Міністерство освіти і науки України
Луцький національний технічний університет

IoT-ТЕХНОЛОГІЇ ДЛЯ КІБЕРФІЗИЧНИХ СИСТЕМ

Конспект лекцій
для здобувачів другого (магістерського) рівня вищої освіти
освітньої програми «Комп'ютерна інженерія»
галузь знань F Інформаційні технології
спеціальності F7 Комп’ютерна інженерія
денної та заочної форм навчання

Луцьк 2026

УДК 004.2
І 75

Рекомендовано до видання вченою радою факультету КІТ ЛНТУ,
протокол № _____ від «___ » _____ 20 26 року.

Голова вченої ради факультету КІТ _____ Інна КОНДІУС

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ
Директор бібліотеки _____ Наталія ПОЛІЩУК

Розглянуто і схвалено на засіданні кафедри комп'ютерної інженерії та безпеки ЛНТУ, протокол № _____ від « _____ » _____ 20 26 року.

Завідувач кафедри КІБ _____ Тарас ТЕРЛЕЦЬКИЙ

Укладачі: _____ Сергій ГРИНЮК, кандидат технічних наук,
Доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

_____ Сергій КОСТЮЧКО, кандидат технічних наук,
кафедри комп'ютерної інженерії та безпеки ЛНТУ

Рецензент: _____ Андрій ЯЩУК, кандидат технічних наук,
доцент кафедри інженерії програмного забезпечення ЛНТУ

Відповідальний за випуск: _____ Тарас ТЕРЛЕЦЬКИЙ, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

І 75 **IoT-технолгії для кіберфізичних систем**: Конспект лекцій для здобувачів другого (магістерського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузь знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання / уклад. С.В. Гринюк Луцьк: ЛНТУ, 2026. 48 с.

Конспект лекцій з дисципліни «**IoT-технології для кіберфізичних систем**»: складений відповідно до діючої програми курсу. Призначений для здобувачів другого (магістерського) рівня вищої освіти спеціальності F7 Комп'ютерна інженерія освітньої програми «Комп'ютерна інженерія».

.

ЗМІСТ

ТЕМА 1. CPS ТА IOT ЯК ОСНОВА ІНДУСТРІЇ 4.0

1. Основні принципи організації та функціонування екосистем Інтернету речей та кіберфізичних систем

Екосистеми Інтернету речей (IoT) та кіберфізичних систем (CPS) є складними мережами пристроїв, які забезпечують обмін даними між фізичним і цифровим світом. Вони використовуються для моніторингу, керування та автоматизації різноманітних процесів у промисловості, сільському господарстві, транспорті, розумних містах та інших сферах.

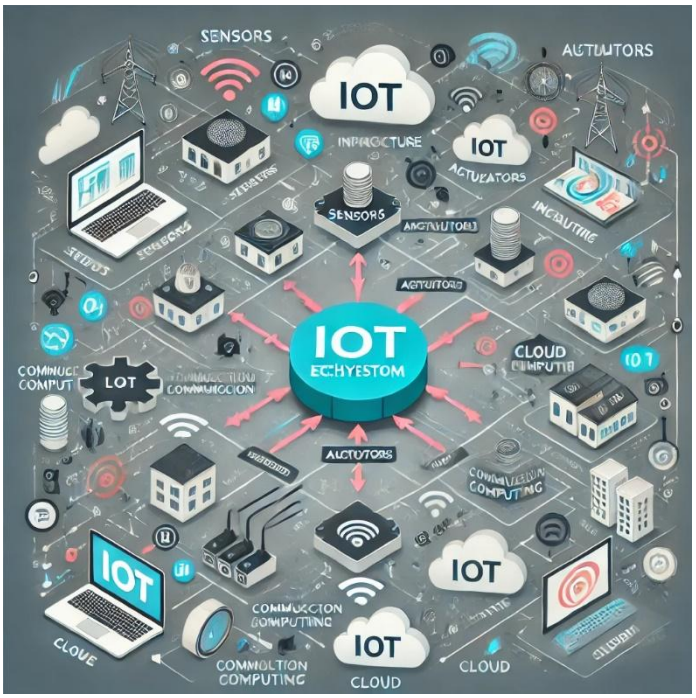

Рисунок 1.1 – Загальна схема екосистеми IoT

Основні компоненти екосистем IoT та CPS:

- Сенсори та виконавчі механізми: Ці компоненти відповідають за взаємодію з фізичним середовищем. Сенсори збирають інформацію, а виконавчі механізми виконують команди, що надходять від системи управління.
- Обчислювальні пристрої: До них відносяться контролери, мікропроцесори та інші пристрої, які обробляють дані, отримані від сенсорів, та приймають рішення на основі цих даних.
- Комунікаційна інфраструктура: Канали передачі даних між сенсорами, виконавчими механізмами, обчислювальними пристроями та хмарними сервісами. До них належать різні мережеві протоколи (Wi-Fi, 4G/5G, Zigbee тощо).

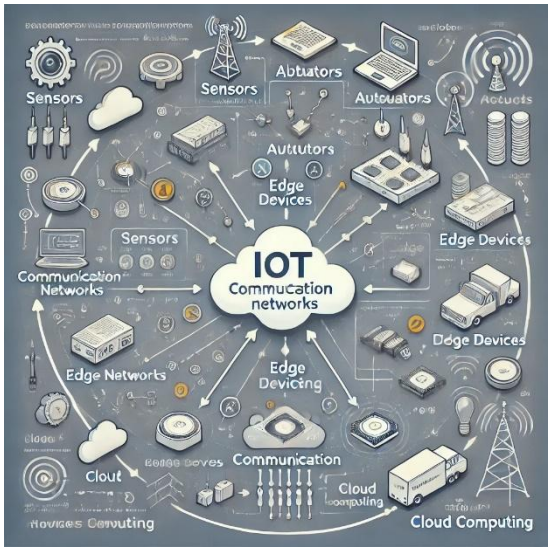

Рисунок 1.2 – Основні компоненти IoT систем

Принципи функціонування екосистем IoT та CPS:

- Збирання даних: Сенсори безперервно збирають дані про навколишнє середовище або про стан об'єктів (температура, тиск, вологість, рух тощо).
- Обробка та аналіз даних: Дані, отримані від сенсорів, передаються на обчислювальні пристрої або у хмару для аналізу. Це може бути як простий аналіз (середні показники), так і складніший (передбачення на основі машинного навчання).
- Прийняття рішень: На основі аналізу даних система може автоматично приймати рішення, наприклад, вмикати охолодження, якщо температура перевищує певний поріг.
- Дії: Виконавчі механізми виконують команди, такі як відкриття/закриття клапанів, вмикання/вимикання пристроїв або надсилання сповіщень користувачам.

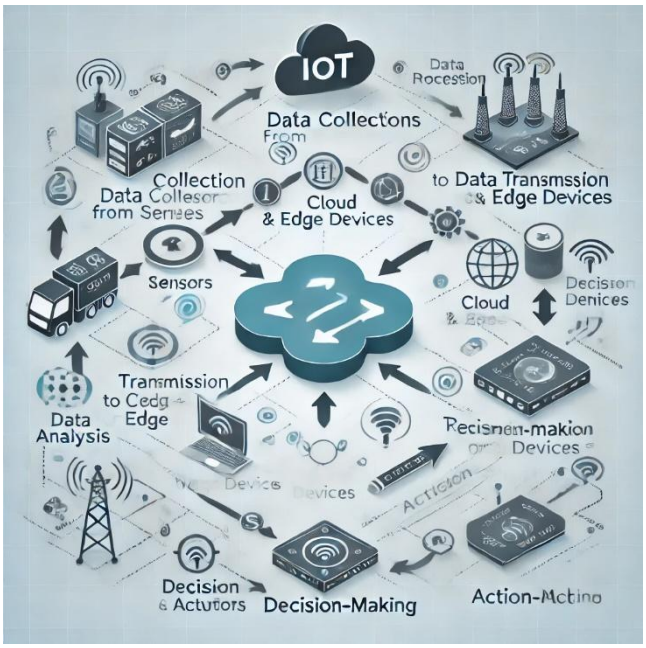

Рисунок 1.3 – Процес збирання, обробки та прийняття рішень в IoT та CPS системах

Підключення та комунікація в IoT та CPS.

Одним з ключових аспектів організації та функціонування екосистем IoT та CPS є надійна комунікація між елементами системи. Це досягається через використання різних

протоколів передачі даних:

- MQTT (Message Queuing Telemetry Transport): Легкий протокол обміну повідомленнями, який широко використовується в IoT завдяки низькому споживанню енергії та ефективності.
- CoAP (Constrained Application Protocol): Протокол для обмежених пристроїв, що дозволяє обмін даними в малопотужних і обмежених мережах.
- LPWAN (Low Power Wide Area Network): Технології на зразок LoRaWAN, які забезпечують зв'язок між пристроями на великі відстані з низьким енергоспоживанням.

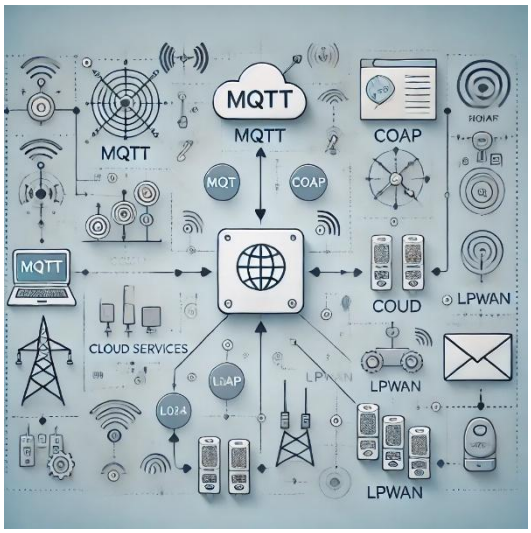

Рисунок 1.4 – Протоколи зв'язку для IoT

Архітектура кіберфізичних систем

Кіберфізичні системи використовують інтеграцію фізичних компонентів з вбудованими комп'ютерними системами для здійснення моніторингу та керування в реальному часі. Архітектура CPS включає кілька рівнів:

- Фізичний рівень: Обладнання, що безпосередньо взаємодіє з фізичним світом.
- Кібернетичний рівень: Комп'ютерні системи та алгоритми, які забезпечують прийняття рішень на основі даних з фізичного рівня.
- Рівень взаємодії з користувачем: Інтерфейси, через які користувачі можуть керувати системою або отримувати від неї інформацію.

Організація та функціонування екосистем IoT і CPS залежать від взаємодії між фізичними пристроями, програмним забезпеченням та комунікаційними мережами. Основними викликами цих екосистем є забезпечення безпеки, надійності та ефективності передачі даних.

2. Системний підхід до аналізу та синтезу структур IoT та CPS.

Системний підхід до аналізу та синтезу структур Інтернету речей (IoT) та кіберфізичних систем (CPS) дозволяє розглядати ці системи як інтегровані та взаємозалежні компоненти, що взаємодіють між собою. Це допомагає глибше зрозуміти процеси в системах, оптимізувати їхню роботу та створювати нові функціональні структури.

Основні принципи системного підходу

1) Цілісність: Система повинна розглядатися як єдине ціле, де всі компоненти взаємопов'язані.
2) Ієрархічність: IoT та CPS мають багаторівневу архітектуру, яка включає фізичний, комунікаційний, обчислювальний і прикладний рівні.
3) Модульність: Система складається з окремих блоків, які можуть бути замінені або

модернізовані.

Рисунок 1.5 – Системний підхід до аналізу та синтезу структур IoT та CPS

Аналіз структури систем полягає в оцінці всіх компонентів, їх взаємозв'язків та функціональних можливостей. Основні кроки:

Ідентифікація компонентів: Визначаються всі складові системи – сенсори, виконавчі механізми, обчислювальні пристрої, комунікаційні мережі.

Оцінка функціональності: Аналізуються можливості кожного компонента щодо збору даних, передачі інформації та прийняття рішень.

Оцінка надійності: Перевіряється стійкість системи до збоїв і її здатність до відновлення після можливих відмов.

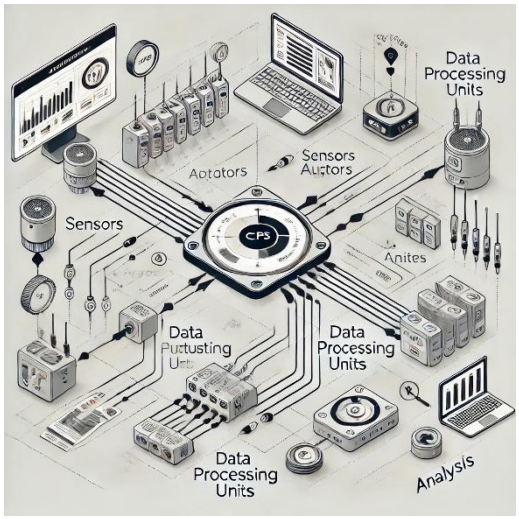

Рисунок 1.6 – Аналіз структури CPS системи

Синтез систем передбачає створення або модернізацію архітектури IoT та CPS для досягнення визначених цілей. Основні етапи:

1) Проектування архітектури: Створення моделі системи з урахуванням взаємодії компонентів.
2) Оптимізація ресурсів: Використання ефективних алгоритмів для мінімізації

використання енергії та пропускної здатності мереж.

3) Забезпечення безпеки: Розробка заходів для захисту від кібератак та витоків даних.

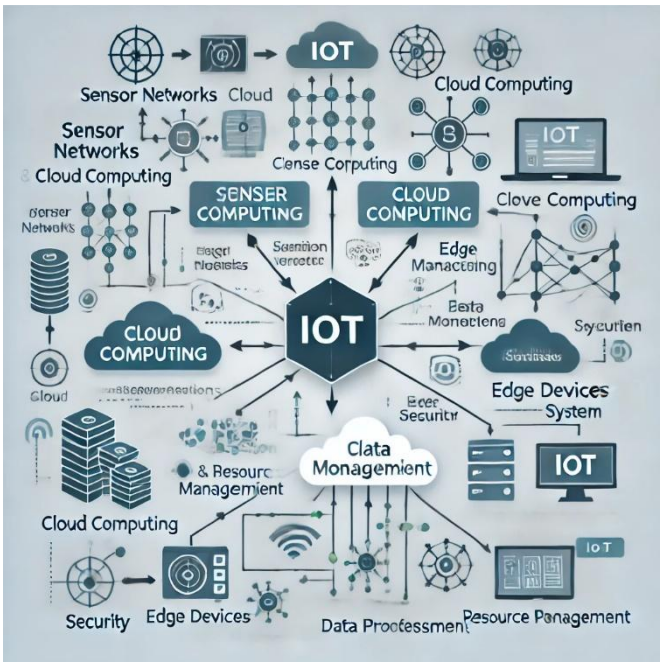

Рисунок 1.7 – Синтез структури IoT системи

До методів системного аналізу та синтезу належать:

1) Математичне моделювання: Використання математичних моделей для опису поведінки системи.
2) Імітаційне моделювання: Моделювання процесів в системі за допомогою програмних інструментів.
3) Оптимізація: Використання методів лінійного програмування для пошуку найкращих рішень.

