Система передавання даних на основі інтерфейсу REDIS та програмної платформи Circle

Гурська Вероніка
ІВТВ-21

Актуальність та проблема

- Потреба у швидкому передаванні даних у реальному часі
- Обмеження повноцінних ОС (затримки, фонові служби)
- Необхідність стабільного й передбачуваного RTT
- Redis як високошвидкісне сховище в LAN

Мета та завдання роботи

- Створення клієнта Redis без використання традиційної ОС
- Спілкування через протокол RESP
- Реалізація підтримки типів JSON
- Вимога: RTT ≤ 35 мс

Technologies & Tools

Апаратна частина

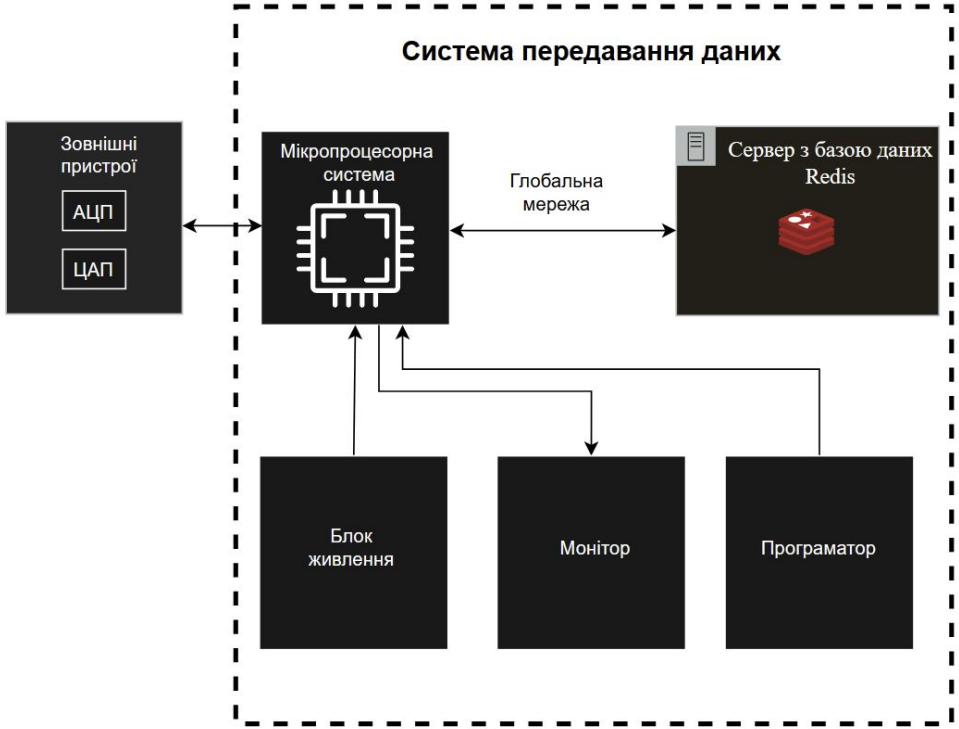
- Raspberry Pi 3
- Ethernet (Wired LAN)

Програмна частина

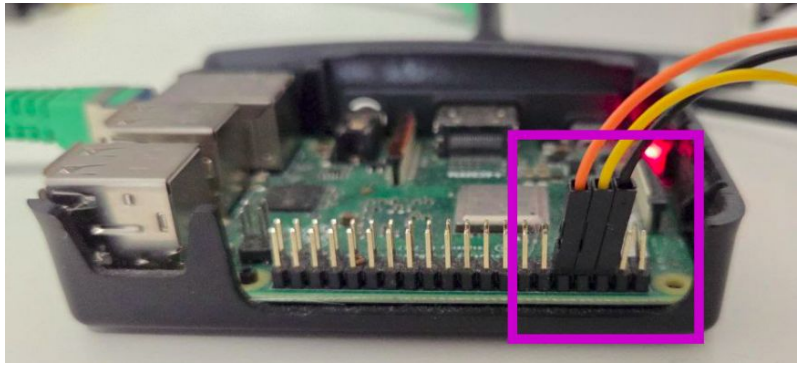
- C++
- Python
- Circle(with stdlibs)
- Redis (as a cloud data store)
- JSON (via nlohmann::json)

