COMPUTER DESIGN SYSTEMS. THEORY AND PRACTICE

Vol. 7, No. 1, 2025

Костянтин Колесник[1], Іван Коземчук[2]

[1] Кафедра систем автоматизованого проектування, Національний університет "Львівська політехніка", вул. С. Бандери 12, Львів, Україна, E-mail: kostyantyn.k.kolesnyk@.lpnu.ua, ORCID0000-0001-9396-595X

[2] Кафедра систем автоматизованого проектування, Національний університет "Львівська політехніка", вул. С. Бандери 12, Львів, Україна, E-mail: ivan.v.kozemchuk@lpnu.ua

АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ПРОЕКТУВАННЯ ВБУДОВАНИХ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ

Отримано: Березень 03, 2025 / Переглянуто: Березень 14, 2025 / Прийнято: Березень 20, 2025

https://doi.org/

Анотація. У статті проведено аналіз методів та засобів проектування вбудованих систем Інтернету речей (IoT). Розглянуто основні етапи розробки IoT-систем, виконано порівняння основних підходів до проектування і визначено їхні переваги та обмеження. Проведено аналіз апаратних платформ, їхніх характеристик, продуктивності, енергоефективності та застосування в різних сферах. Розглянуто програмні інструменти та їхню ефективність у розробці IoT-рішень. Особливу увагу приділено архітектурним рішенням, що впливають на продуктивність, енергоефективність та масштабованість IoT-рішень. Розглянуто виклики, пов'язані з безпекою, стандартизацією та енергоефективністю вбудованих систем. Отримані результати можуть бути корисними для дослідників, інженерів та розробників вбудованих IoT-рішень, а також для оптимізації процесів проектування та вибору найбільш ефективних технологій у цій сфері.

Ключові слова: Інтернет речей (IoT), вбудовані системи, методи проектування, мікроконтролери, мікропроцесори, архітектура систем, розподілені системи, апаратна платформа, Edge Computing.

Вступ

В останні роки Інтернет речей став однією з ключових технологій, яка змінила взаємодію людей, пристроїв та інформаційних систем. Завдяки впровадженню IoT з'явилася можливість автоматизувати процеси, зменшити використання ресурсів і підвищити ефективність у промисловості, медицині чи сільському господарстві. Розробка таких систем вимагає комплексного підходу. Він включає вибір апаратної платформи, створення ефективного програмного забезпечення, а також використання спеціалізованих інструментів для проектування, тестування та оптимізації.

Постановка проблеми

Станом на сьогодні відсутній універсальний підхід до проектування вбудованих IoT-систем. Це ускладнює процес їх розробки, тестування та впровадження. Різні методи проектування, засоби розробки, комунікаційні технології та вимоги безпеки - це все створює додаткові виклики для інженерів та дослідників. Питання які ми обрали для порівняння і аналізу, це питання оптимального вибору підходу до розробки апаратного та програмного забезпечення. Вони все ще потребують системного аналізу та порівняльного дослідження.

В даній статті ми поставили за мету дослідити, порівняти та проаналізувати сучасні методи та засоби проектування вбудованих систем Інтернету речей. Виявити їхні переваги, недоліки та визначити найефективніші підходи для розробки IoT-пристроїв в залежності від конкретних застосувань. Основною проблемою є вибір оптимального методу проектування та вибір серед обмежених апаратних ресурсів IoT-пристроїв.

Огляд сучасних джерел інформації за тематикою публікації

Літературний огляд виконано на основі джерел, які описують методи і засоби розробки та впровадження вбудованих систем Інтернету речей. Різні роботи описують різні аспекти і проблеми в залежності від сфери їх використання. В них часто порівнюються протоколи зв`язку, рішення з безпеки і енергоефективності.

Одним із оглядів у цій сфері є дослідження проміжного програмного забезпечення для систем Інтернету речей [1]. У ньому автори аналізують ключові технології та протоколи, що використовуються у вбудованих IoT-системах. Вони детально порівнюють стандарти зв'язку, обробку даних та мережеву архітектуру. Хоча це дослідження фокусується на технології на рівень вище ніж ми, воно підкреслює важливість ефективної інтеграції апаратних та програмних компонентів. Також при проектуванні приладів в системі IoT важливо враховувати проміжне ПЗ для реалізації оптимального рішення.

У роботі [2] автори досліджують основні технічні виклики питань безпеки для вбудованих систем Інтернету речей. Ця робота розглядає архітектуру таких систем з точки зору виявлення загроз. Для нас корисний цей розбір, оскільки він дозволяє проектувати архітектуру системи з урахуванням викликів та засобів наданих авторами.

Автори праці [3,4] розробили ефективну і доступну систему автоматизації розумного дому. Вони описують методології, якими вони керувались при проектуванні, використанні технології та технічне рішення. Їх реалізація і впровадження підкреслюють наявні проблеми при розробці таких систем, а саме питання сумісності такої системи, планування її архітектури і її імплементації корегуючись відгуками користувачів.

