Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформацій

Кафедра Радіотехнологій інформаційно-комунікаційних систем

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Використання хмарних обчислень в системах збору та обробки даних з розумних пристроїв
(тема)

Виконав:
студент II курсу, групи АПСм -22-1
Малько А. Р.
(прізвище, ініціали)

Спеціальність
126 Інформаційні системи та технології
(код і повна назва спеціальності)

Тип програми
освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Архітектурне проєктування інформаційних систем
(повна назва освітньої програми)

Керівник доцент Бітченко О.М.
(посада, прізвище, ініціали)

Допускається до захисту

В.о. зав. кафедри РТІКС _____ Зарудний О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації

Кафедра Радіотехнологій інформаційно-комунікаційних систем

Рівень вищої освіти другий (магістерський)

Спеціальність 126 Інформаційні системи та технології

Тип програми Освітньо-професійна

Освітня програма Архітектурне проєктування інформаційних систем

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові **МАЛЬКУ АРСЕНІЮ РОМАНОВИЧУ**
(прізвище, ім'я, по батькові)

1. Тема роботи **ВИКОРИСТАННЯ ХМАРНИХ ОБЧИСЛЕНЬ В СИСТЕМАХ ЗБОРУ ТА ОБРОБКИ ДАНИХ З РОЗУМНИХ ПРИСТРОЇВ**

затверджена наказом по університету від **3 жовтня 2023 р**. **№ 1295 Ст**

2. Термін подання студентом роботи до екзаменаційної комісії **10 січня 2024 р.**

3. Вихідні дані до роботи _____
3.1 Сформулювати постановку задачі для подальшої розробки
3.2 Розробити пристрій збору даних з контрольованих об'єктів
3.3 В якості контрольованих параметрів обрати температуру та вологість
3.4 Розробити програму обробки та візуалізації контрольованих параметрів

4. Перелік питань, що потрібно опрацювати в роботі _____
Вступ. 1 Платформи хмарних обчислень та їх розгортання. 2 Способи створення ресурсів у хмарі. 3 Схемотехнічна розробка. 4. Програмна реалізація. Висновки. Перелік джерел посилання. Додатки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____
Комп'ютерна презентація_____
_____
_____
_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вступ	07.10-09.10.2023	
2	Платформи хмарних обчислень та їх розгортання	10.10-25.10.20253	
3	Способи створення ресурсів у хмарі	26.10-10.11.2023	
4	Схемотехнічна розробка	11.11-03.12.2023	
5	Програмна реалізація	04.12-31.12.2023	
6	Висновки	01.01.2024	
7	Оформлення пояснювальної записки	02.01-07.01.2024	
8	Оформлення ілюстрацій	7.01- 09.01.2024	
9	Представлення роботи на кафедру	10.01.2024	

Дата видачі завдання **6 жовтня 2023 р.**

Студент _____ Малько А. Р.
(підпис)

Керівник роботи _____ доц. Бітченко О.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи магістра містить 83 сторінок тексту, 23 рисунків, 22 джерел посилання.

ІНТЕРНЕТ. МІКРОКОНТРОЛЕР. ПЛАТФОРМА. ДАТЧИК. ТЕМПЕРАТУРА. ВОЛОГІСТЬ. ІНТЕРФЕЙС.

Об'єктом розробки є автоматизована система моніторингу та візуалізації параметрів розумних речей.

Метод дослідження - описово-аналітичний.

Предметом дослідження є платформа збору та обробки даних з датчиків.

Мета роботи – розробка пристрою мониторінгу температури та вологості з автоматичною передачею даних в хмару та подальшою візуалізацією.

В процесі виконання роботи було розглянуто основні платформи для роботи з інтернетом речей, розроблено пристрій моніторингу температури та вологості у приміщені. Розроблено програмне забезпечення контролю та візуалізації даних.

Результати дослідження можуть бути використані для виготовлення тестових зразків для розумного дому.

ABSTRACT

The explanatory note of the master's thesis contains 83 pages of text, 23 figures, 22 reference sources.

INTERNET. MICROCONTROLLER. PLATFORM. SENSOR. TEMPERATURE. HUMIDITY. INTERFACE.

The object of development is an automated system of monitoring and visualization of the parameters of smart things.

The research method is descriptive and analytical.

The subject of research is a platform for collecting and processing data from sensors.

The purpose of the work is to develop a temperature and humidity monitoring device with automatic data transfer to the cloud and subsequent visualization.

In the course of the work, the main platforms for working with the Internet of Things were considered, a device for monitoring temperature and humidity in the room was developed. Software for data control and visualization has been developed.

The results of the research can be used to make test samples for a smart home.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів……..….. 6
Вступ…….……. 7
1 Платформи хмарних обчислень та їх розгортання……... 9
1.1 Розумні пристрої як об’єкт хмарних технологій…….. 9
1.2 Огляд популярних платформ хмарних обчислень…… 10
1.3 Хмарні обчислення на платформі Amazon Web Services……. 11
1.4 Хмарні обчислення на платформі Microsoft Azure……... 20
1.5 Моделі розгортання та обслуговування хмарних обчислень……... 23
1.6 Віртуалізація та хмарні технології……. 29
2 Способи створення ресурсів у хмарі…… 33
2.1 Безпека хмарних ресурсів…… 36
2.2 Обробка великих даних……... 40
2.3 Архітектури традиційних інформаційних систем……. 41
3 Схемотехнічна розробка……. 46
3.1 Обґрунтування вибору мікроконтролера…….. 46
3.2 Вибір датчика вологості та температури……... 50
3.3 Вибір точки доступу до хмари……... 53
4 Програмна реалізація…….. 58
4.1 Створення каналу на платформі ThingSpeak IoT……... 58
4.2 Програмування ESP8266 і ESP32 ThingSpeak……... 63
Висновки……. 65
Перелік джерел посилання……...……. 66
Додаток А – Програмні коди…….……...……. 69
Додаток Б – Слайди презентації…….……..…… 73
Додаток В – Відомість кваліфікаційної роботи…..……..…..…… 82

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IP – Internet Protocol;

NoSQL (Amazon DynamoDB) – керована база даних;

SaaS (Software as a Service) – програмне забезпечення як сервіс;

PaaS (Platform as a Service) – платформа як сервіс;

IaaS (Infrastructure as a Service) – інфраструктура як сервіс;

BaaS (Backup as a Service) – резервне копіювання як послуга;

BPaaS (Business Process as a Service) – надання послуг з автоматизації бізнес-процесів у хмарній інфраструктурі;

CaaS (Communications as a Service) – послуга з надання засобів комунікації у хмарі;

DaaS (Desktop as a Service) – віртуальний робочий стіл під власні завдання;

DRaaS (Disaster Recovery as a Service) – аварійне відновлення як послуга;

FaaS (Function as a Service) – функція як сервіс для побудови мікросервісних безсерверних додатків;

HaaS (Hardware as a Service) – обчислювальні потужності/устаткування як послуга, тобто. оренда обладнання сервіс-провайдера.

ВСТУП

Різноманітність хмарних рішень, представлених над ринком, призвело до диверсифікації хмарних рішень, появі різних класів хмарних продуктів. Починають з'являтися спеціалізовані та галузеві хмари. Зростає частка гібридних хмар у загальному обсязі ринку хмарних обчислень. Окрему нішу на ринку хмарних обчислень зайняли хмарні послуги, які постачаються як доповнення до основного продукту IT-компаній.

Розвиток технологій Інтернету речей (IoT) і «розумних технологій» (smart technologies), у тому числі електроніка, що вбудовується і носиться, сприятимуть накопиченню великих обсягів даних з різних датчиків та «розумних речей» у режимі реального часу. Все це викличе ще більшу зацікавленість до технологій бізнес-аналітики, технологій «великих даних» (big data), машинного навчання (machine learning), глибокого навчання (deep learning).

Найближчими роками у сфері хмарних технологій продовжить активно розвиватися віртуалізація на рівні операційної системи (інша назва «хмарна контейнерна технологія»).

До сучасних тенденцій розвитку хмарних технологій можна віднести появу хмарних консорціумів, які об'єднують в пул пропозиції кількох компаній на базі однієї популярної хмарної платформи.

Очікується бурхливе зростання таких сегментів хмарних обчислень як:

- хмарна робототехніка;
- туманні обчислення;
- квантові обчислення.

За допомогою хмарних технологій можна швидко створювати та розгортати легко масштабовані бізнес-рішення в надійній IT-інфраструктурі за порівняно невисоку плату.

Метою роботи є розробка системи збору та візуалізації даних в мережах інтернету речей з застосуванням хмарних технологій.

Задачі розробки:

– провести огляд платформи хмарних обчислень та методів їх розгортання;

– розглянути способи створення ресурсів у хмарі;

– розробити схему збору даних;

– створити канал доступу до хмари;

– розробити програмне забезпечення для вимірювання та візуалізації отриманих даних.

1 ПЛАТФОРМИ ХМАРНИХ ОБЧИСЛЕНЬ ТА ЇХ РОЗГОРТАННЯ

1.1 Розумні пристрої як об’єкт хмарних технологій

Поняття «Інтернет речей» IoT (Internet of Things,) з'явилося 1999 року. Існує кілька визначень, що розкривають суть цього поняття.

За версією компанії IDC «Інтернет речей – це мережа мереж з кінцевими точками, що унікально ідентифікуються, які спілкуються між собою у двох напрямках за протоколами Internet Protocol (IP) і зазвичай без людського втручання» [1].

Фахівці компанії Gartner дають таке визначення: «Інтернет речей – це мережа фізичних об'єктів, які мають вбудовані технології, що дозволяють здійснювати взаємодію із зовнішнім середовищем, передавати відомості про свій стан та приймати дані із поза».

Існує ще більш технологічне визначення компанії McKinsey – «Інтернет речей – це датчики та приводи (виконавчі пристрої), вбудовані у фізичні об'єкти та пов'язані через дротяні або бездротові мережі з використанням протоколу IP, який пов'язує Інтернет».

Іншими словами, «Інтернет речей – це неосяжне скупчення підключених датчиків, камер, смартфонів, комп'ютерів та пристроїв, які взаємодіють із додатками, веб-сайтами, соціальними медіа та іншими пристроями. Щоб отримати з цього максимальну користь, необхідно обробляти та аналізувати дані, які генерують усі ці об'єкти у реальному часі».

Можна вивести умовну формулу, що характеризує Інтернет речей (IoT)

$$\text{IoT} = \text{Сенсори (датчики)} + \text{Дані} + \text{Мережі} + \text{Послуги}. \quad (1.1)$$

Під конкретною «Інтернет річчю (thing)» прийнято розуміти «будь-який натуральний або штучний предмет, якому може бути присвоєна IP-адреса і який має можливість передачі даних по мережі».

Прийнято виділяти рішення Інтернету речей як корпоративного сектору (B2B), так споживчого сектору (B2C).

До рішень IoT для корпоративного сектору (B2B) можна віднести:

- розумне місто (smart city);
- страхова телематика;
- розумні робочі місця;
- розумні електромережі (smart grid);
- розумні заводи (smart factory, industrial Internet, IIoT);
- точне землеробство;
- розумні свердловини (smart well);
- геолокаційний маркетинг;
- розумні склади тощо.

До рішень IoT для споживчого сектору (B2C) віносяться:

- пристрої (електроніка), що носяться;
- розумний дім (smart home);
- розумний одяг;
- smart TV;
- розумні пристрої для тварин тощо.

Очікується, що у недалекій перспективі IoT призведе до трансформації у «Всеосяжний Інтернет».

1.2 Огляд популярних платформ хмарних обчислень

В інформаційних технологіях під терміном «платформа» у широкому розумінні зазвичай розуміється сукупність компонентів: апаратного рішення, операційної системи, прикладних програмних рішень та засобів для їхньої розробки. А у вужчому сенсі виділяють програмну, прикладну та апаратну платформи.

Хмарну платформу можна розглядати як сукупність інструментів для розгортання та використання хмарних обчислень на комп'ютері користувача без додаткової установки спеціалізованого програмного забезпечення та

придбання додаткового обладнання [2].

Загалом модель архітектури хмарних обчислень можна представити так (рисунок 1.1).

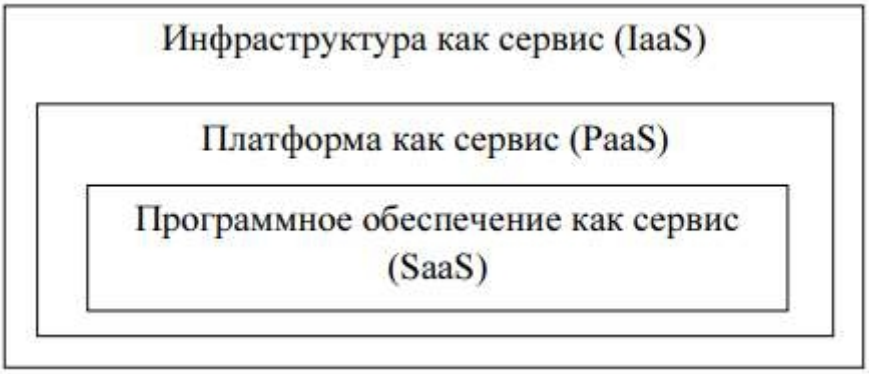

Рисунок 1.1 – Модель архітектури хмарних обчислень

Найбільш популярніші платформи хмарних обчислень наведено в таблиці 1.1.

Таблиця 1.1 – Найбільш популярні платформи хмарних обчислень

Название	IaaS	PaaS	SaaS
Amazon Web Services	+	+	+
IBM Cloud	+	+	
Google Cloud	+	+	
Microsoft Azure	+	+	
Oracle Cloud	+	+	+
App Cloud (SalesForce)		+	
SAP Cloud		+	

1.3 Хмарні обчислення на платформі Amazon Web Services

Можливості платформи хмарних обчислень Amazon Web Services (https://aws.amazon.com/ru) доступні користувачам з 2002 року. Для доступу до можливостей платформи необхідно створити обліковий запис. Для ознайомлювальних цілей є можливість створення безкоштовного облікового запису (рисунок 1.2). Однак у процесі реєстрації потрібно надати дані банківської картки (VISA, MasterCard тощо) для підтвердження вашої особи (наявність 1$

долара на рахунку для перевірки актуальності даних банківської картки). Перший рік користування хмарними сервісами буде безкоштовним, надалі, якщо їх не відключити, почне стягуватися плата.

Рисунок 1.2 – Стартова сторінка створення облікового запису на платформі Amazon Web Services

На безкоштовній основі для ознайомчого облікового запису в перший рік надаються:

– віртуальні машини тільки на базі Amazon EC2 Linux Micro Instance із сумарною тривалістю використання у 750 годин на рік;

– 750 годин використання сервісу Elastic Load Balancer із квотою в 15 Гб оброблених даних;

– 10 гігабайт у сервісі блокового сховища даних Amazon Elastic Block Storage (EBS). У безкоштовний ліміт EBS включено 1 мільйон операцій введення/виводу, 1 гігабайт для зберігання миттєвих знімків, 10000 get запитів та 1000 put запитів до цих миттєвих знімків;

– 5 Гб у системі зберігання Amazon Simple Storage Service (S3). У безкоштовний ліміт S3 включено 20 тисяч get запитів та 2 тисячі put запитів;

– 30 Гб вхідного та вихідного трафіку в рівних пропорціях;

– 25 годин роботи з Amazon SimpleDB та 1 Гб для даних;

– 100 тисяч запитів під час використання сервісу Amazon Simple Queue;

– 100 тисяч запитів, 100 тисяч HTTP повідомлень та 1000 повідомлень

електронною поштою при використанні Amazon Simple Notification;

– 10 метрик, 10 попереджень та 1 млн. запитів до API у сервісі Amazon Cloudwatch.

Особливістю хмарної платформи є відсутність можливості обмежити витрати на місяць якоюсь фіксованою сумою, після досягнення якої робота сервісів блокувалася б. У той же час можна налаштувати попередження про перевитрату коштів. Наприклад, з використанням сервісу CloudWatch, який спрогнозує майбутні витрати на основі поточних витрат, і піднімає тривогу, якщо вони помітно перевищать встановлену межу.

Хмарна платформа Amazon Web Services надає багатий вибір функціональних можливостей, згрупованих у декілька категорій.

Обчислення:

– віртуальні сервери у хмарі (Amazon EC2);

– зберігання та вилучення образів Docker (Amazon Elastic Container Service);

– запуск веб-застосунків та керування ними (AWS Elastic Beanstalk);

– перегляд, встановлення та публікація безсерверних програм (AWS Serverless Application Repository);

– запуск контейнерів Docker та керування ними (Amazon Elastic Container Registry);

– запуск приватних віртуальних серверів та управління ними (Amazon Lightsail);

– балансування навантаження (Elastic Load Balancing);

– автоматизація масштабування (Auto Scaling);

– запуск контейнерів без керуючих серверів або кластерів (AWS Fargate);

– запуск контейнерів для Kubernetes на AWS (Amazon Elastic Container Service for Kubernetes);

– запуск пакетних завдань будь-якого масштабу (AWS Batch);

– запуск власного коду за певних подій (AWS Lambda);

– створення гібридної хмари без спеціального обладнання (хмара VMware у AWS).

Сховище:

– сховище даних, що масштабується (Amazon Simple Storage Service, S3);

– блочне сховище (Amazon Elastic Block Storage, EBS);

– кероване файлове сховище (Amazon Elastic File System, EFS);

– недороге архівне сховище (Amazon Glacier);

– інтеграція гібридних сховищ (AWS Storage Gateway);

– сервіс передачі петабайтів даних (AWS Snowball);

– сервіс передачі петабайтів даних із використанням вбудованих обчислювальних ресурсів (AWS Snowball Edge);

– сервіс передачі ексабайтів даних (AWS Snowmobile).

База даних:

– керована реляційна база даних із високою продуктивністю (Amazon Aurora);

– керований сервіс реляційних баз даних для MySQL, PostgeSQL, SQL Server та Maria (Amazon RDS);

– керована база даних NoSQL (Amazon DynamoDB);

– система кешування у пам'яті (Amazon ElastiCache);

– сервіс сховища даних, сумісний із наявними інструментами бізнес-аналітики (Amazon Redshift);

– сервіс оптимізованих для прикладів використання графових баз даних (Amazon Neptune);

– сервіс для міграції баз даних із мінімальним часом простою (AWS Database Migration Service).

