ЛАБОРАТОРНА РОБОТА №5

Дослідження платформ для реалізації Smart-систем

Дисципліна «Технології Smart-систем»

1. Тема, мета та результати

Тема: Дослідження платформ для реалізації Smart-систем.

Мета: дослідити архітектуру й можливості IoT-платформ, обґрунтувати вибір для Smart-системи з ЛР №1, розгорнути навчальний прототип із телеметрією, дашбордом та правилом обробки подій.

Результат: порівняльна таблиця трьох платформ; працюючий прототип ThingsBoard Community Edition (або обґрунтована альтернатива), Python-симулятор пристрою, дашборд, правило й результати вимірювання затримки за різної кількості пристроїв.

2. Необхідні програмні засоби

ПК із Windows, macOS або Linux, браузер, Python 3.10+, requests, Docker Desktop/Engine з Docker Compose (для локального варіанта). Для ноутбука без достатніх ресурсів дозволено використати доступну хмарну/демоверсію платформи, якщо її умови допускають навчальне тестування. Реальні ESP32 та шлюзи не потрібні.

Перед початком переконайтеся, що Docker працює та ресурси ПК достатні для обраної інсталяції. Версії й офіційні команди розгортання перевіряйте за актуальною документацією платформи; параметри можуть змінюватися.

3. Короткі теоретичні відомості

3.1. Архітектура IoT-платформ

Типова IoT-платформа має модулі реєстрації й автентифікації пристроїв, приймання телеметрії (HTTP/MQTT), обробки правил, зберігання даних, API та візуалізації. Digital twin — цифрове представлення властивостей і стану фізичного об’єкта; наявність дашборда сама по собі ще не означає повноцінного digital twin.

3.2. Класи платформ і порівняння

Платформа	Клас	Сильні сторони	Обмеження
AWS IoT Core	Хмарна	Керований прийом повідомлень, сертифікати, інтеграції	Хмарні витрати, залежність від AWS
ThingsBoard CE	Open source	Self-hosting, телеметрія, правила, дашборди	Ресурси й адміністрування локального сервера
Blynk	Low-code	Швидкий мобільний інтерфейс і робота з пристроями	Ліміти тарифу, залежність від постачальника

Для самостійного порівняння додайте: MQTT/HTTP, вартість, self-hosting, масштабування, модель доступу, можливості API, наявність OTA (рівень підтримки може відрізнятися за версією та ліцензією). Не плутайте Google Cloud IoT Core з чинним сервісом: він припинив роботу у 2023 році.

3.3. Інтероперабельність і edge

Matter стандартизує прикладну взаємодію сумісних пристроїв розумного дому; Thread — мережевий стек для малопотужних IP-пристроїв. oneM2M визначає сервісну архітектуру IoT; OPC UA поширений у промисловій автоматизації. Edge-рішення на кшталт AWS IoT Greengrass виконують частину логіки ближче до обладнання. Вони не замінюють автоматично хмарний IoT Hub або брокер.

4. Практична частина

4.1. Вибір платформи та сценарію

Візьміть Smart-систему з ЛР №1. Для прикладу застосуємо Smart-теплицю з пристроєм greenhouse-esp32-01 і телеметрією temperature, humidity, battery, як у ЛР №2. Обґрунтуйте вибір ThingsBoard CE за критеріями: робота локально, наявність дашборда й rule engine, відсутність необхідності фізичного пристрою.

4.2. Розгортання ThingsBoard

Оберіть офіційний спосіб локального Docker-розгортання ThingsBoard CE відповідно до поточної версії ОС: https://thingsboard.io/docs/user-guide/install/docker/ . Виконайте інструкцію постачальника, відкрийте вебінтерфейс та увійдіть з обліковими даними, створеними/заданими під час інсталяції. Не використовуйте демонстраційні паролі для публічного сервера.

