

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ТА
ЗАСОБІВ ХМАРНОГО КОМП'ЮТИНГУ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS AND
TOOLS FOR CLOUD COMPUTING

Виконав: студент(ка) 2 курсу, групи КНТ-211м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Терещенко О.В.

(прізвище та ініціали)

Керівник Пархоменко А. В.

(прізвище та ініціали)

Рецензент Зеленьова І. Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет комп'ютерних наук та технологій
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Терещенко Олени Віталіївни

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів та засобів хмарного комп'ютингу. Research and Software Implementation of Methods and Tools for Cloud Computing.

керівник проєкту (роботи) Пархоменко Анжеліка Володимирівна, к.т.н., доцент

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

затверджені наказом закладу вищої освіти від “4” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) 1 грудня 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Дослідження віртуального та віддаленого інструментарію для прототипування IoT систем. 3. Проєктування програмного забезпечення. 4. Конструювання програмного забезпечення.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Пархоменко А.В., доцент		
Нормоконтроль	Камінська Ж.К., асистент		

7. Дата видачі завдання “05” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Аналіз віртуального та віддаленого інструментарію для прототипування IoT систем	4-5тиждень	Розділ 2
4	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	6 тиждень	Розділ 3
5	Розробка архітектури програми	7 тиждень	Розділ 3
6	Розробка програми	8-9 тижні	Розділ 4
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	10-13 тиждень	Додатки
8	Нормоконтроль та рецензування.	14 тиждень	
9	Захист роботи	15 тиждень	

Студент(ка)

(підпис) Терещенко О. В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Пархоменко А.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
102 с., 85 рис., 3 табл., 2 дод., 77 джерел.

ХМАРНИЙ КОМП'ЮТИНГ, ІОТ СЕРВІС, ВІДЕОТРАНСЛЯЦІЯ,
ДАТЧИКИ, ВІЗУАЛІЗАЦІЯ ДАНИХ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження - процес застосування хмарного комп'ютингу у сфері IoT.

Предмет дослідження – моделі та засоби розгортання хмарних сервісів для IoT систем.

Мета роботи – дослідження методів та засобів хмарного комп'ютингу та розробка IoT-сервісу для отримання та візуалізації даних з датчиків, а також відеотрансляції для контролю стану IoT пристроїв.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування Python, фреймворк Flask, JavaScript, мова розмітки HTML, персональний комп'ютер з процесором Intel Core i5 під управлінням операційної системи під управлінням операційної системи Linux Mint Uma.

Результати. Створено вебзастосунок, який отримує, обробляє та візуалізує дані з IoT датчиків, забезпечує трансляцію та зберігання відео з камери, що наведена на ці датчики, з використанням хмарного FTP сервісу.

Висновки. Технології хмарного комп'ютингу надають широкі можливості обробки різноманітних даних IoT системи, що сприяє підвищенню наочності представлення інформації, а також ступеня контрольованості процесів функціонування компонентів таких систем.

Галузь використання – IoT системи, системи Розумний будинок, Розумне місто, Розумне підприємство тощо.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 102 pages, 85 figures, 3 tables, 2 appendixes, 77 sources.

CLOUD COMPUTING, IOT SERVICE, VIDEO BROADCAST, SENSORS, DATA VISUALIZATION, SOFTWARE.

Object of study is a process of using a cloud computing in the IoT sphere.

Subject of study are models and methods for deploying cloud services for IoT systems.

The purpose of the work is to study methods and means of a cloud computing and develop an IoT service for receiving and visualizing data from sensors, as well as video broadcasting to monitor the state of IoT devices.

Materials, methods, and tools: structured and object-oriented programming, programming language Python, Flask framework, JavaScript, HTML markup language, personal computer with Intel Core i5 processor running the Linux Mint Uma operating system.

Results. A web application has been created that receives, processes and visualizes data from IoT sensors, provides broadcast and storage of video from a camera pointed to these sensors using a cloud FTP service.

Conclusions. Cloud computing technologies provide ample opportunities for processing various data of the IoT system, which helps to increase the visibility of information presentation, as well as the degree of controllability of the functioning of the components of such systems.

The field of use is a IoT systems, Smart home systems, Smart city, Smart enterprise, etc.

ЗМІСТ

	С.
ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК	7
ВСТУП	8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	9
1.1 Хмарний комп'ютинг.....	9
1.2 Архітектура IoT	19
1.3 Хмарний комп'ютинг у сфері IoT	22
1.4 Постановка задач магістерської роботи.....	26
2 АНАЛІЗ ВІРТУАЛЬНОГО ТА ВІДДАЛЕНОГО ІНСТРУМЕНТАРІЮ ДЛЯ ПРОТОТИПУВАННЯ ІОТ СИСТЕМ.....	28
2.1 Віртуальний робочий простір	28
2.2 Віддалений робочий простір REIoT	49
2.3 Висновки до розділу 2	53
3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	54
3.1 Аналіз вимог	54
3.2 Розробка архітектури системи	55
3.3 Удосконалений метод роботи з відеопотоком	56
3.4 Вибір інструментарію розробки	58
3.5 Висновки до розділу 3	65
4 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ	66
4.1 Алгоритм трансляції та зберігання відео.....	66
4.2 Візуалізація даних у вигляді графіків	71
4.3 Висновки до розділу 4	74
ВИСНОВКИ.....	76
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	77
ДОДАТОК А Текст програми.....	84
ДОДАТОК Б Слайди презентації	95

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
FTP	– File Transfer Protocol;
HTML	- HyperText Markup Language
IaaS	– Infrastructure as a Service;
IDE	– Integrated Development Environment;
IoT	– Internet of Things;
PaaS	– Platform as a Service;
RELDES	– Remote Laboratory for Design of Embedded Systems;
RTSP	– Real Time Streaming Protocol;
SaaS	– Software as a Service;
ВЛ	– віддалена лабораторія;
ПЗ	– програмне забезпечення.

ВСТУП

На сьогоднішній день хмарний комп'ютинг набуває дуже великої популярності. Він проник майже в усі сфери нашого життя. Можна сказати, що метою хмарного комп'ютингу є забезпечення віртуального доступу до великих об'ємів обчислювальної потужності та запропонувати єдиний системний за стосунок. Великий спектр можливостей, доступ до послуг з будь-якого пристрою, підключеного до мережі Інтернет, та плата за послуги у міру використання робить хмарний комп'ютинг все більш популярним.

Сьогодні хмарний комп'ютинг активно проникає в сферу IoT. Більшість сучасних екосистем IoT засновані саме на хмарі. Хмарний комп'ютинг допомагає максимально ефективно використовувати дані, що генеруються датчиками IoT, дозволяючи підключатись до пристроїв з будь-якого місця в будь-який час, та значно скорочує витрати пов'язані зі створенням IoT екосистем. Хмарні IoT сервіси дозволяють зберігати дані IoT та пропонують інструменти для створення IoT застосунків.

Як показали проведені дослідження, існуючі IoT сервіси відрізняються функціоналом, засобами відображення та обробки даних та умовами застосування. Але вони не надають можливості транслявати відео на сторінку IoT сервісу для забезпечення контрольних функцій. Тому, тема роботи є актуальною.

Метою роботи є дослідження методів та засобів хмарного комп'ютингу та розробка IoT сервісу для отримання та візуалізації даних з датчиків, а також відеотрансляції для контролю стану IoT пристроїв.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Хмарний комп'ютинг

На теперішній час технологія хмарного комп'ютингу набуває дуже великої популярності. Це пов'язано з швидко зростаючим числом смартфонів і мобільних пристроїв, які можуть мати доступ до Інтернету. Хмарні обчислення призначені не тільки для організацій та підприємств, а і для звичайних користувачів [1].

Простіше кажучи, хмарний комп'ютинг — це надання обчислювальних послуг, включаючи сервери, сховища, бази даних, мережі, програмне забезпечення та аналітику. Вони пропонують швидші інновації, гнучкі ресурси та економію за рахунок масштабу. Зазвичай користувачі платять лише за використовувані хмарні служби, що допомагає знизити експлуатаційні витрати, ефективніше керувати інфраструктурою та масштабувати у міру зміни потреб [2].

Сама концепція хмарного комп'ютингу виникла в 1950 році разом із впровадженням мейнфреймів, доступних через тонкі/статичні клієнти. З того часу хмарні обчислення пройшли шлях від статичних клієнтів до динамічних та від програмного забезпечення до послуг. На рисунку 1.1 зображено етапи еволюції хмарного комп'ютингу [3].

Хмарний комп'ютинг має п'ять основних характеристик такі, як самообслуговування на вимогу, широкий доступ до мережі, об'єднання ресурсів, швидка еластичність, та вимірювані послуги.

Самообслуговування на вимогу дозволяє клієнтам використовувати хмарні обчислення за потреби без людського контакту між споживачами та постачальниками послуг. Використовуючи функції самообслуговування на вимогу, споживачі можуть організовувати різноманітні хмарні ресурси за потреби. Крім того, що система самообслуговування має бути безпечною та уважною до клієнта, вона має бути зручною для користувача, щоб мати

доступ до різноманітних хмарних ресурсів та ефективно відстежувати пропозиції послуг [4].

Широкий доступ до мережі — це здатність мережевої інфраструктури підключатися до широкого спектру пристроїв, у тому числі тонких і товстих клієнтів, таких як мобільні телефони, ноутбуки, робочі станції та планшети, щоб забезпечити плавний доступ до обчислювальних ресурсів на цих різноманітних платформах. Це ключова характеристика хмарних технологій [5].



Рисунок 1.1 – Еволюція хмарного комп'ютингу [3]

Термін об'єднання ресурсів, що використовується у хмарному комп'ютингу для опису ситуації, в якій постачальники обслуговують кілька клієнтів або «орендаторів» з однаковими часовими та масштабованими послугами. Ці послуги можна налаштувати відповідно до потреб кожного клієнта без будь-яких змін, які будуть очевидними для клієнта або кінцевого користувача [6].

Швидка еластичність у хмарному комп'ютингу означає здатність хмари швидко масштабуватися для задоволення потреб. Споживачі отримують вигоду від швидкої еластичності, тому що вони можуть розширювати або зменшувати свої ресурси, коли і як вони хочуть. Хмарні постачальники з еластичним середовищем дозволяють клієнтам використовувати більше ресурсів, коли вони їм потрібні, або виводити з експлуатації ресурси, які їм більше не потрібні. Швидка еластичність у хмарних обчисленнях відбувається дуже швидко в режимі реального часу [7].

Вимірювана послуга означає, що постачальник послуг відстежує, скільки конкретної послуги споживає кожен клієнт протягом визначеного періоду часу. Вимірювані послуги дають постачальнику уявлення про споживання ресурсів і надають клієнтам прозорість того, як їм виставляються рахунки за послуги, які вони споживають. Вимірювані послуги підтримуються можливостями вимірювання, що відповідають типу споживання послуги. Наприклад, вимірювання ресурсів хмарних обчислень може ґрунтуватися на тому, скільки пам'яті використовує клієнт хмари, скільки обчислювальної потужності вони використовують, яку пропускну здатність мережі вони споживають [8].

Хмарний комп'ютинг включає в себе певні сервіси та моделі, що працюють за лаштунками та роблять хмарний комп'ютинг здійсненими та доступними для кінцевих користувачів.

1.1.1 Моделі розгортання

Моделі розгортання визначають тип доступу до хмари, тобто, як саме розташована ця хмара. Хмара може мати будь-який із чотирьох типів доступу: публічний, приватний, гібридний і ком'юніті.

Публічна хмара — класична модель хмарного комп'ютингу. У загальнодоступній хмарі постачальник послуг керує послугами та

інфраструктурою надаючи доступ до користування хмарою через мережу Інтернет [9]. Схему публічної хмари наведено на рисунку 1.2 [10].

Основними перевагами публічної хмари є:

- економічна ефективність, так як публічна хмара використовує одні й тіж самі ресурси для великої кількості клієнтів, вона є недорогою;
- надійність, завдяки тому що публічна хмара використовує велику кількість ресурсів із різних місць, якщо якийсь із ресурсів виходить з ладу, хмара може використовувати інший;
- гнучкий підхід, публічна хмара може легко інтегруватися з приватною хмарою;
- незалежне розташування.



Рисунок 1.2 – Схема публічної хмари [10]

Основним недоліком публічної хмари є низький рівень безпеки, оскільки дані розміщуються за межами майданчика, а ресурси загальнодоступні [10].

Приватна хмара - модель хмарного комп'ютингу де системи та служби доступні у межах лише однієї організації, тому її ще називають внутрішньою або корпоративною. Така хмара може розміщуватись в середині організації або на стороні постачальника. Рішення приватної хмари пропонують організаціям більше контролю та кращу безпеку хмарних серверів [11]. Модель приватної хмари наведено на рисунку 1.3 [12].

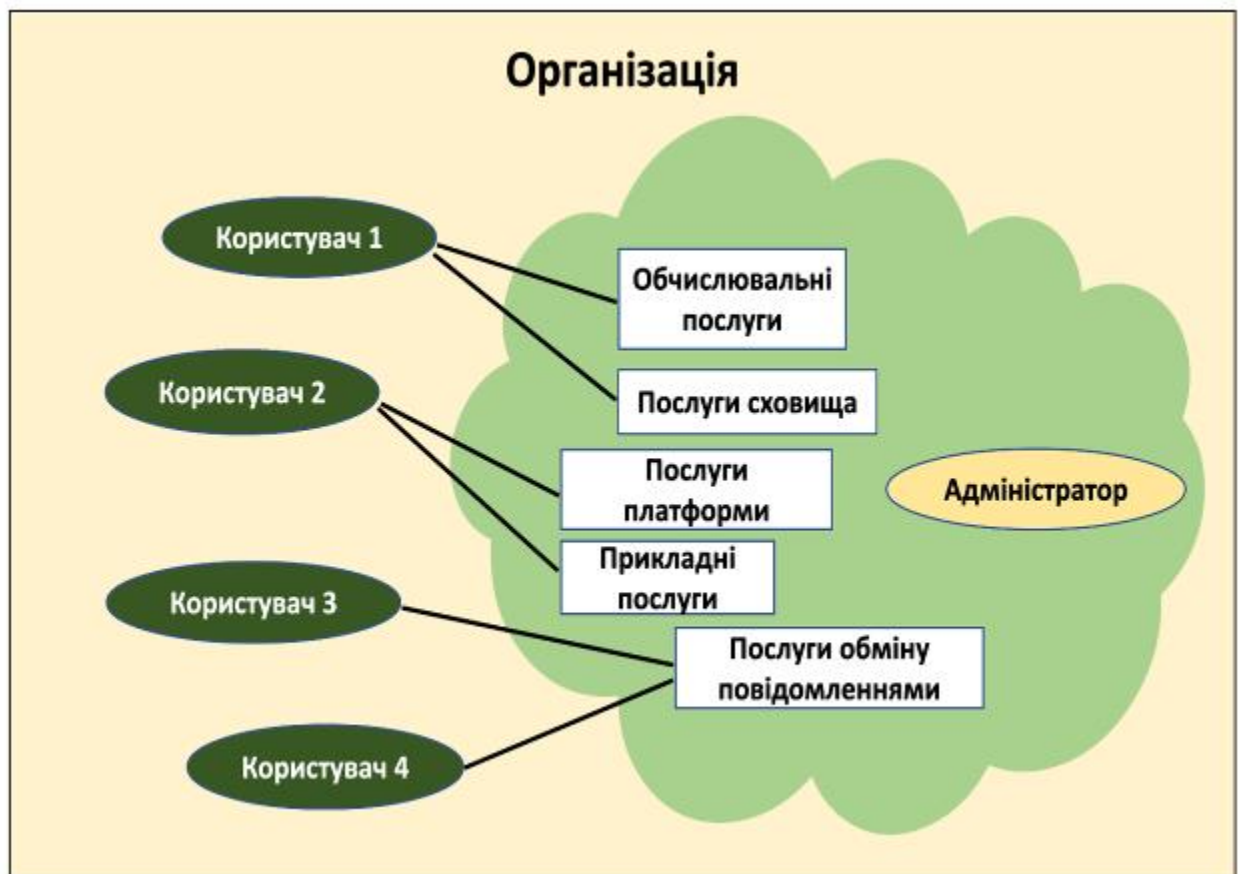


Рисунок 1.3 – Модель приватної хмари [12]

Приватна хмара має наступні переваги:

- високий рівень безпеки та конфіденційності, оскільки операції в приватній хмарі недоступні для користувачів поза організацією, а ресурси спільно використовуються з окремого пулу ресурсів;

– більший контроль над ресурсами та обладнанням, ніж публічна хмара, оскільки доступ до них можливий лише всередині організації.

Серед недоліків можна виділити обмежену зону дії, оскільки доступна лише локально, а розгорнути її глобально дуже складно, обмежену масштабованість, так як приватну хмару можна масштабувати лише в межах можливостей розміщених внутрішніх ресурсів та високу ціну. Також для підтримки хмарного розгортання організаціям потрібні кваліфіковані спеціалісти [13].

Модель ком'юніті хмари значною мірою нагадує приватну, різниця полягає лише у наборі користувачів. У той час як приватна хмара передбачає, що тільки одна компанія володіє сервером, у випадку ком'юніті хмари кілька організацій з однаковим досвідом спільно використовують інфраструктуру і пов'язані ресурси.

Організації з однієї спільноти зазвичай мають схожі вимоги до хмари щодо безпеки та відповідності вимогам, обчислювальних ресурсів та додатків, тому ком'юніті хмара підходить для організацій, які працюють над загальним проєктом, дослідницькою темою або програмою та мають доступ до аналогічних ресурсів. Хмара спільноти дозволяє таким організаціям спілкуватися, ділитися та співпрацювати, не покладаючись на публічні хмари [14]. На рисунку 1.4 наведено схему ком'юніті хмари.

Ком'юніті хмара має тіж переваги, що й приватна хмара, але за більш низькою ціною. Вона менш захищена ніж приватна хмара, але більш безпечніша, ніж публічна хмара.

До недоліків можна віднести складність розподілу обов'язків з управління, безпеки та витрат між організаціями [15].

Гібридна хмара є сумішшю публічної і приватної хмари. В ній розгортання хмари організації розділене між публічною та приватною хмарною інфраструктурою. Конфіденційні дані залишаються в приватній хмарі, де можуть підтримуватись високі стандарти безпеки.



Рисунок 1.4 – Схема ком'юніті хмари [15]

Операції, в яких не використовуються конфіденційні дані, виконуються у публічній хмарі, де інфраструктура може масштабуватися відповідно до вимог.

Такі хмари добре підходять для виконання операцій з великими даними на неконфіденційних даних у публічній хмарі, зберігаючи конфіденційні дані захищеними у приватній хмарі. Гібридні хмари також дають компаніям можливість запускати загальнодоступні програми або ресурсомісткі платформи розробки в загальнодоступній частині хмари, в той час як конфіденційні дані залишаються захищеними [16]. На рисунку 1.5 представлено схему гібридної хмари [17].

Перевагами гібридної хмари є масштабованість, хмара пропонує функції як масштабованості публічної хмари, так і масштабованості приватної хмари, економічність, безпека, так як приватна хмара в гібридній хмарі забезпечує більш високий рівень безпеки.

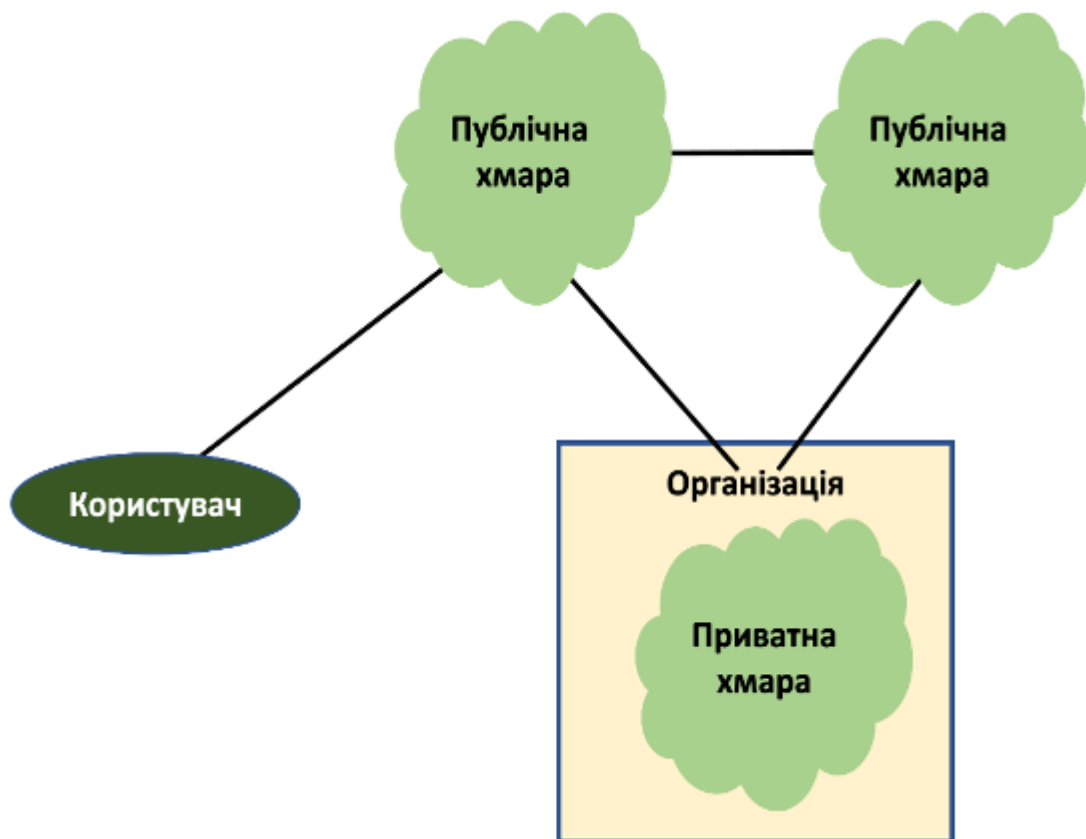


Рисунок 1.5 – Схема гібридної хмари [17]

До недоліків можна віднести складність мережи, яка виникає через наявність приватної та загальнодоступної хмари, інфраструктурну залежність, так як модель гібридної хмари залежить від внутрішньої ІТ-інфраструктури [18].

1.1.2 Типи сервісів

Хмарний комп'ютинг включає в себе три основні типи обслуговування: інфраструктура як послуга (IaaS), платформа як послуга (PaaS) та програмне забезпечення як послуга (SaaS).

IaaS, також відома як хмарні інфраструктурні сервіси, надає віртуалізовану платформу. Така віртуалізація призводить до створення так званого «монітора віртуальної машини». В IaaS користувачі можуть легко запускати різні програмні додатки та операційні системи. У сервісі хмарної

інфраструктури базовий набір керується постачальником хмарних послуг, але відповідна інфраструктура, така як операційна система та програмні додатки, керується та контролюється користувачем. В IaaS всі віртуальні ресурси зіставляються з фізичною системою. Коли користувач взаємодіє зі службою IaaS, він запитує ресурси з віртуальної системи, і ці запити напряду перенаправляються на реальний сервер, який фактично працює із запитами користувача [19].

Ресурси IaaS, такі як віртуальні машини, пристрої зберігання, пропускна спроможність, IP-адреси, моніторингові служби, брандмауери тощо, надаються клієнтам в оренду. Оплата ґрунтується на кількості часу, протягом якого клієнт зберігає ресурс. Також з адміністративним доступом до віртуальних машин клієнт може запускати будь-яке програмне забезпечення, навіть користувацьку операційну систему.

Модель обслуговування IaaS має наступні характеристики:

- Віртуальні машини з встановленим програмним забезпеченням;
- віртуальні машини з встановленими операційними системами, такими як Windows, Linux, Solaris тощо;
- доступність ресурсів на запит;
- дозволяє зберігати копії певних даних у різних місцях;
- обчислювальні ресурси можна легко збільшувати та зменшувати [20].

PaaS тип моделі служби хмарного комп'ютингу, де прикладна програмна платформа надається постачальником в першу чергу для розробників і програмістів. Засоби, які надає PaaS, включають хостинг, впровадження, тестування, дизайн та розробку додатків. Крім того, постачальники хмарних послуг, які пропонують PaaS, включають набір функцій для проектування, тестування, впровадження, управління та запуску додатків. Такий тип сервісу надає апаратне забезпечення, мережу та операційну систему, щоб клієнт міг розробити власний додаток або програмне забезпечення [21]. PaaS допомагає компаніям уникнути турбот і

витрат на встановлення обладнання або програмного забезпечення для розробки або розміщення нових додатків користувача. Команди розробників просто набувають доступу з оплатою в міру використання до всього, що їм потрібно для створення власних додатків, включаючи інфраструктуру, інструменти розробки, операційні системи та багато іншого [22].

Paas має наступні характеристики:

- надання середовища розробки на основі браузера, це дозволяє розробнику створювати базу даних та редагувати код програми або за допомогою інтерфейсу прикладного програмування, або за допомогою інструментів «вказіть та клацніть»;
- забезпечення вбудованою безпекою, масштабованістю та інтерфейсами веб-служб;
- надання вбудованих інструментів для визначення робочого процесу, процесів затвердження та бізнес-правил;
- легке інтегрування з іншими програмами на тій же платформі;
- надання інтерфейсів веб-сервісів, які дозволяють підключати програми поза платформою [23].

SaaS — це прикладне програмне забезпечення, розміщене у хмарі та яке використовується через інтернет-з'єднання через веб-браузер, мобільний додаток або тонкий клієнт. Постачальник SaaS відповідає за експлуатацію, керування та обслуговування програмного забезпечення та інфраструктури, на якій воно працює. Клієнт просто створює обліковий запис, вносить плату та приступає до роботи [24].

SaaS має наступні характеристики:

- доступ до ПЗ через Інтернет;
- програмні застосунки підтримуються постачальником;
- ліцензія на програмне забезпечення може бути заснована на підписці або використанні, що оплачується регулярно;
- доступність на запит;
- масштабованість на запит;

- автоматичне оновлення;
- всі користувачі використовують однакову версію програмного забезпечення [25].

На рисунку 1.6 зображено ключові відмінності між типами сервісних моделей [26].

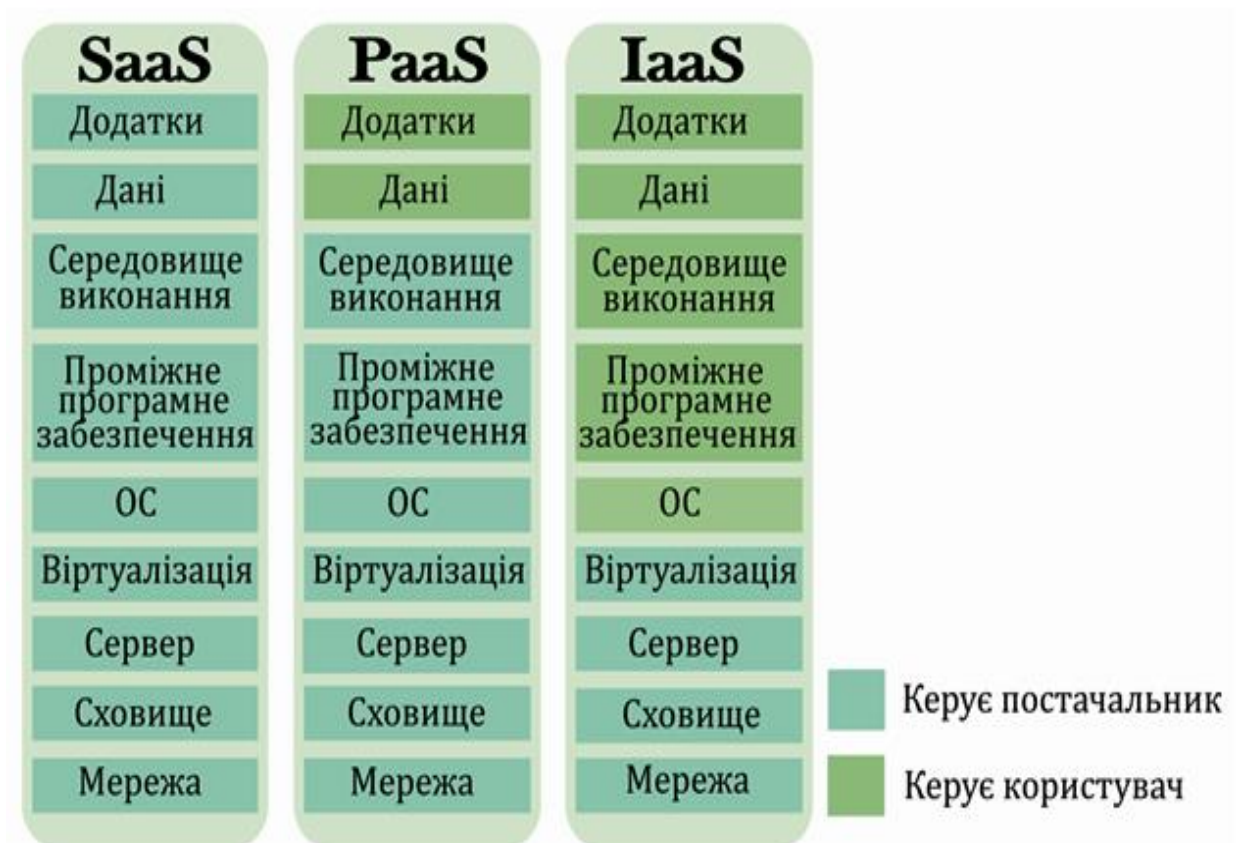


Рисунок 1.6 - Ключові відмінності між типами сервісних моделей [26]

1.2 Архітектура IoT

Інтернет речей або IoT - це, по суті, велетенська мережа з підключеними фізичними пристроями, транспортними засобами, побутовими приладами та іншими речами, які можуть підключатися, збирати та обмінюватися даними по дротовій або бездротовій мережі практично без взаємодії з людьми або втручання людини в комп'ютер. Забезпечуючи інтеграцію та обмін даними між фізичними пристроями та комп'ютером.

Технологія IoT спрямована на те, щоб зробити людське життя простішим та комфортнішим [27].

Загалом, архітектура IoT (рис. 1.7) складається з чотирьох етапів. Перший етап складається, як правило, з датчиків та приводів [28].

