**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

A picture containing text, vector graphics, clipart

Description automatically generatedA picture containing graphics, clipart, logo, symbol

Description automatically generated

**BÁO CÁO ĐỒ ÁN**

HỆ THỐNG NHÚNG MẠNG KHÔNG DÂY

**Tên Đồ Án: Triển khai truyền dữ liệu sức khỏe từ xa thông qua hệ thống nhúng**

**NT131.P13**

**Group: 15**

**Sinh viên thực hiện:**

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên** | **MSSV** |
| 1 | Lê Ngọc Kiều Anh | 22520047 |
| 2 | Phùng Việt Bắc | 22520089 |
| 3 | Trần Phước Đại | 22520184 |

**Giảng viên hướng dẫn:** Lê Trung Quân

**Report:** 5/8/2024 – 10/11/2024

**LỜI CẢM ƠN**

Trước hết, chúng em muốn bày tỏ lòng biết ơn sâu sắc đến Lê Trung Quân, người đã dành cho chúng em không ít thời gian quý báu, hướng dẫn và giúp đỡ chân thành trong suốt quá trình thực hiện đồ án. Sự sẻ chia và sự hướng dẫn tỉ mỉ của Thầy đã là nguồn động viên lớn lao, giúp chúng em vượt qua các khó khăn và hoàn thành đề tài một cách đầy đủ.

Với mong muốn học hỏi và áp dụng kiến thức vào thực tế trong lĩnh vực lập trình ứng dụng mạng và hệ thống nhúng, nhóm chúng em đã lựa chọn đề tài "Triển khai hệ thống truyền dữ liệu sức khỏe sử dụng OLSRd trên Tiny Core." Mục tiêu của đề tài là xây dựng hệ thống thu thập và truyền tải thông tin sức khỏe, sử dụng giao thức OLSR triển khai trên TinyCore để thiết lập mạng ad-hoc nhằm đảm bảo dữ liệu từ các thiết bị nhúng được gửi đến client một cách nhanh chóng và ổn định. Qua quá trình nghiên cứu và phát triển, chúng em tự hào rằng đồ án đã hoàn thành với sự nỗ lực không ngừng.

Mặc dù đã cố gắng hết sức trong quá trình nghiên cứu và thực hiện, chúng em nhận thức được rằng đồ án vẫn có thể tồn tại những hạn chế và thiếu sót. Chúng em rất mong nhận được những góp ý quý báu từ Thầy để giúp hoàn thiện đề tài tốt hơn.

Cuối cùng, chúng em xin kính chúc Thầy luôn mạnh khỏe và tiếp tục sứ mệnh truyền đạt kiến thức, kinh nghiệm quý báu cho các thế hệ sinh viên tiếp theo.

TP. HCM, ngày 20 tháng 10 năm 2024

Sinh viên thực hiện

**LÊ NGỌC KIỀU ANH**

**PHÙNG VIỆT BẮC**

**TRẦN PHƯỚC ĐẠI**

**MỤC LỤC**

[I. GIỚI THIỆU CHUNG 7](#_Toc182174017)

[**1. Giới thiệu 7**](#_Toc182174018)

[**2. Phát biểu vấn đề 7**](#_Toc182174019)

[**3. Phạm vi và giới hạn 7**](#_Toc182174020)

[**4. Công cụ triển khai 7**](#_Toc182174021)

[**4.1. TinyCore 7**](#_Toc182174026)

[**4.2. OLSRd 8**](#_Toc182174027)

[**4.3. USB Wi-Fi chuẩn N TL-WN722N 8**](#_Toc182174028)

[**4.4. Thiết bị mạch nhúng 9**](#_Toc182174029)

[**4.4.1. Kit WiFi BLE ESP32 9**](#_Toc182174030)

[**4.4.2. Cảm biến nhịp tim và oxy trong máu MAX30102 9**](#_Toc182174031)

[**4.5. Arduino IDE (Integrated Development Environment) 10**](#_Toc182174032)

[II. TỔNG QUAN CHỦ ĐỀ 10](#_Toc182174033)

[**1. MANET 10**](#_Toc182174034)

[**1.1. Mạng MANET (Mobile Ad-hoc Network) 10**](#_Toc182174035)

[**1.2. Đặc điểm của mạng MANET 10**](#_Toc182174036)

[**2. OLSR 11**](#_Toc182174037)

[**2.1. Định nghĩa 11**](#_Toc182174038)

[**2.2. Cấu trúc các gói tin 11**](#_Toc182174039)

[**2.2.1. HELLO Messages 11**](#_Toc182174040)

[**2.2.2. Topology Control (TC) Messages 11**](#_Toc182174041)

[**2.2.3. Multiple Interface Declaration (MID) Messages 12**](#_Toc182174042)

[**2.2.4. Host and Network Association (HNA) Messages 12**](#_Toc182174043)

[**3. Cách thức hoạt động: 12**](#_Toc182174044)

[III. PHƯƠNG PHÁP NGHIÊN CỨU 13](#_Toc182174045)

[**1. Đối tượng nghiên cứu 13**](#_Toc182174046)

[**1.1. Cấu trúc của gói tin OLSR 13**](#_Toc182174047)

[**1.2. Thuật toán chọn đường đi tối ưu 13**](#_Toc182174048)

[**1.3. Quá trình trao đổi thông tin liên kết 13**](#_Toc182174049)

[**2. Quy trình nghiên cứu: 14**](#_Toc182174050)

[**2.1. Mô hình triển khai: 14**](#_Toc182174051)

[**2.2. Cách thức triển khai 14**](#_Toc182174052)

[IV. PHÂN TÍCH, DEMO 16](#_Toc182174053)

[**1. Phân tích mã nguồn OLSRd 16**](#_Toc182174054)

[**1.1. Hàm olsr\_openlog 16**](#_Toc182174055)

[**1.2. Các hàm checksum cấu hình 16**](#_Toc182174056)

[**1.3. Hàm print\_version 17**](#_Toc182174057)

[**1.4. Kiểm tra quyền root 17**](#_Toc182174058)

[**1.5. Tải cấu hình từ file 17**](#_Toc182174059)

[**1.6. Xử lý đối số CLI 18**](#_Toc182174060)

[**1.7. Kiểm tra tính hợp lệ cấu hình 18**](#_Toc182174061)

[**1.8. Tạo socket ioctl 18**](#_Toc182174062)

[**1.9. Tạo socket netlink (Linux) 19**](#_Toc182174063)

[**1.10. Tạo socket định tuyến trên các hệ thống BSD và macOS 19**](#_Toc182174064)

[**1.11. Hệ thống gateway (Linux) 19**](#_Toc182174065)

[**1.12. Giao diện NIIT (Linux) 20**](#_Toc182174066)

[**1.13. Thiết lập bộ đếm thời gian TC rỗng 20**](#_Toc182174067)

[**1.14. Chuyển tiếp IP và vô hiệu hóa chuyển hướng 20**](#_Toc182174068)

[**1.15. Bộ phân tích parser 21**](#_Toc182174069)

[**1.16. Bộ xuất định tuyến 21**](#_Toc182174070)

[**1.17. Số thứ tự thông điệp 22**](#_Toc182174071)

[**1.18. Khởi tạo mạng 22**](#_Toc182174072)

[**1.19. Khởi tạo cơ sở dữ liệu giao diện mạng 22**](#_Toc182174073)

[**1.20. Khởi tạo các bảng định tuyến 23**](#_Toc182174074)

[**1.21. Tải các plugin 23**](#_Toc182174075)

[**1.22. Thiết lập các quy tắc với mức độ ưu tiên 24**](#_Toc182174076)

[**1.23. Bắt đầu scheduler 24**](#_Toc182174077)

[**2. Phân tích kết quả mô hình triển khai 24**](#_Toc182174078)

[**2.1. Truyền dữ liệu sức khỏe 24**](#_Toc182174079)

