objects.get(tg_id=message.from_user.id)
    sensors = DHT.objects(user=user)
    if not sensors:
        await message.answer("Датчиків не знайдено",
reply_markup=keyborards.main_keyboard)
        return
    for sensor in sensors:
        last_temp = TemperatureValue.objects(sensor=sensor).first()
        if last_temp:
            await message.answer(f"<b>{sensor.name}</b>\n"
                                 f"Тепература {last_temp.value} °C",
                                 reply_markup=keyborards.main_keyboard,
                                 parse_mode=types.ParseMode.HTML)
        else:
            await message.answer(f"<b>{sensor.name}</b>\n"
                                 f"Даних не знайдено\n"
                                 f"Sensor ID: {sensor.id}",
                                 reply_markup=keyborards.main_keyboard,
                                 parse_mode=types.ParseMode.HTML)


async def get_humidity_handler(message: types.Message):
    user = User.objects.get(tg_id=message.from_user.id)
    sensors = DHT.objects(user=user)
    if not sensors:
```

```
        await message.answer("Датчиків не знайдено",
reply_markup=keyborards.main_keyboard)
        return
    for sensor in sensors:
        last_hum = HumidityValue.objects(sensor=sensor).first()
        if last_hum:
            await message.answer(f"<b>{sensor.name}</b>\n"
                                 f"Вологість {last_hum.value}%",
                                 reply_markup=keyborards.main_keyboard,
                                 parse_mode=types.ParseMode.HTML)
        else:
            await message.answer(f"<b>{sensor.name}</b>\n"
                                 f"Даних не знайдено\n"
                                 f"Sensor ID: {sensor.id}",
                                 reply_markup=keyborards.main_keyboard,
                                 parse_mode=types.ParseMode.HTML)


async def get_door_status_handler(message: types.Message):
    user = User.objects.get(tg_id=message.from_user.id)
    sensors = DoorSensor.objects(user=user.id)
    if not sensors:
        await message.answer("Датчиків не знайдено",
reply_markup=keyborards.main_keyboard)
        return
    for sensor in sensors:
        if sensor.value is None:
            await message.answer(f"<b>{sensor.name}</b>\n"
                                 f"Даних не знайдено\n"
                                 f"Sensor ID: {sensor.id}",
                                 reply_markup=keyborards.main_keyboard,
                                 parse_mode=types.ParseMode.HTML)
        else:
            await message.answer(f"<b>{sensor.name}</b>\n"
                                 f"Двері {'відчинені' if sensor.value else
'зачинені'}",
                                 reply_markup=keyborards.main_keyboard,
                                 parse_mode=types.ParseMode.HTML)


def register_main_handlers(dp: Dispatcher):
    dp.register_message_handler(bot_start, CommandStart())
    dp.register_message_handler(bot_start, lambda message: message.text.lower() in
keyborards.start_button_aliases)

    dp.register_message_handler(get_temperature_handler,
                                lambda message: message.text.lower() in
keyborards.temp_button_aliases,
                                RegisteredFilter(is_registered=True))
    dp.register_message_handler(get_humidity_handler,
                                lambda message: message.text.lower() in
keyborards.hum_button_aliases,
                                RegisteredFilter(is_registered=True))
    dp.register_message_handler(get_door_status_handler,
                                lambda message: message.text.lower() in
keyborards.door_button_aliases,
                                RegisteredFilter(is_registered=True))
```

Лістинг файлу main.py

```
import logging

from aiogram import Bot
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.contrib.middlewares.logging import LoggingMiddleware
from aiogram.dispatcher import Dispatcher
from aiogram.utils import executor

from utils.constants import API_TOKEN, WEBHOOK_URL, WEBAPP_PORT, WEBHOOK_PATH,
WEBAPP_HOST
from utils.misc import is_production

logging.basicConfig(level=logging.INFO)

bot = Bot(token=API_TOKEN)
storage = MemoryStorage()
dp = Dispatcher(bot, storage=storage)
dp.middleware.setup(LoggingMiddleware())


def setup():
    from handlers.main import register_main_handlers
    from handlers.add_sensor import register_add_sensor_handlers
    register_main_handlers(dp)
    register_add_sensor_handlers(dp)


async def on_startup(dp):
    await bot.delete_webhook()
    await bot.set_webhook(WEBHOOK_URL)


async def on_shutdown(dp):
    logging.warning('Shutting down..')

    await dp.storage.close()
    await dp.storage.wait_closed()

    logging.warning('Bye!')


if __name__ == '__main__':
    setup()

    if is_production():
        executor.start_webhook(
            dispatcher=dp,
            webhook_path=WEBHOOK_PATH,
            on_startup=on_startup,
            on_shutdown=on_shutdown,
            skip_updates=True,
            host=WEBAPP_HOST,
            port=WEBAPP_PORT,
        )
    else:
        executor.start_polling(dp, skip_updates=True)
```