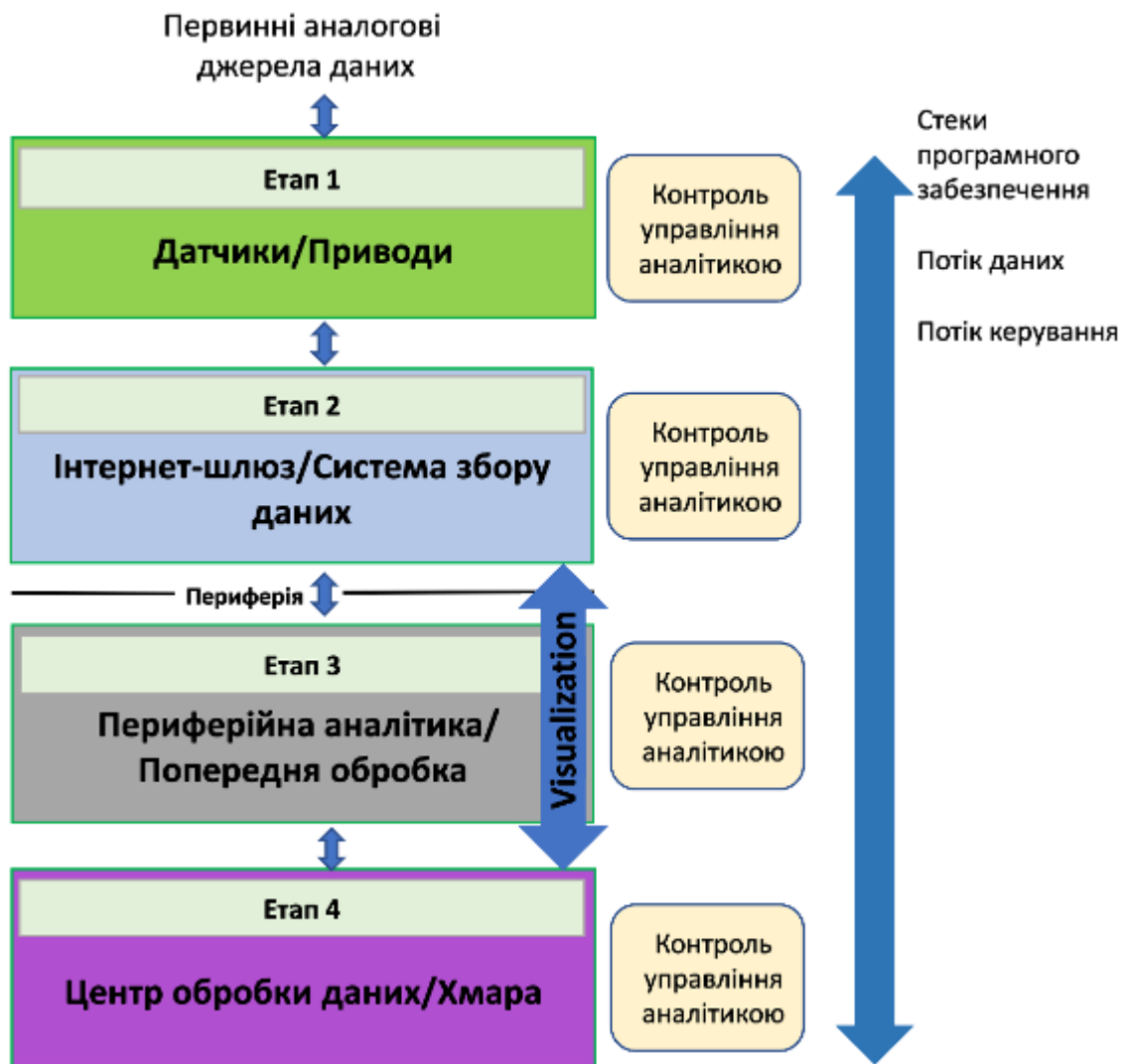


Рисунок 1.7 - Архітектура IoT [28]

Датчики реагують на певні типи умов у фізичному світі та генерують сигнал (зазвичай електричний), такий сигнал може представляти величину стану, що відстежується. Умовами на які реагують датчики можуть бути світло, тепло, звук, відстань, тиск або будь-яка інша більш конкретна ситуація, наприклад, наявність або відсутність газу або рідини. Найчастіше у

IoT використовують датчики температури, тиску, вологості, руху, наближення, газу, диму, оптичні тощо [29].

Приводи це механічні або електромехані пристрої, які перетворюють енергію в рух. У IoT приводи відповідають за фізичний рух об'єкта. Це може бути пристрій, який може переміщати предмети та жити від різних джерел, таких як батарея, електрика або енергія, що виробляється самостійно [30]. Приводи можна розділити на чотири основні категорії залежно від схеми їх побудови та ролі, яку вони грають у конкретному середовищі IoT. Лінійні приводи використовуються для забезпечення руху об'єктів або елементів прямої лінії. Мотори забезпечують точний обертальний рух компонентів пристрою чи цілих об'єктів. Реле — приводи на основі електромагнітів для керування силовими вимикачами в лампах, обігрівачах або навіть у «розумних» транспортних засобах. Соленоїди найбільш широко використовуються в побутовій техніці як частина механізмів блокування або запуску, вони також діють як контролери в системах моніторингу витоків газу та води [31].

Другий етап включає системи агрегації даних датчиків і перетворення аналогових даних в цифрові.

Дані з датчиків надходять в аналоговій формі, ці дані необхідно агрегувати та перетворити на цифрові потоки для подальшої обробки. Системи збору даних виконують функції агрегування та перетворення даних, вони підключається до мережі датчиків, поєднують вихідні дані та виконують аналого-цифрове перетворення. Інтернет-шлюз отримує агреговані та оцифровані дані та направляє їх через Wi-Fi, дротові локальні мережі або Інтернет для подальшої обробки [28].

На третьому етапі периферійні IT-системи виконують попередню обробку даних перед їх переміщенням до центру обробки даних або хмари. Після відцифрування та агрегування IoT даних, їх потрібно буде обробити, перш ніж вони будуть відправлені в центр обробки даних або хмару. Це робиться для того щоб ще більше зменшити їх обсяг. Периферійний пристрій

може виконувати певну аналітику в рамках попередньої обробки. Обробка цього типу зазвичай відбувається на пристрої у місці, близькому до того, де знаходяться датчики, наприклад, у комутаційній шафі на місці [32].

На четвертому етапі виконується аналіз, керування та зберігання даних. Дані, які потребують більш глибокої обробки або аналізу, в кінцевому підсумку можуть бути перенаправлені у фізичний центр обробки даних або на хмарний сервер для зберігання даних [33].

1.3 Хмарний комп'ютинг у сфері IoT

Хмарний комп'ютинг дуже добрий інструмент для сфери IoT, який вирішує потреби, пов'язані з даними.

Хмарні IoT сервіси включають базову інфраструктуру, сервери та сховища, необхідні для операцій в реальному часі та обробки даних. Хмари IoT також включають послуги та стандарти, необхідні для підключення, моніторингу та захисту різних пристроїв IoT та їх додатків [34].

Google Cloud IoT Core платформа сервісу Google Cloud Platform - це повністю керований сервіс, який дозволяє швидко та безпечно підключати, налаштовувати та отримувати дані з великої кількості пристроїв. При використанні Cloud Pub/Sub, Core можна поєднувати дані з децентралізованих пристроїв у єдину глобальну систему. У поєднанні з іншими сервісами Google Cloud IoT Core пропонує комплексне рішення для збирання, аналізу та візуалізації даних IoT у режимі реального часу, що дозволяє створювати багатофункціональні моделі, які оптимізують чи прогнозують дані [35].

Google Cloud IoT Core підтримує звичайні протоколи MQTT та HTTP, що дозволяє клієнтам використовувати безліч існуючих пристроїв для створення своїх IoT систем. Цей сервіс працює в інфраструктурі Google, яка автоматично масштабується в режимі реального часу, дозволяючи за

необхідності підключити мільйон пристроїв [36]. Інтерфейс сервісу Google Cloud IoT Core наведено на рисунку 1.8 [37].

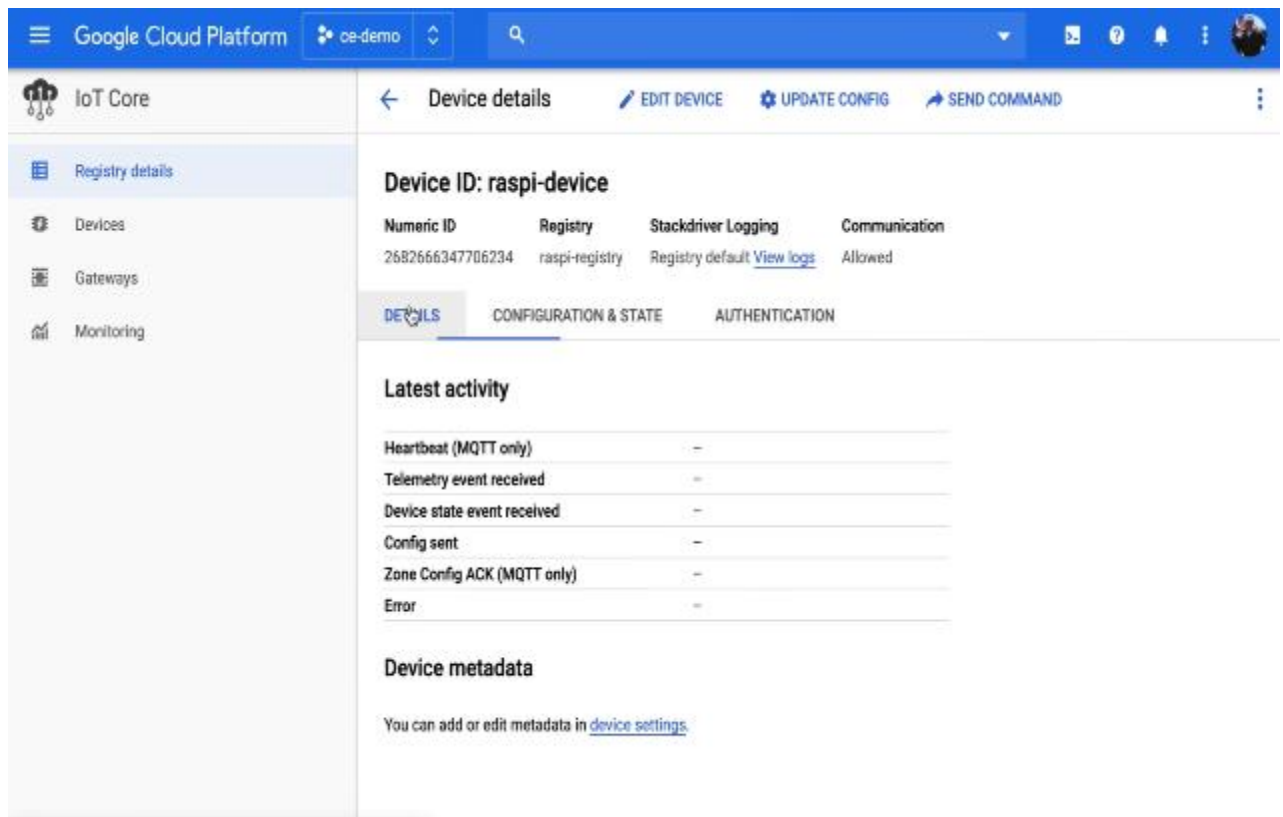


Рисунок 1.8 - Інтерфейс сервісу Google Cloud IoT Core [37]

Microsoft Azure IoT — це набір хмарних служб, що керуються корпорацією Microsoft, які підключають, відстежують і контролюють мільярди ресурсів IoT. Простіше кажучи, рішення IoT складається з одного або декількох пристроїв IoT, які взаємодіють з однією або декількома серверними службами, розміщеними у хмарі [38].

Насамперед Microsoft Azure IoT служить хмарним шлюзом, який з'єднує всі інтелектуальні пристрої з хмарою та встановлює зв'язок між ними. Сервіс може масштабуватися для підключення мільйонів датчиків та може обробляти величезні обсяги даних. Він підтримує кілька протоколів, таких як HTTP, AMQP, MQTT, та забезпечує можливості керування командами. Він також відповідає за аутентифікацію кожного пристрою та відіграє важливу

роль у аспектах безпеки [39]. Інтерфейс сервісу Microsoft Azure IoT наведено на рисунку 1.9 [40].

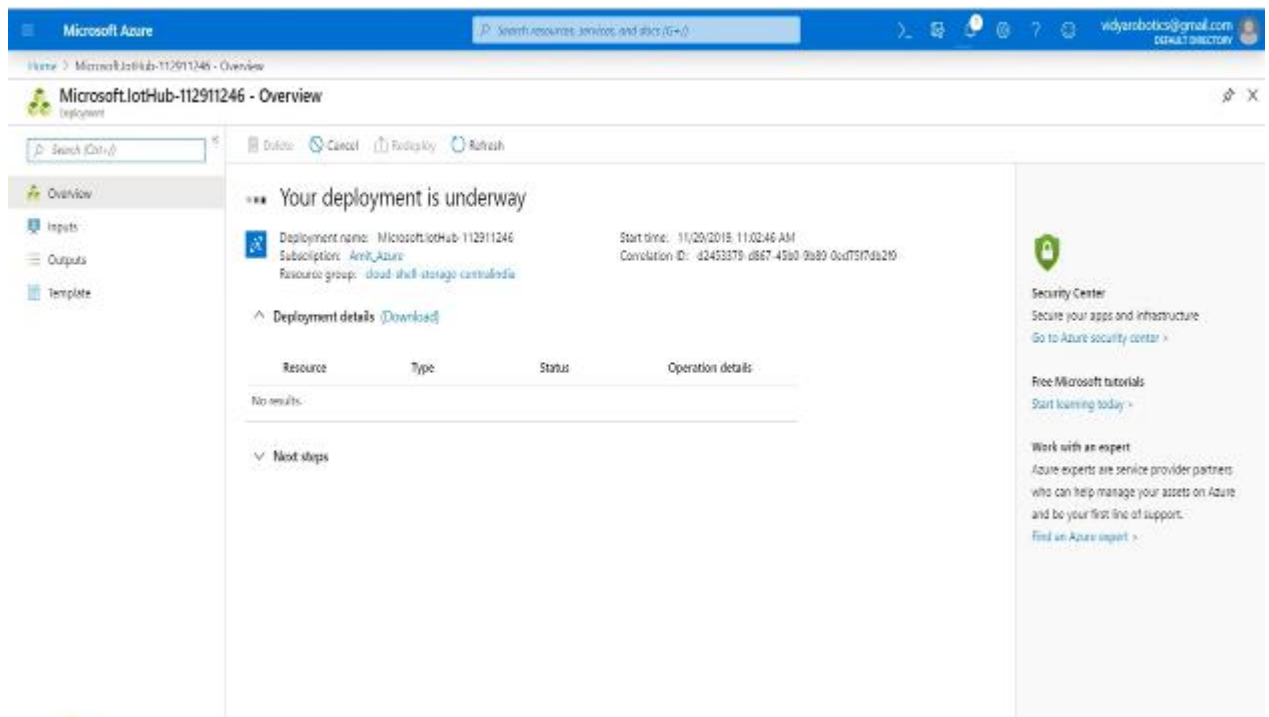


Рисунок 1.9 - Інтерфейс сервісу Microsoft Azure IoT [40]

КааIoT - це комплексна платформа IoT, яка застосовується для корпоративних IoT-проектів будь-якого масштабу. Вона надає ряд функцій, які дозволяють розробникам створювати передові програми для інтелектуальних продуктів, гнучко керувати підключеними пристроями через хмару, організовувати наскрізну обробку даних, аналізувати телеметрію пристроїв та багато іншого. Завдяки готовим функціям IoT, що надаються Каа, можна створювати свої IoT-програми в рази швидше [41].

Всі функції в КааIoT реалізовані за допомогою переносимих мікросервісів, а сам Каа заснований на гнучкій мікросервісній архітектурі. Це означає, що можна окремо налаштовувати кожну функцію КааIoT, додавати нові або замінювати наявні деякими сторонніми інструментами.

За промовчанням КааIoT надає всі функції Інтернету речей, які можуть знадобитися для типового додатка Інтернету речей - від збору даних

та управління пристроями до інформаційних панелей та аналітики Інтернету речей [41]. Інтерфейс платформи КaaIoT зображено на рисунку 1.10 [42].

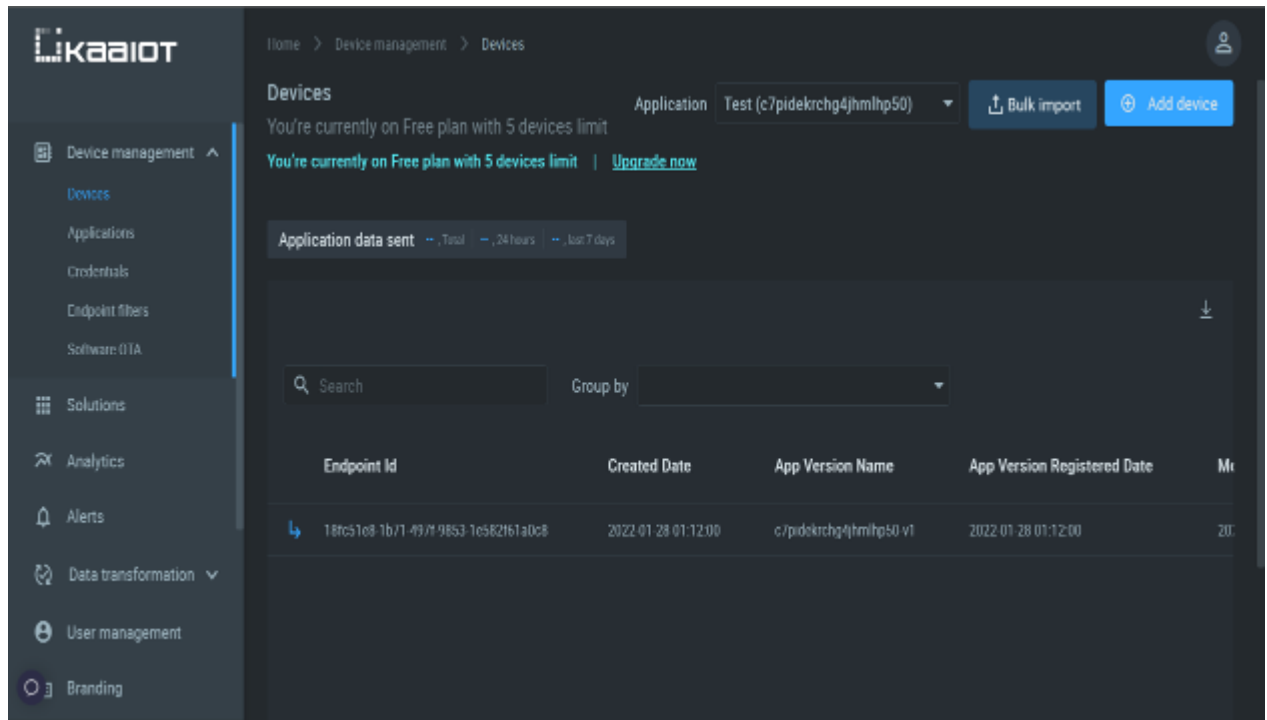
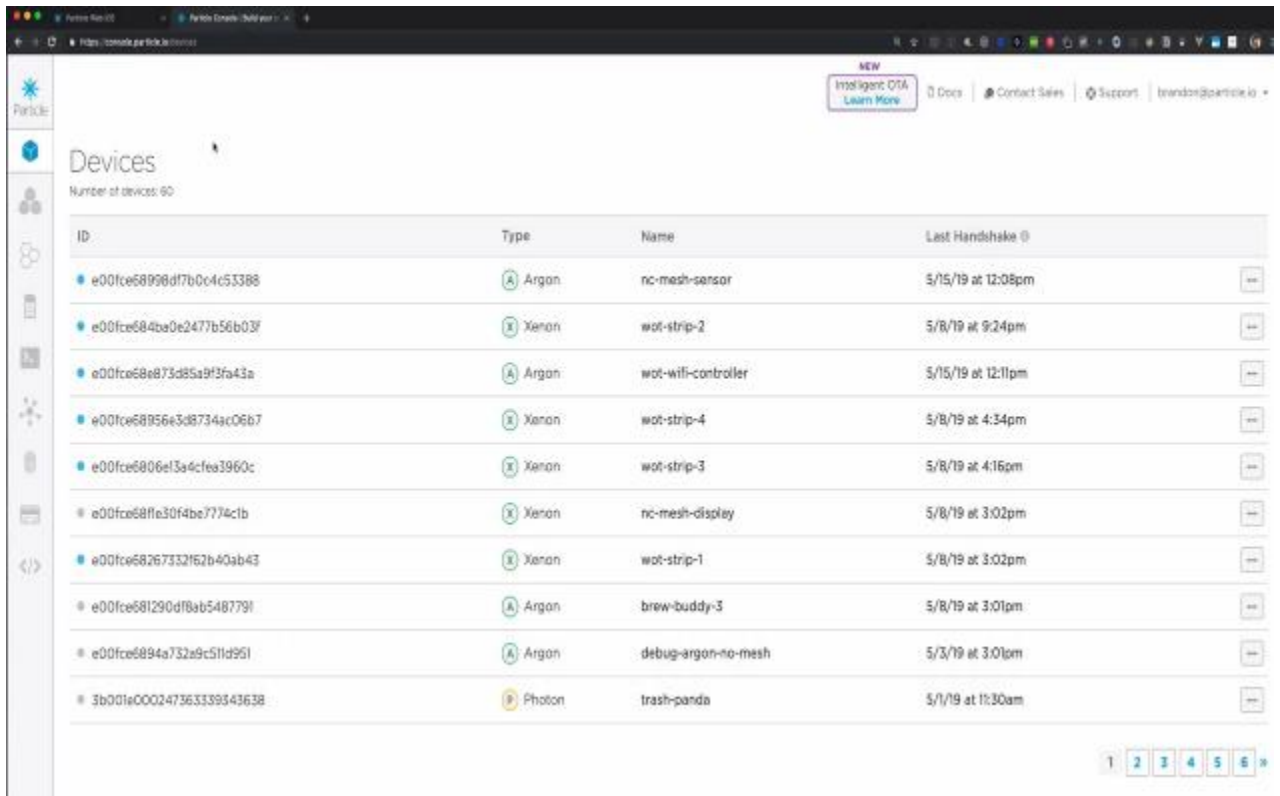


Рисунок 1.10 - Інтерфейс сервісу КaaIoT [42]

Particle - повноцінна хмарна платформа, що відповідає основним потребам IoT, таким як обладнання, операційна система, можливості підключення та надійна хмарна інфраструктура. Крім того, Particle поставляється з багатьма функціями моніторингу, які дозволяють користувачам отримувати повну інформацію про свої пристрої IoT. Також, удосконалений конвеєр даних, представлений усередині, допомагає користувачам із зручним керуванням даними та командами керування пристроями через хмарні операції. Також сервіс дозволяє отримувати інформацію на основі аналітики та маршрутизувати критичні оповіщення у режимі реального часу. Particle також може точно відстежувати фактичне розташування об'єктів IoT, розкиданих по всьому світу [43]. Інтерфейс платформи Particle зображено на рисунку 1.11 [44].



Particle
NEW
Intelligent OTA
Learn More
Docs
Contact Sales
Support
brandon@particle.io

Devices

Number of devices: 10

ID	Type	Name	Last Handshake @
e00fce6898df7b0c4c53388	Argon	nc-mesh-sensor	5/15/19 at 12:08pm
e00fce684ba0e2477b56b03f	Xenon	wot-strip-2	5/8/19 at 9:24pm
e00fce68e873d85a9f3fa43a	Argon	wot-wifi-controller	5/15/19 at 12:11pm
e00fce68956e3d8734ac06b7	Xenon	wot-strip-4	5/8/19 at 4:34pm
e00fce6806e13a4cfea3960c	Xenon	wot-strip-3	5/8/19 at 4:16pm
e00fce68f630f4be7774c1b	Xenon	nc-mesh-display	5/8/19 at 3:02pm
e00fce68267332f62b40ab43	Xenon	wot-strip-1	5/8/19 at 3:02pm
e00fce681290df8ab548779f	Argon	brew-buddy-3	5/8/19 at 3:01pm
e00fce6894a732a9c511d95f	Argon	debug-argon-no-mesh	5/3/19 at 3:01pm
3b001e000247363339343638	Photon	trash-panda	5/1/19 at 11:30am

1 2 3 4 5 6 »

Рисунок 1.11 - Інтерфейс сервісу Particle [44]

1.4 Постановка задач магістерської роботи

Хмарні технології надають безліч переваг для IoT сфери (рис. 1.12). Існує велика кількість хмарних IoT сервісів, які дозволяють візуалізувати та обробити дані, що отримані з датчиків, але на жаль вони не надають можливості переглянути відеозображення цих датчиків у разі надзвичайної ситуації (наприклад, відсутність даних з датчика, незрозумілі різкі стрибки показників тощо) та таким чином контролювати їх стан.

Тому, актуальною задачею є створення IoT сервісу, який буде не лише отримувати, обробляти та візуалізувати дані з датчиків, а й дозволить переглянути відео з камери, що наведена на ці датчики, в певний момент часу.

Метою роботи є дослідження методів та засобів хмарного комп'ютингу та розробка IoT сервісу для отримання та візуалізації даних з датчиків, а також відеотрансляції для контролю стану IoT пристроїв.



Рисунок 1.12 – Переваги хмарного компютингу [45]

Для досягнення поставленої мети необхідно розв'язати наступні завдання:

- аналіз віртуального та віддаленого інструментарію для прототипування IoT систем;
- аналіз вимог та розробка архітектури системи;
- розробка удосконаленого методу роботи з відеопотоком;
- вибір інструментарію програмної реалізації;
- розробка та тестування прототипу IoT сервісу.

2 АНАЛІЗ ВІРТУАЛЬНОГО ТА ВІДДАЛЕНОГО ІНСТРУМЕНТАРІЮ ДЛЯ ПРОТОТИПУВАННЯ ІОТ СИСТЕМ

2.1 Віртуальний робочий простір

SimulIDE [46] – безкоштовний симулятор електронних схем у реальному часі з відкритим вихідним кодом. Це простий інструмент, призначений для навчання. Він розроблений, так щоб бути простим у використанні. SimulIDE надає можливість працювати з мікроконтролерами AVR, Arduino та PIC, і має вбудований відладчик до них [47].

На рисунку 2.1 зображено інтерфейс SimulIDE. У розділі «components» перелічені всі доступні компоненти, з яких можна вибирати. Користувачу просто потрібно перетягнути кожен необхідний компонент на головний екран і з'єднати їх між собою дротами, які автоматично прокладаються, коли користувач малює лінію за допомогою миші. Компоненти згруповані по категоріях, що значно полегшує пошук необхідного.

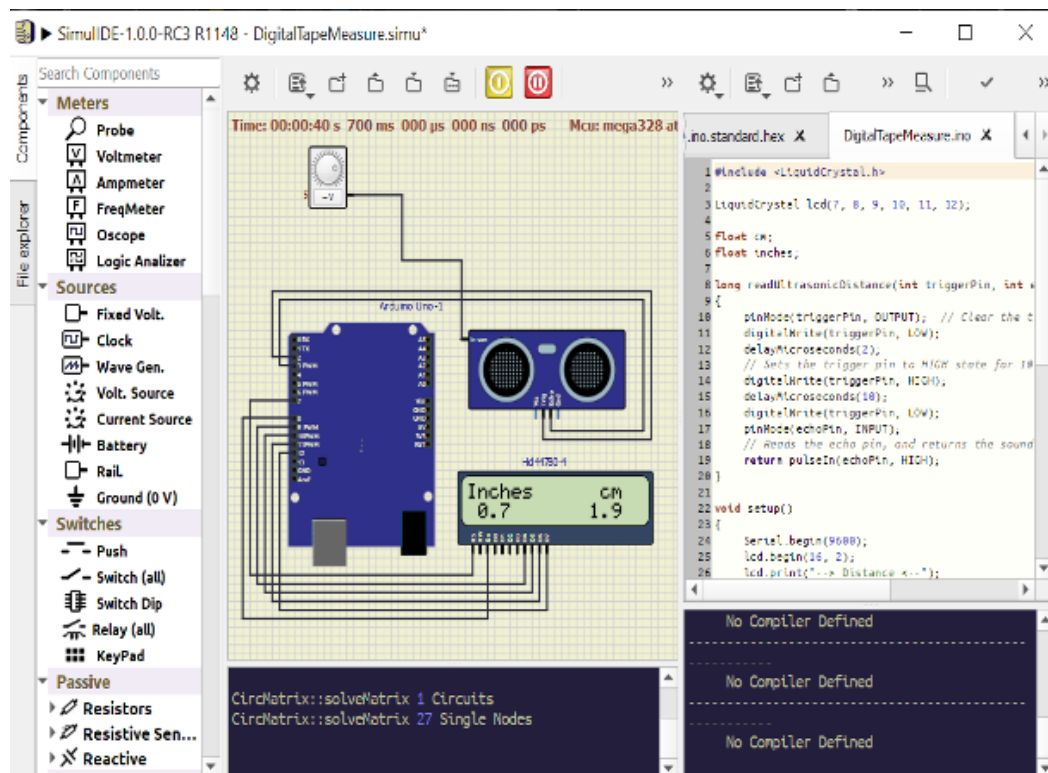


Рисунок 2.1 – Інтерфейс SimulIDE

Wokwi [48] – це безкоштовний онлайн-симулятор, для створення електронних прототипів, який імітує електронні компоненти та підтримує різні плати розробки та мікроконтролери, такі як ESP32, Arduino UNO, Arduino Mega, Raspberry Pi Pico та ATTiny85 (рис 2.2) [49].

На рисунку 2.3 зображено інтерфейс симулятора, у лівій частині знаходиться текстовий редактор, де можна працювати з файлами програми, а у правій частині знаходиться поле для створення прототипів, для вибору елементу необхідно натиснути кнопку «+» та вибрати із списку необхідну деталь.