Після запуску: створіть пристрій greenhouse-esp32-01, скопіюйте його Device Access Token у локальну змінну середовища TB_DEVICE_TOKEN; у звіт токен не вставляйте. Зверніть увагу: навчальний режим може працювати по HTTP localhost, але продакшн-вузол потребує HTTPS/MQTTS та контрольованого доступу.

4.3. Надсилання телеметрії Python-симулятором

ThingsBoard HTTP Device API приймає POST на /api/v1/{ACCESS_TOKEN}/telemetry. Створіть simulator.py. Адреса localhost:8080 наведена як поширений приклад локального доступу й має відповідати фактичному налаштуванню контейнера.

```
import os
import time
import random
from datetime import datetime, timezone
import requests

HOST = os.getenv("TB_HOST", "http://127.0.0.1:8080").rstrip("/")
TOKEN = os.environ["TB_DEVICE_TOKEN"]
URL = f"{HOST}/api/v1/{TOKEN}/telemetry"
rng = random.Random(42)

for i in range(30):
    telemetry = {
        "temperature": round(rng.uniform(18, 32), 1),
        "humidity": round(rng.uniform(35, 75), 1),
        "battery": round(max(0, 100 - 0.2*i), 1),
    }
    start = time.perf_counter()
    try:
```

```
        response = requests.post(URL, json=telemetry, timeout=5)
        response.raise_for_status()
        elapsed_ms = (time.perf_counter() - start) * 1000
        print(datetime.now(timezone.utc).isoformat(),
              telemetry, "HTTP round-trip ms:", round(elapsed_ms, 1))
    except requests.RequestException as exc:
        print("Failed to send:", exc)
    time.sleep(2)
```

Змінна TB_DEVICE_TOKEN у Windows PowerShell задається $env:TB_DEVICE_TOKEN="...", у macOS/Linux — export TB_DEVICE_TOKEN="...". Не плутайте HTTP round-trip із наскрізною затримкою появи на дашборді: сервер може прийняти HTTP-запит раніше, ніж інтерфейс відобразить зміну.

4.4. Створення дашборда та правила

1. У вкладці Devices переконайтеся, що телеметрія надходить (Latest telemetry).

2. Створіть Dashboard та призначте пристрій/alias. Додайте віджет останньої температури, часовий графік температури та графік вологості.

3. У Rule Chains/Rule Engine оберіть відповідний поточній версії механізм обробки повідомлень; створіть правило для temperature > 28 °C з записом alarm або іншим доступним наслідком.

4. Зафіксуйте на скриншоті повідомлення тривоги й подію, що його спричинила. Для автоматичної команди фізичному актуатору потрібний окремий безпечний контур керування; не надсилайте неперевірені команди на реальне обладнання.

Конкретні назви пунктів меню та налаштування Rule Engine можуть відрізнятися між випусками ThingsBoard. Для демонстрації допускається штатне alarm rule у версії платформи, що використовується.

4.5. Дослідження масштабованості

Проведіть три сценарії: 10, 100 і 1000 віртуальних пристроїв. Спершу визначте доступні CPU/RAM та політику обмеження навантаження. 1000 пристроїв — розширений сценарій, лише якщо середовище дозволяє; за браку ресурсів зафіксуйте причину й використайте менший масштаб, зберігши методику. Для кожного сценарію забезпечте окремі токени/ідентичності пристроїв, а не один токен на всі вузли.

Методика: із заданою частотою 1 повідомлення/10 с на пристрій протягом 2–5 хв реєструйте кількість спроб, успішних відповідей, HTTP-коди, середню затримку та p95. Подавайте навантаження поступово; не тестуйте публічні безкоштовні сервіси без дозволу. Для великих N у локальній інсталяції можна оптимізувати генерацію запитів і обмежувати паралелізм.

```
# Приклад фіксації часу одного HTTP-запиту:
start = time.perf_counter()
response = requests.post(URL, json={"temperature": 25.2}, timeout=5)
rtt_ms = (time.perf_counter() - start) * 1000
print(response.status_code, round(rtt_ms, 2))
# Після серії вимірів: p95 = sorted(samples)[int(.95*(len(samples)-1))]
```

