УДК 004.91
DOI 10.36910/10.36910/6775-2313-5352-2024-25-13
Кулакевич О. Р., Грудецький Р. Я., Сацик В. О., Маркіна Л. М.
Луцький національний технічний університет, м. Луцьк, Україна

РОЗРОБКА IOT: IOS - ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ОБЛАДНАННЯМ

У статті розглянуто основні протоколи взаємодії мобільних застосунків із технічним устаткуванням, розроблено мобільний застосунок, який забезпечує взаємодію з технічним устаткуванням за допомогою сучасних протоколів передачі даних. Основними протоколами, використаними у програмі, є Bluetooth, Wi-Fi, MQTT та Bonjour, які дозволяють реалізувати керування пристроями, обмін даними та налаштування параметрів підключення. Проведено аналіз способів комунікації програмного забезпечення з сенсорами, зовнішніми пристроями та програмованими системами для забезпечення стабільного зв'язку.

Ключові слова: *мобільні застосунки, технічне устаткування, протоколи взаємодії, Bluetooth, Wi-Fi, NFC, MQTT, сенсори, датчики, зовнішні пристрої, автоматизація, інтернет речей (IoT), API, інтеграція, передача даних, керування пристроями.*

Постановка проблеми. У сучасному світі розвиток мобільних застосунків та технічного устаткування є важливим фактором автоматизації та оптимізації процесів у різних галузях – від побутових систем до промислових підприємств. Однак ефективна взаємодія між мобільними застосунками та технічним устаткуванням потребує використання надійних і оптимізованих протоколів передачі даних та керування пристроями.

Проблема полягає у забезпеченні стабільного зв'язку, швидкої обробки даних та сумісності програмного забезпечення з різними типами технічного устаткування, включаючи сенсори, зовнішні пристрої та програмовані системи. При цьому необхідно враховувати:

– обмежені ресурси мобільних пристроїв (пам'ять, енергоспоживання, обчислювальна потужність).

– вибір оптимального протоколу для конкретного завдання (наприклад, Bluetooth для близького зв'язку, Wi-Fi для обміну великими даними, MQTT для IoT-рішень).

– високі вимоги до безпеки передачі даних, особливо у промислових та критичних системах.

– швидкість інтеграції та масштабованість рішень у великих системах автоматизації.

Таким чином, виникає необхідність у глибокому аналізі існуючих протоколів взаємодії мобільних застосунків з технічним устаткуванням, виявленні їх переваг та обмежень для розробки ефективних рішень.

Аналіз останніх досліджень і публікацій. Останні роки значна кількість досліджень присвячена оптимізації взаємодії мобільних застосунків із технічним устаткуванням, особливо в контексті Інтернету речей (IoT), автоматизації та цифрових технологій. Розглянемо ключові напрями та результати досліджень у цій сфері. У працях Smith et al. (2022 р.) досліджено ефективність Bluetooth Low Energy (BLE) для інтеграції мобільних застосунків із сенсорами в системах моніторингу здоров'я. Зазначено низьке енергоспоживання BLE, але вказано на його обмежену пропускну здатність, що дозволило використовувати BLE з незначною кількістю даних, яка може передавати в обмеженому радіусі, але одночасно це дозволило використати BLE у пристроях, які не вимагають постійно електроживлення. Робота Higgins et al. (2023 р.) висвітлює використання протоколу MQTT для інтеграції мобільних застосунків з «розумними» системами. MQTT показав високу надійність у передачі даних з мінімальним навантаженням на мережу. Даний протокол на сьогодні дозволяє інженерам створювати чимало програмного забезпечення для керування техустаткуванням, зокрема надавати тим чи інших пристроям (до прикладу верстатам) «розумних функцій». У публікації Rahman et al. (2023 р.) висвітлено питання захисту даних під час взаємодії мобільних застосунків із технічним устаткуванням через Wi-Fi та Bluetooth.Останні дослідження демонструють значний прогрес у розробці та вдосконаленні протоколів взаємодії мобільних застосунків із технічним устаткуванням. Основна увага приділяється енергоефективності, швидкості передачі даних, сумісності та безпеці комунікацій, що створює фундамент для подальшого розвитку систем автоматизації та IoT-рішень.