Нами також були проаналізовані роботи з моделями проектування вбудованих систем Інтернету речей. Аналіз показав що каскадна модель [5, 6] та V-модель [7, 8] все ще активно використовуються для реалізації проектів з чіткими вимогами і потребою в сертифікації, наприклад в промислових IoT-рішеннях[6]. Agile[9 - 14] та Rapid Prototyping[15 - 20] залишаються основними і найпопулярнішими методами. Наприклад, в роботі [13] автори детально розглядають гнучкі методи проектування програмного забезпечення в контексті IoT, транспортних систем та їх виклики в плані безпеки. Автор [12] наводить цифри в їх використанні, Scrum – 42,11%, Rapid Prototyping – 39,47%. Їх імплементація в проектування вбудованих систем постійно вдосконалюється і вони використовуються в промислових рішеннях [14]. З досліджених робіт можна сказати, що метод Model-Based Design [20 - 25] теж активно використовується для IoT. Ба більше, його гнучкість та зручність теж дають можливість розробляти IoT проекти зі строгими вимогами, наприклад для промисловості [25].

Автори [26] детально розглядають технології, протоколи і стандарти для побудови інтелектуальних вбудованих систем на основі інтернету речей. Їх архітектури та застосування служать хорошим підґрунтям для опису засобів для вирішення поставлених проблем.

В роботі [27] детально порівнюються плати розробника на різних платформах для розробки прототипів вбудованих систем Інтернету речей. Автор порівнює технічні характеристики, вартість, надійність і надає поради для створення прототипів. Також потрібно віжзначити обговорення взаємодії платформ з хмарними сервісами. Стаття є корисною для вибору саме апаратної платформи.

Виклад основного матеріалу

Інтернет речей (IoT) — це мережа фізичних пристроїв, транспортних засобів, побутових приладів та інших об'єктів, оснащених електронікою, програмним забезпеченням, датчиками, виконавчими механізмами та засобами зв'язку, що дає змогу цим об'єктам підключатися та обмінюватися даними[28].

Розробка вбудованої системи відрізняється від розробки звичайного програмного забезпечення. Ціна помилки набагато вища. Наприклад, якщо при аналізі вимог було обрано мікроконтролер із недостатньою кількістю пам`яті чи не передбачено додаткову зовнішню пам`ять,

під час розробки необхідно максимально оптимізувати код чи відмовлятись від функціональності, яка була запланована. Інколи, коли ця функціональність є критичною, потрібно відкликати партію вбудованих пристроїв і чим пізніше ця проблема виявлена – тим більша шкода компанії яка розробляє цей продукт [29]. Розглянемо етапи розробки вбудованої системи:

Визначення вимог до системи та планування. На цьому етапі команда збирає та документує вимоги до програмного і апаратного забезпечення. З цих вимог визначаються функціональні та нефункціональні вимоги. До цих вимог входить продуктивність, енергоспоживання, обсяг пам'яті, вимоги до безпеки та реального часу. Проводиться аналіз середовища, в якому система буде працювати, та визначаються ключові задачі, які вона повинна виконувати. З отриманих даних формується початкова документація. Зібрані вимоги формалізуються в детальний специфікаційний документ. На його основі формується план проекту. Це документ у якому вказано обсяг, часові рамки, ресурси та бюджет для розробки. Цей етап є критично важливим, оскільки правильно сформульовані вимоги гарантують, що система буде розроблена відповідно до очікувань користувача.

Вибір апаратної платформи та архітектури. Даний етап починається з аналізу вимог до системи, який був розроблений на попередньому етапі. На його основі інженери визначають, які мікроконтролери, процесори та сенсори підходять для проекту. Основними критеріями при їх виборі є енергоефективність, продуктивність, вартість і сумісність між собою. Також при розробці комплексної системи враховується комунікація з іншими пристроями в мережі та підтримка комунікаційних протоколів для цього спілкування (Wi-Fi, Bluetooth, CAN тощо). Після визначення архітектури пристроїв і системи, команди розробників проектують окремі програмні чи апаратні модулі. Інженери документують структури даних, алгоритми та інтерфейси які будуть використовуватися під час розробки.

Розробка програмного забезпечення. Під час створення програмного забезпечення розробники, на основі сформованої документації, пишуть код для виконання основних завдань системи. Вони реалізують алгоритми, налаштовують управління апаратними компонентами та створюють механізми для збору, обробки та передачі даних. Для взаємодії з сенсорами та іншими пристроями розробляють спеціальні драйвери. Одним із важливих завдань є адаптація алгоритмів до обмежених ресурсів пристрою і зменшення споживання енергії на мобільних вбудованих системах.

Під час цього етапу, розробники програмного забезпечення зазвичай працюють на платах розробників та збирають прототип готового рішення. Це дозволяє максимально близько наблизитись до фінального продукту в плані апаратної частини. Таким чином, команда, яка розробляє ПЗ може швидко знайти недоліки та правки і надати їх команді по розробці апаратної частини.

Тестування та налагодження. На цьому етапі перевіряють, чи всі компоненти працюють правильно. Тестують як програмне забезпечення, так і апаратну частину. Для перевірки часто використовують емулятори та тестові середовища. Це потрібно для моделювання роботи системи в умовах, наближених до реальних.

Верифікація та інтеграція. Після успішного тестування здійснюється верифікація системи, яка передбачає перевірку відповідності кінцевого продукту початковим вимогам. Даний етап включає інтеграцію всіх компонентів системи в єдине ціле. Проводиться фінальне тестування в реальних умовах експлуатації. Якщо всі вимоги виконано, система готова до виробництва або впровадження в роботу.

