



BÁO CÁO ĐỒ ÁN CUỐI KHÓA
Đề tài: Giải pháp IoT dành cho thiết bị “Công tắc cầu thang”

Học viên thực hiện: Trần Doãn Mạnh
Mã sinh viên: FX17476
Mentor hướng dẫn: Nguyễn Phú Phượng

Quảng Ninh, ngày 01 tháng 7 năm 2024

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

Quảng Ninh, ngày 01 tháng 7 năm 2024

NHIỆM VỤ ĐỒ ÁN

- Họ và tên:** Trần Doãn Mạnh
- Mã sinh viên:** FX17476
- Tên đồ án:** Giải pháp IoT dành cho thiết bị “Công tắc cầu thang”
- Mentor hướng dẫn:** Nguyễn Phú Phượng
- Ngày nhận đồ án:** 08/4/2024
- Ngày hoàn thành đồ án:** 29/6/2024./.

SINH VIÊN THỰC HIỆN
(Ký, ghi rõ họ tên)

Mạnh
Trần Doãn Mạnh

MỤC LỤC

Contents

MỤC LỤC	3
DANH MỤC HÌNH ẢNH	7
DANH MỤC BẢNG BIỂU	10
LỜI CẢM ƠN	11
LỜI NÓI ĐẦU	12
CHƯƠNG I: TỔNG QUAN VỀ ĐỒ ÁN	13
1. Tính cấp thiết của đồ án	13
2. Phạm vi đồ án	14
3. Tổng quan thiết bị Công tắc cầu thang	14
3.1. Thành phần Công tắc cầu thang	14
3.2. Tổng quan tính năng thiết bị Công tắc cầu thang	15
3.3. Mô hình thiết bị Công tắc cầu thang	16
CHƯƠNG II: CƠ SỞ LÝ THUYẾT	17
1. Mạng Zigbee	17
1.1. Tổng quan về mạng Zigbee	17
1.2. Zigbee, Bluetooth và IEEE 802.15.4	17
1.3. Mối quan hệ giữa Zigbee và chuẩn IEEE 802.15.4	18
1.3.1. Tầng Physical (<i>PHY</i>)	19
1.3.2. Tầng Medium Access Control (<i>MAC</i>)	20
1.3.3. Tầng Network (<i>NWK</i>)	20
1.3.4. Tầng Application (<i>APL</i>)	21
1.4. Loại thiết bị và vai trò của thiết bị trong mạng Zigbee	22
1.4.1. Loại thiết bị trong mạng Zigbee	22
1.4.2. Vai trò của thiết bị trong mạng Zigbee	22
1.5. Zigbee 3.0	24
1.5.1. Zigbee 3.0 là gì	24
1.5.2. Các mô hình mạng của Zigbee	25
1.5.3. Khởi tạo mạng, gia nhập mạng	26
1.5.4. Hướng giao tiếp trong mạng Zigbee	28
1.5.5. Định danh mạng Zigbee	30
1.5.6. Địa chỉ mạng của thiết bị trong mạng Zigbee	31

1.5.7. Các phương thức truyền tin trong mạng Zigbee	32
1.5.8. Zigbee Routing.....	34
1.5.9. Route repair	36
1.5.10. Tính năng bảo mật trong mạng Zigbee	37
1.5.11. Binding, Group và Scenes trong mạng Zigbee	39
2. Vi điều khiển ESP32	41
2.1. Giới thiệu về vi điều khiển ESP32	41
2.2. Kiến trúc ESP32	41
2.2.1. CPU	41
2.2.2. Bộ nhớ	42
2.2.3. Khối Wi-Fi, Bluetooth.....	44
2.2.4. Bộ quản lý năng lượng	44
2.2.5. Tính năng bảo mật tích hợp	45
2.3. Công cụ lập trình ESP32	45
2.4. Kết nối Wi-Fi, Bluetooth.....	47
2.5. Ứng dụng của ESP32	47
3. Web	48
3.1. Sơ lược mô hình Client – Server	48
3.1.1. Khái niệm Client, Server.....	48
3.1.2. Mô hình Client - Server	50
3.1.3. Quá trình giao tiếp giữa Client – Server	51
3.2. Các thành phần cơ bản của một Website.....	52
3.2.1. Phân loại Website	52
3.2.2. Các thành phần của Static Website	53
3.3. Kỹ thuật, giao thức sử dụng trong giao tiếp giữa Client – Server	60
3.3.1. Kỹ thuật AJAX (<i>Asynchronous JavaScript and XML</i>)	60
3.3.2. Giao thức TCP/IP	63
3.3.3. Giao thức DHCP	64
3.3.4. Giao thức HTTP	65
CHƯƠNG III: THIẾT BỊ CÔNG TẮC CẦU THANG	67
1. Tính năng thiết bị Công tắc cầu thang	67
1.1. Giao diện điều khiển	67
1.1.1. Đăng ký, đăng nhập mật khẩu để truy cập vào trình điều khiển	67
1.1.2. Tính năng mạng Zigbee trên Web	67

1.1.3. Cập nhật thông số cảm biến	68
1.1.4. Điều khiển đèn cầu thang.....	69
1.2. Tính năng của Gateway (<i>Zigbee Coordinator kết hợp ESP32</i>)	69
1.3. Tính năng của công tắc cầu thang.....	70
2. Hoạt động của thiết bị Công tắc cầu thang.....	70
2.1. Mô tả chi tiết hoạt động của công tắc cầu thang.....	70
2.1.1. Hoạt động của Zigbee Coordinator.....	70
2.1.2. Hoạt động của Below Switch.....	73
2.1.3. Hoạt động của Upper Switch	77
2.2. Luồng hoạt động của thiết bị công tắc cầu thang.....	80
2.2.1. Luồng gia nhập mạng Zigbee của Switches	80
2.2.2. Luồng giao tiếp giữa Web & Switches	81
2.2.3. Luồng giao tiếp giữa Switches & Web	82
3. Khung truyền UART	83
3.1. Vai trò của khung truyền UART.....	83
3.2. Các thành phần trong khung truyền UART	83
3.3. Sơ đồ thuật toán UART	84
3.3.1. Sơ đồ thuật toán truyền dữ liệu UART	84
3.3.2. Sơ đồ thuật toán xử lý dữ liệu UART nhận được	85
4. Sơ đồ hoạt động, thuật toán và phân tầng	86
4.1. Khái niệm về các loại sơ đồ	86
4.2. Zigbee Coordinator	86
4.2.1. Vai trò	86
4.2.2. Sơ đồ hoạt động của ZC.....	87
4.2.3. Sơ đồ thuật toán của ZC	88
4.2.4. Sơ đồ phân tầng của ZC	92
4.3. Below Switch	93
4.3.1. Vai trò	93
4.3.2. Sơ đồ hoạt động của Below Switch	93
4.3.3. Sơ đồ thuật toán của Below Switch	97
4.3.4. Sơ đồ phân tầng của Below Switch	101
4.4. Upper Switch.....	102
4.4.1. Vai trò	102
4.4.2. Sơ đồ hoạt động của Upper Switch.....	102

4.4.3. Sơ đồ thuật toán của Upper Switch.....	105
4.4.4. Sơ đồ phân tầng của Upper Switch.....	107
4.5. Giao diện Web điều khiển & ESP32	108
4.5.1. Vai trò	108
4.5.2. Sơ đồ hoạt động của giao diện Web điều khiển & ESP32	108
4.5.3. Sơ đồ thuật toán của giao diện Web điều khiển & ESP32	109
KHÓ KHĂN TRONG QUÁ TRÌNH HOÀN THIỆN ĐO ÁN	113
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	114
DANH MỤC TÀI LIỆU THAM KHẢO.....	115

DANH MỤC HÌNH ẢNH

DANH MỤC HÌNH ẢNH CHƯƠNG I

Hình 1.1: EFR32 Board Kit	15
Hình 1.2: Vi điều khiển ESP-WROOM-32.....	15
Hình 1.3: Mô hình thiết bị công tắc cầu thang.....	16

DANH MỤC HÌNH ẢNH CHƯƠNG II

Hình 2.1: So sánh chuẩn Zigbee với Bluetooth và IEEE 802.11b	17
Hình 2.2: ZigBee Wireless Networking Protocol Layers	18
Hình 2.3: ZigBee Wireless Networking Protocol Layers (Detail)	19
Hình 2.4: Loại thiết bị trong mạng Zigbee.....	22
Hình 2.5: Vai trò của thiết bị trong mạng Zigbee.....	22
Hình 2.6: Mạng Centralized and Distributed network.....	24
Hình 2.7: Mô hình Star Network.....	25
Hình 2.8: Mô hình Full Mesh Network.....	26
Hình 2.9: Mô hình Hybrid Mesh Network.....	26
Hình 2.10: Các bước khởi tạo mạng của Zigbee Coordinator	27
Hình 2.11: Quá trình gia nhập mạng	28
Hình 2.12 : Quá trình giao tiếp giữa Device & ZC sử dụng Beacon.....	29
Hình 2.13: Quá trình giao tiếp giữa Device & ZC không sử dụng Beacon.....	29
Hình 2.14: Quá trình giao tiếp giữa ZC & Device sử dụng Beacon.....	29
Hình 2.15: Quá trình giao tiếp giữa ZC & Device không sử dụng Beacon.....	30
Hình 2.16: Quá trình giao tiếp giữa 2 devices	30
Hình 2.17: Phương thức truyền tin Unicast	32
Hình 2.18: Phương thức truyền tin Broadcast.....	32
Hình 2.19: Phương thức truyền tin Multicast	33
Hình 2.20: Phương thức truyền tin Many-To-One.....	34
Hình 2.21: Mô hình truyền dữ liệu.....	34
Hình 2.22: Mô hình tính toán giá trị Link Cost	35
Hình 2.23: Routing Discovery	36
Hình 2.24: Route repair	37
Hình 2.25: Quá trình cấp phát TCLK và NWK	37
Hình 2.26: Quá trình cấp phát Application Key.....	38
Hình 2.27: Binding giữa hai nút mạng	39
Hình 2.28: Quá trình Binding giữa hai nút mạng	40
Hình 2.29: Mô hình Group trong mạng Zigbee	40

Hình 2.30: Các kiểu value trong cặp Key-Value lưu trữ trong NVS Flash	42
Hình 2.31: Namesapce-Key-Value lưu trữ trong NVS Flash.....	42
Hình 2.32: Thực thể page trong NVS Flash.....	43
Hình 2.33: Thành phần của thực thể page trong NVS Flash	43
Hình 2.34: Mô hình Client – Server.....	50
Hình 2.35: Mô hình giao tiếp giữa Client – Server.....	51
Hình 2.36: Mô hình Static and Dynamic Website.....	52
Hình 2.37: Cấu trúc cơ bản của Static Website.....	53
Hình 2.38: Cấu trúc cơ bản của HTML	55
Hình 2.39: Bộ cục của CSS.....	56
Hình 2.40: Cấu trúc của CSS	56
Hình 2.41: Cách nhúng CSS vào Website.....	57
Hình 2.42: Nhúng Javascript trực tiếp vào HTML	59
Hình 2.43: Nhúng Javascript vào HTML bằng tệp tin riêng biệt	59
Hình 2.44: Minh họa mối quan hệ giữa HTML, CSS và JS	60
Hình 2.45: Mô hình thông thường và Mô hình AJAX.....	62
Hình 2.46: Mô hình giao tiếp Client – Server sử dụng giao thức HTTP	66

DANH MỤC HÌNH ẢNH CHƯƠNG III

Hình 3.1: Giao diện Register.....	67
Hình 3.2: Giao diện Login	67
Hình 3.3: Giao diện Zigbee Network.....	68
Hình 3.4: Giao diện Environmental Sensor	68
Hình 3.5: Giao diện Stair Switch	69
Hình 3.6: Luồng giao nhập mạng Zigbee của Switches	80
Hình 3.7: Luồng giao tiếp giữa Web & Switches	81
Hình 3.8: Luồng giao tiếp giữa Switches & Web	82
Hình 3.9: Khung truyền UART	83
Hình 3.10: Sơ đồ thuật toán truyền dữ liệu UART	84
Hình 3.11: Sơ đồ thuật toán xử lý dữ liệu UART nhận được	85
Hình 3.12: Sơ đồ hoạt động của ZC.....	87
Hình 3.13: Sơ đồ thuật toán ZC xử lý bản tin nhận được từ Web	88
Hình 3.14: Sơ đồ thuật toán ZC xử lý report nhận được từ SWs và gửi cho Web .	89
Hình 3.15: Sơ đồ thuật toán xử lý nút bấm	90
Hình 3.16: Sơ đồ thuật toán điều khiển LED.....	91
Hình 3.17: Sơ đồ phân tầng của ZC	92
Hình 3.18: Sơ đồ hoạt động của Below Switch	93

Hình 3.19: Sơ đồ hoạt động nhánh Zigbee Network của Below Switch	94
Hình 3.20: Sơ đồ hoạt động lựa chọn chế độ “Auto”, “Manual” của Below SW ..	94
Hình 3.21: Sơ đồ hoạt động chế độ “Auto”, “Manual” của Below Switch	95
Hình 3.22: Sơ đồ hoạt động nhánh Sensor của Below Switch	96
Hình 3.23: Sơ đồ hoạt động nhánh Binding của Below Switch	96
Hình 3.24: SĐTT Below SW xử lý bản tin nhận được từ ZC và Upper SW.....	97
Hình 3.25: Sơ đồ thuật toán Hàm đọc dữ liệu cảm biến ánh sáng.....	98
Hình 3.26: Sơ đồ thuật toán Hàm đọc dữ liệu cảm biến nhiệt độ, độ ẩm.....	98
Hình 3.27: Sơ đồ thuật toán Hàm “I2C_Transfer”.....	99
Hình 3.28: Sơ đồ thuật toán sử dụng Binding truyền dữ liệu	100
Hình 3.29: Sơ đồ phân tầng của Below Switch	101
Hình 3.30: Sơ đồ hoạt động của Upper Switch.....	102
Hình 3.31: Sơ đồ hoạt động nhánh Zigbee Network của Upper Switch.....	103
Hình 3.32: Sơ đồ hoạt động lựa chọn chế độ “Auto”, “Manual” của Upper SW.	103
Hình 3.33: Sơ đồ hoạt động chế độ “Auto”, “Manual” của Upper Switch.....	104
Hình 3.34: Sơ đồ hoạt động nhánh Binding của Below Switch	105
Hình 3.35: SĐTT Upper SW xử lý bản tin nhận được từ ZC và Below SW.....	106
Hình 3.36: Sơ đồ phân tầng của Upper Switch.....	107
Hình 3.37: Sơ đồ hoạt động của giao diện Web điều khiển & ESP32	108
Hình 3.38: Sơ đồ thuật toán đăng ký, đăng nhập tài khoản	109
Hình 3.39: Sơ đồ thuật toán ESP32 xử lý bản tin nhận được từ ZC.....	110
Hình 3.40: Sơ đồ thuật toán ESP32 xử lý bản tin nhận được từ ZC.....	111
Hình 3.41: Sơ đồ thuật toán Web truyền lệnh điều khiển	112

DANH MỤC BẢNG BIỂU**DANH MỤC BẢNG BIỂU CHƯƠNG II**

Bảng 2.1: So sánh sự khác nhau giữa Client và Server	50
Bảng 2.2: So sánh mô hình thông thường và mô hình AJAX.....	61

DANH MỤC BẢNG BIỂU CHƯƠNG III

Bảng 3.1: Hoạt động của Zigbee Coordinator	73
Bảng 3.2: Hoạt động của Below Switch	76
Bảng 3.3: Hoạt động của Upper Switch.....	79
Bảng 3.4: Thành phần khung truyền UART	84

LỜI CẢM ƠN

Trước tiên, tôi xin giới thiệu sơ qua về bản thân mình. Tôi đã tốt nghiệp đại học ngành Kỹ thuật Cơ khí, chuyên ngành Cơ khí Động lực vào đầu năm 2020. Trong quá trình học Đại học và làm việc, tôi luôn có ước mơ tạo ra những sản phẩm đáp ứng nhu cầu thực tế cuộc sống. Tuy nhiên, sau khi ra trường làm việc, công việc hiện tại không đáp ứng được nguyện vọng của bản thân, do đó tôi đã tìm một hướng đi mới – **Internet of things**. Bước vào IoT, nó là một ngành hoàn toàn mới, nên tôi cảm thấy rất khó khăn và mông lung trước một khối lượng kiến thức khổng lồ, tôi không biết bắt đầu từ đâu, không biết cần trang bị cho bản thân những kiến thức nền tảng nào để bắt đầu con đường này... Và tôi luôn mong ước tìm được một người thầy hoặc một chương trình dạy online uy tín để giúp tôi giải quyết những vấn đề trên. Thật may mắn, trong một lần sử dụng Facebook, một Group Điện tử mà tôi tham gia đã chia sẻ chương trình học IoT của Funix. Tôi đã cảm thấy rất vui, hào hứng và sau một thời gian ngắn tìm hiểu, tôi đã liên hệ với nhà trường, quyết định theo học. Đến thời điểm hiện tại, tôi cảm thấy mình đã có một quyết định đúng đắn. Funix đã giải quyết cho tôi tất cả những vấn đề mà tôi gặp phải khi bước chân vào con đường IoT - một chương trình học bài bản, một đội ngũ mentor có chất lượng, kinh nghiệm. Bên cạnh đó, Funix đã tạo điều kiện gia hạn thời gian học cho tôi nhiều lần để giúp tôi hoàn thành trọn vẹn chương trình đào tạo. Tôi xin gửi lời cảm ơn chân thành đến Funix!

Với mong muốn sau khi học xong các môn cơ sở tại Funix, tôi có thể lập trình một sản phẩm IoT điều khiển đồng bộ giữa Web, App và phần cứng. Tham khảo ý kiến của mentor hướng dẫn Nguyễn Phú Phượng, tôi đã lựa chọn đồ án “**Giải pháp IoT dành cho thiết bị công tắc cầu thang**”. Từ những kiến thức nền tảng về lập trình C (*cơ bản, nâng cao*), kiến thức về vi điều khiển (*STM32*), mạng truyền thông không dây Zigbee được học tại Funix kết hợp tìm hiểu kiến thức mới về Web, ESP32 và đặc biệt là sự định hướng, hướng dẫn nhiệt tình của mentor Nguyễn Phú Phượng đã giúp tôi hoàn thành trọn vẹn các yêu cầu đặt ra với đồ án này. Tôi xin gửi lời cảm ơn chân thành đến **mentor Nguyễn Huy Hoàng, Nguyễn Phú Phượng và hanah Nhu, hanah Trần Nhung, hanah Thùy Trinh** – những người đã đồng hành, giúp đỡ, hướng dẫn, giải đáp thắc mắc, truyền đạt kinh nghiệm làm việc cho tôi trong suốt quá trình học ngành IoT!

LỜI NÓI ĐẦU

Ngày nay, nhà thông minh đang dần trở thành xu hướng tất yếu trong cuộc sống hiện đại. Nhờ sự ứng dụng của công nghệ Internet vạn vật (*IoT*), các thiết bị trong nhà được kết nối và điều khiển tự động, mang đến sự tiện nghi, an ninh và tiết kiệm năng lượng cho người sử dụng.

Công tắc cầu thang hay còn gọi là công tắc đảo chiều là một trong những thiết bị không thể thiếu khi thi công nhà cao tầng, giúp cho việc bật/tắt đèn cầu thang được dễ dàng và thuận tiện. Tuy nhiên, để mang đến trải nghiệm tiện lợi hơn cho người sử dụng so với công tắc cầu thang truyền thống như tính năng tự động hóa việc bật/tắt, tính năng điều khiển từ xa bằng Web, đồ án này tạo ra "***Giải pháp IoT dành cho thiết bị Công tắc cầu thang***". Thông qua đồ án này, việc bật/tắt đèn cầu thang sẽ được thực hiện đồng bộ thông qua việc điều khiển bằng nút bấm cơ, cảm biến chuyển động PIR và điều khiển từ xa bằng Web.

CHƯƠNG I: TỔNG QUAN VỀ ĐỒ ÁN

1. Tính cấp thiết của đồ án

Hiện nay, các từ khóa như IOT hay Smart Home, Smart Things không còn là những thuật ngữ xa lạ, trái lại IOT đang ngày càng là xu hướng được ưu chuộng trên thế giới và cả Việt Nam. Theo thông tin được cung cấp bởi Statista (*một nguồn thông tin uy tín về thống kê và dữ liệu thị trường*):

- ❖ Doanh thu trên thị trường nhà thông minh tại Việt Nam dự kiến đạt 328,6 triệu USD vào năm 2024.
- ❖ Mức độ phổ biến của nhà thông minh tại Việt Nam dự kiến là 15,4% vào năm 2024 và dự kiến đạt 25,7% vào năm 2028.

Với **những thông tin trên**, không khó để nhìn thấy rằng xu hướng phát triển của IoT nói chung và Smart Home nói riêng tại thị trường Việt Nam là rất tiềm năng. Bên cạnh đó, Nhà nước cũng ban hành nhiều chính sách nhằm hỗ trợ cho sự phát triển công nghệ IoT như cấp phép việc sử dụng dải tần dưới 1GHz tại Việt Nam vào năm 2018. Việc cấp phép này cung cấp thêm một giải pháp mạng không dây được sử dụng cho IoT, đó là Z-Wave. Ngoài ra, cơ sở hạ tầng mạng Internet cũng đang được cải thiện đáng kể (*Hiện nay, nước ta đang triển khai hệ thống mạng 5G, cải thiện đáng kể tốc độ truyền nhận dữ liệu cùng nhiều tính năng ưu việt khác*).

Đối với Smart Home, sự thiếu vắng của IoT trong nhà sẽ khiến cho mọi thứ trở nên tẻ nhạt, lặp đi lặp lại và mất nhiều thời gian cho những công việc đơn giản, mọi thứ gần như được làm thủ công và chúng ta sẽ phải ghi nhớ, lên kế hoạch và tự làm rất nhiều thứ. Nhưng kể từ khi IOT tham gia vào hộ gia đình, ngôi nhà trở nên thú vị hơn rất nhiều:

- ❖ Máy giặt có thể tự động giặt giũ theo lịch đã đặt trước, tích hợp nhiều tính năng như phân tích lượng nước cần sử dụng, cách thức giặt,...
- ❖ Tủ lạnh có thể tự động điều chỉnh nhiệt độ, tự động đưa ra các công thức nấu ăn với các thành phần đang có sẵn hoặc có thể thông báo cho người dùng khi thực phẩm hết hạn hoặc cần phải mua thêm...
- ❖ Đèn tự động cũng giúp rất nhiều trong ngôi nhà của bạn, thắp sáng theo từng bước chân, điều đó làm cho bạn không cần phải động tới một công tắc nào, không còn phải lẩn mò trong bóng tối. Điều này thật sự tinh tế đối với những người già hoặc trẻ nhỏ.

Tất nhiên là vẫn còn rất nhiều ứng dụng khác mà IOT mang lại, có thể nói rằng với IOT, cả căn nhà như đang nằm gọn trong lòng bàn tay của chúng ta. Một thành phần cũng rất quan trọng trong Smart Home đó là Công tắc cầu thang. Công tắc cầu thang hay còn gọi là công tắc đảo chiều là một trong những thiết bị không

thể thiếu khi thi công nhà cao tầng, giúp cho việc bật/tắt đèn cầu thang được dễ dàng và thuận tiện. Tuy nhiên, công tác cầu thang truyền thống còn đơn giản, chưa mang lại sự tiện lợi trong cách sử dụng cho người sử dụng và dần trở nên lỗi thời khi Khoa học – Kỹ thuật phát triển như ngày nay. Hiểu được yêu cầu đó, tôi đã quyết định thực hiện đồ án “***Giải pháp IoT dành cho thiết bị Công tắc cầu thang***”.

2. Phạm vi đồ án

Đồ án này tập trung vào việc phát triển “***Giải pháp IoT dành cho thiết bị Công tắc cầu thang***” sử dụng các kiến thức sau:

- ❖ Lập trình C.
- ❖ Lập trình vi điều khiển ESP32, EFR32.
- ❖ Mạng truyền thông không dây Zigbee.
- ❖ Lập trình Web.

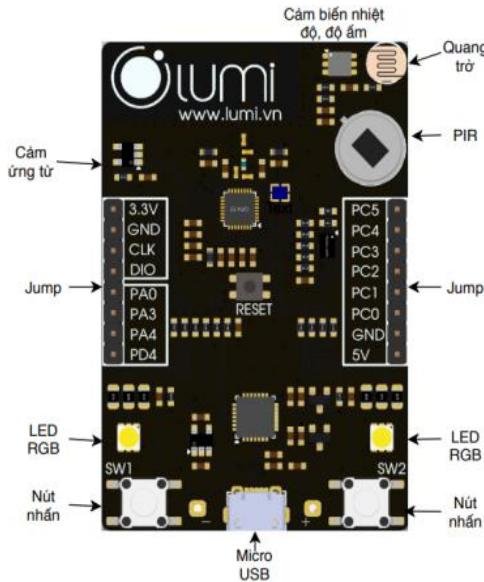
3. Tổng quan thiết bị Công tắc cầu thang

3.1. Thành phần Công tắc cầu thang

Thiết bị Công tắc cầu thang trong đồ án này bao gồm các thành phần sau:

- ❖ Sử dụng **giao diện Web** để thực hiện các tính năng của mạng Zigbee, điều khiển đèn cầu thang và cập nhật thông số của cảm biến.
- ❖ **01 kit ESP32** có nhiệm vụ tạo mạng và quản lý toàn bộ các thiết bị gia nhập mạng được gọi là Zigbee Coordinator (ZC). Trên kit EFR32, sử dụng phần cứng với mục đích như sau:
 - ◆ LED_RGB1 sử dụng để báo trạng thái mạng Zigbee.
 - ◆ LED_RGB2 sử dụng để báo mở mạng, đóng mạng Zigbee và thiết bị rời mạng.
 - ◆ SW1 sử dụng để tạo mạng, mở mạng, đóng mạng Zigbee.
 - ◆ SW2 sử dụng để xóa thiết bị trong mạng Zigbee.
 - ◆ Sử dụng cặp chân GPIO là PC1, PC2 cấu hình chức năng UART để giao tiếp với ESP32.
- ❖ **02 kit EFR32 (Zigbee Router – ZR)** đóng vai trò là 02 công tắc cầu thang (*Upper Switch & Below Switch*) có nhiệm vụ bật/tắt đèn cầu thang sử dụng đồng bộ các chế độ điều khiển Auto, Manual, cảm biến PIR, nút bấm cơ và điều khiển bằng trình duyệt Web. Trên mỗi kit EFR32, sử dụng phần cứng như sau:
 - ◆ 01 cảm biến phát hiện chuyển động PIR.
 - ◆ SW2 sử dụng cho các tính năng: điều khiển đèn cầu thang, chọn chế độ điều khiển AUTO-MANUAL, binding để điều khiển đồng bộ 2 công tắc và sử dụng để thiết bị rời mạng (*Trong đồ án này, tôi không sử dụng SW1 để lập trình các tính năng, do SW1 trên Kit hiện bị hỏng*).
 - ◆ LED_RGB1 sử dụng để báo chế độ hoạt động AUTO-MANUAL.

- ◆ LED_RGB2 sử dụng để báo chế độ tìm mạng, gia nhập mạng, rời mạng (*Ngoài ra, LED_RGB2 trên Below Switch còn được sử dụng làm đèn cầu thang*).
- ◆ Cảm biến nhiệt độ, độ ẩm, quang điện trở trên Below Zigbee để report thông số môi trường lên Web.



Hình 1.1: EFR32 Board Kit

- ❖ 01 module ESP32 đóng vai trò là Server trong giao tiếp giữa trình duyệt (*Client*) với ESP32 (*Server*) và được sử dụng để giao tiếp với ZC. Để truy cập vào Web điều khiển, chúng ta sẽ sử dụng địa chỉ IP mà ESP32 được cấp phát sau khi kết nối đến mạng Wifi khả dụng.



Hình 1.2: Vị điều khiển ESP-WROOM-32

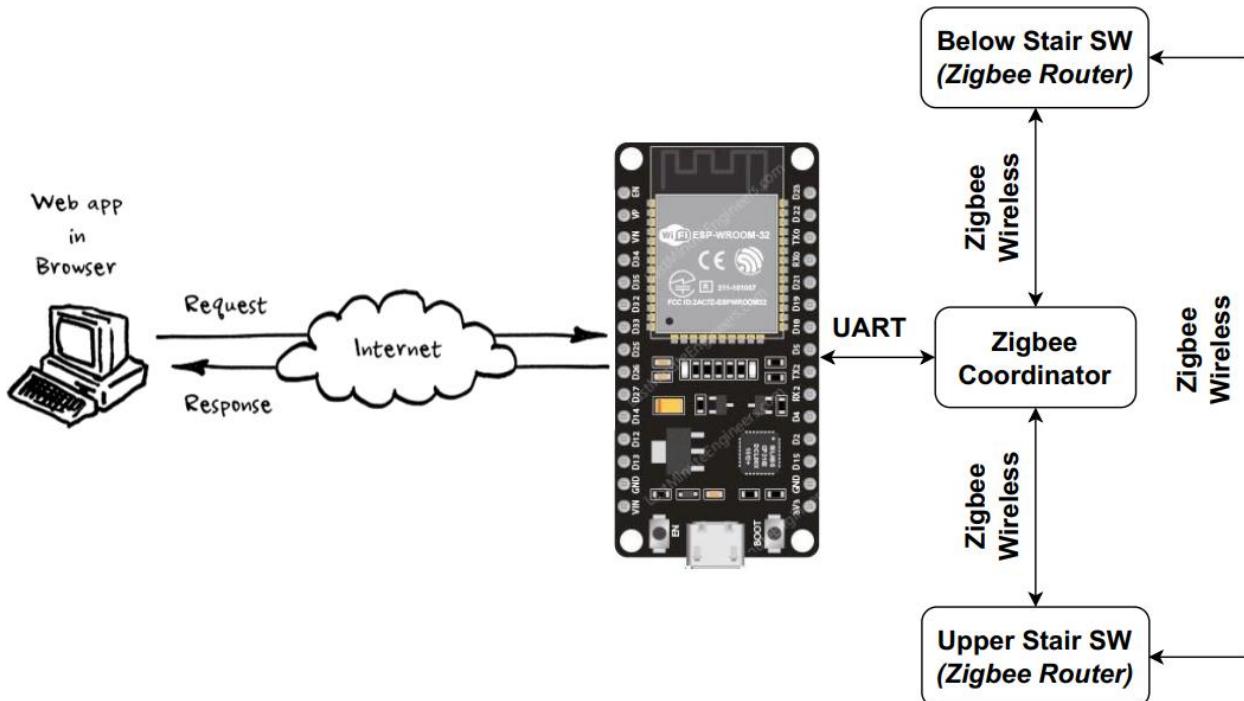
3.2. Tổng quan tính năng thiết bị Công tắc cầu thang

- ❖ Thực hiện việc tạo mạng, mở mạng, dừng mở mạng và xóa thiết bị khỏi mạng sử dụng SWs trên ZC và Web.
- ❖ Update trạng thái gia nhập, rời mạng của các SWs lên Web.