Міграція:

– централізоване відстеження міграцій (AWS Migration Hub);

– сервіс збору даних про IT-інфраструктуру корпоративних клієнтів

для подальшої міграції в хмару (AWS Application Discovery Service);

– сервіс міграції баз даних у хмару (AWS Database Migration Service);

– сервіс міграції локальних серверів у хмару (AWS Server Migration Service);

– сервіс передачі великих обсягів даних як у хмару і з нього (AWS Snowball);

– сервіс передачі петабайтів даних з використанням будованих обчислювальних ресурсів (AWS Snowball Edge);

– сервіс для швидкого фізичного перенесення даних AWS Snowmobile за допомогою місткого накопичувача інформації (80 Тб) з встановленими програмами для шифрування та захисту даних, що надсилається клієнту. Після завершення запису клієнт відправляє диск назад до Amazon для запису у «хмару».

Мережева конфігурація та доставка контенту:

– ізольовані хмарні ресурси (Amazon VPC);

– глобальна мережа доставки контенту (Amazon CloudFront);

– система доменних імен, що масштабується (Amazon Route 53);

– розробка, розгортання та керування API (Amazon API Gateway);

– організація виділених мережних підключень (AWS Direct Connect);

– балансування навантаження у великих масштабах (Elastic Load Balancing);

Інструменти для розробників:

– розробка та розгортання програм (AWS CodeStar);

– зберігання коду у приватних репозиторіях Git (AWS CodeCommit);

– складання та тестування коду (AWS CodeBuild);

– автоматизоване розгортання коду (AWS CodeDeploy);

– сервіс безперервної інтеграції та безперервної доставки для швидкого та безвідмовного оновлення програм та інфраструктури (AWS CodePipeline);

– створення, запуск та налагодження коду у хмарній IDE (AWS

Cloud9);

– аналіз та налагодження програм (AWS X-Ray);

– пакети SDK та інструменти AWS;

– інтерфейс командного рядка AWS.

Інструменти керування:

– моніторинг ресурсів та програм (Amazon CloudWatch);

– шаблони для створення ресурсів та управління (AWS CloudFormation);

– відстеження активності користувачів та використання API (AWS CloudTrail);

– облік ресурсів та відстеження змін (AWS Config);

– автоматизація роботи (AWS OpsWorks);

– керовані послуги (AWS Service Catalog);

– централізований збір операційних даних із різних сервісів та автоматизація завдань (AWS Systems Manager);

– оптимізація продуктивності та безпеки (AWS Trusted Advisor);

– консоль керування AWS (AWS Personal Health Dashboard).

Сервіси мультимедіа:

– масштабований сервіс для перетворення мультимедійного контенту (Amazon Elastic Transcoder);

– сервіс для обробки та аналізу відеопотоків (Amazon Kinesis Video Streams);

– сервіс перетворення файлового відеоконтенту (AWS Elemental MediaConvert);

– сервіс перетворення відеоконтенту у реальному часі (AWS Elemental MediaLive);

– сервіс підготовки та стиснення відео (AWS Elemental MediaPackage);

– сервіс зберігання мультимедійних даних (AWS Elemental MediaStore);

– сервіс для додавання до відеоконтенту індивідуально таргетованої реклами (AWS Elemental MediaTailor).

Безпека, ідентифікація та відповідність вимогам:

– керування доступом та ключами шифрування (AWS Identity and Access Management, IAM);

– сервіс створення хмарних каталогів для зберігання ієрархічно пов'язаних даних (Amazon Cloud Directory);

– управління ідентифікацією, авторизацією та контролем доступу користувачів до мобільних та інтернет-додатків (Amazon Cognito);

– обслуговування виявлення загроз на основі аналізу логів подій (Amazon GuardDuty);

– автоматизований сервіс оцінки безпеки, який дозволяє забезпечити відповідність вимогам та підвищити рівень безпеки програм (Amazon Inspector);

– сервіс безпеки на основі технологій машинного навчання для автоматичного виявлення, класифікації та захисту конфіденційних даних (Amazon Macie);

– сервіс для надання, розгортання сертифікатів SSL/TLS для використання разом із сервісами AWS (AWS Certificate Manager);

– хмарний керований апаратний модуль безпеки (HSM), який дозволяє легко генерувати та використовувати у хмарі AWS власні ключі шифрування. (AWS CloudHSM);

– розміщення каталогу Active Directory та керування ним (AWS Directory Service);

– сервіс для створення та керування ключами шифрування (AWS Key Management Service);

– сервіс управління безліччю облікових записів AWS на основі політик (AWS Organizations);

– сервіс аутентифікації за технологією «єдиного входу» (AWS Single

Sign-On);

– захист від DDoS – атак (AWS Shield);

– фільтрація веб-трафіку (AWS WAF);

– звіти з безпеки та дотримання SLA (AWS Artifact).

Аналітика:

– інтерактивний сервіс запитів, що дозволяє аналізувати дані в Amazon S3 стандартними засобами SQL (Amazon Athena);

– розміщене середовище Hadoop (Amazon EMR);

– сервіс, що забезпечує пошукові рішення для веб-сайтів та програм (Amazon CloudSearch);

– сервіс, що надає прості API та можливості аналітики в режимі реального часу (Amazon Elasticsearch Service);

– сервіс збору, обробки та аналізу потокових даних у режимі реального часу (Amazon Kinesis);

– високошвидкісне сховище даних (Amazon Redshift);

– сервіс візуалізації результатів аналізу "на льоту" (Amazon QuickSight);

– веб-сервіс, що допомагає надійно та із заданими інтервалами обробляти дані та переміщувати їх між різними обчислювальними сервісами та сервісами сховищ AWS, а також локальними джерелами даних (AWS Data Pipeline);

– сервіс AWS Glue для вирішення задач ETL (вилучення, перетворення та завантаження даних).

Машинне навчання:

– сервіс для створення, навчання та розгортання моделей машинного навчання (Amazon Sagemaker);

– обслуговування обробки природної мови з використанням технології машинного навчання (Amazon Comprehend);

– сервіс Amazon Lex щодо створення голосових та текстових

діалогових інтерфейсів (ботів);

– сервіс, що перетворює текст на природне мовлення (Amazon Polly);

– обслуговування розпізнавання об'єктів, осіб, людей, тексту, сцен і дій (Amazon Rekognition) на основі технології глибокого навчання;

– послуги машинного навчання (Amazon Machine Learning);

– сервіс онлайн перекладу (Amazon Translate);

– обслуговування автоматичного розпізнавання мови (Amazon Transcribe);

– програмована відеокамеру, навчальні посібники, код та заздалегідь навчені моделі, створені для розширення навичок роботи з глибоким навчанням (AWS DeepLens);

– образи віртуальних машин для освоєння технології глибокого навчання (AWS Deep Learning AMI);

– масштабована система навчання та отримання логічних висновків із простим у використанні компактним API для машинного навчання (Apache MXNet на AWS);

– бібліотека штучного інтелекту з відкритим кодом (TensorFlow на AWS).

Мобільні послуги:

– набір сервісів для розробки мобільних програм AWS Mobile Hub);

– сервіс для створення, публікації, обслуговування, моніторингу та забезпечення безпеки API у будь-яких масштабах (Amazon API Gateway);

– сервіс для надсилання електронної пошти, SMS-повідомлень та мобільних push-повідомлень, у тому числі таргетованих повідомлень (Amazon Pinpoint);

– сервіс оновлення даних у мобільних та інтернет-додатках у режимі реального часу (AWS AppSync);

– сервіс тестування програм для Android, iOS, Інтернету, що емулюють одночасну роботу кількох пристроїв (AWS Device Farm);

– інструменти розробника (AWS Mobile SDK).

Інтернет речей:

– підключення пристроїв до хмари (AWS IoT Core);

– операційна система IoT для мікроконтролерів (Amazon FreeRTOS);

– локальні обчислення, надсилання повідомлень та синхронізація пристроїв (AWS Greengrass);

– створення тригерів AWS Lambda для IoT (AWS IoT 1-Click);

– аналітика для пристроїв IoT (AWS IoT Analytics);

– програмована хмарна кнопка Dash Button (AWS IoT Button – простий у налаштуванні пристрій з підключенням по Wi-Fi, який покликаний допомогти розробникам розпочати роботу з AWS IoT Core, AWS Lambda, Amazon DynamoDB, Amazon SNS та багатьма іншими сервісами Amazon Web Services без написання) коду для конкретного пристрою;

– керування безпекою для пристроїв IoT (AWS IoT Device Defender);

– сервіс для безпечного підключення, керування та моніторингу підключених пристроїв IoT (AWS IOT Device Management).

1.4 Хмарні обчислення на платформі Microsoft Azure

Microsoft (Windows) Azure надає можливість розробки та виконання програм та зберігання даних на серверах, розташованих у розподілених дата-центрах. Спочатку називалася Windows Azure. У 2014 році платформа була перейменована на Microsoft Azure [3].

Microsoft Azure реалізує дві хмарні моделі обслуговування – платформа як сервіс (PaaS) та інфраструктура як сервіс (IaaS). Працездатність платформи Windows Azure забезпечує мережу глобальних дата-центрів Microsoft.

В основі роботи Microsoft Azure лежить запуск віртуальної машини для кожного екземпляра програми. Розробник визначає необхідний обсяг зберігання даних і необхідні обчислювальні потужності (кількість віртуальних машин), після чого платформа надає відповідні ресурси. Коли початкові потреби

в ресурсах змінюються, відповідно до нового запиту замовника платформа виділяє додаткові додатки або скорочує невикористовувані ресурси дата-центру.

Microsoft Azure як PaaS забезпечить не тільки всі базові функції операційної системи, але й додаткові: виділення ресурсів на вимогу для необмеженого масштабування, автоматичну синхронну реплікацію даних для підвищення стійкості до відмов, обробку відмов інфраструктури для забезпечення постійної доступності та багато іншого.

Для розробника найбільший інтерес можуть мати такі можливості хмарної платформи Microsoft Azure:

– можливість здійснити перенесення до хмар пакетних та кластерних додатків;

– можливість створення привабливих програм для iOS, Android та Windows;

– легка інтеграція хмарних програм Azure з соціальними мережами Facebook, Twitter, Google;

– автоматизація керування даними у масштабі підприємства з використанням гібридного хмарного сховища StorSimpe;

– підтримка Microsoft SQL Server та SharePoint Server, повна інтеграція з усіма документами Office;

– можливість роботи з базами даних SQL Azure, Oracle, MySQL, Redis, MongoDB;

– петабайти сховища даних та гео-надлишкове сховище на відстані в сотні кілометрів;

– робота з DocumentDB. База даних NoSQL JSON дозволяє розробляти програми швидко та ітераційно;

– можливість розгортання додатків та інтерфейсів API з високою доступністю та нескінченною масштабованістю;

– виклик кінцевих точок веб-служб за протоколом HTTP/HTTPS;

– підтримка Java, Node.js, PHP, Python, .NET та Ruby;

– підтримка REST, Java, C++, PowerShell;

– підтримка WordPress, Umbraco, Joomla, Drupal та багато іншого;

– забезпечення безпеки даних шляхом шифрування під час передачі та на місці;

– єдиний вхід у всі хмарні сховища та локальні веб-додатки;

– попередня інтеграція із Salesforce.com, Office 365, Box та іншими рішеннями;

– примусове застосування багатофакторної автентифікації з використанням SaaS;

– надсилання push-повідомлень на будь-яку платформу (iOS, Android, Windows, Kindle) з будь-якого сервера за допомогою концентратора повідомлень;

– інтеграція програм на базі служб Biztalk для рішень класу EDI, B2B, EAI, IoT;

– забезпечення зв'язку додатків та пристроїв у приватних та загальнодоступних хмарних середовищах;

– створення надійних та гнучких хмарних додатків з функцією обміну повідомленнями;

– захист додатків від тимчасового пікового навантаження;

– розповсюдження повідомлень з різних незалежних серверних систем;

– служби для команд розробників з обміну кодом, відстеження версій робочих файлів проекту з урахуванням Visual Studio TeamSystem;

– інтегровані функції відстеження працездатності, моніторингу та балансування навантаження розроблених та розгорнутих у хмарі додатків.

Microsoft Azure як IaaS надає користувачеві готову інфраструктуру, коли з'являється можливість оренди таких ресурсів, як сервери, пристрої зберігання даних та мережеве обладнання. Управління всією інфраструктурою здійснюється постачальником, споживач керує лише операційною системою та встановленими програмами.

Зовнішній вигляд інформаційного порталу для адміністрування облікового запису Microsoft Azure представлений на рисунку 1.3.

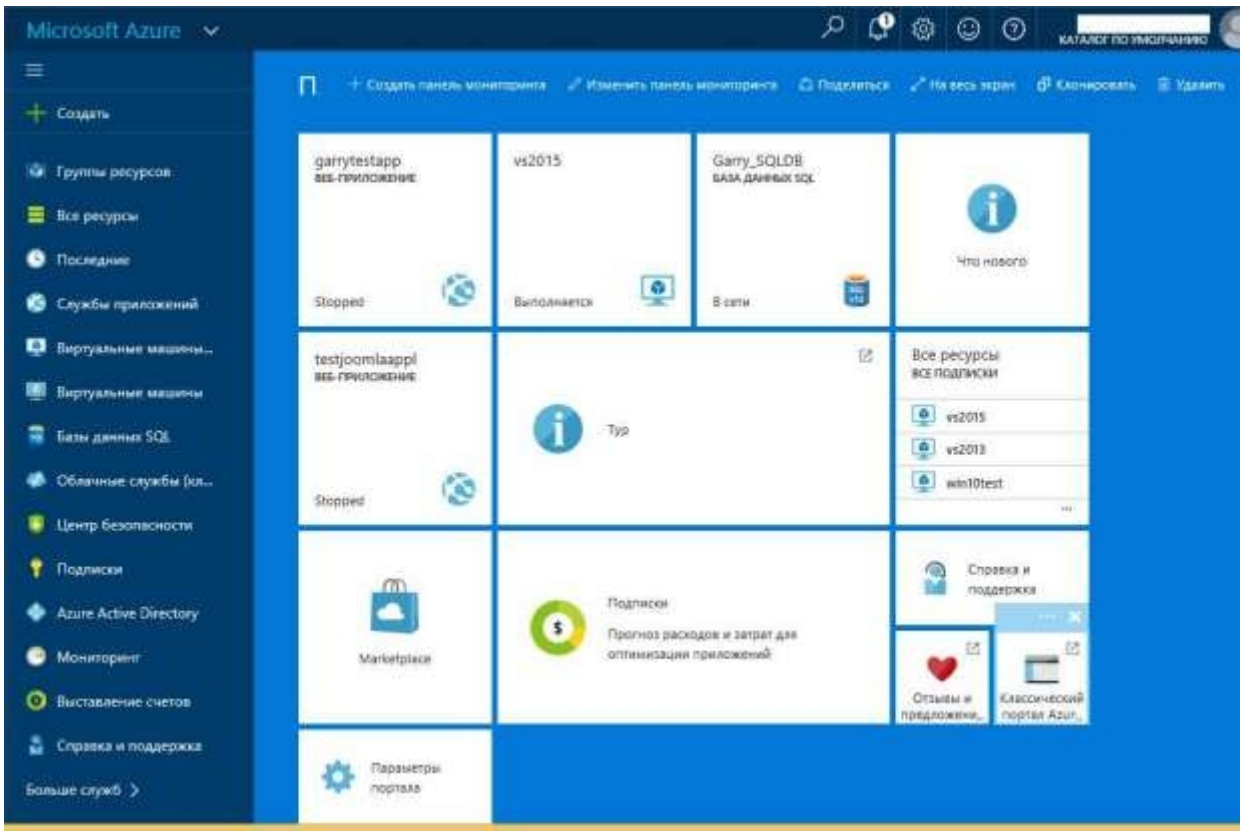

Рисунок 1.3 – Microsoft Azure. Зовнішній вигляд порталу

В основі Windows Azure лежать багато в чому ті ж технології, що і в серверній операційній системі Windows Server, у тому числі:

– Active Directory (автентифікація);

– Hyper-V (віртуалізація);

– System Center (управління);

– Visual Studio .NET (інструмент та середовище розробки).

Тому, працюючи в Windows Azure, IT-спеціалісти можуть повною мірою використати свій Windows-досвід і з мінімальними зусиллями трансформувати вже існуючі програми в сучасні хмарні сервіси.

1.5 Моделі розгортання та обслуговування хмарних обчислень

"Хмарні обчислення" (від англ. cloud computing) – це обчислення, які є динамічно масштабований спосіб доступу до зовнішніх обчислювальних ресурсів у вигляді сервісу, що надається за допомогою Інтернету [4].

За іншим визначенням «хмарні обчислення» – це модель забезпечення повсюдного та зручного мережного доступу на вимогу до загального пулу конфігурованих обчислювальних ресурсів (мереж передачі даних, серверів, пристроїв зберігання даних, додатків та IT - сервісів – як разом, так і окремо), які можуть бути оперативно надані та звільнені з мінімальними витратами та/або зверненнями до провайдера.

Різні моделі хмарних обчислень базуються на використанні технологій віртуалізації, які дають змогу відокремити «фізичну» складову від «логічної», абстрагуючи використання ресурсу від складнощів фактичного апаратного забезпечення. Стосовно серверної інфраструктури, віртуалізація дозволяє об'єднати ресурси серверів (пам'ять, процесор, дисковий простір) в одну велику хмару ресурсів з подальшим створенням віртуальної машини або машин необхідної замовнику конфігурації.

