МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи № 1
«Проєктування архітектури Smart-систем»
з дисципліни «Технології Smart систем»

Для здобувачів вищої освіти

Запоріжжя — 2026

1. Загальні положення

Тема: Проєктування архітектури Smart-систем.

Мета: спроєктувати апаратну й програмну архітектуру обраної Smart-системи, обґрунтувати технологічні рішення та проаналізувати ризики надійності й кібербезпеки.

Результати навчання: після виконання роботи студент уміє формулювати вимоги, описувати багаторівневу IoT-архітектуру, підбирати компоненти, порівнювати канали зв’язку, створювати UML-діаграми, визначати загрози та пропонувати контрзаходи.

Форма роботи: індивідуальний проєкт. Реальне обладнання не обов’язкове; достатньо аргументованого проєктування, діаграм і технічних специфікацій. Допускаються симулятори та прототипи як додатковий результат.

2. Теоретичні відомості

2.1. Smart-системи та їх архітектура

Smart-система збирає дані за допомогою сенсорів, аналізує стан середовища й виконує керувальні дії автоматично або після підтвердження оператора. Типовий ланцюг: вимірювання → передавання → обробка → рішення → дія → зворотний зв’язок. На відміну від простої системи з таймером, Smart-система враховує актуальні вимірювання, правила та контекст.

Рівень	**Складові**	**Призначення**
Сприйняття	Сенсори, актуатори, ESP32	Вимірювання та вплив на середовище
Мережевий	Wi-Fi, Zigbee, LoRaWAN, шлюз	Доставка телеметрії й команд
Обробки	Edge, локальний сервер, хмара, БД	Логіка, агрегація, збереження
Прикладний	Вебпанель, мобільний застосунок, API	Візуалізація та взаємодія

2.2. Edge, fog і cloud

Edge — обробка безпосередньо на пристрої: наприклад, ESP32 вимикає насос після спрацювання датчика рівня. Fog — обчислення на локальному вузлі або шлюзі, що координує групу пристроїв. Cloud — віддалені сервери для тривалого зберігання, аналітики та доступу з різних мереж. Критичні правила безпеки потрібно виконувати локально, незалежно від доступності Інтернету.

2.3. Протоколи та архітектурні стилі

MQTT — легкий протокол publish/subscribe поверх TCP: пристрій публікує повідомлення у topic, брокер пересилає його підписникам. HTTP/REST — запит/відповідь, зручний для API, керування конфігурацією та вебінтерфейсів. CoAP орієнтований на обмежені пристрої, AMQP

— на складнішу чергову обробку повідомлень. Zigbee і LoRaWAN — мережеві технології; MQTT працює поверх IP-мережі, тому пряме порівняння Zigbee та MQTT без уточнення рівнів некоректне.

Монолітна архітектура простіша для малого прототипу. Подієво-орієнтована архітектура природна для телеметрії; мікросервіси доцільні за потреби незалежного масштабування компонентів, але збільшують складність супроводу.

2.4. Безпека за проєктом

На кожному рівні визначають активи, довірчі межі та загрози. Мінімальний набір заходів: окрема ідентичність кожного пристрою, автентифікація, розмежування доступу до MQTT topics/API, шифрування зв’язку (TLS), захищене зберігання секретів, журналювання, контроль цілісності прошивки, безпечні OTA-оновлення, резервне копіювання й безпечний режим актуаторів при відмові. TLS захищає канал, але сам по собі не замінює авторизації.

3. Завдання лабораторної роботи

1. Опрацювати лекційні матеріали й 3–5 наукових публікацій щодо IoT-архітектур.
2. Обрати Smart-систему (будинок, теплиця, енергосистема, моніторинг здоров’я тощо) та сформулювати вимоги.
3. Спроєктувати апаратну структуру та специфікацію компонентів із орієнтовними цінами.
4. Порівняти щонайменше два практично реалізовні варіанти комунікацій.
5. Описати програмний стек, базу даних, API, безпеку; створити три UML-діаграми: Deployment, Component, Sequence.
6. Побудувати таблицю ризиків і запропонувати заходи їх зниження.
7. Оформити звіт і завантажити його до системи електронного забезпечення навчання ЗНУ.

4. Порядок виконання

Крок 1. Постановка задачі

Сформулюйте проблему, цільову групу користувачів, межі системи та сценарії роботи. Для функціональних вимог використовуйте перевірювані формулювання «Система повинна…». Для нефункціональних — числові показники, якщо це можливо: частота вимірювання, допустима затримка, автономність, доступність, кількість вузлів, вимоги до захисту.