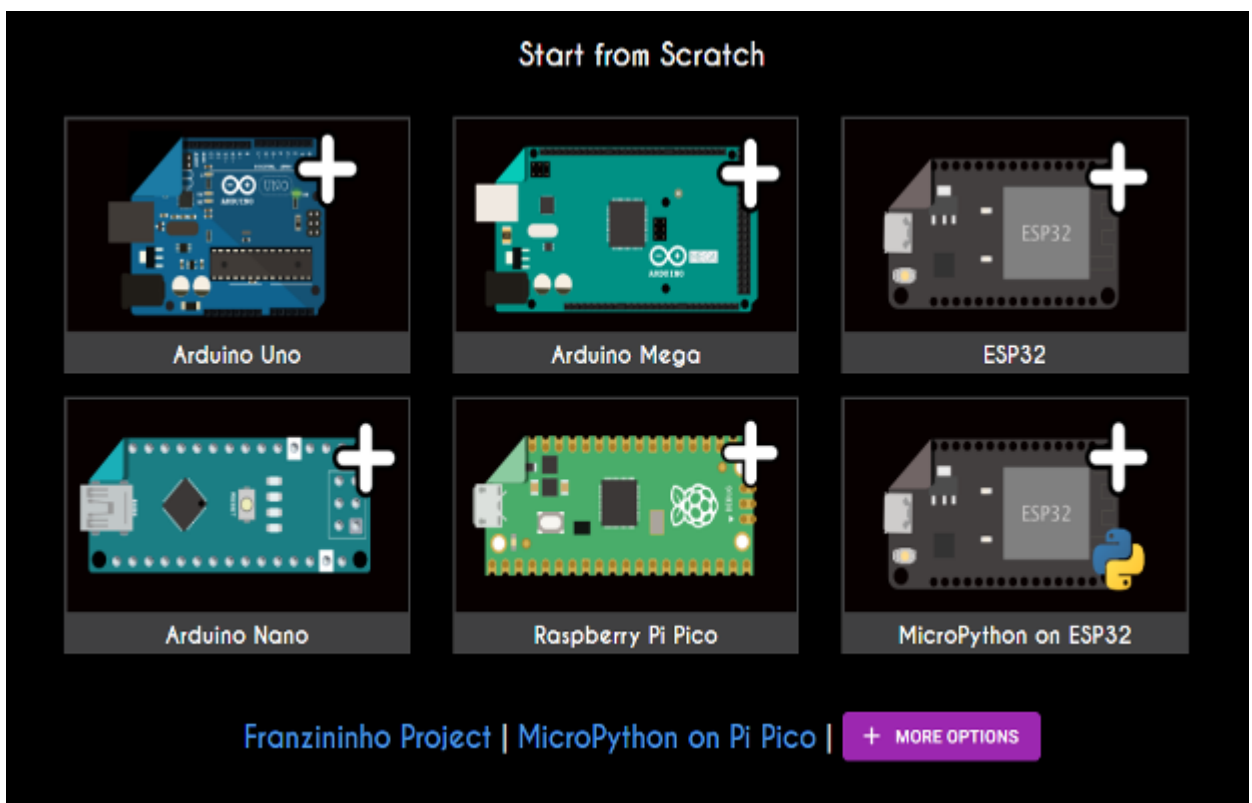


Рисунок 2.2 – Плати та мікроконтролери, які підтримує Wokwi

Simuli [50] (раніше відомий як IoTIFY Virtual Lab) - це повністю інтерактивний робочий простір для створення електронних прототипів на основі браузера. Він дозволяє розробляти прошивку та програми IoT у браузері на поширених вбудованих апаратних платформах. Крім того, сервіс дозволяє отримати доступ до своїх проєктів із будь-якого місця через браузер [51].

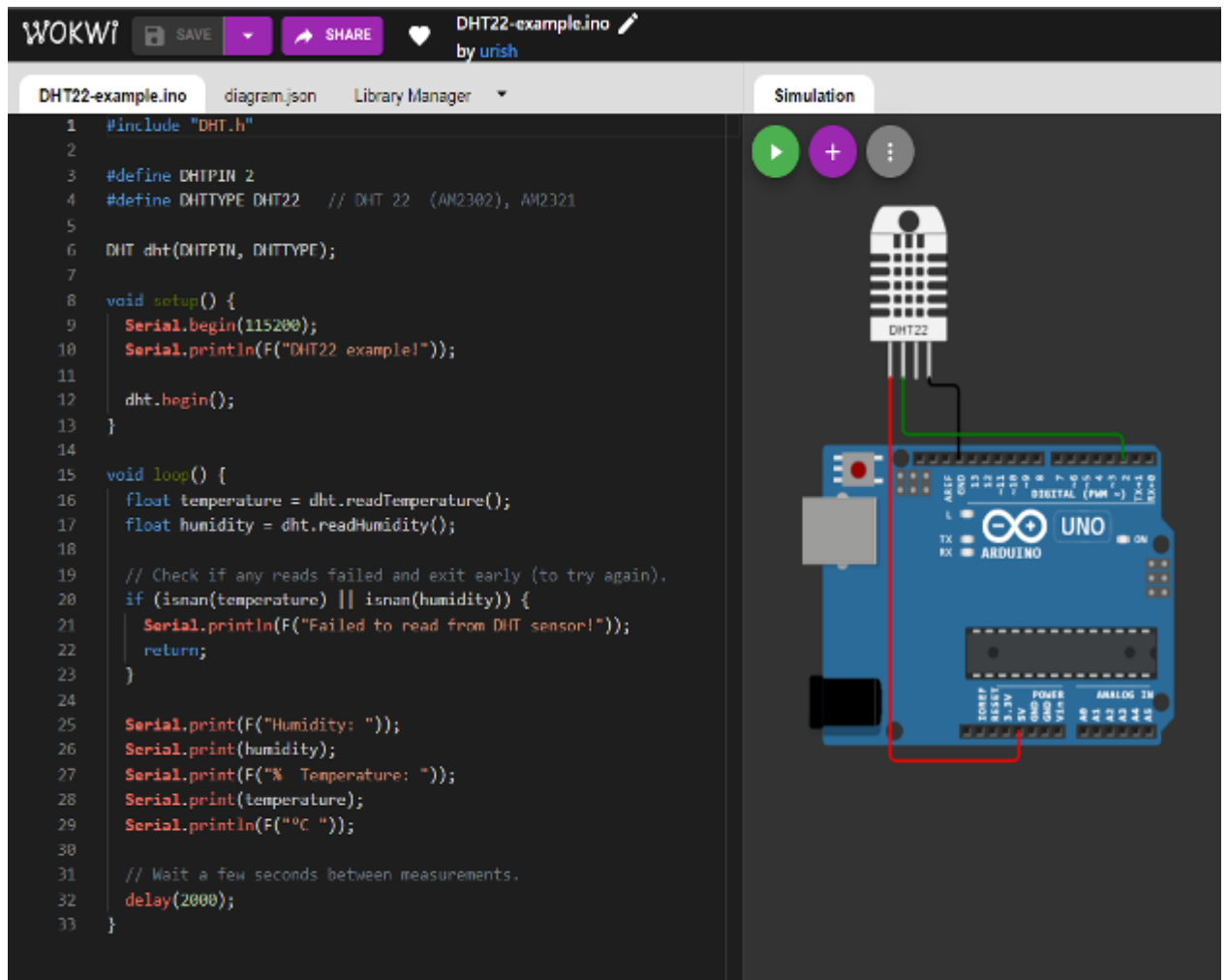


Рисунок 2.3 – Інтерфейс Wokwi

На даний момент Simuli підтримує Arduino Uno, STM32 Nucleo F411RE, Raspberry Pi та ряд периферійних пристроїв (рис. 2.4 – 2.5) . Сервіс надає надає онлайн робочий простір, де можна швидко зібрати своє власне віртуальне обладнання, створити та завантажити прошивку або відразу ж приступити до програмування. Все, що для цього потрібно, це браузер з підключенням до Інтернету.

Безкоштовна версія сервісу дає можливість працювати з Arduino Uno та STM32 Nucleo F411RE, та надає дві години роботи проекту на місяць, тому слід не забувати призупиняти роботу проекту, якщо він не використовується. Також безкоштовна версія дозволяє створювати лише один проект.

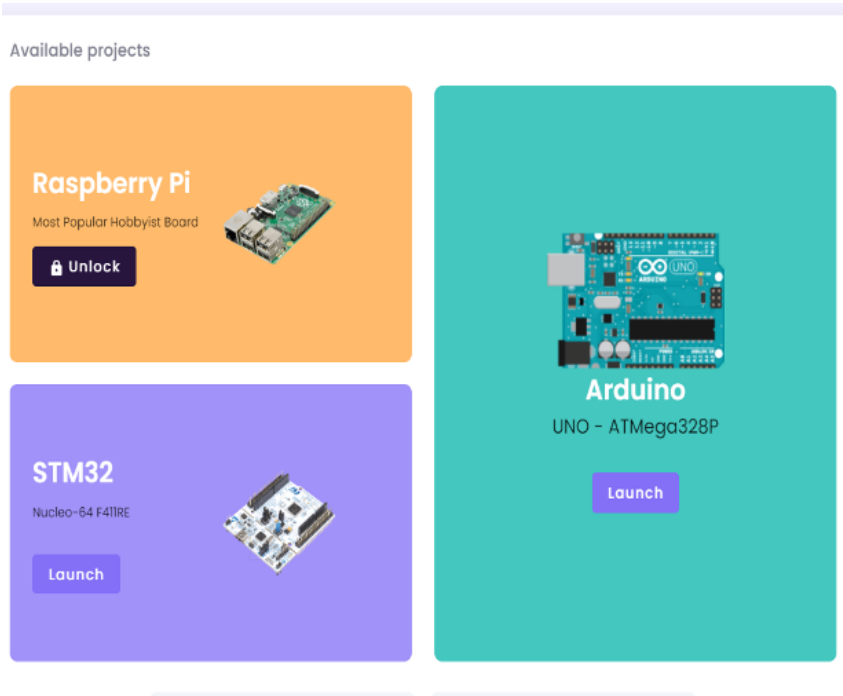


Рисунок 2.4 – Плати, які підтримує Simulі

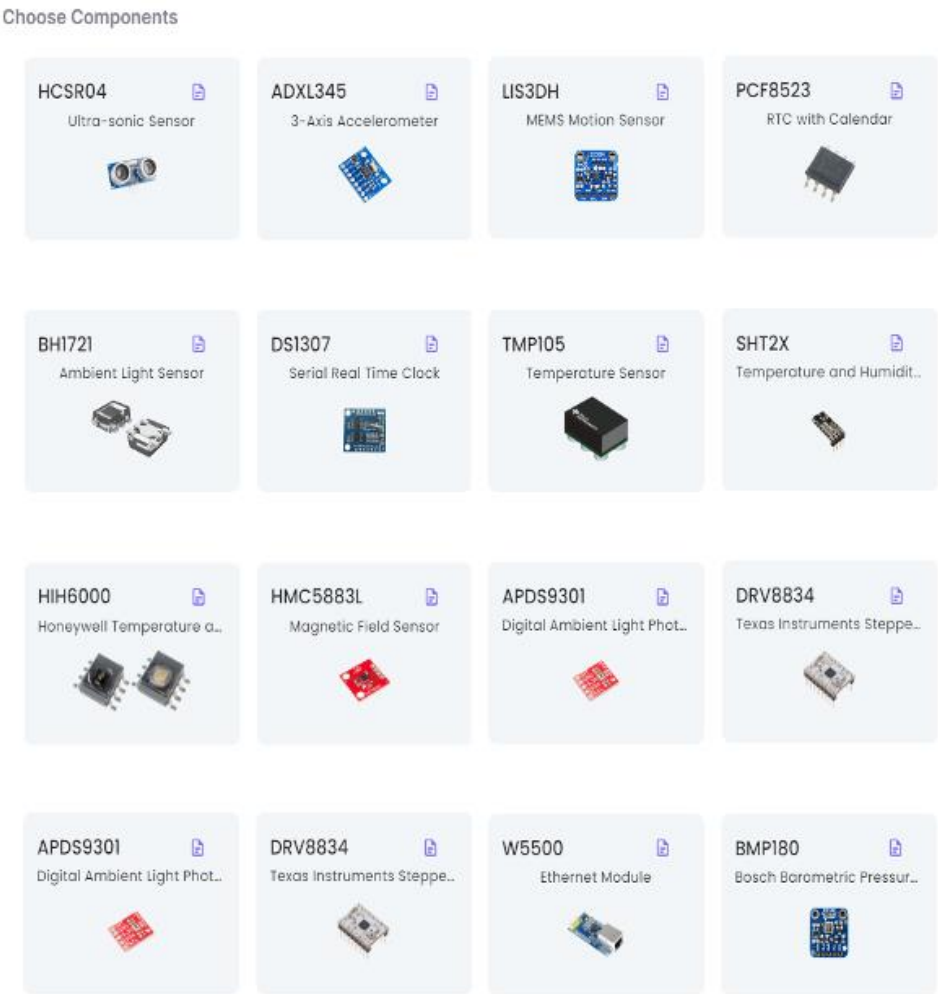


Рисунок 2.5 – Пристрої, які підтримує Simulі

1.2.1 Робота з платою STM32 Nucleo 64 датчиком BMP180 та вбудованим IDE у Simuli

Плати STM32 Nucleo від компанії STMicroelectronics є альтернативою платам Arduino. Зараз компанія пропонує три типи плат STM32 Nucleo: Nucleo-32, Nucleo-64, Nucleo-144. Плати Nucleo-32 сумісні із сокетом Arduino Nano. Плати Nucleo-64 та Nucleo-144 мають підтримку Arduino Uno [52].

Simuli Virtual Lab дає можливість працювати з платою Nucleo-64 F411RE (рис. 2.6).

Датчик BMP180 (рис. 2.7) є недорогим і простим у використанні сенсорним датчиком, що дозволяє виміряти атмосферний тиск і температуру навколишнього середовища. Використовують даний датчик для визначення висоти (чим вище, тим менший тиск), а також у саморобних метеостанціях.

Технічні характеристики:

- напруга живлення: 3.3 В – 5 В;
- робочий струм: 0.5 мА;
- діапазон вимірюваного тиску: 300 гПа – 1100 гПа;
- діапазон вимірюваної температури: -40 °С - +85 °С;
- інтерфейс: I2C [53].

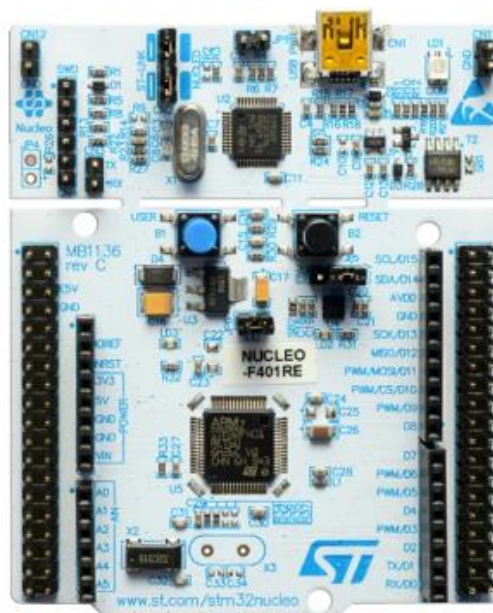


Рисунок 2.6 – Плата STM32 Nucleo-64 F411RE [54]



Рисунок 2.7 – Датчик BMP180 [53]

У віртуальній лабораторії Simulі слід обрати STM32 та натиснути кнопку «Launch» (рис. 2.8).

Це відкриє меню конфігурації. Спочатку необхідно вказати назву проєкту. Потім додати датчик барометричного тиску і температури BMP 180, вибравши його зі списку доступних компонентів (рис. 2.9).

Available projects

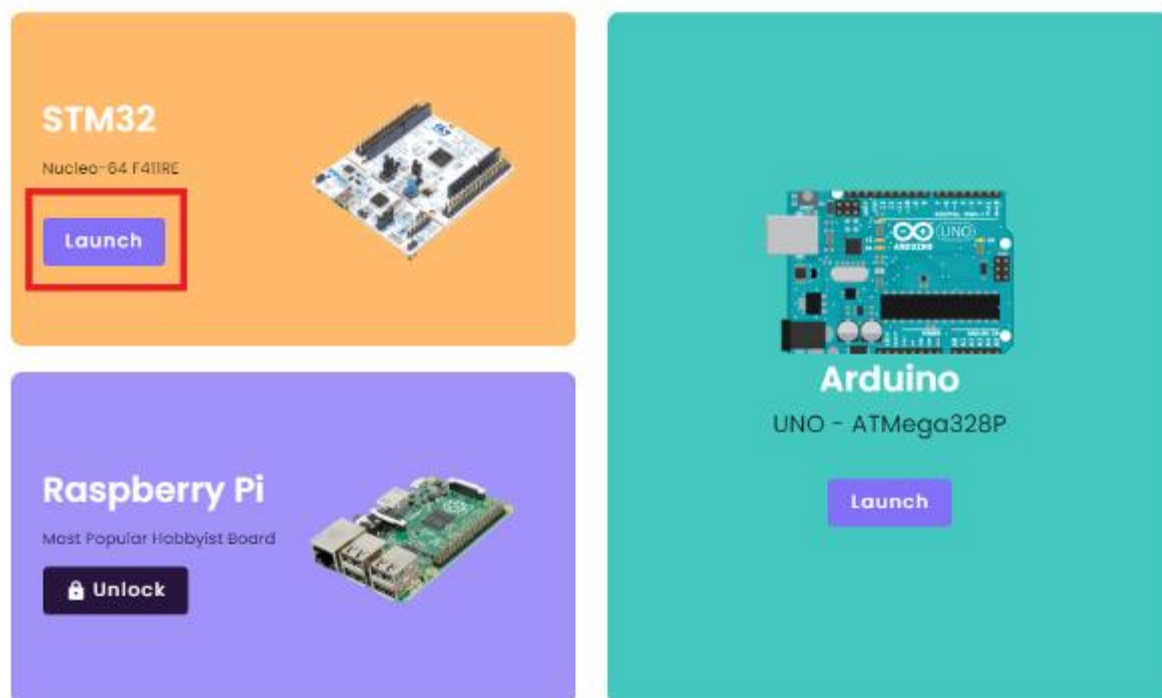


Рисунок 2.8 –Вибір STM32 Nucleo

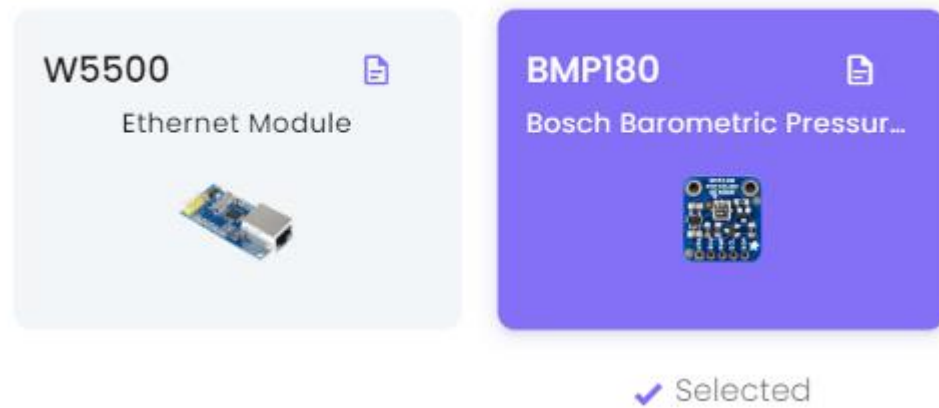


Рисунок 2.9 – Вибір датчика BMP 180

Після того, як буде натиснута кнопка «Launch», буде створено новий екземпляр STM32. Це може зайняти кілька хвилин. Коли новий екземпляр буде готовий, з'явиться кнопка «Open». Слід натиснути на цю кнопку, щоб відкрити та почати працювати з STM32.

Відкриється нова вкладка з вбудованим середовищем розробки (рис. 2.10). З лівого боку знаходиться вікно редактора коду разом із терміналом внизу. На правій стороні знаходиться плата STM32 Nucleo з терміналом для взаємодії з платою та датчиком BMP180. Одразу над платою розташоване посилання, якщо перейти за цим посиланням, то у новій вкладці відкриється повномасштабне вікно з платою.

Спочатку потрібно ініціалізувати новий проєкт за допомогою Platform ІО. Для цього необхідно створити нову папку, в якій буде зберігатися весь наш код, пов'язаний з проєктом. Слід клацнути правою кнопкою миші в будь-якому місці панелі «Explorer», вибрати «New Folder», назвати папку та натиснути «ОК». Тепер, коли папка готова, треба у терміналі перейти до щойно створеної папки за допомогою команди:

```
cd /workspace/folder_name/
```

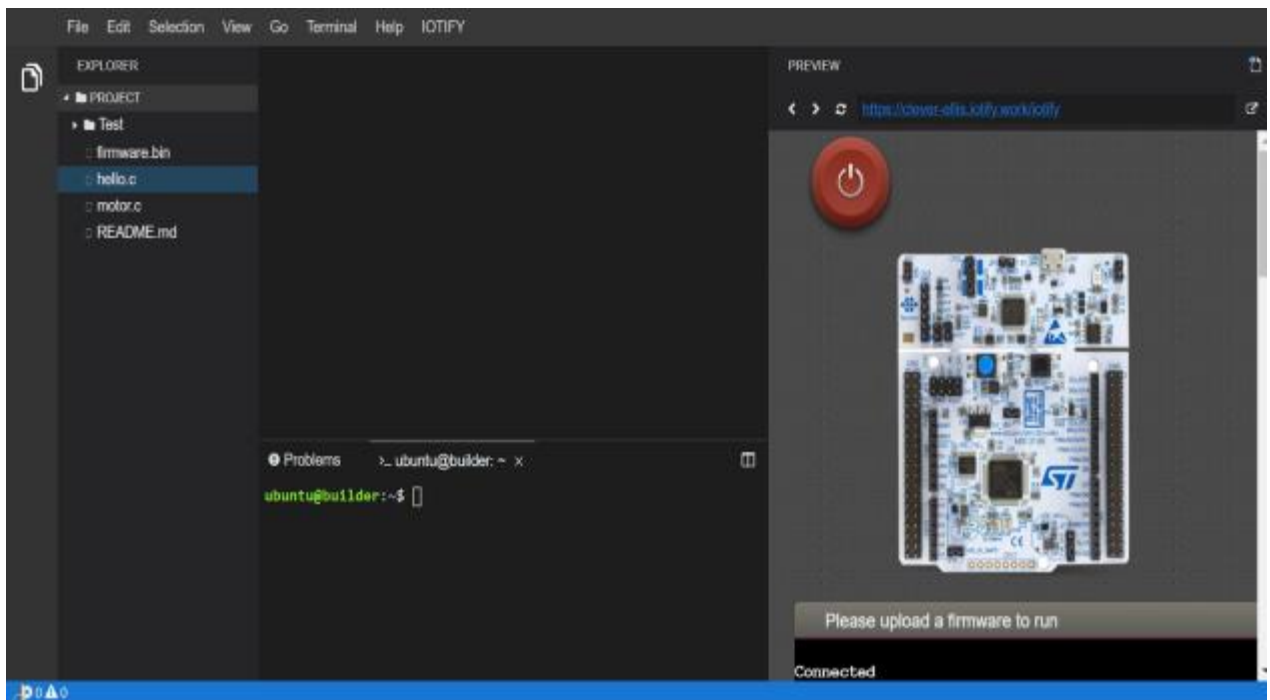


Рисунок 2.10 – Вікно проєкту з платою STM32 Nucleo

Далі необхідно ініціалізувати новий проєкт за допомогою Platform IO. Для цього слід ввести наступну команду в терміналі:

```
pio project init --board nucleo_f411re
```

Після успішного виконання команди «init» у створеній папці з’явиться кілька нових папок. У папці src слід створити новий файл під назвою main.cpp. Цей файл міститиме код програми BMP180. Слід натиснути на файл main.cpp, щоб відкрити його в редакторі коду. Далі необхідно встановити бібліотеку Adafruit BMP085 та написати код, який буде зчитувати данні з датчика та виводити їх у консоль.

Для встановлення бібліотеки слід у терміналі ввести команду:

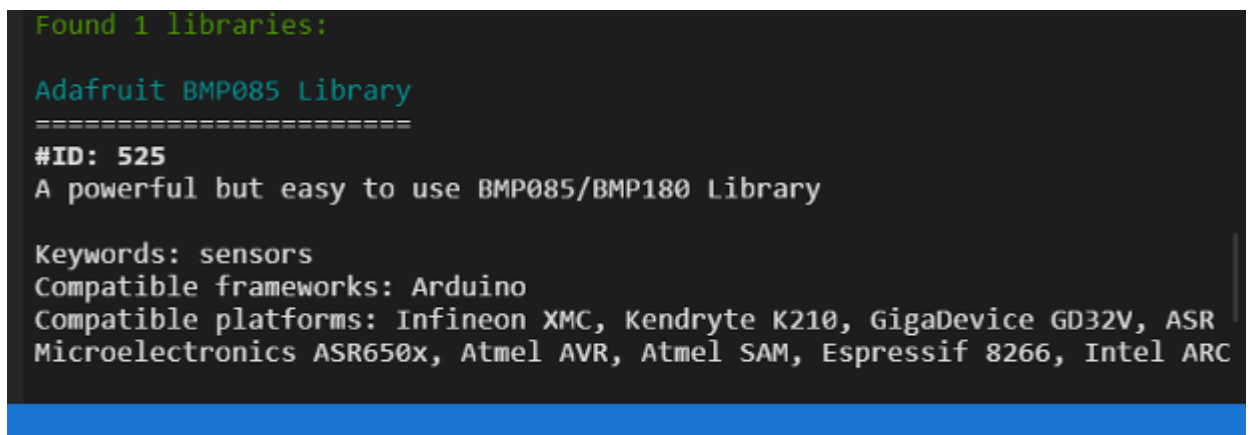
```
pio lib search Adafruit_BMP085
```

Ця команда шукатиме відповідні бібліотеки та надасть результати (рис. 2.11). В результаті можна побачити ідентифікатор бібліотеки, яку

шукали. В даному випадку це 525. Самей цей ідентифікатор буде використано для встановлення бібліотеки. Для встановлення бібліотеки в консолі слід ввести наступну команду:

```
pio lib install 525
```

Це встановить бібліотеку Adafruit BMP085 разом з усіма її залежностями.



```
Found 1 libraries:
Adafruit BMP085 Library
=====
#ID: 525
A powerful but easy to use BMP085/BMP180 Library

Keywords: sensors
Compatible frameworks: Arduino
Compatible platforms: Infineon XMC, Kendryte K210, GigaDevice GD32V, ASR
Microelectronics ASR650x, Atmel AVR, Atmel SAM, Espressif 8266, Intel ARC
```

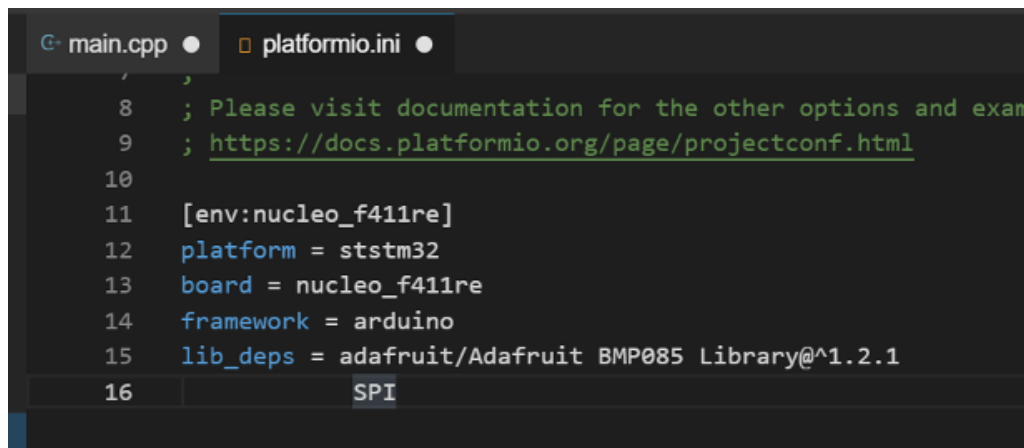
Рисунок 2.11 – Результати пошуку бібліотеки

Далі потрібно додати SPI як залежність для цього проєкту. Для цього треба відкрити файл platformio.ini в папці проєкту, яку було створено, і додати SPI під lib_deps, так, щоб він вміст файлу виглядав так, як зображено на рисунку 2.12.

Тепер код готов до компіляції. Необхідно у терміналі ввести команду:

```
pio run
```

Після компіляції з'явиться файл firmware.bin. Треба скопіювати цей файл мікропрограми з .pio/build/nucleo_f411re до /workspace.



```

8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:nucleo_f411re]
12 platform = ststm32
13 board = nucleo_f411re
14 framework = arduino
15 lib_deps = adafruit/Adafruit BMP085 Library@^1.2.1
16           SPI

```

Рисунок 2.12 – Редагування файлу platformio.ini

Зазвичай потрібно використовувати команду `sr` у терміналі, але для зручності можна використати наступну спеціальну команду:

`load-firmware`

Після її виконання файл `firmware.bin` буде скопійовано до папки робочої області. Після того, як файл `firmware.bin` буде скопійовано, все, що потрібно зробити, це натиснути червону кнопку над STM32. Результати роботи наведено на рисунку 2.13.

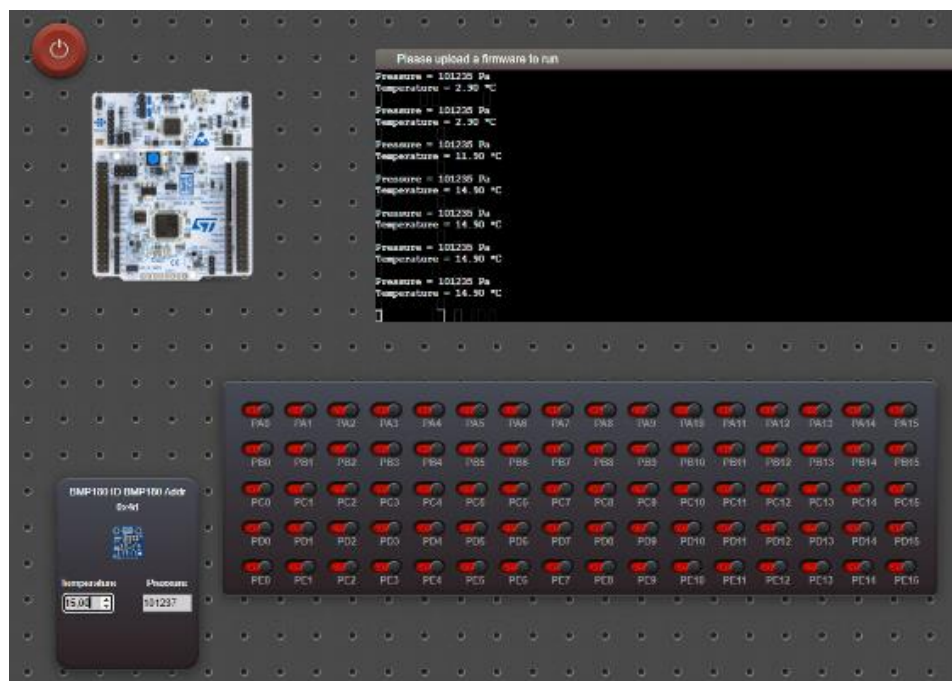


Рисунок 2.13 – Результат роботи

1.2.2 Робота з платою Arduino Uno модулем W5500 та сторонніми сервісами у Simuli

Adafruit IO – це платформа, розроблена для відображення, відповіді та взаємодії з даними IoT проекту. Вона призначена в першу чергу для зберігання та отримання даних [55].