[**2.2. Gửi mail thông báo tới Client 26**](#_Toc182174080)

[**3. Demo 26**](#_Toc182174081)

[V. TỔNG KẾT 27](#_Toc182174082)

[**1. Thuận lợi 27**](#_Toc182174083)

[**2. Khó khăn 27**](#_Toc182174084)

[**3. Kết quả 27**](#_Toc182174085)

[**4. Kết luận 27**](#_Toc182174086)

[**5. Phân công, đánh giá 28**](#_Toc182174087)

[VI. TÀI LIỆU THAM KHẢO 29](#_Toc182174088)

**DANH MỤC TỪ VIẾT TẮT**

OLSR: Optimized Link State Routing protocol

OLSRd: Optimized Link State Routing daemon

MANET: Mobile Ad-hoc Network

RAM: Random Access Memory

LAN: Local Area Network

BLE: Bluetooth Low Energy

IoT: Internet of Things

SpO2: Saturation of Peripheral Oxygen

SMTP: Simple Mail Transfer Protocol

HTTP / HTTPS: Hypertext Transfer Protocol / Hypertext Transfer Protocol SSL

ICMP: Internet Control Message Protocol

CONFIG: Configuration

MPR: Multi-Point Relay

MID: Multiple Interface Declaration

HNA: Host and Network Association

FIB: Forwarding Information Base

IBSS: Independent Basic Service Set

POSIX: Portable Operating System Interface

CLI: Command Line Interface

IOCTL: Input Output Control

**DANH MỤC HÌNH ẢNH**

[Hình 1: Hình ảnh TL-WN722N 8](#_Toc182138122)

[Hình 2: Hình ảnh ESP32 và cảm biến MAX30102 9](#_Toc182138123)

[Hình 3: Logo của Arduino IDE 10](#_Toc182138124)

[Hình 4: MANET Topology 10](#_Toc182138125)

[Hình 5: Cấu trúc gói tin HELLO 11](#_Toc182138126)

[Hình 6: Cấu trúc gói tin TC 11](#_Toc182138127)

[Hình 7: Cấu trúc gói tin MID 12](#_Toc182138128)

[Hình 8: Cấu trúc gói tin HNA 12](#_Toc182138129)

[Hình 9: Topology 14](#_Toc182138130)

[Hình 10: Hàm olsr\_openlog 16](#_Toc182138131)

[Hình 11: Các hàm checksum cấu hình 16](#_Toc182138132)

[Hình 12: Hàm print\_version 17](#_Toc182138133)

[Hình 13: Kiểm tra quyền root 17](#_Toc182138134)

[Hình 14: Tải cấu hình từ file config 17](#_Toc182138135)

[Hình 15: Xử lý đối số CLI 18](#_Toc182138136)

[Hình 16: Kiểm tra tính hợp lệ cấu hình 18](#_Toc182138137)

[Hình 17: Tạo socket ioctl 18](#_Toc182138138)

[Hình 18: Tạo socket netlink 19](#_Toc182138139)

[Hình 19: Tạo socket định tuyến trên BSD, macOS 19](#_Toc182138140)

[Hình 20: Hệ thống gateway (Linux) 19](#_Toc182138141)

[Hình 21: Giao diện NIIT (Linux) 20](#_Toc182138142)

[Hình 22: Bộ đếm thời gian TC rỗng 20](#_Toc182138143)

[Hình 23: Chuyển tiếp IP và vô hiệu hóa chuyển hướng trên kernel linux 21](#_Toc182138144)

[Hình 24: Bộ phân tích parser 21](#_Toc182138145)

[Hình 25: Bộ xuất định tuyến 21](#_Toc182138146)

[Hình 26: Tạo số thứ tự thông điệp 22](#_Toc182138147)

[Hình 27: Khởi tạo hệ thống mạng OLSRd 22](#_Toc182138148)

[Hình 28: Khởi tạo cơ sở dữ liệu giao diện mạng 22](#_Toc182138149)

[Hình 29: Khởi tạo các bảng định tuyến 23](#_Toc182138150)

[Hình 30: Tải các plugin 23](#_Toc182138151)

[Hình 31: Thiết lập các quy tắc với mức độ ưu tiên 24](#_Toc182138152)

[Hình 32: Vòng lặp scheduler chính 24](#_Toc182138153)

[Hình 33: Sơ đồ đường đi của dữ liệu 25](#_Toc182138154)

[Hình 34: ESP32 gửi dữ liệu tới node triển khai OLSR gần nhất 25](#_Toc182138155)

[Hình 35: Chuyển tiếp dữ liệu đến node gateway của mạng client 26](#_Toc182138156)

[Hình 36: Chuyển dữ liệu tới client 26](#_Toc182138157)

# 

# GIỚI THIỆU CHUNG

## Giới thiệu

Ngày nay, thế giới ngày càng phát triển, con người ngày càng bận hơn bao giờ. Họ không có quá nhiều thời gian để quan tâm đến sức khỏe đặc biệt là những nhân viên văn phòng, nhân viên lập trình. Vấn đề sức khỏe đối với họ dường như không quan trọng và dễ bị phớt lờ. Một chút sự thay đổi trong dao động nhịp tim bình thường cũng có thể là dấu hiệu sức khỏe suy giảm. Chính vì vậy các thiết bị cảm biến đo nhịp tim ra đời giúp cho người dùng có thể tự theo dõi sức khỏe tại nhà mà không tốn quá nhiều thời gian.

## Phát biểu vấn đề

Không phải ai cũng có thể dễ dàng sử dụng, thao tác với các thiết bị này. Chẳng hạn như với những người lớn tuổi chưa tiếp cận nhiều đến các công nghệ hiện đại, việc sử dụng các mạch cảm biến này có thể gây nhiều khó khăn. Nếu như một gia đình muốn theo dõi sức khỏe từ xa của người thân, điều đó khá phức tạp và khó có thể thực hiện. Chính vì các lý do đó, nhóm chúng em quyết định chọn đề tài **“Triển khai truyền dữ liệu sức khỏe từ xa thông qua hệ thống nhúng”**. Với đề tài này, vấn đề truyền dữ liệu từ xa sẽ được giải quyết, có thể phát triển để áp dụng vào cuộc sống thực tiễn.

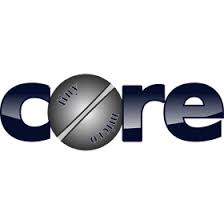
## Phạm vi và giới hạn

* + - * + Chỉ dùng cảm biến nhịp tim để thu thập dữ liệu sức khỏe, không phải là cả thiết bị theo dõi sức khỏe, có độ chính xác không tuyệt đối.
        + Không có quá nhiều mạng LAN khác nhau vì không sử dụng đến Router.
        + Số lượng thiết bị giới hạn (có 3 thiết bị dùng làm trung tâm cho giao thức OLSR và 1 thiết bị khách).
        + Sử dụng môi trường MANET, môi trường internet.

## Công cụ triển khai



### TinyCore

TinyCore là một bản phân phối Linux siêu nhẹ, được thiết kế để sử dụng rất ít tài nguyên hệ thống với mục tiêu tạo ra một hệ điều hành tối giản, phù hợp cho các hệ thống có phần cứng yếu hoặc yêu cầu nhanh chóng khởi động và chạy mà không cần nhiều tài nguyên.

Đặc điểm:

* Có dung lượng nhỏ: khoảng 16MB, giúp tiết kiệm không gian lưu trữ.
* Nhanh chóng: Hệ điều hành TinyCore khởi động và chạy rất nhanh vì không yêu cầu các cài đặt phức tạp, mà có thể chạy trực tiếp từ RAM.
* Linh hoạt: TinyCore có thể mở rộng dễ dàng, cho phép người dùng thêm các ứng dụng và gói phần mềm từ kho lưu trữ của nó.

