УДК 004.422.8:004.77 **doi: 10.32620/aktt.2024.3.09**

О. О. ВДОВІЧЕНКО

Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна

АНАЛІЗ ТЕХНОЛОГІЙ РЕКОНФІГУРАЦІЇ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ НА РІВНІ ПРОГРАМНИХ МОДУЛІВ ТА ЗАВАНТАЖУВАЧІВ

Предметом *вивчення в статті є технології програмування і взаємодії вузлів, а також окремих малогабаритних складових компонентів вбудованих систем Internet of Things (IoT).* ***Метою*** *є підвищення гнучкість компонентів систем IoT за рахунок спрощення процесу реконфігурації окремих вузлів і компонентів.* ***Завдання****: проаналізувати існуючі підходи до побудови конфігурованих систем, проаналізувати та описати існуючі рішення, які реалізують описані підходи; проаналізувати інструменти для реалізації вибраних підходів, описати метод побудови живучої системи на основі аналізованого підходу, надати технічний приклад запропонованого методу програмування.* *Відповідно до поставлених завдань, були отримані наступні* ***результати****. Проведено аналіз можливих шляхів реконфігурації компонентів IoT. Розглянуто способи реконфігурації на рівні мережі та часткової реконфігурації вузла. Проаналізовано та класифіковано шляхи забезпечення гнучкості компонентів IoT. Проведено порівняльний аналіз зведених параметрів завантажувачів. Проаналізовано можливість використання програмних компонентів у якості незмінних блоків коду для модифікації програми мікроконтролера як способу реконфігурації. Запропоновано ідею використання змінних програмних модулів для представлення програми мікроконтролера з модифікованим завантажувачем. Запропоновано елементи методу для перепрограмування змінних програмних модулів мікроконтролера як проміжниого програмного забезпечення.* ***Висновки****. Відповідно до огляду існуючих типів систем, їх було класифіковано за рівнем організації їхньої реконфігурованої частини. Відповідно до отриманої попередньої класифікації розглянуто низку архітектурних рішень з точки зору покращення експлуатаційних характеристик. Було виділено основні переваги для кожного типу організації реконфігурованої частини систем. Проведено аналіз практичних рішень щодо модифікації вбудованих компонентів мікроконтролерів для управління та оновлення їх мікропрограм. Практичне значення даного дослідження полягає запропонуванні методу програмування зі змінними програмними модулями для мікроконтролера та отримані результати конкурентного аналізу способів модифікації компонентів і особливостей завантажувачів.*

Ключові слова: *реконфігурація програмного забезпечення, реконфігуровані вузли, Fog computing, завантажувач, Edge-IoT, змінні програмні модулі.*

Вступ

У зв'язку з інтенсифікацією зростання попиту на обчислювальні потужності сучасних систем виникає потреба в їх постійному розширенні та модифікації. Для вбудованих систем, що реалізуються у фіксованому об'ємі, у тому числі бортових та космічних систем, існує межа, за якою модифікація і додавання обладнання стають неможливими. У зв'язку з розширенням комунікаційних можливостей та збільшенням кількості готових компонентів для створення складних обчислювальних систем зростає і різноманіття підходів до їх побудови. Залежно від наявних ресурсів та базових умов розробки, можна налаштовувати тип розроблюваного рішення.

Виробники сучасних виконавчих модулів представляють широкий вибір мікросхем різної цінової категорії та якості виконання [1]. Залежно від призначення, модуль може бути виконаний або у вигляді компонента, що являє собою плату з базовими інтерфейсами взаємодії та набором виводів для вільного використання, або у вигляді готового пристрою в монолітному корпусі з набором спеціалізованих інтерфейсів взаємодії та мінімальними надмірностями [2]. Такий контраст пояснюється сферою застосування модуля та розміром серії, в рамках якої він був виготовлений.

Незважаючи на різноманітність випущених обчислювальних модулів і пристроїв для побудови вбудованих систем, їх ресурсні та часові можливості залишаються обмеженими [3]. Окремі протоколи обміну та моделі взаємодії пристроїв дозволяють виконувати модифікацію програмного коду пристроїв [4]. Частина таких систем та компонентів надає можливість управління та моніто-

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

рингу з використанням мобільного телефону [5]. Така можливість може бути використана для забезпечення реконфігурації пристроїв, у тому числі дистанційної [6].

Для систем розрахованих на довгий період роботи інженери розробляють способи їх обслуговування і підтримки, що дозволяють доповнювати і модифікувати вже розгорнуті системи. На жаль, можливість розширення систем також може бути обмежена через габарити та енергоспоживання. Існує також межа складності систем, коли кількість задіяних в ній пристроїв досягає значень, при яких ефективне управління ресурсами стає складним, що впливає на швидкість і якість виконання покладених на систему завдань.

Мотивацією до проведення поточного дослідження виступає потреба в альтернативному підході до модернізації електронних вбудованих систем, заснованих на взаємодії декількох пристроїв між собою. Враховуючи той факт, що серійні пристрої виробляються стандартизовано, у вигляді, сумісному з поширеними рішеннями та інтерфейсами, при застосуванні на практиці частина закладених в них ресурсів може бути не використана. У зв'язку з цим стандартні модулі мають природний ресурсний надлишок, який при правильному підході може бути використаний для потреб всієї системи.

Враховуючи все вищесказане, виникає ідея сформулювати підхід, що використовує притаманну контролерам природну надлишковість з метою підвищення надійності та довговічності інтегрованих систем. Такий підхід передбачає врахування та використання надлишкових компонентів як архітектури, так і складових компонентів системи, що дозволить уникнути збоїв або перетворити їх на плавну деградацію.

Згідно **сучасного стану** досліджень з цього напрямку, основним споживачем апаратних ресурсів є програма, що виконується контролером, виникає завдання, пов'язане з управлінням вмістом внутрішньої пам'яті і мікропрограми. Компонент, який входить до складу багатьох мікроконтролерів, називається завантажувачем, добре підходить для цього завдання.