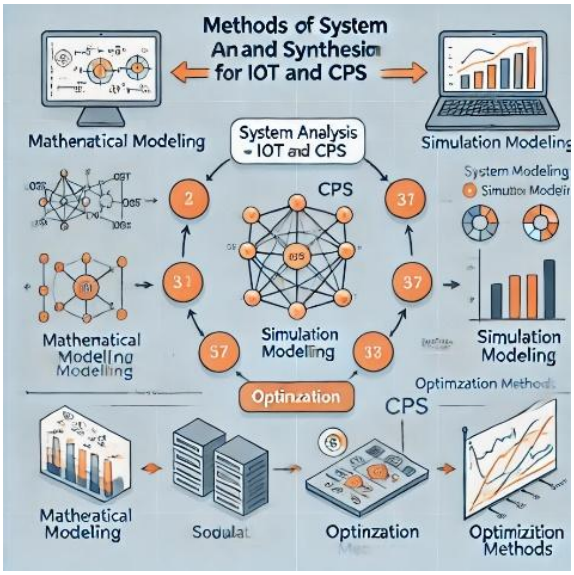

Рисунок 1.8 – Методи системного аналізу та синтезу IoT та CPS

Системний підхід до аналізу та синтезу структур IoT та CPS дозволяє більш ефективно створювати, впроваджувати та керувати цими системами. Він забезпечує цілісність,

масштабованість та стійкість, що є критичними для сучасних кіберфізичних систем.

3. Математичне та інформаційне забезпечення технологій IoT та CPS

Математичне та інформаційне забезпечення технологій IoT та кіберфізичних систем (CPS) є основою для ефективної роботи цих систем. Використання математичних моделей дозволяє описувати та прогнозувати поведінку систем, а інформаційне забезпечення гарантує точну обробку та управління даними.

Математичні моделі в IoT та CPS допомагають описувати різні процеси: від збирання даних до прийняття рішень та контролю. Основні напрямки математичного забезпечення:

1) Моделювання процесів: Використання диференційних рівнянь, стохастичних моделей та мереж Петрі для опису процесів в реальному часі, таких як передача даних та управління виконавчими механізмами.
2) Теорія графів: Використовується для моделювання складних мереж зв'язку між IoT-пристроями та фізичними компонентами CPS.
3) Математичне програмування та оптимізація: Для пошуку оптимальних рішень у використанні ресурсів системи, таких як енергоспоживання, пропускна здатність мережі та ефективність обчислень.

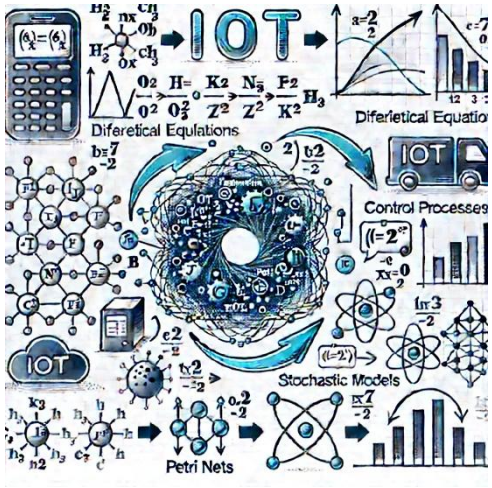

Рисунок 1.9 – Математичне моделювання IoT та CPS систем

Інформаційне забезпечення складається з набору методів та інструментів для збору, зберігання, обробки та передачі даних. Основні компоненти:

1) Системи управління даними: Це платформи, які забезпечують зберігання і аналіз великих обсягів даних, отриманих від сенсорів IoT.
2) Протоколи обміну даними: До них належать MQTT, CoAP, та RESTful API, що дозволяють обмінюватися даними між пристроями в межах системи.
3) Хмарні сервіси та обчислення на межі (Edge Computing): Хмари використовуються для аналізу великих обсягів даних, а обчислення на межі дозволяють виконувати швидкі обчислення ближче до сенсорів.

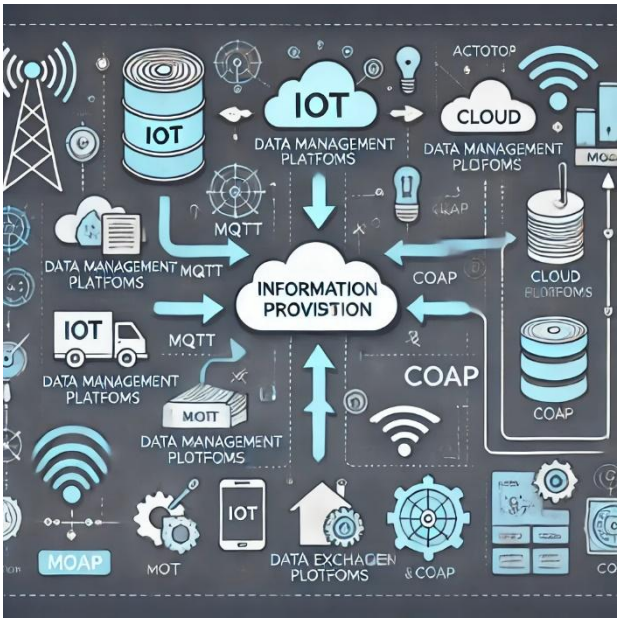

Рисунок 1.10 – Інформаційне забезпечення для IoT систем

Ефективне функціонування IoT та CPS значною мірою залежить від правильного збору та обробки даних:

1) Обробка даних в реальному часі: Використання алгоритмів для обробки даних, що надходять від сенсорів без затримок.
2) Машинне навчання: Методи машинного навчання застосовуються для прогнозування подій, виявлення аномалій та прийняття рішень на основі зібраних даних.
3) Аналіз великих даних (Big Data): Системи Big Data допомагають обробляти та аналізувати величезні обсяги даних, які генерують IoT пристрої та CPS.

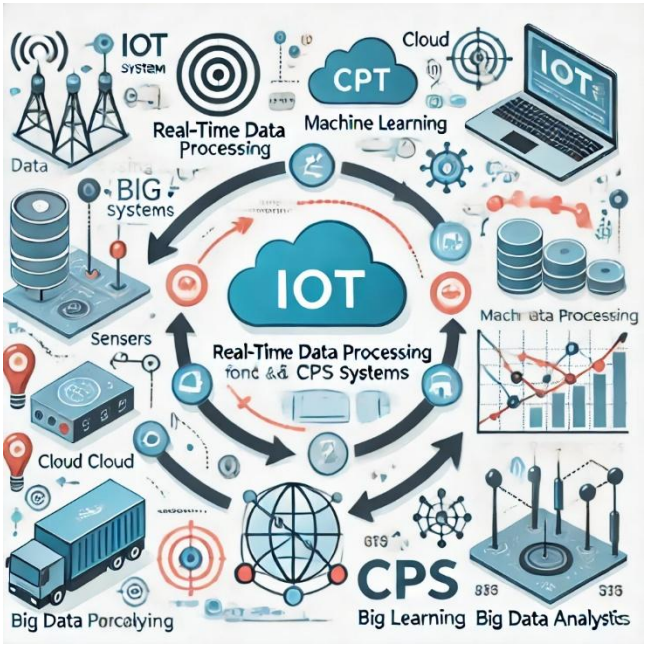

Рисунок 1.11 – Методи аналізу та обробки даних для IoT та CPS

Зважаючи на велику кількість даних, що обробляються в IoT та CPS, питання безпеки стає критичним. Методи захисту включають:

1) Шифрування даних: Використання алгоритмів шифрування для забезпечення конфіденційності даних.

2) Аутентифікація: Механізми перевірки ідентичності пристроїв та користувачів в системі.
3) Виявлення аномалій: Алгоритми машинного навчання, які допомагають виявляти підозрілі дії або вторгнення в систему.

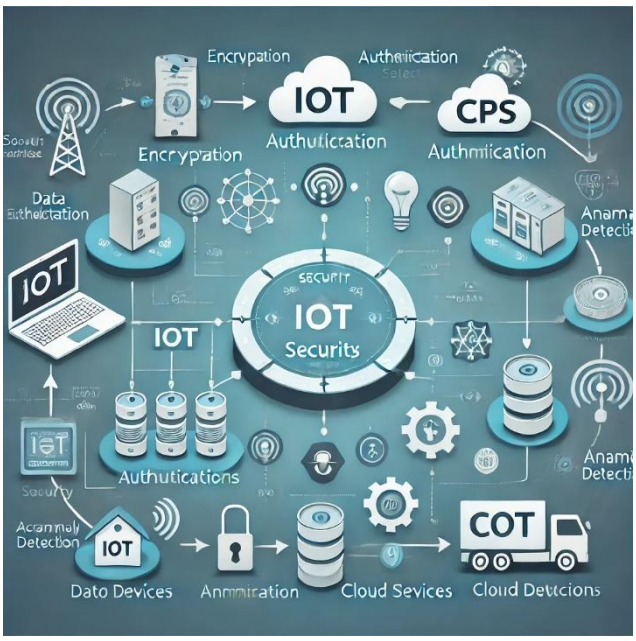

Рисунок 1.12 – Методи безпеки в IoT та CPS

Математичне та інформаційне забезпечення є основою для ефективної роботи IoT та CPS. Воно включає моделювання процесів, управління даними та забезпечення безпеки систем, що дозволяє знизити ризики та підвищити ефективність використання ресурсів.

ТЕМА 2. ІОТ-ТЕХНОЛОГІЇ В ЗАДАЧАХ СИНТЕЗУ ТА АНАЛІЗУ КІБЕРФІЗИЧНИХ СИСТЕМ

1. Сучасна елементна та технологічна база для CPS та IoT інтерфейси відкритих систем та мережеві протоколи IoT

Кіберфізичні системи (CPS) та Інтернет речей (IoT) використовують різноманітні технологічні елементи для взаємодії з фізичним світом і цифровою інфраструктурою. Основою таких систем є сенсори, виконавчі механізми, контролери, а також комунікаційні протоколи та відкриті інтерфейси, що забезпечують обмін даними між компонентами.

Елементна база CPS та IoT включає:

1) Сенсори: Вимірювальні пристрої, які збирають дані про фізичні параметри (температура, вологість, тиск, світло тощо). Наприклад, сенсори температури DHT11, датчики руху PIR, фотодатчики тощо.
2) Мікроконтролери та мікропроцесори: Вони обробляють дані від сенсорів і приймають рішення. До прикладу, контролери сімейств Arduino, STM32, ESP8266/ESP32 широко використовуються в IoT проєктах.
3) Виконавчі механізми: Це пристрої, що реагують на команди від системи — мотори, реле, серводвигуни тощо, які виконують фізичні дії.
4) Комунікаційні модулі: Модулі, що забезпечують бездротовий зв'язок, такі як Wi-Fi (ESP8266), Zigbee (Xbee), Bluetooth (HM-10), LoRa тощо.

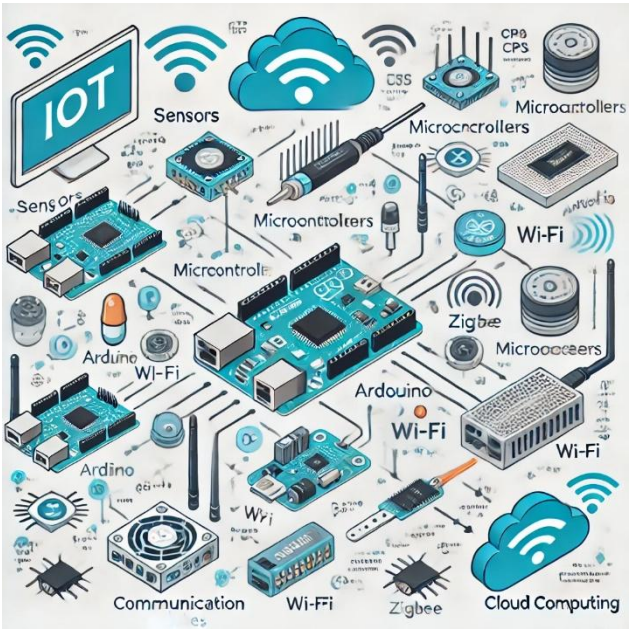

Рисунок 2.1 – Елементна база для IoT та CPS систем

Технологічна база визначає, як компоненти взаємодіють між собою і з зовнішніми системами:

1) Обчислювальна база: Використовуються різні обчислювальні платформи, які підтримують IoT-додатки, наприклад, Raspberry Pi для локальних обчислень або хмарні платформи (AWS IoT, Microsoft Azure IoT).
2) Edge Computing: Ця технологія дозволяє обробляти дані ближче до джерела їх генерації (сенсорів), що знижує затримку в прийнятті рішень і розвантажує хмару.

3) Технології штучного інтелекту: Алгоритми AI використовуються для обробки даних у

реальному часі та прогнозування подій на основі зібраних даних.

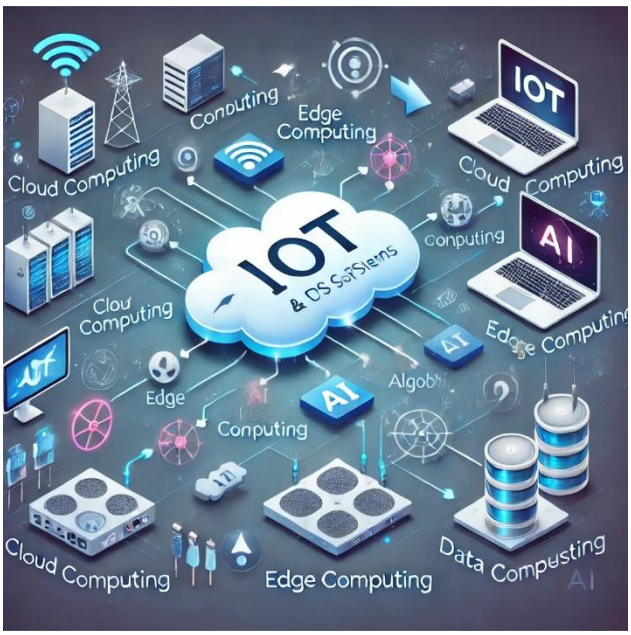

Рисунок 2.2 – Технологічна база для IoT та CPS

Інтерфейси відкритих систем дозволяють різним IoT пристроям і системам обмінюватися даними незалежно від їхнього виробника. До таких інтерфейсів відносяться:

1) API (Application Programming Interface): Інтерфейс, що дозволяє програмам обмінюватися даними. REST API є найпоширенішим для IoT завдяки своїй простоті та масштабованості.
2) MQTT (Message Queuing Telemetry Transport): Легкий протокол передачі повідомлень, який часто використовується для зв'язку між пристроями в IoT через мінімальне споживання ресурсів.
3) CoAP (Constrained Application Protocol): Протокол, що розроблений для малопотужних пристроїв у обмежених мережах. Він забезпечує передачу даних в реальному часі з мінімальними затримками.

