Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра САП

Лабораторна робота №5
На тему: **« Розробка та програмування власної інтелектуальної вбудованої системи: Формування завдань і вимог.»**

З дисципліни: «Програмування інтелектуальних вбудованих систем»

Виконали:
студенти групи ПП-31
Гаврилюк Назар
Герчаківський Данило
Заліщук Ольга
Зайцева Юліана
Данильчук Богдан
Прийняв:
Колесник К. К.

Львів – 2025р.

Мета роботи: Ознайомитися з основами розробки інтелектуальних вбудованих систем, визначити вимоги до системи, сформулювати ключові завдання, та виконати початкове проектування на базі сучасних мікроконтролерів.

Тема проєкту: Система моніторингу вузлів локальної мережі на базі ESP32 з автоматичним відстеженням доступності пристроїв.

Суть: Інтелектуальна IoT-система на базі ESP32, яка безперервно контролює доступність вузлів у локальній мережі. Мікроконтролер підключається до мережі через Wi-Fi або Ethernet, автоматично виявляє активні пристрої, періодично пінгує їх для перевірки доступності та передає результати на веб-платформу або мобільний додаток для аналізу та прийняття управлінських рішень.

1. Функціональні вимоги

1.1.Збір даних

- Сканування мережі: Система повинна автоматично виявляти всі активні вузли в заданій підмережі (наприклад, 192.168.1.0/24) з точністю до 1 секунди на хост.
- Отримання мережевих параметрів: Необхідно фіксувати IP-адреси, MAC-адреси та, за можливості, імена хостів (через mDNS/NetBIOS) для кожного виявленого пристрою.
- Моніторинг доступності: Система має використовувати протокол ICMP Echo Request (ping) для перевірки онлайн-статусу кожного вузла з інтервалом 5-10 секунд.
- Вимірювання затримки мережі: Відстеження часу відгуку (RTT - Round Trip Time) для кожного пристрою в мілісекундах.
- Детекція змін у мережі: Виявлення нових пристроїв, що підключилися, або відключення існуючих хостів у реальному часі.

1.2.Обробка та логіка

- Формування таблиці вузлів: Контролер повинен збирати "сирі" дані з мережі та створювати структуровану таблицю, аналогічну ARP-таблиці, з полями: IP, MAC, hostname, status, last_seen.
- Багаторівнева класифікація стану: Система повинна розрізняти як мінімум 3 стани вузла:
    - Онлайн (ping успішний, RTT < 100 мс)
    - Повільний відгук (RTT > 100 мс, але < 1000 мс)
    - Офлайн (ping timeout або недоступний)

1.3.Інтерфейси та взаємодія

- Локальна індикація: Світлодіодний індикатор стану системи (зелений - працює, червоний - втрачено з'єднання з мережею).
- Бездротова передача даних: Система повинна передавати дані на сервер через Wi-Fi (стандарт 802.11 b/g/n) або Ethernet у форматі JSON через HTTP/MQTT.
- Веб-інтерфейс користувача: Відображення таблиці всіх вузлів з їх статусами, графіків uptime, часових рядів доступності та спливаючих сповіщень (alerts) при падінні критичних хостів.
- Мобільний додаток: Додаток на Flutter з підтримкою push-нотифікацій при зміні статусу вузлів.

2. Аналоги для прикладу

- IoT Network Scanner: Використовує ESP32 для сканування мережі через ARP-запити, зберігає дані у локальній SD-карті та відображає статус на OLED-дисплеї. Підтримує веб-інтерфейс через embedded HTTP server.
- Ping Monitor Arduino: Простий моніторинг пінгу на Arduino Uno + Ethernet Shield з відправкою статусу на Blynk через REST API. Обмежена функціональність через низьку продуктивність ATmega328P.

3. Нефункціональні вимоги

3.1.Часові параметри та продуктивність

Система повинна опитувати вузли мережі з таким інтервалом, щоб забезпечити актуальний стан без надмірного навантаження на мережу та сам контролер. Для підмережі типу /24 повний цикл перевірки всіх адрес не має тривати довше кількох хвилин, а реакція на падіння важливого вузла повинна відбуватися практично в реальному часі (затримка не більше кількох секунд).

3.2.Обмеження по пам’яті та ресурсах мікроконтролера

Прошивка повинна вміщуватися в доступну флеш-пам'ять мікроконтролера (стандартно 4 МБ для ESP32) та ефективно використовувати оперативну пам'ять (SRAM).