Мета роботи полягає у вивченні актуальних та прогресивних протоколів, таких як Bluetooth, Wi-Fi, NFC, MQTT, та їх застосуванню у різних сценаріях взаємодії з сенсорами, зовнішніми пристроями та системами автоматизації, а також розробці власного універсального

рішення на базі мобільного застосунку, який матиме можливість взаємодіяти з усіма протоколами передачі даних.

Викладення основного матеріалу. Розвиток мобільних технологій дає можливість програмним застосункам взаємодіяти з різноманітним технічним устаткуванням, що відкриває широкі можливості для автоматизації як побутових, так і промислових завдань. Основні протоколи взаємодії – Bluetooth, Wi-Fi, NFC та MQTT – забезпечують передачу даних і керування устаткуванням через мобільні застосунки [4]. Використовуючи мову програмування Swift, можна створювати рішення, які інтегруються зі станками, сенсорами та зовнішніми пристроями для контролю, моніторингу та управління. Одним із найпоширеніших сценаріїв є взаємодія з Bluetooth-пристроями [4]. Наприклад, сучасні станки на виробництві можуть бути обладнані модулями Bluetooth Low Energy (BLE), що дозволяють передавати стан сенсорів, лічильники виконаних операцій та інші дані. У Swift для роботи з BLE використовується фреймворк CoreBluetooth, який забезпечує можливість сканування пристроїв, встановлення з'єднання та обміну даними. Приклад реалізації зображено на рисунку 1.

```swift
import CoreBluetooth

class BLEManager: NSObject, CBCentralManagerDelegate, CBPeripheralDelegate {
    var centralManager: CBCentralManager!
    var discoveredPeripheral: CBPeripheral?
    let targetDeviceName = "ProductionMachine123"

    override init() {
        super.init()
        centralManager = CBCentralManager(delegate: self, queue: nil)
    }

    func centralManagerDidUpdateState(_ central: CBCentralManager) {
        if central.state == .poweredOn {
            print("Bluetooth увімкнено, починається сканування...")
            centralManager.scanForPeripherals(withServices: nil, options: nil)
        } else {
            print("Bluetooth вимкнено.")
        }
    }

    func centralManager(_ central: CBCentralManager, didDiscover peripheral: CBPeripheral, advertisementData: [String: Any], rssi RSSI: NSNumber) {
        if let name = peripheral.name, name == targetDeviceName {
            print("Знайдено пристрій: \(name)")
            discoveredPeripheral = peripheral
            centralManager.stopScan()
            centralManager.connect(peripheral, options: nil)
        }
    }

    func centralManager(_ central: CBCentralManager, didConnect peripheral: CBPeripheral) {
        print("Підключено до \(peripheral.name ?? "невідомого пристрою")")
        peripheral.delegate = self
        peripheral.discoverServices(nil)
    }
}
```

Рисунок 1 – Приклад використання фреймворку CoreBluetooth мовою Swift для під'єднання та зчитування інформації з технічного станка, який обладнаний BLE

Код на рисунку 1 демонструє підключення до BLE-пристрою, сканування його характеристик та встановлення зв'язку. У випадку зі станками це може бути передача даних про температуру, швидкість обертання шпинделя чи статус виконання операції. Використовуючи BLE, можна налаштувати мобільний застосунок для контролю ключових параметрів обладнання та надсилання сигналів для корекції роботи у реальному часі.

Іншим важливим способом комунікації є використання Wi-Fi для обміну великими обсягами даних між мобільним застосунком і технічним устаткуванням. Приклад реалізації зображено на рисунку 2. У цьому контексті застосунки часто використовують REST API для взаємодії з серверами або пристроями, що підтримують протоколи HTTP/HTTPS. Наприклад, станок на заводі може передавати журнали операцій, помилки та параметри роботи на сервер, звідки їх може отримати застосунок.

```swift
import Foundation

func fetchMachineStatus() {
    let url = URL(string: "http://factory-server.com/api/machine/status")!
    let task = URLSession.shared.dataTask(with: url) { data, response, error in
        if let error = error {
            print("Помилка: \(error.localizedDescription)")
            return
        }

        if let data = data, let jsonResponse = try? JSONSerialization.jsonObject(with: data) {
            print("Статус станка: \(jsonResponse)")
        }
    }
    task.resume()
}
```

Рисунок 2 – Приклад отримання статусу верстату за допомогою REST API взаємодії

Запит може використовуватися для отримання параметрів продуктивності або діагностики станка. Наприклад, якщо температура двигуна станка перевищує критичний рівень, застосунок може надіслати сигнал технічному персоналу про необхідність обслуговування [1, 2, 3].