Завантажувач - це програма, призначена в першу чергу для полегшення покращення або зміни системного програмного забезпечення без необхідності використання спеціалізованих інструментів для оновлення прошивки. Хоча завантажувачі можуть виконувати різні функції, їхня основна роль полягає у нагляді за програмою. Вони можуть використовувати різні протоколи, такі як UART, CAN, I2C, I2S, Ethernet або USB, щоб ініціювати зв'язок і розпочати процес оновлення прошивки [7].

Однак у своєму стандартному вигляді завантажувач не дозволяє перепрограмувати мікроконтролери, які він обслуговує, без використання сторонніх пристроїв і програм. З цієї причини вважається доцільним використовувати модифікований завантажувач, щоб уможливити віддалене програмування або реконфігурацію [6, 7].

Метою даної роботи є підвищення гнучкості компонентів систем IoT за рахунок спрощення процесу віддаленої реконфігурації окремих вузлів і компонентів.

Для досягнення цієї мети необхідно виконати наступні **завдання**:

1) проаналізувати існуючі підходи до побудови реконфігурованих систем;

2) описати та проаналізувати існуючі рішення, які реалізують описані підходи;

3) проаналізувати інструменти для реалізації описаних підходів;

4) запропонувати елементи методу побудови живучої системи;

5) надати технічний приклад запропонованого методу програмування.

1. Аналіз можливостей реконфігурації компонентів IoT

Аналіз способів реконфігурації на рівні мережі. Для Індустрії 4.0 пропонується змінити підхід до управління пристроями Інтернету речей, де велика кількість пристроїв Інтернету речей взаємодіє один з одним із забезпеченням ефективної взаємодії між машинами. У випадку, коли традиційні методи управління є неефективними через проблеми розподіленого управління та неефективного розподілу ресурсів. Пропонується використовувати програмно-визначену мережу (SDN), яка розділяє управління мережею та передачу даних між пристроями.

Програмно-визначені мережі дають можливість оптимізувати управління традиційними мережами. Особливо SDN-контролер має глобальне уявлення про мережу датчиків і виконавчих механізмів, що дозволяє динамічно реконфігурувати вузли мережі і потоки даних в режимі реального часу. Це особливо актуально для пристроїв з обмеженими комунікаційними та обчислювальними ресурсами. Запропонована архітектура SDN-IoT зосереджена на розробці контролера, який динамічно оптимізує доставку наскрізних потоків в режимі реального часу. Основою для цього є адаптація політик динамічної маршрутизації на основі безперервної оцінки стану мережі, що дозволяє

їм спільно мінімізувати затримки та енергоспоживання, тим самим збільшуючи термін служби мережі.

Проект ITEA 2 "Мережа об'єктів" пропонує рішення, спрямоване на взаємодію пристроїв на основі Інтернету речей в контексті управління надзвичайними ситуаціями. У цьому рішенні обмін повідомленнями є семантично збагаченим. Пропонується вирішення трьох взаємопов'язаних проблем: забезпечення сумісності управління пристроями на основі відповідної розподіленої архітектури; налаштування робочих процесів для спільної роботи пристроїв при забезпеченні автономності; встановлення оптимального формату обміну даними для пристроїв.

Концепція смарт-об'єктів (SObjs) забезпечує підхід до конфігурації та управління смарт-пристроями як мережевими об'єктами. Управління об'єктами SObj відіграє важливу роль у запобіганні потенційним проблемам перевантаження мережі IoT та зменшенні затримок. Запропонований метод, відомий як MbDSAS, є підходом до реконфігурації мережевих блоків або шлюзів без необхідності оновлення або зміни програмного забезпечення для управління і виявлення об'єктів SObj і підтримки динамічності мережі IoT. Використовуючи цей метод, шлюзи налаштовуються для ефективного управління SObj за допомогою оновлень або реконфігурацій програмного забезпечення з подальшим "теплим" запуском. MbDSAS пройшла пілотне тестування, щоб підтвердити свою придатність як рішення для управління сценаріями IoT і визначити найкращу комбінацію технологій для його успішного впровадження.

В іншому рішенні пропонується перенести обчислювальні операції з Edge в IoT, змінивши поведінку його вузлів відповідно до вимог програми або системи. Для цього реалізована технологія оновлення Multi-Hop-Over-The-Air, яка автоматично налаштовує IoT-пристрої на базі мікроконтролерів. З урахуванням вузлів IoT, підключених до mesh-мережі, розроблена розподілена екосистема для сприяння співпраці та швидкого розгортання коду на гетерогенних вузлах mesh-мережі Edge-IoT. Таким чином, система автоматично розгортає нові сервіси, коли це необхідно.

Промислові системи Інтернету речей (IP) надають різноманітні послуги на периферійних вузлах для підвищення ефективності та автоматизації роботи системи. Виникає концепція хостингу сервісів, де периферійні вузли динамічно реконфігуруються для розміщення останніх запитаних сервісів від сенсорних вузлів. Враховуючи обмеженість ресурсів зберігання та обчислювальних ресурсів на периферійних вузлах, вводиться функція реконфігурації, яка може розширити кількість та типи сервісів, розміщених на цих вузлах.

У випадку збору даних у реальному часі за допомогою бездротових датчиків, підключених до Інтернету речей, виникають проблеми із забезпеченням достатнього покриття, довговічності мережі та підтриманням бездротового з'єднання. Для додатків глобального дистанційного моніторингу (WARM) пропонується легкий, динамічний і автоматично реконфігурований протокол зв'язку (LDAP). Цей протокол включає мобільний поглинач даних для забезпечення ширшого покриття бездротової сенсорної мережі (WSN) і можливість автоматичної реконфігурації для адаптації до динамічних топологій мережі. Покращення покриття та терміну служби WSN також досягається завдяки використанню бездротового інтерфейсу дальнього радіусу дії (LoRa).