3.3.Споживання енергії

Пристрій розрахований на живлення від стандартного джерела 5 В (наприклад, USB-порт чи зарядний адаптер), а піковий струм при активному Wi-Fi/ETH та передачі даних не повинен виходити за межі, прийнятні для типового настільного або мережевого обладнання. Це дозволяє використовувати систему в серверних, офісах чи навчальних аудиторіях без додаткових вимог до живлення.

3.4.Надійність роботи та умови експлуатації.

У випадку втрати з’єднання з мережею контролер не повинен “падати”: він продовжує працювати, періодично пробуючи перепідключитися та фіксуючи свій стан локально (наприклад, через індикацію світлодіодом). Система орієнтована на тривалу безперервну роботу в приміщенні в стандартному діапазоні температур, характерному для офісних і навчальних приміщень.

4. Вибір апаратної платформи

Для реалізації системи було обрано мікроконтролер ESP32, оскільки він поєднує в собі достатню продуктивність, вбудований мережевий стек та підтримку бездротового підключення.

Технічні характеристики ESP-32:

- Процесор: 32-бітний двоядерний Xtensa LX6.
- Тактова частота: до 240 МГц.
- Пам'ять: 520 КБ SRAM (оперативна), 4 МБ Flash.
- Інтерфейси: GPIO, I2C, SPI, UART, PWM.
- Wi-Fi: 802.11 b/g/n (до 150 Мбіт/с).
- Ethernet: Підтримка через зовнішній PHY (LAN8720, TLK110).
- Напруга живлення: 3.3В логіка, 5В через USB.

На відміну від простіших плат (наприклад Arduino Uno), ESP32 не потребує додаткових модулів для роботи з TCP/IP, а порівняно з Raspberry Pi є дешевшим та енергоощаднішим, що важливо для компактних вбудованих систем.

5. Обґрунтування архітектури без фізичних сенсорів

Цей проєкт не потребує фізичних датчиків. Вся інформація збирається програмно через мережеві протоколи та інтерфейси.

Використовувані протоколи:

- ARP (Address Resolution Protocol) — для отримання MAC-адрес пристроїв у локальній мережі.
- ICMP Echo Request/Reply (ping) — для перевірки доступності вузлів та вимірювання затримки.
- mDNS (Multicast DNS) — для отримання людино-читабельних імен хостів.
- DHCP сканування — для виявлення нових пристроїв що підключаються.

Мережеві інтерфейси ESP32:

- Wi-Fi 802.11 b/g/n — основний спосіб підключення до мережі
- Ethernet (опціонально) — через зовнішній PHY модуль для підвищеної стабільності

6. Архітектура системи та логіка роботи

Архітектура побудована за модульним принципом та складається з трьох основних рівнів: Рівень мережі → Рівень обробки → Рівень представлення.

6.1.Модульна структура системи

Систему розбито на незалежні модулі, кожен з яких виконує конкретну задачу:

Модуль 1: Підключення до мережі

ESP32 встановлює з'єднання з локальною мережею через Wi-Fi або Ethernet. Після успішного підключення мікроконтролер отримує IP-адресу через DHCP або використовує статичну конфігурацію. Передбачено механізм автоматичного перепідключення у випадку втрати з'єднання з інтервалом спроб кожні 10 секунд.

Модуль 2: Виявлення вузлів

Виконується сканування заданої підмережі (наприклад, 192.168.1.0/24) для виявлення активних хостів. Система створює внутрішню таблицю вузлів, аналогічну ARP-таблиці, та збирає інформацію про кожен пристрій: IP-адресу, MAC-адресу, hostname (через mDNS). Дані зберігаються у структурованому вигляді в SRAM мікроконтролера з полями: IP, MAC, hostname, status, last_seen, RTT.

Модуль 3: Моніторинг доступності

Система періодично надсилає ICMP Echo Request (ping) до кожного виявленого вузла з інтервалом 5-10 секунд. Вимірюється час відгуку (RTT - Round Trip Time) та оновлюється статус кожного хоста в реальному часі. Для автоматизації процесу використовуються таймери FreeRTOS, що забезпечують стабільність роботи без блокувань основного циклу програми.

Модуль 4: Обробка та класифікація

На основі результатів пінгу система визначає стан кожного вузла та розраховує показник доступності (Availability Score). Дані формуються у JSON-пакети для подальшої передачі на сервер або клієнтську частину.