- ❖ Thiết lập 02 chế độ điều khiển sử dụng đồng bộ nút bấm cơ, Web và update chế độ điều khiển lên Web:
 - ◆ *Chế độ Auto:* Sử dụng Web + Nút bấm cơ + cảm biến chuyển động PIR để bật/tắt đèn cầu thang.
 - ◆ *Chế độ Manual:* Sử dụng Web + Nút bấm cơ để bật/tắt đèn cầu thang.
- ❖ Sử dụng nút bấm cơ điều khiển đồng bộ 2 công tắc để bật/tắt đèn cầu thang và update trạng thái đèn lên Web.
- ❖ Sử dụng cảm biến PIR điều khiển bật/tắt đèn cầu thang và update trạng thái đèn lên Web (*điều khiển đồng bộ với nút bấm cơ*).
- ❖ Sử dụng Web để điều khiển bật/tắt đèn cầu thang và update trạng thái đèn lên Web (*điều khiển đồng bộ với nút bấm cơ + PIR*).

→ **Đồ án này tôi xây dựng các tính năng dựa trên Sản phẩm Công tắc cầu thang thông minh tích hợp cảm biến của Công ty Nhà thông minh LUMI. Link sản phẩm: <https://lumi.vn/san-pham/cong-tac-cau-thang-tich-hop-cam-bien-2-trong-1.html>.**

3.3. Mô hình thiết bị Công tắc cầu thang



Hình 1.3: Mô hình thiết bị công tắc cầu thang

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

1. Mạng Zigbee

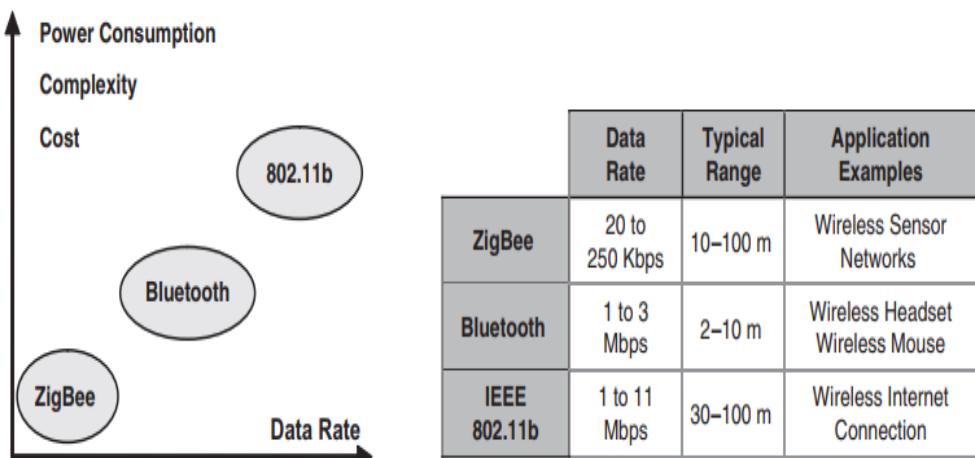
1.1. Tổng quan về mạng Zigbee

Mạng Zigbee được phát triển bởi Liên minh Zigbee (*Zigbee Alliance*), nơi mà có hàng trăm các công ty thành viên từ ngành công nghiệp bán dẫn, phát triển phần mềm cho đến các nhà sản xuất thiết bị. Zigbee Alliance được thành lập năm 2002, là liên hiệp các công ty làm việc cùng nhau để cho phép và kiểm soát các hoạt động mạng không dây tốc độ thấp, chi phí thấp, ít tiêu hao năng lượng và có tính bảo mật cao. Là một tổ chức độc lập và hợp tác phi lợi nhuận. Nó tạo ra các tiêu chuẩn cho Zigbee, cấp chứng nhận, chứng chỉ, phát triển thương hiệu, thị trường.

Zigbee là loại mạng không dây tầm ngắn - hoạt động trên 3 dải tần (868 MHz, 915 MHz và 2.4 GHz), tốc độ truyền tin thấp (*tốc độ tối đa là 250 kbps*) được sử dụng chủ yếu cho các thiết bị chạy bằng pin cái mà yêu cầu tốc độ truyền tin thấp, chi phí thấp và thời gian sử dụng của pin dài. Trong nhiều ứng dụng Zigbee, tổng thời gian mà các thiết bị không dây hoạt động là rất hạn chế, phần lớn thời gian chúng ở chế độ ngủ (*sleep mode*) để tiết kiệm năng lượng (*power-saving mode*). Chính vì vậy mà các thiết bị Zigbee có khả năng hoạt động một vài năm trước khi pin của chúng bị thay thế.

Một trong những ứng dụng của Zigbee là thiết bị giám sát bệnh nhân tại nhà bằng cách đo huyết áp, nhịp tim... thông qua các thiết bị được mang trên người. Thiết bị giám sát sử dụng các cảm biến để thu thập các thông tin liên quan đến sức khỏe. Sau đó, các thông tin này sẽ được truyền không dây đến máy chủ (*server*), ví dụ như máy tính cá nhân trong nhà bệnh nhân, nơi mà các phân tích ban đầu được thực hiện. Cuối cùng, các thông tin quan trọng được gửi đến y tá hoặc bác sĩ của bệnh nhân để phân tích thêm.

1.2. Zigbee, Bluetooth và IEEE 802.15.4



Hình 2.1: So sánh chuẩn Zigbee với Bluetooth và IEEE 802.11b

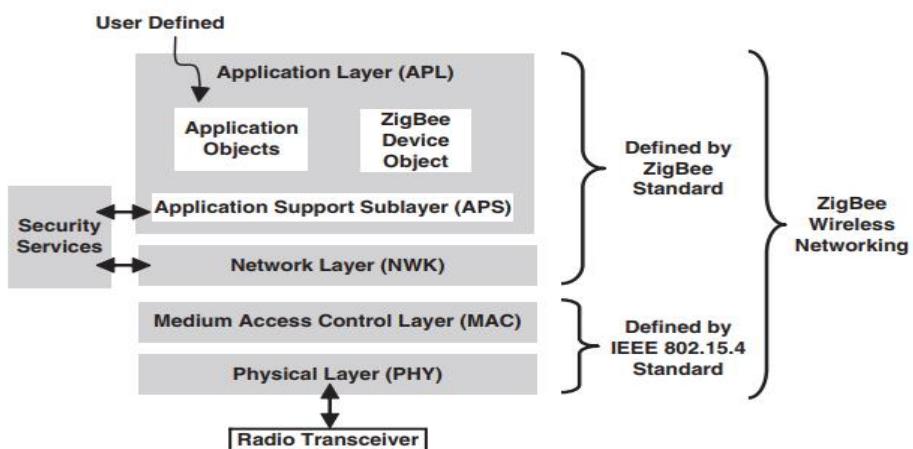
Việc so sánh giữa tiêu chuẩn Zigbee với Bluetooth và IEEE 802.11 WLAN sẽ giúp chúng ta hiểu về sự khác nhau giữa các tiêu chuẩn mạng không dây này. Tiêu chuẩn mạng Zigbee, Bluetooth hoạt động ở dải tần 2.4 GHz. Để so sánh với tiêu chuẩn IEEE 802.11 WLAN, phiên bản IEEE 802.11b được lựa chọn vì nó hoạt động cùng dải tần trên.

Đối với tiêu chuẩn IEEE 802.11b – ra đời vào tháng 7/1999, sử dụng tốc độ truyền tin cao (*lên đến 11 Mbps*) và một trong những ứng dụng điển hình của nó là cung cấp khả năng kết nối Internet không dây. Phạm vi truyền tải dữ liệu của tiêu chuẩn này nằm trong khoảng từ 30m đến 100m.

Đối với Bluetooth – ra đời vào 20/5/1999, được chuẩn hóa bởi Bluetooth Special Interest Group, sử dụng tốc độ truyền tải thấp hơn (*nhỏ hơn 3 Mbps*). Một trong những ứng dụng điển hình của nó là các tai nghe không dây nơi mà Bluetooth cung cấp phương tiện giao tiếp với một thiết bị di động. Phạm vi truyền tải dữ liệu của tiêu chuẩn này nằm trong khoảng từ 2m đến 10m.

Và cuối cùng là Zigbee – được hình thành vào năm 1998, đây là tiêu chuẩn có tốc độ truyền tin thấp nhất, độ phức tạp nhất trong 3 tiêu chuẩn mạng không dây này (*tốc độ tối đa là 250 kbps*). Tốc độ truyền tin thấp của Zigbee đồng nghĩa với việc nó không phải là lựa chọn tốt nhất cho việc thực hiện kết nối Internet không dây hoặc tai nghe không dây nơi mà tốc độ truyền tin yêu cầu phải lớn hơn 1 Mbps. Tuy nhiên, nếu mục đích của giao tiếp không dây là truyền nhận các lệnh đơn giản hoặc thu thập thông tin từ cảm biến, ví dụ như cảm biến nhiệt độ, độ ẩm thì Zigbee cung cấp hiệu quả nhất về mặt chi phí và năng lượng so với Bluetooth và IEEE 802.11b.

1.3. Mối quan hệ giữa Zigbee và chuẩn IEEE 802.15.4

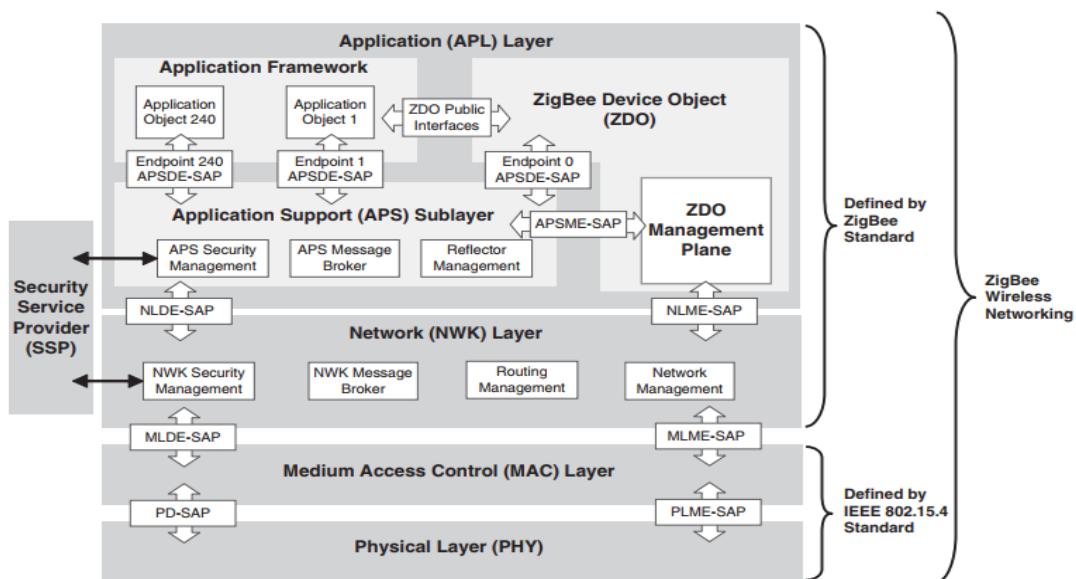


Hình 2.2: ZigBee Wireless Networking Protocol Layers

Một trong những phương pháp chung nhất để hình thành một mạng không dây hoặc có dây là sử dụng khái niệm về các tầng mạng. Mỗi tầng mạng chịu trách nhiệm cho một số chức năng trong mạng đó. Các tầng mạng thường chỉ truyền dữ liệu và lệnh trực tiếp đến các tầng ngay phía trên và phía dưới chúng. Việc phân chia một

giao thức mạng thành các tầng mạng đem lại một số ưu điểm. Ví dụ, nếu một giao thức mạng thay đổi theo thời gian, việc thay thế hoặc chỉnh sửa một tầng mạng, tầng mà bị ảnh hưởng bởi sự thay đổi đó sẽ dễ dàng hơn việc thay thế toàn bộ giao thức mạng. Ngoài ra, trong việc phát triển một ứng dụng, các tầng thấp hơn của giao thức mạng độc lập với ứng dụng và có thể được lấy từ bên thứ 3, vì vậy tất cả những thứ cần làm là tạo ra sự thay đổi trên tầng Application của giao thức.

Zigbee là một giao thức mạng không dây, do đó nó cũng tuân theo nguyên tắc trên, đó là phân chia thành các tầng mạng khác nhau. Việc phân chia này dựa trên mô hình tham chiếu OSI (*Open System Interconnect Basic Reference Model*). Trên hình 2.3 đã thể hiện, Zigbee chỉ định nghĩa các tầng Networking, Application và Security của giao thức này và sử dụng tầng PHY, MAC của tiêu chuẩn IEEE 802.15.4 như là một phần của giao thức mạng Zigbee. Mỗi tầng trong mạng Zigbee giao tiếp với tầng liền kề thông qua cơ chế Services Access Point (*SAPs*) và gồm 2 loại dịch vụ chính: Data (*Dịch vụ dữ liệu - Phụ trách phần giao tiếp*) và Management (*Dịch vụ quản lý - Chịu trách nhiệm cho các tính năng, chức năng của tầng đó*).



Hình 2.3: ZigBee Wireless Networking Protocol Layers (Detail)

1.3.1. Tầng Physical (PHY)

Tầng này đảm nhiệm một số chức năng sau:

- ❖ Bật/tắt bộ truyền nhận radio.
- ❖ Quét năng lượng trên các kênh tần để đánh giá, lựa chọn kênh tần tốt nhất cho mạng Zigbee.
- ❖ Xác định các chỉ số truyền nhận dữ liệu: LQI (*Link Quality Indicator*) – xác định chất lượng của gói dữ liệu được nhận, đây là một trong những thông tin để lựa chọn router; RSSI (*Received Signal Strength Indicator*) – chỉ báo cường độ tín hiệu nhận được.

- ❖ Quét năng lượng trên kênh truyền sử dụng cơ chế CS hoặc ED xác định kênh truyền bận/rảnh để các thiết bị trong mạng lựa chọn thời điểm phù hợp để gửi dữ liệu.
- ❖ Chuyển những dữ liệu ở dạng nhị phân thành sóng điện từ để thực hiện truyền dữ liệu và ngược lại chuyển thông tin từ sóng điện từ thành dạng nhị phân để chuyển thông tin lên các tầng trên phục vụ xử lý.

1.3.2. Tầng Medium Access Control (*MAC*)

Tầng **Medium Access Control** cung cấp 2 dịch vụ: MAC Data Service và MAC Management Service

- ❖ **MAC Data Service:** Cung cấp các dịch vụ giao tiếp do tầng NET yêu cầu. Có 3 sự lựa chọn cho việc truyền tải dữ liệu: Truyền tải dữ liệu sử dụng ACK, UNACK; Truyền tải dữ liệu trong khoảng thời gian GTS or CAP; Truyền tải dữ liệu trực tiếp hoặc gián tiếp.
- ❖ **MAC Management Service:** Trong tầng này MAC sẽ cung cấp những dịch vụ để tầng NET sử dụng như:
 - ◆ **MAC Reset:** Dịch vụ để tầng mạng sử dụng để xóa hết những thông tin đang được nắm giữ trong tầng MAC, đưa giá trị của chúng về mặc định.
 - ◆ **Device association & disassociation:** Sử dụng để tiến hành quá trình tham gia và rời mạng của thiết bị.
 - ◆ **Trạng thái kết nối:** Dịch vụ này giúp tầng mạng biết được trạng thái kết nối của thiết bị là như thế nào. Nếu có lỗi xảy ra nó cũng cung cấp các nguyên nhân gây ra lỗi.
 - ◆ **Kích hoạt/tắt kích hoạt bộ nhận Rx:** Dịch vụ này giúp tầng mạng có thể dễ dàng thao tác với bộ nhận Rx và dịch vụ này mang tính chất tùy chọn ở cả hai loại thiết bị FFDs và RFDs.
 - ◆ **Quản lý khe thời gian GTS tạo cấu trúc Superframe:** Được sử dụng để phân luồng dữ liệu để việc truyền nhận bản tin không bị xảy ra xung đột, thất thoát dữ liệu.

1.3.3. Tầng Network (*NWK*)

Tầng NWK cung cấp 2 dịch vụ: NWK Layer Data Entity Service và NWK Layer Management Entity Service

- ❖ **NWK Layer Data Entity Service:**
 - ◆ Nhận và xử lý dữ liệu từ tầng APS.
 - ◆ Đóng gói dữ liệu thành các Data Frame, khởi tạo các giá trị như Radius, Sequence number. Radius là giá trị thể hiện cho việc bản tin đó được chuyển tiếp máy lần trên đường truyền mạng hay nó có thể đi

qua bao nhiêu bước nhảy, bao nhiêu hops. Ví dụ Radius = 3, thì bản tin đó sẽ không được chuyển tiếp quá 3 lần khi được gửi từ thiết bị nguồn tới thiết bị đích. Sequence number là giá trị ban đầu được tạo bởi một số ngẫu nhiên ở thiết bị gửi và con số đó sẽ được tăng lên một đơn vị mỗi khi một data frame được truyền đi. Ở bên phía thiết bị nhận tầng mạng sẽ nhận về Data Frame, LQI, Sequence Number – Giá trị này sẽ được xử lý bằng cách giá trị mới phải lớn hơn giá trị cũ, nếu thấp hơn thì giá trị Data Frame sẽ được bỏ qua.

- ◆ Cung cấp các giao thức truyền gói tin: Broadcast, Multicast, Unicast.
- ◆ Lựa chọn cấu trúc mạng: Nó luôn luôn được đặt là Net Mesh working.

❖ **NWK Layer Management Entity Service:**

- ◆ Khởi tạo mạng.
- ◆ Gia nhập/rời mạng.
- ◆ Route discovery: Tìm kiếm định tuyến đường đi.
- ◆ Route maintenance: Duy trì đường định tuyến.

1.3.4. Tầng Application (APL)

❖ **Tầng ứng dụng gồm 3 phần chính.** Trong đó, tập trung nghiên cứu hai phần quan trọng nhất đó là Application Framework và Zigbee Device Object. Đối với Application Support Sublayer là một tầng phụ, nó phụ trách việc xử lý thông tin giao tiếp giữa tầng mạng và tầng ứng dụng. Phần xử lý cho phần này đã được nhúng sẵn trong phần stack của nhà sản xuất nên chúng ta không cần quan tâm đến phần này.

❖ **Application Framework:**

- ◆ Môi trường chứa các đối tượng ứng dụng được lưu trữ để kiểm soát và quản lý các tầng giao thức trong một nút mạng Zigbee. Đối tượng ứng dụng được phát triển bởi các nhà sản xuất và đó là nơi mà thiết bị được tùy chỉnh cho nhiều loại ứng dụng khác nhau và có thể có tới 240 đối tượng ứng dụng trong một thiết bị. Mỗi đối tượng ứng dụng sẽ được tồn tại tại một địa chỉ riêng biệt và được gọi là Endpoint. Endpoint sẽ được đánh số từ 1-240. Đối với Endpoint 0 thì là Endpoint đặc trưng và là địa chỉ được sử dụng bởi Zigbee Device Object.
- ◆ Ngoài ra, đối với mỗi Endpoint, thì chúng ta sẽ có thêm các thông tin về Zigbee Descriptor và Cluster. Cluster là một giá trị 16 bit – đại diện cho chức năng của sản phẩm, chứa danh sách các Attribute và phục vụ cho việc giữ trạng thái và điều khiển thiết bị.

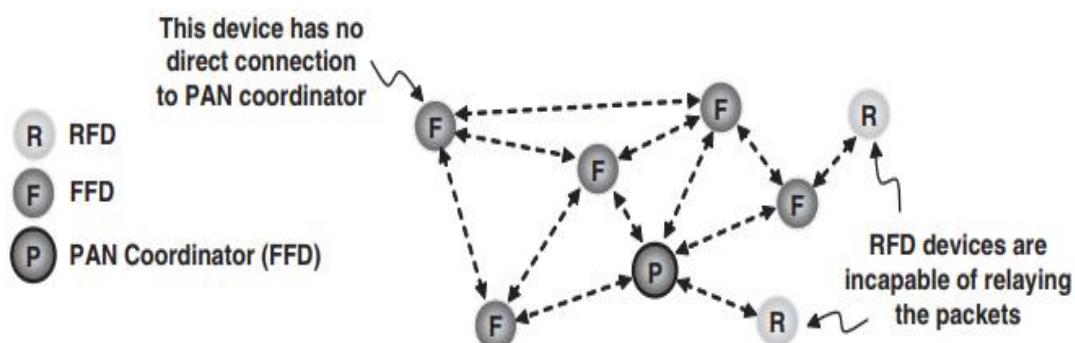
❖ **Zigbee Device Object (ZDO):** Là nơi chứa tính năng chung cho toàn bộ cả thiết bị.

1.4. Loại thiết bị và vai trò của thiết bị trong mạng Zigbee

1.4.1. Loại thiết bị trong mạng Zigbee

Trong mạng Zigbee có hai loại thiết bị đó là FFD (*Full-Function Device*) và RFD (*Reduced-Function device*)

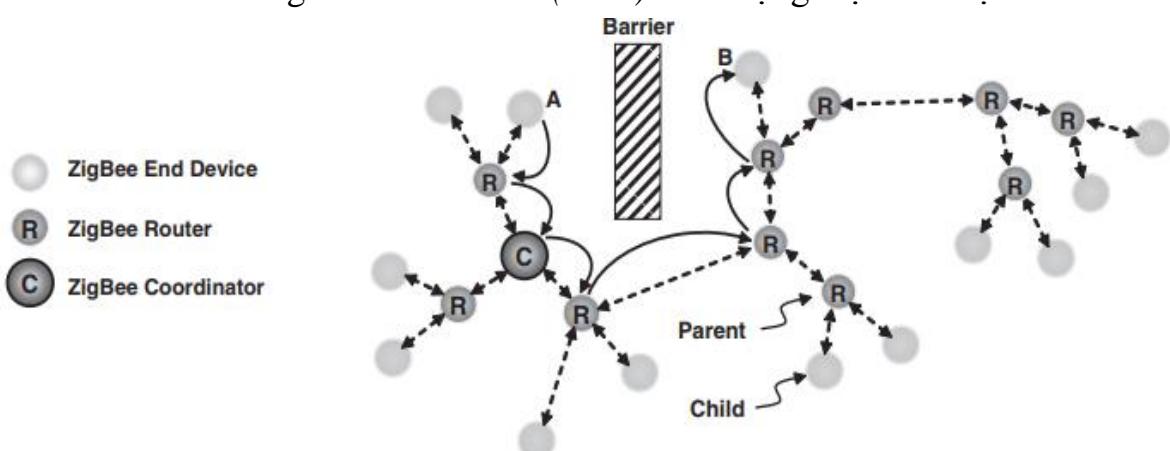
- ❖ **Full-Function Device:** Đây là loại thiết bị có đầy đủ tất cả các tính năng tồn tại trong mạng như kết nối giao tiếp với các nút mạng và đặc biệt là có khả năng chuyển tiếp bản tin. Những nút mạng màu xanh lá cây là các FFDs.
- ❖ **Reduced-Function Device:** Đây là loại thiết bị hạn chế chức năng và nó chỉ có thể giao tiếp với một thiết bị FFD duy nhất và không có khả năng chuyển tiếp bản tin. Một thiết bị FFD thì có thể kết nối với 1 hoặc nhiều RFDs nhưng một thiết bị RFD thì chỉ có thể kết nối duy nhất đến một thiết bị FFD. Khi bản tin muốn gửi đến được thiết bị RFD thì nó bắt buộc phải đi qua đường có thiết bị FFD giao tiếp trực tiếp với nó. Các nút mạng màu xanh nước biển chính là RFDs.



Hình 2.4: Loại thiết bị trong mạng Zigbee

1.4.2. Vai trò của thiết bị trong mạng Zigbee

- ❖ Trong mạng Zigbee các thiết bị có 3 vai trò: Zigbee Coordinator (ZC) – Sử dụng loại thiết bị FFD, Zigbee Router (ZR) – Sử dụng loại thiết bị FFD và Zigbee End Device (ZED) – Sử dụng loại thiết bị RFD.



Hình 2.5: Vai trò của thiết bị trong mạng Zigbee

❖ **Zigbee Coordinator (ZC):** Thiết bị giữ vai trò quản lý toàn mạng, là bộ điều khiển chính. Một mạng Zigbee chỉ tồn tại duy nhất 1 thiết bị được gọi là ZC. Thiết bị ZC thực hiện các chức năng sau:

- ◆ Chịu trách nhiệm trong việc lựa chọn kênh phù hợp để tạo mạng sau khi thực hiện xong quá trình quét kênh thực hiện ở tầng PHY. Sau quá trình quét kênh hoàn tất, ZC sẽ biết được kênh nào đang tồn tại nhiều, kênh nào ít nhiễu từ đó nó sẽ lựa chọn kênh phù hợp cho chính mạng của nó để ít bị ảnh hưởng từ môi trường bên ngoài nhất.
- ◆ ZC tạo ra mạng Centralized Network. Nó phải là thiết bị đầu tiên tồn tại trong mạng.
- ◆ ZC mỗi khi tạo mạng xong sẽ luôn có một địa chỉ mạng cố định 0x0000. Địa chỉ này được áp dụng cho tất cả các loại mô hình mạng Zigbee.
- ◆ Quy định thông tin cho chính mạng của mình: PAN-ID, Extended PanID, Trust Center Link Key, Network Key. Đây là những thông tin thiết yếu cho một mạng giúp ZC quản lý cũng như tăng cường tính bảo mật trong mạng của mình.
- ◆ Quản lý các hoạt động: Gia nhập, trao đổi thông tin.
- ◆ Thiết lập các tính năng để tăng cường tính bảo mật cho mạng của mình.

❖ **Zigbee Router (ZR) đảm nhiệm các chức năng sau:**

- ◆ Truyền bản tin.
- ◆ Nhận bản tin.
- ◆ Chuyển tiếp bản tin: Khi nhận được một bản tin, tầng MAC, NET sẽ kiểm tra xem nó có phải là bản tin được gửi đến thiết bị của mình hay không. Nếu không nó sẽ thực hiện chuyển tiếp bản tin cho các thiết bị lân cận của mình.
- ◆ 1-hop neighbor: Là những thiết bị mà Router có thể gửi trực tiếp bản tin đến nó. Trong quá trình hoạt động của mạng Router sẽ lưu và cập nhật những thiết bị Neighbor của mình để có thể phục vụ cho bài toán định tuyến đường đi.
- ◆ Cung cấp dịch vụ định tuyến đường đi cho hạ tầng mạng Zigbee.
- ◆ Chịu trách nhiệm về lưu thông mạng.
- ◆ Là thiết bị được thiết kế ở chế độ luôn luôn thức vì vậy những thiết bị sử dụng PIN không phù hợp cho Router.

❖ **Zigbee End Device (ZED):** Không tham gia vào giao thông mạng tức là không tham gia vào quá trình chuyển tiếp bản tin trong mạng. Chỉ giao tiếp được với 1 và chỉ một thiết bị có khả năng chuyển tiếp bản tin. Nó có thể là kết nối với Router hoặc là kết nối trực tiếp đến ZC và các thiết

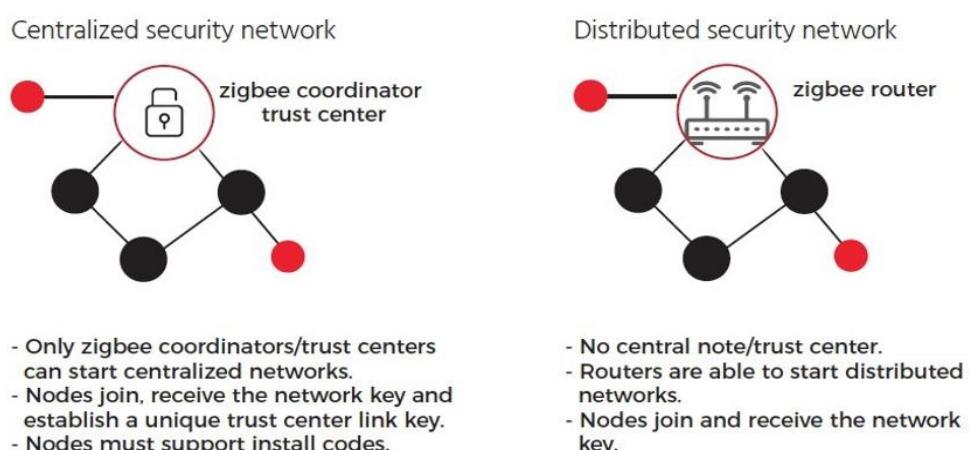
bị này được gọi là parent của ZED. ZED sẽ tham gia vào quá trình giao tiếp trong mạng thông qua Parent của nó tức là một thiết bị muốn truyền đến hoặc truyền đi từ thiết bị phải thông qua Parent. Khi mới vào mạng thì ZED sẽ lựa chọn thiết bị nào có LQI với nó lớn nhất để lựa chọn là parent và thông thường đó là những vị trí gần nó nhất. ZED được chia làm 2 loại:

- ◆ **Sleepy End Device:** Là thiết bị luôn ngủ, chỉ thức dậy theo định kỳ để hỏi parent xem có bản tin nào được gửi đến nó hay không hoặc là thức dậy bắt buộc tùy thuộc vào người lập trình để kiểm tra xem có bản tin nào cần được gửi đi hay không. Khi thức dậy ZED muốn kiểm tra xem có bản tin nào được gửi đến nó hay không, thì nó phải thực hiện một thao tác đó là gửi một bản tin Poll đến cho parent nếu không sẽ không có một bản tin nào gửi đến cho nó cho đến khi parent nhận được bản tin Poll này.
- ◆ **NoSleepy End Device:** Được thiết kế là một thiết bị luôn thức, tuy nhiên nó không tham gia vào quá trình giao thông trong mạng.

1.5. Zigbee 3.0

1.5.1. Zigbee 3.0 là gì

Giao thức không dây Zigbee được chia thành một số mô hình ứng dụng như sau: Zigbee Home Automation (*HA 1.2*) - Dùng cho các ứng dụng cho Home như đèn, công tắc, cảm biến... mà cần sự quản lý của Coordinator; Zigbee Light Link - Dùng cho những ứng dụng đơn giản như Switch điều khiển trực tiếp đèn mà không cần Coordinator; Zigbee Green Power - Dành cho các ứng dụng siêu tiết kiệm năng lượng và Zigbee 3.0. Trong các loại mô hình ứng dụng trên, Zigbee 3.0 là phiên bản hoàn chỉnh nhất ra đời năm 2015. Nó là tổng hợp của cả 3 mô hình bên trên. Mô hình này gồm 2 loại mạng:



Hình 2.6: Mạng Centralized and Distributed network

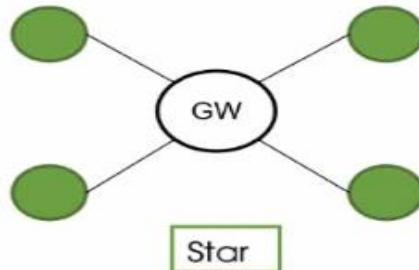
- ❖ **Distributed Network:** Là loại mạng được tạo ra bởi một nút mạng Zigbee. Loại mạng này giống như sử dụng Zigbee Light Link. Tùy thuộc vào cấu hình khác nhau mà những thiết bị sẽ lần lượt được lựa chọn thành Initiator or Target
- ❖ **Centralized Network:** Đây là loại mạng được tạo ra bởi một thiết bị mạng trung tâm Gateway. Loại mạng này có mô hình giống như Zigbee HA 1.2. Tuy nhiên tồn tại trong mạng này, chúng ta có thể cấu hình thêm vào các thiết bị động giống như Green power bằng cách kết nối các thiết bị sử dụng Green power với nhau.