ID	**Приклад вимоги**	**Критерій перевірки**
FR-01	Зчитувати вологість ґрунту	Показник з’являється в системі кожні 60 с
FR-02	Вмикати полив за порогом	Насос запускається за заданої умови

NFR-01	Локальна реакція	Не довше 2 с після вимірювання
NFR-02	Безпечна відмова	За втратою даних насос вимкнено

Числові значення у прикладі — проєктні припущення, а не гарантовані характеристики готового обладнання.

Крок 2. Апаратна структура

Покажіть усі сенсори, актуатори, контролери, шлюз, мережу й джерела живлення. Вкажіть інтерфейси (GPIO, I²C, ADC, UART), робочі напруги, сумісність логічних рівнів і правила підключення навантажень. Насос не можна живити безпосередньо з GPIO ESP32: потрібний драйвер/реле, відповідний блок живлення та захисні компоненти.

Компонент	**Приклад / роль**	**К-сть**	**Орієнтир, USD***
ESP32 DevKit	Контролер і Wi-Fi	1	6–12
Ємнісний датчик вологості	Вхідна телеметрія	1	2–7
BME280	Температура/ вологість повітря	1	4–12
Реле або MOSFET-драйвер	Керування насосом	1	2–8
Насос низької напруги	Актуатор	1	8–20
Блок живлення, кабелі, корпус	Електроживлення та монтаж	1	12–30

* Навчальні оціночні діапазони, не актуальні комерційні пропозиції. У власному звіті студент зазначає продавця, дату та посилання на ціну; перевіряє сумісність конкретних моделей.

Крок 3. Вибір мережі та протоколів

Порівнюйте повний зв’язний варіант, наприклад: A — ESP32 → Wi-Fi → MQTT-брокер; B — Zigbee-вузол → Zigbee-шлюз → MQTT-брокер. MQTT не є альтернативою Zigbee, оскільки вони працюють на різних рівнях.

Критерій	**A: Wi-Fi + MQTT**	**B: Zigbee + шлюз**
Радіодальність	Залежить від точки доступу; типово десятки метрів у приміщенні	Типово десятки метрів на перехід; можливий mesh
Енергоспоживання	Зазвичай вище для постійного з’єднання	Зазвичай нижче для батарейних датчиків

Пропускна здатність	Вища	Нижча; достатня для телеметрії
Інфраструктура	Wi-Fi AP і брокер	Zigbee coordinator/шлюз і брокер
Вартість	Низький поріг входу для ESP32	Додаткова вартість шлюзу
Рекомендація	Живлення від мережі, невеликий об’єкт	Багато батарейних сенсорів

Оцініть залежність вибору від площі, перешкод, доступного живлення, затримки, масштабу та бюджету. Для віддалених об’єктів доцільно окремо розглянути LoRaWAN із власним/операторським шлюзом.

Крок 4. Програмна архітектура

Опишіть прошивку пристрою (зчитування, фільтрація, локальна логіка, зв’язок, watchdog), MQTT-брокер (Mosquitto), сервіс обробки (Python/FastAPI або Node.js), базу даних (PostgreSQL/TimescaleDB чи InfluxDB), вебінтерфейс та сховище логів. Визначте структуру повідомлень, конфігурацію, права доступу та поведінку під час відмови Інтернету.

Приклад MQTT topics: greenhouse/01/telemetry, greenhouse/01/command, greenhouse/01/status. Приклад payload: {"device_id":"greenhouse-01","soil_moisture_pct":34,"timestamp":"2026-10-12T10:00:00Z"}. Показник вологості повинен мати визначену процедуру калібрування; не всі датчики вимірюють його у фізично точних відсотках.

Крок 5. UML-діаграми

Діаграми можна створити у PlantUML, draw.io чи Lucidchart. До звіту включіть зображення діаграм та коротке пояснення. Нижче наведені шаблони PlantUML, які слід адаптувати під власний проєкт.

4.5.1. Deployment — розгортання вузлів

```
@startuml
node "Smart greenhouse" {
  device "ESP32" as esp
  device "Soil sensor" as soil
  device "Pump + driver" as pump
  soil --> esp : ADC
  esp --> pump : GPIO
}
node "Local server" {
  artifact "MQTT broker" as mqtt
  artifact "Backend + database" as backend
}
node "Web browser" as ui
esp --> mqtt : Wi-Fi / MQTT over TLS
mqtt --> backend : telemetry and commands
ui --> backend : HTTPS
@enduml
```

4.5.2. Component — програмні компоненти

```
@startuml
component "Sensor reader" as reader
component "Local controller" as logic
component "MQTT client" as client
component "MQTT broker" as broker
component "API service" as api
database "Time-series storage" as db
component "Dashboard" as ui
reader --> logic : measurements
logic --> client : events
client --> broker : publish/subscribe
broker --> api : telemetry
api --> db : store/query
ui --> api : HTTPS
@enduml
```