### OLSRd

**A black and white penguin with yellow feet

Description automatically generated**OLSRd (Optimized Link State Routing daemon) là một phần mềm thực hiện giao thức OLSR trên các hệ thống thực tế. Đây là một ứng dụng mã nguồn mở chạy trên hệ điều hành Linux cho phép các thiết bị sử dụng giao thức OLSR để thực hiện định tuyến trong mạng ad-hoc.

OLSRd xử lý các chức năng của giao thức OLSR, như phát các gói tin Hello để nhận biết các nút lân cận, xây dựng bảng định tuyến qua cơ chế MPR, từ đó tự động cập nhật và duy trì bảng định tuyến toàn mạng dựa trên các gói tin Hello và Topology Control (TC), giúp các thiết bị luôn biết sẵn đường đi ngay khi có yêu cầu truyền dữ liệu.

### USB Wi-Fi chuẩn N TL-WN722N



Hình 1: Hình ảnh TL-WN722N

* USB WiFi adapter TP-Link TL-WN722N là một adapter không dây phổ biến, hỗ trợ chuẩn 802.11n và thường được sử dụng trong các ứng dụng như tạo kết nối mạng trên các thiết bị không có WiFi tích hợp, có khả năng hỗ trợ chế độ monitor mode (giám sát gói tin).
* Các đặc điểm chính:
* **Chuẩn WiFi**: Hỗ trợ chuẩn 802.11n với tốc độ lên đến 150Mbps, hoạt động trên băng tần 2.4GHz.
* **Ăng-ten rời**: Có ăng-ten 4dBi có thể tháo rời, giúp tăng phạm vi thu sóng và có thể thay thế khi cần thiết.

**Hỗ trợ chế độ monitor và packet injection**: Một số phiên bản của TL-WN722N (phiên bản v1) được hỗ trợ tốt trong chế độ monitor và khả năng packet injection, giúp người dùng phân tích lưu lượng mạng.

**Tương thích đa nền tảng**: Tương thích với nhiều hệ điều hành, bao gồm Windows, Linux, và macOS, nhưng cần chú ý đến phiên bản driver.

**Hỗ trợ mạng ad-hoc:** USB WiFi adapter TP-Link TL-WN722N (phiên bản v1) có khả năng hỗ trợ mạng ad-hoc, cho phép các thiết bị kết nối trực tiếp với nhau mà không cần thông qua một điểm truy cập (access point).

### Thiết bị mạch nhúng

#### Kit WiFi BLE ESP32

Là một module phát triển dựa trên vi điều khiển ESP32, nổi bật với khả năng kết nối Wi-Fi và Bluetooth Low Energy (BLE). Với chip ESP32 của hãng Espressif, nó mang đến một hệ thống đơn-chip mạnh mẽ với bộ xử lý kép (dual-core) và tích hợp bộ nhớ trong lớn, thích hợp cho các ứng dụng IoT. Kit này được đánh giá cao trong việc phát triển các ứng dụng mạng không dây, kết nối cảm biến, điều khiển từ xa, và các ứng dụng BLE.



Hình 2: Hình ảnh ESP32 và cảm biến MAX30102

#### Cảm biến nhịp tim và oxy trong máu MAX30102

Là một module đo nhịp tim và nồng độ oxy trong máu (SpO2) phổ biến, được sử dụng rộng rãi trong các ứng dụng y tế và thiết bị đeo thông minh. Cảm biến này do hãng Maxim Integrated phát triển và được thiết kế để cung cấp dữ liệu chính xác về nhịp tim và SpO2 thông qua phương pháp quang học.

### Arduino IDE (Integrated Development Environment)



Hình 3: Logo của Arduino IDE

Trình soạn thảo mã bằng ngôn ngữ lập trình dựa trên C/C++, giúp viết, kiểm tra và biên dịch mã thành ngôn ngữ máy sau đó tải mã lên bo mạch Arduino qua cổng USB. Arduino IDE hỗ trợ hầu hết các bo mạch Arduino (như Uno, Mega, Nano, ESP32, ESP8266,...) giúp người dùng linh hoạt trong việc lựa chọn phần cứng.

# TỔNG QUAN CHỦ ĐỀ

## MANET



Hình 4: MANET Topology

### Mạng MANET (Mobile Ad-hoc Network)

Là một loại mạng di động tự phát, trong đó các thiết bị di động kết nối và giao tiếp với nhau mà không cần đến cơ sở hạ tầng cố định như router hoặc điểm truy cập trung gian. Các thiết bị trong mạng MANET có thể tự động phát hiện, kết nối và định tuyến dữ liệu qua các thiết bị khác trong mạng**.**

### Đặc điểm của mạng MANET

* *Tính linh động cao:* Các thiết bị trong mạng MANET thường là các thiết bị di động, có thể di chuyển, gia nhập hoặc rời khỏi mạng bất kỳ lúc nào.
* *Tự tổ chức:* Mạng MANET không cần cơ sở hạ tầng cố định, các thiết bị trong mạng tự động phát hiện và kết nối với nhau.
* *Định tuyến động:* đường đi của dữ liệu cũng thay đổi liên tục, đòi hỏi các giao thức định tuyến thích ứng để đảm bảo dữ liệu được chuyển đến đúng nơi.
* *Khả năng mở rộng:* Mạng MANET có khả năng mở rộng, nhưng việc quản lý và bảo mật trở nên khó khăn khi số lượng thiết bị tăng cao.

1. **OLSR**
   1. **Định nghĩa**

OLSR (Optimized Link State Routing) là một giao thức định tuyến tối ưu hóa cho mạng di động. Giao thức này thuộc nhóm "proactive routing" - định tuyến chủ động, luôn duy trì bảng định tuyến cập nhật để giúp các gói dữ liệu có thể được truyền ngay lập tức khi có yêu cầu.

* 1. **Cấu trúc các gói tin**
     1. **HELLO Messages**

Các gói tin "HELLO" được gửi định kỳ để phát hiện các nút lân cận và xác định các liên kết trực tiếp. Mỗi nút gửi các gói Hello này để duy trì thông tin về các nút xung quanh và chọn ra các nút làm MPR.

**MPR(N):** là tập hợp các nút lân cận (cách 1 nút) được nút N chọn để phát lại các gói tin quảng bá từ N, nhằm đảm bảo đến được tất cả các nút lân cận (cách 2 nút).

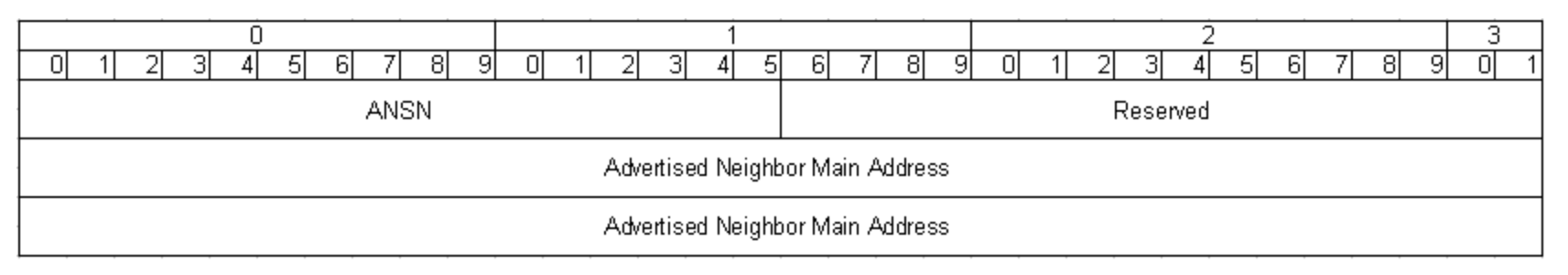
A close-up of a document

Description automatically generated

Hình 5: Cấu trúc gói tin HELLO

* + 1. **Topology Control (TC) Messages**

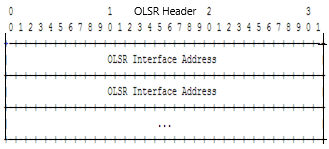
Các gói TC được gửi bởi các nút MPR nhằm phát quảng bá thông tin định tuyến đến toàn bộ mạng. Gói TC chứa thông tin về các liên kết và cho phép các nút duy trì bảng định tuyến đầy đủ.