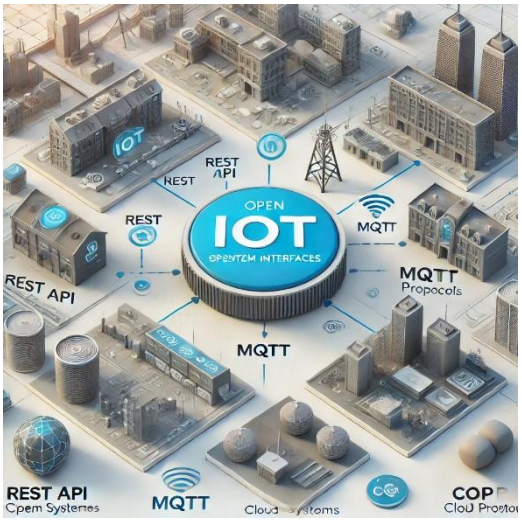

Рисунок 2.3 – Інтерфейси відкритих систем для IoT

Мережеві протоколи є важливою складовою IoT, оскільки вони забезпечують зв'язок між пристроями:

1) Zigbee: Безпровідний протокол для передачі даних на малих відстанях з низьким споживанням енергії. Широко використовується в розумних будинках.
2) LoRaWAN: Протокол для передачі даних на великі відстані з мінімальним споживанням енергії. Застосовується для створення міських IoT-мереж.
3) NB-IoT (Narrowband IoT): Протокол для роботи в стільникових мережах, розроблений для пристроїв з низьким енергоспоживанням та невеликими обсягами даних.

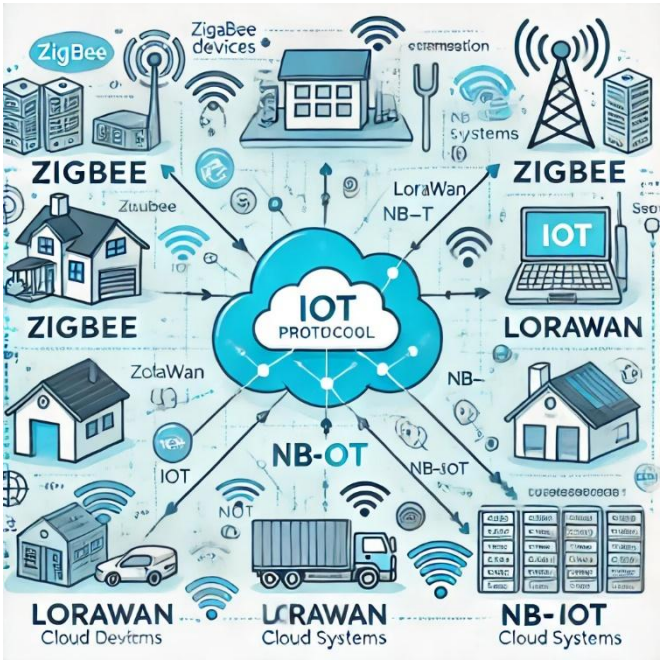

Рисунок 2.4 – Мережеві протоколи для IoT

Сучасна елементна та технологічна база для CPS та IoT включає широкий спектр компонентів, від сенсорів до складних хмарних систем. Інтерфейси відкритих систем і мережеві протоколи IoT забезпечують надійну та безпечну комунікацію між пристроями, що дозволяє створювати високоефективні, масштабовані системи.

2. Спеціалізовані програмні пакети для моделювання та синтезу IoT та CPS

Спеціалізовані програмні пакети для моделювання та синтезу IoT (Інтернету речей) та CPS (кіберфізичних систем) відіграють ключову роль у проектуванні, тестуванні та оптимізації складних систем. Ці програмні інструменти дозволяють візуалізувати структури IoT/CPS, аналізувати їхні процеси і забезпечити інтеграцію фізичних і цифрових компонентів.

Програмні пакети для моделювання IoT та CPS

MATLAB є одним з найпотужніших інструментів для моделювання фізичних процесів та їх математичного аналізу. Simulink — розширення MATLAB для графічного моделювання динамічних систем. Використовується для моделювання IoT-систем, електронних пристроїв, обчислювальних процесів, а також для синтезу керуючих алгоритмів.

Програмне середовище, яке використовується для моделювання фізичних процесів в кіберфізичних системах. ANSYS дозволяє моделювати механічні, теплові, електромагнітні і рідинні процеси.

Використовується для симуляції різноманітних компонентів IoT та CPS, зокрема для моделювання сенсорів, актюаторів та інфраструктури.

Хмарна платформа для моделювання IoT-систем, що дозволяє створювати віртуальні

пристрої, підключати їх до хмарних сервісів і тестувати їх в умовах реальних сценаріїв.
Використовується для перевірки масштабованості та тестування IoT мереж без фізичного обладнання.

Cisco Packet Tracer – інструмент для мережевого моделювання, який дозволяє проектувати та симулювати IoT та CPS мережі.
Він дозволяє користувачам моделювати роботу пристроїв і мережевих протоколів в реальних умовах, створюючи віртуальні мережі та системи.

Програмні пакети для синтезу IoT та CPS

Tinkercad

Онлайн-платформа для моделювання і прототипування електронних пристроїв. Підтримує роботу з Arduino і сенсорами, дозволяє синтезувати IoT-проекти та налаштовувати взаємодію з різноманітними компонентами. Використовується для швидкого створення прототипів IoT-систем і CPS.

LabVIEW

Програмне середовище для візуального моделювання, яке дозволяє проектувати вимірювальні та керуючі системи для IoT та CPS. Забезпечує інтеграцію з фізичними пристроями, сенсорами та контролерами.
Особливо корисний для створення систем реального часу і контролю кіберфізичних систем.

OpenModelica

Це середовище для моделювання та синтезу динамічних систем. Його можна використовувати для моделювання складних процесів в кіберфізичних системах, таких як енергетичні системи або системи автоматизації.
Використовується для багатофізичних симуляцій і оптимізації систем IoT та CPS.

CoppeliaSim (V-REP)

Платформа для робототехнічного моделювання, яка широко використовується для створення роботизованих систем, що інтегруються з IoT і CPS. Вона підтримує симуляцію реального часу та має можливість підключення до фізичних пристроїв.

Спеціалізовані програмні пакети є незамінними інструментами для проектування, тестування та впровадження IoT та CPS. Вони дозволяють:

- Створювати віртуальні моделі складних систем без необхідності фізичних прототипів.
- Виконувати симуляції роботи в умовах реальних сценаріїв.
- Оптимізувати роботу IoT мереж та кіберфізичних систем за допомогою синтезу керуючих алгоритмів та комунікаційних протоколів.

Спеціалізовані програмні пакети для моделювання та синтезу IoT і CPS дають змогу розробникам швидко і якісно моделювати, тестувати та впроваджувати складні системи. Вони допомагають досягати високої продуктивності та забезпечувати надійність роботи як окремих компонентів, так і систем в цілому.

ТЕМА 3. ПЕРЕТВОРЮВАЧ НА ОСНОВІ POWER-OVER-ETHERNET МЕРЕЖІ ДЛЯ КІБЕРФІЗИЧНИХ СИСТЕМ

1. Інтернет речей та масштабованість кіберфізичних систем

Інтернет речей (IoT) і кіберфізичні системи (CPS) тісно пов'язані з концепцією взаємодії фізичних і цифрових світів. Масштабованість є важливим аспектом цих систем, оскільки дозволяє їх адаптувати до різних рівнів складності та кількості пристроїв. Ефективне масштабування систем є ключовим для підтримання надійності, продуктивності та безпеки у великих інфраструктурах.

Масштабованість IoT відображає здатність системи розширюватися без втрати продуктивності чи надійності. Основні виклики масштабованості IoT:

1) Кількість пристроїв: У великих IoT-системах може бути підключено тисячі або навіть мільйони пристроїв. Важливо забезпечити ефективну обробку даних та комунікацію між ними.
2) Мережеве навантаження: Зростання кількості пристроїв збільшує обсяг трафіку в мережі. Потрібно використовувати протоколи, які оптимізують використання смуги пропускання, такі як MQTT, CoAP або LPWAN.
3) Обробка даних: Величезні обсяги даних вимагають надійної інфраструктури для їхнього аналізу, що включає обчислення на межі (Edge Computing) і хмарні платформи (Cloud Computing).

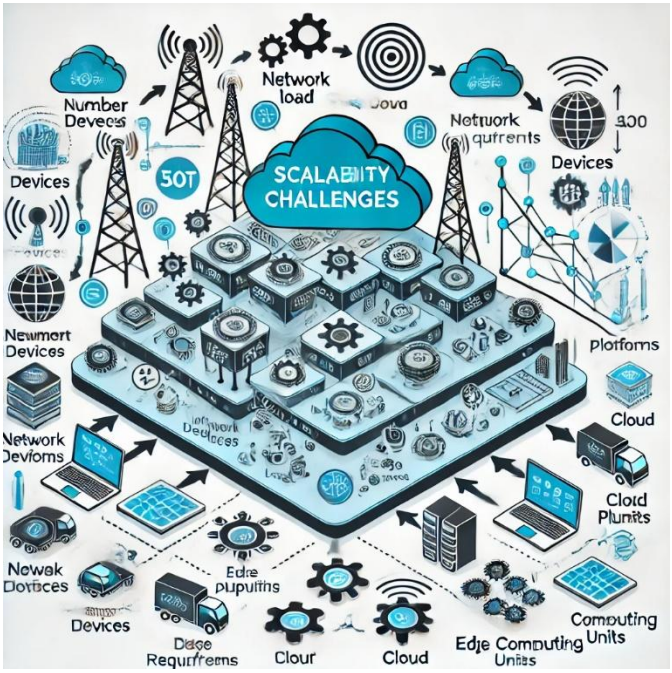

Рисунок 3.1 – Масштабованість в IoT системах

Стратегії масштабування IoT:

1) Горизонтальне масштабування: Додаються нові пристрої та вузли, при цьому кожен новий елемент працює незалежно від інших.
2) Вертикальне масштабування: Підвищується продуктивність окремих компонентів системи, наприклад, збільшення потужності серверів для обробки більшого обсягу даних.
3) Модульність: Розбиття системи на модулі дозволяє масштабувати її поетапно, додаючи нові елементи або вдосконалюючи існуючі.

Кіберфізичні системи також вимагають масштабованості через складність інтеграції фізичних та інформаційних компонентів. Основні аспекти масштабованості CPS:

1) Обчислювальні ресурси: У CPS системах велике значення має швидкість обробки даних та ефективне управління обчислювальними ресурсами.
2) Комунікаційні протоколи: Масштабованість CPS залежить від можливості комунікаційних протоколів забезпечувати стабільну передачу даних між фізичними та цифровими компонентами.
3) Безпека: Масштабованість CPS повинна враховувати питання кібербезпеки, адже зростання кількості пристроїв збільшує ймовірність атак на систему.

Рисунок 3.2 – Масштабованість кіберфізичних систем

Технології для забезпечення масштабованості IoT та CPS

1) Хмарні технології (Cloud Computing)

Хмарні платформи, такі як AWS IoT, Microsoft Azure IoT та Google Cloud IoT, забезпечують масштабованість систем через розподіл обчислювальних ресурсів. Хмари дозволяють зберігати, обробляти та аналізувати великі обсяги даних, що генерують IoT пристрої та CPS.

2) Обчислення на межі (Edge Computing)

Обчислення на межі знижує навантаження на хмару, дозволяючи виконувати частину обчислень ближче до джерела даних, тобто до сенсорів і пристроїв.

3) Мережеві протоколи

Мережеві протоколи, такі як Zigbee, LoRaWAN, NB-IoT, забезпечують масштабованість через низьке споживання енергії та високу ефективність передачі даних на великі відстані.

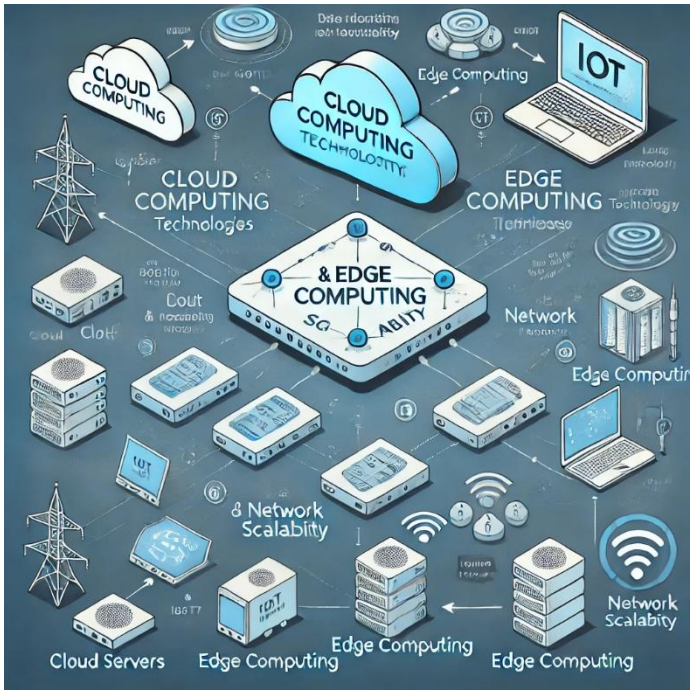

Рисунок 3.3 – Хмарні та edge технології для масштабованості

Зростання кількості IoT-пристроїв та CPS елементів вимагає підвищеної уваги до безпеки та продуктивності. Основні проблеми:

- Безпека: Збільшення числа пристроїв підвищує ризик кібератак. Тому масштабованість повинна супроводжуватися використанням надійних методів шифрування та аутентифікації.
- Продуктивність: Щоб система залишалась ефективною при збільшенні навантаження, необхідно оптимізувати використання обчислювальних та мережевих ресурсів.

Масштабованість IoT та кіберфізичних систем є критично важливою для забезпечення їхньої надійності та продуктивності в умовах зростання числа підключених пристроїв. Використання хмарних технологій, обчислень на межі, ефективних мережевих протоколів і заходів безпеки дозволяє адаптувати системи до змінного навантаження.

2. Концепція та переваги технології Power over Ethernet

Технологія Power over Ethernet (PoE) дозволяє передавати електроживлення та дані одночасно через стандартний Ethernet-кабель. Це значно спрощує інфраструктуру мереж та знижує витрати на прокладку окремих кабелів для живлення. PoE широко використовується в системах IP-камер, точках доступу Wi-Fi, телефонах VoIP та інших мережевих пристроях.