4.5.3. Sequence — типовий сценарій поливу

```
@startuml
actor User
participant "ESP32" as E
participant "MQTT Broker" as M
participant "Backend" as B
participant "Dashboard" as D
E -> E : Measure soil moisture
alt Below threshold and safe to irrigate
  E -> E : Enable pump locally
end
E -> M : Publish telemetry (TLS)
M -> B : Deliver measurement
B -> B : Validate and store
B -> D : Updated status (API/polling)
User -> D : View irrigation status
@enduml
```

Перевірте відповідність між схемами: ті самі компоненти й інтерфейси повинні узгоджуватися на всіх трьох UML-діаграмах. За потреби додайте альтернативний сценарій — зникнення Wi-Fi, відмова сенсора або небезпечний рівень води.

Крок 6. Аналіз ризиків

Ризик	**Наслідок**	**Мінімізація**
Втрата Інтернету	Немає віддаленого керування	Локальний автомат, буфер телеметрії
Відмова сенсора	Хибне рішення про полив	Перевірка допустимих значень і тайм-аут
Компрометація MQTT-облікових даних	Несанкціоновані команди	TLS, ACL, індивідуальні ключі, ротація
Зависання ESP32	Припинення керування	Watchdog, safe state, журнал

		перезапусків
Збій живлення	Недоступність системи	Моніторинг живлення, резервування за потреби
Невдала OTA-прошивка	Пристрій не завантажується	Підписані оновлення, rollback, A/B partition

Для кожного ризику визначте ймовірність та вплив за шкалою 1–3 і пріоритет як добуток цих оцінок. Список загроз має відповідати конкретним сценаріям системи.

5. Структура та оформлення звіту

1. Титульна сторінка: ЗВО, дисципліна, назва лабораторної, група, ПІБ студента та викладача, рік.
2. Мета роботи й постановка задачі: користувачі, призначення, межі системи.
3. Короткі теоретичні відомості з посиланнями на 3–5 наукових джерел.
4. Функціональні та нефункціональні вимоги (таблиця з ID).
5. Апаратна структурна схема та специфікація з джерелами цін.
6. Порівняння комунікаційних рішень і обґрунтований вибір.
7. Програмна архітектура, стек, формат телеметрії, API та безпека.
8. Три UML-діаграми з поясненнями.
9. Таблиця загроз/ризиків та захисних заходів.
10. Висновки і список використаних джерел.

Рекомендоване оформлення: аркуш A4, Times New Roman 14 пт, міжрядковий інтервал 1,5, поля 2 см (ліве 3 см за вимогами кафедри), нумерація сторінок, підписи до рисунків і таблиць. Якщо кафедра встановила інші вимоги, вони мають пріоритет. Подання: один PDF через СЕЗН ЗНУ; вихідні файли PlantUML — за вимогою викладача.

6. Контрольні питання

1. Чим Smart-система відрізняється від звичайної автоматизованої системи?
2. Назвіть рівні архітектури IoT і призначення кожного з них.
3. Чим різняться edge, fog і cloud?
4. Порівняйте MQTT і HTTP/REST для телеметрії та команд.
5. Які нефункціональні вимоги критичні для вашої системи?
6. Коли потрібен gateway, а коли можна обійтися без нього?
7. Які загрози виникають у пристроїв, каналу зв’язку й серверів?

8. Що демонструє Deployment Diagram?
9. Як масштабувати систему від 5 до 500 пристроїв?
10. Що відбудеться при втраті Інтернету під час активного керування актуатором?
11. Які механізми забороняють одному пристрою керувати чужими MQTT topics?
12. Чому надійний зв’язок не гарантує фізичної безпеки системи?

7. Рекомендовані джерела

1. Al-Fuqaha A. et al. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials. 2015. DOI: 10.1109/COMST.2015.2444095.
2. Kalyani Y., Collier R. A Systematic Survey on the Role of Cloud, Fog, and Edge Computing Combination in Smart Agriculture. Sensors. 2021. 21(17), 5922. DOI: 10.3390/s21175922.
3. Sicari S. et al. Security, privacy and trust in Internet of Things: The road ahead. Computer Networks. 2015. 76, 146–164. DOI: 10.1016/j.comnet.2014.11.008.
4. PlantUML. Офіційна документація: https://plantuml.com/
5. OASIS. MQTT Version 5.0: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

Наукові публікації 1–3 — приклад стартової бібліографії; для виконання вимоги «3–5 наукових джерел» студент може додати ще 1–2 тематичні статті. Документація PlantUML та MQTT не замінює наукових публікацій.