**

Hình 6: Cấu trúc gói tin TC

* + 1. **Multiple Interface Declaration (MID) Messages**

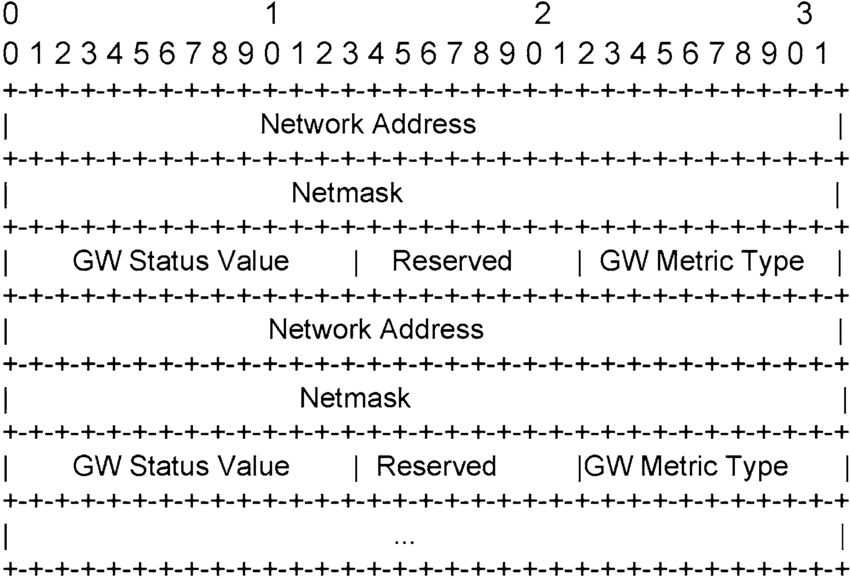
Gói tin MID được sử dụng để thông báo về các địa chỉ IP bổ sung của một node OLSR. Nếu một node có nhiều địa chỉ IP (ví dụ có nhiều giao diện mạng), gói tin MID cho phép node đó thông báo cho các node khác biết về tất cả các địa chỉ IP của mình. Điều này giúp các node trong mạng nhận biết rằng những địa chỉ IP khác nhau thuộc về cùng một node, tránh tình trạng coi chúng như những node riêng biệt.



Hình 7: Cấu trúc gói tin MID

* + 1. **Host and Network Association (HNA) Messages**

Gói tin HNA được sử dụng để thông báo về khả năng kết nối tới các mạng khác của một node trong mạng OLSR. Node nào có khả năng truy cập các mạng bổ sung (chẳng hạn như mạng Internet hoặc mạng con khác) sẽ gửi gói tin HNA để các node khác biết rằng có thể định tuyến thông qua nó để truy cập các mạng này. HNA thường được sử dụng để chỉ định một node làm gateway trong mạng OLSR.



Hình 8: Cấu trúc gói tin HNA

1. **Cách thức hoạt động:**

* Mỗi nút trong mạng gửi các gói tin HELLO để tìm ra các nút lân cận và xác định các liên kết. Dựa vào thông tin từ các gói HELLO, mỗi nút chọn ra một số nút lân cận làm MPR và chỉ các nút MPR mới phát lại các gói TC, giúp giảm tải cho mạng và tối ưu hóa việc phát quảng bá gói tin.
* Các nút MPR phát quảng bá các gói TC định kỳ, chứa thông tin về các liên kết đến các nút mà nó kết nối. Từ đó, mỗi nút có thể xây dựng và duy trì bảng định tuyến đầy đủ, cho phép truyền dữ liệu đến bất kỳ nút nào trong mạng một cách hiệu quả.
* Nếu một node có nhiều giao diện mạng (tức là nhiều địa chỉ IP), nó sẽ gửi gói MID để thông báo tất cả các địa chỉ IP của nó. Các node khác khi nhận gói MID sẽ biết rằng các địa chỉ IP khác nhau này thuộc cùng một node và sẽ lưu thông tin này vào bảng định tuyến để tránh việc xem các địa chỉ IP này như các node riêng biệt.
* Node có khả năng truy cập mạng bên ngoài (ví dụ như Internet hoặc các mạng nội bộ khác) sẽ gửi gói HNA để thông báo về các mạng mà nó có thể đóng vai trò gateway. Các node khác khi nhận được gói HNA sẽ cập nhật bảng định tuyến, thiết lập node gửi HNA làm gateway để kết nối tới các mạng ngoài này.
* Khi có yêu cầu truyền dữ liệu, nút gửi chỉ cần tra cứu bảng định tuyến đã có sẵn để tìm đường đi tối ưu và gửi gói tin theo đường dẫn này.

# PHƯƠNG PHÁP NGHIÊN CỨU

**Đối tượng nghiên cứu**

* 1. **Cấu trúc của gói tin OLSR**

Hiểu cách mà giao thức xử lý thông tin định tuyến thông qua các gói tin HELLO và TC (Topology Control).

* 1. **Thuật toán chọn đường đi tối ưu**

Đánh giá thuật toán này trong việc chọn đường truyền dữ liệu tối ưu, giảm thiểu độ trễ và tối đa hóa băng thông.

* 1. **Quá trình trao đổi thông tin liên kết**

Quan sát và phân tích việc cập nhật trạng thái giữa các nút trong mạng ad-hoc để đảm bảo sự ổn định và hiệu quả của hệ thống​​.

1. **Quy trình nghiên cứu:**
   1. **Mô hình triển khai:**

A diagram of a computer network

Description automatically generated

Hình 9: Topology

* Khu vực mạng MANET: 3 máy ảo TinyCore Linux đảm nhận vai trò các node chính tạo thành mạng MANET và triển khai giao thức OLSR giúp định tuyến mạng với nhau.
* Khu vực mạng WiFi: ESP32 dùng mạng WiFi SP\_Network để tiến hành gửi các dữ liệu đến node triển khai giao thức OLSR.
* Khu vực mạng LAN: máy ảo TinyCore Linux đóng vai trò gateway cho các thiết bị client nhận dữ liệu.
* Gmail SMTP Server: ESP32 gửi dữ liệu bằng giao thức SMTP đến máy chủ Gmail SMTP, các thiết bị client dùng giao thức IMAP/POP3 nhận dữ liệu từ máy chủ.
  1. **Cách thức triển khai**
  + Định tuyến trong mạng MANET:
* Tạo mạng MANET
* Thiết lập card wireless ở chế độ ad-hoc bằng lệnh:

***sudo iw dev*** *<interface>* ***set type ibss***

*ibss*:Independent Basic Service Set - các thiết bị có thể tự động phát hiện và kết nối với nhau trong phạm vi mà không cần cơ sở hạ tầng như router hoặc Access Point.