PoE використовує існуючі Ethernet-кабелі для передачі живлення разом з даними. Технологія відповідає стандартам IEEE 802.3af, 802.3at (PoE+), а також новішому стандарту 802.3bt (PoE++). Ключові компоненти PoE-систем:

- PSE (Power Sourcing Equipment): Джерело живлення, яке забезпечує електроживлення для підключених пристроїв. Це може бути PoE-комутатор або інжектор живлення.
- PD (Powered Device): Пристрій, який отримує живлення від Ethernet-кабелю. Це можуть бути IP-камери, точки доступу, телефони VoIP та інші IoT пристрої.

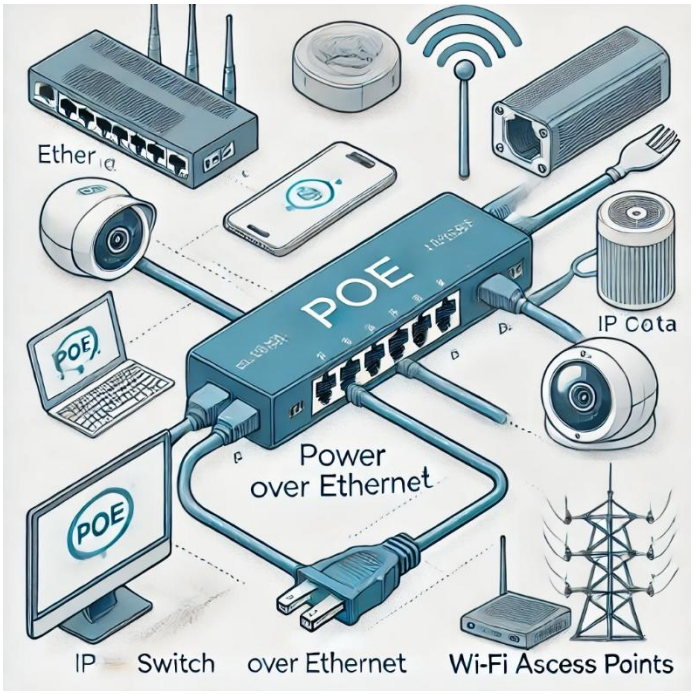

Рисунок 3.4 – Концепція технології PoE

PoE використовує 2 або 4 пари проводів Ethernet-кабелю (в залежності від стандарту). Для передачі живлення використовуються ті ж кабелі, що і для даних, що дозволяє уникнути необхідності прокладання додаткових силових кабелів. Основні етапи:

1) Визначення пристрою: PSE виявляє, чи підтримує підключений пристрій PoE.
2) Передача живлення: Якщо пристрій сумісний, подається живлення через кабель, який також використовується для передачі даних.
3) Управління потужністю: PSE контролює потужність, яку отримує PD, щоб уникнути перевантаження.

PoE дозволяє скоротити кількість кабелів, необхідних для живлення та передачі даних, оскільки використовується один Ethernet-кабель. Це особливо важливо в умовах, коли важко або дорого прокладати додаткові кабелі для електроживлення. Завдяки PoE пристрої можуть бути встановлені в місцях, де немає розеток або доступу до електромережі, наприклад, на стінах, стелях або віддалених ділянках будівель. PoE зменшує витрати на прокладку окремих електричних кабелів та установку розеток, що особливо актуально для великих інфраструктурних проектів. Крім того, використання PoE дозволяє знизити витрати на обслуговування. PoE дозволяє централізовано керувати живленням пристроїв через PoE-комутатор. Це зручно для віддаленого вимкнення або перезавантаження пристроїв, що підвищує керованість мережі.

Рисунок 3.5 – Переваги використання PoE

Приклади використання PoE

1) IP-камери

PoE дозволяє живити та підключати камери відеоспостереження через один кабель, що полегшує встановлення систем безпеки.

2) Точки доступу Wi-Fi

Багато точок доступу Wi-Fi використовують PoE для забезпечення живлення та передачі даних, що дозволяє встановлювати їх у важкодоступних місцях.

3) IoT-пристрої

PoE ідеально підходить для живлення різноманітних IoT-пристроїв, які часто встановлюються на віддалених локаціях і потребують надійного з’єднання з мережею.

Максимальна довжина Ethernet-кабелю для PoE становить 100 метрів. Це може бути обмеженням для великих об'єктів або проектів, що вимагають покриття великих площ.

Стандарти PoE мають обмеження на потужність. Наприклад, стандарт 802.3af підтримує до 15.4 Вт, а PoE+ (802.3at) — до 30 Вт. Це може бути недостатнім для потужних пристроїв.

PoE є інноваційною технологією, яка значно спрощує інфраструктуру мереж та знижує витрати на їхнє розгортання. Вона дозволяє легко підключати та живити різноманітні пристрої через Ethernet-кабель, забезпечуючи при цьому гнучкість, надійність і централізоване управління мережею.

ТЕМА 4. СИСТЕМНА ІНЖЕНЕРІЯ НА ОСНОВІ МОДЕЛЕЙ ДЛЯ КІБЕРФІЗИЧНИХ СИСТЕМ

1. Методології моделювання для GPS

Кіберфізичні системи (CPS) — це інтеграція обчислювальних систем із фізичними процесами. Вони використовуються в різноманітних галузях, таких як транспорт, енергетика, медицина і промисловість. Моделювання CPS допомагає аналізувати їхню поведінку, проектувати та оптимізувати системи для досягнення високої надійності та продуктивності.

CPS складаються з таких основних елементів:

1) Фізична частина: Пристрої, сенсори, виконавчі механізми, які здійснюють фізичні дії.
2) Кібернетична частина: Алгоритми управління, обробка даних, системи комунікацій.
3) Інтерфейси: Засоби інтеграції кібернетичної та фізичної частини для забезпечення взаємодії в реальному часі.

Моделювання CPS є комплексним, оскільки необхідно враховувати взаємодію між фізичними та кібернетичними компонентами, а також вплив затримок у передачі даних, реальних фізичних умов та обмежень.

Моделювання на основі системного підходу

Цей підхід передбачає розгляд CPS як інтегрованої системи, що складається з кібернетичних і фізичних елементів, де враховуються їхні взаємодії. Основні етапи:

1) Аналіз компонентів: Визначення всіх елементів системи, їхніх функцій та взаємодій.
2) Інтеграція модулів: Розробка моделей, що об’єднують фізичні та кібернетичні процеси.
3) Аналіз взаємодій: Вивчення затримок, ефективності комунікацій та можливих збоїв.

MATLAB/Simulink є одним із найпопулярніших інструментів для моделювання CPS. Він дозволяє створювати моделі фізичних і кібернетичних компонентів, об’єднуючи їх у єдину систему для симуляції в реальному часі. Simulink надає можливість моделювати фізичні процеси, а також алгоритми управління та комунікації.

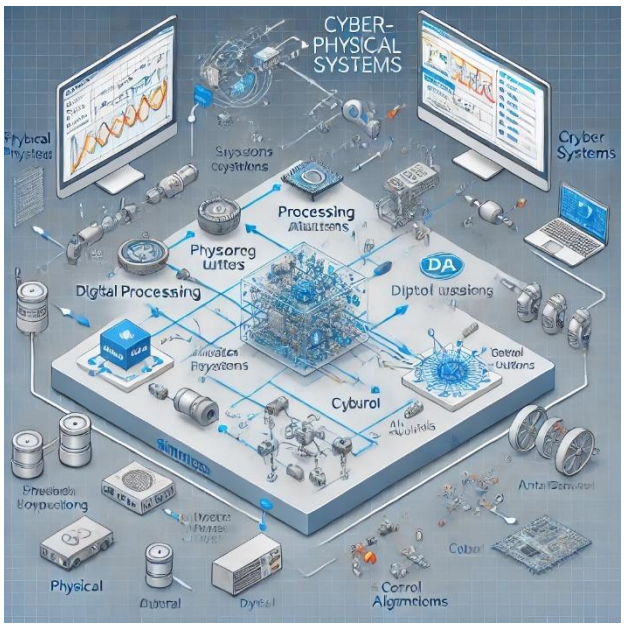

Рисунок 4.1 – Моделювання CPS у Simulink

Імітаційні моделі дозволяють віртуально відтворювати поведінку CPS у різних умовах. Це забезпечує можливість тестування систем до їх фізичної реалізації. Інструменти для імітаційного моделювання включають:

1) AnyLogic: Платформа для мультиагентного моделювання складних систем.

2) OPNET: Використовується для моделювання мережевих комунікацій між компонентами CPS.

Мережі Петрі є математичним інструментом для моделювання синхронізованих систем. Вони використовуються для моделювання взаємодії між процесами та синхронізації дій між кібернетичними й фізичними компонентами CPS.

Переваги: Можливість аналізу паралельних процесів, виявлення конфліктів та оптимізація часу виконання.

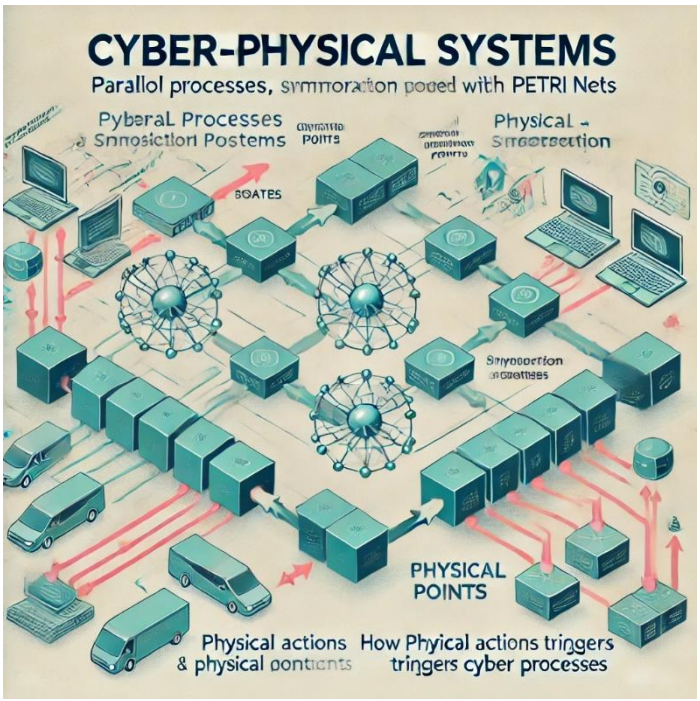

Рисунок 4.2 – Моделювання CPS за допомогою мереж Петрі

Інструменти для моделювання CPS

LabVIEW дозволяє створювати вимірювальні системи та системи управління для CPS. Його можливості включають роботу з фізичними сенсорами та алгоритмами керування в реальному часі.

Це середовище для моделювання динамічних систем, яке використовується для симуляції механічних, електричних та теплових процесів. OpenModelica дозволяє моделювати складні CPS з високою точністю.

CoppeliaSim (V-REP)

Система для робототехнічного моделювання, яка підтримує симуляції в реальному часі та інтеграцію з реальними пристроями. Вона широко використовується для моделювання роботів, що є частиною CPS.

Методи верифікації та тестування CPS

CPS часто використовують у критично важливих системах, тому тестування їхньої роботи в реальному часі є ключовим аспектом. Це включає перевірку взаємодії між фізичними та кібернетичними компонентами.

Одним із важливих параметрів CPS є час реакції на події, особливо в системах, де затримки можуть призвести до критичних наслідків. Для аналізу цього параметра використовуються різні алгоритми оптимізації, такі як фільтр Калмана.

CPS широко застосовуються в системах автоматизації промислових процесів, таких як автоматичні виробничі лінії, енергетичні системи та транспортні мережі.

Інтеграція IoT з CPS дозволяє створювати розумні міста, розумні будинки, а також

системи моніторингу та управління критично важливими інфраструктурами.

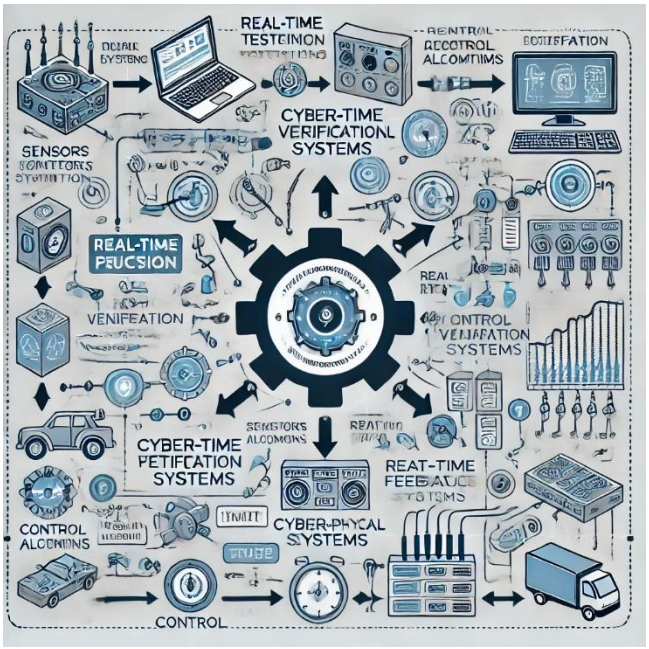

Рисунок 4.3 – Впровадження CPS у реальних умовах

Моделювання CPS є важливим інструментом для проектування та оптимізації складних систем. Використання різних методів моделювання, таких як MATLAB/Simulink, мережі Петрі та імітаційні моделі, дозволяє створювати надійні й ефективні системи, які можуть працювати в реальному часі та взаємодіяти з фізичним світом.

2. Методології моделювання для GPS

MARTE (Modeling and Analysis of Real-Time and Embedded systems) — це профіль UML, який розширює можливості моделювання систем реального часу та вбудованих систем. MARTE надає засоби для опису часових обмежень, управління ресурсами та оптимізації вбудованих систем. Використовується для аналізу продуктивності та моделювання апаратних і програмних компонентів систем.

UML (Unified Modeling Language) — це стандартна мова моделювання, яка використовується для опису, проектування та документування програмних і апаратних систем. Основні типи діаграм UML:

- Діаграми класів: Відображають структуру системи, її класи та взаємозв'язки.
- Діаграми послідовностей: Моделюють обмін повідомленнями між об'єктами в часі.
- Діаграми діяльностей: Показують процеси або потоки робіт у системі.
- Діаграми станів: Відображають стани об'єкта та переходи між ними.