Відповідно до запропонованих рішень можна зробити висновок, що в умовах обмеженого простору та доступу можна підвищити показники продуктивності системи за рахунок зміни внутрішньої структури мережі. Завдяки такій реконфігурованості мережі для цього типу мереж з'являється можливість призначати ролі і права пристроям-учасникам, що, в свою чергу, впливає на розподіл обчислювального і трафікового навантаження. У цьому випадку можна припустити, що регулювання, перерозподіл і контроль пристроїв можуть виступати інструментами забезпечення надійності системи. Іншими словами, за рахунок розподілу обчислювального і трафікового навантаження зменшується знос пристроїв, задіяних в мережі, а завдяки управлінню пристроями на мережевому рівні з'являється можливість, при необхідності, видаляти несправних або тих, що вийшли з ладу учасників мережі без її перезавантаження або оновлення. В описаному випадку можна скоротити час простою і збільшити термін служби системи.

Аналіз часткової реконфігурації вузлів показує, що реконфігуровані обчислювальні архітектури успішно використовуються в критично важливих для безпеки сферах. У міру розвитку цільової архітектури виникає необхідність її віддаленого оновлення на відповідній платформі. Цей процес схильний до ризику віддаленого втручання, коли зловмисник може зловмисно змінити конфігурацію цільового обладнання, що реконфігурується. Запропоновано архітектуру, яка включає довірене апаратне забезпечення з використанням криптографічних співпроцесорів та модулів довіреної платформи (TPM), а також реалізацію оновлень через бездротову мережу. Платформа, що розробляється, реалізує протокол безпечного

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

завантаження на польових програмованих матрицях (ПЛІС) з використанням технології Xilinx. Проект демонструє успішну конфігурацію бітового потоку, інтеграцію процесу завантаження з TPM та безпечне бездротове оновлення для реконфігурації обладнання.

Деякі системи IoT надають можливості віддаленої динамічної часткової реконфігурації (DPR) для сучасних ПЛІС. Ці можливості дозволяють вносити зміни в схему, відображену на ПЛІС, щоб модифікувати або розширити функціональність пристрою без необхідності відключення його від мережі, за допомогою віддаленого мережевого зв'язку.

Для таких рішень потрібна безпечна та надійна система оновлення. Це критичний елемент для успішного впровадження Інтернету речей, особливо в масштабах великих комерційних рішень. При аналізі специфічних проблем інфраструктур, призначених для віддаленого розгортання і управління додатками, виділяються проблеми управління, пов'язані з сенсорними системами IoT. Запропоновано математичну модель та методологію розв'язання цих задач. Для оцінки ефективності моделі її було реалізовано у вигляді програмної інфраструктури для повноцінних комерційних продуктів у сфері Інтернету речей [8].

В іншому прикладі пропонується рішення, яке включає легкі криптографічні алгоритми Espresso і Grain 128 [9] для забезпечення конфіденційності даних в проектах на основі технології FPGA в IoT, а також алгоритм CRC32 для перевірки цілісності даних. Це рішення призначене для використання на платах Digilent Basys 3 Artix-7 з використанням Vivado. Для оптимізації продуктивності передбачена часткова реконфігурація, що дозволяє перемикатися між алгоритмами в залежності від потоку трафіку IoT та умов безпеки.

Використання часткової реконфігурації на основі різниці, хоча і простіше в реалізації через відсутність необхідності попереднього планування поверху, рекомендується обмежити невеликими змінами через її непередбачуваність. У цій статті пропонується механізм, який запобігає цій проблемі, зберігаючи глобальний стан системи. Таким чином, не має значення, як частковий бітовий потік на основі відмінностей вплине на апаратну конфігурацію. Це означає, що проектування частково реконфігурованої системи-на-кристалі вимагає менших зусиль в процесі розробки.

На основі запропонованих рішень можна зробити висновок, що при обмежених можливостях розширення системи можна використовувати обчислювальні потужності представлених в ній пристроїв, переосмисливши їх ролі в системі і способи взаємодії. Представляючи пристрої як рівноправні вузли системи, стає можливим перекваліфікувати їх відповідно до поточної задачі. Завдяки такій вузловій реконфігурованості з'являється можливість оптимізувати робочий процес системи в цілому, а також регулювати точність і швидкість виконуваних обчислень. У цьому випадку можна припустити, що оптимізація робочого процесу призведе до рівномірного і доцільного розподілу навантаження між виконавчими вузлами. Іншими словами, завдяки оптимізації зменшується кількість операцій, які виконує система, що зменшує знос пристроїв, які виступають в ролі вузлів. Завдяки реконфігурації пристроїв на рівні виконуваної програми з'являється можливість обійти апаратні дефекти за рахунок допустимого зниження точності або швидкості обчислень. В описаному випадку вдається підвищити живучість і термін служби системи.

Розбір випадків використання завантажувачів. Завантажувач є важливим компонентом вбудованої системи ARM, і реалізація цього завантажувача тісно пов’язана з апаратними характеристиками. У ньому наведено короткий опис плати розробки S3C2440 і режим її запуску. Основна увага приділяється ініціалізації кожного функціонального модуля в S3C2440 під час запуску системи, а також представлена схема спрощеного завантажувача для цієї плати. Після випробувань цей навантажувач продемонстрував високу ефективність і мобільність [10].