Mặc dù mô hình hoạt động vừa giống **Zigbee HA 1.2** vừa giống **Zigbee Green Power** nhưng nó có sự cải tiến lớn trong một số tính năng đặc biệt như tăng cường các thách thức tạo mạng và tìm mạng Zigbee, tăng cường khả năng bảo mật và cải thiện thuật toán tìm đường đi của các thiết bị trong Mesh Networking.

1.5.2. Các mô hình mạng của Zigbee

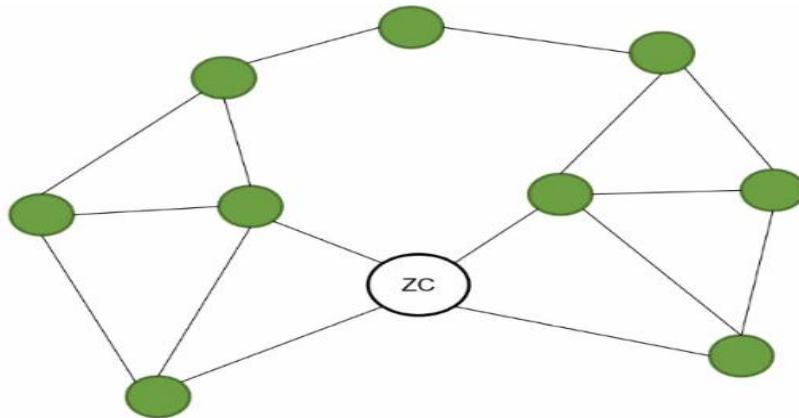
Zigbee hoạt động dựa trên 3 mô hình mạng:

- ❖ **Star Network:** Trong mô hình Star Network, các thiết bị trong mạng chỉ có thể kết nối đến thiết bị Zigbee Coordinator. Nó là thiết bị trung tâm kiểm soát toàn bộ hành động của một mạng. Trong mạng Star Network, các thiết bị khi gia nhập vào mạng không được cài đặt các chức năng chuyển tiếp bản tin.



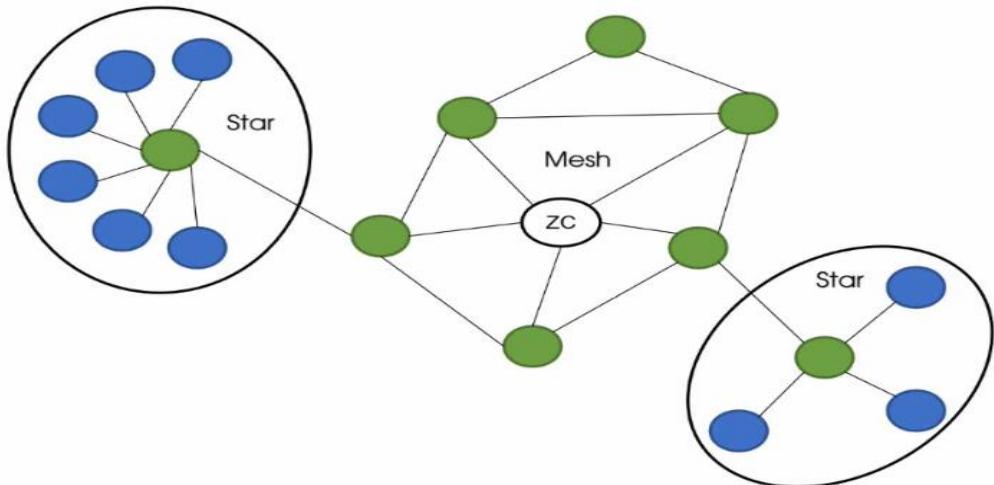
Hình 2.7: Mô hình Star Network

- ❖ **Full Mesh Network:** Trong mô hình mạng Full Mesh Network, các thiết bị ngoài việc kết nối trực tiếp với thiết bị trung tâm, các nút mạng có thể liên kết trực tiếp với nhau. Mỗi nút mạng có thể hoạt động giống như Router bao gồm cả ZC khi nó đã thực hiện xong việc tạo mạng. Các nút mạng đều có thể truyền, nhận và chuyển tiếp bản tin, nên đây có thể nói là mô hình chật chẽ nhất. Hơn thế nữa nó có thể linh hoạt tạo ra một liên kết mới khi liên kết cũ gặp sự cố. Tuy nhiên trong một mạng dày đặc các nút thực hiện chức năng Router, sẽ có nhiều nhiễu xảy ra giữa các thiết bị, ảnh hưởng đến khả năng truyền tin của chúng. Để giải quyết bài toán này chúng ta phải sử dụng kết hợp giữa Router và ZED.



Hình 2.8: Mô hình Full Mesh Network

❖ **Hybrid Mesh Network:** Mô hình mạng Hybrid Mesh Network thừa kế tất cả ưu điểm của cấu trúc mạng Star Network, Full Mesh Network và có thể tùy biến những nhược điểm của hai mạng này thành tính năng của mình. Ví dụ như nhược điểm độ mở rộng của mạng sao đã được mô hình mạng này khắc phục. Khi Zigbee muốn truyền từ ZC xuống ZED nó không cần phải truyền thẳng xuống ZED mà chỉ cần truyền đến Router của nó, việc còn lại do chính Router thực hiện. Mô hình này cho phép thiết kế mạng ở những khoảng cách rộng và khả năng thiết kế phân cấp nhiều hơn so với cấu trúc mạng Mesh. Chính là số ZED mà một Router có thể quản lý ổn định.



Hình 2.9: Mô hình Hybrid Mesh Network

1.5.3. Khởi tạo mạng, gia nhập mạng

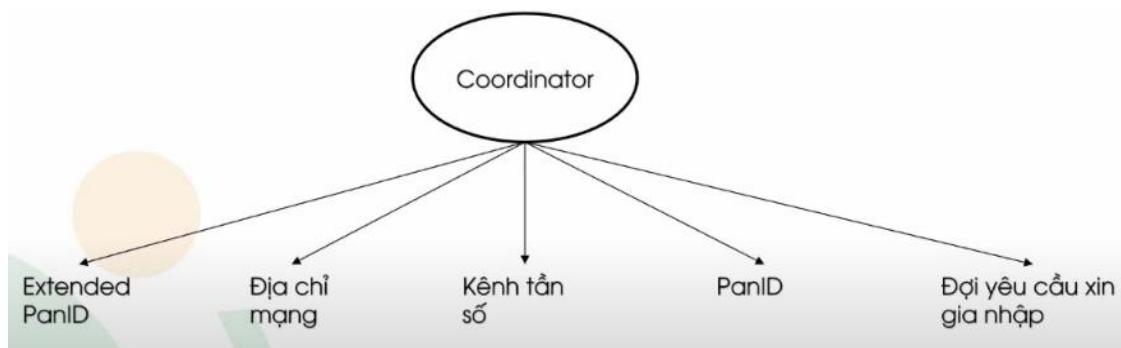
❖ **Khởi tạo mạng**

- ◆ Coordinator được khởi tạo đầu tiên.
- ◆ Cài đặt địa chỉ Extended PanID cho mạng của mình dựa theo tầng ứng dụng của ZC. Người lập trình khi muốn đặt giá trị này thì phải đặt nó trên tầng ứng dụng. Và khi khởi tạo mạng, ZC sẽ lấy luôn giá trị này

để làm Extended PanID cho mình. Và nếu người lập trình không lập trình giá trị này, thì giá trị này luôn được để mặc định là 0. Khi khởi tạo mạng, ZC sẽ tự động kiểm tra, nếu là giá trị mặc định thì nó sẽ tự động quét năng lượng và khởi tạo ra giá trị duy nhất Extended PanID. Khi sử dụng chúng ta không nên cài đặt trước giá trị này mà nên để ZC sử dụng thuật toán để chọn ra giá trị tốt nhất hoặc trừ khi bạn phải kiểm soát hoặc tính toán chính xác ra số lượng Extended PanID có trong mạng. Vì đây phải là một giá trị đảm bảo duy nhất trên kênh tần. Và nó không được xử lý khi phát hiện có sự trùng lặp giá trị. Nên nếu để xảy ra trùng lặp Extended PanID thì nó là một thảm họa đối với Zigbee và không xử lý được.

- ◆ Sau khi lựa chọn giá trị Extended PanID xong, ZC sẽ tự động cấu hình cho địa chỉ mạng của mình thành 0x0000 và đây là giá trị cố định của ZC trong tất cả các mạng Zigbee
- ◆ ZC thực hiện việc quét năng lượng trên 16 kênh tần để tìm ra kênh ít nhiễu, năng lượng sạch nhất để làm kênh tạo cho mạng của mình.
- ◆ Sau khi kênh tần được lựa chọn thì ZC sẽ lựa chọn ra địa chỉ 16 bit cho định danh Pan ID của mình. Nó sẽ nghe các luồng mạng khác đang tồn tại và xác định địa chỉ Pan ID của các luồng mạng này. Để tránh xung đột xảy ra, ZC sẽ lựa chọn một Pan ID không trùng lặp với những định danh đang tồn tại.

→ Mạng được tạo thành công bởi ZC và công việc tiếp theo của nó là nhận và không nhận yêu cầu gia nhập mạng của ZR và ZED.

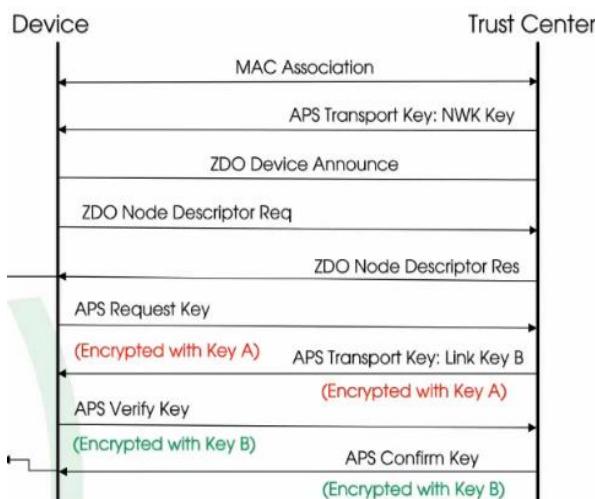


Hình 2.10: Các bước khởi tạo mạng của Zigbee Coordinator

❖ Gia nhập mạng

ZC cần phải thực hiện quá trình mở mạng, Device nó cần phải thực hiện quá trình đi tìm mạng của mình. Cả hai thiết bị này phải sử dụng bản tin MAC Association để có thể thực hiện quá trình gia nhập mạng. Đối với ZC khi thực hiện quá trình mở mạng và nhận bản tin MACA đến từ thiết bị thì nó sẽ gửi lại cho thiết bị một bản tin APS... để gửi NWK đến cho thiết bị để phục vụ cho quá trình mã hóa

và giải mã. Sau khi thiết bị nhận được bản tin APS... thì nó sẽ gửi lại cho ZC bản tin ZDO Device Announce – Là một bản tin Broadcast và đây là bản tin thông báo cho toàn bộ mạng về thông tin thiết bị của nó. Tiếp đó thiết bị sẽ gửi bản tin ZDO Node... cho ZC để có thể nhận về những thông tin của thiết bị ZC và kiểm tra xem thiết bị ZC có phải là thiết bị Zigbee R21 hay không. Bởi vì chỉ có Zigbee R21 thì mới có khả năng cung cấp TCLK. Nếu thiết bị là Zigbee R21, thì Device sẽ gửi bản tin APS request key để yêu cầu TC cung cấp giá trị TCLK cho mình. Bản tin này được mã hóa bởi NWK. Sau khi TC nhận được bản tin cung cấp TCLK, thì nó sẽ gửi lại cho Device một TCLK mới. Khi Device nhận được TCLK mới thì nó sẽ lưu trữ và xử lý các bản tin của mình bằng loại Key mới này và gửi lại cho TC bản tin APS Verify Key nhằm mục đích xác nhận với TC rằng nó đã nhận được Key và sẽ sử dụng loại Key này cho việc giao tiếp. Khi TC nhận được bản tin đó, thì nó sẽ sử dụng loại mã mới để giải mã bản tin đó và cũng sẽ gửi lại bản tin Confirm Key để xác nhận sẽ sử dụng TCLK mới cho giao tiếp thay vì NWKA.

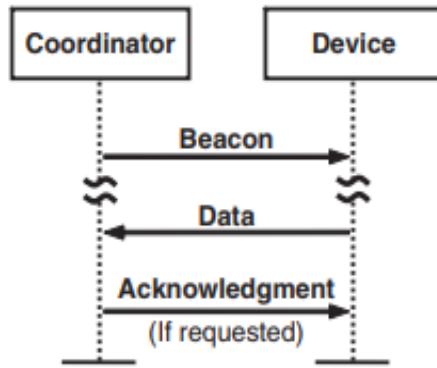


Hình 2.11: Quá trình gia nhập mạng

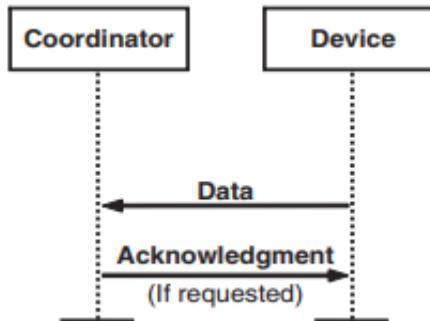
1.5.4. Hướng giao tiếp trong mạng Zigbee

Trong mô hình mạng Zigbee có 3 hướng giao tiếp:

- ❖ **Coordinator ← Device:** Đối với mạng sử dụng BEACON, khi thiết bị muốn gửi dữ liệu đến Coordinator thiết bị đầu tiên sẽ phải đồng bộ xung thời gian thường xuyên và để gửi bản tin đến cho Coordinator, tùy thuộc Co có cung cấp khe thời gian GTS cho thiết bị hay không, nếu có thì thiết bị sẽ truyền bản tin trong khoảng thời gian CFP, nếu không thiết bị sẽ phải sử dụng cơ chế CSMA/CA để xác định xem kênh truyền có đang rảnh rỗi để truyền hay không. Sau khi Co nhận được bản tin dữ liệu thì nó sẽ gửi lại bản tin ACK cho thiết bị. Trong mạng không sử dụng BEACON dữ liệu chỉ được gửi khi và chỉ khi kênh truyền đang rảnh rỗi và nó phải áp dụng cơ chế CSMA/CA.

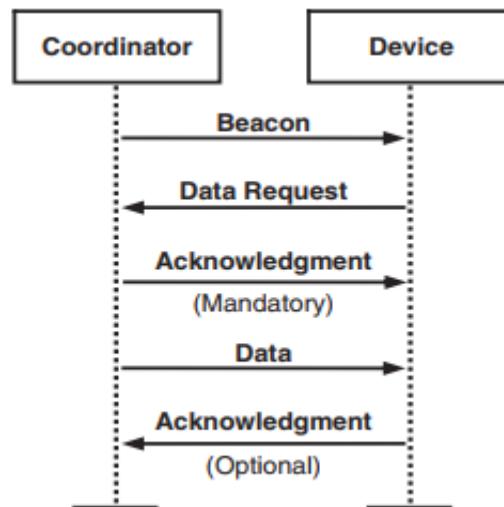


Hình 2.12 : Quá trình giao tiếp giữa Device & ZC sử dụng Beacon



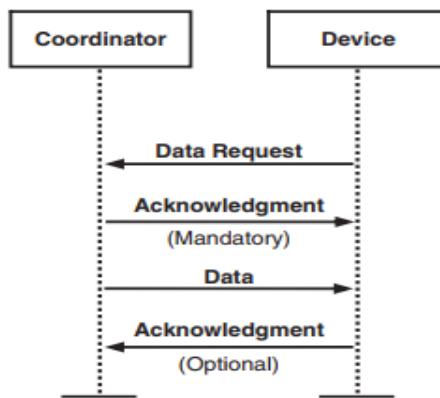
Hình 2.13: Quá trình giao tiếp giữa Device & ZC không sử dụng Beacon

- ❖ **Coordinator → Device:** Khi Co có ý định gửi dữ liệu đến cho một thiết bị cụ thể nó sẽ biểu thị bản tin BEACON được gửi đến cho thiết bị rằng nó đang có bản tin chờ để gửi đến cho thiết bị đó. Sau đó thiết bị sẽ gửi một bản tin Data Request đến cho Co để báo hiệu rằng nó đã sẵn sàng để nhận dữ liệu. Co trước tiên sẽ gửi lại bản tin ACK báo hiệu cho thiết bị rằng nó đã chấp nhận Data request và sau đó nó sẽ gửi dữ liệu đến cho thiết bị. Khi thiết bị nhận được dữ liệu nó sẽ gửi lại ACK cho Co và báo hiệu rằng dữ liệu đã nhận thành công.



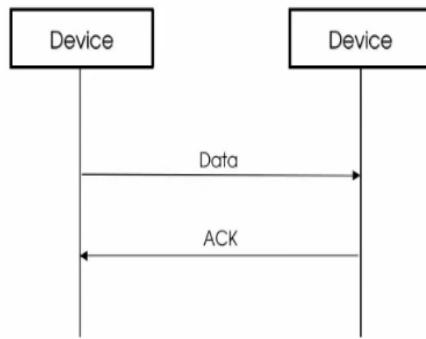
Hình 2.14: Quá trình giao tiếp giữa ZC & Device sử dụng Beacon

Trong mạng không sử dụng BEACON, Co sẽ phải đợi cho đến khi thiết bị gửi dữ liệu Data request. Ở đây có 2 trường hợp xảy ra, nếu thiết bị gửi Data request nhưng Co không có dữ liệu gửi đến thiết bị đó thì nó sẽ gửi lại bản tin ACK với nội dung đặc biệt báo hiệu rằng không có dữ liệu nào đang chờ để gửi đến cho thiết bị hoặc nó sẽ gửi đến cho thiết bị một dữ liệu với độ dài dữ liệu bằng 0. Trong trường hợp còn lại nếu có dữ liệu gửi cho thiết bị, thì đầu tiên nó sẽ gửi lại ACK để báo hiệu rằng nó chấp nhận Data request và sau đó sẽ gửi bản tin cần gửi đến cho thiết bị. Khi thiết bị nhận được dữ liệu nó cũng sẽ gửi bản tin ACK đến cho Co.



Hình 2.15: Quá trình giao tiếp giữa ZC & Device không sử dụng Beacon

❖ **Thiết bị ↔ Thiết bị:** Đây là hình thức giao tiếp Peer to Peer, trong mạng BEACON hay NONBEACON thì thiết bị sẽ phải sử dụng cơ chế CSMA/CA để thực hiện truyền bá tin. Khi một thiết bị muốn truyền tin đến một thiết bị khác thì đầu tiên nó cần phải sử dụng thuật toán CSMA/CA để xác định xem kênh truyền có đang rảnh rỗi hay không, nếu có thì nó sẽ gửi bản tin đến thiết bị nhận và khi thiết bị nhận nhận được nó cũng sẽ trả lại bản tin ACK.



Hình 2.16: Quá trình giao tiếp giữa 2 devices

1.5.5. Định danh mạng Zigbee

Khi một thiết bị tồn tại trong một mạng nó cần phải có một định danh cụ thể để báo hiệu nó đang tồn tại và sẵn sàng thực hiện quá trình kết nối đến các thiết bị khác. Tương tự trong Zigbee, khi một mạng được tạo ra bởi ZC, nó cũng cần có một định

danh cụ thể để nhận dạng. Điều này cho phép nhiều mạng Zigbee cùng hoạt động trong một dải tần mà không bị ảnh hưởng lẫn nhau. Zigbee sử dụng 2 loại nhận diện:

- ❖ **Pan ID:** Sử dụng giá trị định danh 16 bit sử dụng giao tiếp trong mạng để các nút biết được chúng đang ở trong cùng một mạng. Giá trị Pan ID được khởi tạo ngẫu nhiên bởi ZC khi khởi tạo mạng. Khi các nút gia nhập mạng nó phải ghi nhớ giá trị Pan ID và sử dụng nó để tiến hành giao tiếp trong mạng. Cũng giống như việc chọn kênh ZC phải đo tín hiệu trong kênh của mình, sau đó nó sẽ lựa chọn giá trị Pan ID sao cho không trùng lặp với các giá trị quét được. Nhưng trong một số trường hợp, khi đang hoạt động nếu ZC phát hiện ra giá trị Pan ID bị trùng lặp thì nó sẽ tự động thay thế giá trị này.
- ❖ **Extended ID:** Định danh sử dụng 64 bit để tạo mạng. Giá trị này có thể được định nghĩa trước bởi người dùng hoặc ZC sẽ sử dụng thuật toán quét năng lượng để lựa chọn ra giá trị tốt nhất cho mạng của mình. Khác với Pan ID thì Extended ID là giá trị duy nhất không bị thay đổi theo thời gian cho dù ở bất kỳ trường hợp nào. Khi một nút gia nhập thành công nó sẽ ghi nhớ giá trị Extended ID của mạng và sẽ sử dụng giá trị này khi có bất cứ sự thay đổi của mạng hay mỗi khi mạng xảy ra vấn đề.

1.5.6. Địa chỉ mạng của thiết bị trong mạng Zigbee

Mỗi thiết bị trong mạng đều có một địa chỉ riêng. Có hai phương thức định địa chỉ cho các thiết bị trong mạng :

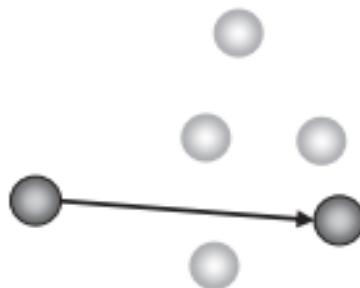
- ❖ **Địa chỉ IEEE:** Là loại địa chỉ 64 bit, được quy định và cung cấp bởi IEEE và nó là loại định danh duy nhất cho thiết bị. Địa chỉ IEEE của thiết bị là địa chỉ tồn tại duy nhất trên thế giới, sẽ không có một thiết bị thứ hai nào được sinh ra với giá trị địa chỉ này. Trên thực tế loại địa chỉ này được gọi là MAC hay Extended address. Đây chính là địa chỉ được tàng MAC lưu trữ và sử dụng.
- ❖ **Địa chỉ mạng:** Cho phép các nút mạng giao tiếp nội bộ trong một mạng đơn túc là địa chỉ này có thể tồn tại trên nhiều mạng Zigbee khác nhau. Ví dụ mỗi Co đại diện cho một mạng Zigbee và các Co khi tạo mạng sẽ luôn có giá trị là 0x0000. Vì vậy 0x0000 được lặp lại ở các mạng khi được tạo, tương tự với giá trị khác. Địa chỉ này có giá trị trong nội bộ một mạng cụ thể. Tương tự như biến global - IEEE, local – địa chỉ mạng trong lập trình. Việc sử dụng địa chỉ mạng sẽ giúp giảm tải được độ dài của bản tin, tiết kiệm được bộ nhớ cần thiết được cấp phát để lưu trữ địa chỉ và nó có thể được gọi với tên là short address. Vì vậy thường khi giao tiếp trong mạng chúng ta sẽ sử dụng loại địa chỉ mạng để giao tiếp giữa các nút mạng. Tuy nhiên có những ứng dụng hay loại bản tin liên quan đến

độ chính xác tuyệt đối thì chúng ta nên sử dụng loại địa chỉ MAC để đạt hiệu quả chính xác nhất.

1.5.7. Các phương thức truyền tin trong mạng Zigbee

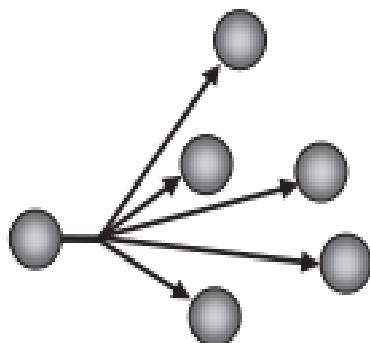
Mạng Zigbee có 04 phương thức truyền tin:

- ❖ **Phương thức truyền tin Unicast:** Đây là phương thức truyền tin Peer-To-Peer phục vụ cho việc truyền tin 1-1 giữa các thiết bị, sử dụng địa chỉ mạng để thực hiện việc truyền bản tin. Đây là một kiểu truyền tin mặc định trong Zigbee nếu thiết bị này muốn gửi bản tin đến thiết bị khác.



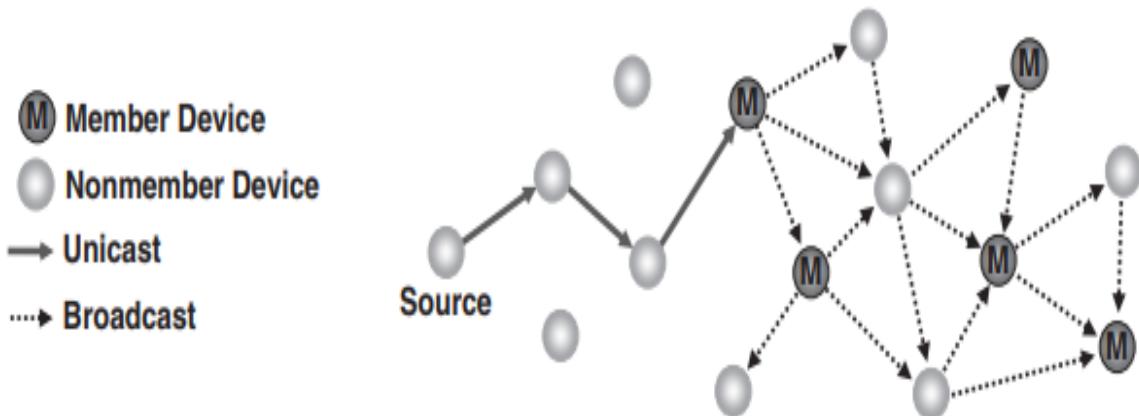
Hình 2.17: Phương thức truyền tin Unicast

- ❖ **Phương thức truyền tin Broadcast:** Bản tin sẽ được nhận bởi tất cả các thiết bị có trong cùng một mạng trên cùng một kênh tần số. Đối với một thiết bị cụ thể, với mỗi bản tin nhận được thì nó sẽ kiểm tra đây có phải bản tin được gửi đến nó hay không nhờ bóc tách các thông tin mạng có trong từng tầng OSI, cụ thể đó là thông tin về địa chỉ mạng. Nếu địa chỉ mạng được bóc tách là địa chỉ của chính thiết bị đó thì nó sẽ giữ lại bản tin và xử lý nếu không thì nó sẽ thực hiện việc chuyển tiếp bản tin. Trong Broadcast, địa chỉ mạng được sử dụng và đặt là 0xFFFF và địa chỉ 0xFFFF là địa chỉ được tất cả các thiết bị chấp nhận mỗi khi nhận được bản tin và chúng coi địa chỉ 0xFFFF như là một địa chỉ riêng, hợp pháp của mình.



Hình 2.18: Phương thức truyền tin Broadcast

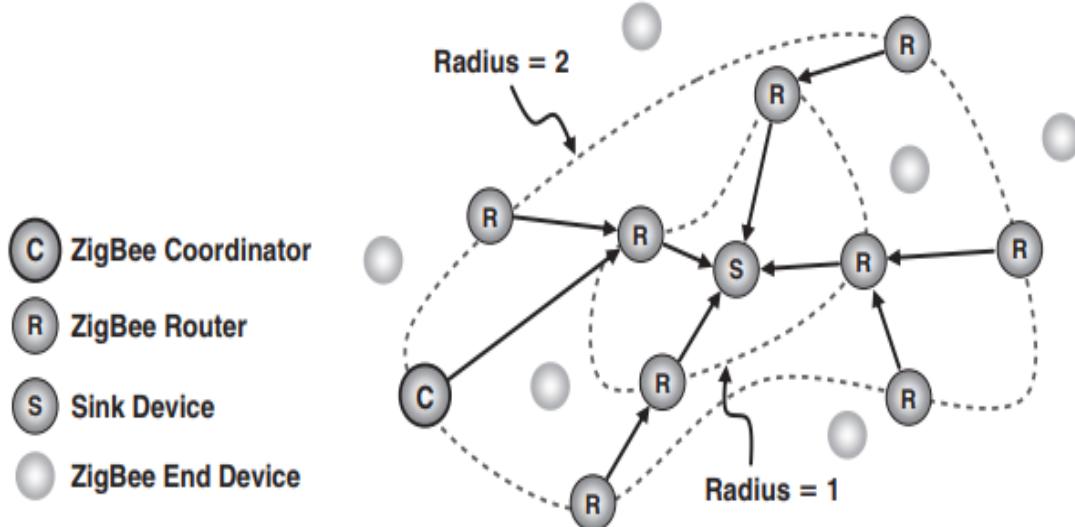
❖ **Phương thức truyền tin Multicast:** Là hình thức gửi bản tin đến một nhóm thiết bị cụ thể trong mạng. Mỗi nhóm thiết bị được định danh bởi một giá trị 16 bit được gọi là Group ID và các thiết bị trong một nhóm được gọi là Group Member. Mỗi một thiết bị có thể là thành viên của nhiều nhóm khác nhau. Một thiết bị nếu không phải là một thành viên trong nhóm thì vẫn có thể sử dụng Multicast để gửi bản tin đến nhóm đó. Trong quá trình hoạt động có hai kiểu hoạt động của thiết bị: Member Mode – Multicast sẽ được tạo bởi thiết bị thành viên và gửi chúng đến cho các thiết bị khác cùng nhóm, NonMemberMode – Thiết bị không trực thuộc nhóm hoạt động nhưng nó sẽ có trách nhiệm trong việc tạo đường định tuyến cho bản tin để nhóm đó có thể nhận được dữ liệu. Trong frame của tầng mạng, có thông số để kiểm tra rằng dữ liệu được gửi từ thiết bị Member hay NonMember. Một thiết bị khi nhận được bản tin nó cũng sẽ biết nó có thuộc nhóm được gửi hay không vì trong Payload trên tầng ứng dụng nó sẽ có giá trị Group ID được gửi. Giá trị này sẽ được thiết bị kiểm tra xem có trùng với nhóm nào của mình hay không.