Розгортання (випуск), технічне обслуговування та оновлення. На цьому етапі програмне забезпечення встановлюється на пристрій. Після розгортання система тестується на фабриці працівниками, а також в цільовому середовищі, зазвичай, клієнтами. Якщо будь-які дефекти або проблеми виявляються – вони негайно усуваються. Якщо пристрій підтримує оновлення, це відбувається через випуск нової версії програмного забезпечення. Іноді це потрібно для додавання нових функцій, покращення продуктивності або щоб залишатися надійними та відповідати новим

стандартам.

Рівень деталізації та конкретні процеси на кожному етапі можуть відрізнятися. Все залежить від складності вбудованої системи, галузі, та вимогам, яким вона має відповідати.

Вбудовані системи Інтернету речей можна класифікувати за різними критеріями, для розробки ми використали таку категоризацію:

За рівнем обчислювальної потужності IoT-системи поділяються на мікроконтролерні (MCU-based) та мікропроцесорні (MPU-based) [30]. Мікроконтролерні системи базуються на 8-, 16- або 32-бітних мікроконтролерах і зазвичай використовуються для мобільних та енергоефективних пристроїв. Також вони використовуються для систем з вимогами реального часу. Вони часто функціонують без операційної системи (Bare-metal) або з легкими операційними системами реального часу (RTOS). Мікропроцесорні системи мають більшу обчислювальну потужність завдяки процесорам на базі ARM Cortex-A або RISC-V. Це дозволяє їм працювати з повноцінними операційними системами, такими як Linux або Android Things. Вони застосовуються для вирішення складних завдань, як обробка відео, штучний інтелект на периферії (Edge AI).

За архітектурою обробки даних IoT-системи поділяються на централізовані та розподілені [31]. Централізовані системи передають усі зібрані дані для обробки та аналізу на центральний сервер або хмарну платформу (наприклад, AWS IoT або Microsoft Azure IoT Hub). При такому підході спрощується керування системою та її підтримка, але створюється залежність від стабільного підключення до мережі. У розподілених IoT-системах обробка даних відбувається безпосередньо на пристроях або на локальних вузлах. Це зменшує затримку в передачі даних та навантаження на мережу. При такому підході також підвищується рівень автономності пристроїв.

Останнім важливим критерієм класифікації є енергоспоживання. IoT-системи можуть бути системи, що працюють від мережевого живлення або автономними пристроями. Девайси першого типу потребують постійного електроживлення через високі вимоги до обробки та передачі даних. Другий тип це бездротові датчики, переносні пристрої та автономні IoT-модулі. Вони працюють на батарейках або використовують технології енергозбору (Energy Harvesting). Також вони активно використовують ефективні режими сну (deep sleep) і оптимізовані для зменшення споживання енергії. Для передачі даних застосовуються енергоефективні протоколи зв’язку, як Bluetooth Low Energy (BLE), LoRaWAN або ZigBee.

Серед основних моделей SDLC, що застосовуються в розробці вбудованих систем, виділяють V-подібну модель, каскадну модель та гнучкі методології (Agile, SCRUM) [32].

Каскадна модель (Waterfall Model) – один із найстаріших методів проектування. Цей метод передбачає послідовне виконання етапів, при цьому кожен етап розпочинається лише після завершення попереднього. Цю модель використовують для проектів з чіткими та стабільними вимогами, що і є її основною перевагою. Вона активно використовується у військовій та аерокосмічній галузях. Її головний недолік, це неможливість зробити зміни після завершення етапу. Також мінусом є те, що тестування проводиться тільки після завершення всієї реалізації. Це призводить до виявлення критичних помилок на дуже пізньому етапі.

V-модель (V-Model) – це покращена версія каскадної моделі. В ній кожен етап передбачає одночасне проектування і тестування. Це дозволяє виявляти помилки на ранніх стадіях і забезпечує високу надійність. Вона активно використовується в медичних пристроях, авіаційній електроніці або автомобільних системах. Недоліки схожі на недоліки каскадної моделі.

Гнучкі методології розробки (Agile [33] та SCRUM [34]), орієнтовані на швидку адаптацію до змін та постійне вдосконалення продукту. На відміну від каскадної та V-моделі, Agile підхід передбачає ітеративну розробку. Робота ділиться на короткі цикли (ітерації або спринти). Кожен цикл включає аналіз вимог, розробку, тестування та демонстрацію результатів. Проте у великих комплексних проектах із жорсткими вимогами до сертифікації можуть виникнути перевищення бюджету і переноси кінцевих термінів. Незважаючи на це, подана методологія найчастіше використовується в проектуванні IoT проектів.

Швидке прототипування (Rapid Prototyping) – цей метод часто використовують з методами швидкої розробки застосунків(RAD) та Agile. Його суть полягає в розбитті процесу на три етапи які зациклені – створення прототипу, оцінка відгуків про прототип та покращення продукту на основі цих відгуків. Хоча цей метод значно пришвидшує швидкість розробки і надає максимальну гнучкість, з ним важко реалізувати проекти з жорсткими вимогами. Як і в Agile, велика кількість ітерації може збільшити вартість та час розробки.