Для роботи з сервісом слід зареєструватися на сайті Adafruit IO [56]. Далі необхідно створити канал. У цьому випадку буде створено канал під назвою lamp-assistant. Для створення каналу на головній сторінці Adafruit IO слід обрати IO >Feeds >view all (рис. 2.14), Це відкриє список усіх каналів, які зараз налаштовані у обліковому записі. Далі слід створити канал, для цього необхідно натиснути кнопку New Feed, це відкриє спливаюче вікно, у якому у полі Name слід ввести назву, наприклад lamp-assistant (рис. 2.15) та натиснути «Create».

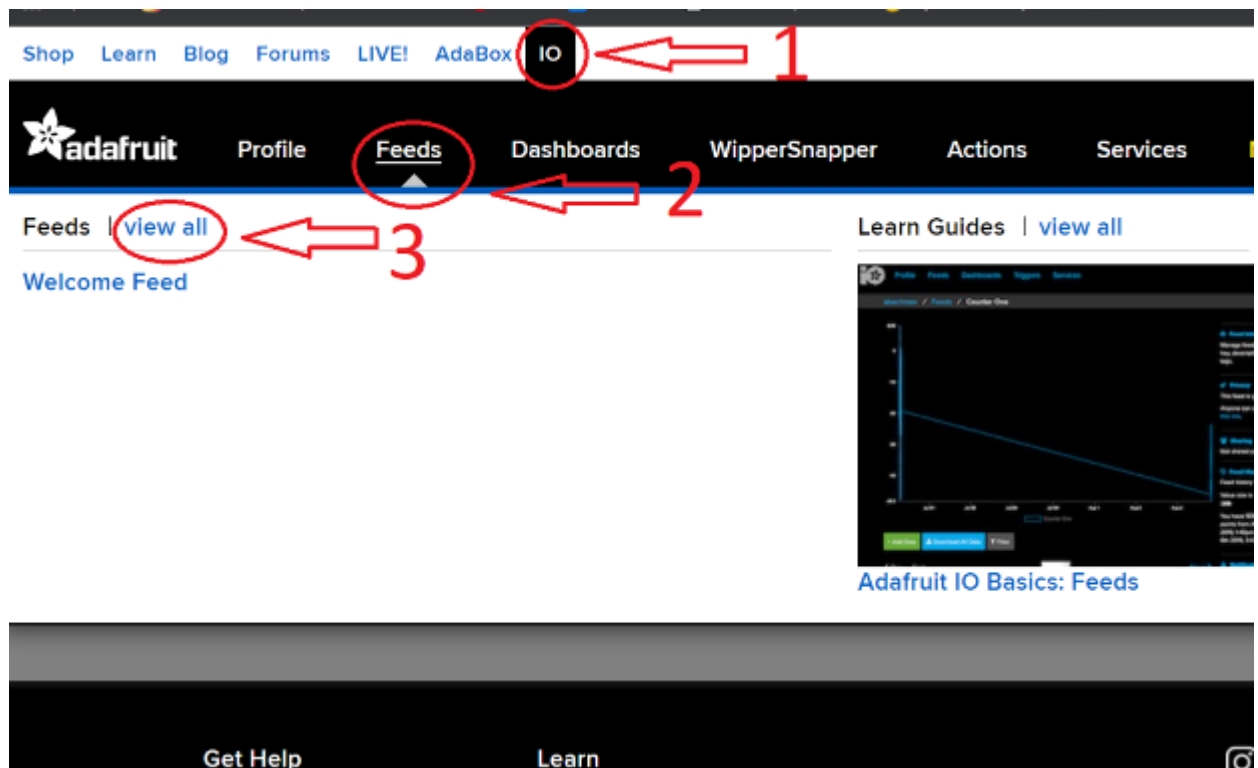


Рисунок 2.14 – Порядок виконання дій для створення каналу

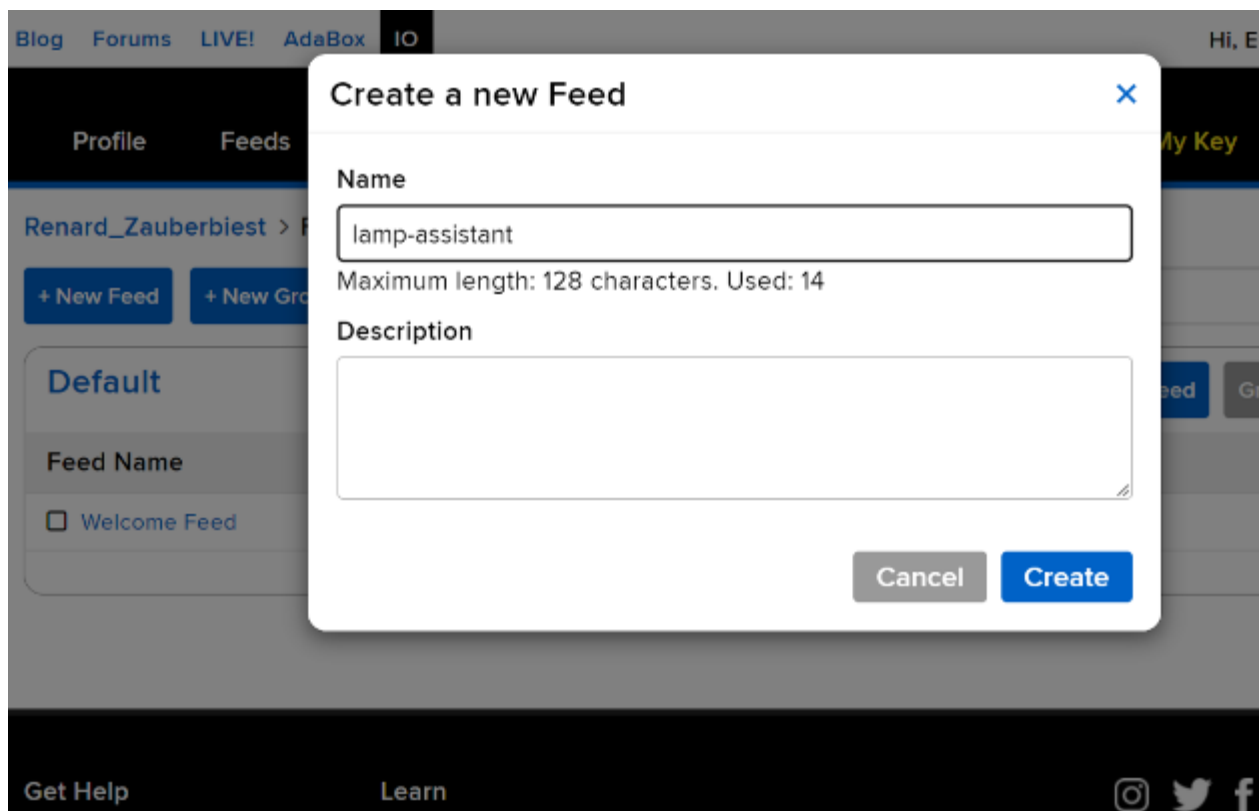


Рисунок 2.15 – Створення каналу

IFTTT [57] – це сервіс, який дозволяє автоматизувати взаємодії різних соцмереж, додатків або систем розумного будинку за допомогою простих правил «Якщо це, то те».

Працює IFTTT за принципом реалізації завдання, так званого «Рецепту», якщо щось відбувається в одному додатку, ця подія запускає дію в іншому додатку [58].

Для початку роботи слід завантажити додаток IFTTT на свій телефон із Play Store (для Android) або App Store (для пристроїв iOS).

Після завантаження програми необхідно створити новий обліковий запис або увійти за допомогою стороннього сервісу (рис. 2.16).

Далі слід обрати «Create» (рис. 2.17).

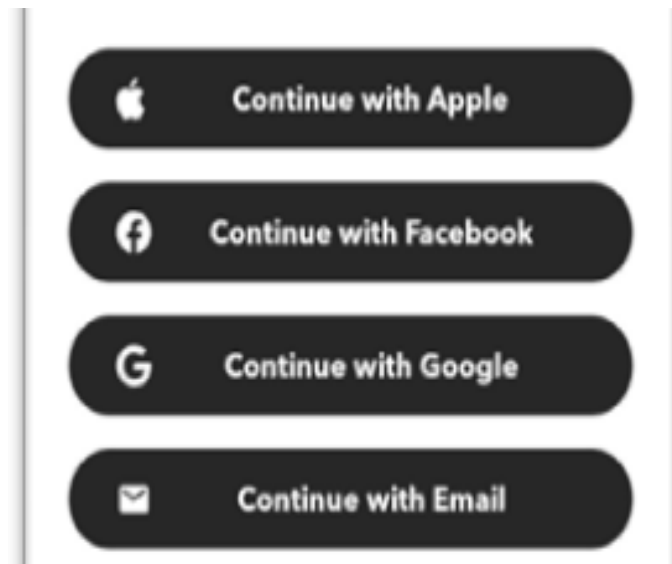


Рисунок 2.16 – Вікно авторизації

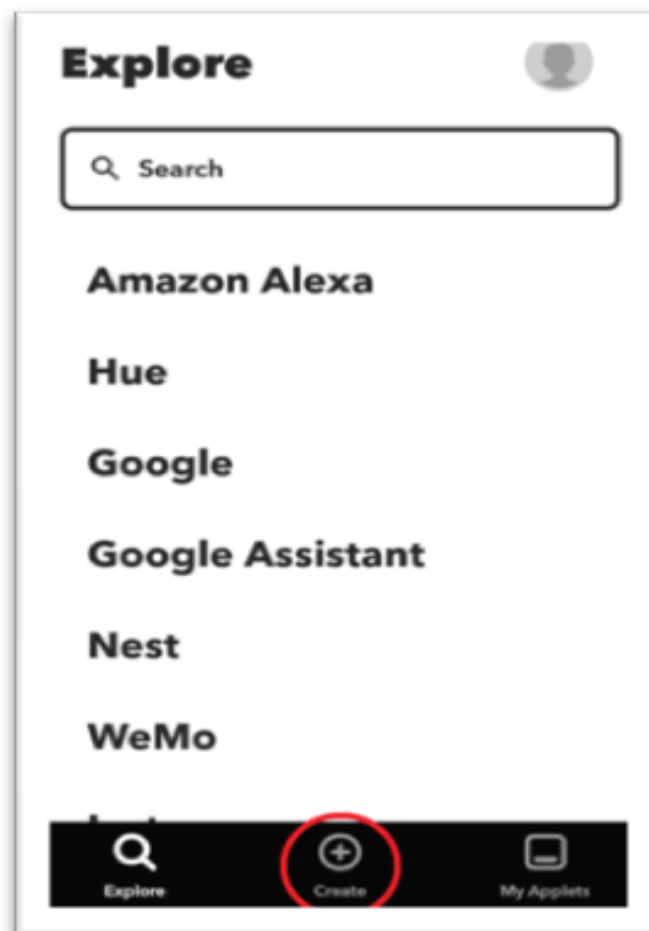


Рисунок 2.17 – Вибір Create

Післ цього потрібно обрати «If This» (рис. 2.18). Відкриється пошук сервісів, тут необхідно знайти «Google Assistant» (рис. 2.19).

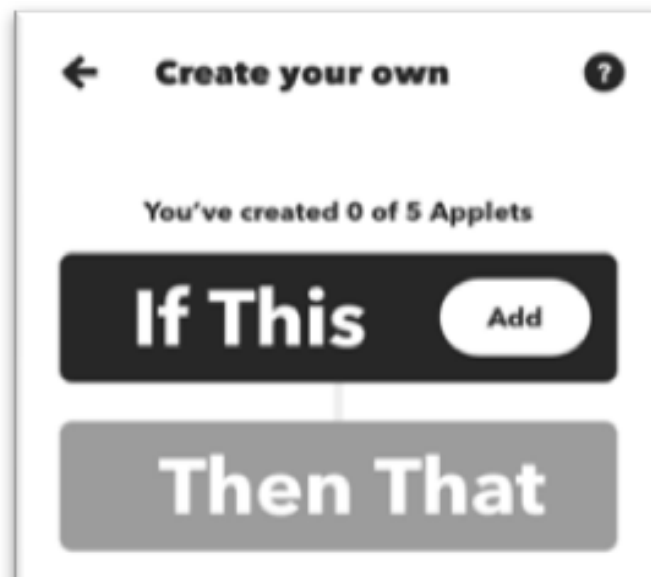


Рисунок 2.18 – Вибір If This

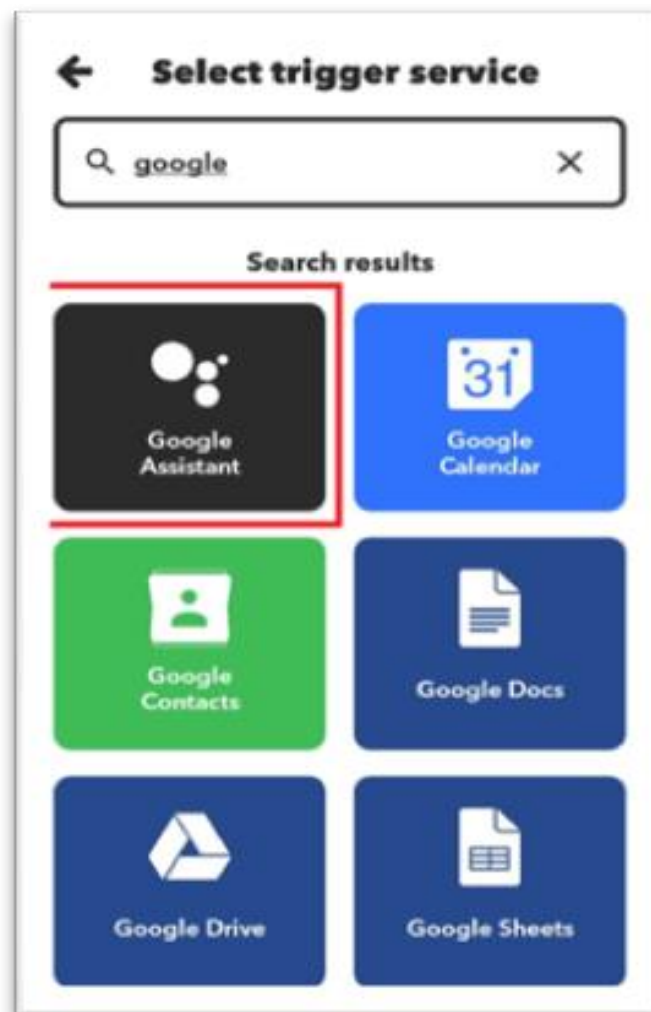


Рисунок 2.19 – Пошук Google Assistant

Далі слід обрати «Say a simple phrase» (рис. 2.20), після чого відкриється форма, яку слід заповнити так як зображено на рисунку 2.21 та натиснути «Continue».



Рисунок 2.20 – Вибір Say a simple phrase

Рисунок 2.21 – Заповнення форми для вмикання лампи

Далі слід обрати «Then That» (рис. 2.22) та у пошуку знайти Adafruit (рис. 2.23), після чого застосунок запропонує увійти у обліковий запис Adafruit, після входу слід надати необхідні дозволи. Тепер можна обрати дію, яку потрібно виконати за допомогою служби Adafruit. Слід обрати «Send data to Adafruit IO» (рис. 2.24). У новому вікні у полі «Feed» name слід обрати «lamp-assistant», а у полі «Data to save» ввести «1» та натиснути «Continue» (рис. 2.25).

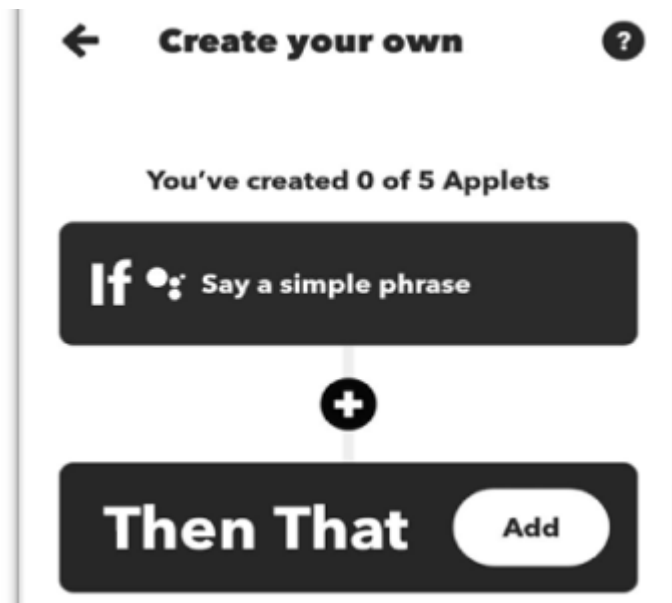


Рисунок 2.22 – Вибір Then That

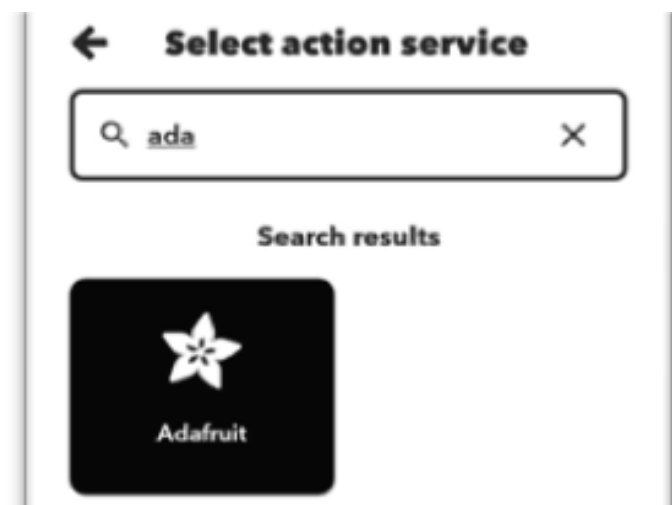


Рисунок 2.23 – Пошук Adafruit

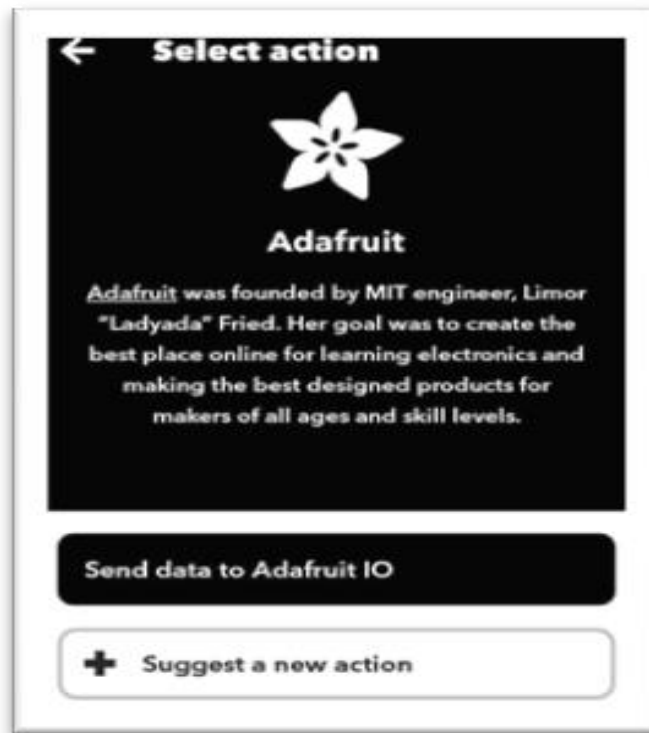


Рисунок 2.24 – Вибір Send data to Adafruit IO

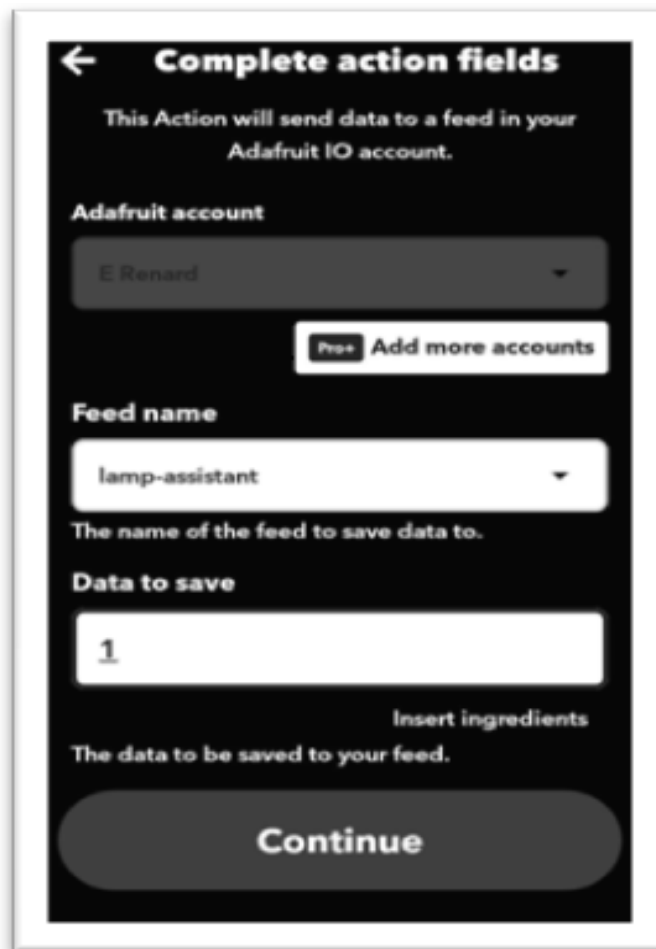


Рисунок 2.25 – Налаштування підключення до Adafruit

Далі необхідно завершити створення правила, натиснувши «Finish» (рис. 2.26).

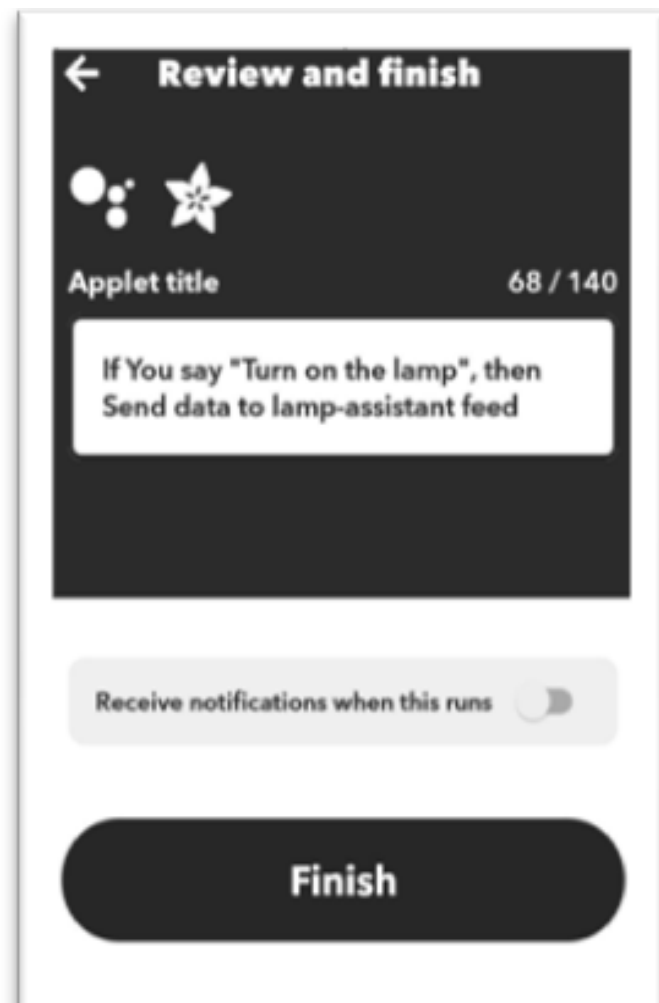


Рисунок 2.26 – Завершення налаштування правила

Тепер необхідно таким же чином створити інше правило, яке прослуховує команду на вимкнення лампи (рис. 2.27) та надсилає «0» як дані до каналу «lamp-assistant» на Adafruit IO (рис. 2.28).

What do you want to say?

Turn off the lamp

What's another way to say it? (optional)

Switch off the lamp

What's another way to say it? (optional)

Switch off the lamp

And another way? (optional)

Turn the lamp off

What do you want the Assistant to say in response?

Switching off the lamp

Language

English

Рисунок 2.27 – Заповнення форми для вимикання лампи

← **Complete action fields**

This Action will send data to a feed in your Adafruit IO account.

Adafruit account

E Renard

Pre Add more accounts

Feed name

lamp-assistant

The name of the feed to save data to.

Data to save

0

Insert ingredients

The data to be saved to your feed.

Рисунок 2.28 – Налаштування підключення до Adafruit

В результаті повинно бути два правила, одне для вмикання лампи, а інше для її вимикання (рис. 2.29).

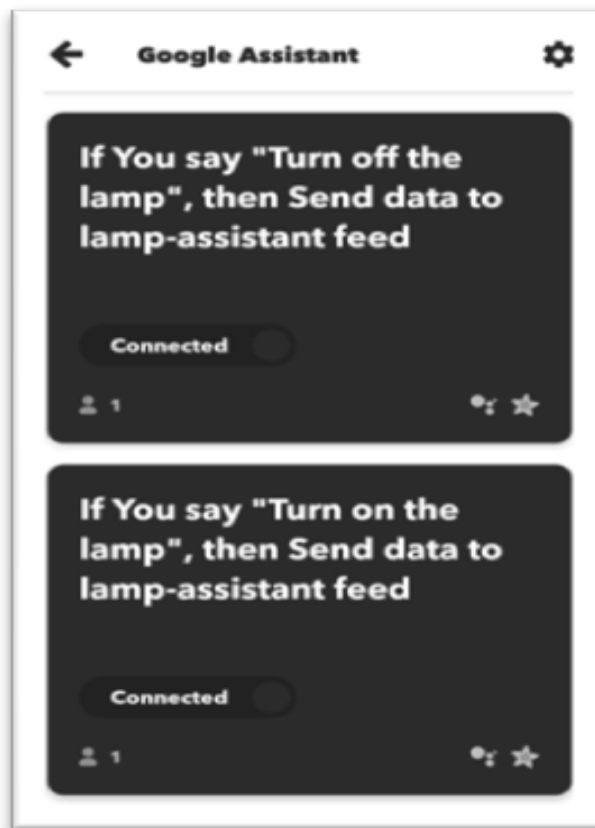


Рисунок 2.29 – Створені правила

Далі у Simulі необхідно створити проєкт з Arduino та обрати модуль W5500 (рис. 2.30).

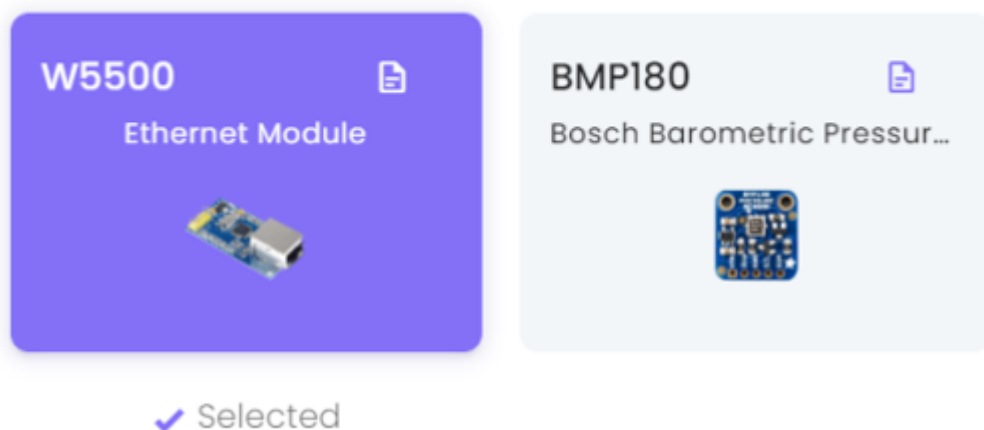


Рисунок 2.30 – Вибір модуля W5500

W5500 — це контролер Ethernet (рис 2.31), за допомогою якого можна надати доступ до Інтернету таким пристроям, як Arduino. Він підтримує TCP, UDP, ICMP, IPv4, ARP, IGMP і PPPoE. Користувачі можуть отримати доступ до 8 незалежних сокетів одночасно [59].

Використання модуля Ethernet дозволяє створювати проекти на основі IoT за допомогою Arduino Uno.



Рисунок 2.31 – Модуль W5500 [59]

Arduino IDE пропонує повне рішення для розробки проектів на основі різних плат мікроконтролерів. Середовище має редактор коду для написання програм, які називаються скетчі, і дозволяє одним натиском миші завантажувати скетчі до мікроконтролерів [60].

У Arduino IDE слід встановити бібліотеку для зв'язку з Adafruit IO. Для цього слід перейти Tools >Manage Libraries, у пошуку ввести Adafruit MQTT, отримавши результати пошуку слід обрати Adafruit MQTT Library та натиснути Install (рис. 2.32). Після завершення установки необхідно закрити вікно менеджера.

Далі слід написати необхідний код, де треба вказати ім'я користувача Adafruit IO, та ключ Adafruit IO. Їх можна подивитись у Adafruit у розділі «My Key» (рис. 2.33).