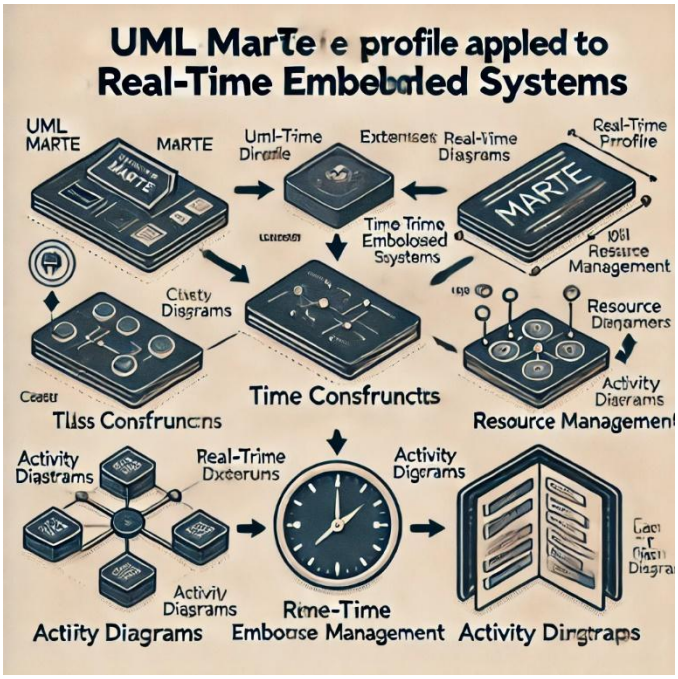

Рисунок 4.4 – Основні типи діаграм UML

MARTE — це стандарт UML для моделювання та аналізу систем реального часу і вбудованих систем. Він надає додаткові елементи для:

- Опису часових характеристик.
- Моделювання використання ресурсів.
- Аналізу продуктивності та енергоспоживання.

MARTE додає нові специфікації до стандартних UML-діаграм для вбудованих систем. Основні концепції MARTE:

- Ціноутворені моделі ресурсів (RSM): Дозволяють моделювати ресурси, такі як процесори, пам'ять, і моделювати їхню взаємодію.
- Часові обмеження (Time): Дозволяють моделювати часові аспекти систем, включаючи часові затримки та періодичні події.

MARTE використовується в поєднанні з класичними UML діаграмами, зокрема:

- Діаграми діяльностей: Використовуються для моделювання потоків управління з урахуванням часових аспектів.

- Діаграми компонентів: Описують апаратні компоненти та їхні інтерфейси, зокрема для вбудованих систем.

MARTE додає можливість моделювати часові обмеження, наприклад, затримки, цикли обробки та виконання завдань у реальному часі. Це особливо важливо для вбудованих систем, де час виконання критичний.

MARTE дозволяє моделювати розподіл ресурсів у системах реального часу, включаючи процесорний час, пам'ять і енергоспоживання. Наприклад, можна вказати, який компонент має доступ до певного ресурсу, а також які затримки можуть виникнути.

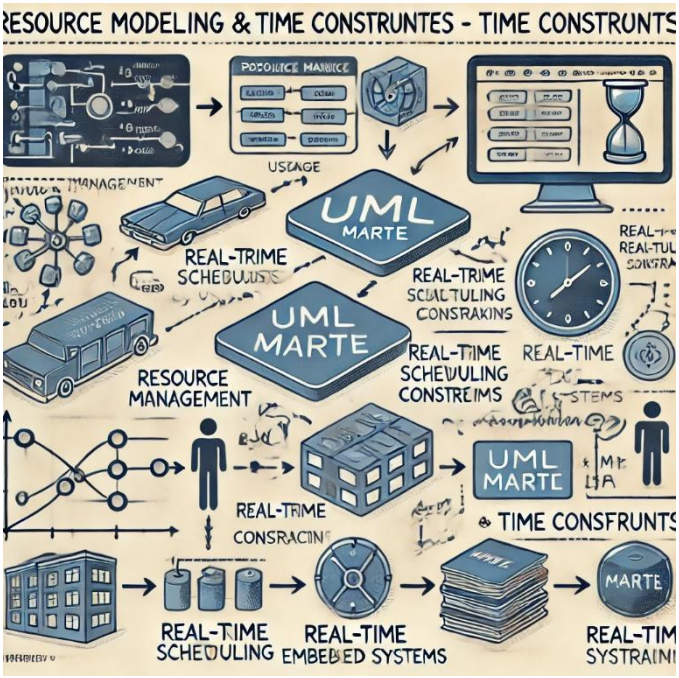

Рисунок 4.5 – Моделювання часових обмежень з MARTE

MARTE також дозволяє аналізувати продуктивність систем за допомогою спеціальних розширень для вимірювання часу виконання процесів, використання ресурсів та енергоспоживання. Це особливо корисно для вбудованих систем, де ефективність роботи є ключовою.

Продуктивність системи: Визначає, наскільки швидко і ефективно система виконує завдання.

Енергоспоживання: Аналіз енергетичних затрат при виконанні процесів для систем з обмеженими ресурсами.

MARTE часто використовується для моделювання вбудованих систем, таких як мікроконтролери або системи на кристалі. Завдяки цьому можна описувати час виконання команд, доступ до пам'яті та інші критичні аспекти.

В авіаційній галузі MARTE використовується для моделювання систем реального часу, які управляють польотами, зв'язком та іншими критичними процесами, де час реакції має ключове значення.

MARTE є потужним інструментом для розширення UML та моделювання систем реального часу і вбудованих систем. Він дозволяє моделювати часові обмеження, розподіл ресурсів і аналізувати продуктивність, що особливо корисно для систем з обмеженими ресурсами і критичними вимогами до часу виконання.

ТЕМА 5. ПРОГРАМНІ ЗАСОБИ ДЛЯ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ МОДЕЛЮВАННЯ

1. Моделювання IoT-пристроїв на базі платформи Arduino

Платформа Arduino є однією з найпопулярніших для створення IoT-проєктів. Завдяки відкритому апаратному та програмному забезпеченню, вона дозволяє швидко моделювати та прототипувати пристрої для Інтернету речей (IoT), з'єднуючи їх з датчиками та виконавчими механізмами для збору й обміну даними.

Arduino складається з апаратної та програмної частин:

1) Апаратна частина: Мікроконтролери сімейства Arduino (UNO, Mega, Nano), до яких можна підключати різні сенсори, виконавчі механізми та модулі зв'язку.
2) Програмна частина: Arduino IDE, що дозволяє програмувати пристрої мовою C/C++ та контролювати їхню роботу.

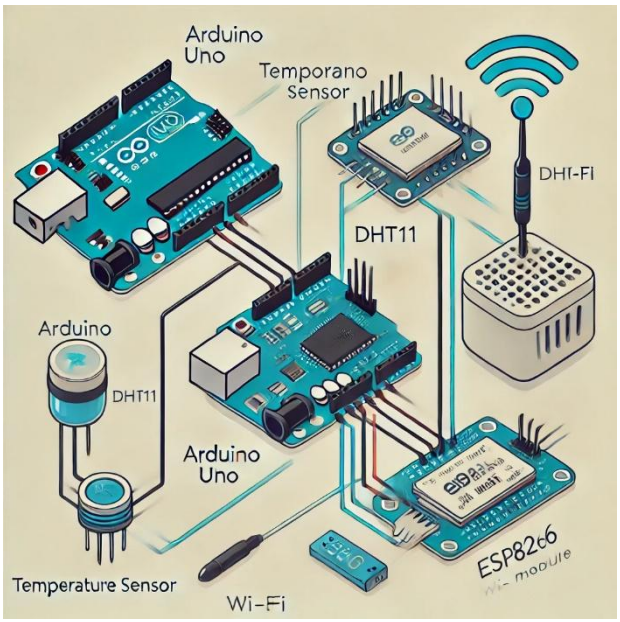

Рисунок 5.1 – Основні компоненти платформи Arduino для IoT

Для створення IoT-проєктів на базі Arduino необхідні модулі для передачі даних:

- ESP8266/ESP32: Модулі Wi-Fi для бездротового підключення до мережі Інтернет.
- GSM/GPRS модулі: Для підключення до Інтернету через мобільні мережі.
- Ethernet Shield: Для підключення Arduino до мережі через кабель Ethernet.

Для моделювання IoT-пристрою на базі Arduino необхідно вибрати компоненти, такі як:

- Сенсори (температури, вологості, руху, освітлення) для збору даних.
- Виконавчі механізми (реле, мотори, серводвигуни) для виконання дій.
- Модулі зв'язку для передачі даних в Інтернет.

Програмування IoT-пристроїв на Arduino здійснюється за допомогою мови Arduino C/C++. Програма, яку називають "скетчем", містить основні інструкції для роботи пристрою:

- Підключення бібліотек: Для сенсорів та модулів зв'язку використовуються спеціальні бібліотеки, що спрощують програмування.
- Налаштування: Функція setup() налаштовує пристрій (ініціалізація сенсорів, налаштування мережі).
- Основний цикл: Функція loop() безперервно виконує інструкції (зчитування даних, передача їх на сервер, керування виконавчими механізмами).

Для збору та обробки даних від IoT-пристроїв на Arduino використовуються хмарні платформи, такі як:

- ThingSpeak: Платформа для збору та аналізу даних з IoT-пристроїв.
- Adafruit IO: Хмарна платформа для візуалізації даних і управління IoT-пристроями.
- Blynk: Платформа для створення мобільних додатків для управління IoT.

Arduino — це ідеальна платформа для швидкого прототипування IoT-пристроїв. Завдяки великій кількості бібліотек та модулів для підключення до мережі, вона дозволяє легко створювати проекти для моніторингу та управління різними процесами. Інтеграція з хмарними сервісами дає змогу збирати та аналізувати дані в реальному часі.

2. Розробка програмного забезпечення. Скетч Arduino C/C++.

Arduino C/C++ — це мова програмування, яка використовується для написання скетчів (програм) для мікроконтролерів на платформі Arduino. Вона базується на стандартних бібліотеках C/C++ і забезпечує простоту написання коду для роботи з сенсорами, виконавчими механізмами та модулями зв'язку.

Програма на Arduino складається з двох основних функцій:

- setup(): Ця функція виконується один раз після запуску або перезавантаження пристрою. Вона використовується для початкової ініціалізації компонентів.
- loop(): Функція loop() виконується безперервно, поки працює мікроконтролер. Вона містить основну логіку програми, наприклад, зчитування даних з сенсорів та управління виконавчими механізмами.

Arduino має велику кількість бібліотек, які спрощують роботу з різними модулями та сенсорами. Для підключення бібліотеки використовується директива #include. Наприклад, для роботи з Wi-Fi модулем ESP8266 необхідно підключити відповідну бібліотеку:

```
#include <ESP8266WiFi.h>
```

Arduino дозволяє працювати з цифровими піномі для зчитування сигналів і керування виконавчими пристроями:

- **pinMode(pin, mode)**: Встановлює режим роботи піну (вхід/вихід).
- **digitalWrite(pin, value)**: Виводить сигнал на цифровий пін (HIGH або LOW).
- **digitalRead(pin)**: Читає стан цифрового піну (HIGH або LOW).

Для відлагодження програм на Arduino використовується серійний монітор в IDE, який дозволяє виводити дані для аналізу роботи пристрою.

- Serial.begin(baud_rate): Ініціалізує серійний зв'язок з вказаною швидкістю (наприклад, 9600 бод).
- Serial.print() / Serial.println(): Виводить дані в серійний монітор для моніторингу роботи програми.

Мова програмування Arduino C/C++ є зручною та ефективною для написання скетчів для мікроконтролерів. Вона дозволяє швидко створювати програми для роботи з сенсорами, керування виконавчими механізмами та підключення до Інтернету за допомогою Wi-Fi модулів.

ТЕМА 6. ТРИРІВНЕВЕ МОДЕЛЮВАННЯ СИСТЕМ НА ОСНОВІ IOT/IOE З ВИКОРИСТАННЯМ UML-ДІАГРАМ, МЕРЕЖ ПЕТРІ ТА ЧАСОВОЇ ЛОГІКИ

1. Моделювання та верифікація в архітектурі систем на основі IoT та IoE систем з використанням візуальних UML-діаграм

Моделювання та верифікація архітектури систем на основі IoT (Інтернету речей) та IoE (Інтернет усього) є критично важливими для забезпечення надійності та ефективності їхньої роботи. UML (Unified Modeling Language) забезпечує зручний спосіб візуалізації компонентів системи, потоків даних і взаємодії між ними, що дозволяє точно моделювати процеси і верифікувати їх на різних етапах розробки.

UML забезпечує різні типи діаграм для опису структури, поведінки та взаємодії в IoT/IoE системах:

- Діаграми класів використовуються для опису статичної структури IoT/IoE системи. Вони відображають класи, атрибути, методи та взаємозв'язки між компонентами:
- Класи пристроїв IoT: Описують різні типи сенсорів, контролерів, виконавчих механізмів та їхні атрибути (наприклад, тип сенсора, потужність).
- Взаємозв'язки: Моделюють взаємодії між пристроями, контролерами та серверами.

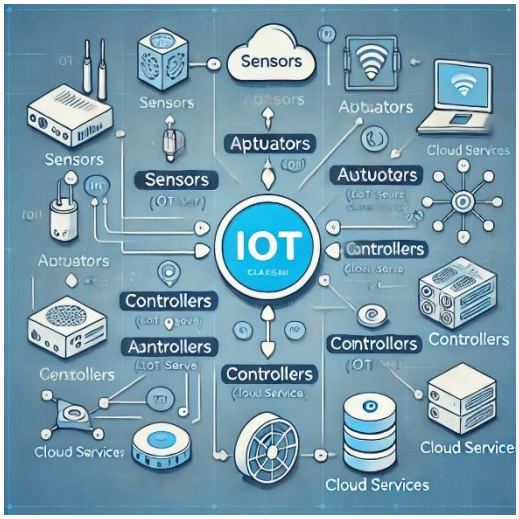

Рисунок 6.1 – Діаграма класів для IoT-систем

Діаграми послідовностей відображають процес обміну повідомленнями між об'єктами системи в часі. Вони дозволяють моделювати сценарії, такі як збір даних з сенсорів та передача їх до хмарної платформи. Приклади сценаріїв: Запит даних від сенсора, обробка даних на сервері, передача команд від сервера до виконавчих механізмів.

Рисунок 6.2 – Діаграма послідовностей для IoT-системи

Діаграми діяльності відображають процеси та потоки робіт у системах IoT та IoE. Вони використовуються для моделювання алгоритмів обробки даних і прийняття рішень на основі отриманих показників. Приклад: Процес зчитування даних з сенсора, обробка на edge-пристрої та передача до хмари.

IoT системи складаються з великої кількості фізичних та цифрових компонентів, таких як сенсори, мережеві інтерфейси, обчислювальні платформи та хмарні сервіси. Використання діаграм компонентів дозволяє моделювати ці складові та їхню взаємодію.

Компонентна діаграма: Відображає фізичні модулі IoT системи (сенсори, контролери, хмарні сервіси) та їхні інтерфейси.

Одним із основних аспектів моделювання IoT є зв'язок з хмарними обчислювальними сервісами для обробки великих обсягів даних. UML-діаграми допомагають моделювати цей зв'язок, зокрема за допомогою діаграм розгортання, які показують фізичне розташування компонентів системи.

Діаграма розгортання: Відображає фізичне розміщення компонентів IoT/IoE системи, таких як хмарні сервери та пристрої на рівні edge.

UML-діаграми дозволяють виконувати статичний аналіз архітектури IoT систем для перевірки правильності її проектування. Це включає перевірку цілісності взаємозв'язків між компонентами та відповідність архітектури вимогам.