* Tạo/tham gia mạng MANET:

***sudo iw dev*** *<interface>* ***ibss join*** *<ssid> <channel frequency>*

*ssid:* tên mạng MANET

*channel frequency:* 2412 MHz (kênh 1), 2437 MHz (kênh 6), 5500 GHz (kênh 100),…

* Cấu hình IP tĩnh cho card wireless:

***sudo ifconfig*** *<interface> <ipaddress>* ***netmask*** *<subnet>* ***up***

* Dùng giao thức OLSR để định tuyến
* Mã nguồn OLSRd: sử dụng phiên bản OLSRd 0.9.9 trên <https://github.com/OLSR/olsrd/archive/master.zip>
* Biên dịch và cài đặt OLSRd:
* Tải mã nguồn OLSRd trên github.com:

***wget*** [***https://github.com/OLSR/olsrd/archive/master.zip***](https://github.com/OLSR/olsrd/archive/master.zip)

* Biên dịch mã nguồn: ***make***
* Cài đặt OLSRd: ***sudo make install***
* Cấu hình OLSRd:
* Chỉnh sửa tệp cấu hình:

***sudo nano /etc/olsrd/olsrd.conf***

Các thông số cần chú ý:

* *DebugLevel*: mức độ debug, ghi log
* *IpVersion*: sử dụng giao thức IPv4 hay IPv6
* *FIBMetric*: cách tính metric cho FIB (Forwarding Information Base)
* *ClearScreen*: bật hay tắt xóa màn hình
* *Hna4*: thông báo các mạng mà node có thể truy cập được
* *AutoDetectChange:* phát hiện các thay đổi trên giao diện
* *Ip4Broadcast:* địa chỉ broadcast
* *HelloInterval, HelloValidityTime:* khoảng thời gian và hiệu lực gói tin HELLO
* *TcInterval, TcValidityTime:* khoảng thời gian và hiệu lực gói tin TC
* *MidInterval, MidValidityTime:* khoảng thời gian và hiệu lực gói tin MID
* *HnaInterval, HnaValidityTime:* khoảng thời gian và hiệu lực gói tin HNA
* Quảng bá mạng ngoài phía client
  + - * Trong tệp cấu hình, chỉnh sửa hoặc thêm thông số Hna4 (Host Network Association for IPv4) giúp quảng bá mạng ngoài vào trong môi trường OLSR

*# OLSRd configuration*

***Hna4 {***

***<network address> <subnet>***

***…***

***}***

*Some configuration*

*# End file configure*

* + Truyền dữ liệu bằng giao thức TCP và SMTP:
* Đoạn code Arduino C++: thực hiện các chức năng kết nối WiFi, đo nhịp tim, gửi mail. Link code: <https://github.com/PhungVietBac/WENS_project_NT131.P13_group15/blob/main/WENS_project_NT131.P13_group15.ino>
* Mã code nhận và chuyển tiếp gói tin nhằm mục đích giúp cho gói tin phải đi vào vùng mạng MANET để nghiên cứu giao thức OLSR.

Link code: <https://github.com/PhungVietBac/WENS_project_NT131.P13_group15/blob/main/ReceiveAndForward.py>

* Kích hoạt cổng trên thiết bị khách để chấp nhận gói tin chuyển đến:

***sudo python3 -m http.server <port number>***

# PHÂN TÍCH, DEMO

* 1. **Phân tích mã nguồn OLSRd**
  2. **Hàm olsr\_openlog**

Giúp ghi lại các hoạt động, thông điệp log quan trọng vào hệ thống, hỗ trợ theo dõi, kiểm tra sự cố khi daemon hoạt động

A computer screen shot of a program code

Description automatically generated

Hình 10: Hàm olsr\_openlog

* 1. **Các hàm checksum cấu hình**

Thiết lập checksum cho cấu hình để xác minh rằng cấu hình không bị thay đổi ngoài ý muốn

A computer code with white text

Description automatically generated

Hình 11: Các hàm checksum cấu hình

* 1. **Hàm print\_version**

In ra phiên bản hiện tại của OLSRd lên màn hình

**A computer screen with a black background

Description automatically generated**

Hình 12: Hàm print\_version

* 1. **Kiểm tra quyền root**

Trên các hệ điều hành POSIX, nếu không thực thi chương trình với tư cách superuser root, chương trình OLSRd sẽ thoát với lỗi. Còn trên hệ điều hành Windows, ngăn chặn chuyển hướng gói tin IMCP và khởi tạo WinSock quản lý kết nối mạng

A computer screen shot of text

Description automatically generated

Hình 13: Kiểm tra quyền root

* 1. **Tải cấu hình từ file**

Giúp tải lên cấu hình đã được thiết lập trong file config, thường là file olsrd.conf. Nếu không tải lên được, chương trình sẽ thoát với lỗi

A screen shot of a computer

Description automatically generated

Hình 14: Tải cấu hình từ file config

* 1. **Xử lý đối số CLI**

Giúp lấy cấu hình mặc định của các giao diện mạng; phân tích, xử lý các đối số dòng lệnh của OLSRd; thiết lập cấu hình mặc định cho các giao diện được chỉ định từ dòng lệnh

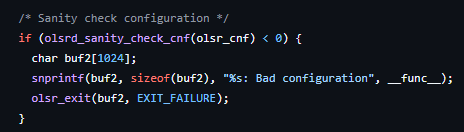
A computer screen shot of a program code

Description automatically generated

Hình 15: Xử lý đối số CLI

* 1. **Kiểm tra tính hợp lệ cấu hình**

Đảm bảo các tham số trong cấu hình đều hợp lệ, nếu không sẽ thoát với lỗi



Hình 16: Kiểm tra tính hợp lệ cấu hình

* 1. **Tạo socket ioctl**

Cho phép OLSRd thực hiện các thao tác trên giao diện mạng

A computer screen with white text

Description automatically generated

Hình 17: Tạo socket ioctl

* 1. **Tạo socket netlink (Linux)**

Tạo socket tương tác với kernel Linux thông qua giao thức Netlink, giúp quản lý định tuyến giao diện mạng và các thay đổi liên quan đến hệ thống mạng

A computer screen shot of a program code

Description automatically generated

Hình 18: Tạo socket netlink

* 1. **Tạo socket định tuyến trên các hệ thống BSD và macOS**

Giúp theo dõi, quản lý, định tuyến, nhận thông tin về các thay đổi mạng từ người dùng xuống kernel trên các hệ điều hành BSD hoặc macOS

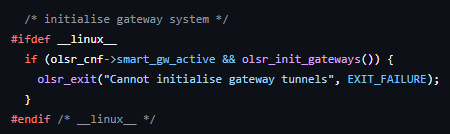
A computer screen shot of a program code

Description automatically generated

Hình 19: Tạo socket định tuyến trên BSD, macOS

* 1. **Hệ thống gateway (Linux)**

Khởi tạo các gateway cần thiết để OLSRd có thể kết nối ra ngoài hoặc giữa các mạng



Hình 20: Hệ thống gateway (Linux)

* 1. **Giao diện NIIT (Linux)**

Kích hoạt giao diện NIIT (NAT64 Integration Interface Tunneling) hỗ trợ tương tác giữa IPv4 và IPv6

A computer screen shot of white text

Description automatically generated

Hình 21: Giao diện NIIT (Linux)

* 1. **Thiết lập bộ đếm thời gian TC rỗng**

Tạo ra một timer để gửi gói tin TC rỗng khi node rời khỏi mạng, nhằm thông báo với các node khác rằng node này không còn phát sóng nữa

A screen shot of a computer code

Description automatically generated

Hình 22: Bộ đếm thời gian TC rỗng

* 1. **Chuyển tiếp IP và vô hiệu hóa chuyển hướng**

Bật chức năng chuyển tiếp IP và tắt chuyển hướng ICMP trên hệ thống để tránh các sự cố về định tuyến khi OLSRd chạy, đảm bảo rằng các gói tin có thể được chuyển tiếp giữa các giao diện mạng

A computer screen shot of a program code

Description automatically generated

A screenshot of a computer program

Description automatically generated

Hình 23: Chuyển tiếp IP và vô hiệu hóa chuyển hướng trên kernel linux

* 1. **Bộ phân tích parser**

Khởi tạo bộ phân tích để xử lý các gói tin mà OLSRd nhận được từ các node khác trong mạng, như các gói Hello, TC, MID và HNA

A screen shot of a computer program

Description automatically generated

Hình 24: Bộ phân tích parser

* 1. **Bộ xuất định tuyến**

Xuất và chia sẻ thông tin định tuyến cho các hệ thống hoặc ứng dụng khác nếu cần thiết