Для хмарних обчислень Національним інститутом стандартів та технологій США (National Institute of Standards and Technology, NIST, https://www.nist.gov) зафіксовано такі обов'язкові характеристики:

– самообслуговування на вимогу (англ. self service on demand), споживач самостійно визначає та змінює обчислювальні потреби, такі як серверний час, швидкості доступу та обробки даних, обсяг даних, що зберігаються без взаємодії з представником постачальника послуг;

– універсальний доступ по мережі; послуги доступні споживачам по мережі передачі даних незалежно від термінального пристрою, що використовується;

– об'єднання ресурсів (англ. resource pooling), постачальник послуг об'єднує ресурси обслуговування великої кількості споживачів у єдиний пул для динамічного перерозподілу потужностей між споживачами за умов постійного зміни попиту потужності; при цьому споживачі контролюють лише основні параметри послуги (наприклад, обсяг даних, швидкість доступу), але фактичний розподіл ресурсів, що надаються споживачеві, здійснює постачальник (у деяких випадках споживачі таки можуть керувати деякими фізичними

параметрами перерозподілу, наприклад, вказувати бажаний центр обробки даних з міркувань географічної близькості);

– еластичність, послуги можуть бути надані, розширені, звужені будь-якої миті часу, без додаткових витрат за взаємодію Космосу з постачальником, зазвичай, у автоматичному режимі;

– облік споживання, постачальник послуг автоматично обчислює спожиті ресурси на певному рівні абстракції (наприклад, обсяг даних, що зберігаються, пропускна спроможність, кількість користувачів, кількість транзакцій), і на основі цих даних оцінює обсяг наданих споживачам послуг.

Використання хмарних технологій потребує спеціалізованої IT-інфраструктури, розгортання центру обробки даних (ЦОД).

Прийнято виділяти такі моделі розгортання «хмарних середовищ» з урахуванням ЦОД [5; 6]:

– приватна (закрита) хмара (private cloud);

– публічна (відкрита) хмара (public cloud), яку використовує «хмарний» провайдер для надання зовнішнім замовникам сервісів хмарної структури;

– громадська хмара (community cloud), призначена для використання конкретним співтовариством споживачів із організацій, які мають спільні завдання. Може перебувати у кооперативній (спільній) власності, управлінні та експлуатації однієї чи кількох організацій спільноти або у третьої сторони;

– змішана (або гібридна) хмара (hybrid cloud), що використовується спільно два перерахованих вище варіанти розгортання.

"Приватна хмара" (від англ. private cloud) – інфраструктура, призначена для використання однією організацією, що включає кілька споживачів (наприклад, підрозділів однієї організації), можливо також клієнтами та підрядниками цієї організації. Приватна хмара може перебувати у власності, управлінні та експлуатації як самої організації, так і третьої сторони (або будь-якої їхньої комбінації) [7].

"Публічна хмара" (від англ. Public cloud) – інфраструктура, призначена для вільного використання широкою публікою. Публічна хмара може

перебувати у власності, управлінні та експлуатації комерційних, наукових та урядових організацій (або будь-якої їхньої комбінації). Публічна хмара фізично існує у юрисдикції власника - постачальника послуг.

"Гібридна хмара" (від англ. hybrid cloud) – це комбінація з двох або більше різних хмарних інфраструктур (приватних, публічних або громадських), що залишаються унікальними об'єктами, але пов'язані між собою стандартизованими або приватними технологіями передачі даних та додатків (наприклад, короткочасне використання) ресурсів публічних хмар для балансування навантаження між хмарами).

Набули поширення такі моделі обслуговування:

– програмне забезпечення як сервіс (Software as a Service, SaaS), що передбачає [8], що споживачеві надається можливість використання прикладного програмного забезпечення провайдера, що працює у хмарній інфраструктурі;

– платформа як сервіс (Platform as a Service, PaaS), в якій контроль та управління основною фізичною та віртуальною інфраструктурою хмари здійснюється хмарним провайдером, за винятком розроблених або встановлених програм, а також, по можливості, параметрів конфігурації середовища (платформи);

– інфраструктура як сервіс (Infrastructure as a Service, IaaS). У моделі обслуговування IaaS споживач може контролювати операційні системи, віртуальні системи зберігання даних та встановлені програми, а також обмежений контроль набору доступних сервісів. Контроль та управління основною фізичною та віртуальною інфраструктурою хмари здійснюється хмарним провайдером;

Також можуть зустрітися такі згадки різновидів хмарних послуг:

– BaaS (Backup as a Service) – резервне копіювання як послуга;

– BaaS (Backend as a Service) – набір готової серверної функціональності, який дозволяє спростити та прискорити розробку додатків (наприклад, повноцінне середовище розробки, розміщене у хмарі);

– BPaaS (Business Process as a Service) – надання послуг з автоматизації бізнес-процесів у хмарній інфраструктурі. Доцільні при автоматизації типових робіт, що повторюються, різними подібними за профілем замовниками;

– CaaS (Communications as a Service) – послуга з надання засобів комунікації у хмарі (телефонія, передача SMS, відеоконференції силами сервіс-провайдера);

– CaaS (Container as a Service) – контейнер як послуга. Полягає у наданні клієнту можливості організувати, запустити чи зупинити контейнер за допомогою веб-інтерфейсу чи засобів API;

– CaaS (Car as a Service) - машина (автомобіль) як сервіс із щохвилини оплатою використання;

– DaaS (Desktop as a Service) – віртуальний робочий стіл під власні завдання. Дозволяє використовувати програми різного рівня без прив'язки до можливостей свого настільного ПК;

– DaaS (Data as a Service) – дані як сервіс. Інша назва – Dbaas (Database as a Service). У даній моделі обслуговування можлива реалізація трьох основних сценаріїв: кожному користувачеві виділяється окрема база даних із власними схемами даних, може бути надана база даних у спільне користування з поділом доступу на основі різних схем, а також можливий варіант із спільно використовуваною базою даних із спільно використовуваною схемою [9]. Для реалізації останнього варіанта використовуються спеціальні налаштування;

– DRaaS (Disaster Recovery as a Service) – аварійне відновлення як послуга. передбачає можливість відновлення працездатності власної віртуальної інфраструктури у хмарі сервіс-провайдера у разі аварії чи катастрофи;

– FaaS (Function as a Service) – функція як сервіс для побудови мікросервісних безсерверних додатків;

– HaaS (Hardware as a Service) – обчислювальні потужності/устаткування як послуга, тобто. оренда обладнання сервіс-провайдера;

– IDaaS (Identity as a Service) - ідентифікація як сервіс для управління обліковими даними та доступу до роботи як локально, так і в хмарній інфраструктурі;

– MaaS (Monitoring as a Service) – моніторинг як послуга. Відстеження подій та стану у власній інфраструктурі за допомогою програмних засобів, розміщених у хмарі сервіс-провайдера;

– NaaS (Network as a Service) – мережа як послуга. Послуга передбачає надання мережевої інфраструктури, інструментів маршрутизації;

– SECaaS (Security as a Service) – безпека як послуга (перевірка антивірусом мережевого трафіку; безпека електронного листування, шифрування, ін. послуги інформаційної безпеки в хмарній інфраструктурі);

– STaaS (Storage as a Service) – дискове сховище як послуга. Найчастіше входить до складу інших послуг (IaaS, SaaS, PaaS);

– SEaaS (SEnsor as a Service) – дані з датчиків та сенсорів як послуга. Активно розвивається разом із технологією Інтернет речей (IoT);

– WaaS (Workspace as a Service) – робоче оточення як послуга або надання комплекту програмного забезпечення моделі SaaS, призначеного для створення робочого місця, переважно орієнтованого на роботу з офісними пакетами програм;

– EaaS (Everything as a Service) – концептуальна модель, що включає елементи всіх перелічених рішень. На даний момент повної її реалізації не існує. Ідеал для великих хмарних компаній.

У таблиці 1.2 представлені переваги та обмеження при використанні хмарних обчислень.

Таблиця 1.2 – Переваги та обмеження при використанні технологій хмарних обчислень

Переваги хмарних технологій	Обмеження хмарних технологій
Гнучка система оплати за спожиті ресурси (принцип "pay-as you-go")	Інформаційна безпека. Не кожна компанія готова довірити конфіденційні дані хмарному провайдеру
Користувачеві не потрібно купувати дороге обладнання та економія площ, які займає обладнання	Необхідність у наявності постійного високошвидкісного з'єднання
Обсяг ресурсів у хмарному середовищі практично обмежений вартістю, яку готовий заплатити користувач	Залежність від хмарного провайдера та його цінової політики

Продовження таблиці 1.2

Переваги хмарних технологій	Обмеження хмарних технологій
У більшості сценаріїв немає додаткових витрат на розгортання та обслуговування складної інфраструктури	Функціональність хмарних програм не завжди збігається з їхньою настільною версією
Технології доступні з будь-якої точки світу, де є доступ до Інтернету	Не кожна хмарна програма дозволяє зберегти отримані результати у зручному для користувача вигляді або форматі
Динамічна масштабованість по вимогою	Завжди існує ризик втрати або крадіжки даних у разі виникнення проблем на стороні хмарного провайдера
Надійність, гарантована хмарним провайдером у рамках угоди про рівень сервісу (SLA)	При переході від одного хмарного провайдера до іншого можуть виникнути технічні проблеми через різні технології або технічні налаштування.
За допомогою хмарних технологій можна моделювати технічну реалізацію різних проектів без придбання необхідного обладнання та програмного забезпечення	

1.6 Віртуалізація та хмарні технології

Хмарні технології базуються на використанні технологій віртуалізації та передбачають віртуалізацію серверів, систем зберігання, мереж тощо.

У спрощеному розумінні, якщо користувач на персональному комп'ютері запускає за допомогою гіпервізора одну або кілька віртуальних машин (у середовищі Microsoft Virtual PC або Hyper-V, Oracle VirtualBox, VMware або ін.) і потім з ними працює, то в певному сенсі можна говорити про роботі з локальною (або приватною) хмарою.

Альтернативним рішенням є IBM Cloud Private для організації локальної приватної хмари, схожої за функціональністю з публічною.

У тому випадку, якщо віртуальні машини запущені у провайдера і користувач має віддалений доступ до них, має місце робота з публічною хмарою.

Гібридна хмара - це поєднання мінімум однієї публічної хмари з не менш

ніж однією приватною хмарою.

Як приклад, можна згадати набір засобів Microsoft Azure Stack для створення гібридної інфраструктури.

Тепер зупинимося докладніше на технічних аспектах, пов'язаних із віртуалізацією.

В інформаційних технологіях під терміном «віртуалізація» зазвичай розуміється абстракція обчислювальних ресурсів (найчастіше серверів) та надання користувачеві системи, яка «інкапсулює» (приховує в собі) власну реалізацію [10].

Інше визначення свідчить: «Віртуалізація – це технологія, що дозволяє надавати обчислювальні ресурси, абстраговані від апаратної частини та при цьому логічно ізольовані один від одного. Тобто на одному фізичному сервері можна створити багато віртуальних, які працюватимуть незалежно».

Вирізняють такі види віртуалізації:

- для операційної системи;
- програмна;
- апаратна;
- для додатків;
- віртуалізація програм;
- віртуалізація сервісів;
- для пам'яті;
- віртуалізація пам'яті;
- віртуальна пам'ять);
- для сховища даних;
- віртуалізація зберігання даних (блочна/файлова);
- розподілена файлова система;
- віртуальна файлова система;
- гіпервізор зберігання даних;
- віртуалізація пристроїв зберігання даних;

– для бази даних
– віртуалізація даних;
– віртуалізація баз даних;
– для мережі;
– віртуалізація мережі;
– приватна віртуальна мережа.

За іншою класифікацією виділять такі види віртуалізації:

– віртуалізація платформ. Продуктом цього виду віртуалізації є віртуальні машини – деякі програмні абстракції, які запускаються на платформі реальних апаратно-програмних систем;

– віртуалізація ресурсів. Даний вид віртуалізації має на меті комбінування або спрощення представлення апаратних ресурсів для користувача і отримання деяких абстракцій користувача обладнання, просторів імен, мереж тощо.

Під гіпервізором розуміють програму, яка дозволяє реалізувати технологію віртуалізації, забезпечує керування та налаштування віртуальних машин, а також мереж, програмних комутаторів та маршрутизаторів.

"Віртуальна машина" - аналог фізичного комп'ютера, реалізований у віртуальному середовищі.

Перерахуємо найбільш популярні платформи, що використовують апаратну віртуалізацію:

– IBM LPAR;
– VMware;
– Hyper-V;
– Xen;
– KVM;
– Bhyve.

Найбільш поширеними технологіями віртуалізації є Microsoft Hyper-V та VMware Virtual Infrastracture. Власні технології віртуалізації мають компанії

Cisco, Oracle, Citrix, Hewlett Packard, Intel, AMD.

До переваг використання технологій віртуалізації належать:

– раціональність використання обчислювальних ресурсів (замість кількох легко завантажених серверів можна використовувати один із максимальним навантаженням, економія електроенергії, скорочення витрат на серверне обладнання);

– економія на додатковій інфраструктурі (системи охолодження тощо);

– зниження витрат на програмне забезпечення (за деякими програмами ліцензування одна ліцензія на операційну систему дає право використовувати її не лише на одному фізичному сервері, а ще й на кількох віртуальних);

– гнучкість перерозподілу ресурсів фізичної машини між декількома віртуальними за необхідності;

– можливість працювати з кількома різними операційними системами та версіями додатків у рамках однієї фізичної машини;

– гнучкість міграції віртуальних машин між фізичними комп'ютерами з допомогою сучасних засобів адміністрування;

– висока стійкість до відмов та простота відновлення після збоїв (за рахунок миттєвих знімків образів віртуальної машини);

– простота налаштування віртуальних машин та велика керованість IT - інфраструктурою.

2 СПОСОБИ СТВОРЕННЯ РЕСУРСІВ У ХМАРІ

Хмарні провайдери мають в основі своїх сервісів величезні дата-центри, обчислювальні ресурси яких за допомогою системи віртуалізації поділяються на невеликі частини: голі віртуальні машини різних розмірів із встановленою операційною системою (IaaS) та групи віртуальних машин із встановленим софтом, що надає доступ лише до своїх можливостей (PaaS) [11]. Так ось, здати хмарний ресурс - означає надіслати запит контролеру ресурсів, розміщеному в хмарному ЦОДі, на виділення необхідних обчислювальних ресурсів з пулу доступних. Тобто, по суті, ресурс не створюється з нічого, а лише виділяється на вимогу. І тут можлива ситуація (рідко, але буває), коли користувач попросив ресурси у контролера, а вони не з'явилися. Це трапляється через те, що фізичні ресурси, де розміщуються віртуальні, вже задіяні іншими користувачами. Завдання оптимального розподілу доступних ресурсів між користувачами повністю вирішує контролер. І якщо користувач при створенні ресурсів зіткнувся з проблемою, він повинен повторити спробу, вдавшись до різних варіацій (повторити через деякий час, змінити регіон і повторити, змінити обліковий запис і повторити тощо) [12].

Існує чотири способи керування хмарною інфраструктурою (рисунок 2.1). Перший, найпростіший і очевидніший — задіяти веб-портал. При цьому користувач має відповідні права на створення ресурсів. Ручний спосіб дуже простий: у всіх хмарних провайдерів є зручні портали, велика документація, відеоінструкції тощо. Не потрібні додаткові сервіси, SDK тощо.

Однак цей спосіб має недоліки:

– тривалий час створення інфраструктури;

– недостатня надійність (у разі проблем з ресурсами їх доведеться перетворювати вручну, з усіма ручними налаштуваннями, конфігуруванням тощо);

– трудність перенесення інфраструктури в новий регіон або обліковий запис. Її знадобиться вручну клонувати або копіювати (даний недолік частково

згладжується тим, що хмарні провайдери дозволяють копіювати або клонувати ресурси, але ця процедура все одно вимагає ручного ініціювання);

– процес створення ресурсів у цьому випадку неможливо автоматизувати.

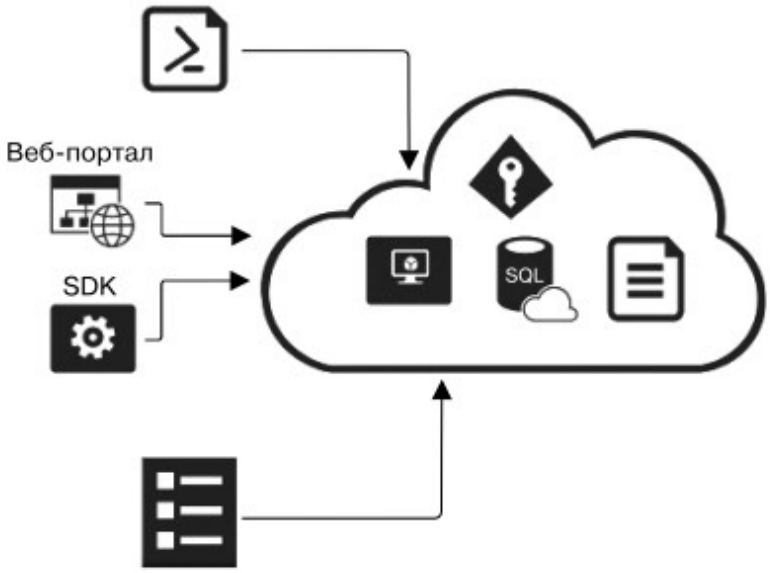

Рисунок 2.1 – Способи управління хмарною інфраструктурою

Другий спосіб - застосувати програмні бібліотеки (Software Development Kit, SDK), що забезпечують доступ до ресурсів хмари з коду програм користувача. Як правило, SDK є набором класів і методів, що полегшують програмні операції з ресурсами хмари. Щоб забезпечити доступ до таких ресурсів, програма з хмарним SDK повинна містити ключі облікового запису, який матиме доступ до хмари. У хмарі ці ключі зареєстровані у вигляді користувача в активному каталозі хмарного облік`ового запису, що володіє правами виконувати програмне маніпулювання ресурсами хмари (такий користувач називається принципалом – service principal). І керування цими обліковими записами відбувається так само, як і обліковими записами користувачів хмарного веб-порталу. Серед переваг такого підходу — можливість створення програм, які самі собі створюють хмарні ресурси, а також автоматизованого управління хмарним акаунтом.