На практиці, інтеграція з обладнанням може охоплювати такі приклади: ЧПУ-станки, роботизовані маніпулятори, конвеєрні лінії та автоматизовані верстати. Наприклад, станок з ЧПУ (числовим програмним управлінням) може надавати дані про програму, що виконується, точність операцій та технічний стан. Мобільний застосунок, у свою чергу, може віддалено оновлювати програму станка або налаштовувати його параметри.

Таким чином, використовуючи можливості Swift та сучасні протоколи взаємодії, можна створити потужні мобільні застосунки, що інтегруються з технічним устаткуванням. Це відкриває широкі можливості для моніторингу, управління та автоматизації як на невеликих виробництвах, так і у великих промислових системах.

Додавши протокол MQTT, який забезпечує швидкий обмін повідомленнями у реальному часі, що критично важливо для моніторингу стану станка, таких як поточна температура, оберти двигуна або виявлення критичних помилок. Застосунок через MQTT може миттєво отримувати повідомлення про збої в роботі чи надмірне нагрівання станка й реагувати на них, наприклад, зупиняючи обладнання чи сповіщаючи оператора. На рівні коду Swift це дозволяє поєднати дві архітектури — періодичне оновлення даних через REST-запити та обмін короткими повідомленнями через MQTT [5].

Для прикладу, REST API може використовуватися для завантаження журналів роботи станка за день. Виклик здійснюється через HTTP-запит, і застосунок отримує JSON-дані з усіма логами операцій. У той час MQTT з’єднання працює паралельно для отримання реального часу статусу сенсорів. Якщо температура станка перевищує 90°C, MQTT-повідомлення негайно надсилається застосунку, і оператор бачить відповідне сповіщення з вимогою провести перевірку або зупинити роботу обладнаннях [5].

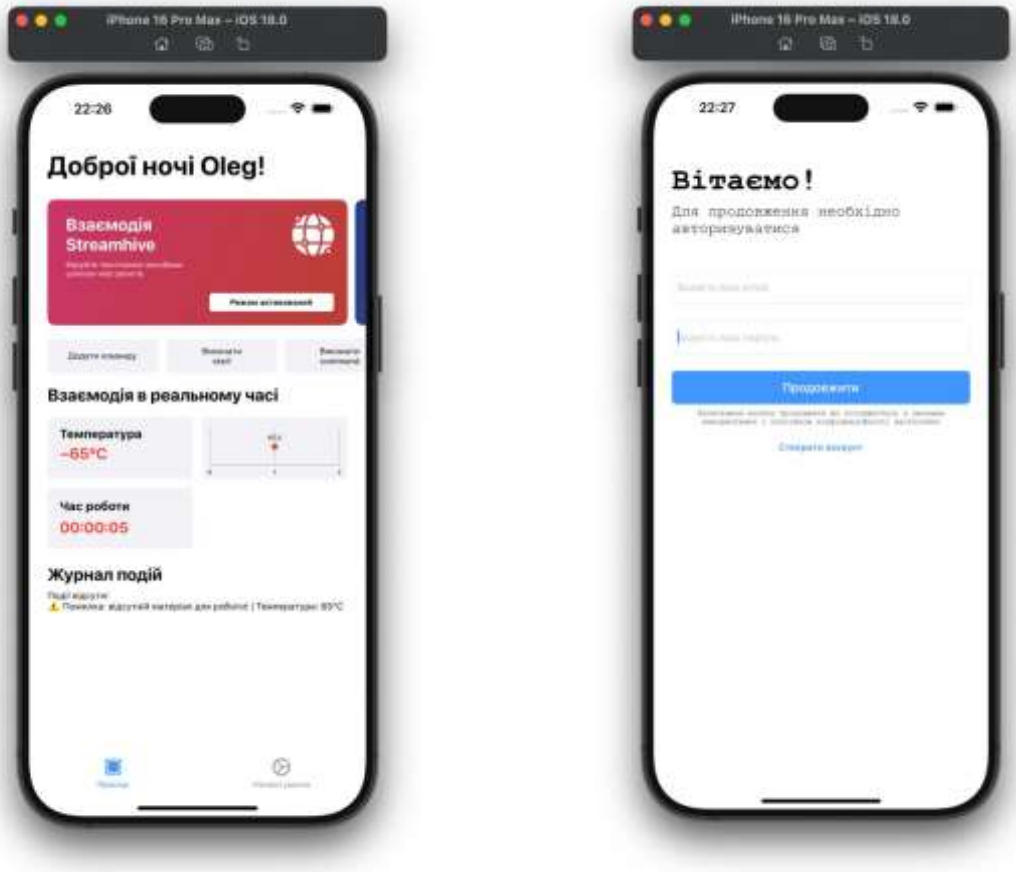
Уявімо ситуацію, коли технічний устаткуванням обладнаний датчик температури, підключений через MQTT. У Swift ми підписуємося на топік «factory/sensors/temperature». Коли застосунок отримує повідомлення зі значенням температури, воно обробляється миттєво у відповідному делегаті. Якщо значення перевищує порогове, застосунок автоматично публікує команду на топік «factory/commands» зі значенням «STOP_MACHINE». Таке управління дозволяє ефективно реагувати на критичні ситуації без затримок. З іншого боку, якщо оператор у застосунку хоче надіслати команду для зміни швидкості обертання двигуна, REST API використовується для відправлення даних конфігурації на сервер, де вони синхронізуються з обладнанням у відповідний час.