Hình 2.19: Phương thức truyền tin Multicast

❖ **Phương thức truyền tin Many-To-One (được sử dụng chủ yếu trong tìm kiếm đường định tuyến):** Đây là hình thức gửi bản tin từ nhiều thiết bị đến một thiết bị duy nhất. Thiết bị nhận bản tin được gọi là Sink Device. Trong Zigbee thì Sink chính là thiết bị thành lập các đường định tuyến từ chính nó đến các ZR còn lại trong mạng. Thường các thiết bị Sink trong Zigbee sẽ được đặt thành ZC trong mạng để phục vụ cho việc duy trì Route từ nó đến các thiết bị khác trong mạng. ZC là thiết bị trung tâm trong mạng, nó có trách nhiệm trong việc quản lý và duy trì các nút trong mạng. Vì vậy nó luôn phải cập nhật thông tin của từng trạng thái thiết bị. Để làm điều đó nó luôn phải cập nhật các đường định tuyến theo thời gian. Trong Zigbee, thiết bị Sink sẽ cần gửi một bản tin có tên là MTORRs, bản tin này sẽ được gửi theo dạng Broadcast đảm bảo rằng các

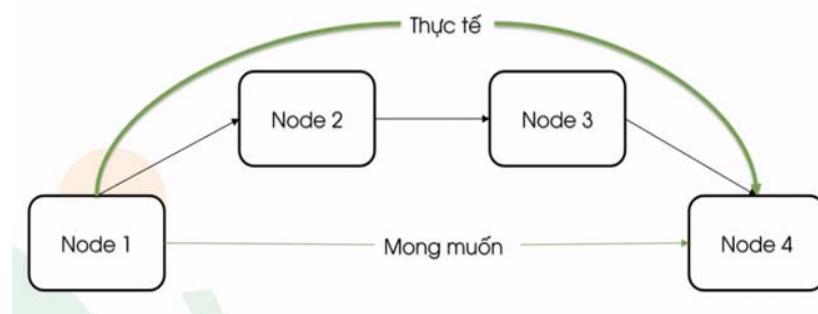
Route đều nhận được. Khi các Route nhận được bản tin MTORRs nó sẽ lần lượt phản hồi Unicast về cho thiết bị Sink một bản tin có tên là Route Record. Bản tin Route Record này chứa đường định tuyến đi từ thiết bị cho đến Sink và Sink có thể sử dụng luồng thông tin này để cập nhật vào bảng định tuyến của mình. Quá trình Many-To-One chỉ diễn ra từ thiết bị Sink đến thiết bị Route. Các thiết bị ZED không tham gia vào quá trình này bởi vì các Route sẽ chịu trách nhiệm quản lý các thiết bị con của nó.



Hình 2.20: Phương thức truyền tin Many-To-One

1.5.8. Zigbee Routing

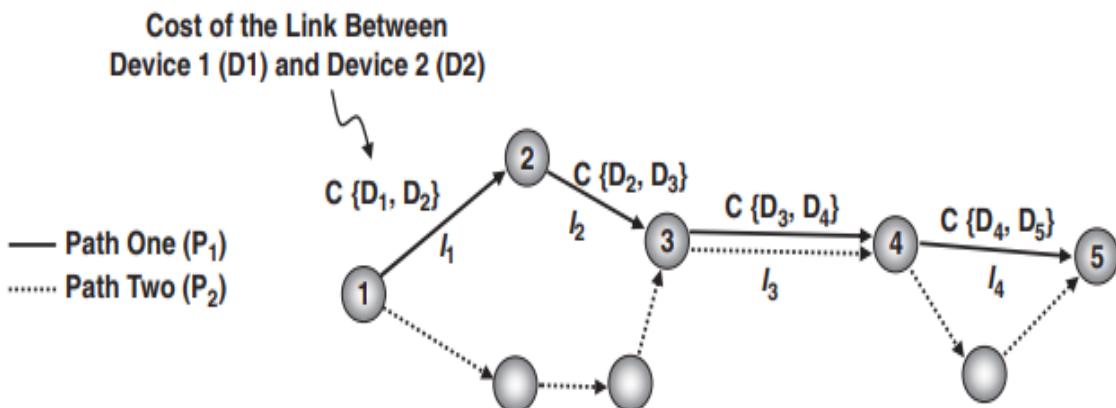
Hoạt động cơ bản trong một mạng là truyền dữ liệu từ nút mạng này sang nút mạng khác. Dữ liệu sẽ được khởi tạo từ nút nguồn và được truyền đến một nút khác có thể phân tích và sử dụng dữ liệu. Trường hợp ưu tiên nhất là dữ liệu được truyền trực tiếp từ thiết bị nguồn đến thiết bị đích. Tuy nhiên trong trường hợp hai thiết bị cách quá xa nhau hoặc tồn tại trong một môi trường khắc nghiệt nhiễu sóng thì quá trình truyền dữ liệu trực tiếp không diễn ra được mà khi đó thiết bị nguồn sẽ gửi dữ liệu đến các nút mạng nằm trong dải radio của mình và bản tin cứ thế được truyền đi đến thiết bị đích. Đây là quá trình Routing được diễn ra.



Hình 2.21: Mô hình truyền dữ liệu

Routing là quá trình diễn tả đường định tuyến qua đó bản tin sẽ được chuyển tiếp đến thiết bị đích. Các thiết bị ZC, ZR sẽ chịu trách nhiệm tìm kiếm và duy trì các đường định tuyến trong mạng.

Trong **Hình 2.22** có thể thấy 2 đường định tuyến để đi từ nút 1 sang nút 5. Length - Độ dài là số lượng thiết bị có trong đường định tuyến. Đường liên kết giữa các nút mạng là Link. Có hai thông số để quyết định lựa chọn đường định tuyến tối ưu: Link Quality, Number of Hops. Mỗi một đường liên kết sẽ được gán giá trị Link Cost – Thể hiện xác suất của bản tin được gửi thành công ở trên mỗi đường link liên kết. Và nếu xác suất gửi đi bản tin thành công càng thấp thì giá trị Link Cost càng cao. Giá trị của Link Cost thường là một số nguyên nằm trong khoảng giá trị từ 0 – 7. Mỗi đường link liên kết ta sẽ có giá trị của Link Cost. Giá trị này sẽ liên tục được cập nhật mỗi khi nút mạng thực hiện quá trình gửi và nhận bản tin. Để quyết định đường định tuyến bản tin nào được chọn thì nút mạng sẽ thực hiện việc tính tổng các Link Cost trên một đường. Và giá trị Link Cost của đường định tuyến nào thấp nhất sẽ được lựa chọn cho việc gửi và nhận bản tin. Để thực hiện được quá trình này, một nút mạng cần phải lưu lại các thông tin mạng hiện hữu xung quanh mình thông qua các bảng.

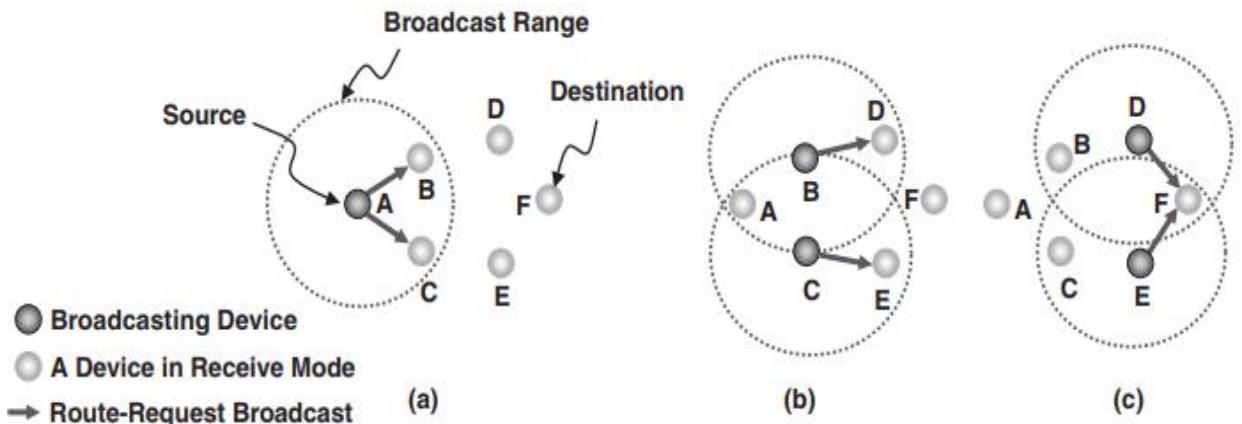


Hình 2.22: Mô hình tính toán giá trị Link Cost

Route Discovery là quá trình được thực hiện ở tầng mạng thông qua các yêu cầu từ tầng ứng dụng. Tầng mạng sẽ sử dụng bản tin Route Request để thực hiện quá trình lấy Route thông qua hình thức Unicast, Multicast và Many-To-One. Nếu trong bản tin Route Request chứa thông tin về địa chỉ của thiết bị nhận thì ở tầng mạng sẽ thực hiện quá trình Route theo hình thức Unicast. Quá trình định tuyến Unicast nghĩa là bản tin Route Request được khởi tạo từ thiết bị nguồn và kết thúc tại thiết bị đích. Còn nếu trong bản tin Route Request chứa thông tin về Group ID của một nhóm thì tầng mạng sẽ thực hiện quá trình Route Request theo hình thức Multicast. Nếu tầng ứng dụng không cung cấp giá trị của địa chỉ đích thì tầng mạng sẽ thực hiện theo hình thức Many-To-One. Unicast, Multicast ở đây không phải là hình thức gửi bản

tin theo dạng này mà Unicast biếu thị cho việc lấy Route từ thiết bị A đến thiết bị B và Multicast đại diện cho việc lấy Route từ A đến một nhóm B. Bản tin Route Request sẽ luôn luôn được gửi dưới dạng bản tin Broadcast vì đây là quá trình các thiết bị đi tìm đường định tuyến lẫn nhau và ở trong chúng chưa hề có bất cứ thông tin nào về tuyến đường sẽ cần phải đi. Sử dụng Broadcast sẽ giúp chúng có nhiều đường định tuyến khác nhau để đưa ra sự lựa chọn tốt nhất.

Ví dụ về quá trình thực hiện Route. Trong ví dụ tất cả các thiết bị đều là Router, và thiết bị A tìm đường định tuyến đến thiết bị F, F không nằm trong dải tầm Radio của A. Đầu tiên thiết bị A sẽ gửi bản tin Route Request thông qua hình thức Broadcast cho toàn mạng. Route Request là bản tin chứa đựng các thông tin Route Request ID, giá trị này sẽ luôn được tăng lên 1 mỗi khi tầng mạng thực hiện xong, địa chỉ đích và giá trị Path Cost. Thiết bị A sẽ set giá trị Path Cost về 0 trước khi thực hiện gửi bản tin Route Request. Do là bản tin Broadcast nên các thiết bị nằm trong vùng phủ sóng radio của A sẽ nhận được, cụ thể là hai thiết bị B, C. Khi thiết bị B, C nhận được bản tin này thì nó sẽ kiểm tra trong bảng Routing table của chúng trước và sau đó chúng sẽ kiểm tra giá trị Path Code từ $A \rightarrow B$, $A \rightarrow C$ và tiếp tục bản tin Route Request này đi. Bản tin này sẽ được xử lý tương tự cho tới F. F sẽ nhận được các bản tin Route Request và từ đó nó có thông tin về Path Cost để chọn lựa đưa ra đường định tuyến tốt nhất từ $A \rightarrow F$ và gửi lại bản tin phản hồi là Route Reply. Khi gửi Route Reply, thì bản tin này sẽ được gửi thông qua hình thức Unicast về cho A.

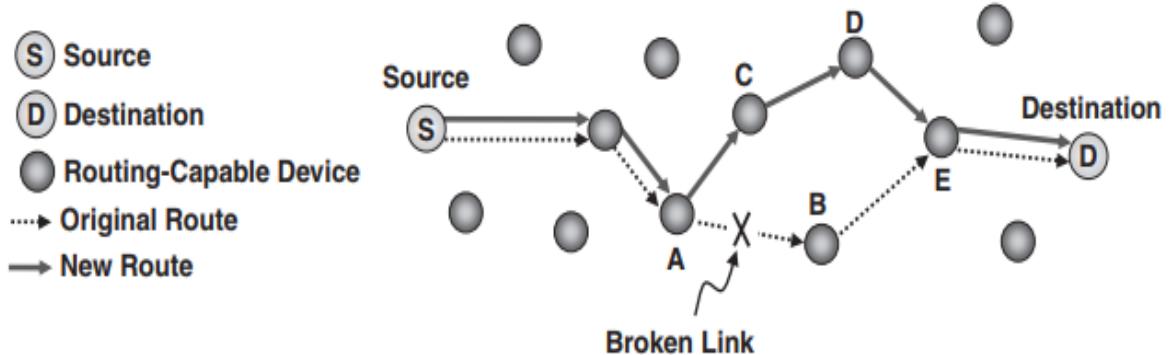


Hình 2.23: Routing Discovery

1.5.9. Route repair

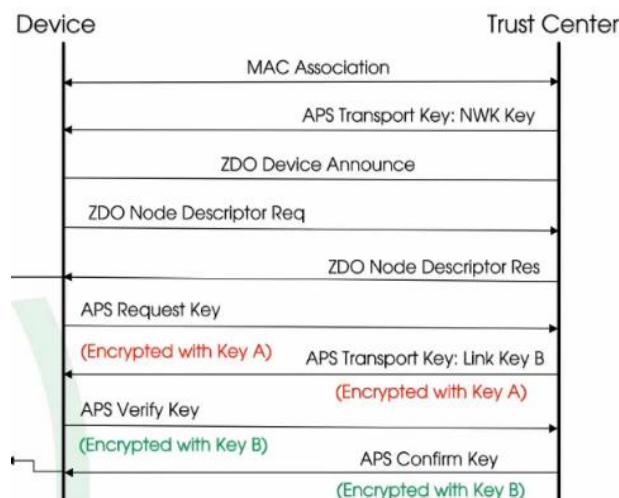
Mạng Zigbee sử dụng cấu trúc mạng Mesh có tích hợp thêm phần xử lý định tuyến thông minh để chắc chắn rằng bản tin đã được gửi đi đến địa chỉ đích của nó. Nếu một đường định tuyến đang gặp sự cố vì một lý do nào đó thì mạng Zigbee sẽ thực hiện cơ chế Discover và thiết lập một đường định tuyến thay thế để phục vụ cho việc truyền nhận bản tin và được gọi là Route repair.

Định tuyến có thể tạm hiểu là đường truyền bản tin ngắn nhất từ thiết bị nguồn đến thiết bị đích. Khi một nút mạng tồn tại trong mạng thì nó sẽ lưu lại những đường định tuyến để có thể gửi đến các thiết bị khác. Tuy nhiên vì bộ nhớ có giới hạn nên một nút mạng sẽ chỉ lưu những đường định tuyến đến những thiết bị nó gửi gần nhất. Khi một đường định tuyến được tạo nó sẽ phục vụ cho việc truyền tải bản tin giữa hai thiết bị với nhau. Thường thì đường định tuyến này sẽ được cố định nếu tất cả các nút mạng ở trong một vị trí cố định. Tuy nhiên có những lý do khiến đường định tuyến này bị ảnh hưởng như route bị gỡ bỏ, hoặc mất nguồn hoặc có sự tác động từ môi trường bên ngoài. Và để quyết định có tạo lại một đường định tuyến mới hoặc là tái tạo lại đường định tuyến cũ thì thuật toán định tuyến không khuyến khích chúng ta cho phép các nút mạng thực hiện việc này ngay khi đường định tuyến gặp sự cố mà thay vào đó chúng ta sẽ cần một bộ đếm để đếm được những bản tin không truyền ra ngoài bị lỗi. Và quá trình Route repair sẽ được thực hiện khi giá trị bộ đếm sẽ đạt giá trị tối hạn nào đó. Trên tầng Application, các nhà sản xuất và phát hành cho phép người lập trình có thể cài đặt thông số này.

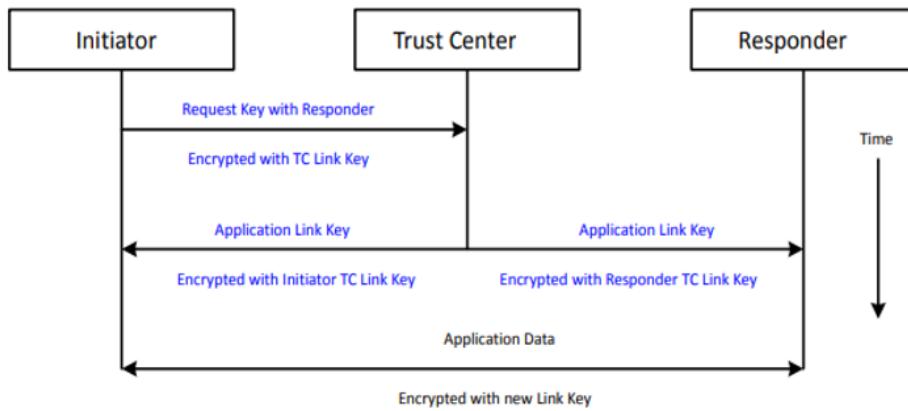


Hình 2.24: Route repair

1.5.10. Tính năng bảo mật trong mạng Zigbee



Hình 2.25: Quá trình cấp phát TCLK và NWK



Hình 2.26: Quá trình cấp phát Application Key

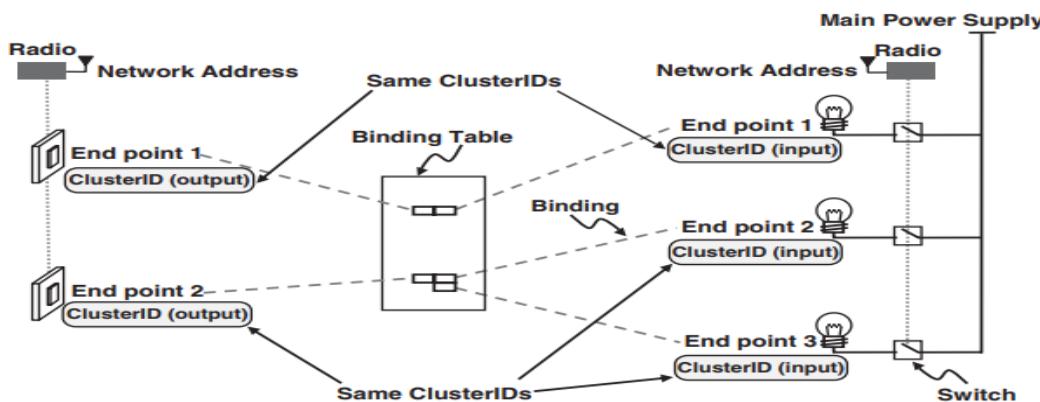
Zigbee 3.0 sử dụng hai loại mô hình mạng đó là Distributed Network và Centralized Network. Trong đó, mô hình mạng chính sử dụng trong Zigbee 3.0 đó là mô hình mạng Centralized Network. Trong mô hình mạng này ZC có một tên gọi khác là Trust Center bởi vì nó là thiết bị quản lý và cung cấp các thông tin bảo mật cho toàn mạng. Trust Center có thể do Router quản lý nhưng để nhất quán trong việc quản lý nên thường các mạng Zigbee sẽ để ZC thực hiện nhiệm vụ này luôn. Trong mạng Zigbee 3.0 sử dụng 3 loại mã bảo mật như sau:

- ❖ **Network Key:** Đây là một loại Key chung được sử dụng để mã hóa giao tiếp giữa các nút trong mạng với các nút ngoài mạng. Các nút trong cùng một mạng sẽ sử dụng cùng một Network Key với nhau. Khi một bản tin được gửi đến, các nút sẽ sử dụng Network Key để giải mã bản tin. Network key được khởi tạo ngẫu nhiên bởi TC và loại key này sẽ được cung cấp cho các nút ngay khi vào mạng thành công. Nó là một loại key bắt buộc cho tất cả các nút ở trong mạng để có thể truyền và nhận bản tin. Nhờ có Network Key, thì mạng Zigbee có thể tránh được các hacker nghe lén mạng và làm nhiễu loạn mạng. Network Key được dùng để mã hóa tầng mạng.
- ❖ **Trust Center Link Key (TCLK):** Đây là loại key phục vụ cho việc giao tiếp giữa TC và các nút trong mạng. TCLK được sử dụng để giải mã các bản tin ở trên tầng ứng dụng. Nó được khởi tạo ngẫu nhiên bởi TC và được gửi cho các nút liên quan. TCLK dùng để mã hóa tầng App. TC được update định kỳ.
- ❖ **Application Link Keys:** Được sử dụng để mã hóa bản tin giữa hai nút mạng với nhau (*trong đó ko có nút mạng nào là TC*). Việc lấy Link Key sẽ được cấp phát bởi TC do một nút gửi yêu cầu Link Key lên TC.

1.5.11. Binding, Group và Scenes trong mạng Zigbee

❖ Binding

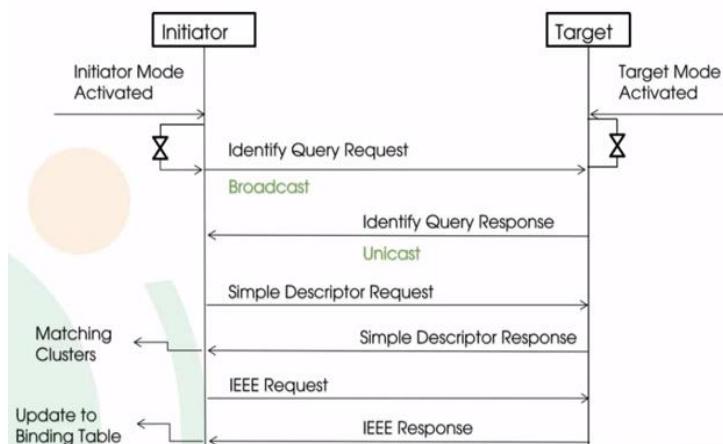
Binding dùng để ghép nối trực tiếp giữa hai nút mạng với nhau, cụ thể là Endpoint của thiết bị này với Endpoint của thiết bị kia với điều kiện là chúng phải sử dụng chung ClusterID. Trong đó một thiết bị sẽ hoạt động dưới dạng Input Cluster và thiết bị khác sẽ hoạt động dưới dạng Output Cluster. Trong Zigbee 3.0 Binding, hai nút mạng được kết nối với nhau phải tồn tại ở một trong hai dạng đó là Initiator và Target, trong đó Initiator có nhiệm vụ khởi tạo ra quá trình liên kết và được cho phép làm thiết bị Control Device. Target có nhiệm vụ nhận và thực hiện điều khiển từ Initiator. Initiator là thiết bị chủ, nó lưu giữ những thông tin mạng của Target và gửi lệnh điều khiển xuống Target khi cần thiết. Để phục vụ việc lưu trữ thông tin mạng của Target thì Binding sử dụng bảng là Binding Table, bảng này chứa đầy đủ thông tin của thiết bị được ghép nối: Endpoint, ClusterID, địa chỉ mạng và địa chỉ IEEE.



Hình 2.27: Binding giữa hai nút mạng

Quá trình Binding diễn ra như sau: Trên Initiator và Target chúng ta cần phải sử dụng một lệnh để kích hoạt sự kiện đó là Initiator Mode Activated và Target Mode Activated. Đối với các lệnh kích hoạt sự kiện chúng ta cần lựa chọn các Endpoint cụ thể để thực hiện quá trình này. Sau sự kiện này, Initiator sẽ chuyển sang chế độ mở làm Initiator bằng việc gửi bản tin Identify Query Request theo dạng Broadcast định kỳ trong khoảng thời gian Timeout. Khoảng thời gian này được quy định bởi người lập trình. Tương tự như vậy, Target sẽ chuyển sang chế độ mở làm Target nhưng thay vì gửi bản tin đi thì nó chỉ mở ra trong khoảng thời gian Timeout để nhận bản tin Identify Query Request và trong khoảng thời gian Target chuyển sang chế độ này nó sẽ xử lý duy nhất một bản tin Identify Query Request và đó là bản tin đầu tiên mà nó nhận được. Khi nhận được bản tin này thì nó sẽ phản hồi cho Initiator bản tin Identify Respond dưới dạng hình thức Unicast. Lúc này các thiết bị đã tìm được đến nhau nên các bản tin cấu hình về sau sẽ hoàn toàn được gửi

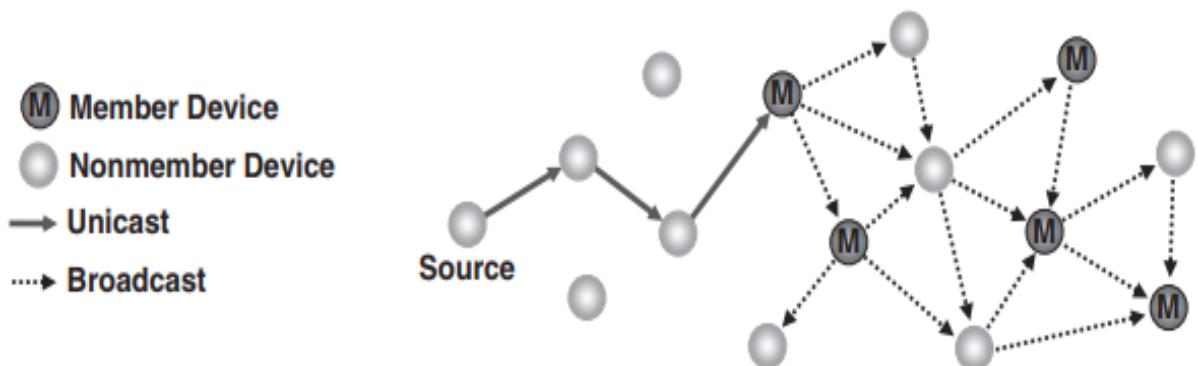
dưới dạng Unicast. Khi Initiator nhận được bản tin Identify Query Respond tức là nó biết rằng có một nút mạng nào đó cũng đang muốn thực hiện quá trình Binding và dưới dạng hình thức Target. Initiator sẽ gửi bản tin Simple Descriptor Target cho Target và Target sẽ gửi lại bản tin phản hồi cho Initiator. Trong bản tin Simple Descriptor có chứa các thông tin về các trường Cluster của thiết bị và nhờ vào các thông tin này Initiator sẽ kiểm tra và ghép nối các Cluster của mình vào Target. Input Cluster sẽ được ghép nối vào Output Cluster và cứ mỗi cặp như vậy Initiator sẽ lưu trữ lại và cập nhật vào bảng Binding Table. Sau đó để hoàn thiện thông tin trong bảng Binding Table, Initiator sẽ tiếp tục gửi bản tin IEEE Request để cập nhật thông tin về địa chỉ IEEE của thiết bị Target.



Hình 2.28: Quá trình Binding giữa hai nút mạng

❖ Group

Group được sử dụng để điều khiển một nhóm nhỏ trong mạng. Mỗi Group sẽ được định danh bằng một giá trị GroupID 16 bits. Khi cần điều khiển chúng ta chỉ cần gửi một lệnh Multicast kèm với định danh GroupID tương ứng.



Hình 2.29: Mô hình Group trong mạng Zigbee

❖ Scenes

Thường được sử dụng kết hợp với Group, nhưng thay vì sử dụng Group một mình, các thiết bị khi nhận lệnh thì sẽ chỉ cùng thực hiện một tác vụ nào đó. Ở

Scene chúng ta hoàn toàn có thể đăng ký các tác vụ thực hiện riêng lẻ cho từng loại thiết bị. Ví dụ trong một môi trường sử dụng đèn RGB với 3 đèn RGB khác nhau, ta mong muốn khi kích hoạt một cảnh mỗi đèn sẽ sáng một màu đỏ, xanh lá cây, xanh da trời. Khi đó chúng ta hoàn toàn có thể ghép 3 thiết bị đó vào một Group và sử dụng Scene để cấu hình cho từng loại thiết bị. Tuy nhiên điểm yếu của Scene là phải thực hiện thao tác trên từng thiết bị một và chúng không được linh hoạt trong quá trình sử dụng.

2. Vi điều khiển ESP32

2.1. Giới thiệu về vi điều khiển ESP32

ESP32 là một vi điều khiển (*MCU*) tích hợp Wi-Fi và Bluetooth được thiết kế cho các ứng dụng Internet vạn vật (*IoT*). Việc kết hợp Wi-Fi, Bluetooth và khả năng xử lý mạnh mẽ khiến ESP32 trở thành lựa chọn lý tưởng cho nhiều loại dự án IoT, từ các thiết bị gia dụng thông minh đơn giản đến các hệ thống giám sát phức tạp.

❖ Đặc điểm nổi bật của ESP32

- ◆ **Hỗ trợ Wi-Fi và Bluetooth:** Cho phép kết nối với Internet, các thiết bị khác và các dịch vụ đám mây.
- ◆ **Hiệu năng mạnh mẽ:** CPU Xtensa LX106 lõi kép 32 bit với tốc độ xung nhịp lên đến 240 MHz, bộ nhớ flash lên đến 4 MB và bộ nhớ SRAM lên đến 8 MB.
- ◆ **Khả năng lập trình dễ dàng:** Hỗ trợ nhiều môi trường lập trình phổ biến như Arduino IDE, ESP-IDF, MicroPython...
- ◆ **Giá cả hợp lý:** Chi phí thấp hơn so với các vi điều khiển khác có cùng tính năng.

→ **Với sự kết hợp mạnh mẽ giữa hiệu năng, tính linh hoạt và giá cả hợp lý, ESP32 là lựa chọn lý tưởng cho các nhà phát triển IoT ở mọi cấp độ.**

2.2. Kiến trúc ESP32

2.2.1. CPU

CPU Xtensa LX106 lõi kép 32 bit là trái tim của ESP32, mang đến hiệu năng mạnh mẽ, khả năng lập trình linh hoạt và các tính năng bảo mật tiên tiến.

Với tốc độ xung nhịp lên đến 240 MHz, bộ nhớ Cache L1 32 KB cho mỗi lõi và FPU, CPU này xử lý nhanh chóng các tác vụ phức tạp. Hỗ trợ bộ lệnh RISC 32 bit tiêu chuẩn và nhiều ngôn ngữ lập trình phổ biến, ESP32 dễ dàng lập trình và phù hợp với nhiều dự án IoT.

Có nhiều chế độ hoạt động, CPU ESP32 tối ưu hóa hiệu năng và mức tiêu thụ điện năng. Các tính năng bảo mật như mã hóa Flash khởi động an toàn và thuật toán mã hóa phần cứng bảo vệ thiết bị và dữ liệu khỏi truy cập trái phép.

Nhờ CPU mạnh mẽ, ESP32 đáp ứng nhu cầu của nhiều ứng dụng IoT đa dạng, từ đơn giản đến phức tạp.

2.2.2. Bộ nhớ

ESP32 sở hữu hệ thống bộ nhớ mạnh mẽ bao gồm:

Bộ nhớ Flash

- ❖ Lưu trữ chương trình và dữ liệu.
- ❖ Dung lượng lên đến 4 MB, đáp ứng nhu cầu của nhiều ứng dụng IoT.
- ❖ Hỗ trợ mã hóa Flash khởi động an toàn để bảo vệ thiết bị khỏi truy cập trái phép.