До третього способу створення хмарних ресурсів відносять спеціалізовані розширення для мов командного рядка — shell, CMD та ін. Для підключення цих розширень до хмарних ресурсів необхідно імпортувати ключі або виконати вхід до облікового запису через форму введення логіна/паролю. Як і у випадку SDK для сценарних мов програмування, SDK для командної оболонки дозволяє описувати хмарну інфраструктуру як набору команд, кожна з яких створює чи конфігурує відповідний хмарний сервіс.

І SDK, і команди оболонки оперують зрештою з API хмарного провайдера (як правило, REST API), доступ до яких також дозволить маніпулювати ресурсами хмари.

Четвертий спосіб створення хмарних ресурсів – застосувати шаблони. У цьому випадку всі необхідні ресурси та зв'язки між ними описуються за допомогою текстового файлу у форматі YAML або JSON. Такий шаблон може бути завантажений у хмарний сервіс безпосередньо через веб-портал або через CLI-команди.

Опис інфраструктури через шаблон — дуже потужний механізм, який широко використовується для конфігурування різних інфраструктур (наприклад, у системах Ansible, Chef, Puppet та ін.). Як уже вказувалося, шаблони є текстовими файлами, які можуть зберігатися в репозиторії шаблонів або найчастіше в репозиторії GitHub. Для хмари AWS сервіс створення ресурсів за допомогою шаблонів називається CloudFormation (підтримує YAML та JSON), у Azure це ARM Template (нині підтримує лише JSON). На веб-порталі AWS є спеціальний редактор, що спрощує створення та конфігурування шаблону. Останній може бути завантажений у файлове сховище S3, репозиторій CodeCommit або інше місце, доступне для сервісу CloudFormation. Цей сервіс створює стек - набір ресурсів, керованих спільно (створення, видалення та оновлення).

Сервіс CloudFormation дуже зручний у застосуванні зі сторонніми конфігураційними сервісами — наприклад, Ansible. Ця програма, що широко використовується, задіює YAML для створення конфігураційних шаблонів, які

служать для адміністрування групи серверів (переважно Linux, але є розширення і для Windows), не вимагаючи інсталяції на цих серверах «агентів». Для роботи Ansible необхідні лише ключі доступу до ресурсів (SSH-ключі для Linux-хостів, сертифікат для PowerShell-доступу до Windows-хостів або ключі доступ до AWS). Шаблон CloudFormation для Ansible представлений у вигляді JINJA, що припускає передачу параметрів через змінні Ansible.

2.1 Безпека хмарних ресурсів

Поряд із незаперечними перевагами, зберігання та обробка даних у хмарних середовищах потенційно може доставити низку проблем, яких немає (або, вірніше, вони виявляються не так чітко) у разі розміщення та обробки даних у власних дата-центрах. Це зумовлено низкою причин. По-перше, хмарні середовища самі собою публічно доступні і всі сервіси, якщо явно не налаштовано інше, доступні для всіх в Інтернеті. По-друге, захист даних та інфраструктури від ненавмисних дій користувачів лежить поза компетенцією хмарного провайдера. Крім того, хмарні інфраструктури, що працюють з великими даними, часто містять у своєму складі великі кластери віртуальних машин, що потребує застосування спеціальних заходів для забезпечення надійної роботи всієї системи.

Крім цього, інформація фізично передаватиметься незахищеними каналами і існує загроза її перехоплення. Розглянемо докладніше всі ці та інші аспекти безпеки хмарних середовищ.

Найбільш поширений спосіб захисту кінцевих точок хмарних сервісів – обмеження доступу до них за допомогою механізмів автентифікації та створення списків дозволених IP-адрес, з яких можна отримати доступ до точок. Розглянемо насамперед різні способи забезпечення доступу із заданого адресного простору.

Сервіси, що стосуються IaaS, а також у ряді випадків до PaaS, вимагають для свого створення налаштованої хмарної віртуальної приватної мережі

(VNet, VPC), розбитої на підмережі [13]. Доступ до кінцевих точок сервісів, розташованих у цих підмережах, можна регулювати за допомогою онфігурування мережевих груп безпеки (Network Security Group, NSG) (рисунок 2.2), які є списками контролю доступу, ACL.

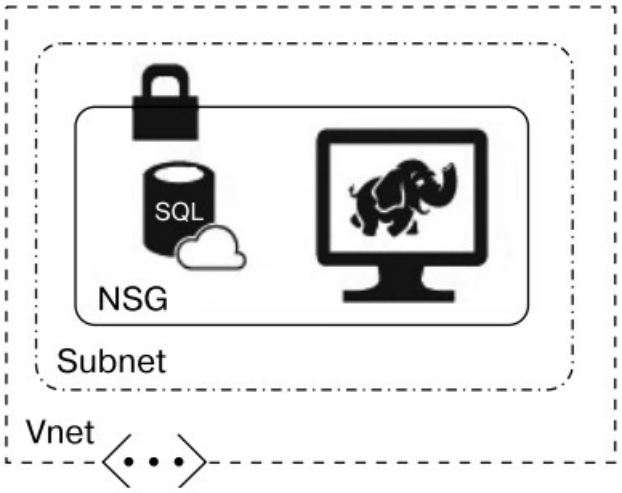

Рисунок 2.2 – Обмеження доступу до кінцевих точок хмарних сервісів за допомогою мережевих груп безпеки

Віртуальна частина мережі – один із базових сервісів IaaS. Він є хмарним аналогом локальної мережі і служить для надання діапазону IP-адрес для розміщення в них ресурсів. Віртуальну приватну мережу можна розділити на підмережі (subnet), а між ними встановити правила маршрутизації IP-пакетів. Крім того, на підмережі можна встановити списки контролю доступу, які називаються мережними групами безпеки. Це дозволяє логічно розділяти архітектури інформаційних систем на різні рівні (наприклад, рівень даних, бізнес-логіки, фронтенд) шляхом розміщення кожного рівня у своїй підмережі та встановлення правил маршрутизації.

NSG є список доступу, що містить набір записів. Кожен запис складається з таких елементів, як:

– назва;

– число, що визначає пріоритет перегляду списку записів;

– діапазон IP-адрес (для однієї конкретної адреси це /32);

– номер порту;

– дія - ALLOW або DENY («Дозволити» або «Відхилити») стосовно запиту, що надійшов з цієї адреси.

Крім того, вказується протокол, до якого застосовується дія ALLOW або DENY (TCP, UDP, ICMP тощо). Безпека кінцевих точок у разі забезпечується обмеженням до них доступу ззовні. Крім NSG, ряд хмарних сервісів, що не потребують віртуальної приватної мережі (наприклад, Azure SQL), мають фаєрволи списки «дозволених» і «заборонених» діапазонів. Хорошою практикою є повсюдне використання NSG та фаєрволів. При цьому необхідно, щоб усі порти, що стосуються віддаленого доступу/керування (наприклад, 22 для SSH, 3388 для RDP) або безпосередньо до сервісу (скажімо, 1433 для MS SQL) були недоступні з Інтернету поза діапазоном адрес віртуальної приватної мережі. Для отримання доступу до сервісів з «дозволеної» локальної мережі або з дозволеного комп'ютера слід встановити VPN-шлюз з локальної мережі або з комп'ютера у віртуальну приватну мережу або безпосередньо до екземпляра сервісу. Крім шлюзу, при з'єднанні локальної мережі з віртуальною приватною мережею необхідно застосувати проміжний хост, проксі-хост (рисунок 2.3), який транслюватиме запити та дозволятиме приватні адреси хмарних ресурсів з локальної мережі.

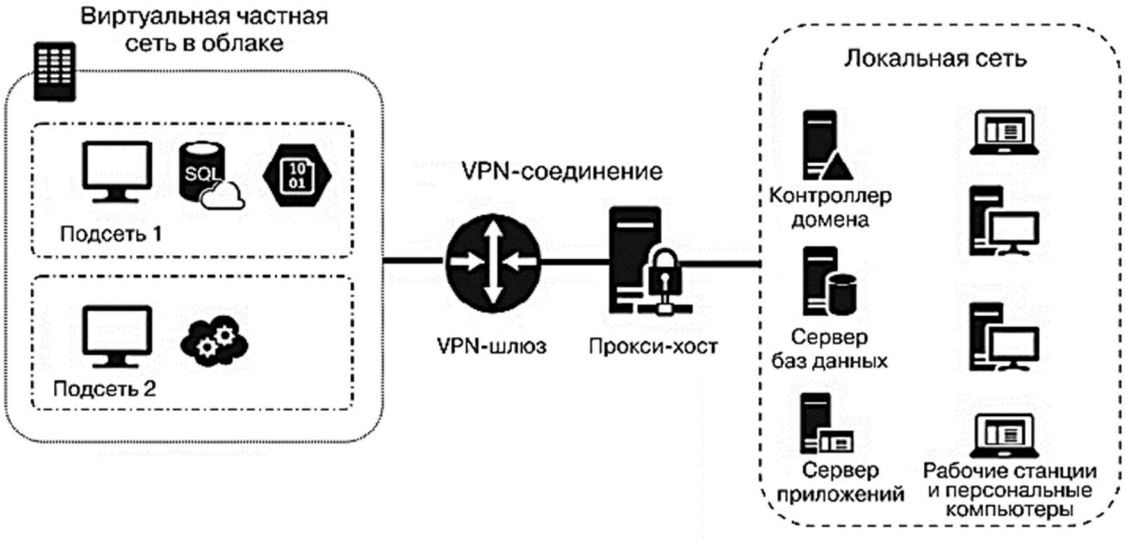

Рисунок 2.3 – Забезпечення захищеного доступу до хмарних ресурсів

Проксі-хост у різних реалізаціях можна розмістити як у хмарній мережі, так і в локальній. В останній може розташовуватися контролер домену, сервер БД з даними, які не можуть бути розміщені у хмарі, а також інші сервери, які можуть бути розміщені лише у локальній мережі.

Деякі кінцеві точки (скажімо, API управління самого хмарного облікового запису) не обладнані ні фаєрволом, ні NSG. Для їхнього захисту використовується як шифрування трафіку (HTTPS), так і спеціальні ключі доступу до ресурсів. Ключі можуть генеруватися безпосередньо сервісом, що захищається, і застосовуватися в заголовках REST-запитів. Наприклад, сервіс сховища Azure Storage використовує автентифікацію на основі HMAC — Hash-based Message Authentication Code, який повинен містити підписаний алгоритмом SHA-256 токен як заголовок автентифікації. Наступний стандартний механізм аутентифікації запитів - застосування протоколу OAuth 2.0, при якому облікові дані користувача зберігаються в сервісі зберігання облікових даних (у разі Azure це Azure Active Directory, а AWS - Cognito) [14]. У загальному випадку хмарний сервіс зберігання облікових даних включає користувачів, ролі та API-ресурси, що захищаються цим протоколом. Будь-який запит до REST API ресурсів, зареєстрованих у цьому сервісі (а це, по суті, всі REST API кінцевих точок управління хмарними ресурсами), повинен у своєму заголовку містити токен, який можна отримати, якщо виконати процедуру введення облікових даних до каталогу.

Ще один «ешелон» захисту даних у хмарних ресурсах – це їхнє шифрування. Поширений підхід у разі — про прозоре шифрування даних (transparent data encryption, TDE). Суть його полягає в тому, що все шифрування та дешифрування даних відбувається «за лаштунками», без участі користувача, за допомогою ключів шифрування, які генеруються та зберігаються самим хмарним акаунтом. У разі Azure ці ключі зберігаються в Azure KeyVault, а у випадку AWS - в AWS KMS.

Наступна ланка захисту — послуги, що відповідають за моніторинг запитів, що надходять до ресурсів і здійснюють аудит активності користувача.

Наприклад, Azure Security Center, який може виконувати моніторинг багатьох ресурсів, видає рекомендації щодо їх захисту і стежить за вхідним трафіком.

2.2 Обробка великих даних

Великі дані можуть бути оброблені в пакетному режимі, коли вони присутні в сховищі. Найчастіше це необхідно для агрегування даних та побудови аналітичних звітів на їх основі.

Розглянемо як приклад систему моніторингу електроенергії мережі будівель. У цій системі вимірювання споживаної потужності передаються з малою періодичністю як повідомлення від кожного вимірювального модуля. Щоб отримати величину денного споживання електроенергії, необхідно скласти всі вимірювання вимірювального модуля кожного користувача окремо. Якщо підсумковий результат необхідно, наприклад, формувати як щоденного (щотижневого, щомісячного тощо.) звіту, то найпростіше реалізувати таку систему в такий спосіб (рисунок 2.4).

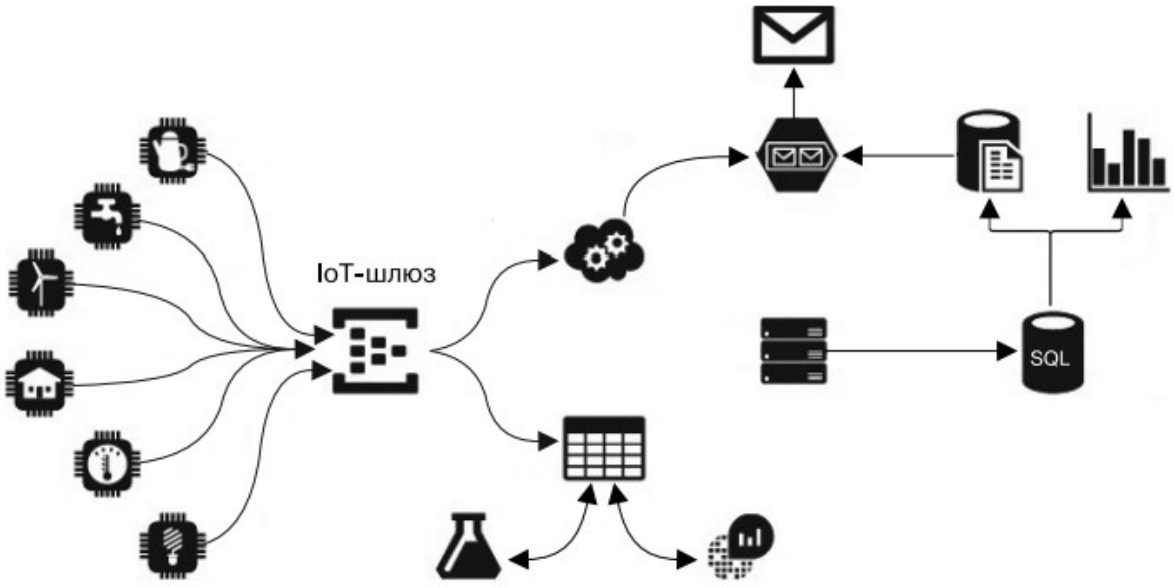

Рисунок 2.4 – Архітектура системи обліку

Всі повідомлення від вимірювальних пристроїв можна зберігати в нереляційному сховищі табличного типу (це питання докладніше розглядається в

частині II), наприклад, HBase або Cassandra. Кожний рядок таблиці міститиме тимчасову мітку (тобто час надходження або відправлення повідомлення), ідентифікатор пристрою, що його надіслав, та власне величину виміру.

Для побудови періодичного звіту за допомогою системи бізнес-аналітики (Business Intelligence, BI) (наприклад, PowerBI або Microsoft SSRS) або відображення цієї величини в браузері необхідно, щоб дані в агрегованому вигляді були розміщені в БД SQL. А чому не можна відразу розміщувати їх у цій базі? Порахуємо.

Припустимо, що вимірювальний пристрій надсилає повідомлення кожні п'ять хвилин. Це означає 12 відліків на годину, або близько 105 тис. на рік. Тепер припустимо, що система моніторингу збирає дані енергоспоживання з кожного пристрою замовника, яких можуть бути десятки, а самих замовників тисячі. У результаті таблиця, що зберігає події в реляційній БД, міститиме багато мільйонів рядків: припустимо, за наявності 100 замовників із середнім числом підключених приладів 20 на рік така таблиця поповниться 200 мільйонами рядків.

2.3 Архітектури традиційних інформаційних систем

Системи обробки великих даних мають на увазі можливість, що дозволяє реалізувати не тільки систему обробки даних, але й додаток, який вирішує конкретне бізнес-завдання. Ця ідея – асинхронна передача повідомлень (подій) між компонентами системи та застосування окремих сервісів, які забезпечують надійну передачу повідомлень. Здавалося б, практично будь-яка інформаційна система має справу з обміном повідомлень, поданим у тому чи іншому вигляді [15].

Багато років інформаційні системи будувалися у вигляді одного великого моноліту з однаковою кодовою базою, яка розгортається на сервері як єдине ціле: разом з усіма модулями, бібліотеками, що підключаються, і т. д.

Такий моноліт (рисунок 2.5) відповідав за все (або майже): за взаємодію з базами даних, забезпечення системи контролю облікових даних, роботу періодичних сервісів синхронізації, логування тощо.

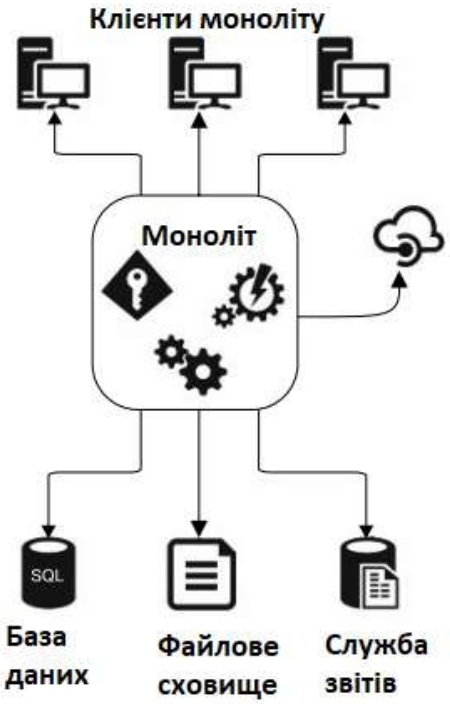

Рисунок 2.5 – Монолітна архітектура

Нижче наведені проблеми, властиві такій архітектурі.