Це поєднання REST API та MQTT дозволяє мобільному застосунку забезпечувати як точний моніторинг стану технічного устаткування, так і гнучке управління з високою швидкістю реагування на події. Технологічно застосунок оптимізує передачу даних, зменшуючи навантаження на мережу та покращуючи загальну продуктивність системи.

Враховуючи вище описане, розглянемо практичне застосування протоколів передачі даних і власне мобільного застосунку, який завдяки своїм технічним можливостям і реалізації конкретних рішень зможе спілкувати зрозумілою мовою із технічним устаткуванням.

Для розробки такого застосунку нам потрібно передбачити просту і зрозумілу архітектуру проекту, яка буде легко читатися та може буде розширена згодом, такою архітектурою було обрано саме VIPER (VIEW-INTERACTOR-PRESENTER-ENTITY-ROUTER). На базі цієї архітектури ми розгортаємо систему авторизації, використовуючи електронну пошту нашого користувача і його пароль, завдяки цьому система набуде потрібного захисту, аби сторонні особи не отримали доступу до робочих приладів без відома відповідальної особи.

Після того як користувач буде авторизований в системі ми пропонуємо йому зручний інтерфейс користувача і систему розширених налаштувань, де потрібно пройти базову конфігурацію, як от налаштувати IP-адресу нашого приладу, який буде отримувати команди. Інша частина застосунку – це і є взаємодія по відповідних протоколах, яких є 4 і всі вони є уніфіковані в своєму роді. Розглянемо один з них – як от Bonjour або ж взаємодія по Wi-Fi. Цей протокол широко використовується у сфері IoT і притаманний виключно пристроям компанії Apple. Даний протокол передбачає наявність пристрою, який є периферійним, буде транслювати свою наявність і отримувати команди від центру управління, що в нашому випадку є мобільним застосунком (рисунок 3: а, б).



а) б)

Рисунок 3 – Інтерфейс мобільного застосунку

Для тестування було розроблено програму симулятор для MacOS, яка і є у нашому випадку периферійним пристроєм, до якого приєднається iOS застосунок і буде передавати команди. Власне надалі MacBook може бути приєднаний до будь-якого апаратного верстата і управляти його діями у вигляді так званого сервера. У будь-якому разі наявність MacBook не обов’язкова для такої взаємодії. Все, що було зроблено – це виключно для демонстрації технічної можливості такої реалізації. Отже, на рисунку 4 чітко відображено принцип взаємодії мобільного застосунку із симулятором за протоколом Bonjour.