Bộ nhớ NVS Flash

- ❖ NVS Flash sử dụng một phần bộ nhớ Flash để lưu các cặp **Key-Value**. Phân vùng NVS trong Flash được chia thành nhiều **partitions (pages)**. NVS sẽ lưu các cặp giá trị Key-Value một cách tuần tự:
 - ◆ Key là ASCII strings với độ dài lớn nhất là 15 characters. Key là giá trị duy nhất. Khi các cặp giá trị Key-Value mới được thêm vào thì sẽ được thêm vào cuối của partition (*page*) muốn đưa vào. Trong trường hợp, một giá trị của Key muốn cập nhật, một cặp Key-Value mới sẽ được thêm vào cuối của page bằng hoạt động write, cặp Key-Value cũ được đánh dấu để xóa. Khi muốn lấy kiểu dữ liệu, giá trị của key đó thì thực hiện read.
 - ◆ Value có một số kiểu giá trị sau

- integer types: `uint8_t`, `int8_t`, `uint16_t`, `int16_t`, `uint32_t`, `int32_t`, `uint64_t`, `int64_t`
- zero-terminated string
- variable length binary data (blob)

Hình 2.30: Các kiểu value trong cặp Key-Value lưu trữ trong NVS Flash

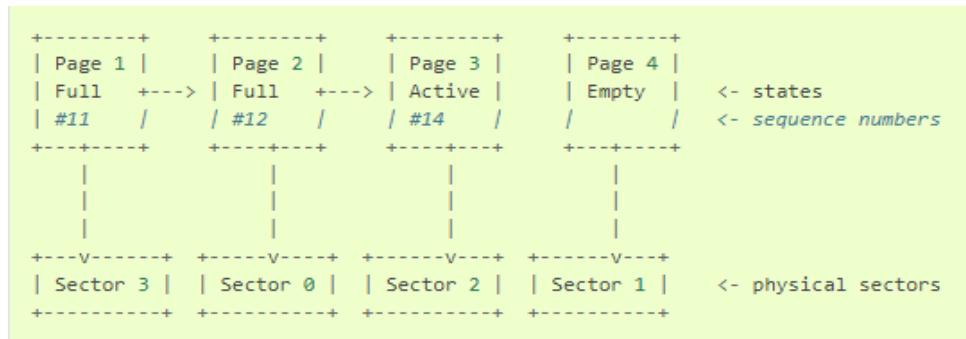
- ◆ Để giảm thiểu việc xung đột tiềm ẩn về Key giữa các thành phần khác nhau, NVS sẽ gắn cho mỗi cặp giá trị Key-Value với một **namespace**. Tên của namespace thuộc kiểu ASCII string, gồm tối đa 15 characters. Ngoài ra, trên một partition có tối đa 254 namespaces khác nhau. Bằng cách này, một trình xử lý được liên kết với namespace và keyname sẽ không xung đột với các tên giống nhau trong các namespaces khác. Mặt khác, các namespaces với tên giống nhau trong các partitions khác nhau của NVS sẽ được coi như các namespaces riêng biệt.

```
+-----+
| NS=0 Type=uint8_t Key="wifi" Value=1 |   Entry describing namespace "wifi"
+-----+
| NS=1 Type=uint32_t Key="channel" Value=6 | Key "channel" in namespace "wifi"
+-----+
| NS=0 Type=uint8_t Key="pwm" Value=2 |   Entry describing namespace "pwm"
+-----+
| NS=2 Type=uint16_t Key="channel" Value=20 | Key "channel" in namespace "pwm"
+-----+
```

Hình 2.31: Namesapce-Key-Value lưu trữ trong NVS Flash

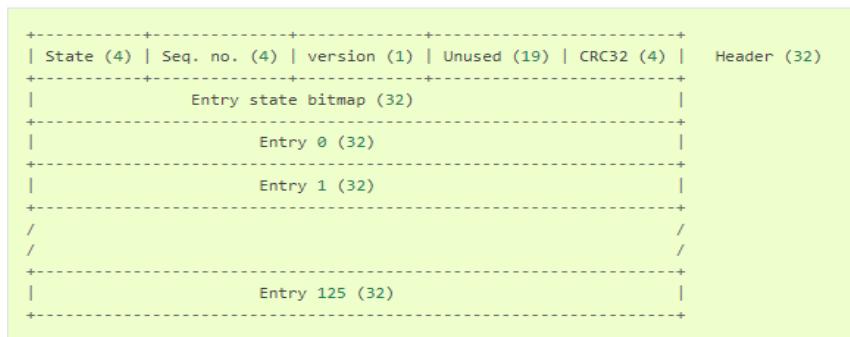
→ Việc truy cập vào một cặp giá trị Key-Value sẽ phải thông qua 3 yếu tố: partition (*page*), namespace, kiểu dữ liệu.

- ❖ NVS sử dụng hai thực thể **pages** và **entries** trong hoạt động của nó:
 - ◆ **pages** là một cấu trúc logic lưu trữ một phần của overall log. Mỗi page sẽ tương ứng với một khu vực vật lý của bộ nhớ flash. Mỗi page sẽ có một số thứ tự liên kết với nó. Khi một page mới được tạo ra, số thứ tự của nó sẽ tăng lên 1 đơn vị so với số thứ tự của tổng page hiện tại. Mỗi page sẽ bao gồm một trong các trạng thái sau: **Empty/uninitialized** – bộ nhớ flash với page trống, page không được sử dụng để lưu bất kỳ dữ liệu nào và không có số thứ tự; **Active** – bộ nhớ flash được khởi tạo, tiêu đề trang được ghi vào flash, trang có số thứ tự hợp lệ, page có thể có một số mục trống và dữ liệu có thể viết vào đó; **Full** – bộ nhớ flash đã chứa đầy các cặp key-values, không thể viết thêm cặp key-values vào page được; **Erasing** – những cặp giá trị key-value không được xóa sẽ di chuyển sang một page khác để xóa page hiện tại; **Corrupt**. Ánh xạ từ các khu vực của flash đến các logical pages không có thứ tự cụ thể.



Hình 2.32: Thực thể page trong NVS Flash

Mỗi pages gồm 3 thành phần: header, entry state bitmap and entrys. Đối với kiểu số nguyên, 1 entry giữ 1 cặp Key-Value. Đối với kiểu strings, blobs, 1 entry giữ 1 phần của cặp Key-Value. Đối với kiểu strings, tất cả các entrys được lưu trữ trên cùng một page. Đối với kiểu blobs, được lưu trữ trên nhiều pages bằng cách phân chung thành các chunks nhỏ hơn.



Hình 2.33: Thành phần của thực thể page trong NVS Flash

- ◆ Mỗi entry có 1 trong 3 trạng thái: **Empty** – Không có gì được viết vào một mục cụ thể, nó trong trạng thái chưa khởi tạo, **Written** – Một cặp key-value đã được viết vào entry, **Erased** – 1 cặp key-value được loại bỏ.

Bộ nhớ SRAM

- ❖ Lưu trữ dữ liệu tạm thời.
- ❖ Dung lượng lên đến 8 MB, cho phép truy cập dữ liệu tốc độ cao.
- ❖ Một số module ESP32 còn hỗ trợ bộ nhớ PSRAM để mở rộng dung lượng lưu trữ.

Quản lý bộ nhớ

- ❖ ESP32 có hệ thống quản lý bộ nhớ hiệu quả, giúp tối ưu hóa việc sử dụng bộ nhớ và tránh tình trạng tràn bộ nhớ.
- ❖ Hỗ trợ phân bổ bộ nhớ động, giúp lập trình viên dễ dàng quản lý bộ nhớ cho các ứng dụng của mình.

2.2.3. Khối Wi-Fi, Bluetooth

ESP32 được tích hợp Wi-Fi 802.11 b/g/n/e/i và Bluetooth 4.2 BLE, cho phép kết nối với Internet, các thiết bị khác và các dịch vụ đám mây.

❖ **Khối Wi-Fi**

- ◆ Hỗ trợ chuẩn Wi-Fi 802.11 b/g/n/e/i.
- ◆ Tốc độ truyền dữ liệu lên đến 150 Mbps.
- ◆ Hỗ trợ các chế độ AP, STA và AP+STA.
- ◆ Hỗ trợ bảo mật WPA2-Enterprise.

❖ **Khối Bluetooth**

- ◆ Hỗ trợ Bluetooth 4.2 BLE.
- ◆ Tốc độ truyền dữ liệu lên đến 1 Mbps.
- ◆ Hỗ trợ các chế độ Classic Bluetooth, BLE và BLE Long Range.
- ◆ Hỗ trợ bảo mật AES-128.

❖ **Khả năng kết nối mạnh mẽ của ESP32 hữu dụng cho các ứng dụng**

- ◆ Thu thập dữ liệu từ các cảm biến.
- ◆ Điều khiển các thiết bị điện tử.
- ◆ Giao tiếp với các thiết bị khác.
- ◆ Kết nối với internet và các dịch vụ đám mây.

2.2.4. Bộ quản lý năng lượng

ESP32 được trang bị bộ quản lý năng lượng tiên tiến giúp tối ưu hóa mức tiêu thụ điện năng cho các ứng dụng IoT với các tính năng như sau:

- ❖ Cho phép ESP32 hoạt động ở các chế độ năng lượng khác nhau:
 - ◆ Chế độ hoạt động: CPU hoạt động, tiêu thụ năng lượng cao.

- ◆ Chế độ ngủ: CPU tắt, tiêu thụ năng lượng thấp.
- ◆ Chế độ Deep Sleep: CPU và hầu hết các ngoại vi đều tắt, tiêu thụ năng lượng rất thấp.
- ❖ Hỗ trợ quản lý nguồn điện cho các thành phần riêng lẻ của ESP32 như CPU, Wi-Fi, Bluetooth,...
- ❖ Cung cấp các API để lập trình viên kiểm soát mức tiêu thụ điện năng của ứng dụng.

2.2.5. Tính năng bảo mật tích hợp

ESP32 được tích hợp nhiều tính năng bảo mật tiên tiến giúp bảo vệ thiết bị và dữ liệu khỏi các truy cập trái phép bằng cách sử dụng các tính năng sau:

- ❖ Mã hóa flash khởi động an toàn: Bảo vệ ESP32 khỏi việc khởi động bằng firmware trái phép.
- ❖ Bộ xử lý mã hóa phần cứng: Hỗ trợ các thuật toán mã hóa phổ biến như AES, RSA,... để bảo mật dữ liệu.
- ❖ Khóa bảo mật: Giúp bảo vệ các cài đặt và dữ liệu nhạy cảm.
- ❖ Cập nhật firmware an toàn: Cho phép cập nhật firmware mới một cách an toàn và bảo mật.

2.3. Công cụ lập trình ESP32

ESP32 là một trong những vi điều khiển được sử dụng phổ biến trong việc phát triển các ứng dụng IoT (*Internet of Things*). Khả năng tích hợp Wi-Fi và Bluetooth, cùng với việc tích hợp nhiều tính năng khác như việc sử dụng năng lượng hiệu quả, làm cho nó trở thành một lựa chọn lý tưởng cho các dự án IoT. Để lập trình trên ESP32, có nhiều phương pháp khác nhau như Arduino IDE, Visual Studio Code hoặc bằng phần mềm chuyên dụng của nhà sản xuất Espressif Tool.

Việc lập trình trên ESP32 thường bắt đầu với việc cài đặt môi trường phát triển phần mềm (*IDE*) phù hợp. Arduino IDE là một trong những lựa chọn phổ biến nhất, đặc biệt là đối với những người mới bắt đầu với vi điều khiển và lập trình IoT. Để bắt đầu, người dùng cần cài đặt gói hỗ trợ ESP32 trong Arduino IDE, sau đó có thể bắt đầu viết code và tải nó lên board ESP32 thông qua cổng USB.

Trong quá trình lập trình, việc hiểu về cấu trúc của ESP32 và cách nó hoạt động là rất quan trọng. ESP32 sử dụng một số chân GPIO (*General Purpose Input/Output*) để kết nối với các cảm biến, các thiết bị ngoại vi và các mạch điều khiển khác. Nắm vững cách sử dụng GPIO, cùng với việc hiểu rõ về các giao thức như Wi-Fi và Bluetooth, sẽ giúp lập trình viên tạo ra các ứng dụng IoT phức tạp.

Một trong những điểm mạnh của ESP32 là khả năng kết nối với Internet thông qua Wi-Fi. Điều này mở ra một loạt các cơ hội cho việc phát triển các ứng dụng IoT

như cảm biến thông minh, hệ thống giám sát, hoặc các dịch vụ kết nối mạng xã hội. Bằng cách sử dụng thư viện như ESP32 WiFi và các giao thức như MQTT, người lập trình có thể dễ dàng tạo ra các hệ thống mạng IoT phức tạp.

Ngoài việc sử dụng Arduino IDE, có những phương pháp lập trình khác cho ESP32 như sử dụng Visual Studio Code hoặc Espressif Tool. Visual Studio Code là một trong những Code Editor mạnh mẽ và phổ biến nhất dành cho lập trình viên nhờ hỗ trợ nhiều ngôn ngữ lập trình phổ biến, tích hợp đầy đủ các tính năng và khả năng mở rộng. Với Visual Studio Code, người lập trình sẽ hiểu được sâu hơn về luồng hoạt động của thiết bị do sử dụng trực tiếp các API của nhà sản xuất thay vì có thể sử dụng thư viện tích hợp sẵn trong Arduino, vì vậy sẽ giúp người lập trình

Bên cạnh đó, sử dụng phần mềm chuyên dụng của nhà sản xuất Espressif Tool cũng là một lựa chọn cho việc lập trình ESP32. Lập trình trên Espressif Tool cũng sử dụng các API của nhà sản xuất giúp người lập trình tận dụng các tính năng mạnh mẽ của ESP32.

Trong tất cả các phương pháp lập trình trên ESP32, việc thử nghiệm và debug là một phần quan trọng của quá trình phát triển. ESP32 hỗ trợ nhiều công cụ debug khác nhau như Serial Monitor trong Arduino IDE hoặc công cụ định tuyến Wi-Fi. Việc sử dụng các công cụ này giúp lập trình viên tìm ra và sửa các lỗi một cách nhanh chóng và hiệu quả.

Tóm lại, việc lập trình trên ESP32 là một quá trình thú vị và đầy tiềm năng. Với khả năng tích hợp Wi-Fi và Bluetooth, cùng với việc hỗ trợ nhiều ngôn ngữ lập trình như Arduino IDE, Visual Studio Code và Espressif Tool, ESP32 là một nền tảng mạnh mẽ cho việc phát triển các ứng dụng IoT đa dạng và phức tạp.

→ Trong đồ án này, tôi sử dụng Visual Studio Code để lập trình ESP32_Wroom_32.

Một số câu lệnh Python sử dụng trong Visual Studio Code để lập trình ESP32_Wroom_32

- ❖ Lệnh thoát khỏi thoát ra thư mục ngoài: **cd ..**
- ❖ Lệnh vào thư mục code: **cd name_of_folder**
- ❖ Lệnh xem các file trong thư mục code: **ls**
- ❖ Lệnh configure: **idf.py menuconfig**
- ❖ Lệnh xóa tất cả: **idf.py fullclean**
- ❖ Lệnh build code: **idf.py build**
- ❖ Lệnh nạp code: **idf.py -p PORT flash.** Trong đó, PORT – Cổng COM kết nối giữa ESP32 với máy tính.
- ❖ Lệnh bật hiển thị trên Monitor: **idf.py monitor -p PORT.** Muốn thoát chế độ xem Monitor, sử dụng **Ctrl + J**.
- ❖ Lệnh nạp code và hiển thị trên Monitor: **idf.py -p PORT flash monitor.**

2.4. Kết nối Wi-Fi, Bluetooth

ESP32 là một trong những vi điều khiển có khả năng kết nối Wi-Fi và Bluetooth một cách mạnh mẽ và linh hoạt. Với khả năng này, nó đã trở thành một lựa chọn ưa thích cho việc phát triển các ứng dụng IoT và các dự án cần sự kết nối không dây.

Khả năng kết nối Wi-Fi của ESP32 được coi là một trong những điểm nổi bật nhất. ESP32 hỗ trợ nhiều chế độ kết nối Wi-Fi như chế độ Station Mode (*Chế độ Điểm truy cập*) và Access Point Mode (*Chế độ Điểm phát mạng cục bộ*). Trong chế độ Station Mode, ESP32 có thể kết nối với mạng WiFi hiện có, cho phép truy cập vào Internet và trao đổi dữ liệu với các thiết bị khác trong mạng. Điều này mở ra cánh cửa cho việc phát triển các ứng dụng IoT phong phú, từ hệ thống giám sát đến các ứng dụng điều khiển từ xa.

Ngoài ra, ESP32 cũng hỗ trợ chế độ Access Point Mode, cho phép nó hoạt động như một điểm truy cập Wi-Fi riêng. Điều này rất hữu ích khi cần tạo ra một mạng WiFi riêng để kết nối các thiết bị IoT với nhau mà không cần phải phụ thuộc vào mạng WiFi ngoại vi. Điều này đặc biệt hữu ích trong các ứng dụng như hệ thống giám sát cảm biến, nơi mà việc triển khai một mạng riêng có thể đảm bảo tính ổn định và bảo mật.

Bên cạnh khả năng kết nối Wi-Fi, ESP32 cũng có khả năng kết nối Bluetooth. Bluetooth là một giao thức không dây tiêu chuẩn được sử dụng rộng rãi cho việc kết nối các thiết bị di động, tai nghe không dây, và nhiều ứng dụng khác. ESP32 hỗ trợ cả Bluetooth Classic và Bluetooth Low Energy (*BLE*), mở ra nhiều cơ hội cho việc phát triển các ứng dụng IoT sử dụng Bluetooth để kết nối với các thiết bị di động hoặc thiết bị khác có hỗ trợ Bluetooth.

→ Trong đồ án này, tôi sử dụng tính năng kết nối Wi-Fi của ESP32

2.5. Ứng dụng của ESP32

Với hiệu năng của vi điều khiển và khả năng kết nối mạng WiFi và Bluetooth của ESP32, điều đó khiến cho ứng dụng của nó trở nên phổ biến và rộng rãi như:

❖ Nhà thông minh

- ◆ Điều khiển thiết bị: Bật/tắt đèn, quạt, rèm cửa, v.v. bằng điện thoại thông minh hoặc giọng nói.
- ◆ Theo dõi môi trường: Giám sát nhiệt độ, độ ẩm, chất lượng không khí,... trong nhà.
- ◆ Hệ thống an ninh: Cảnh báo đột nhập, cháy nổ, rò rỉ khí gas,...

❖ Nông nghiệp thông minh

- ◆ Tưới nước tự động: Tưới nước cho cây trồng dựa trên độ ẩm đất và điều kiện thời tiết.

- ◆ Bón phân tự động: Bón phân cho cây trồng theo nhu cầu dinh dưỡng của từng giai đoạn phát triển.
- ◆ Giám sát môi trường: Theo dõi nhiệt độ, độ ẩm, ánh sáng,... trong khu vực trồng trọt.
- ❖ **Công nghiệp thông minh**
 - ◆ Giám sát và điều khiển quy trình sản xuất: Theo dõi các thông số kỹ thuật của máy móc, thiết bị và điều khiển quy trình sản xuất từ xa.
 - ◆ Bảo trì dự đoán: Dự đoán nguy cơ hỏng hóc của máy móc và thiết bị để thực hiện bảo trì trước khi xảy ra sự cố.
 - ◆ Quản lý năng lượng: Theo dõi và tối ưu hóa việc sử dụng năng lượng trong nhà máy.

3. Web

3.1. Sơ lược mô hình Client – Server

3.1.1. Khái niệm Client, Server

Web server

Web server (*Server*) hay còn gọi là máy chủ web là một thành phần giúp lưu trữ và xử lý các yêu cầu của người dùng Internet. Web server có khả năng tiếp nhận request từ các trình duyệt web và gửi phản hồi đến client thông qua giao thức HTTP - Hypertext Transfer Protocol hoặc các giao thức khác. Web server thường được cấu hình để xử lý các ngôn ngữ lập trình web như HTML, CSS, JavaScript và các ngôn ngữ lập trình phía máy chủ như PHP, Python và Ruby. Các web server phổ biến nhất là Apache, Nginx, Microsoft IIS và Google Web Server. Web server có thể là phần cứng máy chủ web, cũng có thể là một dạng phần mềm hoặc là sự kết hợp của cả hai tùy điều kiện.

- ❖ **Web server phần cứng** có thể là máy tính hoặc vi điều khiển có khả năng kết nối Internet như vi điều khiển họ ESP ... đảm nhiệm chức năng lưu trữ bộ phận không thể thiếu để cấu thành web như: file ảnh, file javascript, HTML,.. Máy chủ phải kết nối với mạng internet và hỗ trợ truy cập thông qua Domain.
- ❖ **Web server phần mềm** đảm nhiệm việc theo dõi người dùng web lúc họ thực hiện truy cập đến file host từ tối thiểu một HTTP server. Mỗi HTTP server lại phù hợp với phần mềm có khả năng đọc URLs. Tất cả các trình duyệt đều cần đến file host vận hành trên web server. Nói chung, trình duyệt đó sẽ gửi yêu cầu file bằng HTTP. Nếu một yêu cầu được gửi đến web server, HTTP cũng sẽ gửi lại một yêu cầu phản hồi.

→ Trong đồ án này, tôi sử dụng ESP32 làm WebServer phần cứng.

Web client

Web client (Client) đơn giản chỉ là một trình duyệt web trên laptop và smart phone của người dùng. Khi Client gửi request đến Server từ một trang web, hoặc tài nguyên cụ thể nào đó, thì Server có nhiệm vụ response bằng cách gửi lại dữ liệu được yêu cầu tương ứng cho Client.

→ Trong một số trường hợp, vai trò của Server và Client có thể bị đảo ngược. Ví dụ, trong mạng ngang hàng (*peer-to-peer network*), mỗi thiết bị đóng vai trò vừa là Server vừa là Client, thực hiện nhiệm vụ chia sẻ tài nguyên và dữ liệu với các thiết bị khác trong hệ thống mạng.

Việc hiểu rõ về vai trò của Server và Client là rất quan trọng đối với các nhà phát triển web, vì nó giúp họ thiết kế và triển khai các web app có khả năng cung cấp nội dung web đến người dùng một cách nhanh chóng và chính xác.

Sự khác nhau giữa Client và Server

Sự khác biệt chính giữa Server và Client là về vai trò của chúng trong hệ thống máy tính nối mạng. Trong đó, Server là một máy tính hoặc ứng dụng phần mềm cung cấp dịch vụ, tài nguyên cho các máy tính hay thiết bị khác trên mạng, còn Client là máy tính hoặc thiết bị yêu cầu và nhận dịch vụ hoặc tài nguyên từ Server. Cụ thể những điểm khác biệt giữa Server và Client sẽ được thể hiện qua bảng sau:

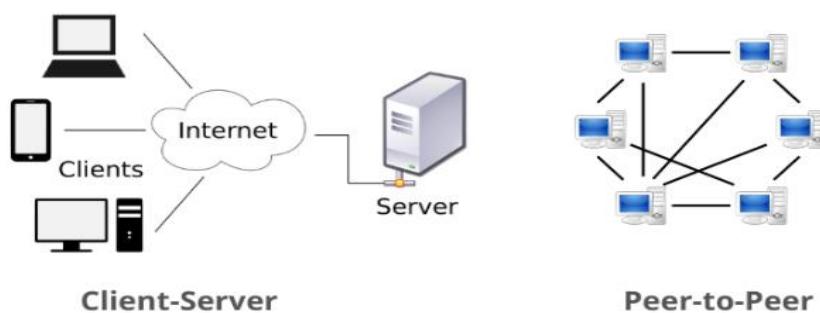
STT	Nội dung so sánh	Server	Client
1	Chức năng	Thiết kế để cung cấp một dịch vụ hoặc tài nguyên cụ thể, chẳng hạn như lưu trữ tệp, quản lý cơ sở dữ liệu hoặc phân phối nội dung web.	Thiết kế để tận dụng hiệu quả các dịch vụ, cũng như nguồn tài nguyên mà Server cung cấp.
2	Hardware	Thường là máy tính hiệu suất cao, với dung lượng bộ nhớ và lưu trữ lớn, cùng các thành phần phần cứng chuyên dụng như giao diện mạng và bộ điều khiển RAID.	Bao gồm từ máy tính để bàn, máy tính xách tay đến các thiết bị di động như điện thoại thông minh, máy tính bảng.
3	Software	Chạy các ứng dụng phần mềm chuyên dụng, cung cấp các dịch vụ cụ thể, chẳng hạn như web server, database servers và email servers.	Chạy nhiều ứng dụng phần mềm cho các mục đích khác nhau, chẳng hạn như trình duyệt web, ứng dụng email và phần mềm năng suất văn phòng.

STT	Nội dung so sánh	Server	Client
4	Khả năng kết nối	Được kết nối với mạng mọi lúc và hoạt động với vai trò xử lý đồng thời nhiều kết nối từ Client.	Thường chỉ được kết nối với mạng khi người dùng cần truy cập tài nguyên, dịch vụ từ Server.

Bảng 2.1: So sánh sự khác nhau giữa Client và Server

3.1.2. Mô hình Client - Server

Mô hình Server - Client là một kiến trúc phổ biến thường được sử dụng trong mạng máy tính, nhằm tạo điều kiện thuận lợi cho việc giao tiếp và trao đổi dữ liệu giữa các máy tính, hoặc thiết bị trên không gian mạng.



Hình 2.34: Mô hình Client – Server

Trong mô hình này, Server sẽ cung cấp dịch vụ, tài nguyên cho Client, thường là các thiết bị người dùng như máy tính, điện thoại thông minh hay máy tính bảng. Client thì yêu cầu dịch vụ, tài nguyên từ Server và nó phải phản hồi bằng cách gửi dữ liệu được yêu cầu hoặc thực hiện hành động được yêu cầu.

Mô hình Server - Client được sử dụng trong nhiều ứng dụng, bao gồm email, duyệt web, chia sẻ tệp và nhắn tin nhanh. Trong mỗi trường hợp, Server sẽ cung cấp một dịch vụ, tài nguyên cụ thể cho các Client và chúng truy cập dịch vụ, tài nguyên đó bằng cách kết nối với Server.

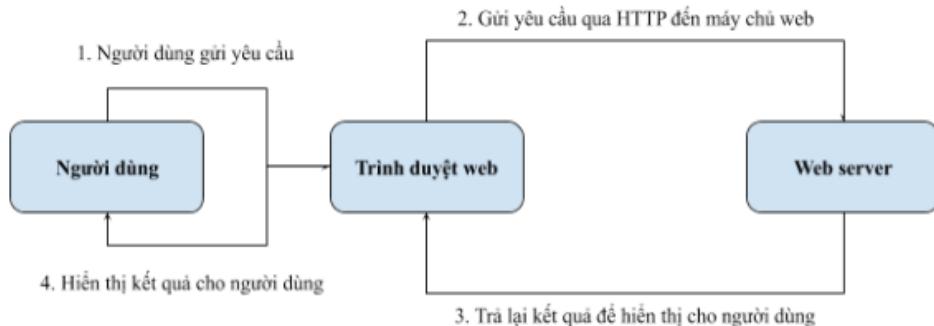
Mô hình này cũng được sử dụng trong các ứng dụng phần mềm Client-Server, trong đó Server cung cấp tính logic và sức mạnh xử lý cho ứng dụng, trong khi Client cung cấp giao diện người dùng và sự tương tác với người dùng. Kiểu kiến trúc này thường được sử dụng trong các web app, trong đó Server có trách nhiệm lưu trữ và quản lý dữ liệu cũng như phản hồi lại các yêu cầu từ Client, thường là trình duyệt web.

Một ưu điểm của mô hình Server - Client là nó cho phép sử dụng hiệu quả tài nguyên mạng, vì nhiều Client có thể kết nối với một Server và thực hiện việc chia sẻ tài nguyên của nó. Đồng thời, nó còn cho phép quản lý và kiểm soát tập

trung, bởi Server có thể giám sát và quản lý các tài nguyên, dịch vụ mà nó cung cấp cho Client.

Có thể nói, mô hình Server - Client là một kiến trúc linh hoạt và hiệu quả trong quá trình xây dựng các ứng dụng, cũng như dịch vụ có kết nối mạng. Nó cũng được sử dụng rộng rãi trong việc phát triển các ứng dụng web và phần mềm hiện đại.

3.1.3. Quá trình giao tiếp giữa Client – Server



Hình 2.35: Mô hình giao tiếp giữa Client – Server

Bước 1: Người dùng gửi yêu cầu

Người dùng Internet sẽ truy cập một Website bất kỳ thông qua một Trình duyệt Web được cài trên máy tính hoặc thiết bị di động. **Trong đồ án này, để truy cập vào giao diện Web, tôi sử dụng địa chỉ IP mà ESP32 được cấp phát sau khi kết nối đến mạng Wi-Fi cụ thể.**

Bước 2: Trình duyệt web gửi yêu cầu tới Web server để xử lý

Lúc này, Trình duyệt Web mà bạn đang sử dụng (*Chrome, Cốc Cốc, ...*) sẽ nhận yêu cầu đó và chuyển đổi từ địa chỉ tên miền sang địa chỉ IP kèm theo tên miền đó. Việc truy xuất thông tin IP này sẽ thông qua các máy chủ DNS (**Trong đồ án này, trình duyệt không thực hiện bước chuyển đổi từ tên miền sang địa chỉ IP.**)

Sau đó, trình duyệt sẽ thông qua giao thức HTTP gửi yêu cầu đến Web server báo là có một người dùng đang cần truy xuất thông tin tại địa chỉ này. Và nó yêu cầu máy chủ trả về kết quả cho người dùng.

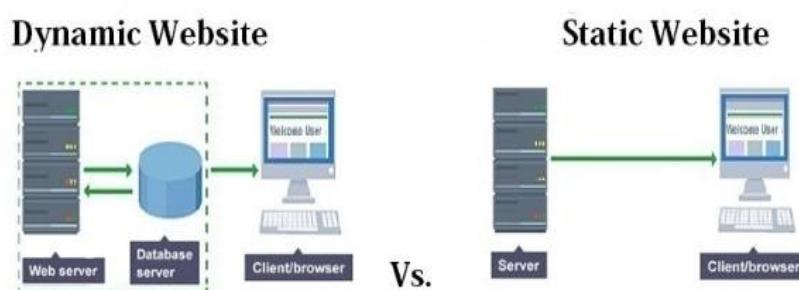
Bước 3: Máy chủ web kiểm tra, trả về kết quả và trình duyệt hiển thị kết quả cho người dùng

Khi nhận được yêu cầu từ trình duyệt, máy chủ web sẽ kiểm tra lại trong hệ thống xem có tài nguyên nào liên quan đến địa chỉ mà người dùng đang cần tìm hay không. Trường hợp có, nó sẽ trả lại thông tin qua giao thức HTTP đến trình duyệt web để hiển thị cho người dùng. Còn nếu không thì nó sẽ xuất hiện các thông báo lỗi hoặc nội dung không tìm thấy. Cứ như vậy quy trình này được lặp đi lặp lại.

3.2. Các thành phần cơ bản của một Website

3.2.1. Phân loại Website

Hiện nay, Website có 2 loại: Website tĩnh (*Static Website*) và Website động (*Dynamic Website*)



Hình 2.36: Mô hình Static and Dynamic Website

Static Website

Static Website là thuật ngữ để chỉ những trang web thường được thiết kế có nội dung ít cần thay đổi và cập nhật, không có hệ quản trị cơ sở dữ liệu (*Database*). Người dùng khi truy cập vào website này sẽ không thể trò chuyện hay có các hoạt động nào khác để tương tác với nó. Do đó, nội dung của web tĩnh đã được lên khuôn ngay từ lúc lập trình và nếu muốn thêm bớt nội dung thì người quản lý phải biết làm lại khuôn mới.

Web tĩnh là một văn bản HTML đơn thuần, sau khi tải trang HTML từ máy chủ xuống, trình duyệt (*CocCoc, Opera, Firefox, ...*) sẽ biên dịch mã và hiển thị nội dung trang web, người dùng hầu như không thể tương tác với trang web. Nhiệm vụ của web tĩnh là đăng tải các thông tin giống như một tờ báo.

Về kiến trúc cơ bản thì web tĩnh thường được xây dựng từ CSS, HTML, JAVASCRIPT (*DHTML*), nhưng với sự phát triển vượt bậc của công nghệ hiện nay, còn có thêm sự hỗ trợ của HTML5 và CSS3. Nhờ được ứng dụng bằng những công nghệ hiện đại đó, mà các web tĩnh ngày càng trở nên sống động hơn trước rất nhiều. **Ngoài ra, nội dung website cũng có thể thay đổi được thông qua việc sử dụng công nghệ DHTML trong phần Client.**

Ứng dụng của web tĩnh: Thường được dùng cho những trang web nhỏ, có phần nội dung nhất định, ít bị thay đổi.