Для оцінки продуктивності системи використовується моделювання поведінки компонентів у різних сценаріях. UML-діаграми послідовностей та діяльності дозволяють виявляти вузькі місця у продуктивності та оптимізувати взаємодію між компонентами.

Часовий аналіз: Оцінює затримки між обробкою даних сенсорами та передачею їх до хмари для подальшого аналізу.

Рисунок 6.3 – Верифікація системи на основі IoT за допомогою UML

Для забезпечення високої надійності IoT систем необхідно моделювати можливі помилки та їхній вплив на роботу системи. UML-діаграми станів допомагають моделювати різні стани системи (нормальна робота, помилка, відновлення) та аналізувати можливі сценарії виникнення збоїв.Приклад: Моделювання відмови сенсора або порушення зв'язку з хмарою.

Візуалізація складних систем: UML надає зрозумілу візуальну мову для опису взаємодії між компонентами IoT/IoE систем.

- Моделювання різних сценаріїв: UML-діаграми дозволяють моделювати різні сценарії роботи системи та перевіряти їхню ефективність до реальної реалізації.
- Аналіз продуктивності та надійності: Верифікація UML-моделей дозволяє виявляти потенційні проблеми продуктивності та підвищувати надійність IoT/IoE систем.

Моделювання та верифікація архітектури IoT та IoE систем за допомогою UML дозволяє проектувати надійні та ефективні системи. Використання діаграм класів, послідовностей, діяльності та компонентів допомагає точно візуалізувати структуру і поведінку IoT систем, забезпечуючи їх коректну роботу на всіх етапах розробки.

2. Моделювання та верифікація поведінки систем IoT та IoE на основі систем масового обслуговування та мереж Петрі

Системи Інтернету речей (IoT) та Інтернету всього (IoE) включають безліч взаємопов'язаних пристроїв і сервісів, які взаємодіють між собою в реальному часі. Для моделювання поведінки цих систем та забезпечення надійності їх роботи часто використовують системи масового обслуговування (СМО) та мережі Петрі. Це дає можливість детально аналізувати роботу систем, включаючи затримки, обробку запитів та взаємодію між компонентами.

СМО описують процеси обробки запитів у системах, де ресурси є обмеженими. В IoT та IoE системах СМО використовуються для моделювання черг та обробки подій.

Компоненти СМО: Джерела запитів (сенсори або пристрої), черги, обслуговуючі пристрої (сервери або контролери), та дисципліни обслуговування (FIFO, LIFO, приоритетне обслуговування тощо).

Параметри СМО: Інтенсивність надходження запитів, інтенсивність обслуговування, кількість черг і обслуговуючих каналів.

СМО використовуються для моделювання черг в системах IoT, наприклад:

Запити від сенсорів можуть надходити до контролера, який їх обробляє і передає на сервер для подальшої обробки. Використання черг дозволяє аналізувати затримки в обробці даних і оптимізувати розподіл ресурсів.

Рисунок 6.4 – Моделювання черг в IoT системах за допомогою СМО

Мережі Петрі — це графова структура, яка використовується для моделювання дискретних подій в системах. В IoT та IoE мережі Петрі використовуються для моделювання поведінки систем та синхронізації подій.

Компоненти мереж Петрі: Вузли (місця), переходи, дуги та маркери. Вузли представляють стани системи, переходи — події, а маркери — ресурс або стан виконання.

Типи мереж Петрі: Стандартні, стохастичні та розподілені мережі Петрі.

Мережі Петрі дозволяють моделювати взаємодію між компонентами IoT-систем, зокрема:

- Синхронізація подій: Наприклад, передача даних між сенсорами та контролерами в умовах синхронізації
- Моделювання конфліктів: Виявлення конфліктів при спільному використанні ресурсів між пристроями.

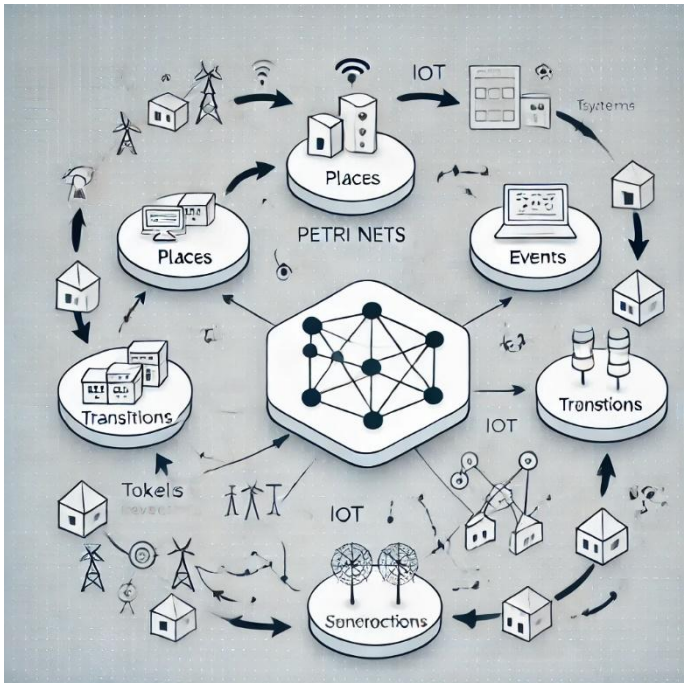

Рисунок 6.5 – Мережі Петрі для моделювання поведінки IoT-систем

Мережі Петрі дозволяють моделювати сценарії поведінки систем, наприклад:

- Збір та обробка даних: Сенсори надсилають дані контролеру, який обробляє їх та передає на сервер.
- Реакція на події: Виконавчі механізми реагують на дані, отримані від сенсорів, виконуючи дії на основі отриманих сигналів.

Мережі Петрі допомагають аналізувати час реакції системи, що є критичним для IoT/IoE у реальному часі. Можна моделювати затримки в передачі даних та обробці сигналів, а також аналізувати наслідки затримок для всієї системи.

Верифікація систем IoT та IoE за допомогою СМО та мереж Петрі включає статичний аналіз (перевірка структурних властивостей) і динамічний аналіз (перевірка продуктивності системи під час роботи).

Статичний аналіз: Включає виявлення можливих "вузьких місць" у системі та конфліктів при використанні ресурсів.

Динамічний аналіз: Оцінка продуктивності системи в умовах реального часу, моделювання робочих сценаріїв для виявлення потенційних проблем.

Моделі на основі СМО та мереж Петрі дозволяють тестувати стійкість системи до відмов, наприклад:

- Моделювання відмови сенсора та аналіз того, як система реагує на відмову.
- Виявлення конфліктів при обміні даними між пристроями та сервісами.

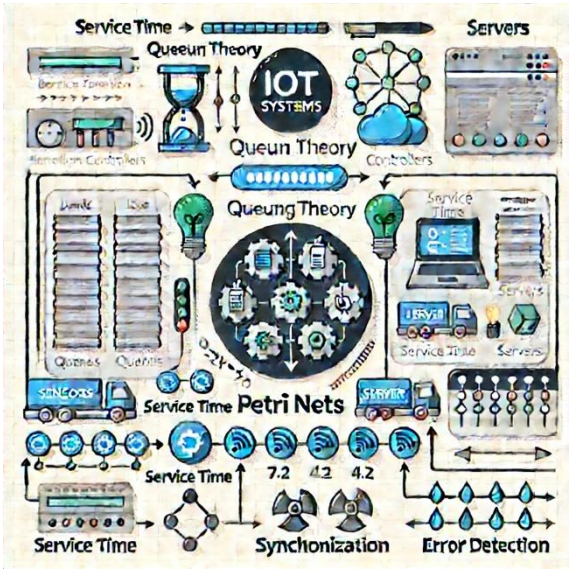

Рисунок 6.6 – Верифікація систем IoT за допомогою СМО та мереж Петрі

Моделювання складних процесів: СМО та мережі Петрі дозволяють моделювати складні взаємодії між компонентами IoT/IoE систем та забезпечувати точність аналізу.

Верифікація в реальному часі: Верифікація на основі цих методів дозволяє забезпечити надійність систем та підвищити їхню ефективність у реальних умовах експлуатації.

Аналіз продуктивності: Ці інструменти допомагають оптимізувати продуктивність системи, знижуючи затримки та підвищуючи ефективність використання ресурсів.

Моделювання та верифікація поведінки систем IoT та IoE на основі СМО та мереж Петрі є потужними інструментами для проектування складних систем. Використання цих методів дозволяє аналізувати роботу систем в умовах реального часу, оптимізувати їхню роботу та підвищити надійність.

ТЕМА 7. МОДЕЛЮВАННЯ ВЗАЄМОДІЇ В СИСТЕМАХ IOT

1. Взаємодія в системах IoT

Взаємодія в системах Інтернету речей (IoT) є ключовим елементом функціонування розумних пристроїв. Системи IoT складаються з різних компонентів, таких як сенсори, виконавчі механізми, контролери та хмарні сервіси, які постійно обмінюються даними для виконання своїх функцій. Ця взаємодія забезпечується через різні комунікаційні протоколи, мережі і архітектури.

Основні компоненти взаємодії в IoT

1) Сенсори

Сенсори збирають дані з фізичного середовища, такі як температура, вологість, рух, світло тощо. Вони є відправною точкою у процесі взаємодії, передаючи інформацію далі для обробки.

2) Виконавчі механізми

Виконавчі механізми відповідають за виконання дій на основі отриманих даних. Наприклад, можуть вмикати або вимикати пристрої, регулювати освітлення, керувати двигунами.

3) Контролери

Контролери здійснюють обробку даних від сенсорів і можуть приймати рішення щодо подальших дій. Вони можуть обробляти інформацію локально або передавати її на хмарні сервери для більш складного аналізу.

4) Хмарні сервіси

Хмарні платформи обробляють та зберігають великі обсяги даних, що надходять від пристроїв IoT. Вони забезпечують аналітику, машинне навчання і управління пристроями через Інтернет.

Рисунок 7.1 – Взаємодія компонентів в системах IoT

MQTT — це легкий протокол обміну повідомленнями, який використовують для передачі даних між пристроями IoT через Інтернет. MQTT ідеально підходить для систем з низьким енергоспоживанням і обмеженими ресурсами.

Переваги: Мінімальне споживання ресурсів, ефективність на малих швидкостях передачі даних.

Застосування: Використовується в розумних будинках, промислових мережах, віддаленому моніторингу.

CoAP — це протокол, призначений для роботи в мережах з обмеженими ресурсами, таких як IoT-пристрої. Він базується на протоколі HTTP, але оптимізований для малих пристроїв і низького енергоспоживання. Переваги: Ефективна передача даних в реальному часі, підтримка малих пристроїв. Застосування: Сенсорні мережі, системи віддаленого моніторингу.

Zigbee — це бездротовий протокол, який широко використовується для зв'язку між IoT-пристроями на короткі відстані. Він підтримує роботу з низьким енергоспоживанням і забезпечує стабільний зв'язок. Переваги: Низьке енергоспоживання, висока надійність Застосування: Системи розумного будинку, промислові мережі.

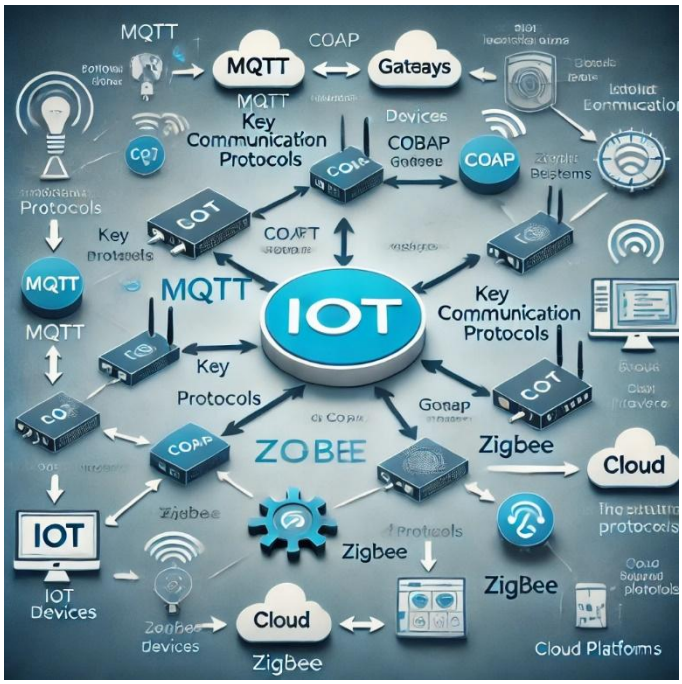

Рисунок 7.2 – Основні протоколи взаємодії для IoT

Взаємодія на рівні мережі

1) Локальні мережі (LAN)

Локальні мережі використовуються для взаємодії пристроїв IoT всередині будівлі або на певній території. Такі мережі можуть бути як дротовими, так і бездротовими.

2) Edge Computing

Edge Computing передбачає обробку даних ближче до джерела їхнього виникнення (наприклад, на контролерах або edge-пристроях), що знижує затримки і навантаження на хмару.

3) Хмарні платформи

Хмари є центральними вузлами для збору та обробки даних у глобальних IoT системах. Вони забезпечують зберігання, аналіз і прийняття рішень на основі даних, що надходять від сенсорів і контролерів.

Інтернет всього (IoE) є наступним кроком після IoT, де до мереж підключають не тільки фізичні пристрої, але й люди, дані та процеси. Взаємодія в системах IoE ще складніша і вимагає інтеграції різних типів даних та технологій. Люди є важливими учасниками взаємодії

в системах IoE. Вони можуть використовувати мобільні додатки для управління IoT пристроями, отримувати повідомлення про стан системи та приймати рішення на основі зібраних даних. IoE включає автоматизацію та управління процесами на основі зібраних даних, що дозволяє оптимізувати роботу підприємств, міст і будинків.

Взаємодія в системах IoT забезпечується через багаторівневу архітектуру, що включає сенсори, контролери, хмарні сервіси та протоколи зв'язку. Використання таких протоколів, як MQTT, CoAP та Zigbee, дозволяє ефективно управляти мережею пристроїв і забезпечувати їхню надійну взаємодію в реальному часі.

2. Мова моделювання потоків взаємодії

Мова моделювання потоків взаємодії (Interaction Flow Modeling Language, IFML) — це стандарт для опису поведінки та взаємодії користувача з програмними системами, зокрема веб-додатками, мобільними додатками та іншими інтерактивними системами. IFML дозволяє візуалізувати потоки взаємодії користувача з інтерфейсом і передавати інформацію між різними компонентами системи.

IFML — це стандартна мова для моделювання потоків взаємодії, яка була розроблена для опису поведінки користувачів і взаємодії з інтерфейсами додатків. Вона підтримує моделювання як статичних, так і динамічних елементів системи, зокрема:

- Інтерфейси користувача: Описує, як виглядатиме інтерфейс і яким чином він реагуватиме на дії користувача.
- Потоки взаємодії: Описує, як дані переміщуються між інтерфейсами та сервісами.