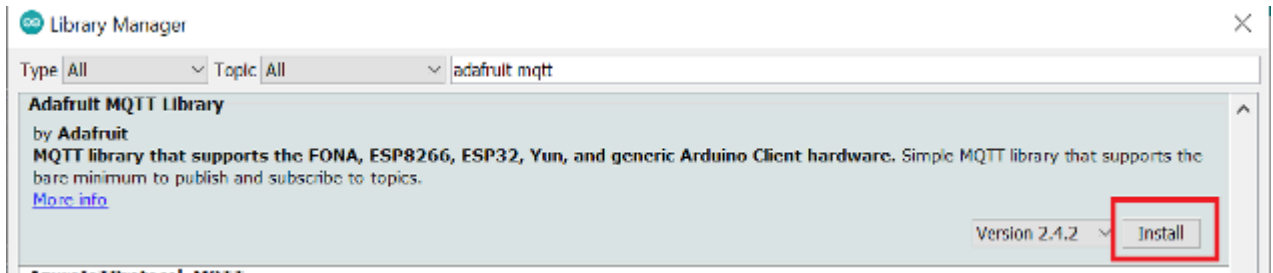


Рисунок 2.32 – Встановлення бібліотеки Adafruit MQTT Library

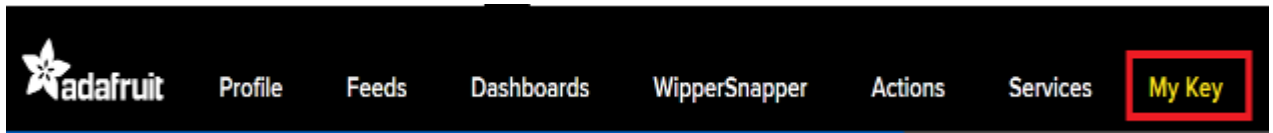


Рисунок 2.33 – Розділ My Key

Далі у Arduino IDE слід обрати Sketch > Export compiled Binary, в результаті створиться бінарний файл, який необхідно завантажити до створеного проєкту.

Для того щоб асистент Google виконував команди для управління лампою, необхідно у налаштуваннях асистенту перейти до пункту «Мови для спілкування з Асистентом» та додати англійську. Результати роботи наведено на рисунках 2.34 – 2.35.

2.2 Віддалений робочий простір REIoT

Комплекс для вивчення технологій IoT і проєктування вбудованих систем REIoT [61] є розробкою кафедри програмних засобів Національного Університету «Запорізька Політехніка». Він складається з двох підсистем: Віддаленої лабораторії (ВЛ) RELDES та ВЛ Smart House & IoT і базується на платформах Arduino, Raspberry Pi та OpenHAB [62].

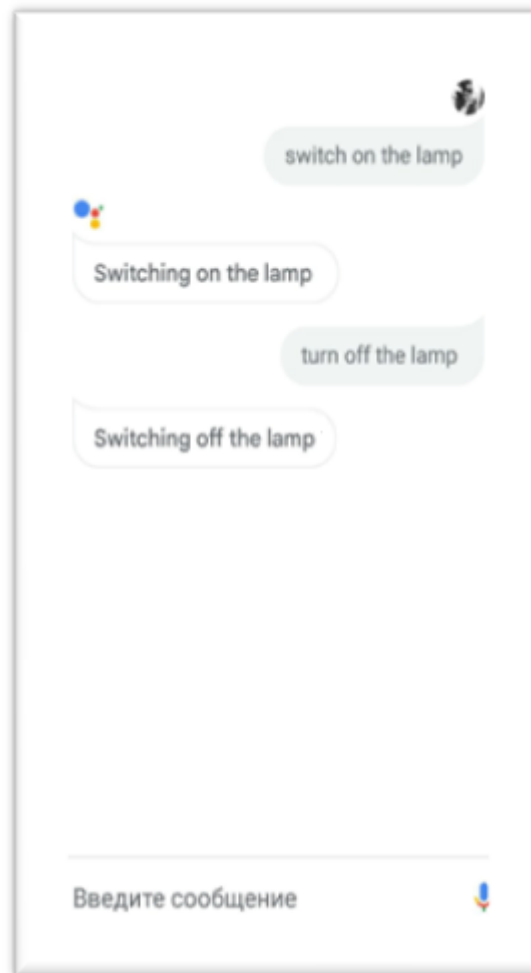


Рисунок 2.34 – Спілкування з Google Assistant

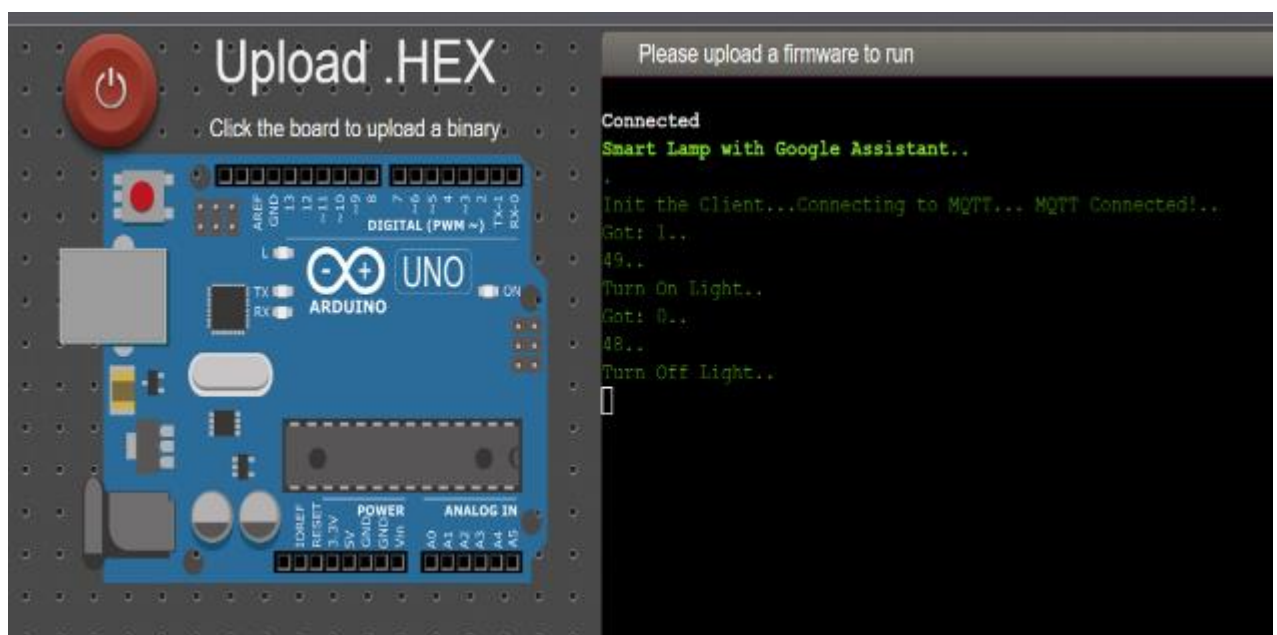


Рисунок 2.35 – Виконання команд від Google Assistant

RELDES (рис. 2.36) надає можливість тестування і швидкого створення прототипів вбудованих систем. На сьогоднішній день реалізовано 4 експерименти на платах Arduino та керуючому обладнанні (виконавчі пристрої, ультразвуковий датчик відстані, світлофор і рідкокристалічний дисплей). Всі специфікації, схеми експериментів і програмний код (шаблони для використання експериментів) доступні на сайті ВЛ [63].

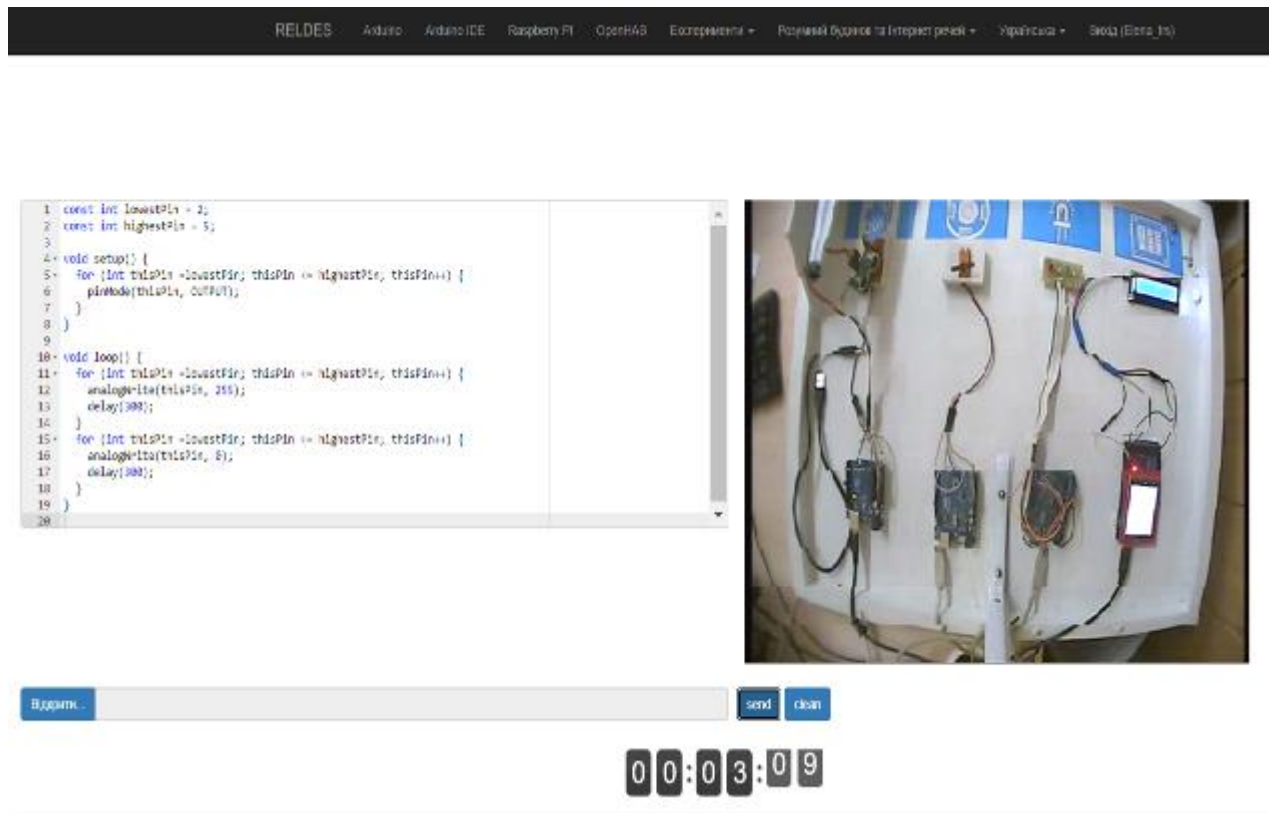


Рисунок 2.36 – Вебінтерфейс експерименту ВЛ RELDES

ВЛ Smart House&IoT (рис. 2.37) - програмно-апаратна платформа для дослідження особливостей розробки систем «Розумний будинок». Вона дозволяє практично вивчати різноманітні платформи, датчики, виконавчі механізми, протоколи та інтерфейси для впровадження технологій IoT.

Smart House & IoT має декілька наборів експериментів. До першого набору можна віднести набір експериментів для вивчення питань ресурсозбереження та енергоефективності.



Рисунок 2.37 – Вебінтерфейс експерименту ВЛ Smart House&IoT [62]

Вони дозволяють контролювати параметри систем освітлення, опалення, кондиціонування, вентиляції та рекуперації, а також виробництва сонячної енергії.

До другого набору можна віднести набір для вивчення питань безпеки, що дозволяє контролювати доступ до контрольованих зон і об'єктів.

До третього набору можна віднести набір для вивчення питань кібербезпеки, які дозволяють використовувати бездротові, хмарні та мобільні технології для моніторингу та контролю системи. [62].

В результаті модернізації ВЛ Smart House&IoT з'явився новий експеримент «Розумна теплиця». Він має декілька датчиків таких як:

- водонепроникний температурний датчик DS18B20 для зчитування температури ґрунту;

- емнісний датчик вологості ґрунту для зчитування вологості ґрунту;
- датчик вологості та температури GY-21 HTU21 для зчитування температури та вологості повітря в теплиці;
- датчик освітленості ТЕМТ6000 для зчитування показань освітленості.

Даний експеримент є дуже новим та не має власної сторінки візуалізації, тому розроблене ПЗ у майбутньому буде використовуватись саме для нього.

2.3 Висновки до розділу 2

Використання віртуального та віддаленого інструментарію при прототипуванні IoT систем дозволяє набагато швидше створювати та досліджувати прототипи, а також економити на обладнанні. Завдяки таким сервісам як Wokwi та Simuli, користувач може створювати та кодувати проєкти майже з любого місця, де є комп'ютер та доступ до мережі Інтернет. При використанні подібних сервісів набагато легше відстежувати помилки, які можуть виникнути при реалізації проєктів, та запобігти їх появі при роботі з реальним обладнанням.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз вимог

Одним із перших та головних етапів у проєктуванні ПЗ є аналіз вимог. Виконавши огляд існуючих IoT сервісів (розд. 1) та аналіз віртуального та віддаленого інструментарію для прототипування IoT систем (розд. 2), було розроблено списки функціональних та нефункціональних вимог, які представлено на рисунку 3.1.

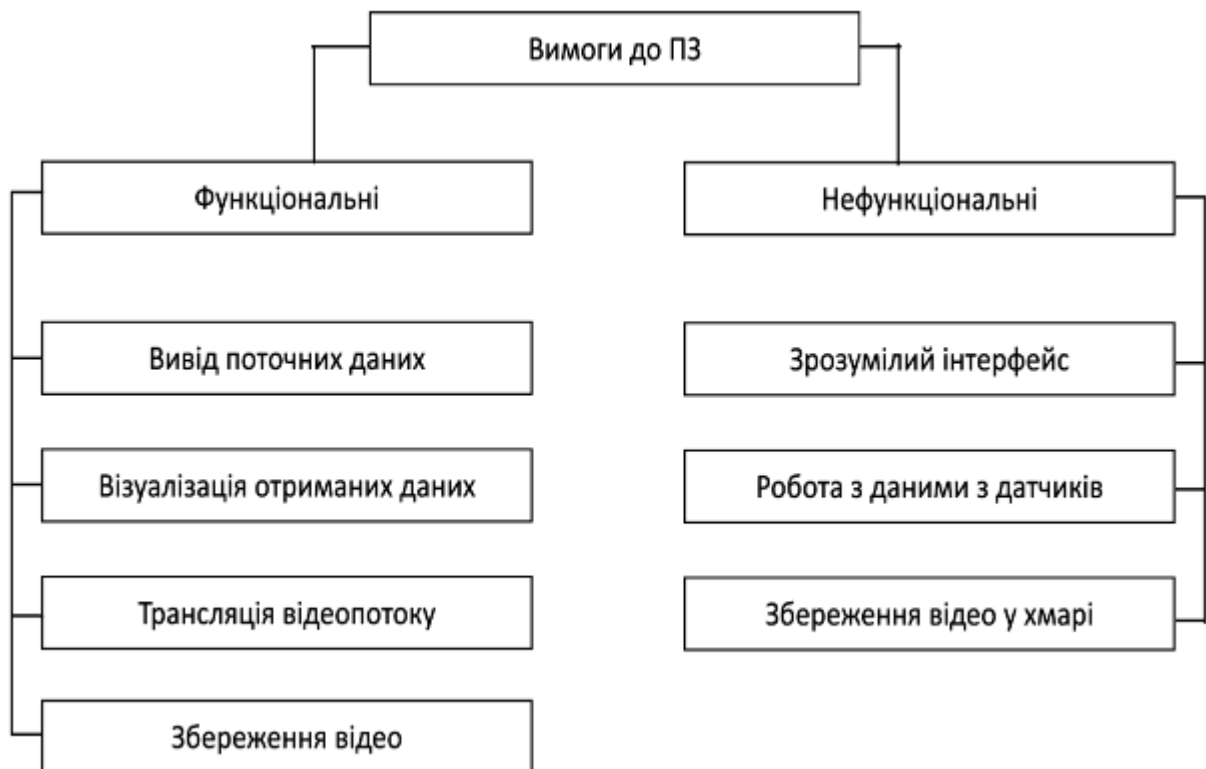


Рисунок 3.1 – Вимоги до ПЗ

При переході на сторінку візуалізації користувач повинен бачити поточні дані з датчиків, які повинні змінюватись в режимі реально часу. Для забезпечення візуалізації кожна зміна даних повинна бути зафіксована на графіку, тобто графіки з даними теж повинні оновлюватись в режимі реального часу. Також користувач повинен візуально спостерігати за датчиками в режимі реального часу щоб переконатись, що ніякі непередбачувані ситуації не заважають коректній роботі датчиків, тому на

сторінці необхідно забезпечити трансляцію відео, оскільки спостерігати постійно за датчиками неможливо, то система повинна зберігати відео. Зберігання відео потребує багато ресурсів, тому краще всього забезпечити зберігання у хмарі.

3.2 Розробка архітектури системи

При реалізації системи, архітектуру якої наведено на рисунку 3.2, виникла задача збереження великих об'ємів даних. Для цього доцільно використовувати хмарні сховища, однак більшість з них є платними. Тому було прийнято рішення використовувати хмарний FTP сервіс, такі сервіси зазвичай дуже дешеві та багато з них мають безкоштовні версії.

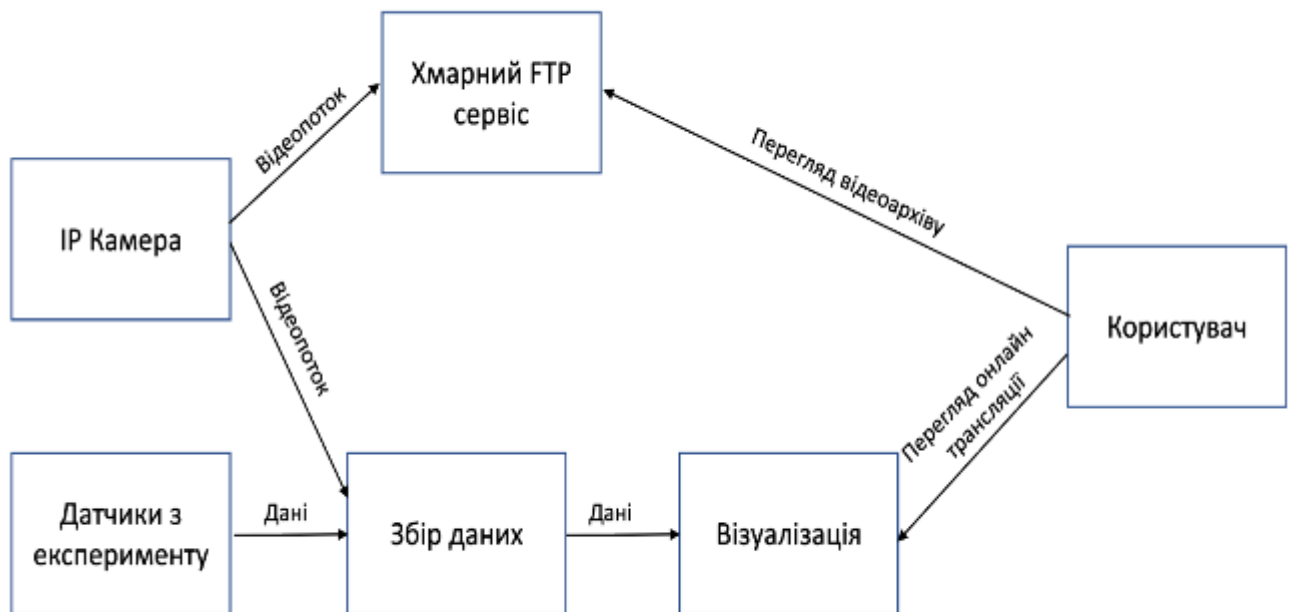


Рисунок 3.2 – Архітектура системи

На рисунку 3.3 зображено діаграму послідовності. Робота IoT сервісу повинна відбуватись наступним чином. Після завантаження коду до мікроконтролеру відбувається передача даних з датчиків до серверної

частини сервісу. Серверна частина приймає дані з датчиків та відеопоток з камери. Користувач відвідує сторінку візуалізації, сторінка відправляє запити до серверу на отримання даних та зображень з камери, сервер відправляє дані та зображення з камери на сторінку візуалізації. Сторінка виводить поточні дані та візуалізує їх у вигляді графіків, та транслює відео в режимі реального часу.

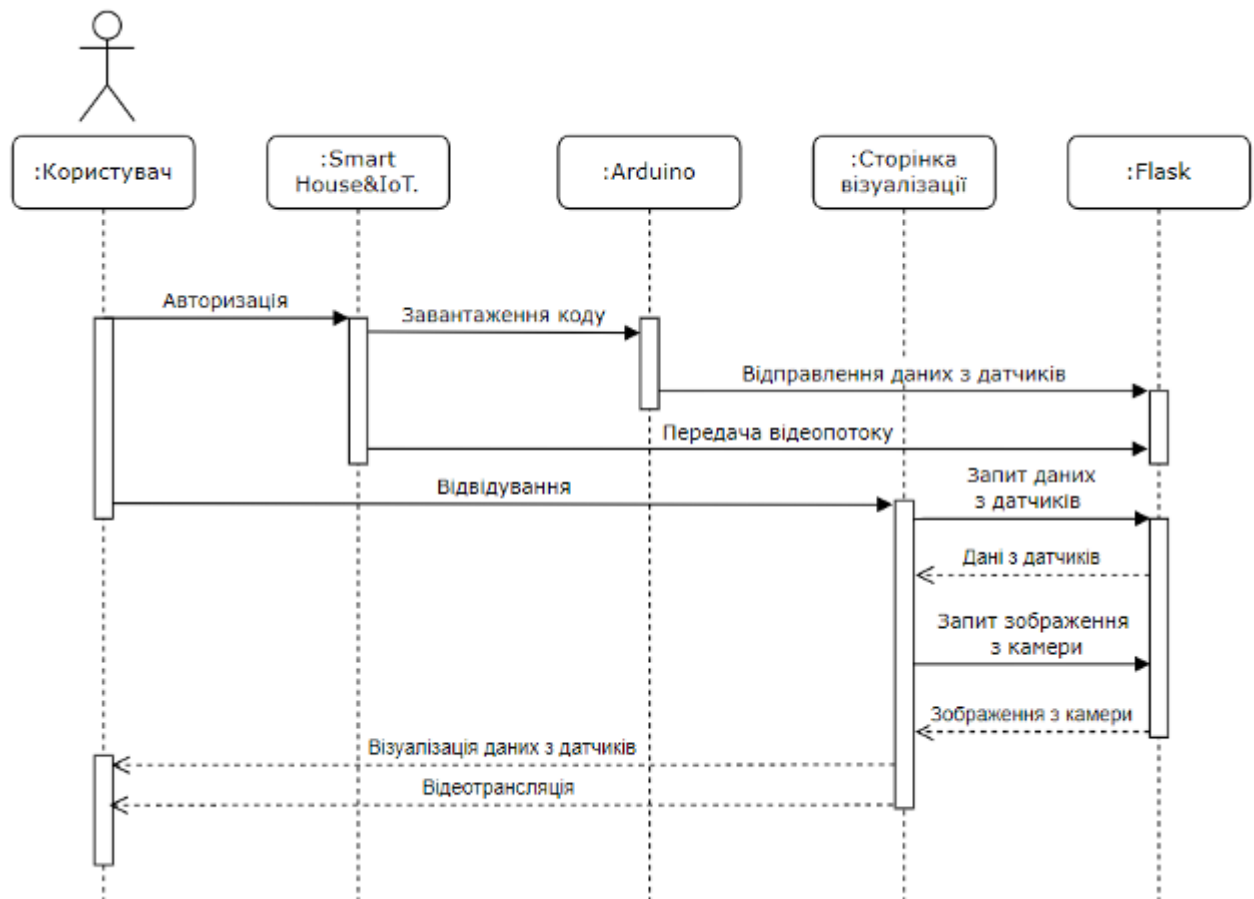


Рисунок 3.3 – Діаграма послідовності роботи IoT сервісу

3.3 Удосконалений метод роботи з відеопотоком

Однією з вимог до IoT сервісу є трансляція та зберігання відео. Найчастіше записане відео займає багато місця, а для того щоб зрозуміти, що сталося достатньо серії статичних знімків. Отже замість збереження відео можна використовувати збереження так званих подій. Подія - це група фотографій, знятих в один часовий інтервал, який зазвичай можна

налаштувати. Наприклад, якщо інтервал дорівнює шестидесяти секундам і за цей час було зроблено двадцять знімків, то вони будуть вважатися однією подією.

Отже збереження трансляції можна реалізувати наступним чином, кожні декілька секунд робити знімок з камери та зберігати його, при цьому збереження можна робити декількома способами, одразу відправляти зображення до FTP сховища або зберігати локально та раз на добу вивантажувати знімки до сховища. FTP сервіси зазвичай дешеві та деякі з них мають безкоштовні версії, а більшість сучасних відеопрогравачів, в яких можна указати FTP, можуть відтворити відео із збережених зображень.

Термін FTP відноситься до процесу, який включає передачу файлів між пристроями по мережі. Процес працює, коли одна сторона дозволяє іншій надсилати або отримувати файли через Інтернет. Спочатку використовувався користувачами як спосіб спілкування та обміну інформацією між двома фізичними пристроями, тепер він широко використовується для зберігання файлів у хмарі [64].

FTP зробив обробку даних через Інтернет набагато простіше та інтуїтивно зрозуміліше. Завдяки FTP та його пізнішим ітераціям користувачі можуть легко транслювати відеоконтент, використовувати відеодзвінки, грати в онлайн-ігри, обмінюватися файлами або користуватися хмарними сховищами [64].

FTP-сервер для IP-камер – це варіант онлайн-сховища для камер відеоспостереження, який допомагає користувачам завантажувати та зберігати відео та зображення з камер відеоспостереження на мережній платформі. Такі сервери забезпечують резервне копіювання важливих даних, передачу великих файлів та дозволяють запланувати час передачі даних [65]. Схему удосконаленого методу роботи з відео наведено на рисунку 3.4.

У роботі використовувався сервіс Camera-ftp [66]. Цей сервіс має безкоштовну версію, яка надає два гігабайти пам'яті, дозволяє підключити дві камери та зберігає файли впродовж п'яти днів.

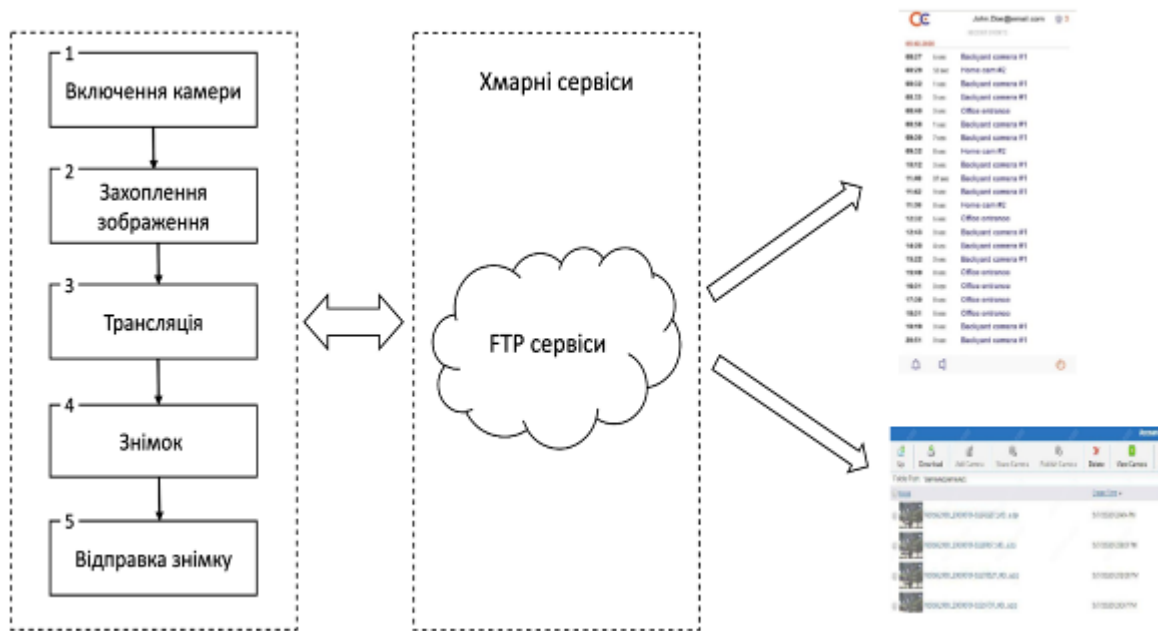


Рисунок 3.4 – Удосконалений метод роботи з відеопотоком

3.4 Вибір інструментарію розробки

3.4.1 Вибір мови програмування

На сьогоднішній день існує досить велика кількість мов для розробки вебзастосунків такі як PHP, Python, Ruby, C# та багато інших.

Для виконання дипломної роботи було обрано мову Python. Python — це популярна мова програмування загального призначення, яку можна використовувати для нарізноманітніших програм. Вона була розроблена наприкінці 1980-х років Гвідо Ван Россумом. Ван Россум змодельював Python за зразком англійської мови, виключивши непотрібний синтаксис, щоб спростити читання та запис, програми [67].

На сьогоднішній день популярність Python продовжує рости, це пов'язано з багатьма факторами.

По-перше, Python — це мова з відкритим кодом, що означає, що вихідний код відкритий для будь-кого, хто може змінити його за своїм бажанням, а за використання мови не потрібно платити. Кожен може вивчити

та використовувати його для створення чогось, навіть компанії можуть отримати його безкоштовно. По-друге, Python надзвичайно простий у використанні завдяки великій підтримці та функціям, що легко інтегруються. Мова програмування проста для розуміння та читання, що полегшує новачкам початок її вивчення [68]. По-третє, широкий спектр застосування. Python використовується у аналізі даних та машинному навчанні, веб розробці, автоматизації або скриптингу, тестування програмного забезпечення та прототипування тощо [69].

3.4.2 Вибір фрейворку

Завдяки великому підтримуючому співтовариству, Python має безліч чудових бібліотек та фреймворків для кожної сфери застосування. Оскільки фреймворки дуже добре полегшують та пришвидчують розробку, наступним кроком у розробці був вибір фреймворку. Найпопулярнішими фреймворками для веб-розробки на мові Python є Django, Flask та FastAPI.