A screen shot of a computer program

Description automatically generated

Hình 25: Bộ xuất định tuyến

* 1. **Số thứ tự thông điệp**

Thiết lập số thứ tự cho các thông điệp được gửi đi từ OLSRd. Số thứ tự này đảm bảo rằng các thông điệp được sắp xếp và xử lý theo đúng thứ tự, tránh nhầm lẫn

A computer code with white text

Description automatically generated

Hình 26: Tạo số thứ tự thông điệp

* 1. **Khởi tạo mạng**

Khởi tạo hệ thống mạng của OLSRd, thiết lập các tham số và chuẩn bị kết nối mạng

A computer screen shot of a program code

Description automatically generated

Hình 27: Khởi tạo hệ thống mạng OLSRd

* 1. **Khởi tạo cơ sở dữ liệu giao diện mạng**

Kiểm tra và thiết lập các giao diện mạng. Nếu không phát hiện giao diện nào, nó sẽ kiểm tra tùy chọn *allow\_no\_interfaces*

A computer screen with white text

Description automatically generated

Hình 28: Khởi tạo cơ sở dữ liệu giao diện mạng

* 1. **Khởi tạo các bảng định tuyến**

Khởi tạo các bảng định tuyến cần thiết để OLSRd lưu trữ thông tin về các kết nối mạng

A screenshot of a computer program

Description automatically generated A screenshot of a computer program

Description automatically generated

Hình 29: Khởi tạo các bảng định tuyến

* 1. **Tải các plugin**

Nạp các plugin đã được cấu hình từ file cấu hình olsrd.conf. Các plugin cung cấp thêm chức năng hoặc mở rộng cho olsrd, như thêm các phương thức định tuyến hoặc khả năng giám sát mạng

A computer screen shot of a program code

Description automatically generated

Hình 30: Tải các plugin

* 1. **Thiết lập các quy tắc với mức độ ưu tiên**

Tạo các quy tắc định tuyến chính sách cho từng bảng định tuyến (rt\_table, rt\_table\_tunnel, và rt\_table\_default) với các mức độ ưu tiên khác nhau, cho phép olsrd kiểm soát chi tiết đường đi của các gói tin

A screenshot of a computer program

Description automatically generated A computer screen shot of a program code

Description automatically generated

Hình 31: Thiết lập các quy tắc với mức độ ưu tiên

* 1. **Bắt đầu scheduler**

Vòng lặp chính của OLSRd, chịu trách nhiệm xử lý tất cả các sự kiện của mạng và gửi thông điệp định kỳ. Scheduler sẽ tiếp tục chạy và quản lý mọi hoạt động của OLSRd cho đến khi nhận được tín hiệu dừng từ hệ điều hành

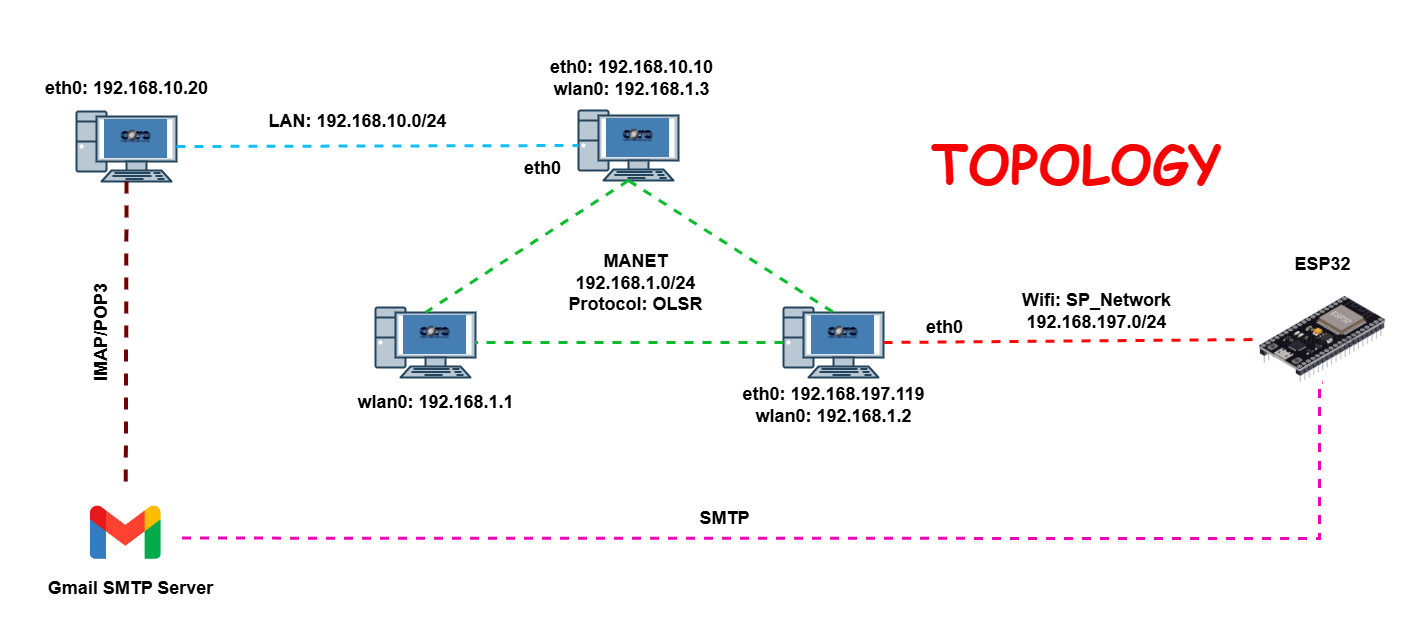
A screenshot of a computer program

Description automatically generated A computer screen with text on it

Description automatically generated

Hình 32: Vòng lặp scheduler chính

* 1. **Phân tích kết quả mô hình triển khai**
  2. **Truyền dữ liệu sức khỏe**
     + - * Việc truyền dữ liệu thông tin sức khỏe của người dùng sẽ được thực hiện theo cấu trúc sau: Một đoạn code được nạp vào ESP32 để tiến hành truyền dữ liệu gửi đi tới máy khách. Muốn dữ liệu tới được đích thì gói tin sẽ cần phải đi qua vùng mạng MANET được cấu hình giao thức OLSR, khi đó máy tham gia vừa có kết nối trực tiếp tới ESP32 vừa tham gia trong vùng mạng MANET sẽ thực thi đoạn code python để mở cổng nhận gói tin truyền vào và chuyển tiếp gói tin tới thiết bị khách thông qua việc tìm kiếm, định tuyến đường đi giữa các máy tính trạm trong MANET và xem xét đường đi có thể đi tới máy khách (nằm ở mạng ngoài).