Завантажувач отримує додаток від хоста через комунікаційний інтерфейс і записує його в програмну пам'ять контролера. Кожен хост завантажувача дотримується певного протоколу передачі файлів, який зазвичай є унікальним для постачальника мікросхеми. У цьому документі представлено автономний і портативний хост мікроконтролера (PIC18F66K80). Цей хост здатний отримувати зображення з ПК через протокол Xmodem, зберігати його у зовнішній флеш-пам’яті, аналізувати шістнадцяткове представлення Intel збереженого зображення та програмувати цільовий контролер (ще один PIC18F66K80) за допомогою уніфікованого протоколу завантажувача. Зв'язок між хостом і метою здійснюється через мережу контролера (CAN). Основною перевагою цієї реалізації є можливість використання портативного та зручного хост-контролера для програмування окремих цілей [11].

Опис розробки та впровадження вбудованого завантажувача для ARM uCOS охоплює наступні аспекти: використання технології самоадаптивного завантаження, використання блоку керування пам’яттю (MMU), обробка переривань, пряма

установка векторів переривань, робоча з великими флеш-сторінками, програмуванням для UART тощо. У статті також наведено конкретний приклад, який ілюструє реальний вплив вбудованого завантажувача під час портування ARM uCOS [12].

Цей підхід використовує модульну структуру коду та спеціальні інструкції, щоб зменшити навантаження на міграцію, а також використовує макроси у файлі визначення для оновлень у реальному часі. Паралельно налаштовуються резервні копії області флеш-пам'яті для підвищення надійності. Розробка цього завантажувача на універсальній платформі PowerPC скорочує процес міграції більш ніж на 60% порівняно з U-Boot. Імовірність успішного запуску завантажувача з резервної області становить 92,5% і вище [13].

Розроблений для сімейства мікроконтролерів Atmel AVR, ATmega перевершує Optiboot з точки зору меншого розміру та високої швидкості, надаючи програмісту більше доступної флеш-пам’яті та скорочуючи час, необхідний для завантаження або оновлення програми мікроконтролера. Концепція завантажувача зіграла ключову роль у популяризації Arduino, усунувши потребу в спеціалізованих пристроях програмування на етапі використання. Однак початкова мета завантажувача полягала в тому, щоб надати кінцевому користувачеві або менш досвідченому обслуговуючому персоналу можливість оновлювати програму продукту або вбудованого пристрою [14].

З огляду на наведені вище приклади використання завантажувача, можна припустити, що цей компонент мікроконтролерних систем є досить гнучким і доступним інструментом для управління мікроконтролером. Завдяки наявності даного компонента в складі програмованих пристроїв в ролі менеджера для запису та виконання програм стає можливим контролювати вміст пам’яті пристрою. А завдяки наявності цього компонента стає можливим модифікувати його під потреби системи. Результатом використання такого інструменту може бути забезпечення часткової або повної реконфігурації пристроїв у складі вже розгорнутої системи.

Ідея системи, де всі вузли можна конфігурувати, спрямована на підвищення гнучкості системи. Можливість виконувати віддалену реконфігурацію через мережу дозволяє кооперувати вузли з можливістю реконфігурації одного вузла іншим за допомогою спеціальних програмних засобів або попередньо запрограмованих завантажувачів [7].

Таким чином, існують різні випадки використання Інтернету речей і додатків-завантажувачів. Можна виділити IoT Medicine як клас, який включає як виробництво обладнання, автоматизацію лабораторної діяльності, так і діагностику, хірургію та реабілітацію. Цей клас за своїм призначенням вимагає надійної та безперебійної роботи використовуваних систем, що відповідає темі дослідження. Іншим класом можна вважати застосування промислового IoT. Він може включати як додатки для моніторингу та контролю виробничих процесів, так і додатки для забезпечення безпеки та безпеки виготовленої продукції під час її зберігання та транспортування.

2. Аналіз методу реконфігурації та програмування

Аналіз реконфігурації програмного забезпечення показує, що підходи, які найкраще відповідають умовам обмежених ресурсів та потужностей спрямовані на підвищення швидкості та надійності системи за рахунок ускладнення її структури та переосмислення традиційних методів взаємодії та поведінки.

Перший підхід, який можна розглянути – це програмно-визначена мережа (Software-defined network). В його основі лежить ідея введення в класичну мережу взаємодії між пристроями арбітра, який має можливість регулювати трафік і перенаправляти потоки даних і обчислень на менш завантажені пристрої в режимі реального часу, тим самим оптимізуючи загальне навантаження на мережу і її учасників. Завдяки цьому значно підвищується життєздатність та швидкість мережі.

Інший підхід, який варто розглянути, – це мережа об'єктів(Network of Objects). Він представляє кожен пристрій мережі як інтелектуальний об'єкт з набором базових операцій і функцій для управління. Такий тип організації значно підвищує ремонтопридатність всієї системи завдяки тому, що кожен об'єкт може бути вилучений з системи без її повної зупинки і перезапуску. Крім того, внесення змін до системних налаштувань не вимагає оновлення системи, що виключає простої і збільшує масштабованість.

Ще один метод організації, на який варто звернути увагу, називається Edge-to-IoT. Він представляє кожного учасника мережі як вузол, який може бути переконфігурований іншим вузлом відповідно до вимог програми. Використовуючи властивості мікроконтролерів багаторазово переписувати виконувану програму, стає можливим переконфігурувати вузол, що містить цей контролер. Це дає можливість тонко налаштовувати пристрої, що використовуються мережею, і створювати на їх основі кооперацію для збалансованого обчислювального процесу.

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

Підсумовуючи результати аналізу, отримані дані були занесені до підсумкової таблиці 1.

Таблиця 1

Порівняння параметрів для трьох підходів до побудови.