Django – це веб-фреймворк Python, створений Адріаном Головаті та Саймоном Віллсоном у 2003 році. Він слідує шаблону Model-Template-View. Головною метою створення Django було спрощення процесу розробки. Розробники хотіли мати можливість створювати веб-програми та запускати їх за лічені години, а не дні. Однією з причин популярності Django є його простота і той факт, що він безкоштовний та з відкритим вихідним кодом [70].

Flask — мікрофреймворк на основі Python, який широко використовується для розробки веб-додатків. Причина, через яку Flask класифікується як мікрофреймворк, полягає у тому, що він не залежить від зовнішніх бібліотек для виконання завдань фреймворку. Flask має свої інструменти, бібліотеки та технології, які допомагають підтримувати всі функції розробки веб-додатків [71].

FastAPI – це сучасна, швидка та надійна платформа, яка допомагає створювати API з версіями Python 3.6 і вище. FastAPI вважається одним із найшвидших фреймворків Python, який швидко кодується з меншою кількістю помилок, спричинених розробниками [70].

Порівняльну характеристику фреймворків наведено у таблиці 3.1. Оскільки Flask дуже добре підходить для односторінкових додатків та не має залежності від зовнішніх бібліотек, що робить його досить легким та простим у застосуванні було вирішено вибрати саме його.

Таблиця 3.1 – Порівняння фреймворків [70]-[71].

Характеристики	Django	Flask	FastAPI
Тип	Full Stack фреймворк	Легкий мікрофреймворк, який пропонує кілька функцій без зовнішніх бібліотек	Високопродуктивний вебфреймворк для створення API
Застосування	Підходить для багатосторінкових додатків та великих проєктів	Підходить для односторінкових додатків та невеликих проєктів	Підходить для багатосторінкових додатків та великих проєктів
Гнучкість	Має безліч вбудованих функцій, через що розробники не можуть вносити зміни до модулів	Неймовірно гнучкий, оскільки не має вбудованих розширень.	Гнучкість забезпечується моделями Pydantic, при цьому код залишається простим та коротким
Контроль	Розробники не мають повного контролю над функціями та модулями	Дозволяє повністю контролювати створення програми, незалежно від будь-яких зовнішніх бібліотек	Дозволяє повністю контролювати створення програми, незалежно від будь-яких зовнішніх бібліотек
Відладчик	Не підтримує віртуальну відладку	Має вбудований відладчик	Має вбудований відладчик

3.4.3 Вибір середовища розробки

Код на мові Python можна писати у будь-якому текстовому редакторі, але для більшої зручності та продуктивності написання програми краще використовувати IDE для Python. На сьогоднішній день існує дуже багато різноманітних IDE, серед найпопулярніших з них можна виділити Pycharm, PyDev та Wing.

PyCharm - одне з найпопулярніших IDE для Python, розроблене компанією JetBrains. Завдяки інструментам підвищення продуктивності, таким як швидкий та безпечний рефакторинг коду, PyCharm справді виділяється на тлі конкурентів. Він доступний у трьох версіях: Community версія з ліцензією Apache, Educational версія для освітніх закладів та Professional версія. Перші дві версії мають відкритий вихідний код та безкоштовні, але за версію Professional доведеться заплатити.

Безкоштовна версія Community пропонує безліч функцій, таких як підсвічування синтаксису, автозаповнення та перевірка коду в реальному часі. Платна версія Professional пропонує розширені функції, такі як повне керування базою даних, а також низку конкретних платформ та інструментів веб-розробки [72]. Інтерфейс IDE Pycharm наведено на рисунку 3.5.

PyDev – зовнішній плагін для Eclipse. По суті це IDE, яке використовується для розробки Python. В основному він фокусується на рефакторингу коду Python, налагодженні графічного шаблону, аналізі коду тощо. Це робить його сильним інтерпретатором Python.

Оскільки це модуль, що підключається для середовища Eclipse, розробники можуть більш гнучко використовувати IDE для розробки додатків. Eclipse є повністю безкоштовним середовищем [73]. Інтерфейс IDE PyDev наведено на рисунку 3.6.

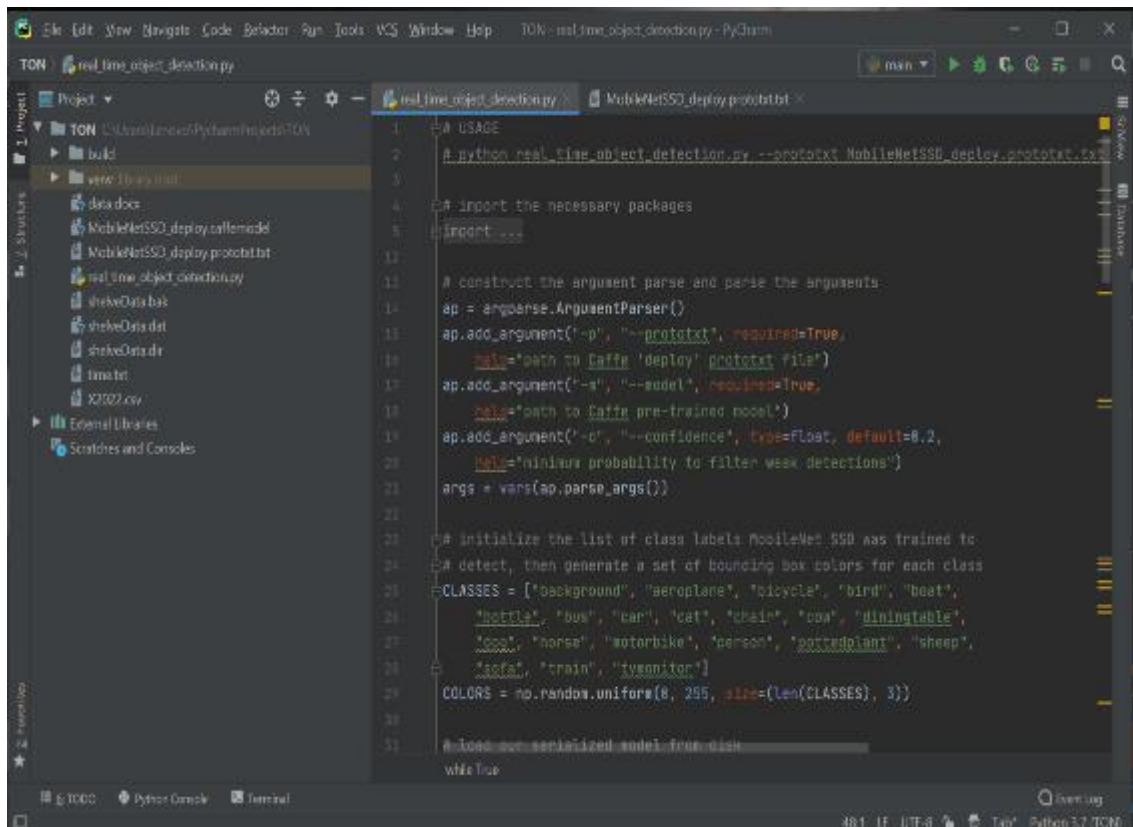


Рисунок 3.5 – Интерфейс IDE Pycharm

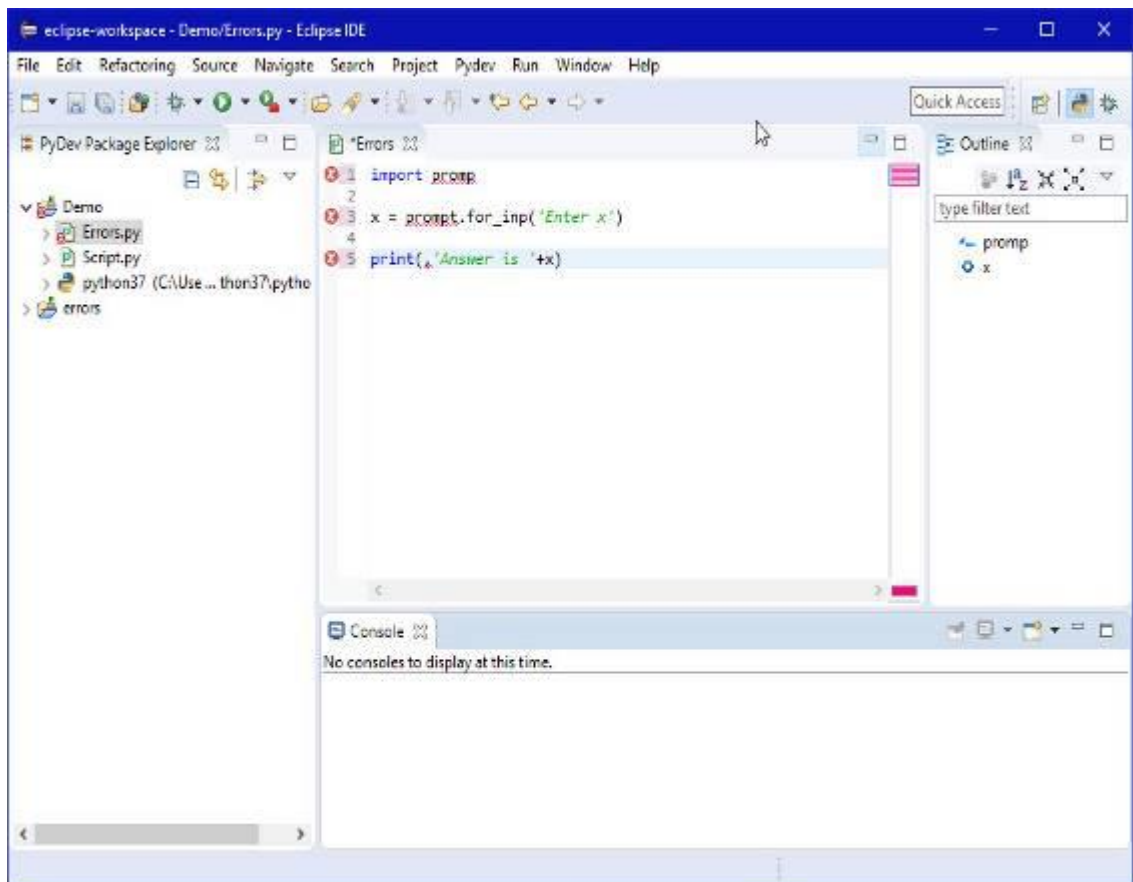


Рисунок 3.6 – Интерфейс IDE PyDev [74]

Wing Python IDE створено спеціально для розробки на Python. Воно випускається у трьох версіях: 101, Personal та Pro. 101 — це спрощена безкоштовна версія з мінімалістичним налаштуванням, а також редактором та функціями пошуку. Версія Personal теж безкоштовна та включає повнофункціональний редактор, а також обмежені функції перевірки коду та управління проєктами. Wing Pro комерційна версія, яка пропонує всі функції безкоштовних версій, а також віддалену розробку, модульне тестування, рефакторинг та багато іншого [75]. Інтерфейс Wing Python IDE наведено на рисунку 3.7.

Кожне з IDE має певні переваги та недоліки, що наведені в таблиці 3.2. Проаналізувавши всі «за» та «проти» було прийнято рішення використовувати IDE PyCham, через його стабільність та зручний інтерфейс.

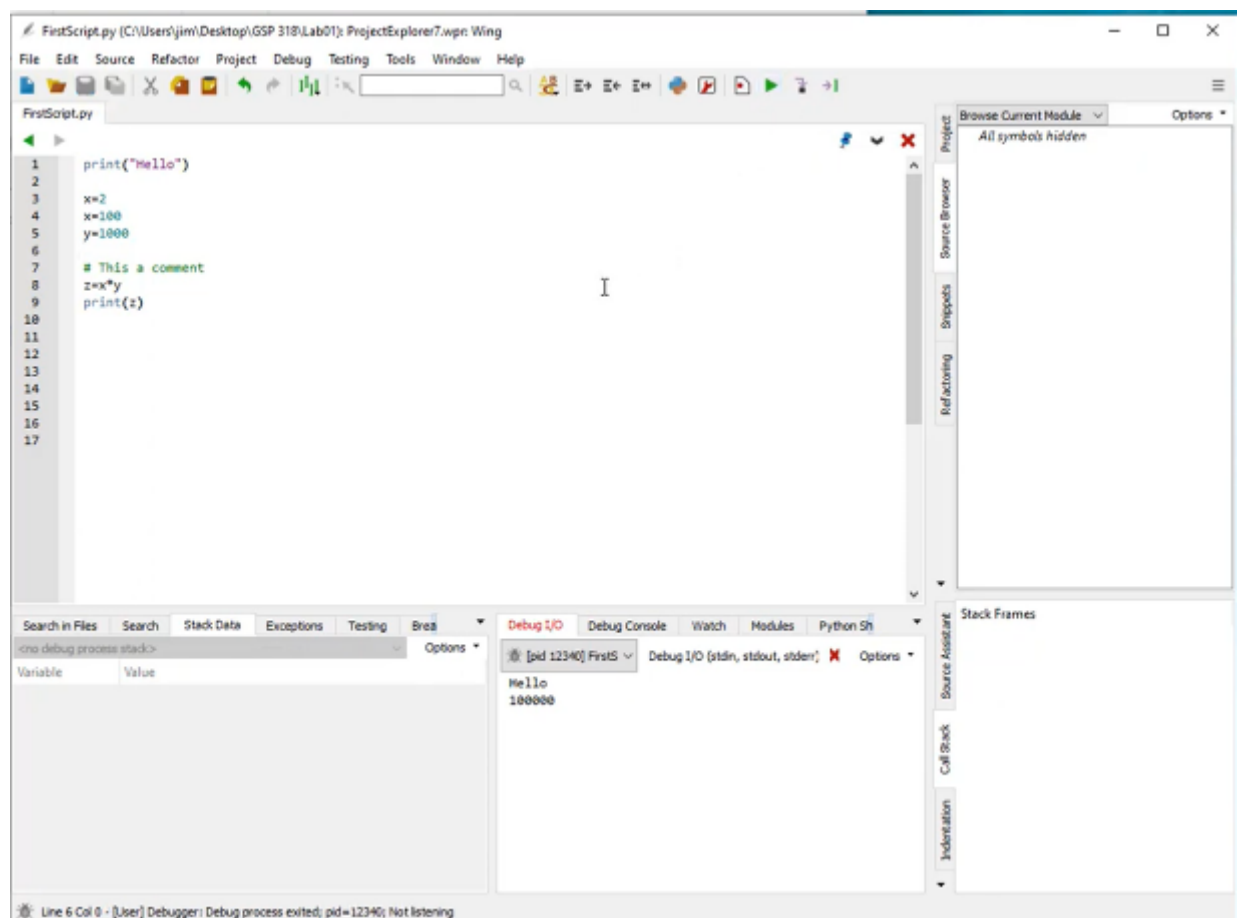


Рисунок 3.7 – Інтерфейс Wing Python IDE [76]

Таблиця 3.2 – Порівняльний аналіз середовищ розробки [72]-[73], [75]

Середовище	Переваги	Недоліки
PyCharm	Підтримка фреймворків таких як Django, Flask тощо. Підсвічування коду. Легкий імпорт бібліотек. Автозавершення коду. Наявність безкоштовної версії. Простий процес установки та налаштування. Вбудований локальний термінал. Кросплатформність.	Займає більшу кількість пам'яті та потребує більше ресурсів у порівнянні з іншими IDE Дуже висока ціна за професійну версію
PyDev	Підтримка фреймворку Django. Підсвічування коду. Автозавершення коду. Повністю безкоштовний. Кросплатформність.	Іноді плагіни PyDev стають нестабільними, створюючи проблеми при розробці програми. Продуктивність PyDev IDE знижується, якщо програма занадто велика з кількома плагінами.
Wing Python IDE	Наявність безкоштовної версії. Автозавершення коду. Підсвічування коду. Підтримка фреймворків Django, Flask тощо. Кросплатформність.	Дуже висока ціна за професійну версію. Не дуже привабливий інтерфейс користувача для новачків.

3.5 Висновки до розділу 3

Виконавши аналіз вимог до проєктованого IoT сервісу, було розроблено його архітектуру.

Розроблено удосконалений метод роботи з відеопотоком, особливістю якого є збереження серії знімків з камери з подальшою відправкою на хмарний FTP-сервер, що дозволяє зменшити обсяги даних за умови отримання достатньої інформації для реалізації контрольних функцій IoT сервісу.

Для програмної реалізації було обрано мову Python через її простоту, популярність та широкі можливості. Для полегшення розробки було прийнято рішення використовувати спеціальний фреймворк, для чого було проаналізовано декілька фреймворків та серед них обрано Flask. Оскільки для розробки програм доцільно використовувати IDE, було оглянуто декілька IDE, які мають безкоштовні версії, та серед них було обрано PyCharm.

4 КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Алгоритм трансляції та зберігання відео

Схему алгоритму роботи з відео з використанням FTP сервісу наведено на рисунку 4.1.

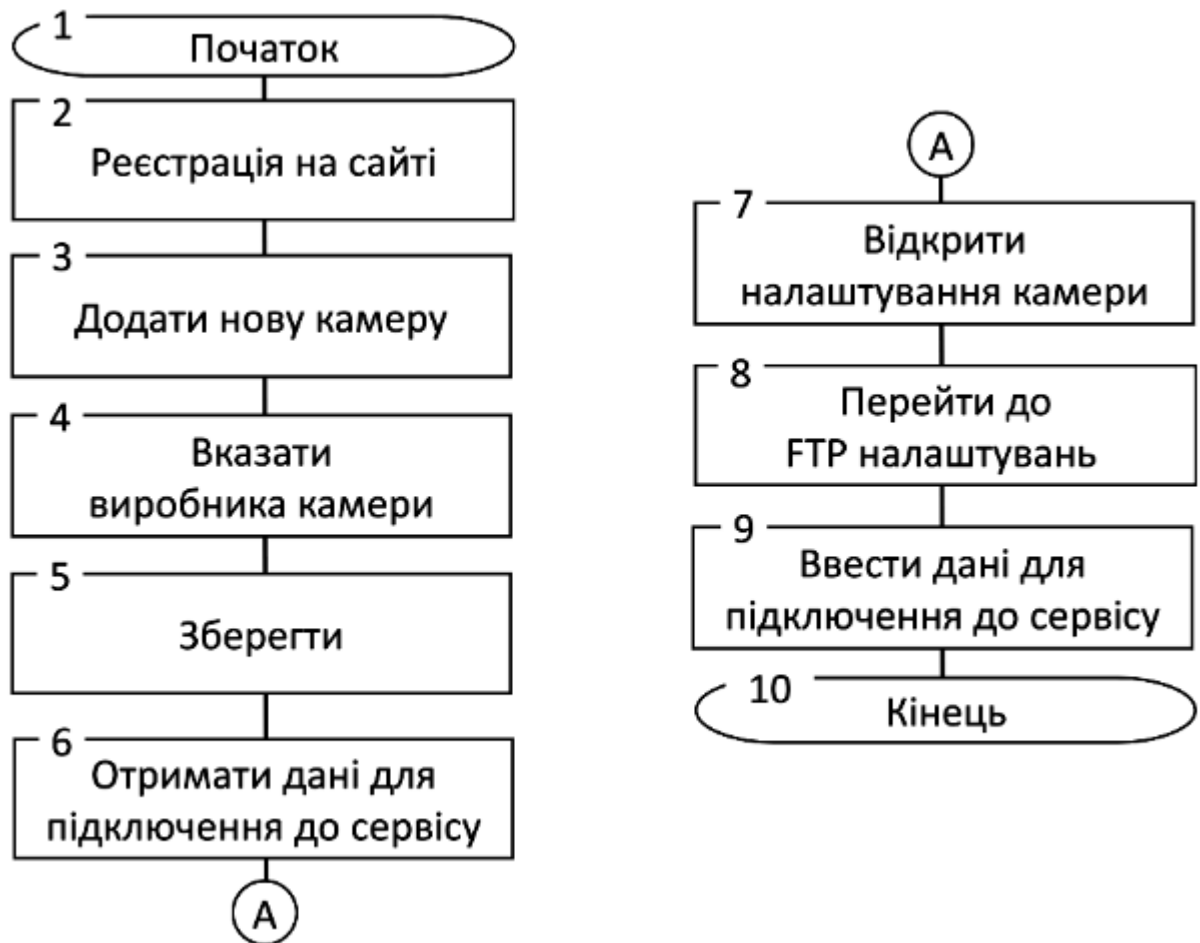


Рисунок 4.1 - Схема алгоритму роботи з відео з використанням FTP сервісу

Для підключення камери до сервісу необхідно зареєструватися на сайті та додати нову камеру, далі слід вказати назву та виробника камери (рис. 4.2) та натиснути «save». Після цього сервіс надасть дані для FTP налаштування камери (рис. 4.3).

Camera label

Cam1

Camera brand and model

HikVision

SAVE

CANCEL

Home | Account

Рисунок 4.2 - Додавання камери до сервісу Camera-ftp

FTP Configuration

FTP Serverftp1.camera-ftp.com

FTP Server IP77.120.117.18

FTP port21

Transfer modePassive

FTP usernamelitexewo

FTP password

[media log_\(for debug.purpse\).](#)

[delete camera](#)

GO TO CAMERA LIST

Рисунок 4.3 - Дані для FTP налаштування

Далі у налаштуваннях камери в розділі FTP слід ввести отримані дані з сервісу (рис. 4.4).

Для забезпечення трансляції відео на веб-сторінці, в першу чергу необхідно підключитись до камери. Для роботи з камерою використовується бібліотека OpenCV. OpenCV - це величезна бібліотека з відкритим вихідним кодом для комп'ютерного зору, машинного навчання та обробки зображень.

The screenshot shows the 'FTP' configuration tab in a camera's settings menu. The fields and their values are as follows:

- FTP Protocol:** FTP (selected from a dropdown)
- Server Address:** 77.120.117.18 (with a green checkmark)
- Port:** 21 (with a green checkmark)
- User Name:** litexwo (with a green checkmark and an 'Anonymous' checkbox that is unchecked)
- Password:** [masked with dots] (with a green checkmark)
- Confirm:** [masked with dots] (with a green checkmark)
- Directory Structure:** Save in the parent directory (selected from a dropdown)
- Parent Directory:** Custom (selected from a dropdown) and DemoCamera (entered in the adjacent text field, with a green checkmark)
- Picture Filing Interval:** 1 (selected from a dropdown) Day(s)
- Picture Name:** Default (selected from a dropdown)
- Upload Picture:** [checked checkbox]
- Test:** [button]
- Save:** [red button with a save icon]

Рисунок 4.4 - Налаштування камери

Для захоплення зображення з камери, змінній «source» (рис. 4.5) необхідно надати значення RTSP адреси камери у вигляді «rtsp://USERNAME:PASSWORD@IP:PORT/STREAM», де:

- USERNAME - ім'я користувача або логін у камері;
- PASSWORD - пароль користувача в камері;
- IP - локальна IP-адреса камери;
- PORT - RTSP порт;
- STREAM – якість відео.

```
source = "rtsp://camera_test:gy78skY5SP3@192.168.1.10:554/stream2"
camera = cv2.VideoCapture(source)
```

Рисунок 4.5 - Захоплення зображення з камери

Для трансляції відео на веб-сторінці необхідно реалізувати потокову передачу відповідей. Flask забезпечує вбудовану підтримку потокової передачі відповідей за рахунок використання функцій генератора. Генератор – це спеціальна функція, яка не зберігає усі значення в пам'яті, вона генерує значення на льоту. У мові Python ключове слово "yield" використовується як "return", за винятком того, що функція повертає генератор. Функція-генератор може послідовно повертати кілька результатів. Flask використовує цю характеристику функції-генератора для реалізації потокової передачі.

У таблиці 4.1 наведені функції бібліотеки OpenCV, які використовувались для забезпечення захоплення та обробки відео. Розроблену функцію `generate_frames()`, яка обробляє зображення з камери, наведено на рисунку 4.6.

Таблиця 4.1 - Функції бібліотеки OpenCV, що використовувались для забезпечення захоплення та обробки відео

Назва функції	Властивості
<code>VideoCapture</code>	Функція відкриває відеофайл або послідовність файлів зображень або пристрій захоплення чи IP-потік відео для захоплення відео.
<code>read</code>	Функція повертає логічне значення True/False. Якщо кадр читається правильно, то це буде True.
<code>imencode</code>	Функція стискає зображення, кодує та зберігає його у буфері пам'яті, розмір якого змінюється відповідно до результату
<code>tobytes</code>	Функція перетворює закодоване зображення у байти.

Маршрут /video повертає потокову відповідь (рис. 4.7). Оскільки цей потік повертає зображення, які мають відображатися на веб-сторінці, то в HTML файлі необхідно прописати тег зображення, а в його src-атрибуті розмістити URL-адресу цього маршруту.

```
def generate_frames():
    while True:

        ## read the camera frame
        success, frame = camera.read()
        if not success:
            break
        else:
            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()

        yield (b'--frame\r\n'
              b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

Рисунок 4.6 - Функція, що обробляє зображення з камери

```
@app.route('/video')
def video():
    return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

Рисунок 4.7 - Потокова відповідь із зображенням

Браузер автоматично оновлюватиме елемент зображення, відображаючи в ньому потік зображень JPEG. Flask має вбудовану підтримку динамічного механізму шаблонів, відомого як Jinja. Flask дозволяє передавати будь-який об'єкт Python у шаблон, на який потім можна посилатися всередині шаблону з використанням синтаксису Jinja. Отже для виводу зображення на веб-сторінку необхідно використовувати роздільник `{{...}}` (рис. 4.8), цей роздільник друкує вміст між фігурними дужками у

вихідний файл шаблону. Результат роботи трансляції наведено на рисунку 4.9.

```
<div class="col-xs-12 col-sm-6 col-md-5 col-lg-5 pull-center video_stream p-2 mx-1">
  
</div>
<br>
```

Рисунок 4.8 - HTML-код для виводу трансляції відео на вебсторінку

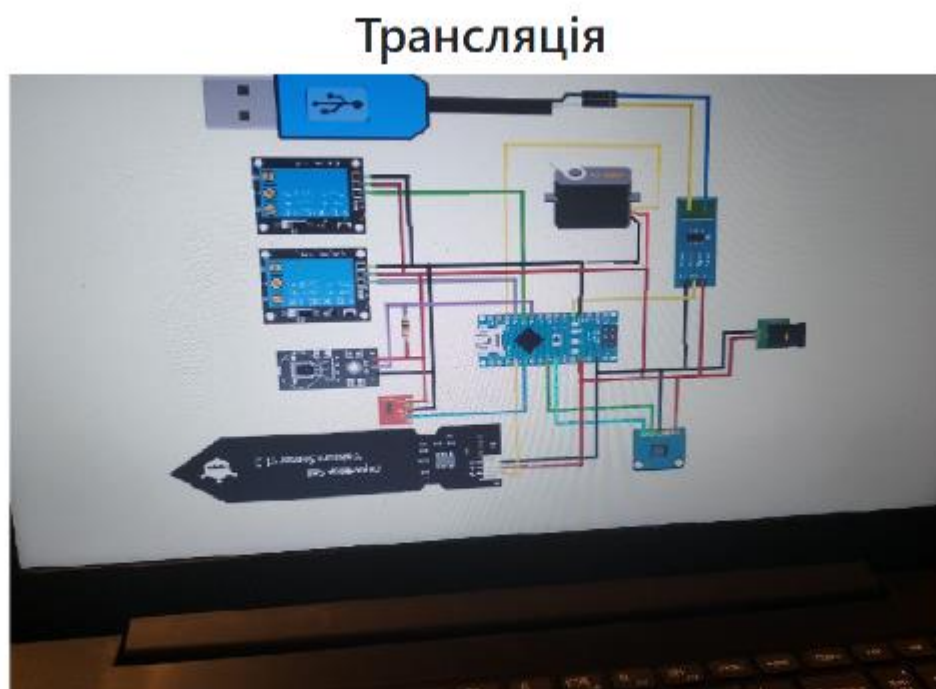


Рисунок 4.9 – Результат відеотрансляції

4.2 Візуалізація даних у вигляді графіків

Для візуалізації даних у вигляді графіків на вебсторінці було використано бібліотеку Highcharts.

Highcharts [77] це сучасна мультиплатформна бібліотека на основі масштабованої векторної графіки. Вона дозволяє легко и швидко додавати діаграми до вебдодатків та дуже широко використовується саме для демонстрації даних і статистики. Ця бібліотека є безкоштовною для

використання у некомерційних цілях. На рисунку 4.10 наведено код підключення бібліотеки та модулів для оновлення графіків в режимі реального часу. Highcharts має два основних способи роботи з потоком даних у режимі реального часу. Перший - це використання модулю даних із опитуванням, другий - налаштування власного підключення з використанням методів API Highcharts для підтримки графіку в актуальному стані. У роботі було використано другий спосіб, оскільки він забезпечує більш програмний контроль.

```
<!--High CHART LIVE -->
<script src="http://code.highcharts.com/highcharts.js"></script>
<script src="http://code.highcharts.com/highcharts-more.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>
```

Рисунок 4.10 - HTML-код для виводу трансляції відео на веб-сторінку

Для його реалізації було створено функцію requestData(). В першу чергу вона відправляє запит на сервер для отримання даних (рис.4.11). Після успішного отримання даних точка додається до серії графіка (рис. 4.12), для цього використовується метод addPoint(). При цьому якщо довжина ряду буде більшою ніж тридцять, то перша точка буде зміщуватись ліворуч (рис. 4.13), це зроблено для того, щоб весь ряд зміщувався ліворуч, а не просто щільно стискав точки на графіку.

```
function requestData()
{
    // Ajax call to get the Data from Flask
    var requests = $.get('/data');

    var tm = requests.done(function (result)
    {
```

Рисунок 4.11 – Відправка запиту до серверу

```

//Soil
chartTemperature.series[0].addPoint(data1, true, shiftTemperature);
chartHumidity.series[0].addPoint(data2, true, shiftHumidity);
//Air
chartTemperature.series[1].addPoint(data3, true, shiftTemperature);
chartHumidity.series[1].addPoint(data4, true, shiftHumidity);
//Light
chartLight.series[0].addPoint(data5, true, shiftLight);

```

Рисунок 4.12 – Додавання точок до графіків

```

// Temperature
var seriesTemperature = chartTemperature.series[0],
    shiftTemperature = seriesTemperature.data.length > 30;
// Humidity
var seriesHumidity = chartHumidity.series[0],
    shiftHumidity = seriesHumidity.data.length > 30;
//Light
var seriesLight = chartLight.series[0],
    shiftLight = seriesLight.data.length > 30;

```

Рисунок 4.13 – Встановлення довжини ряду для графіків

На рисунку 4.14 зображено результати візуалізації даних в режимі реального часу з використанням бібліотеки Highcharts.