Будь-яка зміна в будь-якому компоненті моноліту, навіть незначна, вимагає компілювання і розгортання всієї системи, що при великому розмірі моноліту завдання дуже нешвидке.

Моноліт дуже погано масштабується і схильний до суттєвих проблем з використанням ресурсів. Справді, зазвичай, подібні архітектури розгортаються однією сервері чи вигляді повних копій групи серверів. Тому будь-який компонент (або компоненти), який задіює ресурси сервера (скажімо, оперативну пам'ять, процесор і т. д.), значною мірою може утруднити роботу всього моноліту і вимагатиме збільшення продуктивності всього сервера, на якому розташована дана архітектура. Компоненти моноліту неможливо масштабувати незалежно.

Моноліт жорстко диктує стек технологій програмування, і оновити його, а також оновити саму архітектуру — отже, по суті, переписати весь код наново. Сюди ж можна віднести проблеми зі швидким «старінням» моноліту, схильністю до обростання спагетті-кодом, а також «тупиковою внутрішньою архітектурою» — це коли неможливо виправити баг чи покращити щось, не викликавши появи нового бага чи погіршення системи.

Супровід моноліту може завдати безліч незручностей програмістам, системним адміністраторам, тестувальникам.

Вирішити зазначені проблеми була покликана багатошарова архітектура (рисунок 2.6). У ній за специфічне завдання відповідає окремий шар: інтерфейсу користувача (user interface layer, UI layer), бізнес-логіки (business logic layer, BL layer) і доступу до даних (data access layer, DAL).

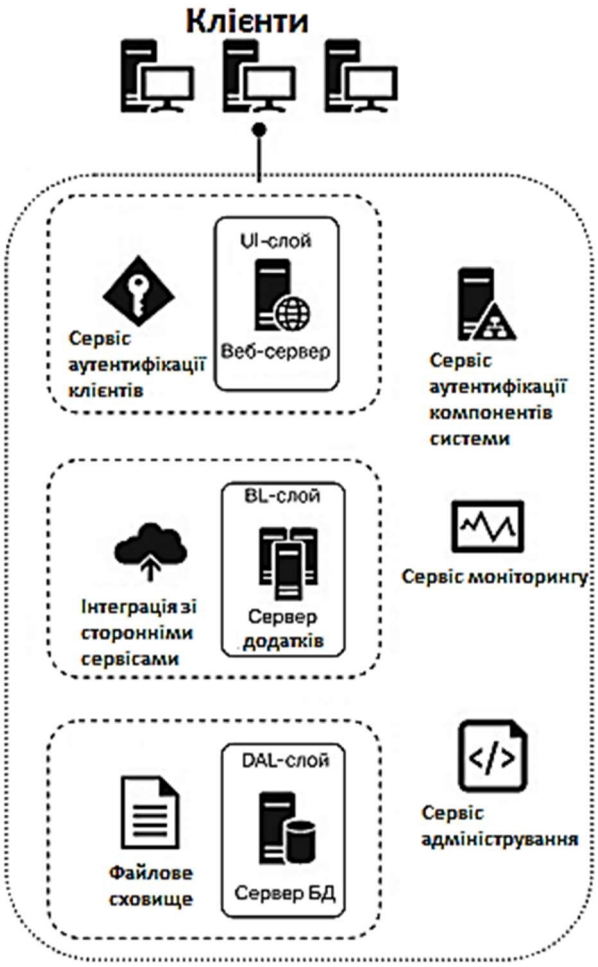

Рисунок 4.6 – Багатошарова архітектура

Кожен шар відповідає лише за певну групу завдань. UI-шар містить лише веб-сервери, що є джерелом веб-сторінок, або розміщує сервіси (REST, SOAP або ін.) для взаємодії з клієнтськими програмами. Крім того, цей шар приймає запити від клієнтів та транслює їх шаром бізнес-логіки. Оскільки з клієнтом безпосередньо взаємодіє лише UI-шар, то одному рівні з ним у тісній інтеграції знаходиться сервіс аутентифікації клієнтів. Шар BL містить сервери додатків та відповідає, власне, за бізнес-логіку та інтеграцію зі сторонніми сервісами. Шар DAL забезпечує програмний доступ шару BL до бази даних та файлового сховища.

Така архітектура проти монолітної має такі переваги.

Поділ на шари дозволяє реалізувати в кожному шарі найбільш підходящий для нього стек технологій. Можна незалежно оновлювати кадри на кожному шарі.

Логічне поділ на шари суттєво спрощує процес розробки та супровід усієї системи. Розподіл команд програмістів за шарами та спеціалізація розробників (фронтенд, бекенд), зменшення обсягу коду на кожному шарі, а також незалежний деплой (розгортання) значно покращують якість усієї системи, роблячи її більш гнучкою та придатною для супроводу.

Можливе незалежне масштабування кожного шару.

Найчастіше подібна архітектура реалізується як серверів, розташованих у локальної мережі, розбитої на підмережі з налаштованими фаєрволами. Адреси серверів (IP або URL) різних шарів та облікові дані для доступу до них прописані у конфігураційних файлах інших серверів. У цій архітектурі необхідно централізовано зберігати облікові дані, мати сервери DNS, сервіс зберігання логів і моніторингу, а також сервер для адміністрування. У монолітній архітектурі все це могло бути розташоване на одному великому загальному сервері. Логування також було дуже простим, оскільки вся система будувалася в рамках одного стеку технологій. У багатошаровій архітектурі кожен шар може генерувати логи, які істотно відрізняються від логів іншого шару. Крім того, збільшення кількості серверів теж веде до збільшення кількості та видів логів.

Багатошарова архітектура інформаційних систем в даний час надзвичайно поширена, вона гнучкіша і зручніша в порівнянні з монолітною, проте не може вирішити ряд проблем. Зокрема, при розростанні проекту до такого масштабу, що шар BL стає порівнянним із монолітом, з'являються аналогічні проблеми з масштабуванням, продуктивністю, супроводом тощо.

3 СХЕМОТЕХНІЧНА РОЗРОБКА

3.1 Обґрунтування вибору мікроконтролера

В даний час для убудованих систем збору й обробки інформації найбільше поширення одержали мікроконтролери сімейства STM32.

Мікроконтролери (МК) STM32 засновані на 32-бітних RISC ядрах ARM Cortex M0/M0+, M3, M4 і M7 і поділяються на 4 основні сімейства [16]:

1. Для бездротових рішень: STM32WB3 і STM32WL3
2. З низьким споживанням: STM32L0, STM32L1, STM32L5, STM32L4 і STM32L4+
3. Базові характеристики: STM32F0, STM32G0, STM32F1, STM32F3 і STM32G4
4. Висока продуктивність: STM32F2, STM32F4, STM32F7 і STM32H7

Різновиди МК STM32 приведені на рисунку 3.1.

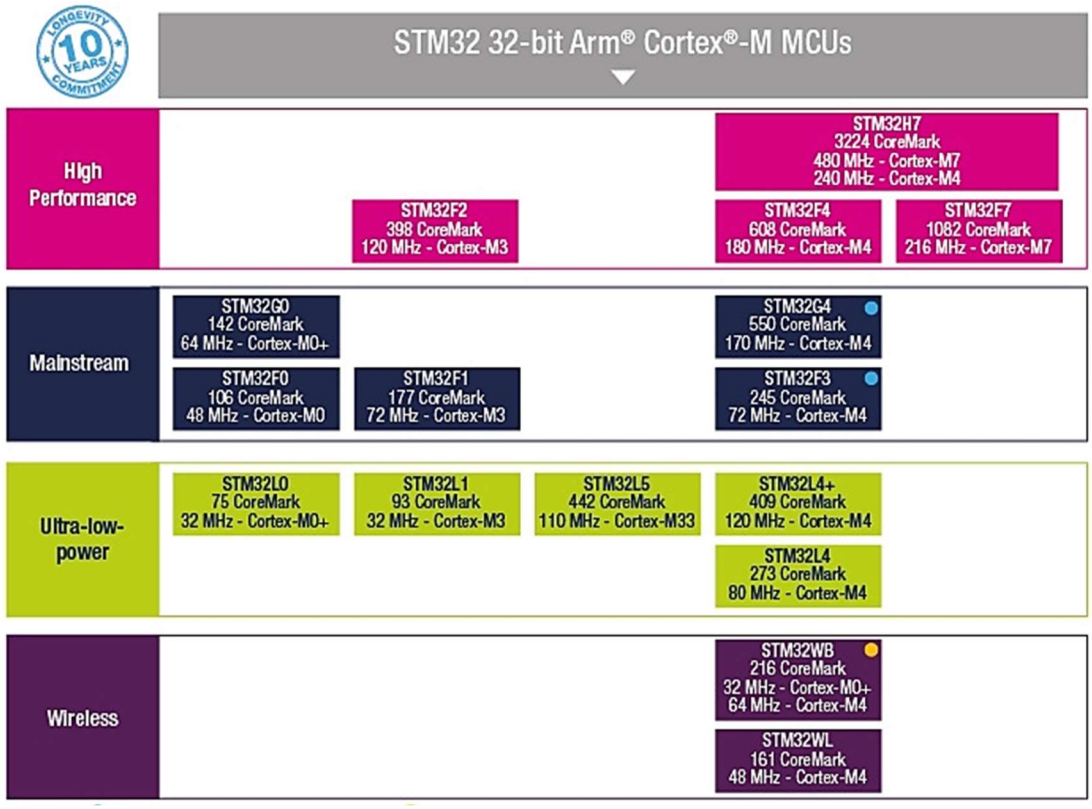

Рисунок 3.1 – Різновиди МК STM32

Огляд і аналіз приведених типів МК показав, що в проектованому пристрої найбільше доцільно використовувати мікроконтролер STM32F103C6T6 з базовими характеристиками, приведеними в таблиці 3.1.

Таблиця 3.1 – Характеристики МК STM32F103C6T6

Використовуваний мікроконтролер	STM32F103C6T6
Ядро	ARM 32-bit Cortex-M3
Напруга живлення плати	2,7…5 В
Максимальна тактова частота чіпа	72 МГц
Обсяг флеш пам'яті	64 Кб
Обсяг оперативної пам'яті SRAM	20 Кб
Кількість виводів GPIO	34
Кількість АЦП	2x 12 bit (16 каналів)
Розрядність ШІМ	16 bit
Діапазон робочих температур	-40 ^{0}C…+85 ^{0}C
Розміри плати	56x43 мм

STM32F103C6T6 і STM32F103C8T6 є мікроконтролерами із сімейства STM32F1 від STMicroelectronics. Вони є популярними мікроконтролерами, що базуються на ядрі ARM Cortex-M3.

Мікроконтролер STM32F103C6T6 має три апаратних UART інтерфейси, 2x SPI, 2x I2C, 1x USB 2.0, 1x CAN, 7-канальний DMA контролер, інтерфейси SWD і JTAG, годинник реального часу і три 16-бітних таймери.

Плати розроблювача, що підтримують ці мікроконтролери, можуть мати різні розширені функціональні можливості й убудовані периферійні пристрої, такі як LED індикатори, кнопки, реле, розширення інтерфейсу, рознімання для підключення інших модулів і багато чого іншого. Ці плати розроблювача можуть використовуватися для прототипування, розробки і тестування різних проектів, включаючи системи керування, убудовані системи, робототехніку й інші додатки.

Для програмування мікроконтролера використовується програматор ST-Link.

Структурна схема мікроконтролера приведена на рисунку 3.2 [17].

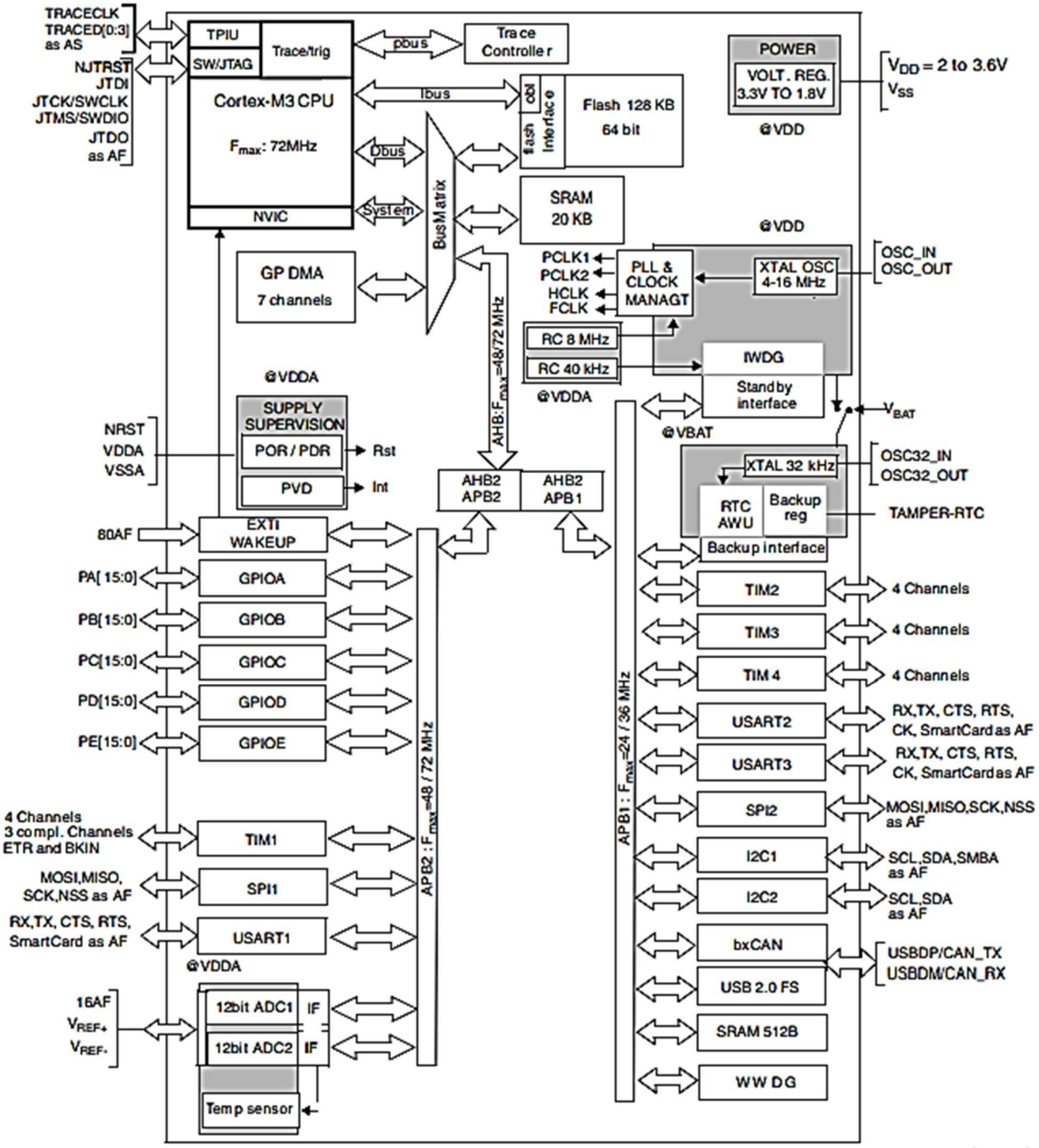

Рисунок 3.2 – Структурна схема МК STM32F103C6T6

Плата розроблювача на базі 32-бітного мікроконтролера STM32F103C8T6 з ядром ARM Cortex-M3 приведена на рисунку 3.3 [18].

На платі маються всі необхідні елементи для початку роботи з даним мікроконтролером:

– 34 порти GPIO, розведених на контактні площадки;

– мікросхема EEPROM пам'яті на 4 кБіт AT24C04;

– два кварца – 8 МГц для тактування ядра і 32768 Гц для тактування RTC;

– стабілізатор напруги 3.3 В для забезпечення можливості живлення плати від 5 В;

– рознімання для підключення SWD програматора або JTAG-віладчика;

– порт MiniUSB (з'єднаний з апаратним USB інтерфейсом);

– кнопка перезавантаження Reset S1;

– користувальницька кнопка S2 (pin PA0);

– два світлодіода (один з них – індикатор подачі живлення, другий – підключений до порту PC13);

– рознімання SPI для підключення OLED дисплея.

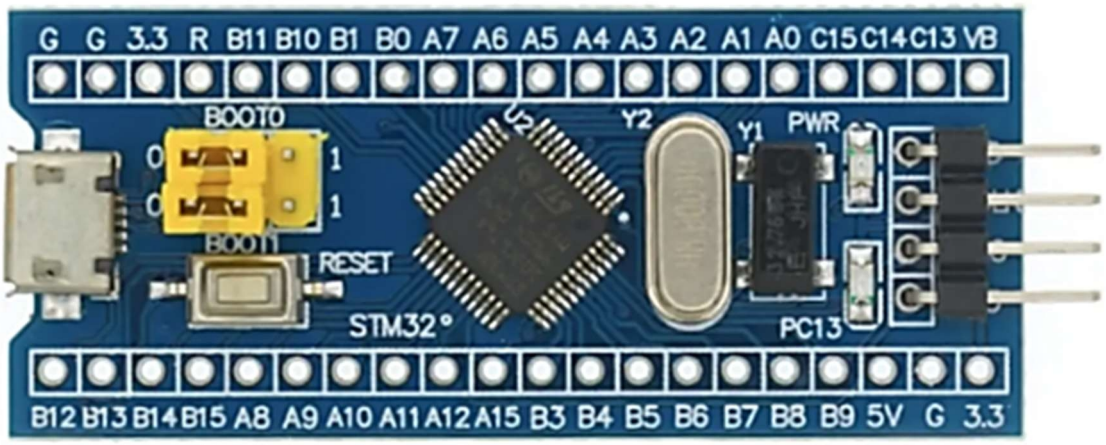

Рисунок 3.3 –Плата STM32F103C6T6

На платі зібрані всі необхідні елементи для початку роботи з даним сімейством мікроконтролерів.

Для розробки коду і прошивання STM32F103C6T6 можна використовувати як спеціалізовані IDE – Keil, IAR, Eclipse тощо, так і Arduino IDE.

Призначення альтернативних виходів плати приведено на рисунку 3.4.