Архітектура	Стратегія побудови	Переваги
Програмно-визначена мережа	Адаптація політик динамічної маршрутизації до стану мережі	– мінімізація затримок – оптимізація навантаження – доставка в режимі реального часу
Мережа об'єктів	Конфігурація та керування інтелектуальними пристроями як мережевими об'єктами	– зменшити затримку – забезпечення ремонтопридатності – забезпечення масштабованості
Edge-to-IoT	Зміна поведінки вузлів відповідно до системних вимог	– забезпечення автономності – можливість реконфігурації – спільна робота пристроїв

Таким чином, можна зробити висновок, що всі описані методи мають ряд схожих характеристик, таких як підвищення стійкості системи до збоїв і продовження терміну служби системи. Два з описаних методів вимагають значних модифікацій з боку системи, де вони застосовуються, як на рівні пристроїв, так і на рівні інфраструктури.

Порівняльний аналіз завантажувача. Для досягнення описаного ефекту підвищення якісних характеристик системи шляхом ускладнення її внутрішньої структури було розглянуто спосіб забезпечення функції реконфігурації окремих елементів системи. Було обрано метод із використанням завантажувача як засобу реконфігурації. У цьому випадку проводився пошук варіантів реалізації кастомного завантажувача для мікроконтролерів різних виробників.

Розглядався випадок використання модифікованого завантажувача для мікроконтролера S3C2440 виробництва Samsung. Незважаючи на заощадливо великий обсяг пам'яті даних мікроконтролера, для реалізації пристрою на його основі з можливістю перенастроювання за допомогою завантажувача знадобилося приблизно 30% від загального обсягу його вбудованої пам'яті. Таким чином, ця модифікація зменшила ресурс контролера на третину.

Для мікроконтролерів серії PIC18F66K80 від компанії-виробника Microchip ситуація виглядає набагато кращеПри використанні модифікованого завантажувача вдалося вмістити його в 6% від загального обсягу вбудованої пам'яті мікроконтролера. Слід зазначити, що контролер має істотно менший обсяг пам'яті.

У випадку мікроконтролерів серії ATMega виробництва корпорації Atmel можна помітити, що завдяки спільному розвитку цих контролерів кількість варіантів модифікації завантажувачів набагато більше, ніж у розглянутих раніше аналогів. Зважаючи на це, існує тенденція до того, що розмір завантажувача залежить від функцій, вбудованих у нього. Розглядаючи найпоширеніший варіант завантажувача, виявилося, що обсяг займаної ним пам'яті не перевищує 3% від загального обсягу вбудованої пам'яті.

На основі наведених вище даних була створена таблиця 2 для порівняння внутрішніх ресурсів розглянутих контролерів з вимогами, необхідними для модифікації.

Виходячи з результатів, слід зазначити, що всі описані реалізації завантажувача мають ряд недоліків, пов’язаних з особливостями мікроконтролера. В одному випадку завантажувач може бути реалізований з обмеженими можливостями, в іншому він може бути надлишковим для сфери застосування. Аналіз показав, що якість модифікації багато в чому залежить від використовуваного пристрою.

Таблиця 2

Порівняння параметрів завантажувачів для різних мікроконтролерів.

Виробник	Мікроконтролер	Пам'ять програм/даних, контакти	Вимоги до завантажувача
Samsung	S3C2440	47 KB/16 KB, 289 pins	16 KB програмної пам'яті (34%)
Microchip	PIC18F66K80	64 KB/3648 B, 64 pins	4 KB програмної пам'яті (6%)
Atmel	ATmega	32 KB/1KB, 32 pins	1 KB програмної пам'яті (3%)

3. Запропонований метод програмування завантажувача як проміжного програмного забезпечення

Окремим елементом може бути можливість використання проміжного рівня для організації виконання обчислень у складі таких завантажувачів. Іншими словами, це можна назвати проміжним рівнем у завантажувачі. Пропонується використовувати пресет у випадку, коли неможливо перепрошити завантажувач, використовуючи його ресурси для виконання операції. Це передбачає створення завантажувача з примітивним API для реалізації всього іншого. Ідея полягає в тому, що

під час розробки завантажувача має бути можливість переналаштувати його за допомогою деякого вищого рівня, деякого програмного інтерфейсу, який дозволяє виконувати ці команди в мікроконтролері, щоб, на основі цих команд у мікроконтролері, виконувати більш складні послідовності дій без прямого доступу до цього мікроконтролера. Також пропонується реалізувати ідею можливості переналаштування та перепрошивки мікроконтролера на основі таких рішень. У цьому випадку стає можливим здійснити дистанційне перепрограмування мікроконтролера за допомогою вбудованих API або певних попередньо встановлених можливостей у складі такого завантажувача, щоб здійснити перепрограмування на більш високому рівні модуля.

Прикладом цього може бути надсилання запиту від набору команд високого рівня до описаного завантажувача. Він може записати ці команди в EEPROM, а потім, отримавши повний набір послідовності команд, починаэ перегляд їх почергового виконання за допомогою вбудованих елементів і на основі цієї послідовності, відповідно до Essentially, перепрошити частини поведінки.

Іншим прикладом може бути використання команд в EEPROM як безпосередньої програми для виконання мікроконтролером. У цьому випадку існує можливість використання такого мікроконтролера або набору мікроконтролерів з основною ідеєю, коли такі, власне, спеціалізовані команди і завдання, які повинні бути реалізовані за допомогою цих мікроконтролерів, можуть бути записані безпосередньо не на флеш-пам'ять. , але до EEPROM мікроконтролера.

При цьому абсолютно всі вузли спочатку повинні бути прошиті за допомогою спеціалізованого для таких завдань завантажувача, котрий забезпечує набір таких викликів. За допомогою стандартних інтерфейсів взаємодії між мікроконтролерами, в тому числі віддалених, стаэ можливим безпосередньо реалізувати програми в кожен окремий вузол, щоб пізніше записати його в EEPROM.

У цьому випадку EEPROM може виступати у якості пам'яті для програми, і спочатку всі контролери можуть містити той самий завантажувач, але фактичний набір інструкцій, які потрібно виконати, може міститися в EEPROM. У цьому випадку сам завантажувач і програма, записана в контролер, можуть виступати в якості деякого базового механізму і основи для виконання інструкцій, записаних в EEPROM. При цьому існує заборона на використання більшого обсягу флеш-пам'яті в складі мікроконтролера для запису туди, в тому числі, цих програм.