Також в розробці вбудованих систем застосовується системне проектування на основі моделей (Model-Based Design). Використання моделей для створення вбудованих систем дозволяє спочатку проектувати, симулювати і перевіряти систему на рівні абстракції та реальної реалізації. Інструменти як MATLAB/Simulink надають можливість моделювати роботу системи в умовах, близьких до реальних. Це дозволяє швидко тестувати та оптимізувати проекти.

Архітектурний підхід визначає, як саме оброблятимуться дані, які апаратні ресурси будуть використані та як буде організовано програмне забезпечення пристрою. Вибір архітектури впливає на продуктивність, енергоефективність, масштабованість та загальну стабільність системи. Найпоширенішими є мікроконтролерний і мікропроцесорний підходи, архітектури з централізованою та розподіленою обробкою даних, а також використання операційних систем реального часу (RTOS) або програмування на рівні bare-metal [35].

Мікроконтролерний підхід передбачає використання мікроконтролерів (наприклад, STM32, ESP32, ATmega, Nordic nRF52). Вони мають вбудовану пам’ять (Flash, RAM) і периферійні модулі (UART, SPI, I2C, GPIO). Мікроконтролери забезпечують керування сенсорами, актуаторами та виконують алгоритми обробки даних. Системи на мікроконтролерах є енергоефективними, невеликим і з низькою вартістю. Проте вони мають обмежену обчислювальну потужність і розробка продукту для них триває довше ніж на потужніших мікропроцесорах. Для більшості мікроконтролерів існують готові плати розробника, але вони використовуються для створення прототипів. Для повноцінного продукту необхідно спроектувати власну апаратну платформу.

Мікропроцесорний підхід базується на використанні мікропроцесорів (наприклад, ARM Cortex-A, RISC-V, Intel Atom). Вони мають високу продуктивність і підтримують повноцінні операційні системи, такі як Linux або Android. На відміну від мікроконтролерів, мікропроцесори потребують зовнішньої оперативної та постійної пам’яті (RAM, eMMC, SSD), однак можуть виконувати складні багатозадачні процеси. Вони застосовуються у пристроях, які вимагають потужної обробки даних. Але ціною високої потужності є високе енергоспоживання. Для мобільних систем це компенсується великими акумуляторами, жертвуючи розміром. Часто для розробки прототипів використовуються готові одноплатні комп`ютери (Raspberry Pi, NVIDIA Jetson), які можуть бути використані як основна апаратна платформа продукту.

При централізованій обробці даних всі зібрані IoT-пристроями дані передаються на сервер або хмарну платформу. Це дозволяє використовувати потужні обчислювальні ресурси, реалізовувати складні алгоритми штучного інтелекту та Big Data. Основним недоліком цього підходу є залежність від мережевого підключення і ціна хмарних сервісів для великої кількості даних.

Розподілена обробка даних (Edge Computing, Fog Computing) децентралізує обробку даних. Частина або вся інформація аналізується безпосередньо на IoT пристроях або на локальних вузлах. Це підвищує автономність самих девайсів і знижує навантаження на хмарну інфраструктуру. Також такий підхід зменшує затримку в передачі даних, або взагалі її усуває для повністю автономних систем. Можливість швидко обробляти дані і приймати рішення на місці стає все більшою потребою, наприклад для автономного транспорту.

Під час розробки вбудованих IoT-систем важливо правильно обрати метод проектування та інструменти розробки, оскільки від цього залежить швидкість роботи, продуктивність і можливість подальшого вдосконалення цієї системи. Кожен метод має свої плюси і мінуси, які підходять для різних завдань. У таблиці 1 представлено їхній порівняльний аналіз за ключовими критеріями.

Таблиця 1

Порівняння методів проектування

Метод проектування	Гнуч-кість	Масштабо ваність	Швидкість розробки	Область застосування
Каскадна модель	Низька	Низька	Низька	Авіоніка, промисловість, критичні системи
V-подібна модель	Середня	Низька	Середня	Медичні пристрої, автомобільна електроніка
Agile (SCRUM, Kanban)	Висока	Висока	Висока	Розумний дім, стартапи, динамічні проєкти
Rapid Prototyping	Висока	Висока	Висока	IoT-платформи, гнучкі архітектури
Model-Based Design	Висока	Висока	Висока	Автоматизоване проектування, робототехніка

Каскадна модель підходить для систем із чітко визначеними вимогами, де важлива передбачуваність процесу розробки, але вона не підходить для динамічних проектів. Те саме стосується V-подібної моделі. Проте строгий контроль якості під час всіх етапі є необхідним для критичних IoT-систем з чіткими вимогами. Гнучкі методології дозволяють швидко адаптувати систему до змін, що робить їх ідеальними для розробки сучасних IoT-проектів, але постійні зміни можуть негативно позначитись на часі розробки. Rapid Prototyping добре підходить для невеликих проектів з обмеженим часом, де ще не до кінця відоме фінальне бачення продукту. Model-Based Design забезпечує високу точність моделювання та дозволяє оптимізувати систему ще на етапі проектування.

При виборі засобів розробки необхідно враховувати ресурси пристрою, рівень складності розробки на ньому і можливості подальшого покращення системи (Таблиця 2).