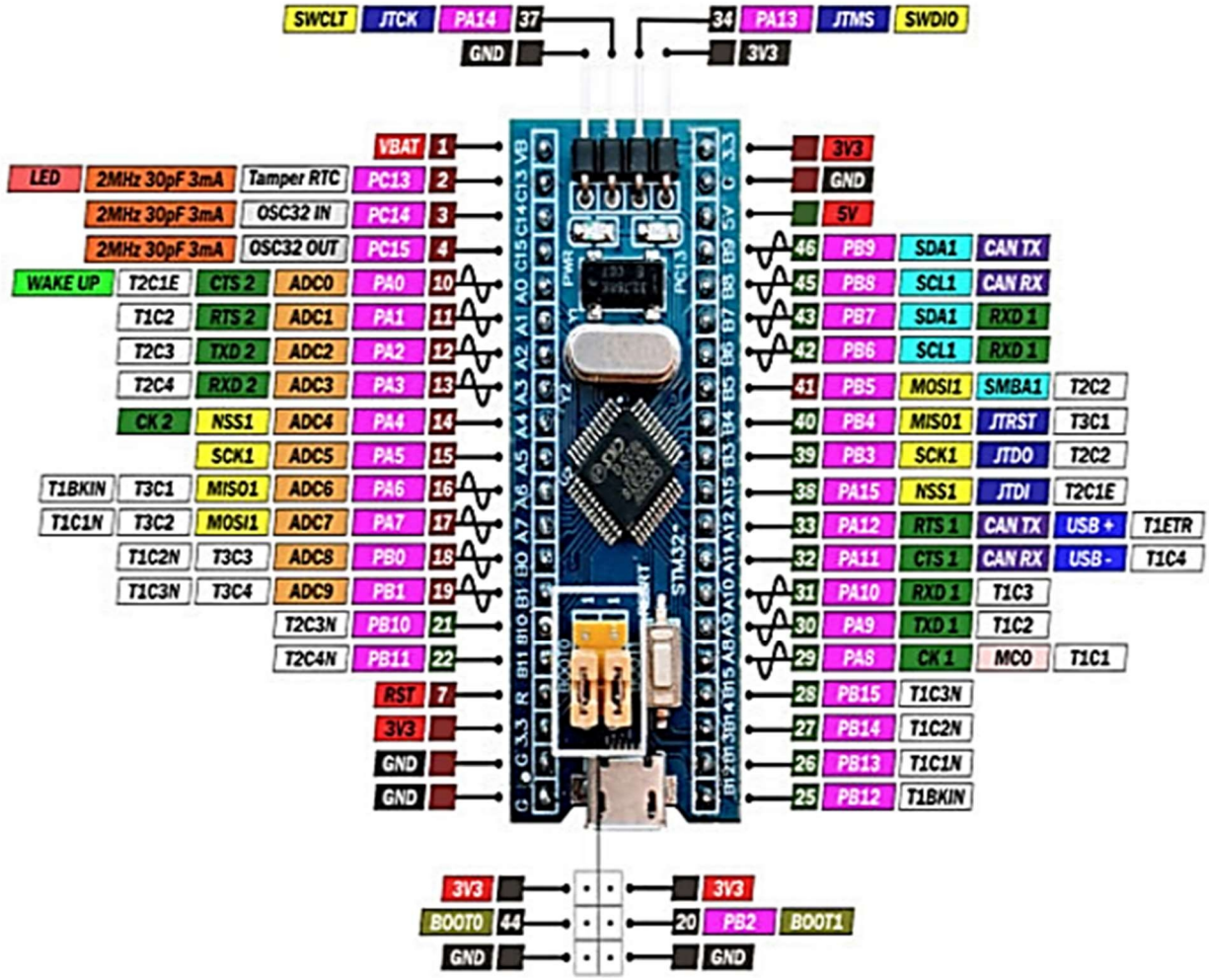

Рисунок 3.4 – Призначення альтернативних виходів плати

3.2 Вибір датчика вологості та температури

У нашому повсякденному житті, температура та вологість – два важливі параметри навколишнього середовища, з якими ми найбільше взаємодіємо з природою. Збір і контроль температури і вологості мають велике значення на виробничих і експериментальних майданчиках, а також у місцях проживання та відпочинку, спеціально для приміщень зі строгими вимогами до температури і вологості, таких як серверні кімнати, музеї, зберігання у холодильнику, так далі. Крім того, це неминуча тенденція до створення віддалених бездротових систем моніторингу температури і вологості в сучасному розвитку науки і техніки.

Датчики DHT22/DHT11 дуже популярні серед любителів електроніки,

оскільки вони дуже дешеві, але при цьому забезпечують відмінну продуктивність. Порівняльні характеристики цих датчиків наведено у таблиці 3.1.

Таблиця 3.1 – Порівняльні характеристики датчиків DHT22/DHT11

Параметр	DHT11	DHT22
Напруга живлення, В	3…5,5	3,3…6
Діапазон вимірів вологості, %	20…90	0…90
Діапазон вимірів температури, ^{0}C	0…50	-20…+60
Точність вимірів температури, ^{0}C	± 2	±0,5
Точність вимірів вологості, %	± 5	±2
Інтерфейс	1-Wire	1-Wire

Огляд та аналіз параметрів наведених датчиків показав доцільність застосування в даній розробці датчика DHT22, який має кращі характеристики.

Цей датчик складається з компонента вимірювання вологості, датчика температури NTC (або термістора) та мікросхеми на задній стороні датчика (рисунок 3.5) [19].

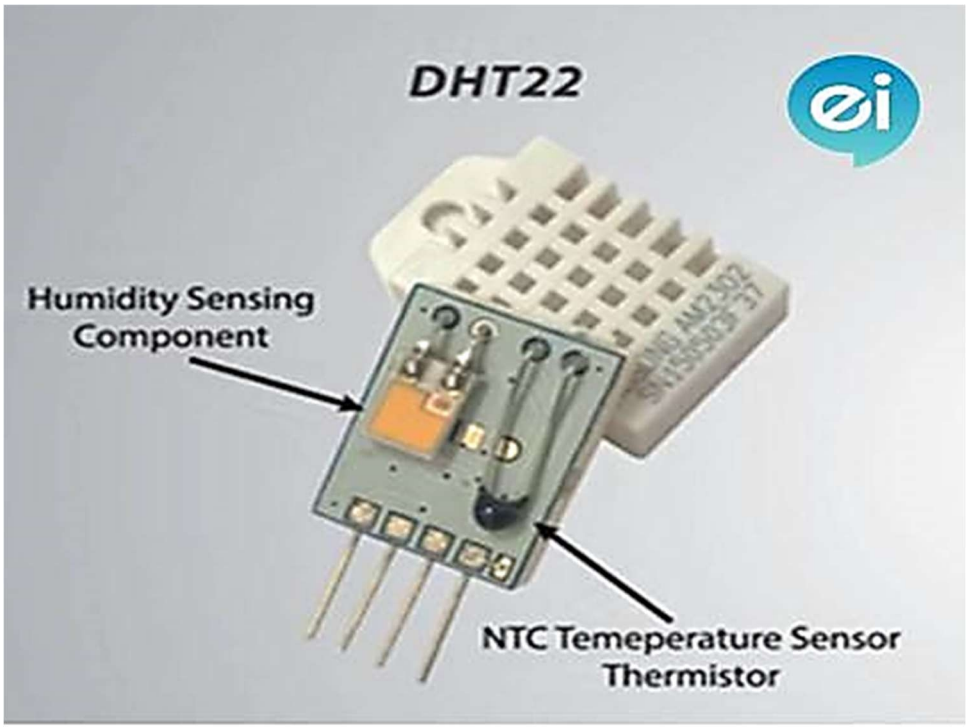

Рисунок 3.5 – Конструкція датчика DHT22

Для вимірювання вологості використовується компонент вимірювання вологості, який має два електроди з вологоутримуючою підкладкою між ними (рисунок 3.2).

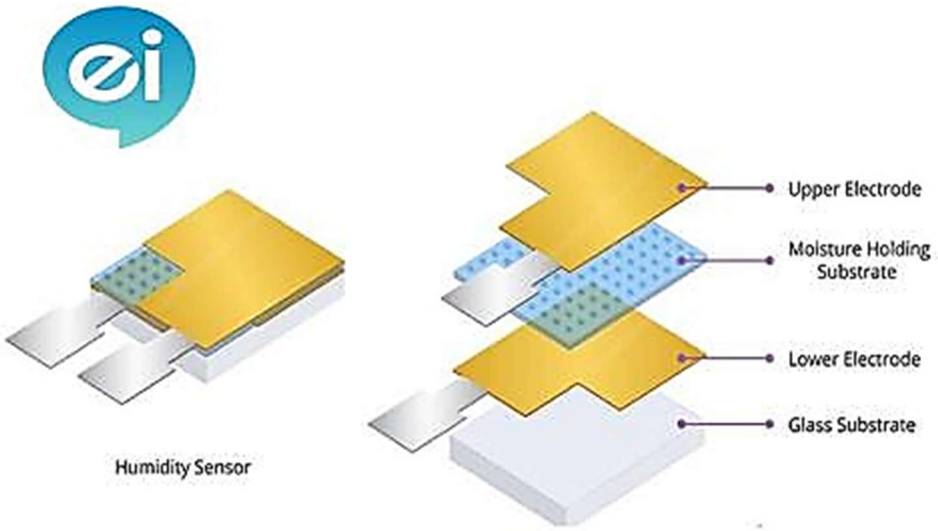

Рисунок 3.2 – Конструкція вимірювача вологості

Таким чином, при зміні вологості змінюється провідність підкладки або змінюється опір між цими електродами. Ця зміна опору вимірюється і обробляється мікросхемою, яка готує його до зчитування мікроконтролером.

З іншого боку, для вимірювання температури в цих датчиках використовується датчик температури NTC або термістор (рисунок 3.3).

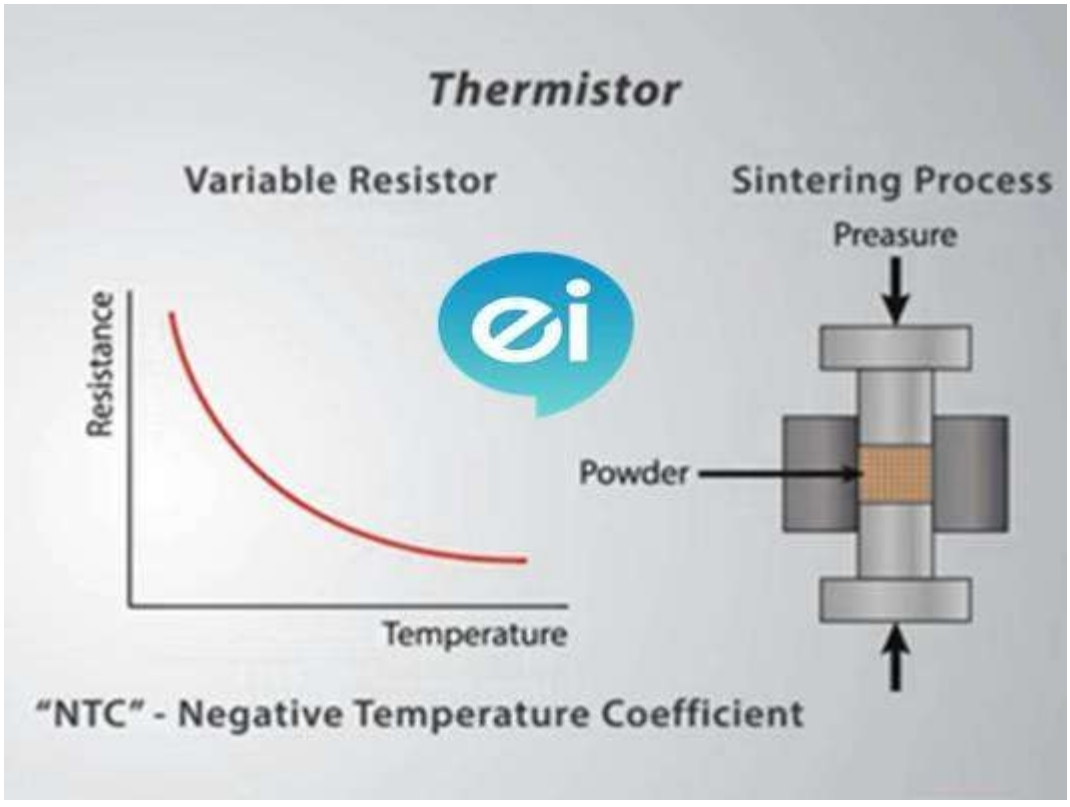

Рисунок 3.3 – Датчик температури NTC

Термістор насправді являє собою змінний резистор, опір якого змінюється при зміні температури. Ці датчики виготовляються шляхом спікання напівпровідникових матеріалів, таких як кераміка або полімери, щоб забезпечити великі зміни опору при невеликих змінах температури. Термін NTC означає "негативний температурний коефіцієнт", що означає, що опір зменшується зі збільшенням температури.

Також датчик містить в собі АЦП для перетворення аналогових значень вологості та температури.

3.3 Вибір точки доступу до хмари

Мікроконтролер ESP8266 – недорога і функціональна мікросхема, від виробника Espressif з підтримкою WiFi-інтерфейсу. Керувати всім цим можна не тільки з браузера, але і з додатків на Android/iOS/Desktop. Якщо МК буде застосовуватися там, куди не дістає WiFi-мережа, то ESP8266 може працювати в режимі точки доступу.

Мікроконтролер ESP8266 працює з зовнішньою flash-пам'яттю по інтерфейсу SPI. Її обсяг варіюється від 512 Кбайт до 4 Мбайт.

Існує біля півтора десятка версій МК серії ESP і величезна кількість плат з ними. Розглянемо самі популярні з них.

Мікроконтролери ESP8266:

– ESP-01. Має 8 розведених контактів (VCC, GND, UTXD, URXD, CH_PD, GPIO0, GPIO2, GPIO6) і PCB-антену (друкований провідник на самій платі).

– ESP-03. Тут з'являється керамічна антена. Вона вважається набагато ефективнішою свого друкованого побратима.

– ESP-07. У цій версії в очі відразу кидається металевий екран (який перед цим з'являється на ESP-06). На платі керамічна антена і рознімання для зовнішньої антени.

– ESP-12. У свою чергу, існує кілька варіантів цієї версії: ESP-12S,

ESP-12F, ESP-12E. Друга і третя версії мають на торці додатково 6 розведених контактів.

Плати:

WeMos D1 mini. Має розпаювання дев'яти GPIO-контактів. На платі мається відомий міст CH34x (такі часто ставлять на клони Arduino). Установлений МК із 4 Мбайт flash-пам'яті.

NodeMCU v0.9/v1. Перше покоління плат серії NodeMCU. На ній розпаяні всі 11 GPIO-портів. Деякі з них мають додаткові функції (UART, I2C, SPI, PWM, ADC).

NodeMCU v3. Фінальна версія плати цієї серії. Існує і v2 «Amica», що менше по габаритам. v3 зветься «LoLin» і відрізняється від попередньої версії тільки розмірами і незначними деталями (наприклад додатковим розпаюванням шини харчування). Крім традиційного моста CH340/CH341 на плати ставлять чип CP2102, так що потрібно бути уважним з вибором драйвера на них.

Усі ці (і не тільки ці) плати виконані на чипсеті мікроконтролера ESP8266EX, а отже, характеристики в них однакові [20]:

- протоколи: 802.11 b/g/n/e/i;
- діапазон частот: 2.4 ГГц – 2.5 ГГц;
- процесорне ядро: Tensilica L106 32 розряду;
- діапазон напруги живлення: 2.5 В – 3.6 В;
- середнє споживання струму: 80 мА;
- режими WiFi: Station/SoftAP/SoftAP+Station;
- безпека: WPA/WPA2;
- шифрування: WEP/TKIP/AES;
- відновлення прошивання: через UART, по радіоканалі (OTA – Over The Air);
- мережні протоколи: IPv4, TCP/UDP/HTTP/FTP;
- підтримка WiFi Direct (P2P), P2P Discovery, P2P GO (Group Owner) mode, GC (Group Client) mode, P2P Power Management;

– убудовані апаратні прискорювачі: CCMP (CBC-MAC, режим лічильника), TKIP (MIC, RC4), WAPI (SMS4), WEP (RC4), CRC;

– підтримка LUA-скриптів.

NodeMCU являє собою плату розроблювача на базі чипа ESP8266 (версія ESP12E), який являє собою UART-WiFi модуль з ультра низьким споживанням. Сам чип проектувався для пристроїв зі світу інтернет речей, а дана плата дозволяє спростити розробку, тому що на ній уже реалізоване підключення по USB, регулятор живлення й усі виводи чипа розведені на гребінки зі стандартним кроком 2.54 мм, що дозволяє вставити його в макетну плату і створити прототип навіть не включаючи паяльник. Крім цього плата поставляється з прошиванням NodeMCU, що дозволяє програмувати її за допомогою мови Lua або за допомогою Arduino IDE [21].

Загальний вигляд плати NodeMCU на базі чипа ESP8266 наведено на рисунку 3.4.

Рисунок 3.4 – NodeMCU на базі чипа ESP8266

Типова плата NodeMCU (якщо вона заснована на оригінальному дизайні NodeMCU Devkit) має 30 контактів. При цьому 8 контактів відносяться до живлення, а 2 зарезервовані. Інші 20 контактів зв'язані з контактами модуля ESP-12E (рисунок 3.5).