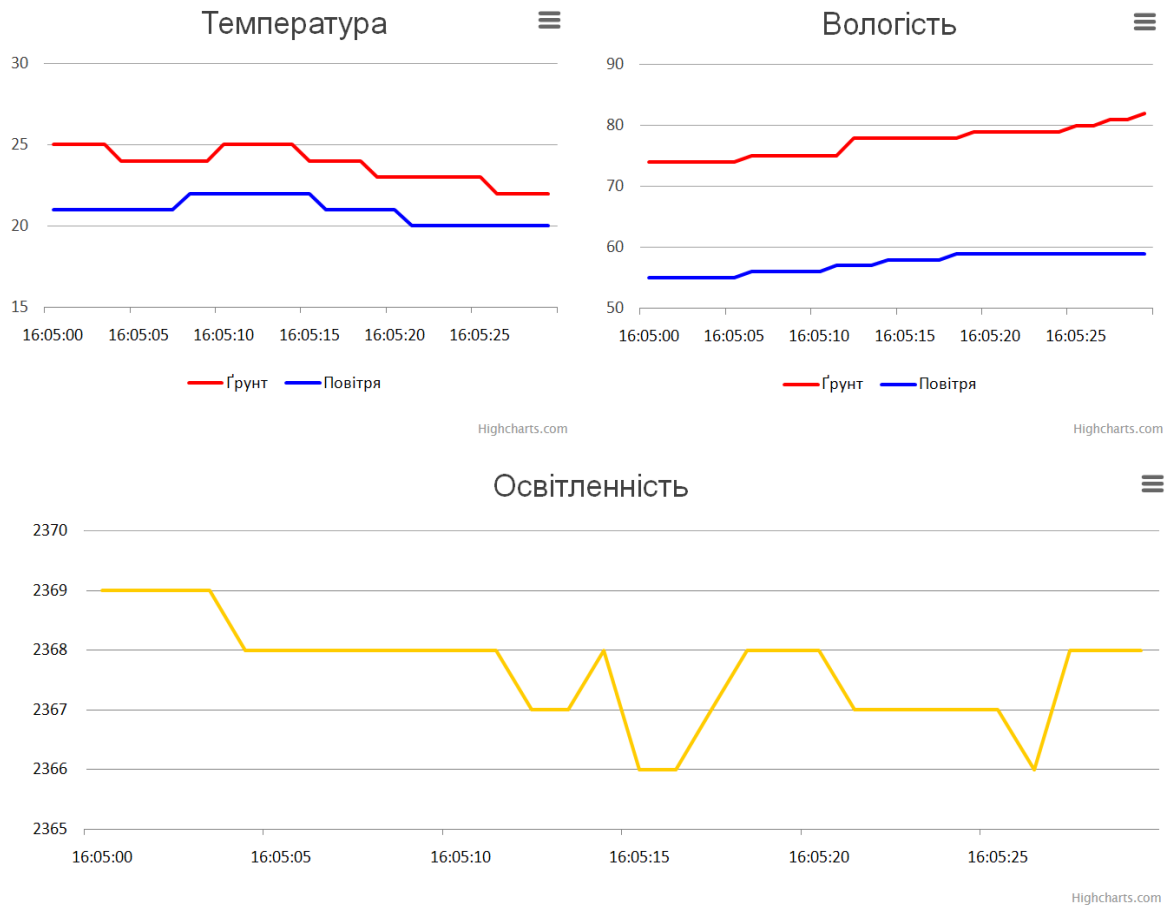


Рисунок 4.14 – Результати візуалізації даних

Результати роботи IoT сервісу наведено на рисунку 4.15.

4.3 Висновки до розділу 4

Використання бібліотеки OpenCV значно спростило роботу з відео зображенням, а особливість фреймворку Flask використовувати функцію-генератор для потокової передачі пришвидшила її реалізацію. Використання бібліотеки Highcharts забезпечило візуалізацію даних у вигляді графіків, а методи API Highcharts дозволили підтримувати ці графіки в актуальному стані.

Показники ґрунту

Температура: 22 °C

Вологість: 82 %

Показники повітря

Температура: 20 °C

Вологість: 59 %

Показники освітленості

2368 lux

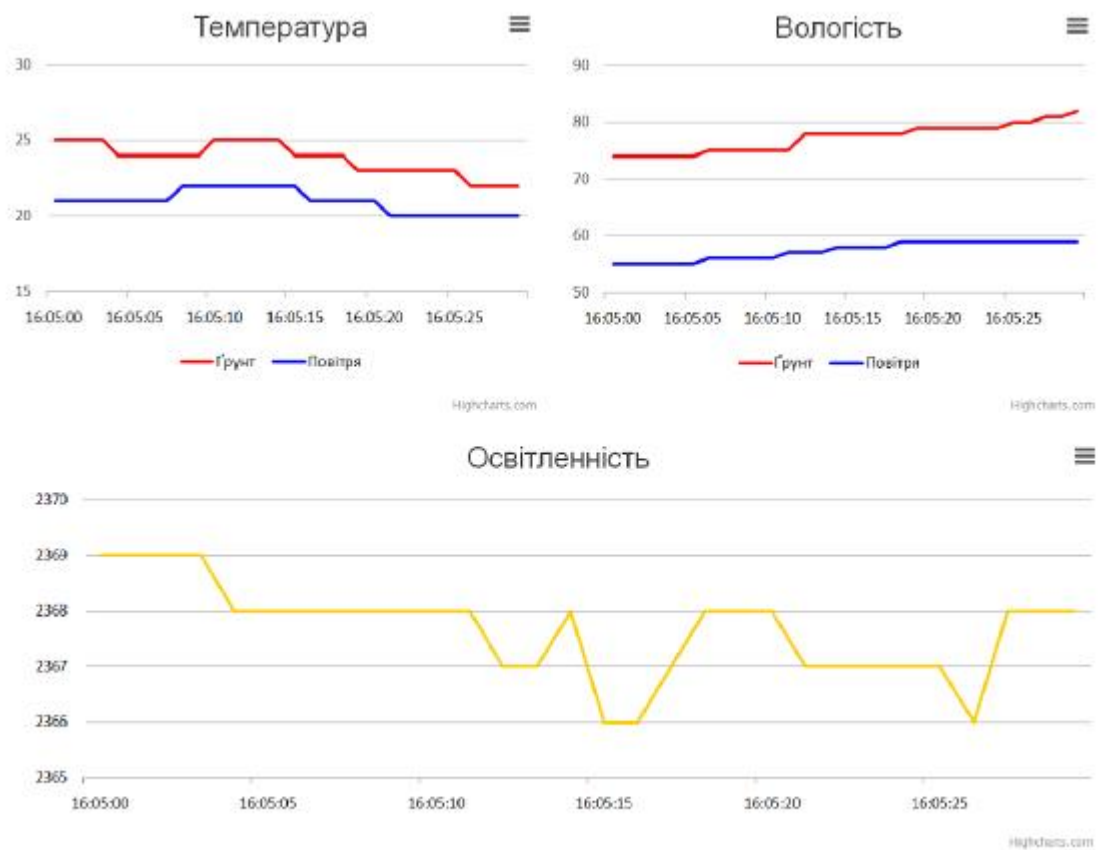
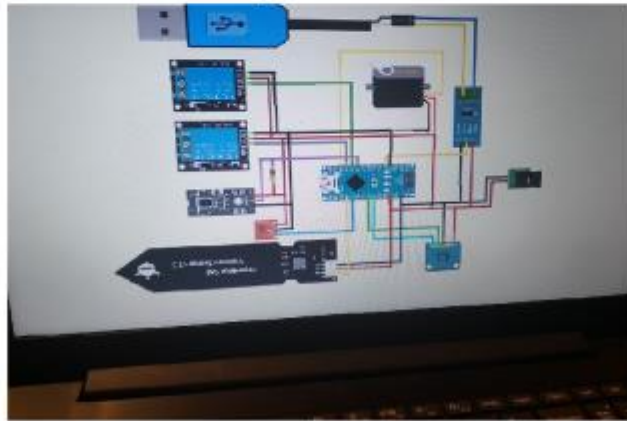
Трансляція

Рисунок 4.15 – Результати роботи IoT сервісу

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи магістра було досліджено методи та типи хмарного комп'ютингу. Хмарні технології проникли майже в усі сфери людської діяльності та сфера IoT не є винятком. Огляд існуючих хмарних IoT сервісів показав, що вони можуть візуалізувати дані, що отримані з датчиків, але на жаль вони не надають можливості переглянути відеозображення IoT датчиків у разі надзвичайної ситуації (наприклад, відсутність даних з датчика, незрозумілі різкі стрибки показників тощо).

В ході досліджень було розроблено IoT сервіс, який отримує та візуалізує дані з датчиків, а також транслює та зберігає відео з камери, що наведена на ці датчики.

Для розробки ПЗ було обрано мову Python та фреймворк Flask. Для збереження відео використовувався хмарний FTP сервіс Camera-ftp.

В подальшій роботі планується удосконалити процес збереження відео, та забезпечити доступ до архіву з відеозаписами через розроблену вебсторінку.

Наукова новизна роботи полягає в тому, що було удосконалено метод роботи з відеопотоком за рахунок збереження серії окремих знімків на FTP сервері, завдяки чому збереження відео потребує менше ресурсів.

Практичне значення роботи полягає у тому, що розроблений IoT сервіс може бути легко інтегровано у вигляді нової вебсторінки до складу віддаленої лабораторії Smart House&IoT, що дозволить розширити її функціональні можливості та організувати взаємодію з новим експериментом Розумна теплиця.

Основні положення та результати роботи були представлені на Міжнародній науковій конференції IRC, м. Дортмунд, Німеччина (1-2 липня 2022 р.), а також на Міжнародній НПК «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Why is cloud computing important? [Electronic resource]. – Access mode: <https://opencirrus.org/cloud-computing-important/>.
2. What is cloud computing? [Electronic resource]. – Access mode: <https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing/>.
3. Cloud Computing Overview [Electronic resource]. – Access mode: https://www.tutorialspoint.com/cloud_computing/cloud_computing_overview.htm.
4. Mohammad, H. Cloud computing uncovered: A research landscape / [H. Mohammad, T. Ladan] // Advances in Computers. – 2012. – Vol. 86 - P. 41-85.
5. What Is Broad Network Access? [Electronic resource]. – Access mode: https://www.spiceworks.com/tech/cloud/articles/what-is-broad-network-access/#_001.
6. Resource Pooling [Electronic resource]. – Access mode: <https://www.techopedia.com/definition/29545/resource-pooling>.
7. How Will Rapid Elasticity in Cloud Computing Help Your Business? [Electronic resource]. – Access mode: <https://www.synopsys.com/cloud/insights/rapid-elasticity-in-cloud-computing-on-demand-scaling-for-business.html#a>.
8. Measured Service in Cloud Computing [Electronic resource]. – Access mode: <https://www.techopedia.com/definition/14469/measured-service-cloud-computing>.
9. Private cloud vs public cloud: dit moet je weten [Elektronische bron] - Toegangsmodus: <https://www.kpn.com/zakelijk/blog/private-cloud-vs-public-cloud-dit-moet-je-weten.htm>.
10. Public Cloud Model [Electronic resource]. – Access mode: https://www.tutorialspoint.com/cloud_computing/cloud_computing_public_cloud_model.htm.

11. What is a Private Cloud? [Electronic resource]. – Access mode: <https://www.vmware.com/topics/glossary/content/private-cloud.html>.
12. Brohi, S. N. Challenges and benefits for adopting the paradigm of cloud computing [Text] / S. N. Brohi, M. A. Bamiah // International Journal of Advanced Engineering Sciences and Technologies - 2011. - Vol. 8, № 2. - P.286-290.
13. Private Cloud Model [Electronic resource]. – Access mode: https://www.tutorialspoint.com/cloud_computing/cloud_computing_private_cloud_model.htm.
14. Overview and Benefits of a Community Cloud Model [Electronic resource]. – Access mode: <https://www.parallels.com/blogs/ras/community-cloud/>.
15. Community Cloud Model [Electronic resource]. – Access mode: https://www.tutorialspoint.com/cloud_computing/cloud_computing_community_cloud_model.htm.
16. What's the Difference between Public, Private, Hybrid, and Community Clouds? [Electronic resource]. – Access mode: <https://www.abacusnext.com/blog/whats-difference-between-public-private-hybrid-and-community-clouds/>.
17. Why Hybrid Cloud Computing? [Electronic resource]. – Access mode: <https://www.empowerit.com.au/blog/technology-trends/why-hybrid-cloud-computing/>.
18. Hybrid Cloud Model [Electronic resource]. – Access mode: https://www.tutorialspoint.com/cloud_computing/cloud_computing_hybrid_cloud_model.htm.
19. Suliman, M. E. A Brief Analysis of Cloud Computing Infrastructure as a Service (IaaS) / M. E. Suliman, K. S. A. Madinah // International Journal of Innovative Science and Research Technology. - 2021. - Vol. 6, № 1. - P. 1409 - 1412.
20. Cloud Computing Infrastructure as a Service (IaaS) [Electronic resource]. – Access mode:

https://www.tutorialspoint.com/cloud_computing/cloud_computing_infrastructure_as_a_service.htm.

21. Types of Cloud Services [Electronic resource]. – Access mode: <https://medium.com/@IDMdatasecurity/types-of-cloud-services-b54e5b574f6>.

22. What is Platform as a Service (PaaS)? [Electronic resource]. – Access mode: <https://cloud.google.com/learn/what-is-paas>.

23. Cloud Computing Platform as a Service (PaaS) [Electronic resource]. – Access mode: https://www.tutorialspoint.com/cloud_computing/cloud_computing_platform_as_a_service.htm.

24. What is SaaS – software-as-a-service? [Electronic resource]. – Access mode: <https://www.ibm.com/topics/saas>.

25. Cloud Computing Software as a Service (SaaS) [Electronic resource]. – Access mode: https://www.tutorialspoint.com/cloud_computing/cloud_computing_software_as_a_service.htm.

26. SaaS Vs PaaS Vs IaaS and Which Model is Better for You [Electronic resource]. – Access mode: <https://tms-outsource.com/blog/posts/saas-vs-paas-vs-iaas/>.

27. Internet of Things: What It Is, How It Works, Examples and More [Electronic resource]. – Access mode: <https://justcreative.com/internet-of-things-explained/>.

28. The 4 stages of an IoT architecture [Electronic resource]. – Access mode: <https://techbeacon.com/enterprise-it/4-stages-iot-architecture>.

29. Different Types of Internet of Things (IoT) Sensors [Electronic resource]. – Access mode: <https://www.thomasnet.com/articles/instruments-controls/types-of-internet-of-things-iot-sensors/>.

30. A Guide To Actuators In IoT: Types, Anatomy & Examples [Electronic resource]. – Access mode: <https://www.intuz.com/blog/actuators-in-iot-guide>.

31. Top sensor types used in IoT [Electronic resource]. – Access mode: <https://www.avsystem.com/blog/iot-sensors-iot-actuators/>.
32. The 4 Stages of IoT Architecture [Electronic resource]. – Access mode: <https://www.digi.com/blog/post/the-4-stages-of-iot-architecture>.
33. A Simple Guide to IoT Architecture [Electronic resource]. – Access mode: <https://www.totalphase.com/blog/2019/10/simple-guide-iot-architecture/>.
34. IoT Cloud in 2022: Benefits, Challenges, Platforms&Functionality[Electronic resource]. – Access mode: <https://research.aimultiple.com/iot-cloud/>.
35. IoT Core [Electronic resource]. – Access mode: <https://cloud.google.com/iot-core>.
36. Google IoT Platform: awesome tools for any project [Electronic resource]. – Access mode: <https://indeema.com/blog/google-cloud>.
37. Google Cloud IoT Core with Raspberry Pi [Electronic resource]. – Access mode: https://www.youtube.com/watch?v=3Zwlj9x96Jg&ab_channel=EvanCrane.
38. Wat is Azure Internet of Things (IoT)? [Electronic resource]. – Access mode: <https://learn.microsoft.com/nl-nl/azure/iot-fundamentals/iot-introduction>.
39. IoT on Microsoft Azure [Electronic resource]. – Access mode: <https://medium.com/@shaha/iot-on-microsoft-azure-8bb02f5b3a50>.
40. Microsoft Azure IoT and Raspberry Pi : Send telemetry data to azure using python [Electronic resource]. – Access mode: https://www.youtube.com/watch?v=7T91jbtm95A&ab_channel=Kitflix.
41. Kaa IoT Platform Features for Enterprise IoT Projects [Electronic resource]. – Access mode: <https://www.kaaiot.com/products/overview>.
42. Webinar - Kaa Identity Access Management (IAM) [Electronic resource]. – Access mode: https://www.youtube.com/watch?v=6WG0zxtkI3w&ab_channel=KaaIoT.
43. Particle [Electronic resource]. – Access mode: <https://www.saasworthy.com/product/particle-io>.

44. Particle 101 - Understanding Particle Variables and Functions [Electronic resource]. – Access mode: https://www.youtube.com/watch?v=POHyi5xV8Mc&ab_channel=Particle.

45. Advantages and Disadvantages of Cloud Computing [Electronic resource]. – Access mode: <https://www.weetechsolution.com/blog/the-know-how-of-cloud-computing>.

46. SimulIDE [Electronic resource]. – Access mode: <https://www.simulide.com/p/home.html>.

47. SimulIDE For Simulating Electronic Circuits [Electronic resource]. – Access mode: <https://www.electronicshobby.com/buyers-guides/simulide-simulating-electronic-circuits>.

48. Wokwi [Electronic resource]. – Access mode: <https://wokwi.com/>.

49. How to use Wokwi Arduino Simulator in 2022? What is Wokwi? [Electronic resource]. – Access mode: <https://create.arduino.cc/projecthub/Hack-star-Arduino/how-to-use-wokwi-arduino-simulator-in-2022-what-is-wokwi-304e6b>.

50. Simuli [Electronic resource]. – Access mode: <https://simuli.co/>.

51. Iotify Introduction [Electronic resource]. – Access mode: <https://iotify.help/virtual-lab/>.

52. STM32 Nucleo [Electronic resource]. – Access mode: <https://www.st.com/en/ecosystems/stm32-nucleo.html>.

53. BMP180 Barometric Pressure/Temperature/Altitude Sensor- 5V [Electronic resource]. – Access mode: <https://www.robotpark.com/91643-BMP180-Barometric-Pressure-Temperature-Altitude-Sensor-5V-ready>.

54. STM32 Nucleo-64 development board with STM32F411RE MCU, supports Arduino and ST morpho connectivity [Electronic resource]. – Access mode: <https://www.st.com/en/evaluation-tools/nucleo-f411re.html>.

55. What is Adafruit IO? [Electronic resource]. – Access mode: <https://learn.adafruit.com/welcome-to-adafruit-io/what-is-adafruit-io>.

56. Adafruit [Electronic resource]. – Access mode:

<https://io.adafruit.com/>.

57. IFTTT [Electronic resource]. – Access mode: <https://ifttt.com/>.
58. How IFTTT Can Help Your Business [Electronic resource]. – Access mode: <https://www.businessnewsdaily.com/4919-ifttt-for-business.html>.
59. W5500 Ethernet Module [Electronic resource]. – Access mode: <https://www.hotmcu.com/w5500-ethernet-module-p-256.html>.
60. Arduino IDE [Electronic resource]. – Access mode: <https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics>.
61. REIoT [Electronic resource]. – Access mode: <http://swed.zntu.edu.ua/>.
62. The Remote Experimentation as the Practical-Oriented Basis of Inclusive Engineering Education [Text] / [H. D. Wuttke, A. Parkhomenko, A. Tulenkov et al.] // iJOE. – 2019. - Vol. 15, № 5. - P. 4-17.
63. Implementation of reusable solutions for remote laboratory development [Text] / [A. Parkhomenko, O. Gladkova, A. Sokolyanskii et al.] // iJOE. - 2016. - Vol.12, No. 7. - P. 24–29.
64. Why Choose Python Over Other Languages? [Electronic resource]. – Access mode: <https://www.binpress.com/why-choose-python/>.
65. 5 Reasons to choose Python programming language [Electronic resource]. – Access mode: <https://content.techgig.com/technology-guide/5-reasons-to-choose-python-programming-language/articleshow/84017951.cms>.
66. Camera-ftp [Electronic resource]. – Access mode: <https://camera-ftp.com/en/>.
67. Why Learn Python? 6 Reasons Why it's So Hot Right Now. [Electronic resource]. – Access mode: <https://codingnomads.co/why-learn-python/>.
68. Django vs Flask vs FastAPI – A Comparative Guide to Python Web Frameworks [Electronic resource]. – Access mode: <https://analyticsindiamag.com/django-vs-flask-vs-fastapi-a-comparative-guide-to-python-web-frameworks/>.
69. Django vs Flask vs FastAPI for Software Founders [Electronic resource]. – Access mode: https://dev.to/kateryna_pakhomova/django-vs-flask-vs-

fastapi-for-software-founders-45k4.

70. 10 Best Python IDE & Code Editors [Updated Guide] [Electronic resource]. – Access mode: <https://hackr.io/blog/best-python-ide>.

71. 12 BEST Python IDE & Code Editors For Mac & Windows In 2022 [Electronic resource]. – Access mode: <https://www.softwaretestinghelp.com/python-ide-code-editors/>.

72. Python Programming in the Eclipse IDE [Electronic resource]. – Access mode: <https://www.ics.uci.edu/~pattis/common/handouts/introtopythoneclipse/>.

73. Best IDE's for Python [Electronic resource]. – Access mode: <https://medium.com/@divya.maddipudi1/best-ides-for-python-82c509bd6041>.

74. GSP 318: Introduction to Python and the Wing IDE [Electronic resource]. – Access mode: https://www.youtube.com/watch?v=4JvxRKLtnXI&ab_channel=JamesGraham.

75. What Is File Transfer Protocol (FTP) and What Is It Used For? [Electronic resource]. – Access mode: <https://www.investopedia.com/terms/f/ftp-file-transfer-protocol.asp>.

76. What is FTP and How Does it Work With Security Cameras? [Electronic resource]. – Access mode: <https://help.annke.com/hc/en-us/articles/4404397349401-What-is-FTP-and-How-Does-it-Work-With-Security-Cameras->.

77. Highcharts [Electronic resource]. – Access mode: <https://www.highcharts.com/>.

ДОДАТОК А
Текст програми

A.1 Код до експерименту з платою STM32 Nucleo 64 датчиком BMP180 та вбудованим IDE у Simuli

```

#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>
#define seaLevelPressure_hPa 1013.25
Adafruit_BMP085 bmp;
void setup() {
    Serial.begin(9600);
    if (!bmp.begin()) {
        Serial.println("Could not find a valid BMP180
sensor, check wiring!");
        while (1) {}
    }
}

void loop() {

    Serial.print("Pressure = ");
    Serial.print(bmp.readPressure());
    Serial.println(" Pa");

    Serial.print("Temperature = ");
    Serial.print(bmp.readTemperature());
    Serial.println(" *C");
    Serial.println();
    delay(1000);
}

...

```

A.2 Код до експерименту з платою Arduino Uno модулем W5500 та сторонніми сервісами у Simuli

```
#include <SPI.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

#include <Ethernet.h>
#include <EthernetClient.h>

bool mqttConnected = false;

//Налаштування підключення
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress iotIP (109, 74, 198, 92);
IPAddress dnsIP (8, 8, 8, 8);

//Налаштування Adafruit.io

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "USERNAME"
#define AIO_KEY          "ADAFRUITIOKEY"

//Налаштування клієнту Ethernet
EthernetClient client;

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER,
AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
```

```

    #define halt(s) { Serial.println(F( s ));
while(1); }

//Налаштування каналів

Adafruit_MQTT_Subscribe lamp =
Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
"/feeds/assistant-lamp");

//ОСНОВНИЙ КОД

void setup() {
    Serial.begin(115200);
    pinMode(7, OUTPUT);

    digitalWrite(7, HIGH);
    delay(2000);
    digitalWrite(7, LOW);
    Serial.println(F("Smart Lamp with Google
Assistant"));

    Serial.print(F("\nInit the Client..."));
    Ethernet.begin(mac, iotIP, dnsIP);
    delay(1000); //give the ethernet a second to
initialize

    mqtt.subscribe(&lamp);
}

```

```

void loop() {

    while (!mqttConnected) {
        MQTT_connect();
    }

    Adafruit_MQTT_Subscribe *subscription;
    while ((subscription =
mqtt.readSubscription(1000))) {
        if (subscription == &lamp) {
            Serial.print(F("Got: "));

            char *received = (char *) lamp.lastread;
            Serial.println(received);
            Serial.println(lamp.lastread[0]);
            if (lamp.lastread[0] == 49) {
                digitalWrite(7, HIGH);
                Serial.println(F("Turn On Light"));
            }
            else if (lamp.lastread[0] == 48) {
                digitalWrite(7, LOW);
                Serial.println(F("Turn Off Light"));
            }
        }
    }

    // ping the server to keep the mqtt connection
alive

    if ((millis() % 1000) == 0) {
        if (! mqtt.ping()) {

```

```

        mqttConnected = false;
    }
}

// Функція для підключення сервера MQTT
void MQTT_connect() {
    int8_t ret;

    if (mqtt.connected()) {
        Serial.println(F("Adafruit MQTT was already
connected"));
        return;
    }

    Serial.print("Connecting to MQTT... ");

    while ((ret = mqtt.connect()) != 0) {
        Serial.println(mqtt.connectErrorString(ret));
        Serial.println("Retrying MQTT connection in 5
seconds...");
        mqtt.disconnect();
        delay(5000);    }
    Serial.println("MQTT Connected!");
    mqttConnected = true;
}

```

A.3 Код з файлу app.py для захоплення та передачі відеопотоку

```

...

from flask import Flask, render_template, url_for,
request, redirect, make_response, jsonify, Response
import json
from time import time
import cv2
from camera import Camera

app = Flask (__name__)
source =
"rtsp://camera_test:gy78skY5SP3@192.168.1.10:554/stream
2"
camera=cv2.VideoCapture(source)

def generate_frames():
    while True:

        ## read the camera frame
        success, frame = camera.read()
        if not success:
            break
        else:
            ret, buffer = cv2.imencode('.jpg',
frame)

            frame = buffer.tobytes()

            yield (b'--frame\r\n'
                b'Content-Type: image/jpeg\r\n\r\n'
+ frame + b'\r\n')

@app.route('/video')
def video():
    return Response(generate_frames(),
mimetype='multipart/x-mixed-replace; boundary=frame')

...