Таблиця 2

Порівняння платформ для розробки вбудований систем

Платформа	Продуктивність	Енергоефективність	Підтримка периферії, окрім базової (UART, I2C, SPI)	Область застосування
STM32	Висока	Висока	Широка	Промисловість, медичні пристрої, IoT-контролери
ESP32	Середня	Висока	Wi-Fi, Bluetooth	Розумний дім, IoT-гаджети, бездротові пристрої
Nordic nRF	Середня	Висока	BLE, Zigbee, Thread	Переносні пристрої, медичні сенсори, трекери
Raspberry Pi	Висока	Низька	USB, Ethernet, HDMI	IoT-шлюзи, відеоаналітика, машинне навчання
NVIDIA Jetson	Дуже висока	Низька	GPU-обчислення	Комп’ютерний зір, автономні системи
AVR (ATmega, ATtiny)	Низька	Висока	-	DIY-проєкти, прості IoT-пристрої
Infineon XMC	Висока	Висока	CAN, Ethernet, PWM	Промислова автоматизація, робототехніка, автомобільна електроніка
Ameba IoT (RTL 8710, 8722)	Середня	Середня	Wi-Fi, Bluetooth	Хмарні IoT-рішення, Smart Home

STM32 є універсальним рішенням із широким спектром застосувань. Вона підтримує RTOS та є широкий вибір в плані промислових стандартів. ESP32 добре підходить для бездротових IoT-рішень, де важливе Wi-Fi або BLE-з'єднання, хоча вибір мікроконтролерів менший, це компенсується високою доступністю. AVR є класичним вибором для простих вбудованих систем які можна інтегрувати в IoT-систему. Великим плюсом є платформа Arduino, проте ці мікроконтролери не дуже підходять для повноцінного чи самостійного IoT-продукту.

Raspberry Pi використовується для складних IoT-рішень, які важко реалізувати на мікроконтролері. NVIDIA Jetson є найкращим вибором для AI та відеоаналітики, хоча ці задачі можна реалізувати і на Raspberry Pi. Зараз, серед виробників мікроконтролерів, набирають популярність потужні рішення, спеціалізовані для Edge Computing та ШІ, проте їх ціна значно вища за наведені одноплатні комп`ютери.

Існують більш спеціалізовані платформи: Nordic nRF, який спеціалізується на Bluetooth LE. Ці мікроконтролери є ідеальними для невеликих енергоефективних пристроїв і активно в них використовуються. Infineon XMC є високопродуктивною платформою для промислових рішень і може підійти, якщо ваш проект потребує швидкої обробки даних з CAN чи Ethernet. Ameba IoT, є спеціалізованою платформою для IoT пристроїв на Wi-Fi та Bluetooth.

Окрім апаратної платформи, велике значення має вибір програмного середовища. Хороше IDE дозволить ефективно розробляти, налагоджувати та тестувати IoT-пристрої (Таблиця 3).

Таблиця 3

Порівняння IDE для розробки вбудований систем

Інструмент	**Гнучкість налаштувань**	**Оптимізація коду**	**Простота використання**	**Підтримувані платформи**
Keil uVision	Висока	Дуже висока	Середня	ARM Cortex
STM32CubeIDE	Висока	Висока	Висока	Продукти ST
PlatformIO	Дуже висока	Висока	Середня	ESP32, STM32, AVR, Nordic nRF, XMC, Ameba IoT
Arduino IDE	Середня	Низька	Дуже висока	ESP32, STM32, AVR, Ameba IoT
IAR Embedded Workbench	Висока	Дуже висока	Середня	ARM Cortex, MSP430, AVR
Atmel Studio (Microchip Studio)	Висока	Висока	Середня	AVR, SAM (Cortex-M)
DAVE (Infineon)	Висока	Висока	Середня	Продукти Infineon

Keil uVision та IAR Embedded Workbench є професійними середовищами для розробки ARM-мікроконтролерів. Вони використовуються у промислових та критичних IoT-системах. Хоча вони є потужним інструментом для розробки і налагодження, вони є платними і мало підходять для невеликих проектів.

Кожен виробник мікроконтролерів зазвичай надає засоби для їх розробки на них. Наприклад STM32CubeIDE від ST. Вона забезпечує інтегрований підхід до розробки на STM32 з широким інструментарієм. Atmel Studio (Microchip Studio) – офіційне середовище для розробки AVR та ARM Cortex-M. DAVE IDE – спеціалізоване середовищем для XMC-мікроконтролерів Infineon. Вони добре підходять для розроби на своїх платформах, але не можуть бути використані на інших, що є їх основним недоліком. При комплексній IoT-системі стається нагромадження різних інструментів строго спеціалізованих під одну задачу.

PlatformIO є універсальним середовищем. Адже воно підтримує багато платформ і не є пропрієтарним. Це робить його ефективним для IoT-проектів, що включають багато платформ.

Проте їй бракує спеціалізованих інструментів для проектування, які присутні в інструментах від виробника. Arduino IDE підходить для швидкого прототипування IoT-рішень, але має менше можливостей оптимізації і налагодження. Плюсом Arduino є її широка громада, яка активно підтримує цей продукт і надає багато готових бібліотек.