API або базовий набір функцій, включений у такий завантажувач, може дозволити це зробити. У той же час ресурс перезапису також повинен дозволяти використовувати флеш-пам'ять для таких завдань. Забезпечення такого механізму значно підвищить гнучкість рішень на основі мікроконтролерів і забезпечить реконфігурацію на апаратному рівні.

4. Практичний приклад використання запропонованого методу програмування

На основі запропонованого методу програмування використання більш високого рівня абстракції зі змінними програмними модулями для програмування мікроконтролерів було створено практичний приклад системи інфрачервоного зв’язку на основі мікросхеми ATmega328P [15]. Цей пристрій є компонентом розумного будинку, що дозволяє дистанційно керувати різними пристроями по інфрачервоному каналу. Для зручності процесу додавання підтримки керування новою домашньою електронікою була реалізована можливість реконфігурації створеного компоненту. Використання текстового формату зв'язку з цим пристроєм через центральний координатор дозволяє виконувати оновлення вмісту EEPROM зі збереженим набором команд без перепрограмування безпосередньо в системі. Можливість комунікації між декількома екземплярами таких пристроїв реалізована за допомогою прототипу власного стабільного протоколу кодування повідомлень.

Для покращення стійкості в зашумленому середовищі та спрощення процесу декодування значення бітів кодується тривалістю між кінцями пакетів.

Існуючі мікроконтролерні рішення для зв'язку по інфрачервоному каналу передбачають можливість призначення набору команд на етапі компіляції проекту або з використанням програматора для зберігання даних з цим набором в EEPROM. Створена реалізація використовує фіксований набір команд у створеному завантажувачі для виконання призначення та оновлення набору команд з використанням послідовного інтерфейсу UART. Запропонована реалізація дозволяє модифікувати набір команд без підключення програматора до мікроконтролера.

Додаткова модифікація програми або завантажувача мікроконтролера дозволяє використовувати флеш-пам'ять для зберігання більшої кількості команд або задіяти мікросхеми без EEPROM.

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

Дискусія

Поточна робота демонструє підходи до гнучкого розширення і модернізації систем в умовах обмежених ресурсів та простору. Було оглянуто низку методів побудови систем з закладеною можливістю зміни внутрішньої структури відповідно встановленій задачі. Виконано попередню класифікацію розглянутих підходів.

Відповідно до отриманої попередньої класифікації було розглянуто низку практичних рішень щодо покращення експлуатаційних характеристик з використанням різних типів реконфігурації. Для розглянутих прикладів було виявлено ряд якісних характеристик, притаманних цим класам організації реконфігурованої частини систем. Також було дано їх первинний аналіз стосовно задачі збільшення терміну служби систем та обходу дефектів, які призводять систему до стану відмови.

Щодо аспектів кібербезпеки, підходи до реалізації доступу та контролю вузлів у реконфігурованих системах можна розглядати як важливе поле для дослідження, як і рівні абстракції керування та структурно-логічні специфікації.

Висновки

У роботі було проаналізовано інструменти, технології та методи реконфігурації компонентів IoT. В подальшому, враховуючи існуючі типи систем, які схильні до зміни внутрішньої структури під потреби системи, вони були класифіковані за рівнем організації їх реконфігурованої частини..

Для вирішення проблеми забезпечення функції реконфігурації систем на базі мікроконтролерів було проведено аналіз практичних рішень щодо модифікації вбудованих компонентів мікроконтролерів для управління та оновлення їх мікропрограмного забезпечення.

Виконана практична реалізація запропонованого методу реконфігурації з представленням завантажувача у вигляді набору API контролера, яка показує можливість поділу програми на інформаційну частину та механізми оновлення цієї частини. Інформаційна частина може бути розміщена як в EEPROM, так і у флеш-пам'яті. Для розглянутих прикладів виявлено низку кількісних характеристик, які демонструють залежність залучених ресурсів від його моделі та конфігурації. Також дається початковий аналіз з точки зору доступності та складності модифікації.

Основним внеском цієї роботи є запропонований метод програмування зі змінними програмними модулями для мікроконтролера та отримані результати конкурентного аналізу способів модифікації компонентів і особливостей завантажувачів.

Удосконалення способів зв’язку та реконфігурація компонентів IoT є основним напрямком подальших досліджень. У рамках подальшого розвитку теми планується вивчити аспекти кібербезпеки каналу реконфігурації. Планується вивчити тему існуючих загроз для систем IoT, проаналізувати уразливості для цих загроз, розглянути методи та засоби захисту.

Конфлікт інтересів

Автор заявляє, що немає конфлікту інтересів щодо цього дослідження, фінансового, особистого, авторського чи іншого, який міг би вплинути на дослідження та його результати, представлені в статті.

Фінансування

Дослідження проводилося без фінансової підтримки.

Доступність даних

Рукопис не має пов'язаних даних.

Використання засобів штучного інтелекту

Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

Автор прочитав та погодився з опублікованою версією рукопису.

Література

1. Vdovichenko, O. Analysis of Technologies for Reconfiguration of IoT Systems at Level of Software Modules and Bootloaders [Text] / O. Vdovichenko, & A. Perepelitsyn // Integrated Computer Technologies in Mechanical Engineering, ICTM 2023, Springer, Cham. – 2023. – vol. 996. DOI: 10.1007/978-3-031-60549-9_36.

2. Vdovichenko, O. Technologies for building systems of remote lining of communication lines: a practical example of implementation [Text] / O. Vdovichenko, & A. Perepelitsyn // Radioelectronic and Computer Systems. – 2021. – No. 2. – P. 31–38. DOI: 10.32620/reks.2021.2.03.