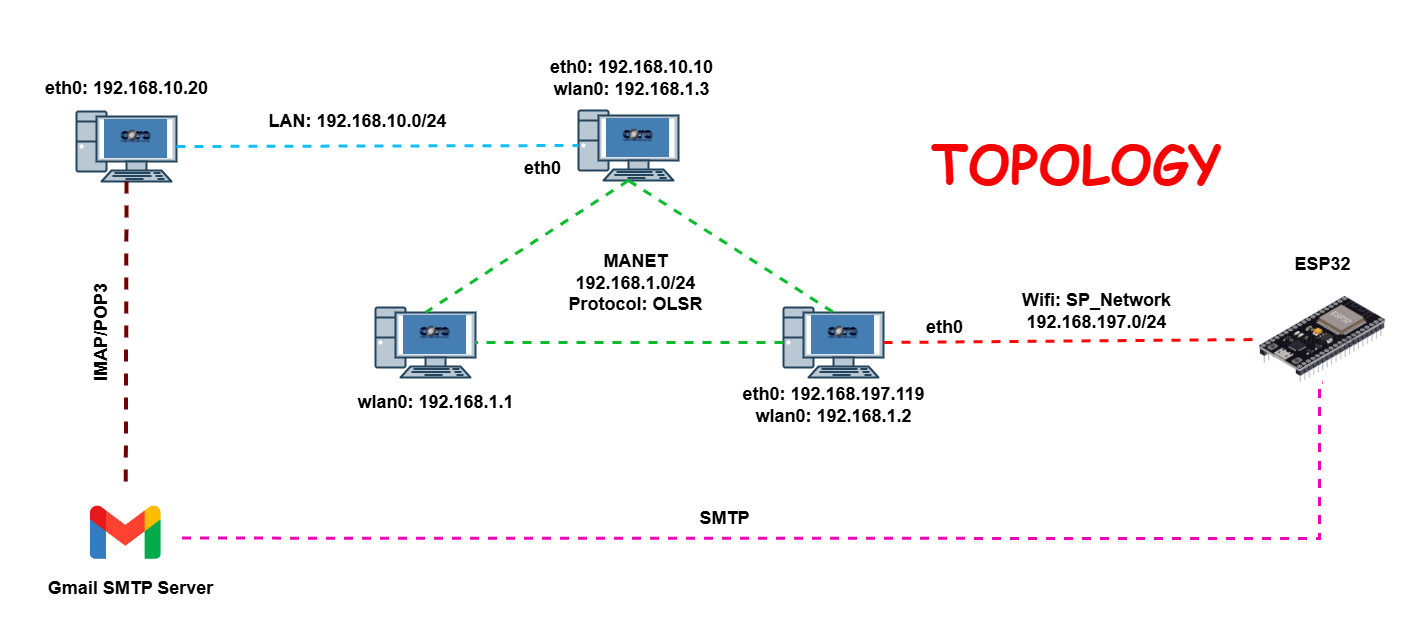


Client

Hình 33: Sơ đồ đường đi của dữ liệu

* + - * + Trình tự thực thi:

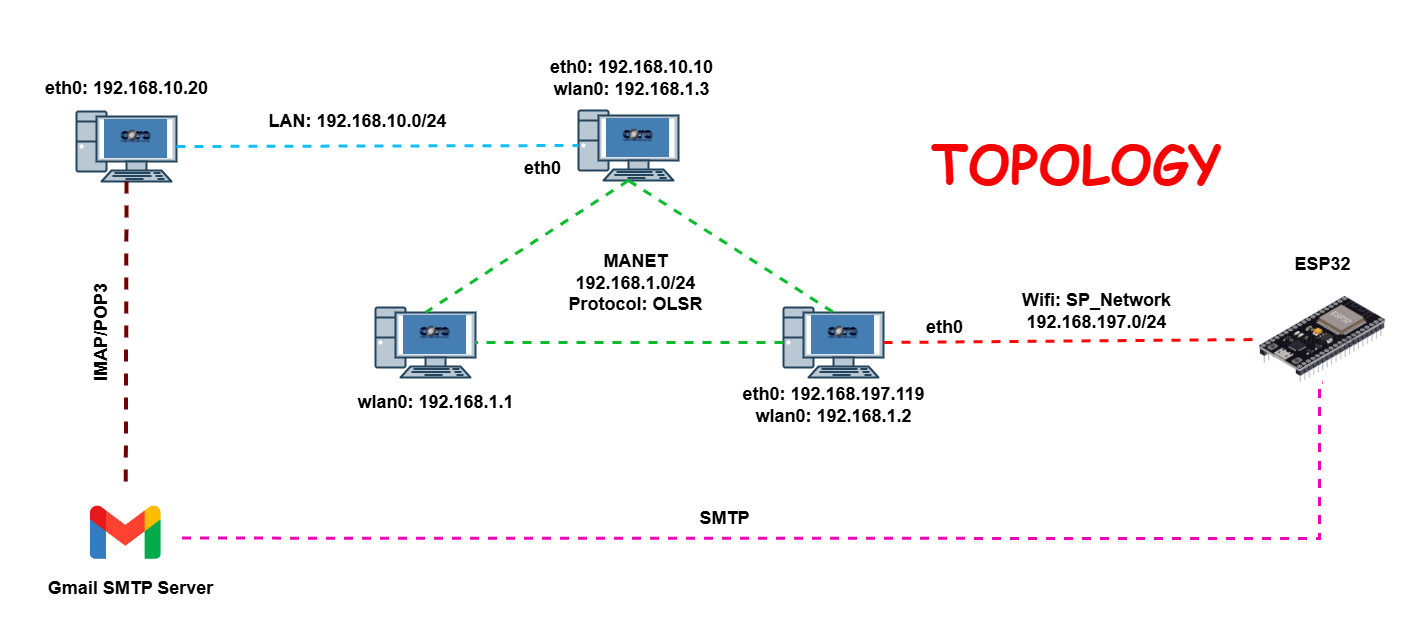
Nạp code vào ESP32 thông qua Arduino IDE, đoạn code có chức năng tìm kiếm địa chỉ đích chính là máy có kết nối trực tiếp tới nó và gửi đi gói tin đã được đo đạc. Vì ESP32 không có khả năng định tuyến nên ta cần phải gửi trực tiếp tới một node cố định có tham gia vào MANET, node ấy sẽ mang nhiệm vụ tìm kiếm đường đi dẫn tới máy khách và chuyển tiếp gói tin tới đường đó.



Client

Hình 34: ESP32 gửi dữ liệu tới node triển khai OLSR gần nhất

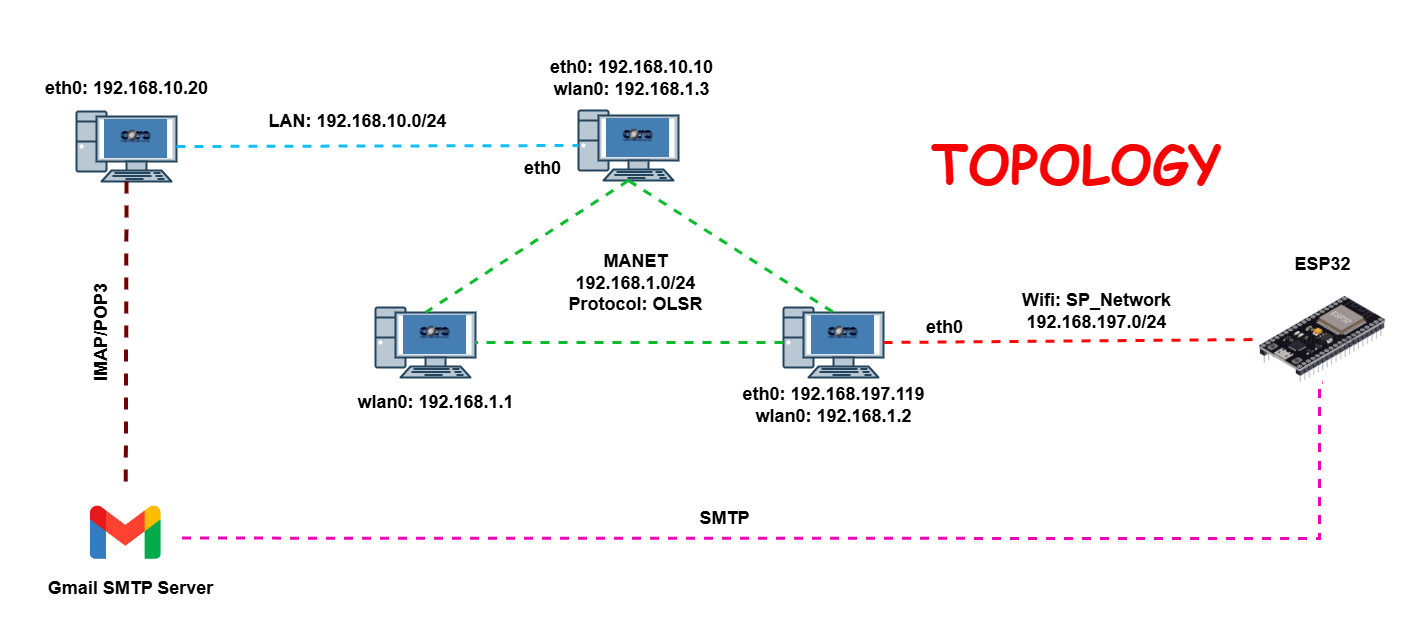
Máy tính vừa tham gia vùng mạng MANET vừa nhận được gói tin khi kết nối chung vùng mạng với ESP32 sẽ có nhiệm vụ chuyển tiếp gói tin tới máy khách sau khi đã học được đường đi trung gian nhờ vào giao thức OLSR.