Висновки

Розробка вбудованих систем Інтернету речей (IoT) є складним багаторівневим процесом. Він вимагає ретельного вибору методології проектування, апаратних засобів та програмного забезпечення. У ході дослідження були розглянуті та проаналізовані основні методи проектування IoT-систем, їхні переваги та недоліки. Було порівняно популярні апаратні платформи і середовища розробки. Отримані результати дозволяють оцінити сучасний стан технологій у цій галузі та визначити основні виклики, які стоять перед розробниками IoT-систем.

Проведений аналіз показав, що вибір методів проектування впливає на швидкість розробки, масштабованість та ефективність системи. Каскадна та V-модель підходять для надійних проектів, тоді як Agile, Rapid Prototyping та Model-Based Design забезпечують більшу гнучкість. Модульний підхід є кращим для великих мереж IoT-пристроїв.

Оцінивши апаратні платформи можна сказати, що кожна з них має своє призначення і ще не існує єдиного універсального рішення. STM32 є хорошим рішенням для IoT-приладів, проте йому бракує засобів бездротової комунікації. ESP32 підходить для бездротових IoT-рішень, проте їй не вистачає протоколів комунікації. Raspberry Pi та NVIDIA Jetson використовуються для обробки великих об`ємів даних та ШІ, але вони не можуть бути використані для невеликих мобільних рішень. XMC та Ameba IoT розширюють можливості вибору платформи, але вони розробляються для вирішення конкретних задач.

Серед програмних середовищ найкращими для професійної розробки є Keil та IAR Embedded Workbench, проте вони є платними. PlatformIO є хорошим універсальним рішенням, проте йому бракує деякого інструментарію. Arduino IDE підходить для прототипування та експериментальних проектів, але не для комерційних рішень. Спеціалізовані середовища забезпечують оптимізацію для апаратних платформ, під які вони розроблялись.

Нами вже використовувались ці методи і засоби при проектуванні системи наведення та слідкування [36], але для прикладу візьмемо вбудовану систему управління, яку ми розробляли для БПЛА [37]. Під час етапу визначення вимог та планування, була описана проблема та основна ідея її вирішення. Оскільки це був прототип, для розробки була обрана методологія Agile, що дозволяло нам покращувати наш продукт безпосередньо під час його розробки.

Вибір апаратної платформи та архітектури був диктований потребою у відеообробці, через це був обраний мікропроцесорний підхід і мікрокомп`ютер NanoPi Neo Plus 2, який є потужною альтернативою Raspberry Pi Zero. Також на цьому етапі була обрана система навігації – Matek F-405 Wing V2 та GPS модуль з компасом. Важливим параметром для вибору компонентів була симуляція та побудова корпусу БПЛА, його габарити і параметри ставили обмеження на розмір та енергоспоживання системи.

Під час розробки програмного забезпечення також відбувалось і збирання прототипу. Під час того як розроблявся код для обробки відео і роботи, був зібраний корпус, що стало можливим завдяки використання модульного підходу. Після цього почалось тестування та налагодження вбудованої системи.

Щодо інтеграції цього продукту в систему Інтернет речей, було обрано розподілену архітектуру обробки даних. Відео буде обробляється безпосередньо на БПЛА, проте ці пристрої є клієнт-серверними, отже керуються з централізованого сервера. В перспективі декілька БПЛА можуть керуватися та моніторитися з одного сервера, а самим вбудованим системам буде наданий певний рівень автономності.

Незважаючи на значний прогрес у сфері IoT, розробники стикаються з низкою викликів, які впливають на ефективність IoT-рішень. Багато IoT-пристроїв працюють на батарейках чи

акумуляторах, тому оптимізація енергоспоживання є ключовим фактором. Використання енергоефективних мікроконтролерів (наприклад, Nordic nRF, STM32 Low Power) та бездротових протоколів (LoRa, Bluetooth LE) дозволяє продовжити автономну роботу пристроїв, однак це потребує складних алгоритмів керуванням енергоспоживанням.

Розширення IoT-екосистеми створює виклики у взаємодії пристроїв різних виробників. Відсутність єдиних стандартів ускладнює інтеграцію пристроїв у загальну систему. Поява Matter (від CSA, Zigbee Alliance) як нового стандарту для IoT має потенціал вирішити цю проблему.

Перелік використаних джерел

[1] Razzaque, Mohammad Abdur & Milojevic, Marija & Palade, Andrei & Clarke, Siobhán. (2015). Middleware for Internet of Things: A Survey. IEEE Internet of Things Journal. 3. 1-1. https://doi.org/10.1109/JIOT.2015.2498900.

[2] Sedrati, Anass & Mezrioui, Abdellatif. (2018). A Survey of Security Challenges in Internet of Things. Advances in Science, Technology and Engineering Systems Journal. 3. 274-280. https://doi.org/10.25046/aj030133.

[3] Halder, Saroj & Adhikary, Arka & Bose, Rayith & Panja, Shuvadeep & Halder, Sourav & Pratihar, Jayanta & Dey, Arindam. (2024). Design and Implementation of an IOT-based Smart Home Automation System in Real World Scenario. EAI Endorsed Transactions on Internet of Things. https://doi.org/10.4108/eetiot.6201.