Пристроїв	Частота / пристрій	Успішність, %	Середнє RTT, мс	p95 RTT, мс
10	0,1 msg/s	Заповнити	Заповнити	Заповнити
100	0,1 msg/s	Заповнити	Заповнити	Заповнити
1000*	0,1 msg/s	Заповнити	Заповнити	Заповнити

* За наявності достатніх ресурсів. Це бланк для власних результатів, а не готові вимірювання. Окремо опишіть відмінності між числом симульованих пристроїв і фактично зареєстрованих у платформі.

4.6. Безпека та сукупна вартість володіння

Ризик / витрата	Що врахувати
Компрометація токена	Окремий токен пристрою, ротація, відсутність секретів у Git
Незахищений канал	HTTPS/MQTTS, обмеження доступу до локального сервера
Переповнення сховища	Retention policy, обсяг телеметрії, резервне копіювання
Недоступність платформи	Перезапуск сервісів, черга/буфер, спостережуваність
Вартість володіння	Обчислення, диски, трафік, обслуговування, ліцензії, персонал

Оцініть вартість для 10, 100, 1000 пристроїв за власними припущеннями (частота повідомлень, розмір payload, термін зберігання). Для хмарної платформи використовуйте актуальний прайс із датою; для self-hosting — приблизні витрати на сервер та адміністрування.

5. Завдання для самостійного виконання

1. Порівняйте одну хмарну, одну open-source та одну low-code IoT-платформу.

2. Оберіть рішення для Smart-системи ЛР №1 і обґрунтуйте його.

3. Розгорніть платформу локально або на дозволеному хмарному тарифі.

4. Створіть та автентифікуйте віртуальний пристрій; передайте телеметрію Python-скриптом.

5. Налаштуйте дашборд із показниками, історією та сповіщенням за порогом.

6. Дослідіть затримку і частку успішних повідомлень за трьома рівнями навантаження.

7. Проаналізуйте захист пристроїв, вартість володіння, vendor lock-in і можливість перенесення рішення.

6. Вимоги до звіту

1. Тема, мета, опис обраної Smart-системи з ЛР №1.

2. Порівняльна таблиця не менше трьох платформ та обґрунтування вибору.

3. Опис розгортання з інформацією про версію платформи й конфігурацію тестового ПК.

4. Схема обміну даними, фрагменти коду, скриншоти реєстрації пристрою та телеметрії.

5. Дашборд, налаштоване правило й приклад спрацювання.

6. Таблиця фактичних вимірів навантаження, аналіз затримок і обмежень.

7. Оцінка безпеки, сукупної вартості й рекомендації; висновки.

8. Оформлений PDF-звіт завантажити до Системи електронного забезпечення навчання ЗНУ.

7. Контрольні питання

1. Які базові компоненти та функції IoT-платформи?

2. Порівняйте хмарну платформу й self-hosting.

3. Чим цифровий двійник відрізняється від звичайного дашборда?

4. Як відбувається автентифікація пристроїв?

5. Що таке vendor lock-in і як його обмежувати?

6. Яке призначення Matter, Thread, oneM2M і OPC UA?

7. Коли потрібна edge-платформа?

8. Чому ви обрали платформу для власного проєкту?

9. Які показники слід використовувати для оцінки масштабованості?

10. Чим HTTP RTT відрізняється від наскрізної затримки візуалізації?

8. Рекомендовані джерела

• ThingsBoard CE: https://thingsboard.io/docs/

• ThingsBoard Docker install: https://thingsboard.io/docs/user-guide/install/docker/

• ThingsBoard HTTP Device API: https://thingsboard.io/docs/reference/http-api/

• AWS IoT Core: https://docs.aws.amazon.com/iot/

• Blynk documentation: https://docs.blynk.io/

• Matter: https://csa-iot.org/all-solutions/matter/

• oneM2M: https://www.onem2m.org/