3. Вдовіченко, О. О. Організація взаємодії пристроїв з доступом в інтернет на основі мікроконтролерів із обмеженою кількістю ресурсів [Текст] / О. О. Вдовіченко, & А. Є. Перепелицин // Авіаційно-космічна техніка і технологія. – 2023. – № 6. – С. 76–85. DOI: 10.32620/aktt.2023.6.09.

4. Perepelitsyn, A. Service for communication of devices with internet access: analysis of technologies and method of creation [Text] / A. Perepelitsyn,

O. Vdovichenko, & V. Mikhalevskyi // Radioelectronic and Computer Systems. – 2023. – No. 4. – P. 197–208. DOI: 10.32620/reks.2023.4.14.

5. Plakhteyev, A. Edge computing for IoT: An educational case study [Text] / A. Plakhteyev, A. Perepelitsyn, & V. Frolov // Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT 2018. – 2018. – P. 130–133. DOI: 10.1109/DESSERT.2018.8409113.

6. Technologies of Embedded Systems Prototyping using Reconfigurable Nodes: Technical Solutions [Text] / A. Perepelitsyn, V. Duzhyi, O. Vdovichenko, & O. Zheltukhin // Proceedings 2022 IEEE 12th International Conference on Dependable Systems, Services and Technologies, DESSERT 2022. – 2022. – 6 p. DOI: 10.1109/DESSERT58054. 2022.10018581.

7. Метод дистанційної діагностики, перепрограмування і реконфігурації вузлів вбудованої системи [Текст] / О. О. Вдовіченко, А. Є. Перепелицин, В. І. Дужий, & О. В. Желтухін // Авіаційно-космічна техніка і технологія. – 2022. – № 6. – С. 66–75. DOI: 10.32620/aktt.2022.6.08.

8. Systematic Review on Internet of Things in Smart Livestock Management Systems [Text] / S. Terence, J. Immaculate, A. Raj, & J. Nadarajan // Sustainability Journal. – 2024. – No. 16(10). – P. 1-37. DOI: 10.3390/su16104073.

9. Development and Assessment of Internet of Things-Driven Smart Home Security and Automation with Voice Commands [Text] / P. Netinant, T. Utsanok, M. Rukhiran, & S. Klongdee // IoT Journal. – 2024. – No. 5(1). – P. 79-99. DOI: 10.3390/iot5010005.

10. Chen, S. Design optimization and implementation of bootloader in embedded system development [Text] / S. Chen, & L. Zhu-Ying // International Conference on Computer Science and Applications (CSA), IEEE, Wuhan, China. – 2015. – P. 151-156. DOI: 10.1109/CSA.2015.37.

11. Priyanka, P. S. Standalone portable host for unified bootloader in PIC devices using CAN interface [Text] / P. S. Priyanka, M. Gnanadas, & P. Supriya // 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE, Kharagpur, India. – 2020. – P. 1-4. DOI: 10.1109/ICCCNT49239.2020.9225574.

12. Bo, Q. Design of built-in boot loader for ARM uCOS [Text] / Q. Bo, & W. Zhaozhi // IEEE International Conference on Computer Science and Automation Engineering (CSAE), IEEE, Zhangjiajie, China. – 2012. – P. 429-433. DOI: 10.1109/CSAE.2012. 6272631.

13. Yanbo, K. Generic bootloader architecture based on automatic update mechanism [Text] / K. Yanbo, C. Jiaxu, & L. Bing // IEEE 3rd International Conference on Signal and Image Processing (ICSIP), IEEE, Shenzhen, China. – 2018. – P. 586–590. DOI: 10.1109/SIPROCESS. 2018.8600478.

14. Lewandowski, M. Dedicated AVR bootloader for performance improvement of prototyping process [Text] / M. Lewandowski, T. Orczyk, & P. Porwik // MIXDES - 24th International Conference "Mixed Design of Integrated Circuits and Systems, IEEE, Bydgoszcz, Poland. – 2017. – P. 553-557. DOI: 10.23919/MIXDES.2017. 8005274.

15. ATmega328P Data Sheet [Electronic resource] / Microchip Technology Inc., 2015. – 294 p. – Available at: https://ww1.micro-chip.com/downloads/en/ DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P _Datasheet.pdf. (accessed 19 October 2023).

References

1. Vdovichenko, O., & Perepelitsyn, A. Analysis of Technologies for Reconfiguration of IoT Systems at Level of Software Modules and Bootloaders. *Integrated Computer Technologies in Mechanical Engineering 2023 – ICTM 2023, Springer, Cham*, 2023, vol. 996. DOI: 10.1007/978-3-031-60549-9_36.

2. Vdovichenko, O., & Perepelitsyn, A. Technologies for building systems of remote lining of communication lines: a practical example of implementation. *Radioelectronic and Computer Systems*, 2021, no. 2, pp. 31-38. DOI: 10.32620/reks.2021.2.03.

3. Vdovichenko, O., & Perepelitsyn, A. Orhanizatsiya vzayemodiyi prystroyiv z dostupom v internet na osnovi mikrokontroleriv iz obmezhe-noyu kil′kistyu resursiv [Organization of communication of devices with internet access based on microcontrollers with limited hardware resources]. *Aviacijno-kosmicna tehnika i tehnologia – Aerospace technic and technology*, 2023, no. 6, pp. 76-85. DOI: 10.32620/aktt. 2023.6.09.

4. Perepelitsyn, A., Vdovichenko, O., & Mikhalevskyi, V. Service for communication of devices with internet access: analysis of technologies and method of creation. *Radioelectronic and Computer Systems*, 2023, no. 4, pp. 197-208. DOI: 10.32620/reks.2023. 4.14.