[4] Kaisti, M., Rantala, V., Mujunen, T. et al. Agile methods for embedded systems development - a literature review and a mapping study. J Embedded Systems 2013, 15 (2013). https://doi.org/10.1186/1687-3963-2013-15

[5] Saopan, M. A. (2023). Waterfall Method of Web-Based System to Develop Warehouse Packing Effectively. Asia Information System Journal, 2(2), 77–86.

[6] Supramono, S., & Prasetyo Adi, P. D. (2021). Waterfall pattern using Omron CP1E PLC, Fuzzy Logic Method, and PLC-IoT approach. Internet of Things and Artificial Intelligence Journal, 1(3), 159–175.

[7] L. Shuping and P. Ling, "The Research of V Model in Testing Embedded Software," 2008 International Conference on Computer Science and Information Technology, Singapore, 2008, pp. 463-466, https://doi.org/: 10.1109/ICCSIT.2008.51.

[8] Gutiérrez Rivas, José & Berthing, Jesper & Fernández García-Valdecasas, David & Díaz, Javier. (2012). Safety-Critical Platform Model Based on Certification Standards.

[9] Moedt, W., Bernsteiner, R., Hall, M., & Fruhling, A. (2023). Enhancing IoT Project Success through Agile Best Practices. ACM Transactions on Internet of Things, 4(1), Article 5.

[10] Kaisti, M., Rantala, V., Mujunen, T. et al. Agile methods for embedded systems development - a literature review and a mapping study. J Embedded Systems 2013, 15 (2013). https://doi.org/10.1186/1687-3963-2013-15

[11] Houyou, Amine & Huth, Hans-Peter & Kloukinas, Christos & Trsek, Henning & Rotondi, Domenico. (2012). Agile Manufacturing: General Challenges and an IoT@Work Perspective. https://doi.org/ 10.1109/ETFA.2012.6489653.

[12] Guerrero Ulloa, Gleiston & Rodríguez-Domínguez, Carlos & Hornos, Miguel. (2023). Agile Methodologies Applied to the Development of Internet of Things (IoT)-Based Systems: A Review. Sensors. 23. 790. https://doi.org/10.3390/s23020790.

[13] Y. M. Tashtoush et al., "Agile Approaches for Cybersecurity Systems, IoT and Intelligent Transportation," in IEEE Access, vol. 10, pp. 1360-1375, 2022, https://doi.org/10.1109/ACCESS.2021.3136861.

[14] Fuchs, Christoph and Hess, Thomas, (2017). "ADAPTING AGILE METHODS TO DEVELOP SOLUTIONS FOR THE INDUSTRIAL INTERNET OF THINGS". In Proceedings of the 25th European Conference on Information Systems (ECIS), Guimaraes, Portugal, June 5-10,2017 (pp. 2852-2861). ISBN 978-0-9915567-0-0 Research-in-Progress Papers.

[15] Yin-Tsung Hwang, Cheng-Ji Chang and Bor-Liang Chen, "A rapid prototyping embedded system platform and its HW/SW communication interface generation and verification," Asia-Pacific Conference on Circuits and Systems, Denpasar, Indonesia, 2002, pp. 481-484 vol.1, https://doi.org/10.1109/APCCAS.2002.1115036.

[16] G. Guan, W. Dong, Y. Gao and Jiajun Bu, "Towards rapid and cost-effective prototyping of IoT platforms," 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, 2016, pp. 1-5, https://doi.org/10.1109/ICNP.2016.7785320.

[17] R. Brzoza-Woch, Ł. Gurdek and T. Szydlo, "Rapid Embedded Systems Prototyping - an Effective Approach to Embedded Systems Development," 2018 Federated Conference on Computer Science and Information Systems (FedCSIS), Poznan, Poland, 2018, pp. 629-636.

[18] G. Tanganelli, C. Vallati and E. Mingozzi, "Rapid Prototyping of IoT Solutions: A Developer's

Perspective," in IEEE Internet Computing, vol. 23, no. 4, pp. 43-52, 1 July-Aug. 2019, https://doi.org/10.1109/MIC.2019.2927202

[19] Hester, J., & Sorber, J. (2017). Flicker: Rapid Prototyping for the Batteryless Internet-of-Things. In Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (SenSys '17) (pp. 1–13). ACM

[20] S. Mora, F. Gianni and M. Divitini, "RapIoT Toolkit: Rapid Prototyping of Collaborative Internet of Things Applications," 2016 International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, USA, 2016, pp. 438-445, https://doi.org/10.1109/CTS.2016.0083

[21] Schätz, B., Pretschner, A., Huber, F., Philipps, J. (2002). Model-Based Development of Embedded Systems. In: Bruel, JM., Bellahsene, Z. (eds) Advances in Object-Oriented Information Systems. OOIS 2002. Lecture Notes in Computer Science, vol 2426. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-46105-1_34

[22] D. de Niz, G. Bhatia and R. Rajkumar, "Model-Based Development of Embedded Systems: The SysWeaver Approach," 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), San Jose, CA, USA, 2006, pp. 231-242, https://doi.org/10.1109/RTAS.2006.30.