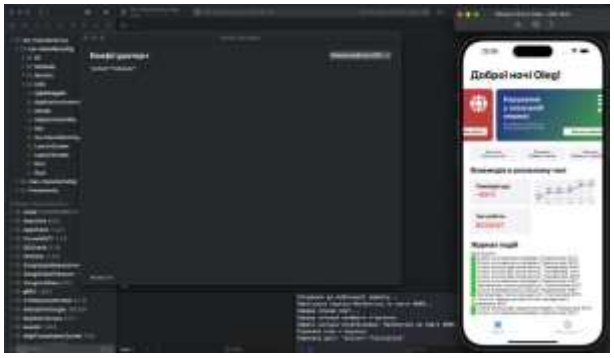


Рисунок 4 – Процес тестування взаємодії по протоколу Bonjour

Висновоки. У цій роботі було проаналізовано можливості взаємодії мобільних застосунків із технічним устаткуванням на основі сучасних протоколів передачі даних, таких як REST API для роботи через Wi-Fi та MQTT для обміну повідомленнями у реальному часі та розроблено мобільний застосунок для тестування таких взаємодій. Поєднання цих підходів дозволяє створити ефективні програмно-апаратні рішення, що забезпечують високу продуктивність, стабільність та гнучкість системи автоматизації. REST API є оптимальним вибором для передачі великих обсягів даних, таких як звіти, журнали або конфігураційні параметри устаткування, тоді як MQTT забезпечує миттєву доставку критично важливих повідомлень і дозволяє реагувати на зміни стану устаткування у реальному часі.

Реалізація такого підходу у мобільних застосунках на платформі iOS за допомогою Swift дозволяє легко інтегрувати функціонал для моніторингу й управління станками, сенсорами або іншими пристроями. Приклад із керуванням температурою станка через MQTT демонструє, як можна швидко обробляти події та приймати рішення, тоді як REST API забезпечує централізовану синхронізацію та аналіз даних. Це відкриває широкі можливості для автоматизації виробництва, впровадження IoT-рішень та оптимізації технічних процесів.

Таким чином, використання REST API та MQTT у мобільних застосунках є ефективним рішенням для інтеграції з технічним устаткуванням, що забезпечує баланс між швидкістю передачі даних, стабільністю з’єднання та можливістю масштабування системи для великих проєктів. Це підхід, який може стати фундаментом для подальшого розвитку автоматизованих систем як у побутовій, так і промисловій сфері.

Інформаційні джерела

1.Higgins et al. The Use of MQTT in IoT-Based Smart Systems. *International Journal of Industrial Automation*, 2023. Vol. 12. No. 4. P. 123-130.

2. Kumar R., Li J., Zhou X. Comparative Analysis of Wi-Fi and MQTT for Industrial Automation. *Journal of Industrial Networking Technologies*, 2022. Vol. 8. No. 3. P. 201-210.

3. Rahman M., Gupta S., Petrov D. Data Security in MQTT and REST API for IoT Applications. *IEEE Transactions on Industrial Informatics*, 2023. Vol. 19. No. 2. P. 450-457.

4. Smith T., Adams J., Chen W. Efficiency of Bluetooth Low Energy (BLE) in Industrial Monitoring Systems. *Sensors and Actuators Journal*, 2022. Vol. 11. No. 5. P. 78-85.

5. Li J., Zhou M., Wang S. Smart Industrial Communication with REST API and MQTT Protocols. *Advances in Automation Engineering*, 2021. Vol. 10. No. 1. P. 67-74.

Kulakevich O., Grudetskyi R., Satsyk V., Markina L.
Lutsk National Technical University, Lutsk, Ukraine

IOT DEVELOPMENT: IOS - APPLICATION FOR EQUIPMENT MANAGEMENT

The article considers the main protocols of interaction of mobile applications with technical equipment, developed a mobile application that provides interaction with technical equipment using modern data transmission protocols. The main protocols used in the program are Bluetooth, Wi-Fi, MQTT and Bonjour, which allow you to implement device management, data exchange and connection settings. An analysis of software communication methods with sensors, external devices and programmable systems to ensure stable communication was carried out.

Keywords: *mobile applications, hardware, interaction protocols, Bluetooth, Wi-Fi, NFC, MQTT, sensors, sensors, external devices, automation, Internet of Things (IoT), API, integration, data transfer, device management.*