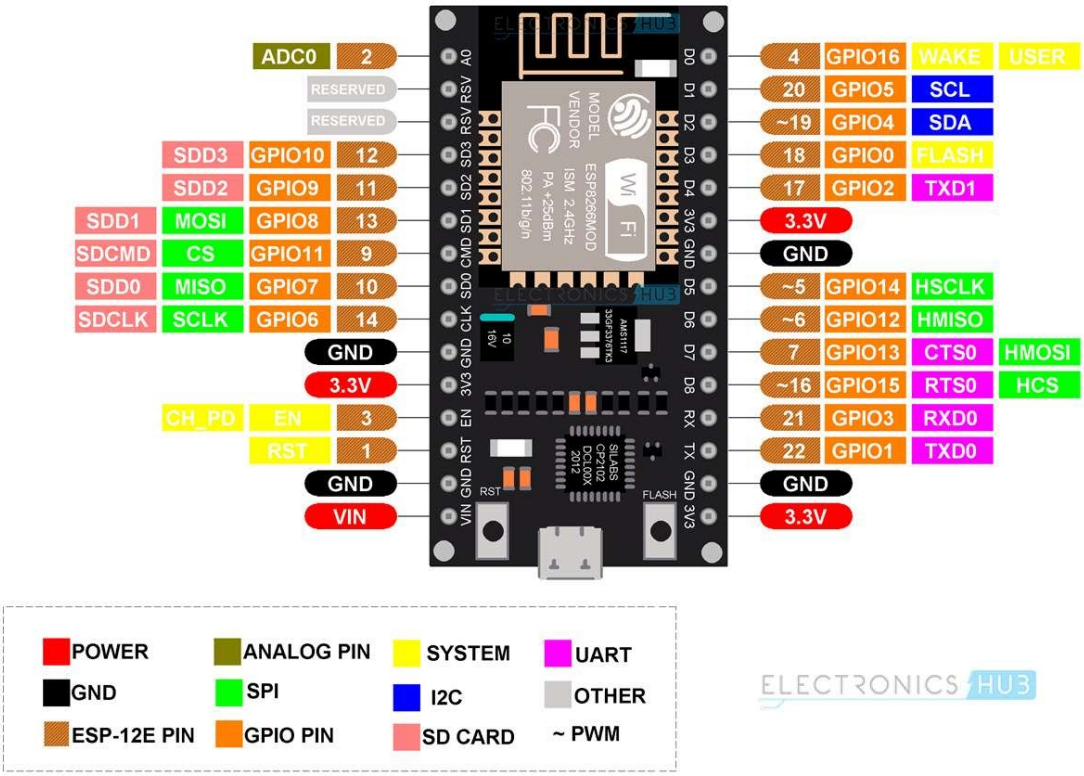

Рисунок 3.5 – Призначення основних і альтернативних виводів

Характеристики:

– WiFi стандарту 802.11 b / g / n;

– підтримка STA / AP / STA + AP режимів;

– убудований стік протоколів TCP / IP з підтримкою множинних клієнтських підключень (до 5);

– D0 ~ D8, SD1 ~ SD3: можуть бути використані як GPIO, PWM, IIC тощо;

– струм на виводі: 15 мА;

– AD0:1 вивід АЦП Power input: 4.5V ~ 9V (10VMAX), USB-powered, providing USB debugging interface;

– живлення: 4.5 - 9В (10 В максимум), живлення від USB з наданням відладочного інтерфейсу;

– споживання: обмін даними - ≈70 мА, чекання - <200 мкА;

– швидкість передачі: 110-460800 б/сек;

– підтримка UART / GPIO інтерфейсів передачі даних;

– перепрошивання з хмари або через USB;
– діапазон робочих температур: -40°C ~ + 125°C;
– тип живлення: подвійний потужний драйвер H-міст.

Схема з'єднання розробленого пристрою наведена на рисунку 3.6.

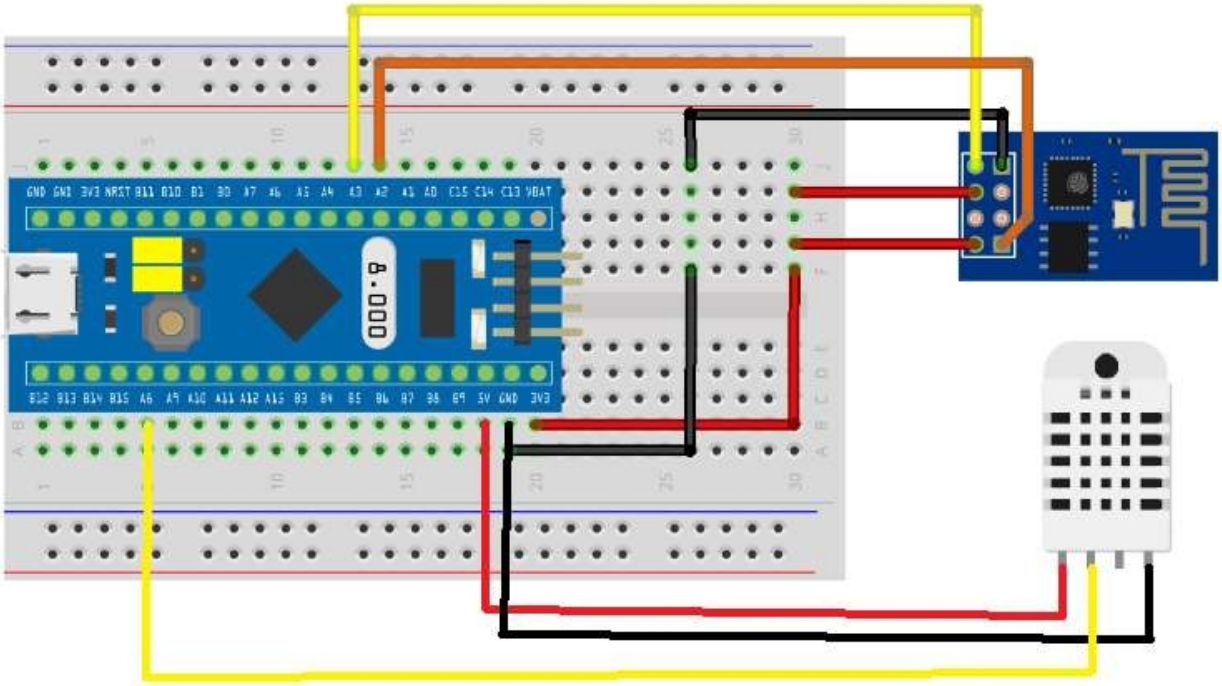

Рисунок 3.6 – Схема з'єднання розробленого пристрою

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Створення каналу на платформі ThingSpeak IoT

Для створення каналу необхідно спочатку зареєструватися в ThingSpeak і створити обліковий запис. Для цього переходимо по посиланню https://thingspeak.com/ і проходимо стандартну процедуру реєстрації [22]. Після цього заходимо у свій аккаунт і вибираємо опцію "Канали" (Channels) (рисунок 4.1) і вибираємо опцію "Мої канали" (My Channels).

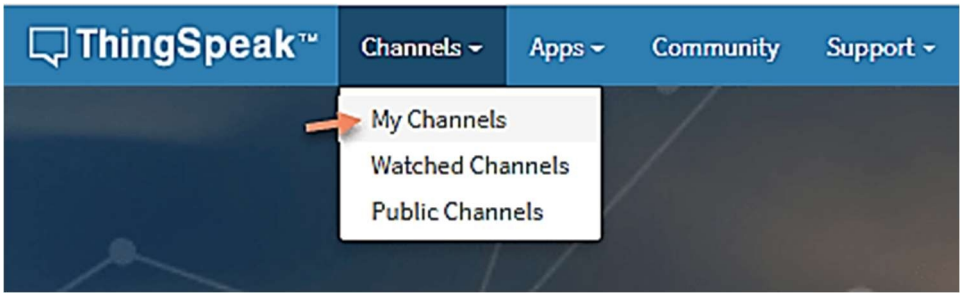

Рисунок 4.1 – Основне меню

У вікні "Мої канали" (My Channels) (рисунок 4.2) натискаємо "Новий канал" (New Cannel)

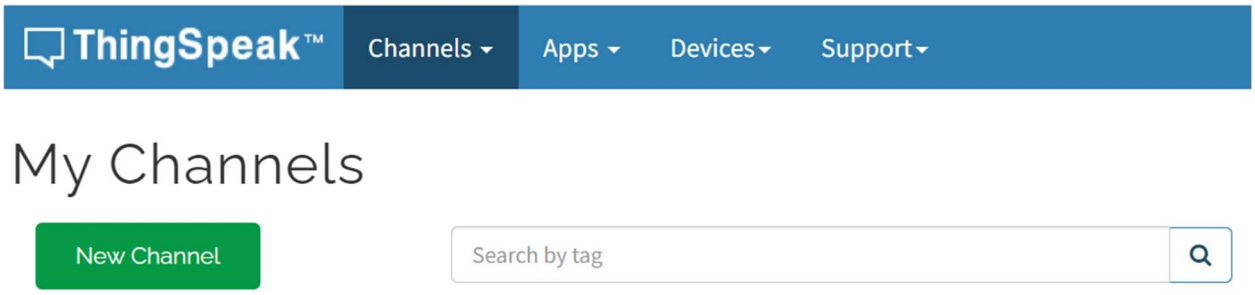

Рисунок 4.2 – Підменю створення каналів

У вікні, що відкрилось, (рисунок 4.3) у поле "Ім'я" (Name) уведіть назву вашого каналу.

Оскільки в даному проекті будемо працювати з точкою доступу ESP8266 і обраним мікроконтролером ESP32 разом на одній платформі Інтернету речей ThingSpeak, присвоюємо ім'я каналу ESP8266 & ESP32.

Поле "Опис" (Description) є не обов'язковим для заповнення.

Оскільки в даному проекті передбачається контроль і аналіз двох параметрів, установлюємо прапорці навпроти полів 1 і 2 і вводимо наступні значення настроювань каналу:

– Field 1: Temperature (температура);

– Field 2: Humidity (вологість).

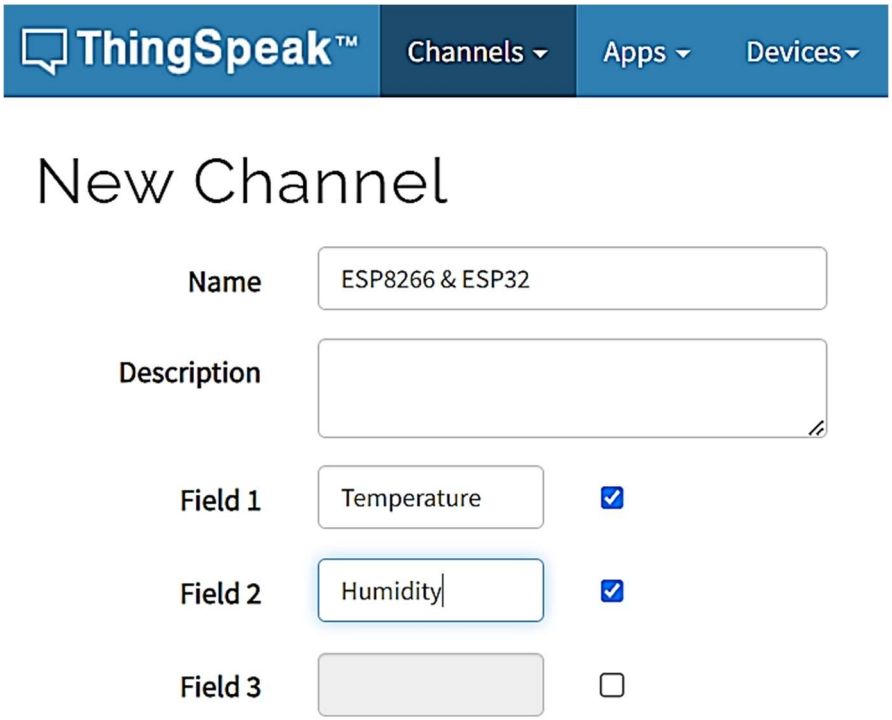

Рисунок 4.3 – Вікно введення параметрів каналу

Крім того, у вікні "Мої канали" (рисунок 4.4) доступні наступні вкладки:

– Private (Приватний перегляд). На цій вкладці відображається інформація про ваш канал, що можете бачити тільки ви;

– Public (Публічний перегляд). Якщо ви вирішили зробити свій канал загальнодоступним, використовуйте цю вкладку для відображення обраних полів і візуалізацій каналу;

– Settings (Настроювання каналу). На цій вкладці показані всі параметри каналу, що ви установили при створенні. На цій вкладці ви можете редагувати, очищати або видаляти канал;

– Sharing (Спільне використання). На цій вкладці показані параметри

загального доступу до каналу. Ви можете зробити канал приватним, загальним для усіх (загальнодоступним) або загальним для визначених користувачів;

– API Keys (Ключі API). На цій вкладці відображаються ключі API вашого каналу. Використовуйте клавіші для читання і запису на свій канал;

– Data import/Export (Імпорт/експорт даних). Ця вкладка дозволяє імпортувати й експортувати дані каналу.

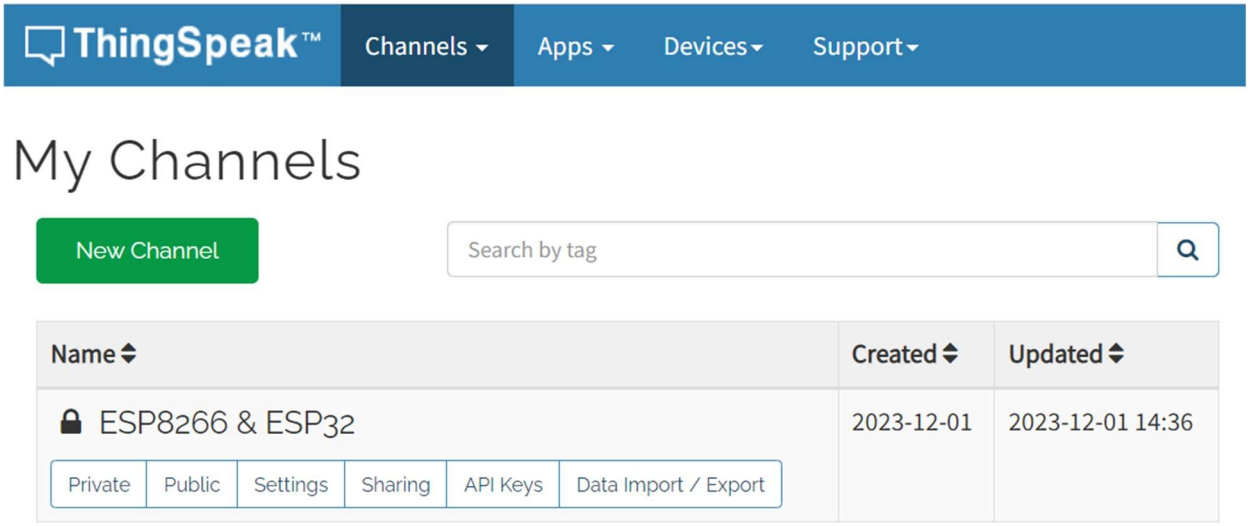

Рисунок 4.4 – Додаткові параметри каналу

Прокрутите вниз і натисніть кнопку «Зберегти канал». У результаті будуть створені два вікна для виводу діаграм температури і вологості відповідно (рисунок 4.5).

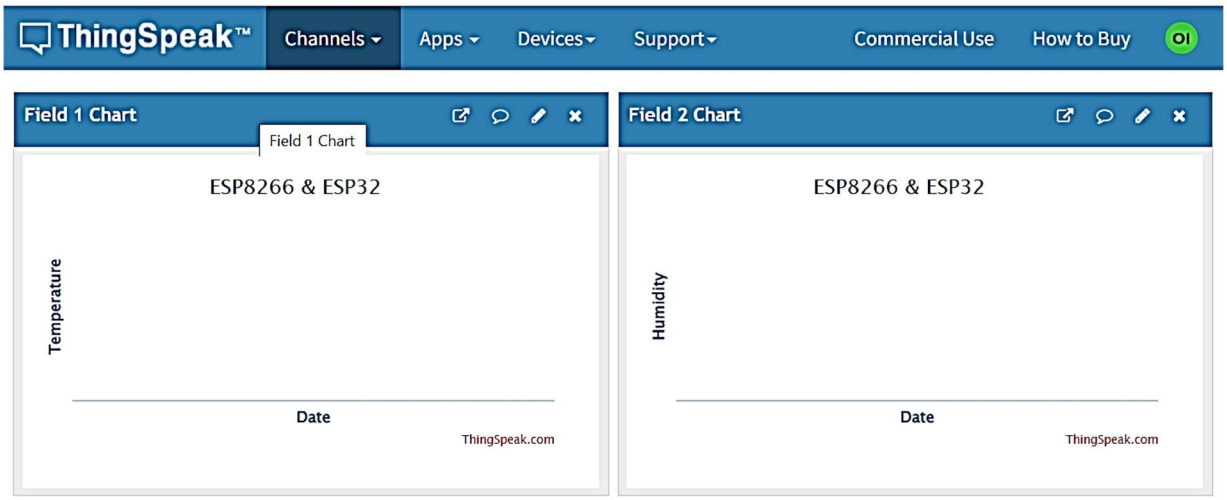

Рисунок 4.5 – Вікна для виводу діаграм температури і вологості

Тепер наступний крок – натиснути на ключі API (API Keys) (рисунок 4.6).