Dynamic Website

Dynamic Website là thuật ngữ chuyên ngành dùng để chỉ những trang web sử dụng hệ quản trị cơ sở dữ liệu. Web động được thiết kế có thêm mục truy xuất dữ liệu và phần xử lý thông tin.

Các thông tin được hiển thị trên web động được gọi là **Database**. Khi người dùng truy cập vào một trang web, các cơ sở dữ liệu này được gửi tới trình duyệt bằng những câu chữ, hình ảnh, âm thanh hay những dữ liệu số hoặc dạng bảng với nhiều hình thức khác nhau.

Ví dụ về một web động đó là: website bán hàng trực tuyến, website thương mại điện tử, các trang mạng thông tin lớn,... Vì vậy, web động sẽ có tính tương tác giữa doanh nghiệp và người dùng cao. Bạn hoàn toàn có thể quản trị nội dung website và điều hành thông qua các phần mềm hỗ trợ.

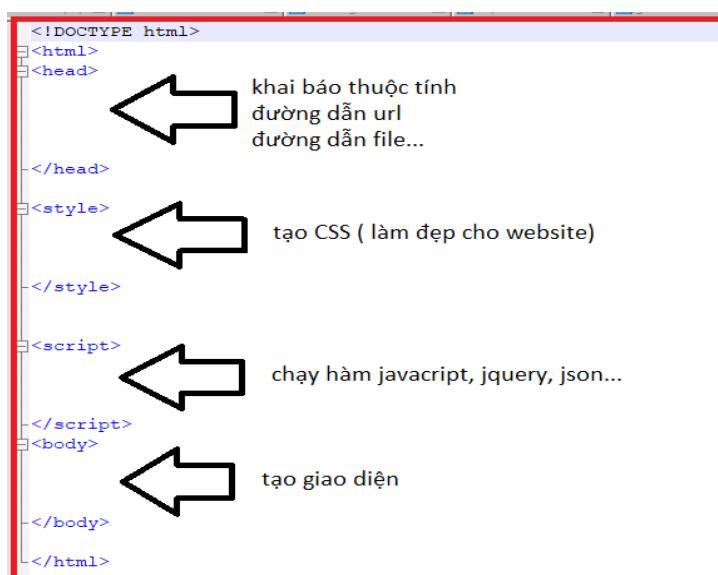
Web động thường được các nhà lập trình phát triển bằng các ngôn ngữ lập trình tiên tiến như: PHP, ASP.NET, ASP, Java, CGI, Perl và sử dụng các cơ sở dữ liệu mạnh như Access, Oracle, My SQL, MS SQL, DB2. Điều này giúp cho chủ sở hữu web động có thể quản lý được nội dung và điều hành, chỉnh sửa và cập nhật thông tin trên trang web dễ dàng mà không cần nhờ đến lập trình viên chuyên nghiệp nào.

Đặc biệt, các chương trình ứng dụng, thông tin được cập nhật, khách hàng có quyền trao đổi thông tin với chủ website và những khách hàng khác. Có thể nói, web động là thành công lớn trong lĩnh vực thiết kế website, là ứng dụng web với nhiều chức năng cao cấp được phát triển không ngừng.

Web động thường được sử dụng đối với những website có tầm cỡ lớn như website thương mại điện tử bán hàng, website tin tức, blog, làm web giới thiệu sản phẩm bán hàng cho các công ty...

→ Căn cứ trên đặc điểm của từng loại Website và nhiệm vụ của đồ án, tôi sử dụng Website tĩnh để tương tác với các thiết bị Zigbee.

3.2.2. Các thành phần của Static Website



Hình 2.37: Cấu trúc cơ bản của Static Website

Hypertext Markup Language (**HTML**)

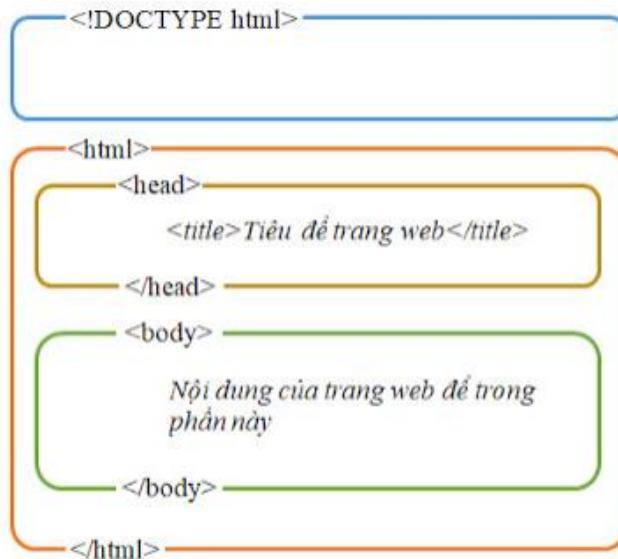
HTML là ngôn ngữ đánh dấu siêu văn bản và được sử dụng rộng rãi nhất để viết các trang Web. **Hypertext** là cách mà các trang Web (*các tài liệu HTML*) được kết nối với nhau. Và như thế, đường link có trên trang Web được gọi là Hypertext. Như tên gọi đã gợi ý, HTML là ngôn ngữ đánh dấu bằng thẻ (*Markup Language*), nghĩa là sử dụng HTML để đánh dấu một tài liệu text bằng các thẻ (*tag*) để nói cho trình duyệt Web cách để cấu trúc nó để hiển thị ra màn hình. Vậy HTML là ngôn ngữ sử dụng các thẻ để xây dựng nên khung sườn của 1 website.

HTML hoạt động bằng cách sử dụng một loạt thẻ để thông báo cho trình duyệt biết nó nên làm gì với văn bản trên trang và nó sẽ tải thêm tài nguyên từ đâu. Hiện có hơn 100 thẻ HTML để sử dụng, mặc dù hầu hết các trang web sẽ chỉ cần một số ít trong số này để hoạt động bình thường. HTML tags có hai loại chính là block-level tags (*phản tử cấp khối*) và inline tags (*phản tử nội tuyến*). Trong đó:

- ❖ Block-level Tags được sử dụng cho toàn không gian của web và bắt đầu bằng dòng mới của trang. 3 block level tags của mỗi trang HTML cần có những tag như là `<html>`, `<head>`, và `<body>`.
- ❖ Ngược lại, Inline Tags chỉ chiếm một phần nhỏ trên không gian web và không cần bắt đầu bằng dòng mới trên trang. Thường được sử dụng để tạo bố cục và định dạng cho phần nội dung của Block-Level Tags. Một số thẻ Inline phổ biến như `<a>`, ``, ``, `<i>`, ...

Cấu trúc cơ bản của HTML bao gồm 5 phần tử chính là `<!DOCTYPE>`, `<html>`, `<head>`, `<title>` và `<body>`:

- ❖ **<!DOCTYPE>**: Mọi tài liệu HTML phải bắt đầu bằng khai báo `<!DOCTYPE>` để thông báo cho trình duyệt web biết được trang được viết bằng phiên bản HTML nào.
- ❖ **<html>**: Cho biết rằng trang sẽ được định dạng bằng HTML và nội dung của trang đó sẽ sử dụng ngôn ngữ nào (*English*, *Vietnamese*, ...) được sử dụng cho một trang web thông qua thuộc tính `<lang>`.
- ❖ **<head>**: Chứa tất cả nội dung bạn muốn đưa vào trang HTML nhưng không hiển thị cho người xem trang. Bao gồm những thứ như từ khóa (*keywords*), mô tả trang (*page description*) mà bạn muốn xuất hiện trên công cụ tìm kiếm. Ngoài ra, `<head>` cũng thường chứa các liên kết CSS, JavaScript hay Meta Tags.
- ❖ **<title>**: Phần tử đặt tiêu đề cho trang, tiêu đề xuất hiện đầu tiên trong tab trình duyệt.
- ❖ **<body>**: Phần tử chứa tất cả nội dung mà bạn muốn hiển thị cho người dùng web khi họ truy cập trang của bạn như văn bản, hình ảnh, video, trò chơi, âm thanh hay bất kỳ nội dung nào khác.



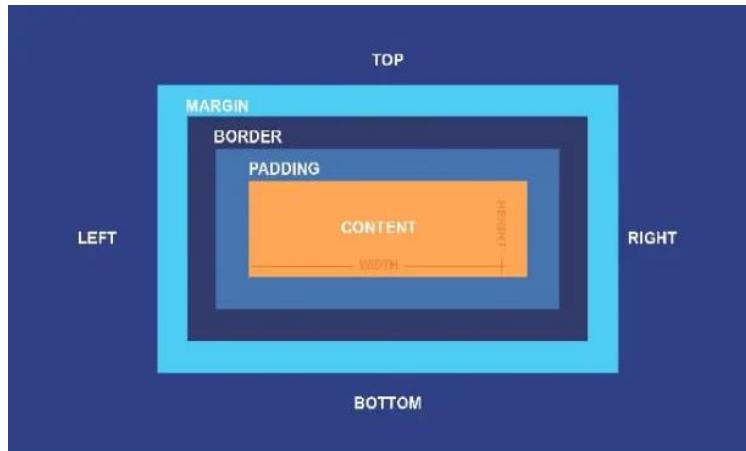
Hình 2.38: Cấu trúc cơ bản của HTML

HTML documents là files được kết thúc .html hay .htm. Với những file này bạn có thể xem bằng cách sử dụng bất kỳ trình duyệt nào (*Google Chrome, Firefox, Safari, ...*). Các trình duyệt đọc những file HTML này và biến đổi chúng thành một dạng visual trên Internet sao cho người dùng có thể xem và hiểu được chúng. Thông thường, trung bình một web chứa nhiều trang web HTML, ví dụ như: trang chủ, trang about, trang liên hệ, tất cả đều cần các trang HTML riêng. Mỗi trang HTML chứa một bộ các **tag** (cũng được gọi là **elements**), bạn có thể xem như là việc xây dựng từng khối của một trang web. Nó tạo thành cấu trúc cây thư mục bao gồm section, paragraph, heading, và những khối nội dung khác. Hầu hết các HTML elements đều có tag mở và tag đóng với cấu trúc như **<tag></tag>**.

Cascading Style Sheets (CSS)

CSS là ngôn ngữ dùng để tìm và định dạng miêu tả lại các phần tử được tạo ra bởi ngôn ngữ Markup cấu trúc HTML. Để nói một cách ngắn gọn, CSS là ngôn ngữ dùng để tạo nên phong cách cho trang web. Chúng ta có thể hiểu đơn giản rằng, nếu HTML đóng vai trò định dạng các phần tử, cấu trúc trên website như việc tạo ra các đoạn văn bản, các tiêu đề heading, bảng,... thì ngôn ngữ CSS sẽ giúp chúng ta có thể định dạng **“phong cách”** cho các phần tử HTML đó như thay đổi bố cục, màu sắc trang, màu chữ, font chữ, cấu trúc... Do đó, mối tương quan giữa HTML và CSS rất mật thiết và không thể tách rời.

Phương thức hoạt động của CSS là nó sẽ tìm dựa vào các vùng chọn, vùng chọn có thể là tên một thẻ HTML, tên một ID, class hay nhiều kiểu khác. Sau đó là nó sẽ áp dụng các thuộc tính cần thay đổi lên vùng chọn đó.



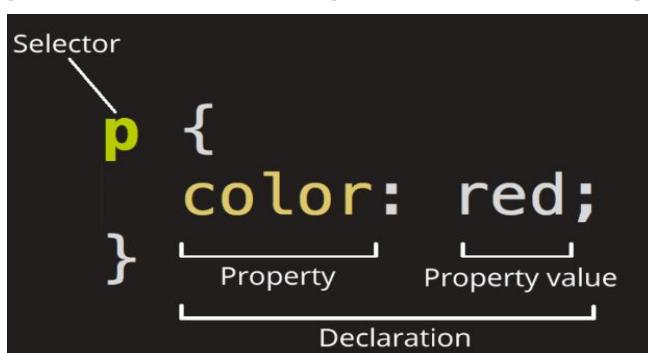
Hình 2.39: Bố cục của CSS

Bố cục CSS thường chủ yếu dựa vào hình hộp và mỗi hộp đều chiếm những khoảng trống trên trang của bạn với các thuộc tính như:

- ❖ Padding: Gồm không gian xung quanh nội dung (ví dụ: xung quanh đoạn văn bản).
- ❖ Border: Là đường连线 nằm ngay bên ngoài phần đệm.
- ❖ Margin: Là khoảng cách xung quanh bên ngoài của phần tử.

Cấu trúc CSS: Sẽ được khai báo bằng vùng chọn, sau đó các thuộc tính và giá trị sẽ nằm bên trong cặp dấu ngoặc nhọn {}. Mỗi thuộc tính sẽ luôn có một giá trị riêng, giá trị có thể là dạng số, hoặc các tên giá trị trong danh sách có sẵn của CSS. Phần giá trị và thuộc tính phải được cách nhau bằng dấu hai chấm, và mỗi một dòng khai báo thuộc tính sẽ luôn có dấu chấm phẩy ở cuối. Một vùng chọn có thể sử dụng không giới hạn thuộc tính.

vùng chọn {thuộc tính : giá trị; thuộc tính: giá trị; }



Hình 2.40: Cấu trúc của CSS

Selector (Bộ chọn): Selector cho phép người sử dụng chọn các phần tử HTML muốn định dạng mà không làm ảnh hưởng đến xung quanh. Có nhiều loại selector khác nhau:

- ❖ Selector theo tên phần tử: Chọn tất cả các phần tử có cùng tên.
- ❖ Selector theo ID: Chọn phần tử có ID cụ thể.

- ❖ Selector theo class: Chọn tất cả các phần tử có class cụ thể.
- ❖ Selector theo mối quan hệ: Chọn các phần tử dựa trên mối quan hệ của chúng với các phần tử khác.

Declaration (Khai báo): Declaration có chức năng xác định thuộc tính của một phần tử bất kỳ trong chương trình. Khối khai báo chứa một hoặc nhiều khai báo, phân tách với nhau bằng các dấu chấm phẩy.

Properties (Thuộc tính): Các thuộc tính CSS mà bạn muốn áp dụng cho các phần tử HTML đã chọn. Có rất nhiều thuộc tính CSS khác nhau:

- ❖ **Thuộc tính color (màu sắc):** Thay đổi màu sắc của văn bản, nền,...
- ❖ **Thuộc tính size (kích thước):** Thay đổi kích thước của văn bản, hình ảnh,...
- ❖ **Thuộc tính Position (vị trí):** Xác định vị trí của các phần tử HTML.
- ❖ **Thuộc tính font:** Thay đổi font chữ của văn bản.

Giá trị thuộc tính: Là giá trị của các thuộc tính CSS. Giá trị của các thuộc tính CSS cũng có nhiều loại khác nhau như:

- ❖ **Giá trị Number (số):** Giá trị số nguyên, số thập phân,...
- ❖ **Giá trị String (chuỗi):** Chuỗi ký tự.
- ❖ **Giá trị Color (màu sắc):** Tên màu, mã hex, giá trị RGB,...

Cách nhúng CSS vào Website: Để CSS có thể thực thi trên website hoặc HTML Documents thì phải tiến hành nhúng CSS vào website. Có 3 cách nhúng CSS vào website:

Inline CSS

```
<p style="color: blue;">This is a paragraph.</p>
```

Internal CSS

```
<head>
<style type = text/css>
  body {background-color: blue;}
  p { color: yellow;}
</style>
</head>
```

External CSS

```
<head>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Hình 2.41: Cách nhúng CSS vào Website

- ❖ **Inline CSS:** Đặt thuộc tính style vào thẻ mở của phần tử HTML, giá trị của thuộc tính style là các cặp thuộc tính định dạng CSS. Mã CSS chỉ tác động đến chính phần tử đó.

- ❖ **Internal CSS:** Đặt các cặp thuộc tính định dạng CSS vào bên trong cặp thẻ `<style type="text/css"></style>`. Cặp thẻ `<style type="text/css"></style>` thì được đặt bên trong cặp thẻ `<head></head>`.
- ❖ **External CSS:** Với External CSS: Ta đặt các thuộc tính định dạng vào bên trong tập tin CSS. Đây là một tập tin hoàn toàn độc lập so (*file này thường được đặt phần mở rộng là .css*) sau đó dùng thẻ link `<link rel="stylesheet" type="text/css" href="đường dẫn đến tập tin CSS">` đặt ở phần head (*cặp thẻ <head></head> của các tập tin HTML*) để có thể thực hiện nhúng tập tin CSS vào trang web.

→ Trong đồ án này, tôi sử dụng kết hợp 2 phương pháp nhúng CSS: **Inline & Internal CSS.**

Javascript (JS)

JavaScript là ngôn ngữ lập trình phổ biến dùng để tạo ra các trang web tương tác, được tích hợp và nhúng vào HTML giúp website trở nên sống động hơn. JavaScript cho phép trình duyệt web phản hồi tương tác của người dùng và thay đổi bố cục, nội dung trên trang web như: nút có thể nhấp vào, menu thả xuống, nội dung bổ sung khi làm mới trang, màu sắc của các phần tử thay đổi linh hoạt trên trang. Tất cả những đoạn mã JS đều sẽ được đặt ở trong cặp thẻ đóng và mở `<script></script>`.

Với khả năng nhúng vào HTML thì đây là những điều JavaScript có thể làm:

- ❖ **Thay đổi kiểu HTML:** Đây chính là một hoạt động biến thể của việc thay đổi thuộc tính của HTML ở trên.
Ví dụ: `document.getElementById('demo').style.fontSize = 35px;`
- ❖ **Ẩn các phần tử HTML:** Một hoạt động tiếp theo là Javascript có thể ẩn được các phần tử HTML. Chúng có thể được thực hiện thông qua hoạt động thay đổi kiểu hiển thị các phần tử HTML.
- ❖ **Hiển thị các phần tử HTML:** Một điểm đặc biệt là JavaScript có thể hiển thị được các yếu tố HTML ẩn. Đồng thời, cũng có thể thực hiện được thông qua cách thay đổi kiểu hiển thị phần tử.
- ❖ **Thay đổi nội dung HTML:** Một trong số nhiều phương thức HTML JavaScript chính là `getElementById()`. Chúng được sử dụng để tìm một phần tử của HTML với id = "demo" và dùng để thay đổi nội dung của phần tử (*Internal HTML*) sang thành "Hello JavaScript"
- ❖ **Thay đổi giá trị thuộc tính HTML:** Tổng quan về javascript còn có thể sử dụng để thay đổi các giá trị của thuộc tính. Ví dụ: thay đổi thuộc tính `src (source)` của tag ``.

Để có thể nhúng Javascript vào trang web bạn có thực hiện bằng 2 cách sau:

❖ **Nhúng Javascript trực tiếp vào HTML:**

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Tokyo Tech Lab Academy</title>
5          <script>
6              // Chưa đoạn mã Javascript ở đây (1)
7          </script>
8      </head>
9      <body>
10         <script>
11             // Chưa đoạn mã Javascript ở đây (2)
12         </script>
13     </body>
14 </html>
```

Hình 2.42: Nhúng Javascript trực tiếp vào HTML

Cách đầu tiên để nhúng Javascript vào trang web là đặt các đoạn mã Javascript trực tiếp vào trong HTML bằng cặp thẻ `<script></script>` và đặt thẻ này trong phần `<head>` hoặc `<body>` của tệp tin HTML của trang web.

❖ **Nhúng Javascript vào HTML bằng tệp tin riêng biệt:** Có thể việc nhúng Javascript vào HTML bằng cách trực tiếp đôi khi không phải là cách hay nhất. Bởi trong một số trường hợp, một vài đoạn mã Javascript có thể được sử dụng nhiều lần ở các trang khác nhau, điều này gây ra khó có thể kiểm soát được các đoạn mã. Vậy nên cách hiệu quả nhất là tạo ra một file Javascript riêng biệt để nhúng các đoạn mã Javascript vào trong HTML thông qua file đó giúp dễ dàng kiểm soát hơn. Với cách này, để nhúng JavaScript từ một tệp tin riêng biệt vào trong HTML, bạn cần sử dụng thẻ `<script>` cùng với thuộc tính `src` để chỉ định đường dẫn đến tệp JavaScript đó.

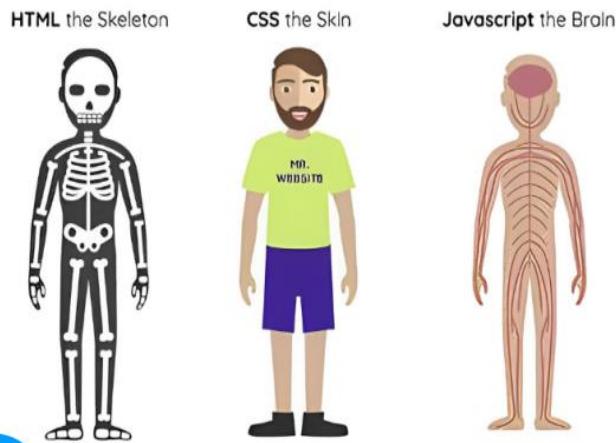
```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Tokyo Tech Lab Academy</title>
5          <script src="Nhập đường dẫn liên kết tại đây"></script>
6      </head>
7  </html>
```

Hình 2.43: Nhúng Javascript vào HTML bằng tệp tin riêng biệt

→ Trong đồ án này, tôi sử dụng kết hợp phương pháp: Nhúng Javascript trực tiếp vào HTML.

✚ Mối quan hệ giữa HTML, CSS và JavaScript



Hình 2.44: Minh họa mối quan hệ giữa HTML, CSS và JS

HTML, CSS và JavaScript được xem là bộ 3 công cụ không thể thiếu dùng để tạo nên một website hoàn chỉnh. HTML đóng vai trò như bộ khung, cấu trúc của website; CSS sẽ đảm nhiệm phần giao diện, màu sắc và mức độ thu hút; JavaScript sẽ nhận trọng trách “động” cho trang web. Do đó có thể thấy cả 3 ngôn ngữ đều có vai trò quan trọng và không thể thiếu để có một website hoàn hảo.

3.3. Kỹ thuật, giao thức sử dụng trong giao tiếp giữa Client – Server

3.3.1. Kỹ thuật AJAX (*Asynchronous JavaScript and XML*)

✚ Khái niệm AJAX

AJAX là một kỹ thuật lập trình web cho phép gửi, nhận dữ liệu giữa trang web và máy chủ mà không cần tải lại trang. Kỹ thuật này sử dụng các công nghệ web như JavaScript và XML (*hoặc JSON*) để giao tiếp với máy chủ và cập nhật dữ liệu trên trang web một cách động. AJAX giúp việc trao đổi dữ liệu nội bộ và Presentation Layer (*lớp hiển thị*) hoạt động đồng thời mà không ảnh hưởng đến chức năng của nhau. AJAX không phải một công nghệ đơn lẻ mà là sự kết hợp một nhóm công nghệ với nhau, trong đó:

- ❖ **Asynchronous**, hay nói ngắn hơn là Async – bất đồng bộ. Bất đồng bộ có nghĩa là một chương trình có thể xử lý không theo tuần tự các hàm. Sẽ không có quy trình, có thể nhảy đi bỏ qua bước nào đó. Ích lợi dễ thấy nhất của bất đồng bộ là chương trình có thể xử lý nhiều công việc một lúc.
- ❖ **JavaScript** là một ngôn ngữ lập trình nổi tiếng. Trong số rất nhiều chức năng của nó là khả năng quản lý nội dung động của website và hỗ trợ tương tác với người dùng.
- ❖ **XML** là một dạng của ngôn ngữ markup như HTML, chữ đầy đủ của nó

là eXtensible Markup Language. Nếu HTML được dùng để hiển thị dữ liệu, XML được thiết kế để chứa dữ liệu.

Cả **JavaScript** và **XML** đều hoạt động bất đồng bộ trong **AJAX**. Kết quả là, nhiều ứng dụng web có thể sử dụng AJAX để gửi và nhận data từ server mà không phải toàn bộ trang.

Nguyên lý hoạt động của AJAX

Ajax hoạt động bằng cách sử dụng các công nghệ web như JavaScript, DOM (*Document Object Model*), và XMLHttpRequest để gửi và nhận dữ liệu từ máy chủ mà không cần tải lại trang web. Các bước cơ bản để thực hiện Ajax như sau:

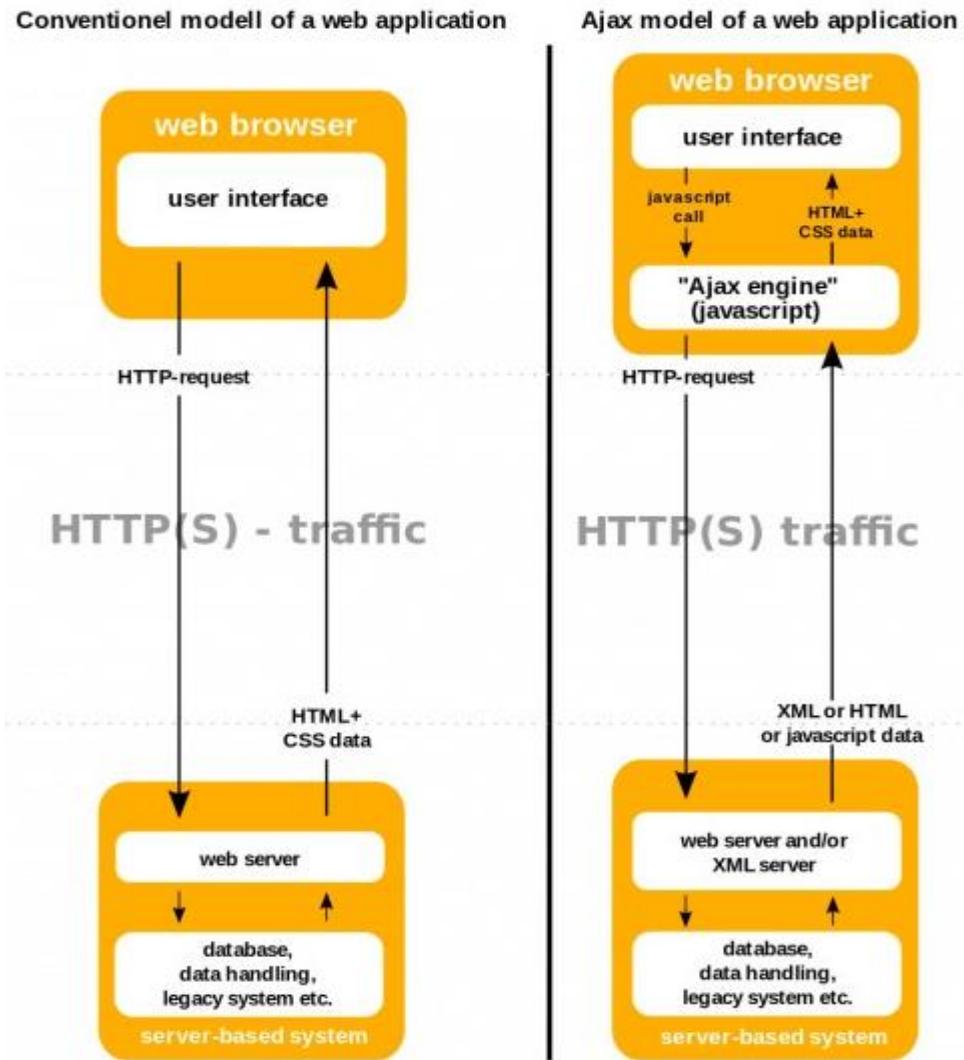
- ❖ Người dùng tương tác với trang web, và kích hoạt sự kiện bằng cách nhấn nút, nhập dữ liệu vào form, ...
- ❖ Mã JavaScript được thực thi để gửi yêu cầu đến máy chủ thông qua đối tượng XMLHttpRequest (*XHR*).
- ❖ Máy chủ xử lý yêu cầu và gửi lại dữ liệu đến máy khách (trình duyệt) dưới dạng XML, JSON hoặc văn bản thuần.
- ❖ Dữ liệu được xử lý bởi JavaScript và thêm vào trang web một cách động mà không cần tải lại trang.

Các bước trên xảy ra một cách đồng bộ trong khi yêu cầu và phản hồi được trao đổi giữa máy khách và máy chủ. Tuy nhiên, Ajax có thể hoạt động bất đồng bộ (*asynchronously*), có nghĩa là JavaScript sẽ tiếp tục thực thi trong khi đang chờ phản hồi từ máy chủ, giúp cho trang web không bị treo hoặc đóng băng trong khi đang tải dữ liệu.

So sánh sơ đồ hoạt động của Mô hình thông thường và Mô hình AJAX

STT	Mô hình thông thường	Mô hình AJAX
1	HTTP được gửi từ trình duyệt lên máy chủ.	Trình duyệt tạo một lệnh gọi JavaScript để kích hoạt XMLHttpRequest.
2	Máy chủ nhận, sau đó phản truy xuất thông tin.	Ở dưới nền, trình duyệt tạo một yêu cầu HTTP gửi lên server.
3	Server gửi dữ liệu được yêu cầu lại cho trình duyệt.	Server tiếp nhận, truy xuất và gửi lại dữ liệu cho trình duyệt.
4	Trình duyệt nhận dữ liệu và tải lại trang để hiển thị dữ liệu lên.	Trình duyệt nhận dữ liệu từ server và ngay lập tức hiển thị lên trang. Không cần tải lại toàn bộ trang.

Bảng 2.2: So sánh mô hình thông thường và mô hình AJAX



Hình 2.45: Mô hình thông thường và Mô hình AJAX

→ Một số ưu điểm của AJAX so với mô hình thông thường

- ❖ **Tiết kiệm băng thông:** AJAX được dùng để thực hiện một callback. Chúng ta sẽ không cần load lại cả trang web khi có một vài thay đổi nhỏ, khiến website tránh phải tải những thứ không cần thiết. Do đó, đối với những server nhỏ sẽ giúp tiết kiệm băng thông đáng kể.
- ❖ **Tăng tốc độ tải trang:** Do chỉ cần load lại một phần nhỏ khi có thay đổi hoặc nên giảm thiểu tốc độ tải trang nhanh hơn giúp nâng trải nghiệm người dùng tốt hơn.
- ❖ Trang web đa dạng và sống động hơn.

💡 Một số ví dụ thực tế về AJAX

- ❖ **Google Maps:** Google Maps sử dụng Ajax để cho phép người dùng tương tác với bản đồ mà không cần tải lại trang web.
- ❖ **Facebook:** Khi bạn trò chuyện với bạn bè trên Facebook, các tin nhắn mới sẽ được tải và hiển thị trực tiếp mà không cần tải lại trang web.

- ❖ **Amazon:** Khi bạn thêm sản phẩm vào giỏ hàng trên Amazon, các thông tin giỏ hàng của bạn sẽ được cập nhật bằng Ajax mà không cần tải lại trang web.
- ❖ **Twitter:** Khi bạn gửi một tweet trên Twitter, các tweet mới nhất của bạn bè và người theo dõi của bạn sẽ được hiển thị trực tiếp bằng Ajax mà không cần tải lại trang web.
- ❖ **Gmail:** Gmail sử dụng Ajax để cho phép người dùng gửi và nhận email mà không cần tải lại trang web hoàn toàn.