Client

Hình 35: Chuyển tiếp dữ liệu đến node gateway của mạng client

Sau khi chuyển tiếp gói tin qua đường đi dẫn tới máy khách, tại máy trạm có kết nối chung vùng mạng với máy khách, khi nhận được sẽ tiếp tục chuyển tiếp gói tin này tới máy khách.



Client

Hình 36: Chuyển dữ liệu tới client

* 1. **Gửi mail thông báo tới Client**
     + - * Sau khi ESP32 thu thập dữ liệu từ cảm biến MAX30102, dữ liệu này được lưu trữ dưới dạng một chuỗi (string).
         * Sử dụng thư viện ESP Mail Client để thiết lập phiên kết nối với máy chủ SMTP (gồm host, port).
         * ESP32 sẽ tạo một tin nhắn email với nội dung đã thu thập từ các cảm biến và kết nối với máy chủ SMTP, nếu kết nối thành công thì thực hiện gửi dữ liệu đến người nhận.
         * Khi email đã được gửi đi, ESP32 đóng kết nối với máy chủ SMTP để giải phóng tài nguyên và đảm bảo an toàn thông tin đăng nhập.
  2. **Demo**

Link video: <https://drive.google.com/file/d/1ysUE0np0qRvB2nrZqaAj0L541FIKm4vS/view?usp=drive_link>

# TỔNG KẾT

* 1. **Thuận lợi**
     + - * Áp dụng kết hợp kiến thức của các môn học trước (Quản trị mạng và hệ thống, Hệ điều hành) cùng với kiến thức ở môn học này một cách hiệu quả trong việc tạo máy ảo, tương tác với hệ điều hành và biên dịch mã nguồn.
         * Có kiến thức nền tảng, hiểu sâu về quá trình định tuyến gói tin bằng giải thuật OLSR.
         * Hệ điều hành TinyCore nhỏ nhẹ, dễ lắp đặt các mã nguồn mà không phải tiêu tốn quá nhiều tài nguyên, dung lượng.
         * Các thành viên năng nổ, tích cực trong việc xây dựng và phát triển đồ án.
  2. **Khó khăn**
     + - * Bị giới hạn về số lượng thiết bị cần thiết trong việc triển khai dự án.
         * Tuy TinyCore là hệ điều hành nhỏ nhẹ, nhưng vẫn khá khó sử dụng khi rất ít tiện ích được tích hợp vào trong hệ điều hành. Đồng thời còn mới lạ với các câu lệnh của hệ điều hành này nên sẽ tốn một khoảng thời gian để làm quen với việc thao tác trên đó.
         * Chi phí giới hạn nên chất lượng thiết bị với chức năng đo đạc vẫn chưa được chuẩn xác và ổn định.
  3. **Kết quả**
     + - * Triển khai thành công một mạng MANET được cấu hình định tuyến bằng giao thức OLSR.
         * Truyền dữ liệu thành công như kịch bản đã đặt ra, ngoài ra mở rộng triển khai hiệu quả phương thức gửi mail (SMTP) tới một máy client.
         * Mô hình thực thi khả quan, phù hợp cấu trúc dự án.
  4. **Kết luận**
     + - * **Đạt được mục tiêu ban đầu**: Đồ án nhằm thu thập, truyền tải dữ liệu từ ESP32 qua mạng MANET (OLSRD) để đưa đến một server hoặc các thiết bị khác. Dữ liệu như thông tin sức khỏe có thể được thu thập từ cảm biến (như MAX30102), xử lý và truyền đi ổn định, đáp ứng được yêu cầu thời gian thực.
         * **Tính khả thi của công nghệ**: Việc sử dụng ESP32 cùng với giao thức OLSR cho thấy tính khả thi cao cho các mạng lưới di động tự tổ chức. Giao thức OLSR hoạt động tốt trong mạng MANET, cho phép các máy tính kết nối với nhau, tạo ra một môi trường truyền tải dữ liệu linh hoạt và mở rộng được phạm vi mạng.
         * **Độ ổn định và hiệu suất**: ESP32 có khả năng xử lý dữ liệu từ cảm biến và truyền qua mạng khá ổn định. Tuy nhiên, hiệu suất có thể bị ảnh hưởng khi số lượng nút tăng lên hoặc khi môi trường có nhiều nhiễu sóng. Việc tối ưu hoá mạng MANET và tối ưu hoá mã trên ESP32 là cần thiết để đạt được hiệu suất cao nhất.
         * **Mở rộng trong tương lai**: Hệ thống có tiềm năng mở rộng thêm nhiều loại cảm biến hoặc tích hợp thêm các giao thức bảo mật dữ liệu. Ngoài ra, có thể tích hợp thêm các thành phần phần mềm như dashboard phân tích dữ liệu trực tiếp hoặc phát triển hệ thống lưu trữ dữ liệu lâu dài để phục vụ nghiên cứu. Bên cạnh đó, việc tăng cường tính tự động khi không cần phải tự cấu hình OLSRd trên các thiết bị cũng được xem xét mở rộng và tối ưu hóa trong tương lai.
  5. **Phân công, đánh giá**
     + - * Lê Ngọc Kiều Anh (rất tích cực)
       - Soạn nội dung Word
       - Thiết kế slide
       - Code Python xử lý nhận và chuyển tiếp gói tin
       - Demo, thuyết trình
         * Phùng Việt Bắc (rất tích cực)
* Soạn nội dung Word
* Phân tích mã nguồn OLSRd
* Code Arduino C++ kết nối, đo nhịp tim và truyền dữ liệu
* Demo, thuyết trình
  + - * + Trần Phước Đại (rất tích cực)
* Soạn nội dung Word
* Thiết kế slide
* Code Arduino C++ gửi mail
* Chỉnh sửa video
* Demo, thuyết trình

# TÀI LIỆU THAM KHẢO

[1] L. Proven, “Tiny Core Linux 15 stuffs modern computing in a nutshell.” Accessed: Nov. 10, 2024. [Online]. Available: https://www.theregister.com/2024/03/04/tiny\_core\_linux\_15/

[2] “Ubuntu Manpage: olsrd.conf - configuration file for olsrd(8).” Accessed: Nov. 10, 2024. [Online]. Available: https://manpages.ubuntu.com/manpages/focal/man5/olsrd.conf.5.html

[3] “Smart Heart Rate Monitoring with ESP32.” Accessed: Nov. 10, 2024. [Online]. Available: https://www.hqonline.com/blog/smart-heart-rate-monitoring-with-esp32?srsltid=AfmBOor9AFXJx07vWIUVFDYiQyH-fFeVLoSigDvGGGi1F8lbdtkDN4i7

[4] “TinyCore Linux - Initial steps,” Layer9. Accessed: Nov. 10, 2024. [Online]. Available: https://tiagojsilva.github.io/en/unixlike/meme/2021-10-29\_tinycorelinux-install/

[5] “OLSR for Linux.” Accessed: Nov. 10, 2024. [Online]. Available: https://tldp.org/HOWTO/OLSR-IPv6-HOWTO/olsrlinux.html