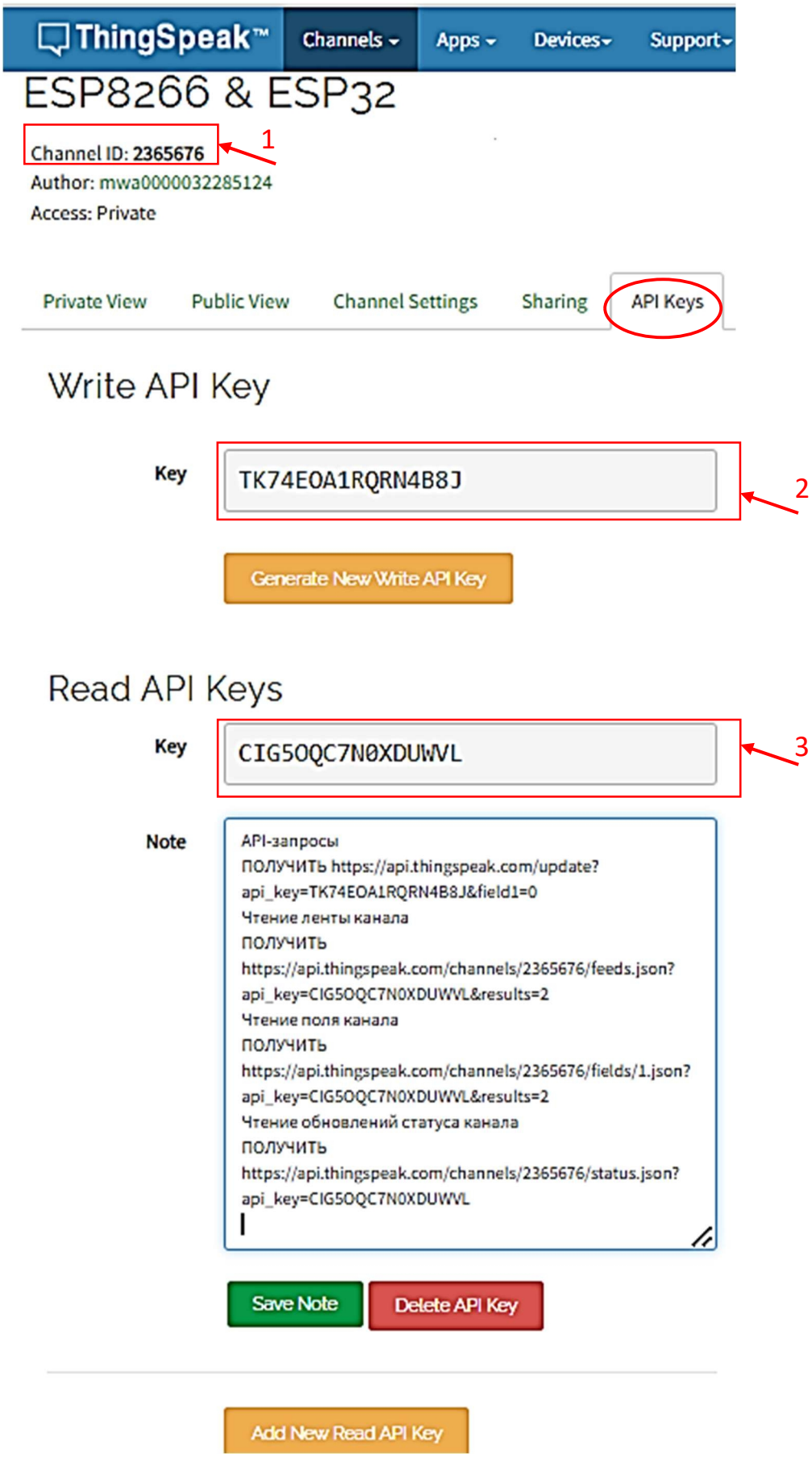

Рисунок 4.6 – Вікно ключів і ідентифікатора каналу

Ключі API генеруються автоматично, коли ви створюєте новий канал.

У даному вікні необхідно скопіювати ідентифікатор каналу (позначений цифрою "1" і два ключі – Write API Key (позначений цифрою "2") і Read API Keys (позначений цифрою "3"). Скопійовані ідентифікатор каналу й обидва ключі необхідно вставити в окремий файл блокнота, вони знадобляться при написанні програми роботи пристрою.

Призначення ключів API:

– Записати ключ API (Write API Key) – використовуйте цей ключ, щоб записати дані в канал. Якщо ви вважаєте, що ваш ключ зламано, натисніть "Створити новий ключ API для запису" (Generate New Write API Key).

– Читати ключі API (Read API Keys) – використовуйте цей ключ, щоб дозволити іншим людям переглядати канали та діаграми ваших приватних каналів. Натисніть "Створити новий ключ API читання", щоб створити додатковий ключ читання для каналові.

Поле "Примітка" (Note) використовується для введення інформації про ключі читання каналові, наприклад такі:

– API-запити – https://api.thingspeak.com/update?api_key=TK74EOA1RQRN4B8J&field1=0

– Читання стрічки каналу – https://api.thingspeak.com/channels/2365676/feeds.json?api_key=CIG5OQC7N0XDUWVL&results=2

– Читання поля каналу – https://api.thingspeak.com/channels/2365676/fields/1.json?api_key=CIG5OQC7N0XDUWVL&results=2

– Читання відновлень статусу каналу – https://api.thingspeak.com/channels/2365676/status.json?api_key=CIG5OQC7N0XDUWVL

Незалежно від того, скільки пристроїв з підтримкою Інтернету речей ви додасте, вам буде потрібно використовувати той самий ключ Write API і ідентифікатор каналу.

Наш канал доступний для подальшого використання, натиснувши "Канали">"Мої канали".

4.2 Програмування ESP8266 і ESP32 ThingSpeak

Порядок програмування ESP8266:

1. Завантажите останню версію Arduino® IDE.
2. Додайте ThingSpeak Library for Arduino and ESP8266:
   - виберіть "Ескіз">"Уключити бібліотеку">"Керування бібліотеками";
   - виберіть ThingSpeak, щоб додати його у свій ескіз;
3. Додайте пакет плати ESP8266:
   - у розділі "File">"Preferences" уведіть Додаткові URL-адреси диспетчера дошок https://arduino.esp8266.com/stable/package_esp8266com_index.json;
   - виберіть "Інструменти">"Дошки">"Менеджер дошок". Уведіть ESP8266 у рядок пошуку та встановіть пакет.
4. Виберіть відповідний порт і плату в Arduino IDE. Апаратне забезпечення, використане для створення цього прикладу, використовувало опцію Node MCU 1.0 (ESP 8266–12E).
5. Створіть додаток. Відкрийте нове вікно в Arduino IDE і збережіть файл. Додайте приведений у додатку А код. Обов'язково вкажіть інформацію про бездротові мережі, ідентифікатор каналу, ключ API читання і ключ API запису. Вам не потрібно змінювати коефіцієнти в коді, оскільки програма зчитує їх з вашого каналу. Після підключення пристрій вимірює дані з датчика кожні дві хвилини і відправляє їх на ваш канал.

При кожнім успішному завантаженні даних на послідовному моніторі відображається повідомлення про успіх (рисунок 4.7).

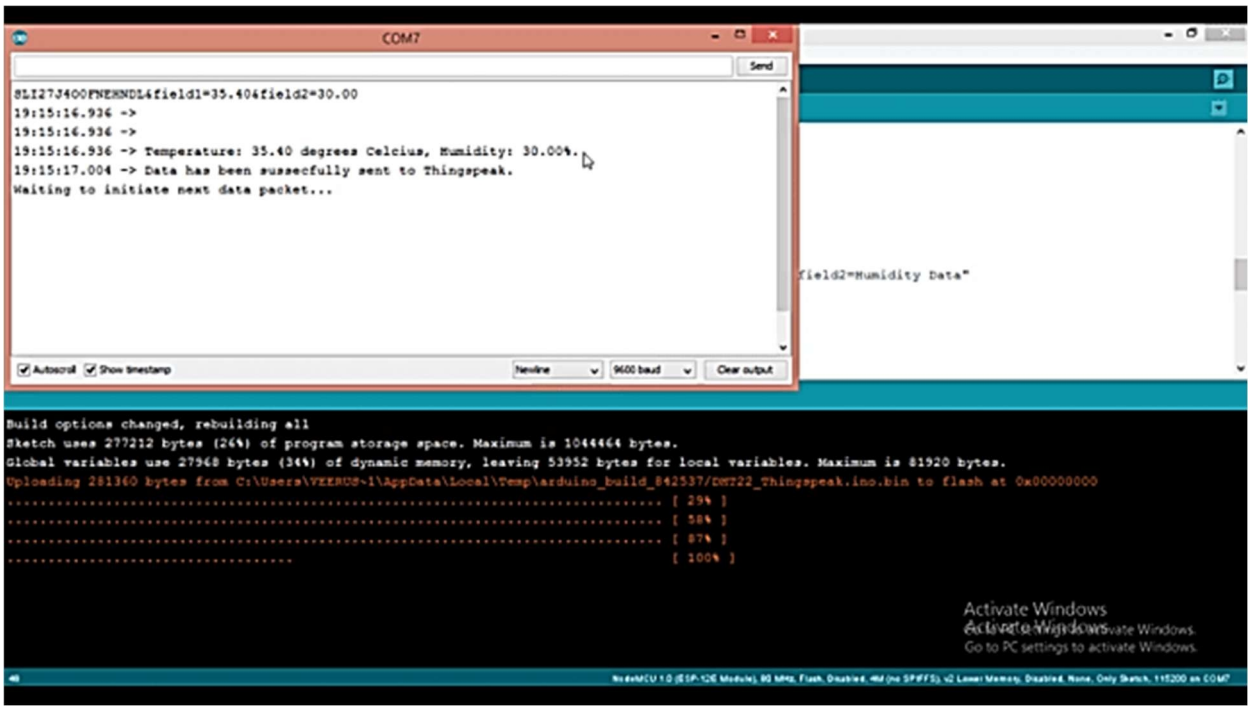

Рисунок 4.7 – Повідомлення на послідовному моніторі

Результати цього проекту можна побачити на Thingspeak. Відкрийте свій канал на Thingspeak, і результати будуть показані, як зазначено на рисунку 4.8.

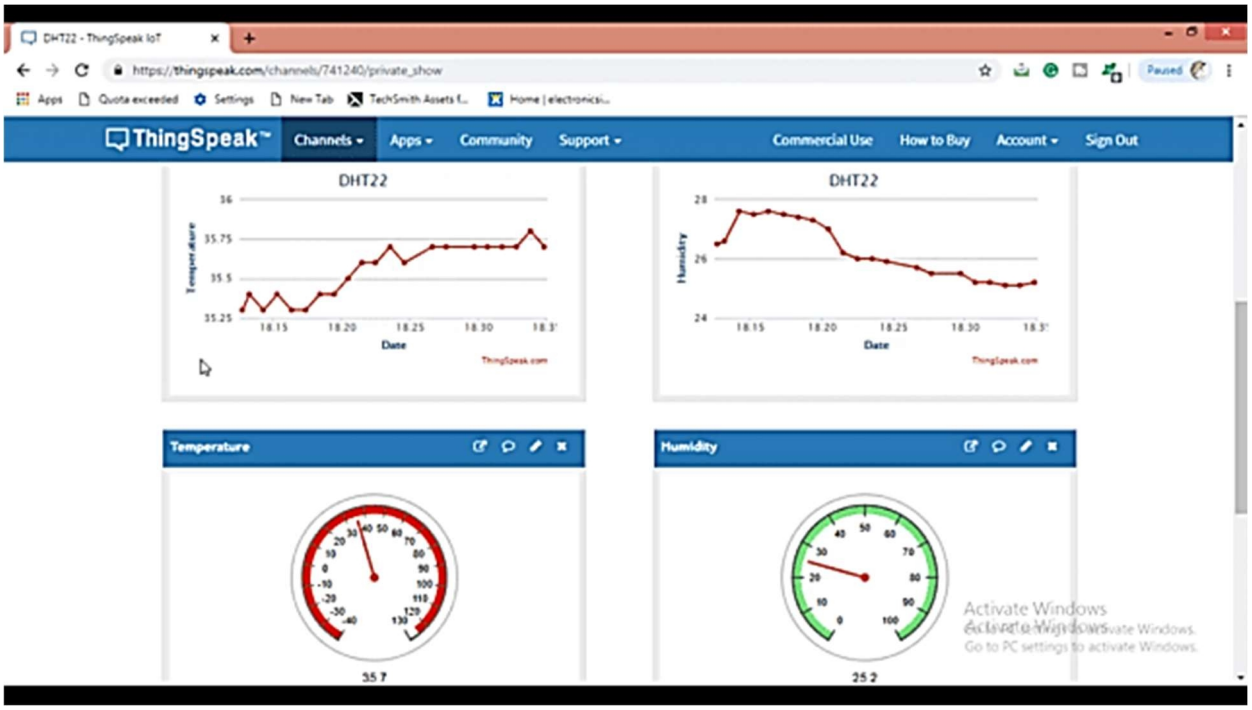

Рисунок 4.8 – Публікація даних DHT22 на Thingspeak за допомогою NodeMCU

ВИСНОВКИ

Метою даної кваліфікаційної роботи була розробка системи збору та візуалізації даних в мережах інтернету речей з застосуванням хмарних технологій.

В першому розділі роботи було проведено огляд платформ хмарних обчислень таких як Amazon Web Services та Microsoft Azure, а також моделі розгортання та обслуговування хмарних обчислень.

В другому розділі роботи розглянуто способи створення ресурсів у хмарі, безпека хмарних ресурсів, обробка великих даних та архітектури традиційних інформаційних систем.

При схемотехнічній розробці системи було проведено обґрунтування вибору мікроконтролера, обрано датчик вологості та температури та точки доступу до хмари та розроблена схема їх з'єднання.

В четвертому, програмному, розділі розглянуто процес створення каналу доступу до хмари на платформі ThingSpeak IoT та розроблена програма роботи ESP8266 і ESP32 ThingSpeak.

Отже, всі поставлені задачі виконано в повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вакалюк Т. А. Види та призначення електронних засобів навчання / Т. А. Вакалюк // Автоматизація та комп’ютерно-інтегровані технології у виробництві та освіті: стан, досягнення, перспективи розвитку: матеріали Всеукраїнської науково-практичної Internet-конференції. – Черкаси, 2014. – С. 110–112.

2. Інфраструктура віртуалізації сервісів на основі кластера СПІІРАН / Бабошин А.А. [та ін.]; Регіональна інформатика (РІ-2010) , XII Санкт-Петербурзька Міжнародна конференція. - Санкт-Петербург, 20–22 жовтня 2010р.: Праці конференції / СПОИСУ. – СПб, 2010. С. 31.

3. Бабич О.А. Обробка інформації в навігаційних комплексах / О.А. Бабич. -М. : Изд-во "Машинобудування". - 1991. - 412 с.

4. Захарін Ф.М. Алгорітмічне забезпечення інерціально-супутникових систем навігації: монографія / Ф.М. Захарін, В.М. Сінєглазов, М.К. Філяшкін. - К.: Вид-во "НАУ-друк", 2011. - 320 с.

5. Яценко В.В., Головань М.С. Хмарні SaaS-сервіси в самостійної роботі з інформатики студентів економічних спеціальностей. [Електронний ресурс] – Режим доступу: http://dspace.uabs.edu.ua/jspui/bitstream /(дата звернення: 10.05.2021). - Хмарні SaaS-сервіси в самостійної роботі з інформатики студентів економічних спеціальностей.

6. Demchenko Y. Defining inter-cloud architecture for interoperability and integration [Text] / Y. Demchenko, C. Ngo, M.X. Makkes, R. Strijkers, C. de Laat // Proceeding of the 3rd International Conference on Cloud Computing, GRIDs and Virtualization, Nice, France, July 22-27, 2012 y. - pp. 174-180

7. Malik N.A. Threat modeling in pervasive computing paradigm [Text] / N.A. Malik, M.Y. Javed, U. Mahmud // Proceedings of the Mobility and Security, New Technologies, Tangier, November 5-7, 2008 y. - pp. 1-5

8. McRee R. PTA: Practical threat analysis [Text] / R. McRee // Proceedings of the Information Systems Security Association, London, September 15 16, 2008 y. - pp. 37-40

9. Bertino E. L. Security for Web Services and Service-Oriented Architectures [Text] / E. L. Bertino // Proceedings of the 2th Annual International Conference on Information Security, New York, USA., September 2-7, 2012 y. - pp. 35-69

10. Soares L.F.B. Secure user authentication in cloud computing management interfaces [Text] / L.F.B. Soares // Proceedings of the IEEE 32nd International Performance Computing and Communications Conference, San Diego, CA, December 6-8, 2013 y. - pp. 1-2.

11. Вишнівський В. В. Підвищення ефективності застосування хмарних сервісів. В. В. Вишнівський, Ю. І. Катков, Ю. В. Каргаполов, Ю. В. Березовська, С. О. Благодир. Видання «Зв'язок», №5, 2020. 5-6 с.

12. Advantages and disadvantages of IaaS [Електронний ресурс] / URL: https://www.hitechnectar.com/blogs/advantages-disadvantages-of-iaasexplained (дата звернення 10.11.2023).

13. Модель сети обратной свертки от Майкрософт. URL: https://docs.microsoft.com/en-us/cognitive-toolkit/Image-Auto-EncoderUsing-Deconvolution-And-Unpooling (дата звернення 10.11.2023).

14. David M. Stein. Software Repackaging and Deployment for the Windows Platform / David M. Stein., 2012. – 146 с. – (Amazon Kindle).

15. Яценко В.В., Головань М.С. Хмарні SaaS-сервіси в самостійної роботі з інформатики студентів економічних спеціальностей. [Електронний ресурс] – Режим доступу: http://dspace.uabs.edu.ua/jspui/bitstream /(дата звернення: 20.11.2023).

16. STM32 32-bit Arm Cortex MCUs https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html (дата звернення 25.11.2023).

17. Medium-density performance line ARM®-based 32-bit MCU https://freedelivery.com.ua/files/datasheets/en.CD00161566.pdf (дата звернення 10.12.2023).

18. Отладочная плата STM32-Smart STM32F103C6T6 Cortex-M3. https://www.mini-tech.com.ua/otladochnaya-plata-smart-stm32 (дата звернення

10.12.2023).

19. DHT22 with NodeMCU. https://electronicsinnovation.com/dht22-with-nodemcu/ (дата звернення 10.12.2023).

20. WIFI МОДУЛЬ ESP8266-PRO, 8MB ROBOTDYN https://miniboard.com.ua/mcu/728-wifi-modul-esp8266-pro-8mb-robotdyn.html (дата звернення 10.12.2023).

21. Wi-Fi модуль NodeMCU V3 ESP8266 ESP-12 (CH340) https://ardushop.in.ua/arduino/wi-fi-module-nodemcu-v3-esp8266-esp-12 (дата звернення 10.12.2023).

22. ESP8266 і ESP32 разом на одній платформі Інтернету речей ThingSpeak https://www.electroniclinic.com/esp8266-and-esp32-together-on-the-same-iot-platform-thingspeak/ (дата звернення 15.12.2023).