3.3.2. Giao thức TCP/IP

- ❖ **TCP/IP (Transmission Control Protocol/Internet Protocol – Giao thức điều khiển truyền nhận/Giao thức liên mạng)** là một bộ giao thức trao đổi thông tin được sử dụng để truyền tải và kết nối các thiết bị trong mạng Internet.
- ❖ TCP/IP là sự kết hợp giữa 2 giao thức TCP và IP:
 - ◆ TCP là một giao thức truyền tải đáng tin cậy được sử dụng để chia nhỏ và lắp ráp các gói dữ liệu trước khi chúng được gửi qua mạng. TCP đảm bảo rằng các gói dữ liệu được gửi đi sẽ đến đích một cách chính xác và đúng thứ tự. Nó xác định các kết nối, đồng bộ hóa truyền tải và quản lý lưu lượng dữ liệu.
 - ◆ IP là một giao thức định tuyến và chuyển tiếp dữ liệu trong mạng. IP định danh và địa chỉ hóa các thiết bị trong mạng, đồng thời quản lý việc định tuyến (*routing*) các gói dữ liệu qua mạng. Giao thức này đảm bảo gói dữ liệu được gửi đến đúng đích và định vị các thiết bị trong mạng.
 - ◆ Trong quá trình này, nếu giao thức TCP nhận thấy gói tin bị lỗi, một tín hiệu sẽ được truyền đi và yêu cầu hệ thống gửi lại một gói tin khác.
- ❖ Bộ giao thức TCP/IP sử dụng mô hình giao tiếp Client – Server. Trong đó, người dùng (*client*) được cung cấp dịch vụ (*nhiều gửi trang web*) bởi một máy chủ (*server*) trong mạng.
- ❖ Bộ giao thức TCP/IP có thể được coi là một tập hợp các tầng, mỗi tầng giải quyết một tập các vấn đề có liên quan đến việc truyền dữ liệu, và cung cấp cho các giao thức tầng cấp trên một dịch vụ được định nghĩa rõ ràng dựa trên việc sử dụng các dịch vụ của các tầng thấp hơn. Về mặt logic, các tầng trên gần với người dùng hơn và làm việc với dữ liệu trừu tượng hơn, chúng dựa vào các giao thức tầng cấp dưới để biến đổi dữ liệu thành các dạng mà cuối cùng có thể được truyền đi một cách vật lý.

- ❖ Các giao thức TCP/IP phổ biến (*HTTP, HTTPS, FTP được coi là 3 giao thức TCP/IP được sử dụng phổ biến nhất hiện nay*)
 - ◆ Giao thức HTTP: Mục đích sử dụng HTTP để truyền dữ liệu không an toàn giữa một web client và một web server. Bạn có thể hiểu đơn giản rằng, một web client (*trình duyệt Internet trên máy tính*) sẽ gửi một yêu cầu đến một web server để xem một trang web. Sau khi tiếp nhận yêu cầu, máy chủ web xử lý và gửi thông tin trang web về cho web client.
 - ◆ Giao thức HTTPS: HTTPS là giao thức được sử dụng để truyền thông tin dữ liệu bảo mật bởi 1 web client và 1 web server. Giao thức này được dùng để gửi dữ liệu giao dịch thẻ tín dụng hoặc các dữ liệu cá nhân khác từ một web client tới một web server.
 - ◆ FTP: FTP là giao thức trao đổi file dùng giữa hai hoặc nhiều máy tính với qua Internet. Nhờ FTP, dù đang ở xa, người dùng vẫn có thể truy cập vào máy chủ để truyền hoặc nhận dữ liệu.

→ Trong đồ án này, tôi sử dụng giao thức HTTP.

3.3.3. Giao thức DHCP

- ❖ **Giao thức DHCP** (*Dynamic Host Configuration Protocol*), tạm dịch là giao thức cấu hình máy chủ, đây là một giao thức quản lý mạng được sử dụng để chỉ định IP cho bất kỳ thiết bị hoặc nút nào trên mạng để chúng có thể giao tiếp với nhau bằng IP. DHCP tự động hóa và quản lý tập trung các cấu hình này thay vì yêu cầu quản trị viên mạng gán địa chỉ IP theo cách thủ công cho tất cả các thiết bị mạng. Máy tính được cấu hình một cách tự động vì thế sẽ giảm việc can thiệp vào hệ thống mạng. Nó cung cấp một database trung tâm để theo dõi tất cả các máy tính trong hệ thống mạng. Mục đích quan trọng nhất là tránh trường hợp hai máy tính khác nhau lại có cùng địa chỉ IP. Khi một thiết bị muốn truy cập vào mạng đang sử dụng DHCP, thiết bị đó sẽ yêu cầu địa chỉ IP từ máy chủ DHCP. Sau đó máy chủ DHCP sẽ phân phối địa chỉ IP đến thiết bị, giám sát việc sử dụng địa chỉ IP. Địa chỉ IP sau đó được sẽ được thu hồi và trả về nhóm địa chỉ do máy chủ DHCP quản lý để gán lại cho một thiết bị khác khi nó yêu cầu quyền truy cập vào mạng. DHCP cũng chỉ định nhiều tham số mạng liên quan bao gồm subnet mask, địa chỉ gateway mặc định và domain name server (*DNS*).
- ❖ **DHCP client** - Máy trạm DHCP: là một thiết bị nối vào mạng và sử dụng giao thức DHCP để lấy các thông tin cấu hình như là địa chỉ mạng, địa chỉ máy chủ DNS.

❖ **DHCP server** - Máy chủ DHCP: là một thiết bị nối vào mạng có chức năng trả về các thông tin cần thiết cho máy trạm DHCP khi có yêu cầu.

3.3.4. Giao thức HTTP

HTTP là viết tắt của HyperText Transfer Protocol, là một giao thức truyền tải siêu văn bản, giúp cho các máy tính có thể giao tiếp với nhau qua mạng. HTTP là nền tảng của World Wide Web kết nối giữa máy chủ (*server*) và máy khách (*client*) trong cùng một hệ thống mạng. Điều này cho phép chúng ta truy cập vào các trang web, tải xuống các tệp tin, xem các hình ảnh, video... Ban đầu khi được thiết kế vào những năm 90, HTTP là một giao thức linh hoạt có khả năng mở rộng theo thời gian. Giao thức này hoạt động trên nền tảng TCP/IP và thường được truyền qua kết nối mã hóa TLS để bảo vệ dữ liệu. Mặc dù về lý thuyết, bất kỳ giao thức truyền tải đáng tin cậy nào cũng có thể được áp dụng.

Với khả năng mở rộng đa dạng, HTTP không chỉ được sử dụng để tải các tài liệu siêu văn bản, mà còn để truyền tải hình ảnh, video và thậm chí để đăng tải nội dung lên máy chủ, chẳng hạn như kết quả của các biểu mẫu HTML. Ngoài ra, HTTP cũng có thể sử dụng để tải lên các phần của trang web, giúp cập nhật nội dung theo yêu cầu.

HTTP được sử dụng rộng rãi trong việc truy cập các trang web trên Internet. Khi bạn nhập một địa chỉ web vào trình duyệt, trình duyệt sẽ gửi một yêu cầu HTTP đến máy chủ web, và máy chủ web sẽ trả về một phản hồi HTTP chứa mã HTML của trang web đó. Trình duyệt sẽ đọc mã HTML và hiển thị trang web cho bạn xem. Bạn có thể thấy các yêu cầu và phản hồi HTTP bằng cách sử dụng công cụ kiểm tra mạng (*network inspector*) của trình duyệt.

Cấu trúc cơ bản của HTTP

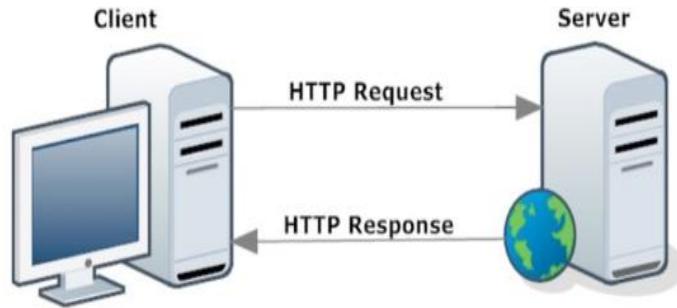
HTTP là một giao thức Request - Response được xây dựng trên cơ sở cấu trúc Client - Server. Trong cấu trúc này, Client (*thường là trình duyệt web*) và Server giao tiếp bằng cách trao đổi các message độc lập thay vì sử dụng một luồng dữ liệu duy nhất. Các yêu cầu là message được gửi bởi Client, trong khi phản hồi là message được gửi bởi Server.

Để hiểu rõ hơn về quá trình này, bạn có thể khám phá các mã trạng thái HTTP, hay còn được gọi là HTTP status code, để có cái nhìn chi tiết về các loại Yêu cầu và Phản hồi trong hệ thống HTTP. Để biết thêm thông tin, bạn có thể xem danh sách đầy đủ các HTTP status code.

Kết nối HTTP

HTTP sử dụng kết nối kiểm soát ở tầng truyền tải. Mặc dù không yêu cầu sự kết nối, HTTP phụ thuộc vào TCP, một giao thức đáng tin cậy, để thiết lập kết nối trước khi trao đổi thông tin giữa client và server. Trong HTTP/1.0, mỗi cặp yêu cầu - phản hồi mở một kết nối TCP mới, làm tăng độ trễ. HTTP/1.1 sử dụng pipelining

và kết nối liên tục để giảm thiểu vấn đề này, trong khi HTTP/2 tổ chức thông điệp qua một kết nối, làm cho kết nối ổn định hơn. Đang có các thử nghiệm để phát triển giao thức truyền tải tốt hơn, như thử nghiệm của Google với QUIC trên UDP để cung cấp sự đáng tin cậy và hiệu quả.



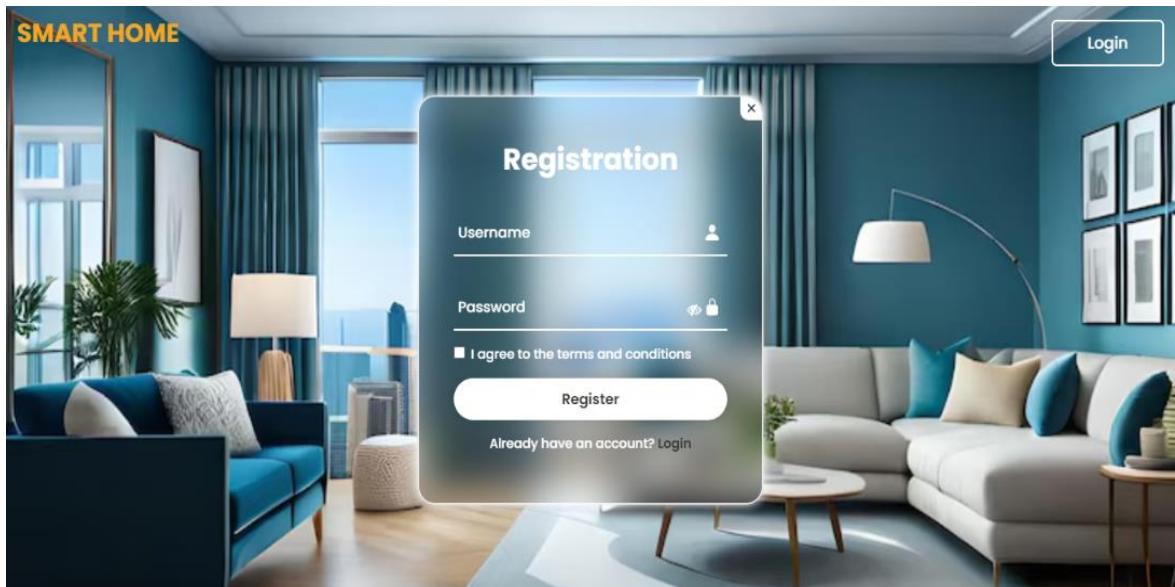
Hình 2.46: Mô hình giao tiếp Client – Server sử dụng giao thức HTTP

CHƯƠNG III: THIẾT BỊ CÔNG TẮC CẦU THANG

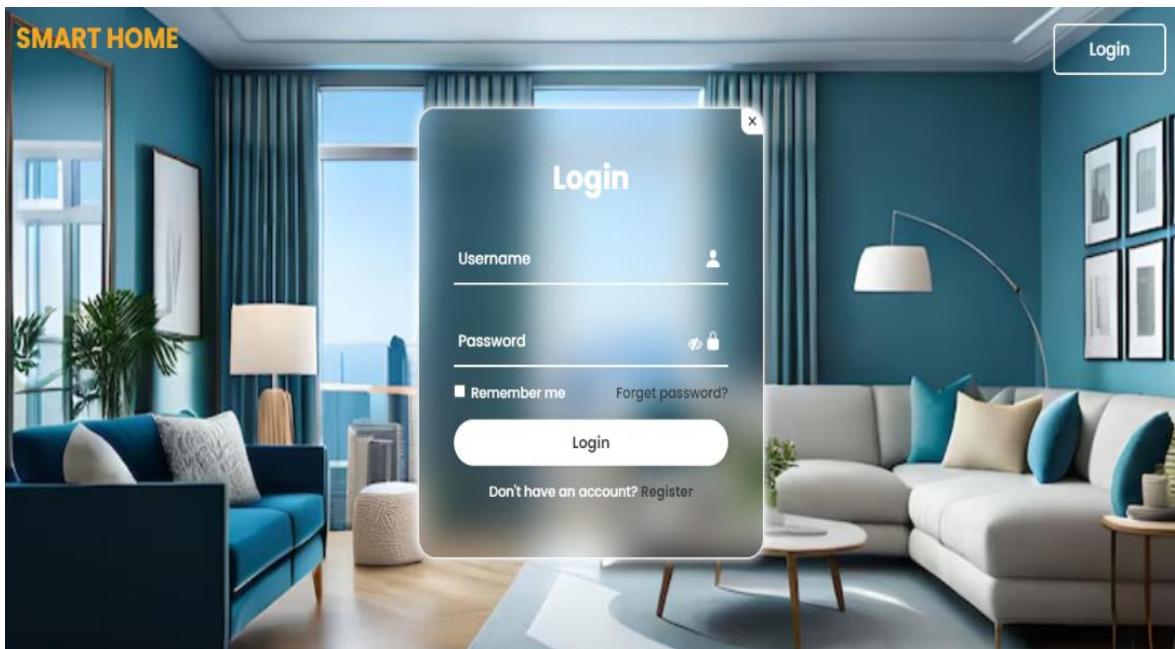
1. Tính năng thiết bị Công tắc cầu thang

1.1. Giao diện điều khiển

1.1.1. Đăng ký, đăng nhập mật khẩu để truy cập vào trình điều khiển



Hình 3.1: Giao diện Register



Hình 3.2: Giao diện Login

1.1.2. Tính năng mạng Zigbee trên Web

❖ Gửi bản tin điều khiển

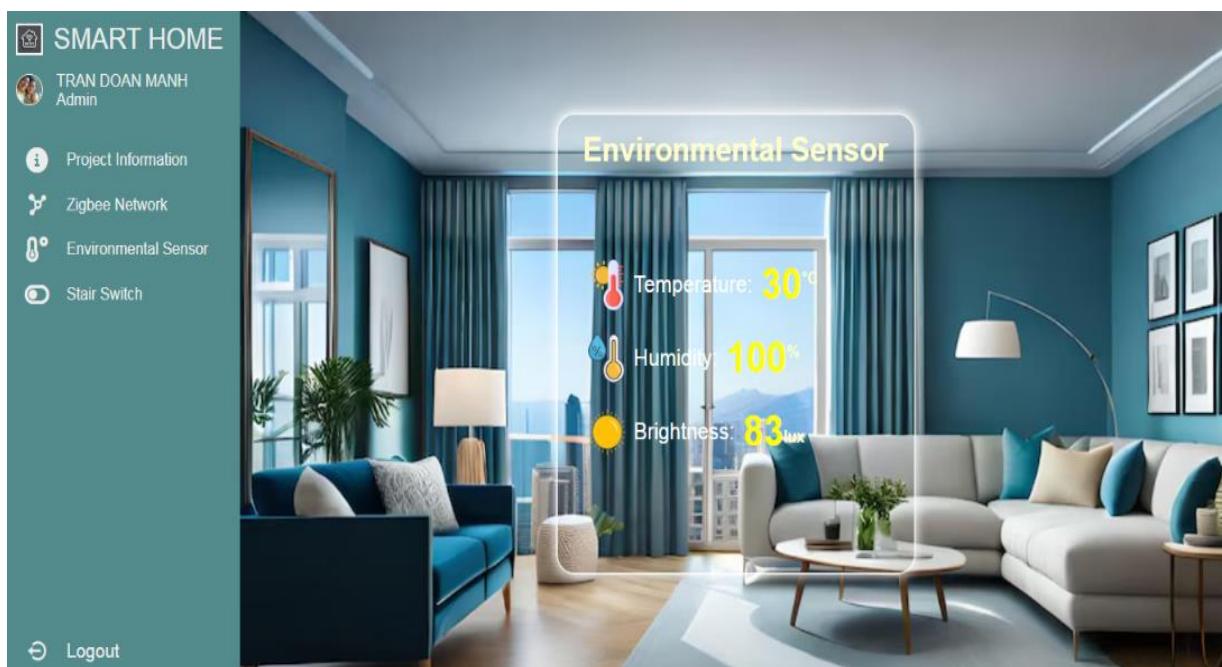
- ◆ Gửi bản tin tạo mạng cho ZC → ZC tạo mạng Centralized Network.
- ◆ Gửi bản tin mở mạng cho ZC → ZC mở mạng để các thiết bị SWs gia nhập mạng.

- ◆ Gửi bản tin dừng mở mạng cho ZC → ZC kết thúc mở mạng, các thiết bị SWs không thể gia nhập mạng.
- ◆ Gửi bản tin xóa thiết bị khỏi mạng cho ZC → ZC sẽ thực hiện xóa thiết bị khỏi mạng.
- ❖ **Nhận bản tin phản hồi**
 - ◆ Nhận response trạng thái của các thiết bị khi gia nhập/rời mạng từ ZC → Update lên Web.



Hình 3.3: Giao diện Zigbee Network

1.1.3. Cập nhật thông số cảm biến



Hình 3.4: Giao diện Environmental Sensor

- ❖ Nhận report giá trị nhiệt độ từ ZC → Update lên Web.
- ❖ Nhận report giá trị độ ẩm từ ZC → Update lên Web.
- ❖ Nhận report giá trị cường độ ánh sáng từ ZC → Update lên Web.

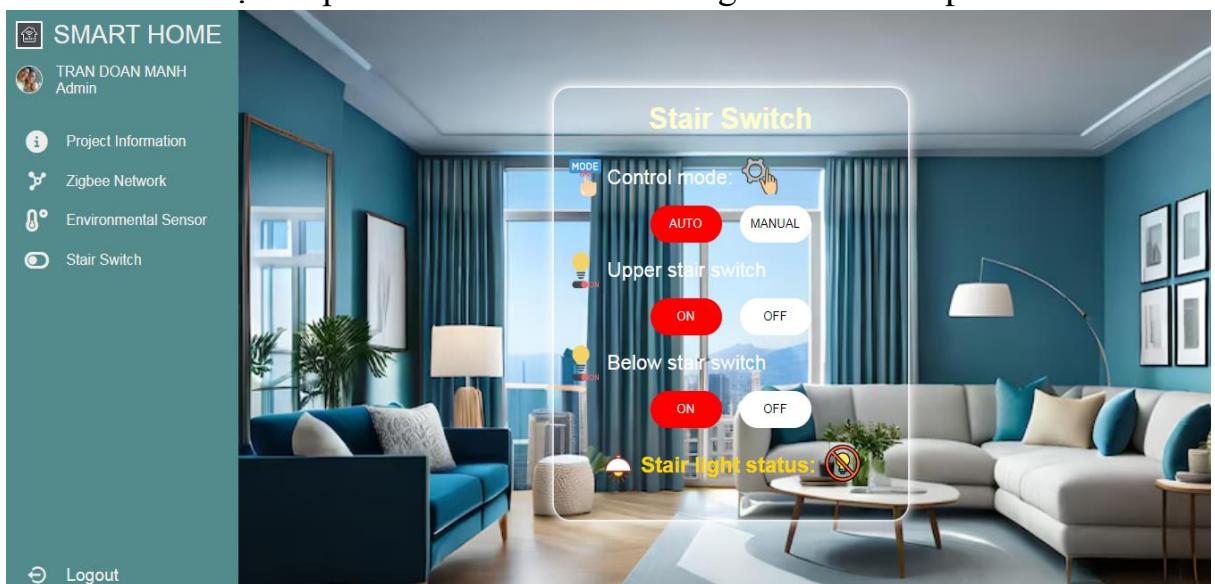
1.1.4. Điều khiển đèn cầu thang

❖ Gửi bản tin điều khiển

- ◆ Gửi bản tin Control Mode cho ZC → ZC gửi bản tin này cho SW để lựa chọn Control Mode.
- ◆ Gửi bản tin điều khiển bật/tắt đèn cầu thang Upper Switch cho ZC → ZC gửi bản tin này cho Upper Switch để điều khiển bật/tắt đèn.
- ◆ Gửi bản tin điều khiển bật/tắt đèn cầu thang Below Switch cho ZC → ZC gửi bản tin này cho Below Switch để điều khiển bật/tắt đèn.

❖ Nhận bản tin phản hồi

- ◆ Nhận response về Control Mode từ ZC → Update lên Web.
- ◆ Nhận response về Status of Stair Light từ ZC → Update lên Web.



Hình 3.5: Giao diện Stair Switch

1.2. Tính năng của Gateway (Zigbee Coordinator kết hợp ESP32)

- ❖ Nhận, xử lý bản tin điều khiển từ Web và truyền xuống các thiết bị trong mạng Zigbee để thực thi:
 - ◆ Tạo mạng **Centralized Network** và quản lý các thiết bị SWs khi chúng gia nhập mạng thành công.
 - ◆ Mở/gia nhập mạng.
 - ◆ Dừng/mở/gia nhập mạng.
 - ◆ Xóa các thiết bị khỏi mạng.
 - ◆ Gửi bản tin Control Mode cho thiết bị Below SW.
 - ◆ Gửi bản tin điều khiển bật/tắt đèn cho các thiết bị SWs.

- ❖ Nhận, xử lý bản tin report từ SWs và gửi lên Web để update
 - ◆ Bản tin report trạng thái gia nhập/rời mạng của các thiết bị SWs.
 - ◆ Bản tin report thông số môi trường: Nhiệt độ, độ ẩm, ánh sáng.
 - ◆ Bản tin report chế độ điều khiển.
 - ◆ Bản tin report trạng thái của đèn cầu thang khi điều khiển.

1.3. Tính năng của công tắc cầu thang

- ❖ Tìm kiếm, gia nhập mạng, rời mạng.
- ❖ Bật/tắt đèn cầu thang sử dụng đồng bộ 2 công tắc.
- ❖ Bật/tắt đèn cầu thang sử dụng cảm biến chuyển động PIR đồng bộ trên 2 công tắc (*kết hợp đồng bộ với sử dụng nút bấm cơ*).
- ❖ Bật/tắt đèn cầu thang khi nhận lệnh điều khiển từ ZC thông qua Web Local (*kết hợp đồng bộ với sử dụng nút bấm cơ và cảm biến chuyển động PIR*).

2. Hoạt động của thiết bị Công tắc cầu thang

2.1. Mô tả chi tiết hoạt động của công tắc cầu thang

2.1.1. Hoạt động của Zigbee Coordinator

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
1	Cấp nguồn cho ZC → Khởi tạo các ngoại vi & đăng ký các hàm callbacks xử lý → Kiểm tra mạng Zigbee của ZC	Chưa tạo mạng Zigbee	<ul style="list-style-type: none"> - Nháy LED1_Red 03 lần với $T = 3s$, $T_{on/off} = 300/300ms$. - Report thông tin ZC cho ESP32 (<i>Chỉ thực hiện 1 lần</i>).
2	Mạng Zigbee	Tạo mạng	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW1 03 lần với timeout giữa các lần nhấn $< 500ms$. - Method 2: Nhấn nút CREATE trên giao diện Zigbee Network của Web điều khiển. <p>Tạo mạng thành công: Nháy LED1_Blue 03 lần, $T_{on/off} = 300/300ms$.</p>
		Mở mạng	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW1 01 lần với timeout giữa các lần nhấn $> 500ms$. - Method 2: Nhấn nút OPEN trên giao diện Zigbee Network của Web điều khiển. <p>Mở mạng thành công: Nháy LED2_Green 01 lần, $T_{on/off} = 300/300ms \rightarrow$ Nhận & xử lý các bản tin reports của SWs (<i>gia nhập mạng, thông số cảm biến, trạng thái đèn, chế độ điều khiển</i>) \rightarrow Report thông tin SWs cho ESP32.</p>
		Đóng mạng	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW1 02 lần với timeout giữa các lần nhấn $< 500ms$. - Method 2: Nhấn nút STOP trên giao diện Zigbee Network của Web điều khiển. <p>Đóng mạng thành công: Nháy LED2_Red 01 lần, $T_{on/off} = 300/300ms$.</p>

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
2	Xóa thiết bị khỏi mạng	Xóa thiết bị Upper SW - Method 1: Nhấn nút SW2 01 lần. - Method 2: Nhấn nút UPPER trên giao diện Zigbee Network của Web điều khiển.	- Nháy LED2_Yellow 01 lần, Ton/off = 300/300ms. - Nhận & xử lý bản tin report rời mạng của Upper SW → Xóa thông tin thiết bị trong NVM3 → Report Upper SW rời mạng cho ESP32.
		Xóa thiết bị Below SW - Method 1: Nhấn nút SW2 02 lần với timeout giữa các lần nhấn < 500ms. - Method 2: Nhấn nút BELLOW trên giao diện Zigbee Network của Web điều khiển.	- Nháy LED2_Yellow 02 lần, Ton/off = 300/300ms. - Nhận & xử lý bản tin report rời mạng của Below SW → Xóa thông tin thiết bị trong NVM3 → Report Below SW rời mạng cho ESP32.
		Xóa cả 2 SWs - Method 1: Nhấn nút SW2 03 lần với timeout giữa các lần nhấn < 500ms. - Method 2: Nhấn nút ALL trên giao diện Zigbee Network của Web điều khiển.	- Nháy LED2_Yellow 03 lần, Ton/off = 300/300ms. - Nhận & xử lý bản tin report rời mạng của các SWs → Xóa thông tin các SWs trong NVM3 → Report các SWs rời mạng cho ESP32.

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
3	Lựa chọn chế độ điều khiển & Điều khiển đèn cầu thang	Nhấn nút AUTO, MANUAL, ON, OFF trên giao diện Stair Switch của Web điều khiển.	Nhận & xử lý lệnh điều khiển từ ESP32 → Gửi lệnh điều khiển xuống SWs tương ứng để thực thi.

Bảng 3.1: Hoạt động của Zigbee Coordinator

2.1.2. Hoạt động của Below Switch

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
1	Cáp nguồn cho Below SW → Khởi tạo các ngoại vi & đăng ký các hàm callbacks xử lý → Kiểm tra kết nối mạng Zigbee của Below SW	Đã kết nối mạng Zigbee	<ul style="list-style-type: none"> - Nháy LED2_Pink 05 lần, $T_{on/off} = 300/300ms \rightarrow$ Report trạng thái kết nối mạng lên ZC. - Kích hoạt Timer đọc thông số cảm biến: Thông số nhiệt độ, độ ẩm cho ZC với $T = 60s$ (<i>điều kiện nếu chênh lệch nhiệt độ $2^{\circ}C$, chênh lệch độ ẩm 2%</i>), thông số ánh sáng với $T = 30s$ (<i>điều kiện nếu chênh lệch 30 lux</i>) thì mới report giá trị thông số cảm biến cho ZC. - Thiết lập chế độ điều khiển MANUAL.
		Chưa kết nối mạng Zigbee	<ul style="list-style-type: none"> - Nháy LED2_Red 03 lần, $T_{on/off} = 300/300ms$. - Thực hiện quá trình tìm kiếm, gia nhập mạng: Thực hiện với $T = 5s +$ nháy LED2_Red 01 lần với $T_{on/off} = 500/500ms$ cho đến khi gia nhập mạng thành công.

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
		Gia nhập mạng thành công	<ul style="list-style-type: none"> - Nháy LED2_Blue 03 lần, $T_{on/off} = 300/300ms \rightarrow$ Report trạng thái kết nối mạng lên ZC. - Kích hoạt Timer đọc thông số cảm biến: Thông số nhiệt độ, độ ẩm cho ZC với $T = 60s$ (<i>điều kiện nếu chênh lệch nhiệt độ $2^{\circ}C$, chênh lệch độ ẩm 2%</i>), thông số ánh sáng với $T = 30s$ (<i>điều kiện nếu chênh lệch 30 lux</i>) thì mới report giá trị thông số cảm biến cho ZC. - Thiết lập chế độ điều khiển MANUAL.
2	Thiết lập Binding chéo với Upper SW để đồng bộ trạng thái đèn cầu thang	Nhấn nút SW2 03 lần với timeout giữa các lần nhấn $< 500ms \rightarrow$ Endpoint 1 sẽ được mở ở dạng Target or Initiator \rightarrow Endpoint 1 sẽ thực hiện quá trình Binding chéo với Endpoint 1 trên Upper SW.	<ul style="list-style-type: none"> - Nháy LED2_Green 01 lần, $T_{on/off} = 300/300ms$ nếu Endpoint 1 được cấu hình làm Target - Nháy LED2_Green 02 lần, $T_{on/off} = 300/300ms$ nếu Endpoint 1 được cấu hình làm Initiator - Nháy LED2_Green 03 lần, $T_{on/off} = 300/300ms$ nếu quá trình Binding thành công.
3	Thiết lập Binding chéo với Upper SW để đồng bộ time cảm biến chuyển động PIR và chế độ AUTO, MANUAL	Nhấn nút SW2 04 lần với timeout giữa các lần nhấn $< 500ms \rightarrow$ Endpoint 5 sẽ được mở ở dạng Target or Initiator \rightarrow Endpoint 5 sẽ thực hiện quá trình Binding chéo với Endpoint 2 trên Upper SW.	<ul style="list-style-type: none"> - Nháy LED2_Blue 01 lần, $T_{on/off} = 300/300ms$ nếu Endpoint 5 được cấu hình làm Target - Nháy LED2_Blue 02 lần, $T_{on/off} = 300/300ms$ nếu Endpoint 5 được cấu hình làm Initiator - Nháy LED2_Blue 03 lần, $T_{on/off} = 300/300ms$ nếu quá trình Binding thành công.