python™

Загальна архітектура



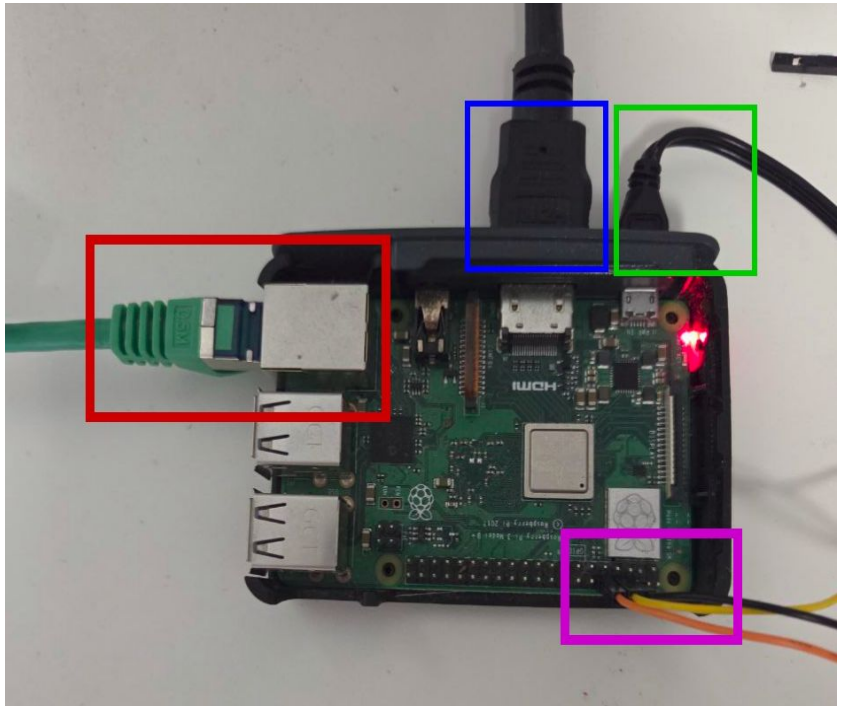
Налаштування апаратної частини



```
        Profile name tui-ethernet
              Device enx98ded005001b (98:DE:D0:05:00:1B)

= ETHERNET                                                  <Show>
= 802.1X SECURITY                                           <Show>

= IPv4 CONFIGURATION <Manual>                               <Hide>
           Addresses 192.168.1.100/24          <Remove>
                     <Add...>
             Gateway 255.255.255.0
         DNS servers <Add...>
     Search domains <Add...>
```



Circle implementation

- Використано вбудований IP та TCP стек середовища Circle
- Безпосереднє керування сокетами через клас CSocket
- Налаштовано статичну IP-адресу, DNS та шлюз
- Встановлено пряме TCP-з’єднання з сервером Redis

JSON формат

- JSON - JavaScript Object Notation.


```
{
    "device": "mass comparator",
    "message_type": "20",
    "message_user": "1",
    "message": "get_state(): Counter = 6398"
}
```

nlohmann vs hiredis

- Обмеження bare-metal середовища виключають використання зовнішніх залежностей або важких бібліотек
- Бібліотека nlohmann::json є header-only та має малу вагу
- Бібліотека hiredis не придатна для bare-metal середовищ
- nlohmann::json легко інтегрується з власним RESP-парсером

REDIS

- Redis — високошвидкісна in-memory база даних типу «ключ–значення»
- Працює як сервер на локальному комп’ютері
- Для обміну використано протокол RESP