Події та дії: Описує, що відбувається при взаємодії користувача з інтерфейсом (натискання кнопки, введення даних тощо).

Основні компоненти IFML включають:

- Видимі компоненти: Елементи інтерфейсу, такі як кнопки, поля введення, списки.
- Потоки даних: Потоки, що передають інформацію між компонентами або між системою і користувачем.
- Події: Дії, які ініціюють зміни в системі, наприклад, натискання кнопки або надсилання форми.

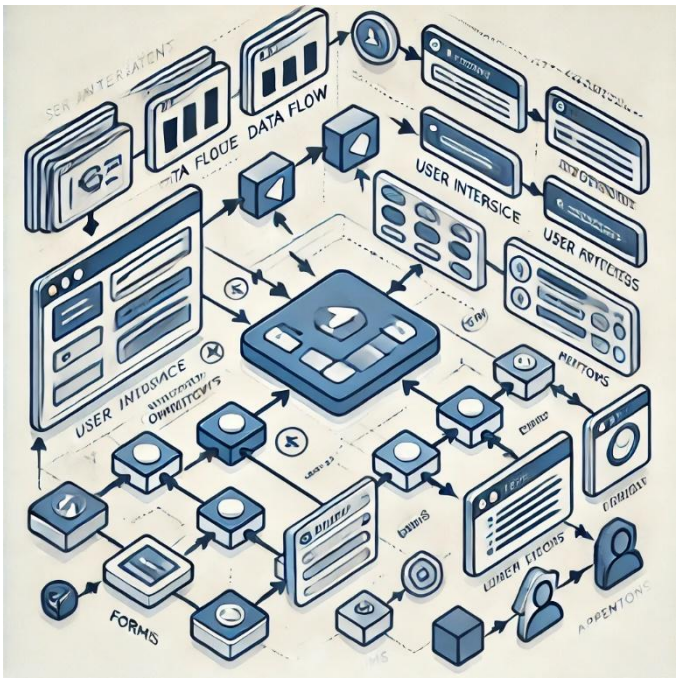

Рисунок 18.3 – Приклад IFML діаграми для веб-додатку

IFML дозволяє візуалізувати всі елементи інтерфейсу користувача, такі як кнопки, форми, меню та інші елементи взаємодії. Кожен елемент представлений у вигляді компонента, який взаємодіє з іншими компонентами або ініціює певні дії.

Події інтерфейсу: Події, що активуються користувачем, включають натискання на кнопку, введення даних або вибір елементів зі списку.

Потоки даних: IFML моделює потоки даних між компонентами інтерфейсу та бекенд-системою для обробки запитів.

Моделювання навігаційних потоків дозволяє описати, як користувач переміщується між різними екранами або сторінками системи, а також як система відповідає на дії користувача.

Навігаційні переходи: Показують зв'язок між екранами або сторінками, наприклад, після входу в систему користувач переміщується на головну сторінку.

Логіка навігації: Моделює умови та правила, за яких відбувається перехід між різними елементами інтерфейсу.

Рисунок 7.4 – Навігаційна діаграма для мобільного додатку

IFML також дозволяє моделювати інтерактивну логіку системи, таку як обробка подій, запити до бази даних, відправка повідомлень або управління контентом.

Обробка подій: Моделювання, яке описує, як система реагує на певні події від користувача.

Дії на стороні сервера: Включають запити до бази даних або відправку даних на сервер для подальшої обробки.

Верифікація та тестування за допомогою IFML

1) Статичне тестування

IFML дозволяє виконувати статичну перевірку логіки взаємодії для перевірки цілісності моделі: чи правильно побудовані переходи, чи коректно визначені події та чи узгоджені елементи інтерфейсу.

2) Динамічне тестування

Моделі IFML можуть бути використані для створення сценаріїв тестування взаємодії

користувача з системою. Це дозволяє оцінити, наскільки ефективно побудована логіка взаємодії та чи виконує система всі дії згідно з очікуваними результатами.

Переваги використання IFML:

- Простота моделювання: IFML дозволяє легко візуалізувати складні взаємодії користувача з системою, забезпечуючи зрозумілий та інтуїтивний інтерфейс для моделювання.
- Сумісність з іншими стандартами: IFML може бути інтегрований з іншими мовами моделювання, такими як UML, для створення повної моделі системи.
- Автоматизація процесу розробки: Моделі IFML можуть використовуватися для автоматичного генерування кодів або шаблонів для веб- та мобільних додатків.

Мова моделювання потоків взаємодії (IFML) є потужним інструментом для розробки та моделювання взаємодії користувачів з інтерфейсами. Вона дозволяє візуалізувати і вдосконалювати логіку додатків ще на етапі проектування, що підвищує ефективність розробки та якість кінцевого продукту.

ТЕМА 8. МОДЕЛІ ДЛЯ ПРИСТРОЇВ ТА ТЕХНОЛОГІЙ НА ОСНОВІ ІОТ ДЛЯ ОБРОБКИ ТА ПЕРЕДАЧІ ДАНИХ

1. Пристрої на основі IoT: моделі та протоколи мережевого зв'язку

Пристрої Інтернету речей (IoT) є основою для створення смарт-систем, таких як розумні будинки, промислові рішення та медичні IoT. Взаємодія цих пристроїв забезпечується через різні мережеві протоколи зв'язку, які підходять для передачі даних в різних умовах, таких як локальні мережі або віддалене керування через Інтернет.

Сенсори є базовими елементами IoT-систем, які збирають дані з фізичного середовища і передають їх на контролери або в хмару для подальшої обробки. Приклади моделей сенсорів:

- Температурні сенсори: DHT11, DS18B20.
- Датчики руху: HC-SR501.
- Газові сенсори: MQ-2 (газ метан), MQ-135 (повітряні забруднення).

Контролери здійснюють управління IoT-пристроями, обробку даних і передачу їх на сервери або в хмару:

- Arduino: Популярна платформа для прототипування, яка підтримує багато сенсорів і модулів зв'язку.
- ESP8266/ESP32: Мікроконтролери з вбудованим Wi-Fi модулем, які широко використовуються в IoT-проектах для підключення до Інтернету.
- Raspberry Pi: Компактний комп'ютер, який може виконувати складні обчислювальні завдання та підключатися до мережі через Wi-Fi або Ethernet.

Виконавчі механізми перетворюють команди системи в дії. Вони можуть бути підключені до контролерів і реагувати на дані, зібрані сенсорами:

- Реле: Використовується для включення/виключення електричних приладів.
- Сервоприводи: Для руху механічних частин, наприклад, у роботах або промислових системах.

MQTT — це легкий протокол обміну повідомленнями, розроблений для IoT-пристроїв з низькою пропускною здатністю мережі. Протокол працює за моделлю "видавець-підписник" і забезпечує передачу повідомлень між пристроями через брокера.

Переваги: Низьке споживання енергії, надійність, мінімальні затримки.

Застосування: Розумні будинки, промислові IoT-системи, віддалений моніторинг.

CoAP — це протокол передачі даних для пристроїв з обмеженими ресурсами. Він побудований на архітектурі HTTP, але значно легший і ефективніший для малопотужних пристроїв.

Переваги: Оптимізований для низької пропускної здатності, підтримує малопотужні пристрої.

Застосування: Сенсорні мережі, смарт-міста, екологічний моніторинг.

LoRaWAN — це протокол для низькоенергетичних широкомасштабних мереж (Low Power Wide Area Networks, LPWAN), який підтримує передачу даних на великі відстані з мінімальними енергозатратами.

Переваги: Велика дальність передачі даних (до 15 км), низьке енергоспоживання.

Застосування: Сільське господарство, промислові IoT, розумні міста.

Zigbee — це бездротовий протокол зв'язку з низьким енергоспоживанням, призначений для коротких дистанцій. Використовується для побудови mesh-мереж, де пристрої можуть передавати дані один через одного.

Переваги: Низьке енергоспоживання, підтримка mesh-топології.

Застосування: Розумні будинки, автоматизація освітлення та опалення, системи безпеки.

Bluetooth LE — це варіант класичного Bluetooth протоколу, що використовує менше енергії, але підтримує передачу даних на короткі відстані.

Переваги: Низьке енергоспоживання, підтримка мобільних пристроїв.

Застосування: Носимі пристрої, системи моніторингу здоров'я, розумні годинники.

У цій моделі IoT-пристрої виступають як клієнти, які збирають і передають дані на сервер для обробки. Сервер відповідає за зберігання та аналіз даних і може відправляти команди назад на пристрої.

Ця модель передбачає, що пристрої (видавці) надсилають дані брокеру, який передає їх підписникам. Це зменшує навантаження на мережу і дозволяє ефективно розподіляти дані між великою кількістю пристроїв.

Модель edge computing передбачає, що частина обробки даних відбувається на периферійних пристроях (на місці збору даних), зменшуючи навантаження на центральні сервери та мережу.

Протоколи мережевого зв'язку відіграють ключову роль у забезпеченні безперебійної роботи IoT-пристроїв. Вибір правильного протоколу залежить від вимог проекту, таких як енергоспоживання, відстань передачі даних та складність інфраструктури. Різні моделі зв'язку, такі як клієнт-сервер і видавець-підписник, дозволяють адаптувати систему до специфічних умов роботи.

2. Технології обробки даних в системах на основі IoT

Системи Інтернету речей (IoT) генерують величезні обсяги даних з різних сенсорів і пристроїв. Для ефективного використання цих даних необхідно застосовувати різноманітні технології обробки, які дозволяють аналізувати інформацію в режимі реального часу, зберігати дані та приймати рішення на їх основі.

У IoT системах основними джерелами даних є сенсори та пристрої. Вони збирають інформацію про стан фізичного середовища, таку як температура, вологість, рівень освітлення, місцезнаходження та багато іншого.

Обробка даних в IoT системах зазвичай складається з таких етапів:

1) Збір даних: Сенсори фіксують показники та передають їх на контролери або хмару.
2) Фільтрація та агрегація: Для зменшення обсягу даних виконується фільтрація, видалення дублікатів і агрегація інформації.
3) Аналіз даних: Використання аналітичних алгоритмів та штучного інтелекту для визначення трендів, аномалій та прийняття рішень.
4) Зберігання даних: Оброблені дані зберігаються для подальшого аналізу або історичних записів.

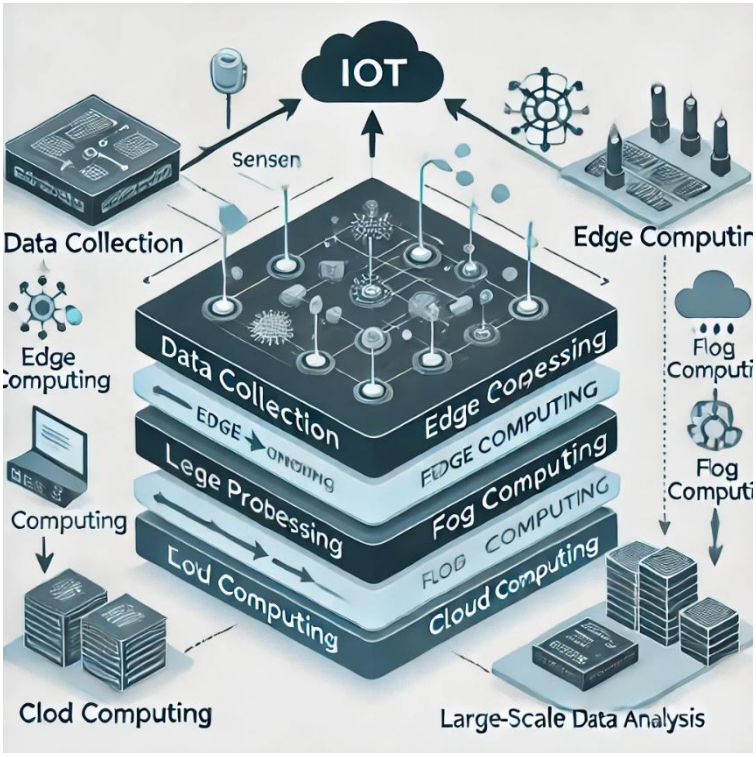

Рисунок 8.1 – Архітектура обробки даних в IoT системах

Edge Computing — це технологія обробки даних на рівні пристроїв або поблизу місця збору даних, а не у віддалених хмарних сервісах. Це дозволяє зменшити затримки в обробці та знизити навантаження на мережу.

Переваги: Низькі затримки, зменшення обсягів даних, що передаються до хмари.

Застосування: Системи реального часу, промислові IoT, системи безпеки.

Fog Computing — це концепція, яка поєднує переваги як edge computing, так і хмарних обчислень. Дані обробляються частково на локальних пристроях, а частково на серверах або в хмарі.

Переваги: Збалансоване використання обчислювальних ресурсів, підвищена ефективність мережі.

Застосування: Смарт-міста, транспортні системи, сільське господарство.

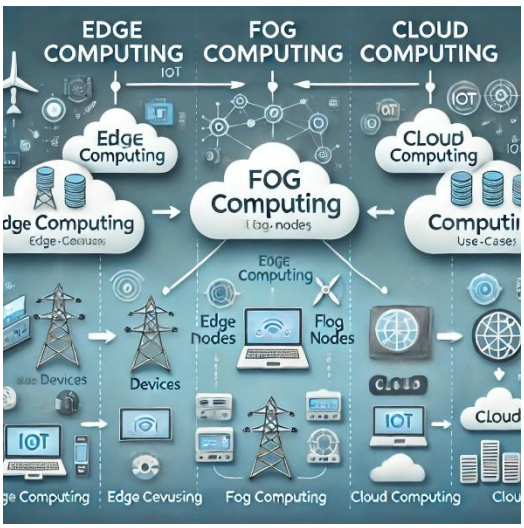

Рисунок 8.2 – Порівняння Edge та Fog Computing

Хмарні обчислення дозволяють обробляти та зберігати великі обсяги даних, що

надходять від IoT-пристроїв. Хмари забезпечують масштабованість і доступ до потужних обчислювальних ресурсів.

Переваги: Масштабованість, можливість обробки великих обсягів даних.

Застосування: Аналітика великих даних, моніторинг у реальному часі, системи зі складними обчисленнями.

Streaming Data Processing — це технологія, що дозволяє аналізувати дані в режимі реального часу, коли вони надходять від сенсорів. Це особливо важливо для систем, де потрібні швидкі реакції на події.