5. Plakhteyev, A., Perepelitsyn, A., & Frolov, V. Edge computing for IoT: An educational case study. *Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT 2018*, 2018, pp. 130-133. DOI: 10.1109/DESSERT.2018.8409113.

6. Perepelitsyn, A., Duzhyi, V., Vdovichenko, O., & Zheltukhin, O. Technologies of Embedded Systems Prototyping using Reconfigurable Nodes: Technical Solutions. *Proceedings 2022 IEEE 12th Inter-national Conference on Dependable Systems, Services and Technologies, DESSERT 2022*, 2022. 6 p. DOI: 10.1109/DESSERT58054.2022.10018581.

7. Vdovichenko, O., Perepelitsyn, A., Duzhyi, V., & Zheltukhin, O. Method of remote diagnostics, reprogramming and reconfiguration of nodes of embedded system. *Aviacijno-kosmicna tehnika i tehnologia – Aerospace technic and technology*, 2022, no. 6, pp. 66-75. DOI: 10.32620/aktt.2022.6.08.

8. Terence, S., Immaculate, J., Raj, A., & Nadarajan, J. Systematic Review on Internet of Things in Smart Livestock Management Systems. *Sustainability Journal*, 2024, no. 16(10), pp. 1-37. DOI:

ISSN 1727-7337 (print)
ISSN 2663-2217 (online)

10.3390/su16104073.

9. Netinant, P., Utsanok, T., Rukhiran, M., & Klongdee S. Development and Assessment of Internet of Things-Driven Smart Home Security and Automation with Voice Commands. *IoT Journal*, 2024, no. 5(1), pp. 79-99. DOI: 10.3390/iot5010005.

10. Chen, S., & Zhu-Ying, L. Design optimization and implementation of bootloader in embedded system development. *International Conference on Computer Science and Applications (CSA), IEEE, Wuhan, China*, 2015, pp. 151-156. DOI: 10.1109/CSA.2015.37.

11. Priyanka, P. S., Gnanadas, M., & Supriya, P. Standalone portable host for unified bootloader in PIC devices using CAN interface. *11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), IEEE Kharagpur, India*, 2020, pp. 1-4. DOI: 10.1109/ICCCNT49239.2020.9225574.

12. Bo, Q., & Zhaozhi, W. Design of built-in boot loader for ARM uCOS. *IEEE International Conference on Computer Science and Automation Engineering (CSAE), IEEE, Zhangjiajie, China*, 2012, pp. 429–433. DOI: 10.1109/CSAE.2012.6272631.

13. Yanbo, K., Jiaxu, C., & Bing, L. Generic bootloader architecture based on automatic update mechanism, IEEE *3rd International Conference on Signal and Image Processing (ICSIP), IEEE, Shenzhen, China*, 2018, pp. 586-590. DOI: 10.1109/SIPROCESS.2018.8600478.

14. Lewandowski, M., Orczyk, T., & Porwik, P. Dedicated AVR bootloader for performance improvement of prototyping process, *MIXDES - 24th International Conference "Mixed Design of Integrated Circuits and Systems", IEEE, Bydgoszcz, Poland*, 2017, pp. 553-557. DOI: 10.23919/MIXDES.2017.8005274.

15. *ATmega328P Data Sheet*. Microchip Technology Inc., 2015. 294 p. Available at: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. (accessed 19.10.2023).

Надійшла до редакції 20.02.2024, розглянута на редколегії 15.06.2024

ANALYSIS OF TECHNOLOGIES FOR RECONFIGURATION OF INTERNET OF THINGS SYSTEMS AT LEVEL OF SOFTWARE MODULES AND BOOTLOADERS

Oleksandr Vdovichenko

The subject of study and research in this article are the technology of programming and interaction of nodes, as well as individual small-sized components of embedded Internet of Things (IoT) systems. The **goal** is to increase the flexibility of IoT system components by simplifying the reconfiguration process of individual nodes and components. The **task** is: to analyze existing approaches to building configurable systems, analyze and describe existing solutions that implement joint approaches; to analyze the tools for implementing the selected approaches; to describe the method of building a living system based on the analysis approach; and to provide a technical example of the proposed programming method. According to the tasks, the following **results** were obtained. Possible ways of reconfiguration of IoT components are analyzed. Methods of reconfiguration at the network level and partial node reconfiguration are considered. Methods to ensure the flexibility of IoT components are analyzed and classified. A comparative analysis of the summary parameters of the loaders was carried out. The possibility of using software components as immutable blocks of code to modify the microcontroller program as a method of reconfiguration was analyzed. The idea of using variable software modules to represent the microcontroller program at a higher level of abstraction with a modified bootloader is thus proposed. Elements of a method for reprogramming variable software modules of microcontrollers as middleware are proposed. **Conclusions.** According to a review of the existing types of systems, they were classified according to the level of organization of their reconfigured parts. According to the obtained preliminary classifications, some architectural solutions were considered from the point of view of improving operational characteristics. The main advantages for each type of organization of the reconfigurable parts of the systems are highlighted. An analysis of practical solutions regarding the modification of the built-in components of microcontrollers for managing and updating their firmware was conducted. The practical significance of this study is the proposal of a method of programming with variable software modules for microcontrollers and the obtained results of a competitive analysis of methods of modifying components and loader characteristics.

Keywords: software reconfiguration, reconfigurable nodes, Fog computing, bootloader, Edge-IoT, replaceable software modules.

Вдовіченко Олександр Олександрович – асп., асист. каф. комп'ютерних систем, мереж і кібербезпеки, Національний аерокосмічний університет ім. М. Є. Жуковського «Харківський авіаційний інститут», Харків, Україна.

Oleksandr Vdovichenko – PhD Student, Assistant at the Computer Systems, Networks and Cybersecurity Department, National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine,
e-mail: o.vdovichenko@csn.khai.edu, ORCID: 0000-0001-8695-1752, Scopus Author ID: 58090577500.