```

A.4 Код функції requestData() з файлу main.js для обробки даних та відправки їх на HTML сторінку

```
function requestData() {
    // Ajax call to get the Data from Flask
    var requests = $.get('/data');

    var tm = requests.done(function(result) {
        // Add the Point
        // Time Soil Temperature
        var data1 = [];
        data1.push(result[0]);
        data1.push(result[1]);
        // Add the Point
        // Time Soil Humidity
        var data2 = [];
        data2.push(result[0]);
        data2.push(result[2]);
        // Add the Point
        // Time Air Temperature
        var data3 = [];
        data3.push(result[0]);
        data3.push(result[3]);
        // Add the Point
        // Time Air Humidity
        var data4 = [];
        data4.push(result[0]);
        data4.push(result[4]);
        // Add the Point
        // Time Light
        var data5 = [];
        data5.push(result[0]);
        data5.push(result[5]);

        // Temperature
        var seriesTemperature =
chartTemperature.series[0],
            shiftTemperature =
seriesTemperature.data.length > 30;
        // Humidity
        var seriesHumidity =
chartHumidity.series[0],
            shiftHumidity =
seriesHumidity.data.length > 30;
```

```

        //Light
        var seriesLight = chartLight.series[0],
            shiftLight = seriesLight.data.length >
30;

        //Soil
        chartTemperature.series[0].addPoint(data1,
true, shiftTemperature);
        chartHumidity.series[0].addPoint(data2,
true, shiftHumidity);
        //Air
        chartTemperature.series[1].addPoint(data3,
true, shiftTemperature);
        chartHumidity.series[1].addPoint(data4,
true, shiftHumidity);
        //Light
        chartLight.series[0].addPoint(data5, true,
shiftLight);
        //Soil
        $(".sensor1").text("");
        $(".sensor1").text("Температура : " +
Math.round(data1[1]) + " °C");

        $(".sensor2").text("");
        $(".sensor2").text("Вологість : " +
Math.round(data2[1]) + " % ");

        //Air
        $(".sensor3").text("");
        $(".sensor3").text("Температура : " +
Math.round(data3[1]) + " °C");

        $(".sensor4").text("");
        $(".sensor4").text("Вологість : " +
Math.round(data4[1]) + " % ");

        //Light
        $(".sensor5").text("");
        $(".sensor5").text(" " +
Math.round(data5[1]) + " lux ");

        // call it again after one second
        setTimeout(requestData, 2000);
        plot_chart()
    });
}

```

A.5 Код з файлу main.html для виводу відеопотоку та даних

```

<div class="container-fluid">
  <div class="row">

    <div class="col-xs-12 col-sm-6 col-md-5
col-lg-3 jumbotron p-2 mx-1">
      <h4>Показники ґрунту</h4>
      <p class="sensor1"> Температура:
</p>

      <p class="sensor2"> Вологість: </p>
      <div class="container-fluid" id="">

        </div>
      </div>

      <br>

      <div class="col-xs-12 col-sm-6 col-md-5
col-lg-3 jumbotron p-2 mx-1">
        <h4>Показники повітря</h4>
        <p class="sensor3">Температура:
</p>

        <p class="sensor4"> Вологість: </p>
      </div>
      <br>
      <div class="col-xs-12 col-sm-6 col-md-5
col-lg-3 jumbotron p-2 mx-1">
        <h4>Показники освітленості</h4>
        <p class="sensor5"> </p>
      </div>
      <br>

    </div>
    <br>

    <div class="row">

      <div class="col-xs-12 col-sm-6 col-md-5
col-lg-5 pull-center video_stream p-2 mx-1">
        <h3>Трансляція</h3>
        
      </div>

```

```

        <br>
    </div>
</div>
</div>

<br>
<br>
<br>
<div class="col-xs-6 col-sm-5 col-md-5 col-lg-
5"> </div>
    <div class="container-fluid">
        <div class="row">
            <div class="col-xs-12 col-sm-5 col-md-5
col-lg-5 container-fluid" id="data-temperature">
                </div>
            <div class="col-xs-12 col-sm-5 col-md-5
col-lg-5 container-fluid" id="data-humidity">
                </div>

        </div>
    </div>
<br>
<br>
<br>
<div class="container-fluid">
    <div class="row">
        <div class="container-fluid" id="data-
light">
            </div>
        </div>
    </div>
</div>

```

ДОДАТОК Б
Слайди презентації

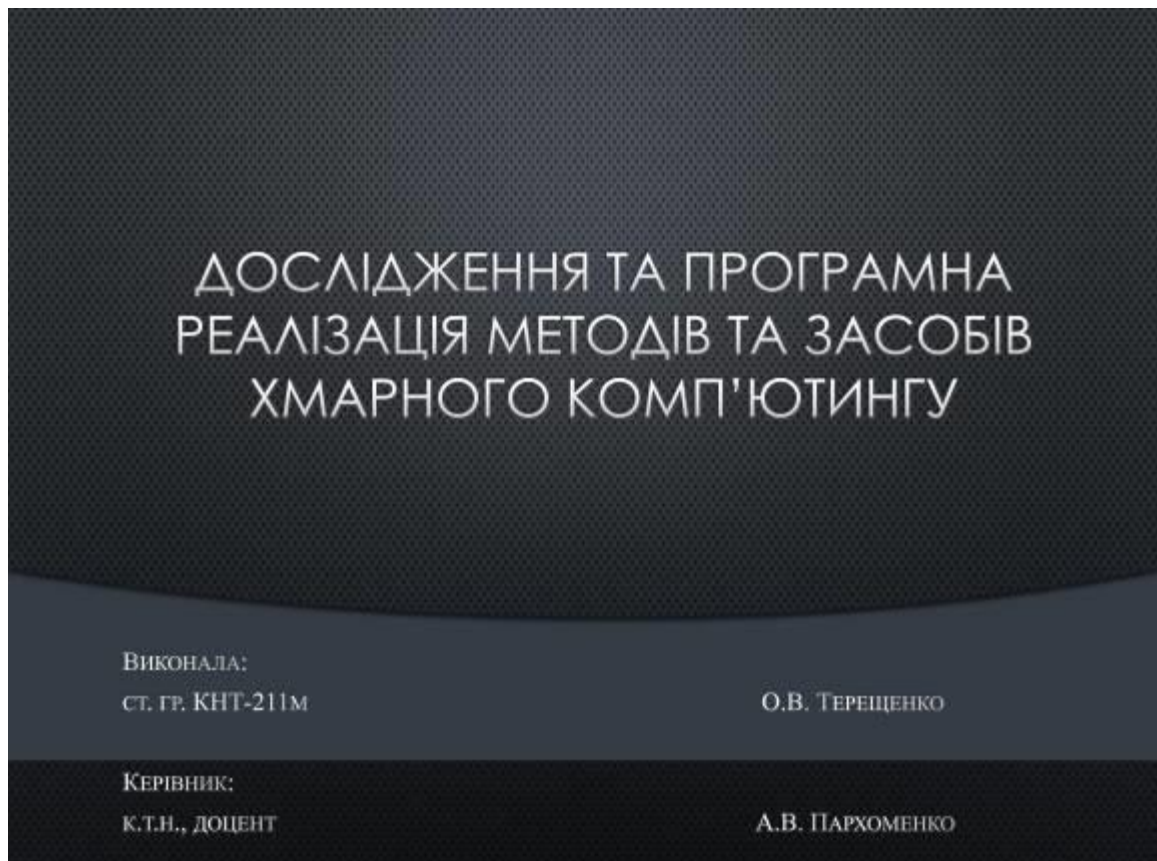


Рисунок Б.1 – Слайд 1



Рисунок Б.2 – Слайд 2

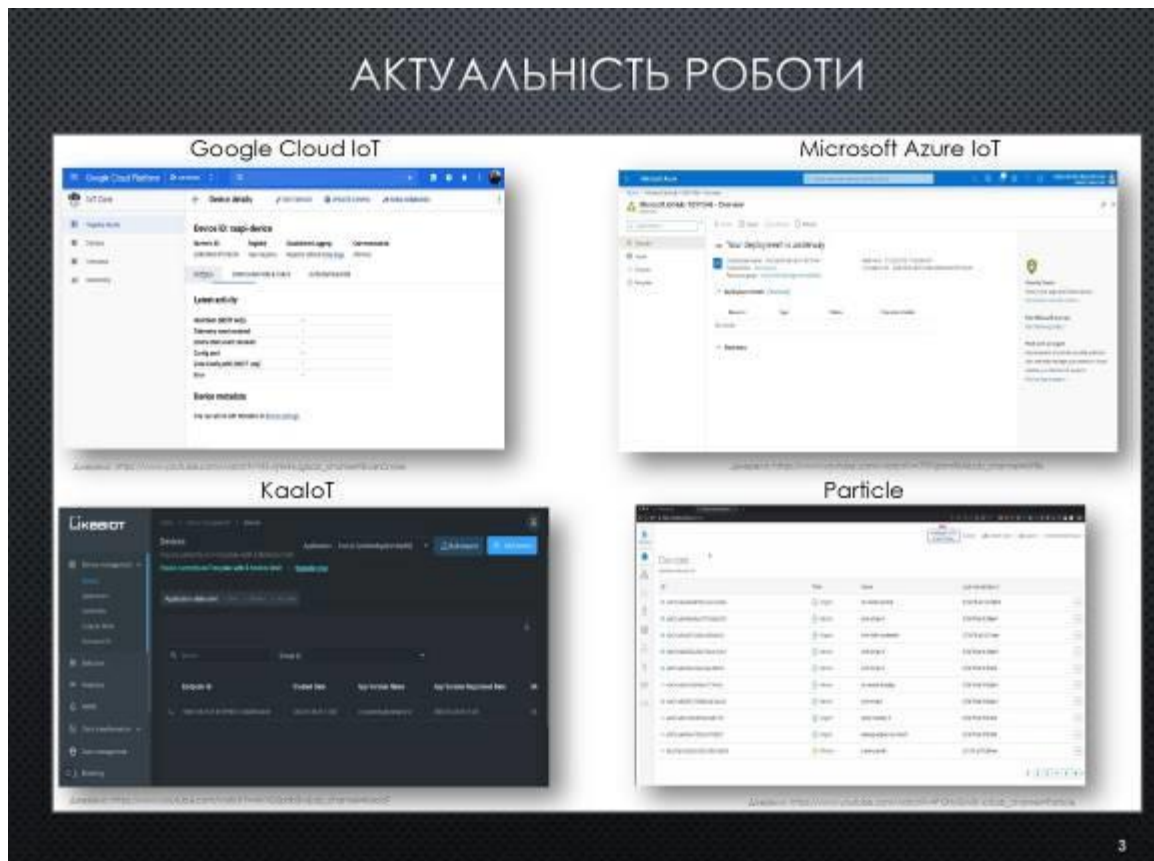


Рисунок Б.3 – Слайд 3

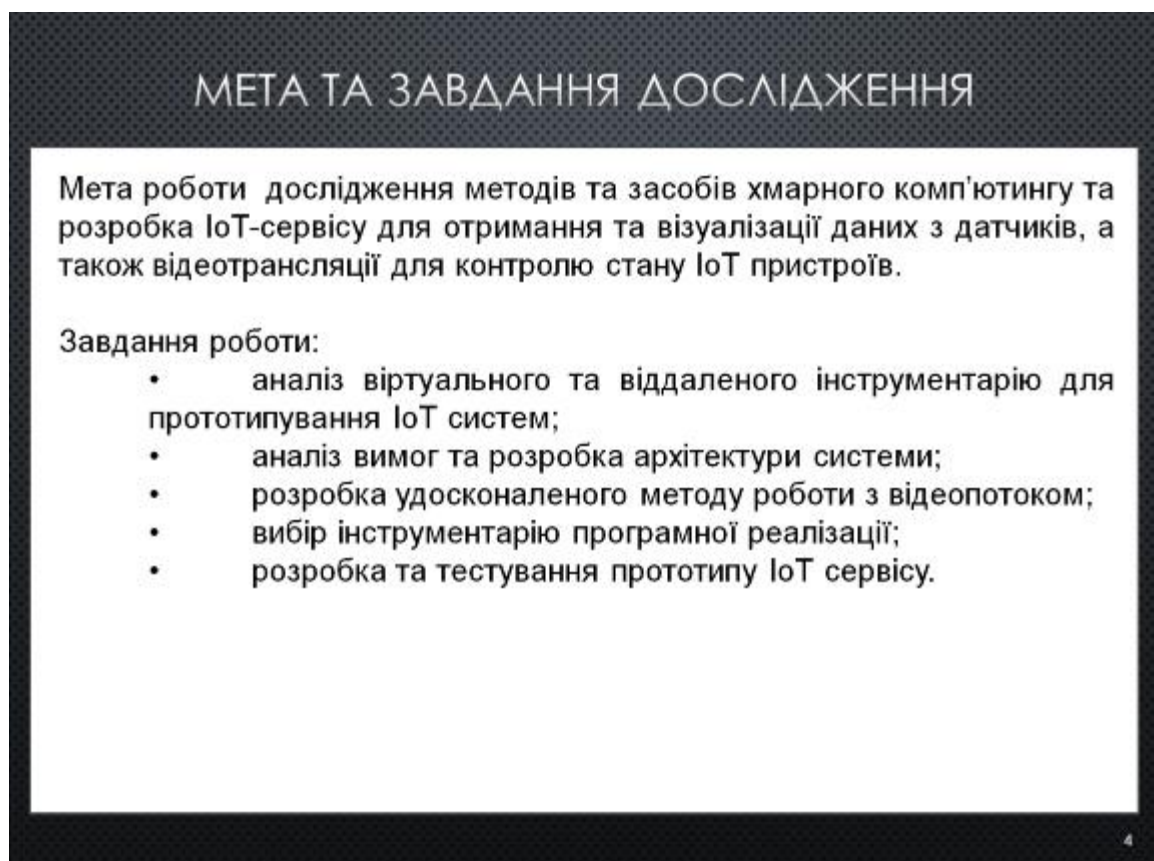


Рисунок Б.4 – Слайд 4

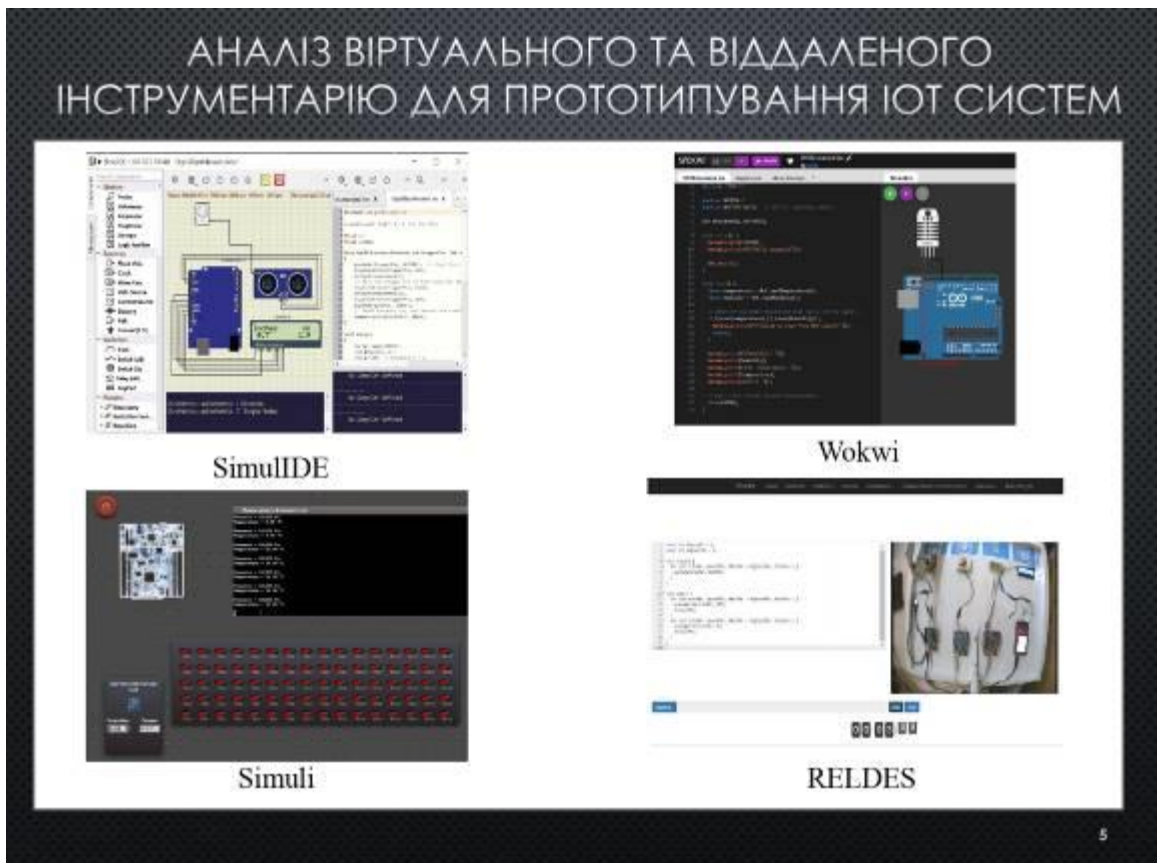


Рисунок Б.5 – Слайд 5

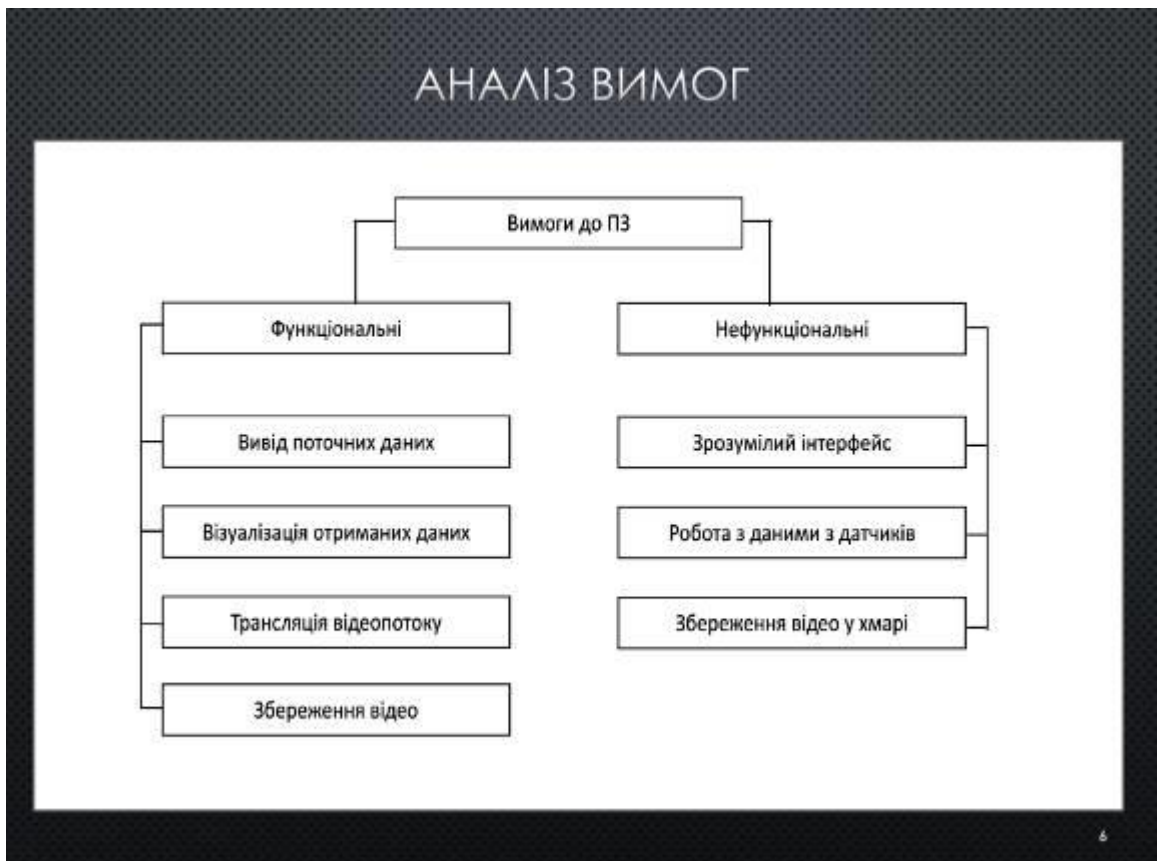


Рисунок Б.6 – Слайд 6



Рисунок Б.7 – Слайд 7



Рисунок Б.8 – Слайд 8

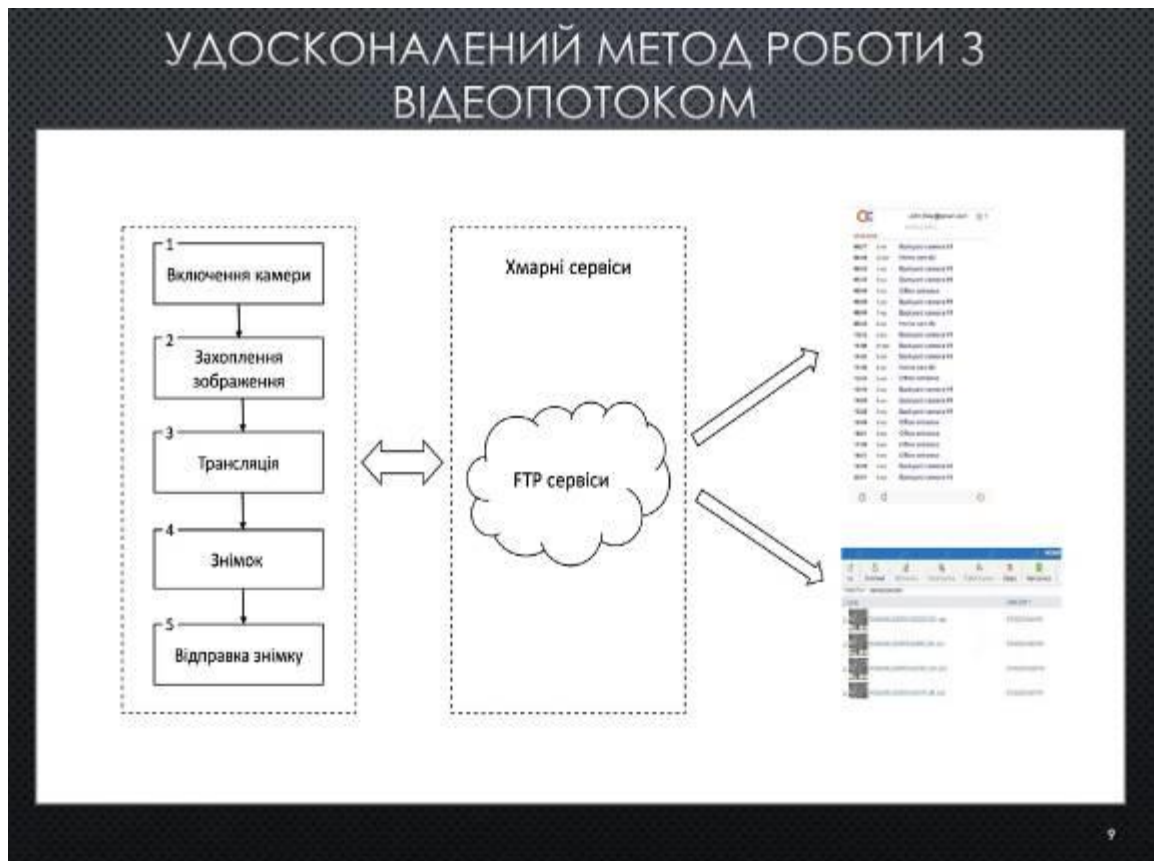


Рисунок Б.9 – Слайд 9

ВИБІР МОВИ ПРОГРАМУВАННЯ ТА ФРЕЙМВОРКУ

Характеристики	Фреймворки		
	Django	Flask	FastAPI
Тип	Full Stack фреймворк	Легкий мікрофреймворк	Веб-фреймворк для створення API
Застосування	Для багатосторінкових додатків та великих проєктів	Для односторінкових додатків та невеликих проєктів	Для багатосторінкових додатків та великих проєктів
Гнучкість	Не дозволяє вносити зміни до модулів	Неймовірно гнучкий, оскільки не має вбудованих розширень	Гнучкість забезпечується моделями Pydantic
Контроль	Не дає повного контролю над функціями та модулями	Дозволяє повністю контролювати створення програми	Дозволяє повністю контролювати створення програми
Відладчик	Не підтримує віртуальну відладку	Має вбудований відладчик	Має вбудований відладчик

Рисунок Б.10 – Слайд 10

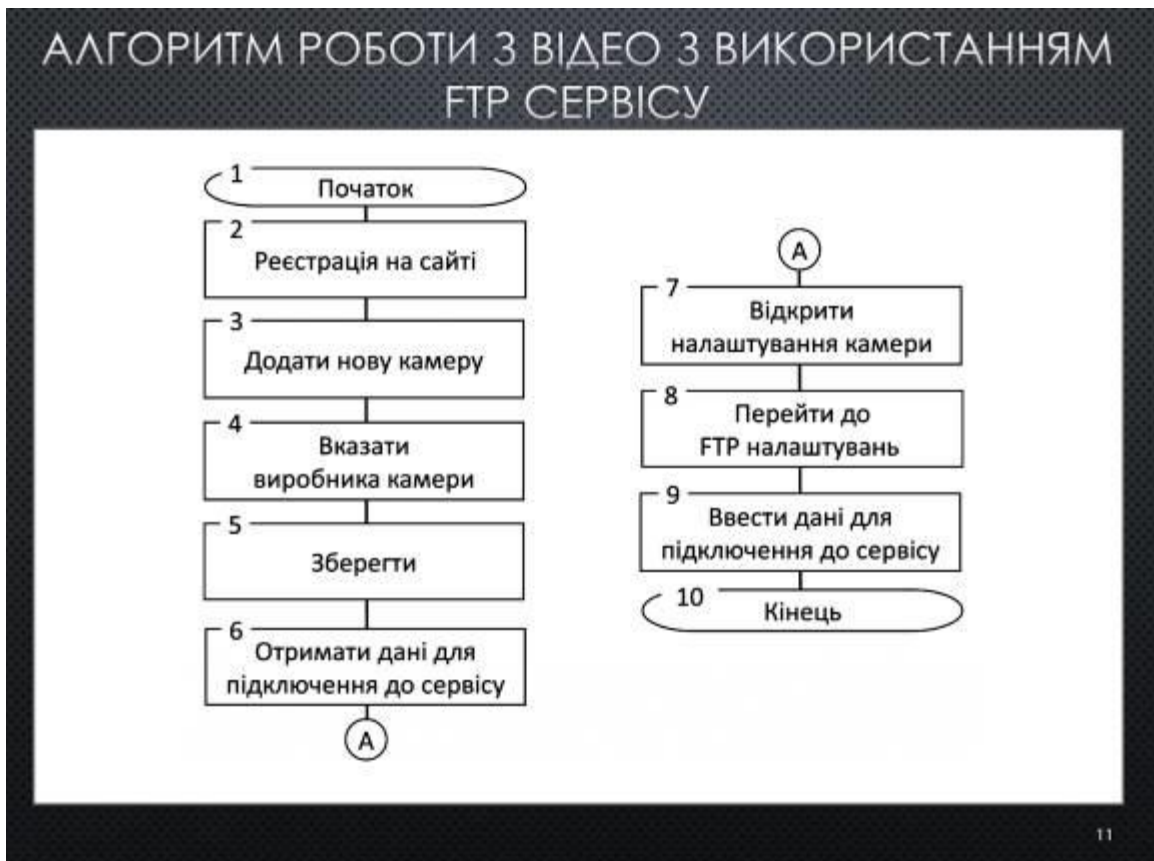


Рисунок Б.11 – Слайд 11

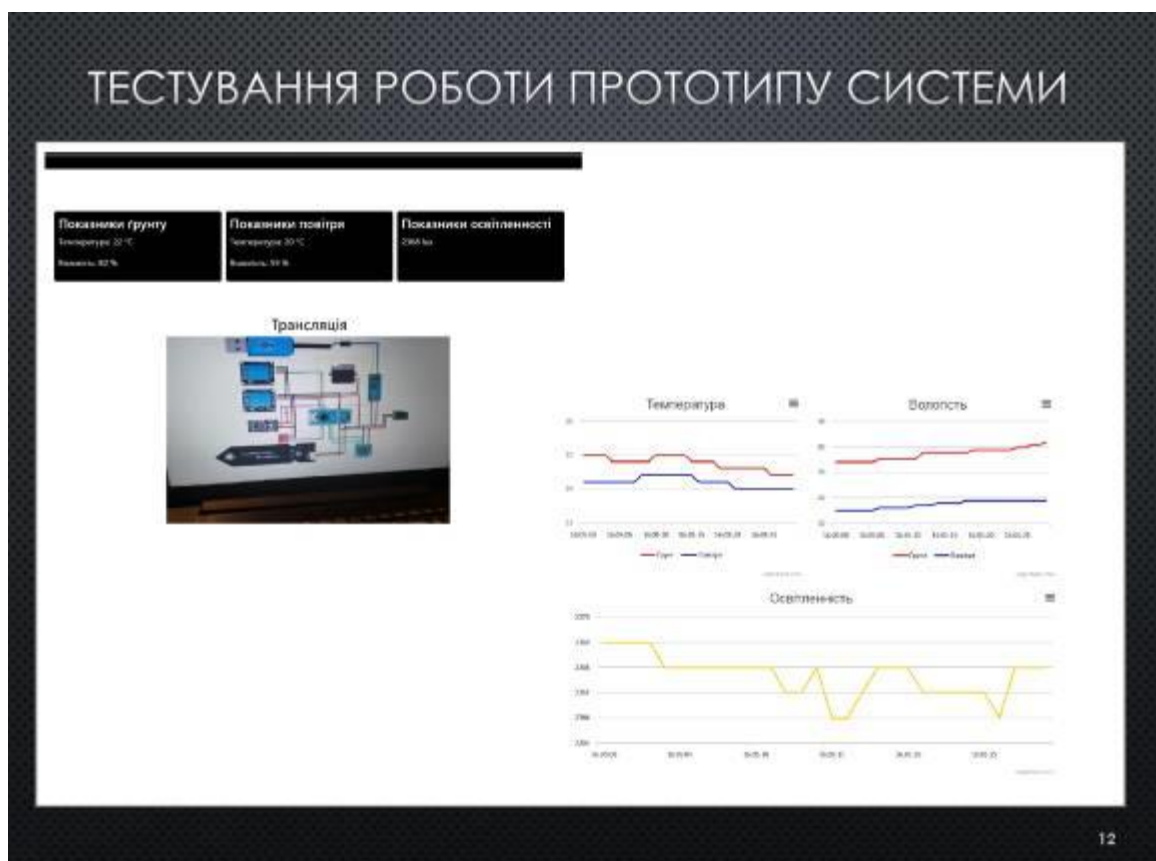


Рисунок Б.12 – Слайд 12



Рисунок Б.13 – Слайд 13

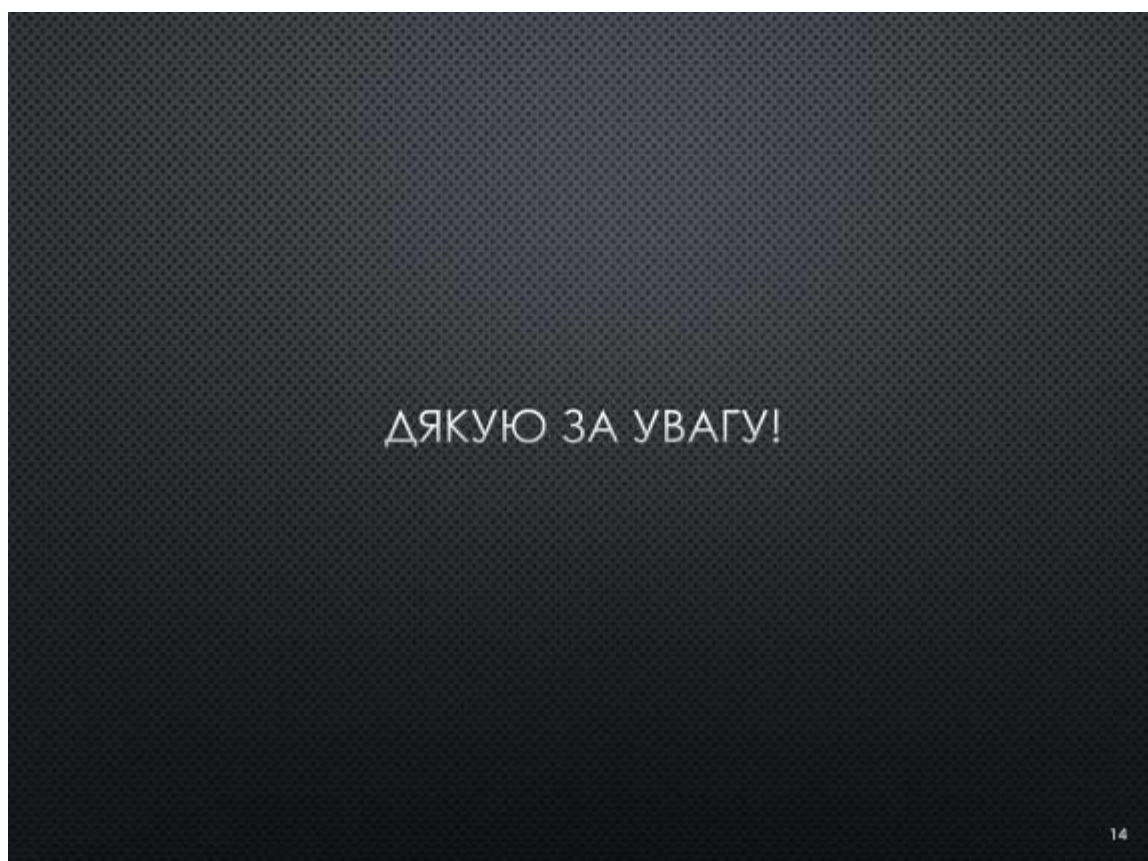


Рисунок Б.14 – Слайд 14