*3\r\n

$3\r\nSET\r\n

$4\r\nkey1\r\n

$5\r\nvalue\r\n

Implementation Details

- AUTH, SET, GET, JSON.GET, JSON.SET
- Робота з потоками даних

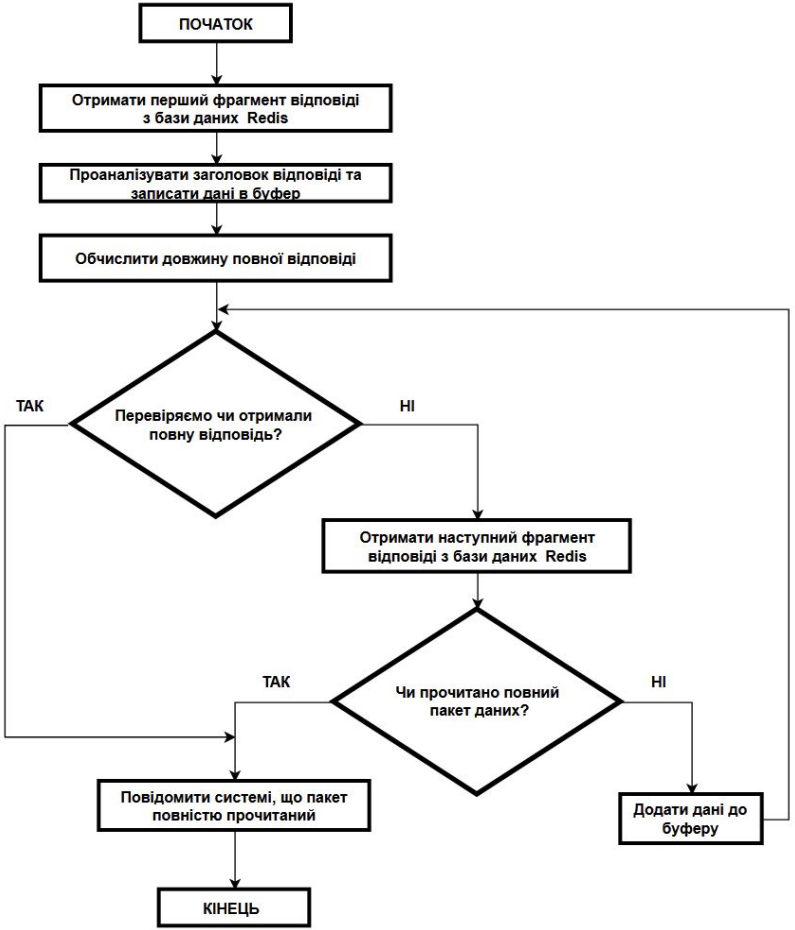
```
command.Format("*2\r\n$3\r\nGET\r\n$%u\r\n%s\r\n",
key.GetLength(), key.c_str());
```

```
command.Format("*3\r\n$3\r\nSET\r\n$%u\r\n%s\r\n$%u\r\n%s\r\n",
                key.GetLength(), key.c_str(),
                value.GetLength(), value.c_str());
```

Challenges & Solutions

- TCP/IP не гарантує, що вся відповідь буде отримана за один Receive
- Довгі RESP-повідомлення можуть надходити частинами

Реалізовано алгоритм пофрагментного читання RESP-відповідей з контролем повної довжини пакета.



Client

- Python-скрипт імітує роботу зовнішнього клієнта
- Підключення до Redis здійснюється за допомогою бібліотеки redis-py
- Виконується безперервне зчитування даних із Redis Streams

Conclusions

Implemented:

- Redis management functions(AUTH, SET, GET, JSON.SET, JSON.GET, XADD)
- JSON type support
- TCP/IP communication
- Loop-based receiving logic for large/partial data
- Python script

Learned:

- C++
- Raspberry Pi 3
- Transmission protocol