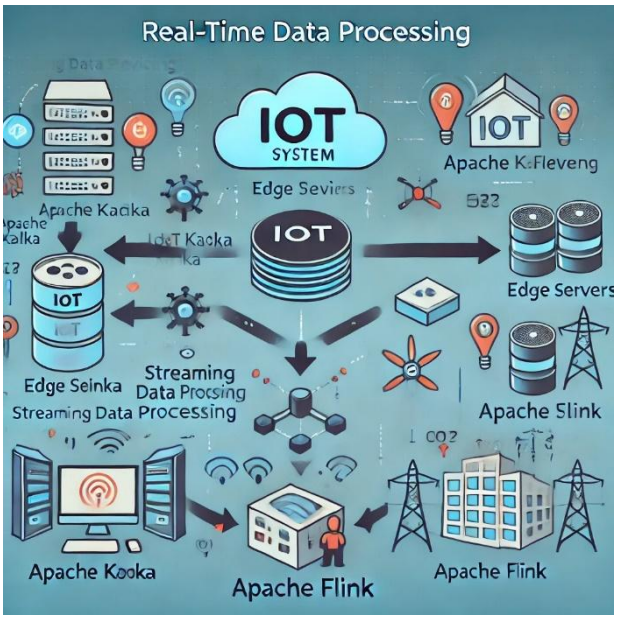

Рисунок 8.3 – Технології обробки даних у реальному часі

Алгоритми аналізу потоків даних використовуються для обробки інформації, що надходить від сенсорів у режимі реального часу. Це дозволяє виявляти аномалії або тренди та реагувати на них.

Приклад: Виявлення несправностей у промислових системах або аналіз поведінки користувачів.

Штучний інтелект (AI) та машинне навчання (ML) допомагають аналізувати дані з IoT-пристроїв для побудови прогнозів і моделей на основі зібраної інформації.

Застосування: Прогнозування поломок обладнання, персоналізовані рекомендації, оптимізація процесів.

IoT-системи генерують великі обсяги даних, що може створювати проблеми зі зберіганням і обробкою. Важливо вибирати відповідні технології та інструменти для обробки даних, щоб забезпечити ефективну роботу системи.

Передача та обробка даних в IoT-системах потребують високого рівня захисту. Шифрування даних, автентифікація пристроїв та контроль доступу є ключовими аспектами забезпечення безпеки.

Технології обробки даних в IoT-системах включають edge та fog computing, хмарні обчислення, а також обробку даних у реальному часі. Вибір технології залежить від вимог до продуктивності, затримок та масштабованості. Крім того, використання алгоритмів машинного навчання дозволяє оптимізувати аналіз даних і покращити ефективність системи.

ТЕМА 9. ПРОТОТИПУВАННЯ ТА ШВИДКИЙ РОЗВИТОК IOT- СИСТЕМИ

1. Принципи прототипування та швидкої розробки

Прототипування та швидка розробка є важливими етапами створення IoT-систем. Ці підходи дозволяють тестувати концепції, експериментувати з різними компонентами та вирішувати технічні проблеми ще до розгортання повномасштабної системи. Прототипування є важливим для мінімізації ризиків і прискорення процесу виходу на ринок.

Прототипування відбувається в ітераційних циклах, кожен з яких передбачає створення базової версії системи з подальшим тестуванням і вдосконаленням:

- Швидке створення MVP (Minimum Viable Product): Спрощена версія продукту, що містить тільки ключові функції.
- Тестування та отримання зворотного зв'язку: Виявлення недоліків та проблем у функціонуванні прототипу.
- Вдосконалення та розширення функцій: Поступове додавання нових можливостей та функціоналу на основі результатів тестування.

IoT-системи складаються з багатьох компонентів, таких як сенсори, виконавчі механізми, контролери та мережеві модулі. Модульний підхід дозволяє створювати окремі прототипи для кожного з компонентів, що полегшує тестування та інтеграцію.

Переваги модульності: Легкість у заміні або вдосконаленні окремих компонентів без необхідності зміни всієї системи.

Прототип повинен бути достатньо гнучким для внесення змін на будь-якому етапі розробки. Це стосується як апаратної частини (плати, сенсори), так і програмної (алгоритми, API).

Платформи, такі як Arduino, ESP32, та Raspberry Pi, спрощують розробку IoT-прототипів, оскільки мають вбудовану підтримку сенсорів і мережевих модулів, а також великий набір готових бібліотек.

Arduino: Використовується для створення простих прототипів зі стандартними сенсорами та виконавчими механізмами.

ESP32: Забезпечує Wi-Fi та Bluetooth підключення, що дозволяє створювати бездротові системи.

Raspberry Pi: Потужна платформа для розробки більш складних IoT-рішень із вбудованими можливостями для обробки даних.

Хмарні сервіси, такі як AWS IoT, Azure IoT та Google Cloud IoT, дозволяють швидко інтегрувати прототип IoT-системи з хмарними сховищами, аналітичними інструментами та обчислювальними ресурсами

Переваги хмарних платформ: Масштабованість, віддалене керування пристроями, аналітика даних у реальному часі.

Для швидкої розробки прототипів важливо мати інструменти, які дозволяють створювати інтерфейси для користувачів. Такі платформи, як Blynk та Node-RED, дозволяють розробляти прості мобільні додатки для керування IoT-пристроями та моніторингу даних.

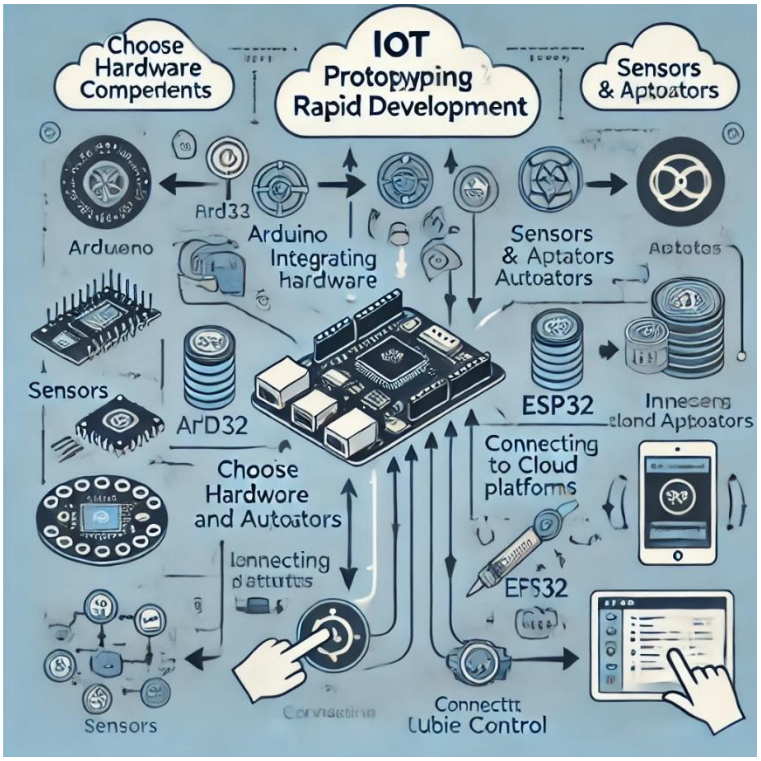

Рисунок 9.1 – Інструменти для швидкого прототипування IoT

Швидка розробка вимагає гнучкості в дизайні системи, яка дозволяє легко змінювати компоненти або архітектуру залежно від вимог. Важливо використовувати стандартизовані компоненти та модулі, які легко інтегруються між собою.

Принцип "fail-fast"

Цей принцип полягає в тому, щоб якнайшвидше виявляти проблеми та помилки на ранніх етапах розробки, виправляючи їх до того, як система буде масштабована. Це дозволяє зекономити час і ресурси на пізніших етапах.

Автоматизовані інструменти для тестування можуть суттєво прискорити процес розробки IoT-систем. Вони дозволяють перевіряти працездатність компонентів системи на різних етапах прототипування та швидко виявляти проблеми.

Прототипування та швидка розробка є важливими етапами в створенні IoT-систем. Використання готових платформ, хмарних сервісів та модульного підходу дозволяє суттєво прискорити процес розробки та тестування, забезпечуючи гнучкість та масштабованість для майбутніх рішень.

2. Приклади швидкої розробки IoT-систем

1) Розумний будинок

Швидка розробка IoT-рішень для розумного будинку передбачає інтеграцію сенсорів та контролерів для автоматизації різних функцій в домашньому середовищі.

Етапи швидкої розробки:

- Вибір контролера: Використання платформи ESP32 для забезпечення бездротового з'єднання через Wi-Fi або Bluetooth.
- Інтеграція сенсорів: Сенсори для моніторингу температури (DS18B20), вологості (DHT11), та руху (PIR).
- Мобільний додаток: Створення мобільного додатку на базі Blynk, що дозволяє користувачеві контролювати освітлення, температуру та інші параметри в реальному часі.

Переваги:

Швидке налаштування системи з використанням готових модулів.

Гнучкість у додаванні нових пристроїв до системи.

2) Моніторинг навколишнього середовища

Прототипування системи моніторингу навколишнього середовища дозволяє створити мережу сенсорів, які збирають дані про якість повітря, температуру, вологість та рівень шуму.

Етапи швидкої розробки:

- Вибір платформи: Використання Arduino для простоти інтеграції сенсорів.
- Сенсори: Газові сенсори (MQ-2), температурні сенсори (DHT22), датчики шуму.
- Обробка даних: Передача даних на хмарну платформу ThingSpeak для візуалізації та аналізу в реальному часі.

Переваги:

Швидка інтеграція сенсорів.

Використання хмарних сервісів для аналізу даних без необхідності складної локальної інфраструктури.

3) Розумне управління вуличним освітленням

Це приклад IoT-системи, яка автоматизує керування вуличним освітленням на основі датчиків руху та освітленості для економії енергії.

Етапи швидкої розробки:

- Контролер: Використання Raspberry Pi для централізованого контролю освітлення на базі мережі сенсорів.
- Сенсори: Датчики руху (PIR) та освітленості.
- Автоматизація: Налаштування системи для автоматичного ввімкнення/вимкнення світла в залежності від зовнішніх умов (рух, рівень освітлення).

Переваги:

Швидка інтеграція з існуючими системами.

Зниження енергоспоживання за рахунок оптимізованої роботи освітлення.

4) Промисловий IoT для моніторингу стану обладнання

Система, яка забезпечує моніторинг промислового обладнання в режимі реального часу, щоб запобігти поломкам і підвищити ефективність виробництва.

Етапи швидкої розробки:

- Вибір контролера: Використання ESP32 для передачі даних з сенсорів до хмарної платформи.
- Сенсори: Вібраційні сенсори для виявлення проблем у роботі обладнання.
- Обробка даних: Передача даних на хмарну платформу AWS IoT для прогнозного аналізу поломок.

Переваги:

Швидке впровадження з використанням готових сенсорів.

Підвищення надійності системи за рахунок автоматичного моніторингу.

Швидка розробка IoT-систем на прикладі розумного будинку, моніторингу навколишнього середовища, управління освітленням та промислового моніторингу показує, що використання готових платформ, сенсорів і хмарних сервісів значно прискорює процес створення функціональних прототипів, знижує витрати і забезпечує гнучкість.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «Internet of Things Online Course | MIT Sloan Executive Education», Executive.mit.edu. URL: https://executive.mit.edu/openenrollment/program/internet-of-thingsbusiness implications-and-opportunities/#.XTysbuZR3IV (дата звернення: 20.05.2026).

2. Define IoT – IEEE Internet of Things». URL: https://iot.ieee.org/definition.html. (дата звернення: 20.05.2026).

3. IoT University, 'IoT UI Development with ThingWorx: Course Summary, Course Milestones'. (n.d.). Available at: https://www.iotu.com/enrollment/student/iot-ui-development-witthingworx (дата звернення: 10.06.2026).

4. Delivering on the IoT customer experience. Business white paper. Hewlett Packard Enterprise. Available at: http://h20195.www2.hpe.com /v2/GetDocument.aspx?docname =4AA6-5128ENW (дата звернення: 10.06.2024).

5. Cloud Customer Architecture for IoT, «Cloud Standards Customer Council». Available: https://www.omg.org (дата звернення: 10.06.2026).

6. B. Schmarzo, «The Internet of Things and Analytics at the Edge» [Online] https://infocus.emc.com/william_schmarzo/internet-of-things-analyticsedge/ (дата звернення: 10.03.2024).

7. The Internet of Things.. Internet of Things and the Prelude to Artificial Intelligence. [online] Available at: http://www.infiniteinformationtechnology.com/the-internet-of-thingsprelude- to-artificial-intelligence (дата звернення: 10.05.2024).

8. Developer documentation archive. – electronic source: https://developer.apple.com/library/archive/ (дата звернення: 15.06.2026).

9. M. Grimheden, "Mechatronics Education at KTH (and Embedded Systems)",https://www.kth.se/polopoly_fs/1.518408.1550157760! /CPSED2014_Berkeley_MartinGrimheden.pdf.http://ela.kpi.ua/bitstream/123456 89/15592/1/GM_Sviatny_Brovkina_N1_20 16.pdf (дата звернення: 15.04.2026).

10. I. Horváth and B. Gerritsen, «Cyber-Physical Systems: Concepts, Technologies and Implementation Principles», 2019. [Online]. Available: https://www.academia.edu/14501665/Cyber-Physical_Systems_Concepts _technologies_and_implementation_principles (дата звернення: 20.05.2024).

11. «About the UML Profile for MARTE Specification Version 1.0», omg.org, 2009. [Online]. Available: https://www.omg.org/spec/ MARTE/1.0 (дата звернення: 20.05.2024).

12. «IoT Simulator, Simulate IoT Devices - Bevywise Networks», Bevywise Networks.. Available: https://www.bevywise.com/iot-simulator/ (дата звернення: 10.05.2026)

Додаткові:

1. Kostiuchko S., Kyryliuk L., Chernyashchuk N., Bortnyk K., Hrunjuk S. Wireless access point with multilayer data protection algorithm. // Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво» – Луцьк: Видавництво ЛНТУ. – Вип. 42. – 2021. – С. 147-151.

2. Гринюк С.В., Поліщук М.М., Гринюк М.В., Шульгат В.В., Терешкович В.І. Інтелектуальна система керування освітленням на базі Arduino Uno // Науковий журнал «Комп'ютерно-інтегровані технології: освіта, наука, виробництво». Луцьк: Видавництво ЛНТУ. Вип. 53. 2023. С. 98-103. https://doi.org/10.36910/6775-2524-0560-2023-53-15.

3. Polishchuk M., Grinyuk S., Kostiuchko S., Tkachuk A., & Savaryn, P. (2023). TESLA SWITCH OF 4 BATTERIES BASED ON THE ARDUINO UNO BOARD. Informatyka,

Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 13(3), 111–116. https://doi.org/10.35784/iapgos.4051

І 75 **ІоТ-технології для кіберфізичних систем**: Конспект лекцій для здобувачів другого (магістерського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузь знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання / уклад. С.В. Гринюк Луцьк: ЛНТУ, 2026. 48 с.

Комп'ютерний набір: С.В. Гринюк
Редактор: С.М. Костючко

Підп. до друку _____ 2026 р.
Формат 60х84/16. Папір офс. Гарнітура Таймс.
Ум. друк. арк.____Тираж ___ прим. Зам._____

Відділ іміджу та промоції
Луцького національного технічного університету
43018 м. Луцьк, вул. Львівська, 75