Модуль 5: Передача даних на клієнт

Система підтримує два варіанти відображення даних:

Веб-інтерфейс: ESP32 надсилає HTTP POST запити на бекенд сервера (FastAPI на Python), який обробляє дані та зберігає їх у базі. Фронтенд (HTML/CSS/JavaScript) відображає таблицю вузлів з їх актуальними статусами. Передбачена можливість додавання авторизації користувачів для розмежування доступу до різних мереж.

Мобільний додаток: Зв'язок здійснюється через MQTT або WebSocket для забезпечення оновлень у реальному часі. Додаток на Flutter приймає дані та візуалізує стан мережі з можливістю відправки push-нотифікацій при падінні критичних хостів.

6.2. Алгоритм роботи системи

Робота системи розділена на дві основні фази:

Фаза 1: Виявлення вузлів

- Підключення до Wi-Fi або Ethernet мережі;
- Отримання IP-адреси через DHCP;
- Сканування всієї підмережі (наприклад, 192.168.1.1 - 192.168.1.254);
- Для кожної активної адреси: надсилання ARP-запиту для отримання MAC-адреси та mDNS запиту для визначення hostname;
- Створення внутрішньої таблиці вузлів у пам'яті з структурою: {IP, MAC, hostname, status, last_seen, RTT};
- Перехід до фази безперервного моніторингу.

Фаза 2: Безперервний моніторинг

У циклі з інтервалом 5-10 секунд:

- Для кожного вузла в таблиці надсилається ICMP Echo Request (ping)
- Очікування відповіді з таймаутом 1000 мс
- При отриманні відповіді: вимірювання RTT, оновлення status = "Online", розрахунок Availability Score
- При таймауті: встановлення status = "Offline", Availability Score = 0
- Формування JSON-пакету з усіма даними
- Відправка через HTTP POST або MQTT на сервер
- Повторення циклу

Багатопоточність через FreeRTOS:

Для забезпечення стабільної роботи використовується двоядерна архітектура ESP32:

- Task 1 (Core 0): Виконує пінгування вузлів та оновлення внутрішньої таблиці
- Task 2 (Core 1): Відповідає за відправку даних на сервер через HTTP або MQTT

Такий розподіл задач запобігає блокуванню основного циклу програми та забезпечує плавну роботу системи навіть при великій кількості хостів.

6.3. Алгоритм оцінки стану вузла

Кожен вузол отримує показник Availability Score (AS) на основі результатів пінгу та часу відгуку. Класифікація здійснюється за наступними критеріями:

- AS = 100 (Відмінно): Ping успішний, RTT < 50 мс

- AS = 75 (Добре): Ping успішний, 50 ≤ RTT < 100 мс
- AS = 50 (Повільно): Ping успішний, 100 ≤ RTT < 500 мс
- AS = 25 (Нестабільно): Ping timeout, але last_seen < 30 секунд
- AS = 0 (Офлайн): Ping timeout, last_seen > 30 секунд

Візуальна індикація на веб-інтерфейсі:

Кожному значенню Availability Score відповідає колір для швидкої візуальної оцінки стану мережі:

- Зелений (AS = 100): Вузол онлайн, відмінна швидкість відгуку
- Синій (AS = 75): Вузол онлайн, добра доступність
- Жовтий (AS = 50): Вузол онлайн, але повільний відгук (можливі проблеми з мережею)
- Помаранчевий (AS = 25): Нестабільне з'єднання, вузол періодично пропадає
- Червоний (AS = 0): Вузол офлайн, ping timeout

6.4. Схема потоків даних

Обробка даних у системі відбувається послідовно через наступні етапи:

1. Збір даних: ESP32 безперервно виконує ping-запити до всіх вузлів у таблиці моніторингу.
2. Локальна обробка: Мікроконтролер аналізує результати кожного пінгу, розраховує Availability Score та оновлює внутрішню таблицю без участі зовнішнього сервера. Це забезпечує автономність системи.
3. Автономний режим: Навіть при відсутності інтернет-з'єднання система продовжує локальний моніторинг та зберігає дані в пам'яті для подальшої синхронізації.
4. Передача даних: Сформовані JSON-пакети надсилаються на сервер через HTTP POST запити або MQTT протокол залежно від налаштувань системи.
5. Візуалізація: Веб-інтерфейс або мобільний додаток отримує дані від сервера та відображає актуальний стан всіх вузлів у зручному табличному вигляді з можливістю фільтрації та пошуку.

Така архітектура забезпечує швидку реакцію на зміни в мережі з затримкою менше 2 секунд від моменту падіння вузла до відображення інформації на клієнтській стороні. Система дозволяє вести детальну статистику доступності для подальшого аналізу та виявлення проблемних пристроїв у мережі.