[23] S. Mazzini, J. Favaro and L. Baracchi, "A Model-Based Approach Across the IoT Lifecycle for Scalable and Distributed Smart Applications," 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Gran Canaria, Spain, 2015, pp. 149-154, https://doi.org/10.1109/ITSC.2015.33.

[24] Rashed, M. & Hassan, Ahmed & Sharaf, Ahmed. (2011). Model Based System Engineering Approach of a Lightweight Embedded TCP/IP. International Journal of Computer Science & Information Technology. 3. https://doi.org/10.5121/ijcsit.2011.3207.

[25] Natarajan, Muthukumar & Srinivasan, Seshadhri & Ramkumar, Kannan & Pal, Deepak & Vain, Juri & Ramaswamy, Srini. (2019). A model-based approach for design and verification of Industrial Internet of Things. Future Generation Computer Systems. 95. https://doi.org/10.1016/j.future.2018.12.012.

[26] Al-Fuqaha, Ala & Guizani, Mohsen & Mohammadi, Mehdi & Aledhari, Mohammed & Ayyash, Moussa. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. IEEE Communications Surveys & Tutorials. 17. Fourthquarter 2015. https://doi.org/10.1109/COMST.2015.2444095.

[27] Mishra, Ayaskanta. (2019). Embedded Development Platforms To Design Prototypes Of Internet Of Things (IoT) Applications: A Study. International Journal of Research in Advent Technology. 7. 344-353. https://doi.org/10.32622/ijrat.742019133.

[28] Shafiq, Muhammad; Gu, Zhaoquan; Cheikhrouhou, Omar; Alhakami, Wajdi; Hamam, Habib. "The Rise of "Internet of Things": Review and Open Research Issues Related to Detection and Prevention of IoT-Based Security Attacks". Wireless Communications and Mobile Computing. 2022: e8669348. https://doi.org/10.1155/2022/8669348. ISSN 1530-8669

[29] “The Complete Guide to Embedded System Development Life Cycle” [electronic resource] – Access mode: https://apptread.com/guides/embedded-system-development-life-cycle/

[30] Wei Zhou, Zhouqi Jiang, Le Guan, “Understanding MPU Usage in Microcontroller-based Systems in the Wild”, Workshop on Binary Analysis Research, 2023.

[31] O. Salman, I. Elhajj, A. Kayssi and A. Chehab, "An architecture for the Internet of Things with decentralized data and centralized control," 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, Morocco, 2015, pp. 1-8, https://doi.org/10.1109/AICCSA.2015.7507265.

[32] Qian, K., den Haring, D., Cao, L. (2009). Embedded Software Design and Development. In: Embedded Software Development with C. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-0606-9_2

[33] “Agile Manifesto” [electronic resource] – Access mode: http://www.agilemanifesto.org

[34] Schwaber K, “SCRUM development process. In Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems”, Languages, and Applications (OOPSLA). Austin, Texas, USA; October 1995:117-134.

[35] Tammy Noergaard, “Architectural Patterns” в “Embedded Systems Architecture 2nd Edition”, Packt Publishing , 2012

[36] K. Kolesnyk, R. Panchak, I. Kozemchuk and Z. Skybinska, "Development of an Automated Subsystem for Modeling and Calculating a Mirror Antenna From its Guiding and Tracking the Target," 2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), Polyana, Ukraine, 2019, pp. 1-5, https://doi.org/10.1109/CADSM.2019.8779297.

[37] Syrotynskyi T., Kolesnyk K., Kozemchuk I., Holovatyy A., Łukaszewicz A., 2024, 3D MODELLING OF UAV AND CREATING IT’S SYSTEM OF CONTROL, CDS. Volume 6, Number 3: 17- 23 https://doi.org/10.23939/cds2024.03.017

Kostyantyn Kolesnyk [1], Ivan Kozemchuk [2],

[1] Computer Design Systems Department, Lviv Polytechnic National University,
12, St. Bandera str., Lviv, Ukraine, E-mail: kostyantyn.k.kolesnyk@.lpnu.ua, ORCID0000-0001-9396-595X
[2] Computer Design Systems Department, Lviv Polytechnic National University,
12, St. Bandera str., Lviv, Ukraine, E-mail: ivan.v.kozemchuk@lpnu.ua

ANALYSIS OF METHODS AND TOOLS FOR DESIGNING EMBEDDED SYSTEMS OF THE INTERNET OF THINGS

Received: March 03, 2025 / Revised: March 14, 2025 / Accepted: March 20, 2025

Adstract. The article analyzes the methods and tools for designing embedded Internet of Things (IoT) systems. The main stages of developing IoT systems are considered, the main design approaches are compared, and their advantages and limitations are identified. The analysis of hardware platforms, their characteristics, performance, energy efficiency, and applications in various fields is conducted. Considered Software tools and their effectiveness in developing IoT solutions. Particular attention is paid to architectural solutions that affect the performance, energy efficiency, and scalability of IoT solutions. The challenges associated with security, standardization, and energy efficiency of embedded systems are considered. The results obtained may be useful for researchers, engineers, and developers of embedded IoT solutions, as well as for optimizing design processes and selecting the most effective technologies in this area.

Keywords: Internet of Things (IoT), embedded systems, design methods, microcontrollers, microprocessors, system architecture, distributed systems, hardware platform, Edge Computing.