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
4	Chọn chế độ điều khiển AUTO or MANUAL	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW2 02 lần với timeout giữa các lần nhấn < 500ms → Chuyển đổi giữa hai chế độ AUTO và MANUAL. - Method 2: Nhấn nút AUTO or MANUAL trên giao diện Stair Switch của Web điều khiển. - Method 3: Lựa chọn chế độ điều khiển từ Upper SW. 	<ul style="list-style-type: none"> - Chế độ AUTO: LED1_Blue sáng → Report chế độ điều khiển cho Upper SW để đồng bộ sử dụng Binding. - Chế độ MANUAL: LED1_Red sáng → Report chế độ điều khiển cho Upper SW để đồng bộ sử dụng Binding. - Report chế độ điều khiển lên ZC.
5	Điều khiển đèn cầu thang	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW2 01 lần với timeout giữa các lần nhấn > 500ms → Chuyển đổi giữa hai trạng thái On/Off đèn cầu thang. - Method 2: Nhấn nút ON/OFF tương ứng với BelowSW trên giao diện Stair Switch của Web điều khiển. - Method 3: Điều khiển đèn cầu thang từ Upper SW. 	<ul style="list-style-type: none"> - TURN ON: <i>Đèn cầu thang - LED2_Blue</i> trên Below SW sáng → Report trạng thái đèn cầu thang cho Upper SW để đồng bộ sử dụng Binding. - TURN OFF: <i>Đèn cầu thang - LED2_Blue</i> trên Below SW tắt → Report trạng thái đèn cầu thang cho Upper SW để đồng bộ sử dụng Binding. - Report trạng thái đèn cầu thang lên ZC.

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
6	Chế độ điều khiển dùng PIR	Khi phát hiện có chuyển động	<ul style="list-style-type: none"> - Report cho Upper SW để đồng bộ time sử dụng Binding. - Nếu đèn OFF, TURN ON đèn cầu thang → Report trạng thái đèn cầu thang cho Upper SW để đồng bộ sử dụng Binding → Report trạng thái của đèn cầu thang lên ZC.
		Thực hiện tìm kiếm chuyển động sau 2s/1 lần.	<ul style="list-style-type: none"> - Nếu phát hiện chuyển động: Report cho Upper SW để đồng bộ time sử dụng Binding → Tiếp tục TURN ON đèn cầu thang (<i>không thực hiện report trạng thái đèn cho Upper SW, không report trạng thái đèn lên ZC</i>). - Nếu phát hiện không còn chuyển động: Tính toán thời gian từ lần chuyển động cuối cùng.
		Sau 30s kể từ lần phát hiện chuyển động cuối cùng mà không có chuyển động	Nếu đèn ON , TURN OFF đèn cầu thang → Report trạng thái đèn cầu thang cho Upper SW để đồng bộ sử dụng Binding → Report trạng thái của đèn cầu thang lên ZC.
7	Điều khiển 02 SWs rời mạng	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW2 07 lần với timeout giữa các lần nhấn < 500ms. - Method 2: Nhấn nút BELLOW, ALL trên giao diện Zigbee Network của Web điều khiển. 	<p>Khi SW nhận được bản tin rời mạng, thì nó sẽ thực hiện:</p> <ul style="list-style-type: none"> - Nháy LED2_Yellow 05 lần, $T_{on/off} = 300/300\text{ms}$. - Report trạng thái rời mạng lên ZC. - Reboot lại thiết bị, bắt đầu lại quá trình tìm kiếm mạng.

Bảng 3.2: Hoạt động của Below Switch

2.1.3. Hoạt động của Upper Switch

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
1	Cấp nguồn cho Upper SW → Khởi tạo các ngoại vi & đăng ký các hàm callbacks xử lý → Kiểm tra kết nối mạng Zigbee của Upper SW	Đã kết nối mạng Zigbee	<ul style="list-style-type: none"> - Nháy LED2_Pink 05 lần, $T_{on/off} = 300/300ms \rightarrow$ Report trạng thái kết nối mạng lên ZC. - Thiết lập chế độ điều khiển MANUAL.
		Chưa kết nối mạng Zigbee	<ul style="list-style-type: none"> - Nháy LED2_Red 03 lần, $T_{on/off} = 300/300ms$. - Thực hiện quá trình tìm kiếm, gia nhập mạng: Thực hiện với $T = 5s +$ nháy LED2_Red 01 lần với $T_{on/off} = 500/500ms$ cho đến khi gia nhập mạng thành công.
		Gia nhập mạng thành công	<ul style="list-style-type: none"> - Nháy LED2_Blue 03 lần, $T_{on/off} = 300/300ms \rightarrow$ Report trạng thái kết nối mạng lên ZC. - Thiết lập chế độ điều khiển MANUAL.
2	Thiết lập Binding chéo với Below SW để đồng bộ trạng thái đèn cầu thang	Nhấn nút SW2 03 lần với timeout giữa các lần nhấn $< 500ms \rightarrow$ Endpoint 1 sẽ được mở ở dạng Initiator or Target → Endpoint 1 sẽ thực hiện quá trình Binding chéo với Endpoint 1 trên Below SW.	<ul style="list-style-type: none"> - Nháy LED2_Green 01 lần, $T_{on/off} = 300/300ms$ nếu Endpoint 1 được cấu hình làm Target - Nháy LED2_Green 02 lần, $T_{on/off} = 300/300ms$ nếu Endpoint 1 được cấu hình làm Initiator - Nháy LED2_Green 03 lần, $T_{on/off} = 300/300ms$ nếu quá trình Binding thành công.

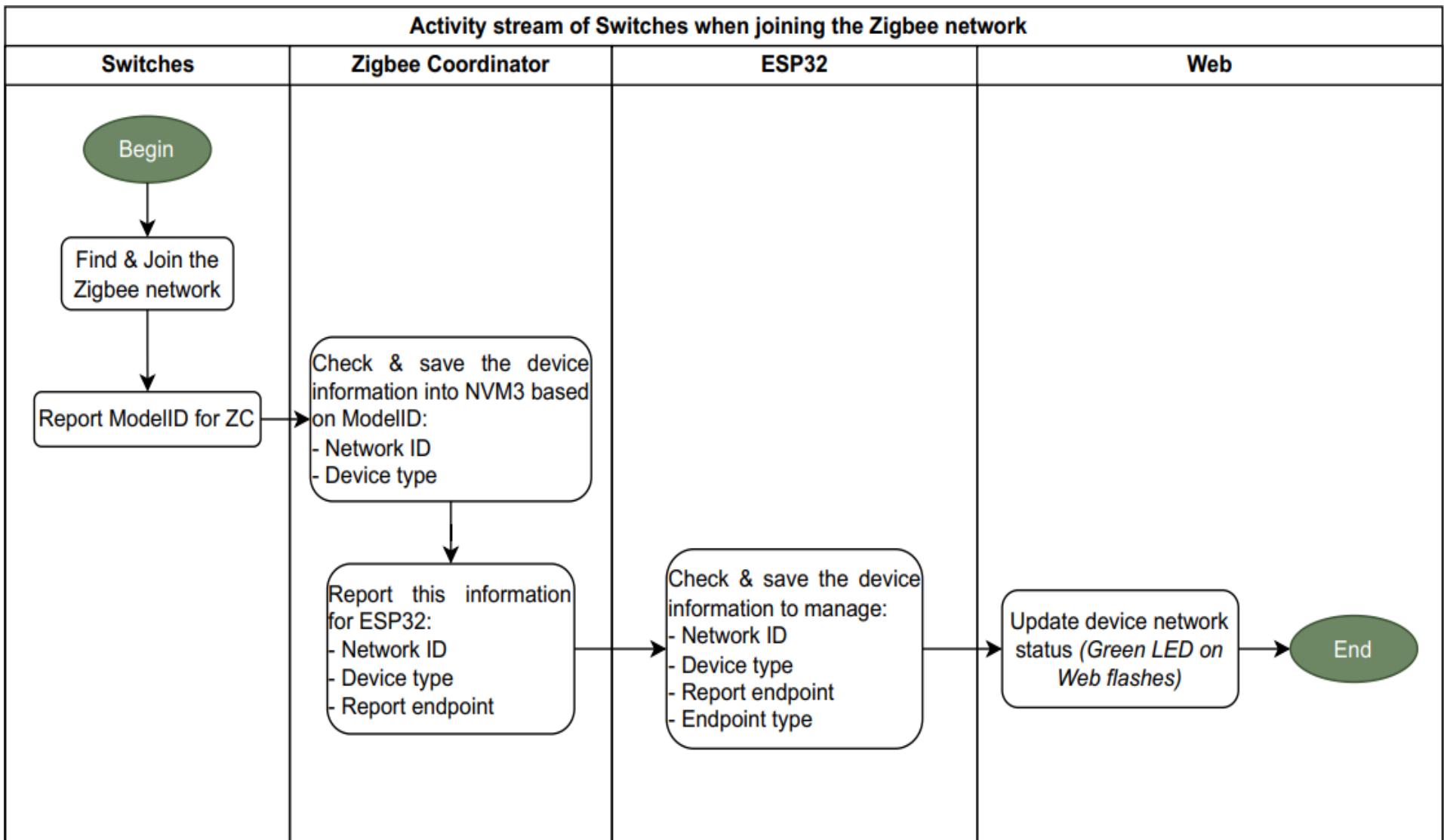
TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
3	Thiết lập Binding chéo với Below SW để đồng bộ time cảm biến chuyển động PIR và chế độ AUTO, MANUAL	Nhấn nút SW2 04 lần với timeout giữa các lần nhấn < 500ms → Endpoint 2 sẽ được mở ở dạng Initiator or Target → Endpoint 2 sẽ thực hiện quá trình Binding chéo với Endpoint 5 trên Below SW.	<ul style="list-style-type: none"> - Nháy LED2_Blue 01 lần, $T_{on/off} = 300/300ms$ nếu Endpoint 2 được cấu hình làm Target - Nháy LED2_Blue 02 lần, $T_{on/off} = 300/300ms$ nếu Endpoint 2 được cấu hình làm Initiator - Nháy LED2_Blue 03 lần, $T_{on/off} = 300/300ms$ nếu quá trình Binding thành công.
4	Chọn chế độ điều khiển AUTO or MANUAL	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW2 02 lần với timeout giữa các lần nhấn < 500ms sẽ chuyển đổi giữa hai chế độ AUTO và MANUAL. - Method 2: Lựa chọn chế độ điều khiển từ Below SW. 	<ul style="list-style-type: none"> - Chế độ AUTO: LED1_Blue sáng → Report chế độ điều khiển cho Below SW để đồng bộ sử dụng Binding. - Chế độ MANUAL: LED1_Red sáng → Report chế độ điều khiển cho Below SW để đồng bộ sử dụng Binding.
5	Điều khiển đèn cầu thang	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW2 01 lần với timeout giữa các lần nhấn > 500ms → Chuyển đổi giữa hai trạng thái On/Off đèn cầu thang. - Method 2: Nhấn nút ON/OFF tương ứng với UpperSW trên giao diện Stair Switch của Web điều khiển. 	<ul style="list-style-type: none"> - TURN ON: Gửi lệnh để Below SW bật đèn cầu thang. - TURN OFF: Gửi lệnh để Below SW tắt đèn cầu thang. <p>(Cập nhật trạng thái đèn sau khi tiếp nhận & xử lý bản tin report trạng thái đèn từ Below SW để điều khiển đồng bộ)</p>

TT	Hoạt động của công tắc cầu thang	Mô tả chi tiết hoạt động	
		Tình huống	Hiệu ứng & Hoạt động
6	Chế độ điều khiển dùng PIR	Khi phát hiện có chuyển động	<ul style="list-style-type: none"> - Report cho Below SW để đồng bộ time sử dụng Binding. - Nếu đèn OFF, Gửi lệnh để Below SW bật đèn cầu thang → Cập nhật trạng thái đèn sau khi tiếp nhận & xử lý bản tin report trạng thái đèn từ Below SW để điều khiển đồng bộ.
		Thực hiện tìm kiếm chuyển động sau 2s/1 lần.	<ul style="list-style-type: none"> - Nếu phát hiện chuyển động: Report cho Below SW để đồng bộ time sử dụng Binding → Tiếp tục TURN ON đèn cầu thang (<i>không gửi lệnh để Below SW bật đèn cầu thang</i>). - Nếu phát hiện không còn chuyển động: Tính toán thời gian từ lần chuyển động cuối cùng.
		Sau 30s kể từ lần phát hiện chuyển động cuối cùng mà không có chuyển động	Đèn cầu thang tắt → Cập nhật trạng thái đèn sau khi tiếp nhận & xử lý bản tin report trạng thái đèn từ Below SW để điều khiển đồng bộ.
7	Điều khiển 02 SWs rời mạng	<ul style="list-style-type: none"> - Method 1: Nhấn nút SW2 07 lần với timeout giữa các lần nhấn < 500ms. - Method 2: Nhấn nút UPPER, ALL trên giao diện Zigbee Network của Web điều khiển. 	<p>Khi SW nhận được bản tin rời mạng, thì nó sẽ thực hiện:</p> <ul style="list-style-type: none"> - Nháy LED2_Yellow 05 lần, $T_{on/off} = 300/300\text{ms}$. - Report trạng thái rời mạng lên ZC. - Reboot lại thiết bị, bắt đầu lại quá trình tìm kiếm mạng.

Bảng 3.3: Hoạt động của Upper Switch

2.2. Luồng hoạt động của thiết bị công tắc cầu thang

2.2.1. Luồng gia nhập mạng Zigbee của Switches



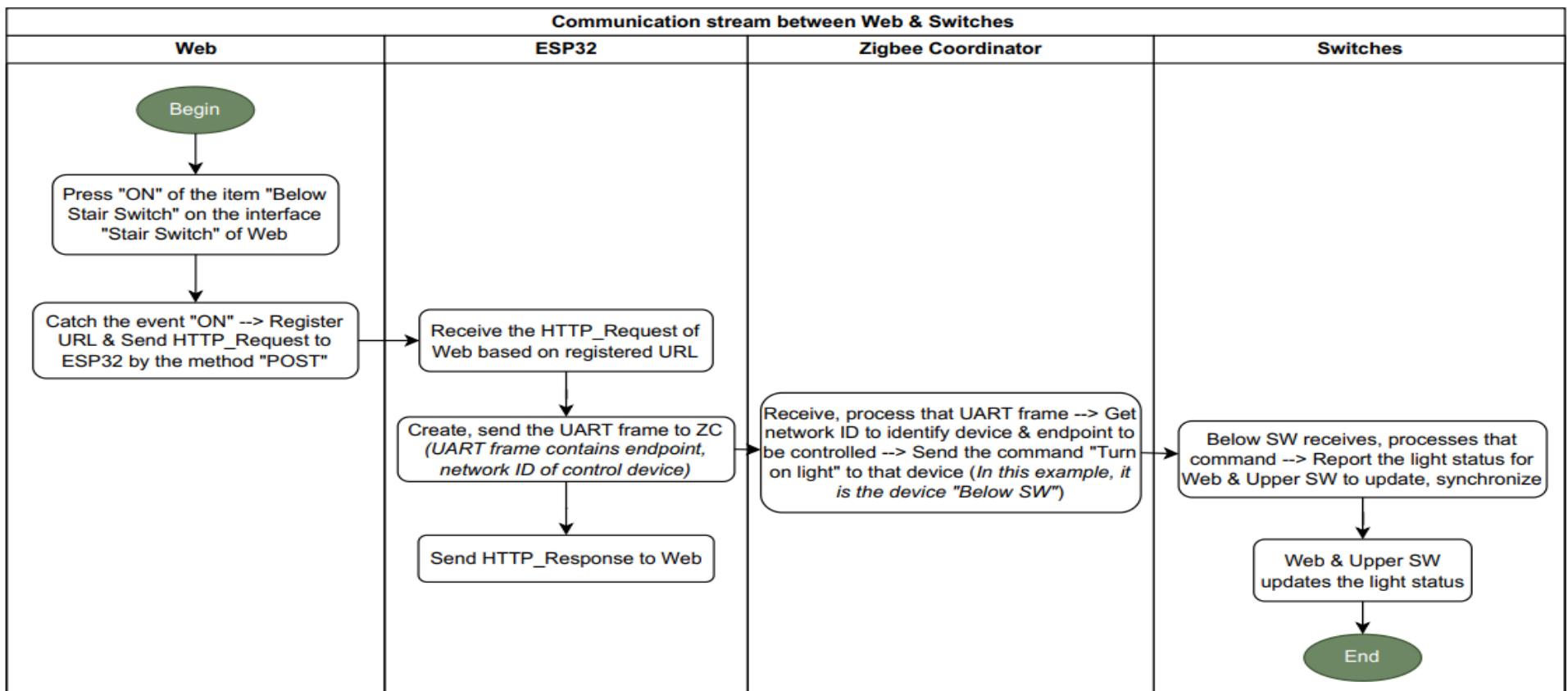
Hình 3.6: Luồng gia nhập mạng Zigbee của Switches

2.2.2. Luồng giao tiếp giữa Web & Switches

Việc điều khiển đèn cầu thang, lựa chọn chế độ điều khiển được thực hiện đồng bộ theo 2 phương pháp:

- **Phương pháp 1:** Sử dụng nút bấm cơ trên Switches.
- **Phương pháp 2:** Sử dụng nút bấm trên giao diện Stair Switch của Web điều khiển.

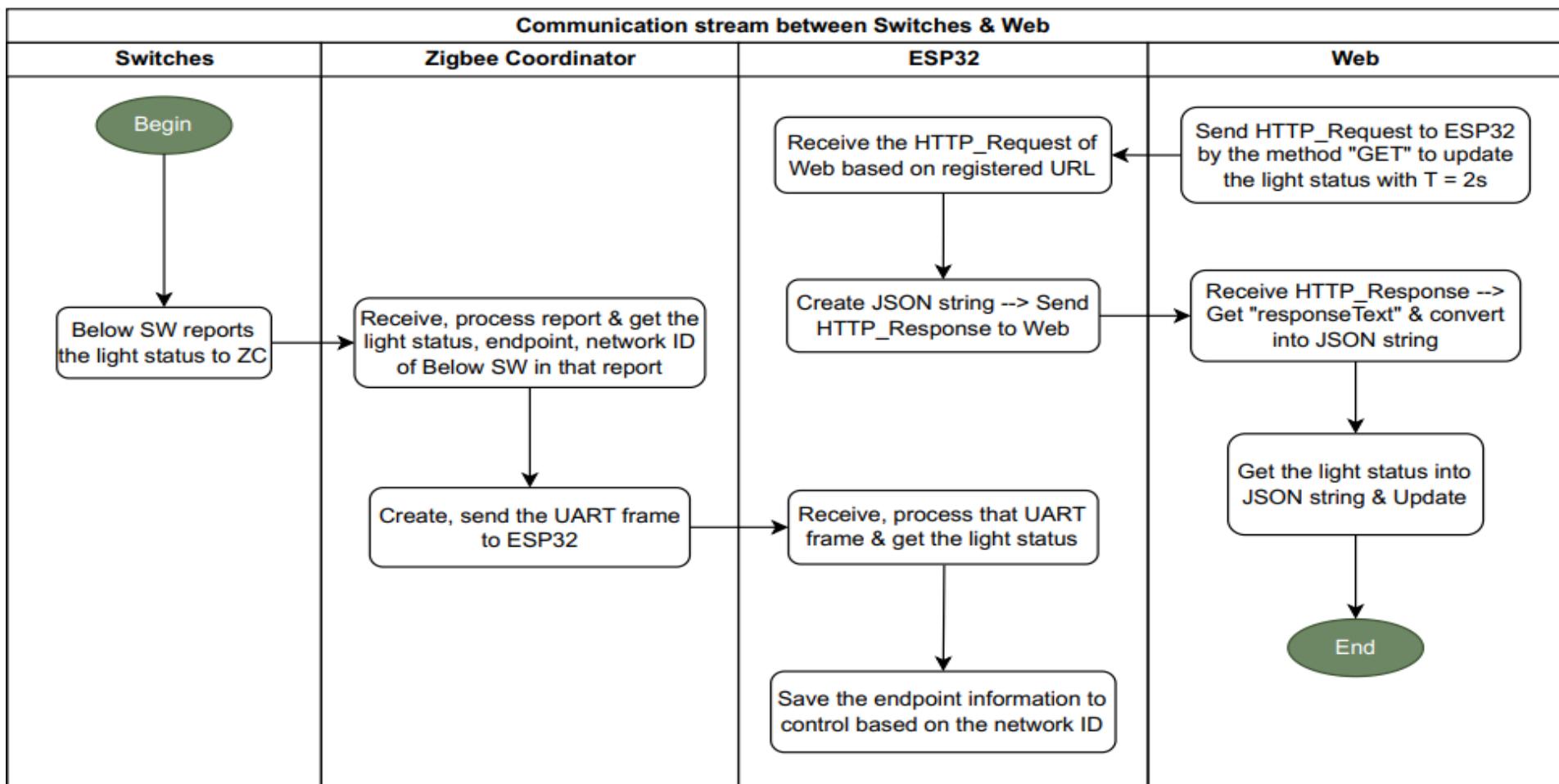
Trong phần này sẽ giải thích luồng giao tiếp giữa Web & Switches khi nhấn nút điều khiển trên giao diện Web. Hình dưới đây là minh họa luồng giao tiếp khi sử dụng Web điều khiển để “**Bật đèn cầu thang trên Below SW**” (*các lệnh điều khiển khác từ Web hoạt động tương tự*):



Hình 3.7: Luồng giao tiếp giữa Web & Switches

2.2.3. Luồng giao tiếp giữa Switches & Web

Sau khi nhận lệnh điều khiển từ Web, Switches thực thi lệnh & Below SW report trạng thái đèn, chế độ điều khiển để cập nhật trên giao diện Web. Ngoài ra, Switches còn report trạng thái gia nhập, rời mạng & Below SW report thông số cảm biến theo định kỳ để cập nhật trên giao diện Web. Hình dưới đây là minh họa luồng giao tiếp khi Below SW report trạng thái đèn để cập nhật trên giao diện Web (*các report khác từ Switches hoạt động tương tự*):



Hình 3.8: Luồng giao tiếp giữa Switches & Web

3. Khung truyền UART

3.1. Vai trò của khung truyền UART

Khung truyền UART sử dụng trong quá trình giao tiếp giữa thiết bị Zigbee Coordinator và module ESP32. Khung truyền UART được sinh ra để đồng nhất quy tắc giao tiếp giữa 2 thiết bị, từ đó có thể dễ dàng phân tích, xử lý hoặc xây dựng bản tin giao tiếp.

Start (1 byte)	Frame Length (1 byte)	Network ID (2 bytes)	Endpoint (1 byte)	Command ID (1 byte)	Command Type (1 byte)	Data Length (1 byte)	Data (n bytes)	CXOR (1 byte)
-------------------	--------------------------	-------------------------	----------------------	------------------------	--------------------------	-------------------------	-------------------	------------------

Hình 3.9: Khung truyền UART

3.2. Các thành phần trong khung truyền UART

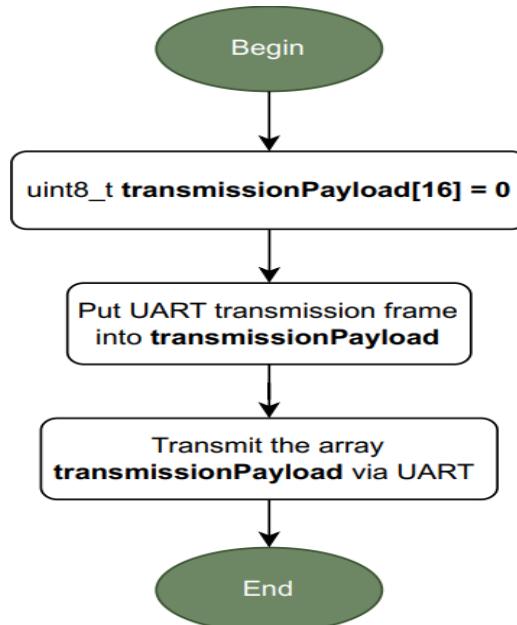
STT	Tên trường	Mô tả & mục đích	Giá trị
1	Start (1 byte)	Byte bắt đầu của khung truyền. Bên nhận chỉ xử lý bản tin nhận được khi xác định được Start byte. Trường hợp không xác định được Start byte, bản tin nhận được không hợp lệ.	0xB1
2	Frame Length (1 byte)	Thể hiện số bytes trong khung truyền sau byte này. Sử dụng byte Frame Length để tính toán, so sánh tính toàn vẹn của bản tin nhận được.	0x00 – 0xFF (Giá trị 0 được xem là khung truyền không hợp lệ).
3	Network ID (2 bytes)	Địa chỉ mạng của thiết bị trong mạng Zigbee (16 bits). Sử dụng Network ID để định danh, lưu trữ, truy xuất thông tin và điều khiển thiết bị trong mạng Zigbee.	0x0000 – 0xFFFF (Địa chỉ 0x0000 là địa chỉ mạng của Zigbee Coordinator).
4	Endpoint (1 byte)	Sử dụng Endpoint để định danh, điều khiển các tính năng của thiết bị.	0 – 240 (Endpoint 0 chứa tính năng chung cho toàn bộ các thiết bị trong mạng Zigbee, Endpoint 1 – 240 thể hiện các tính năng riêng của các thiết bị đó).
5	Command ID (1 byte)	Sử dụng Command ID để định danh bản tin truyền nhận UART.	
6	Command Type (1 byte)	Sử dụng Command Type để chỉ định kiểu bản tin truyền nhận UART, cụ thể như sau:	0x01: SET 0x02: REPORT

STT	Tên trường	Mô tả & mục đích	Giá trị
		<ul style="list-style-type: none"> ➤ Set: Bản tin điều khiển thiết bị ESP32 gửi cho ZC. ➤ Report: Bản tin report ZC gửi cho ESP32. 	
7	Data Length (1 byte)	Độ dài của dữ liệu truyền	0x00 – 0xFF
8	Data (n bytes)	Dữ liệu truyền	
9	CXOR (1 byte)	<ul style="list-style-type: none"> - Sử dụng CXOR để kiểm tra tính toàn vẹn của bản tin nhận được, xác định có lỗi hay không. Trường hợp giá trị XOR tính toán khi nhận được bản tin UART bằng với giá trị CXOR trong bản tin, khi đó lệnh trong bản tin mới được thực thi. - Công thức tính như sau: $\text{CXOR} = 0xFF \wedge \text{NWKID (2 bytes)} \\ \wedge \text{ENDP} \wedge \text{CMDID} \wedge \text{CMDTYPE} \wedge \\ \text{DataLength} \wedge \text{Data (n bytes)}$ 	0x00 – 0xFF

Bảng 3.4: Thành phần khung truyền UART

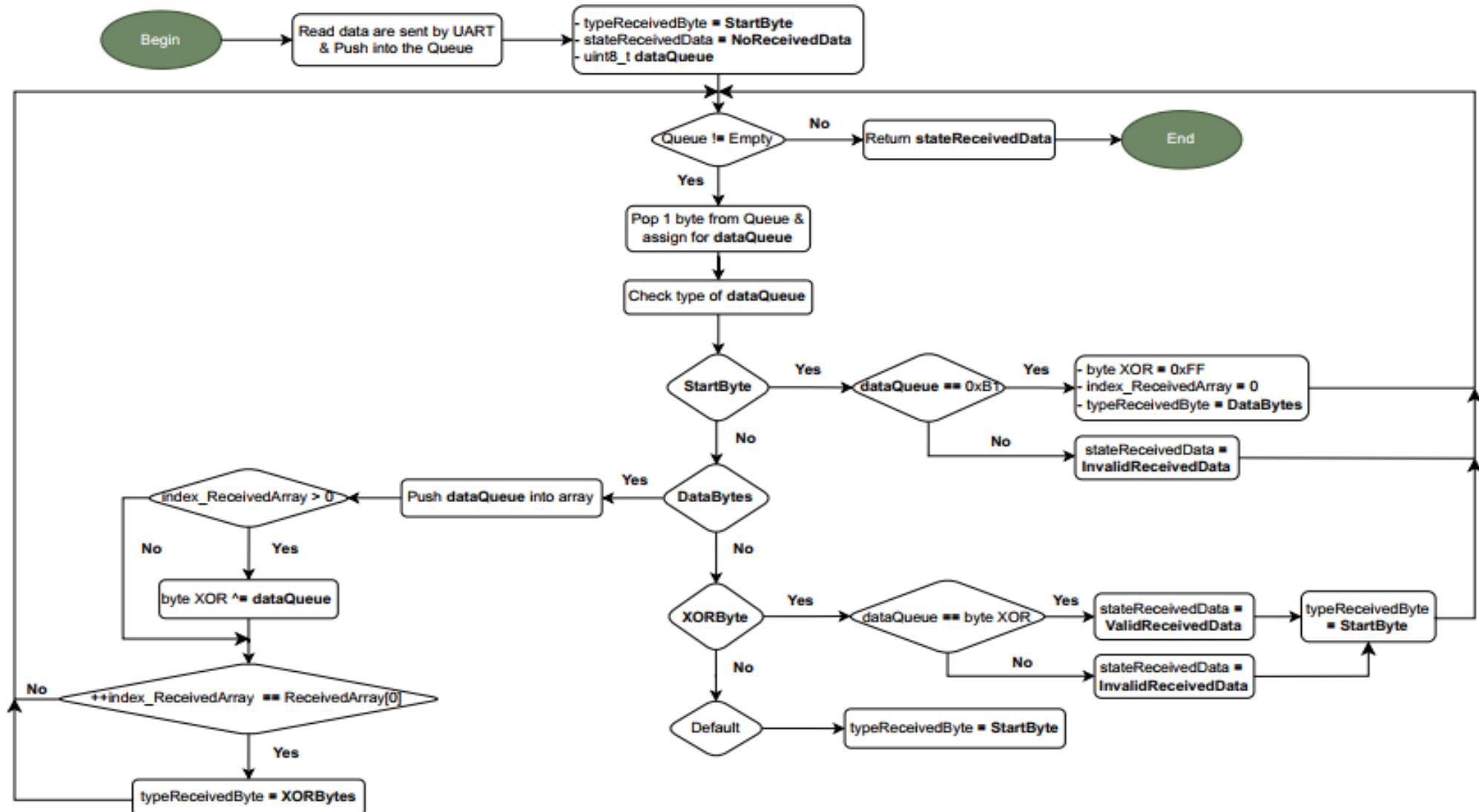
3.3. Sơ đồ thuật toán UART

3.3.1. Sơ đồ thuật toán truyền dữ liệu UART



Hình 3.10: Sơ đồ thuật toán truyền dữ liệu UART

3.3.2. Sơ đồ thuật toán xử lý dữ liệu UART nhận được



Hình 3.11: Sơ đồ thuật toán xử lý dữ liệu UART nhận được

4. Sơ đồ hoạt động, thuật toán và phân tầng

4.1. Khái niệm về các loại sơ đồ

- ❖ **Sơ đồ hoạt động:** Thể hiện tổng thể các tính năng được xây dựng theo yêu cầu của bài toán lập trình. Đây là loại sơ đồ đầu tiên mà một lập trình nhúng phải thực hiện sau khi đọc hiểu yêu cầu của bài toán lập trình, là nền tảng để xây dựng sơ đồ thuật toán và sơ đồ phân tầng. Lập sơ đồ hoạt động giúp lập trình viên hình dung rõ hơn về sản phẩm, về các bước cần thiết để tạo ra sản phẩm và tránh lập trình thiếu các tính năng so với bài toán yêu cầu.

Sau khi hoàn thiện sơ đồ hoạt động, từ sơ đồ đó lập trình viên thực hiện chia nhỏ những tính năng thành những module nhỏ hơn để độc lập chức năng của từng block, không gây rối. Khi thiết bị xảy ra lỗi, lập trình viên dễ dàng biết được loại module nào đang gây lỗi và khi sửa hoàn toàn không gây ảnh hưởng tới luồng hoạt động của cả toàn bộ sản phẩm.

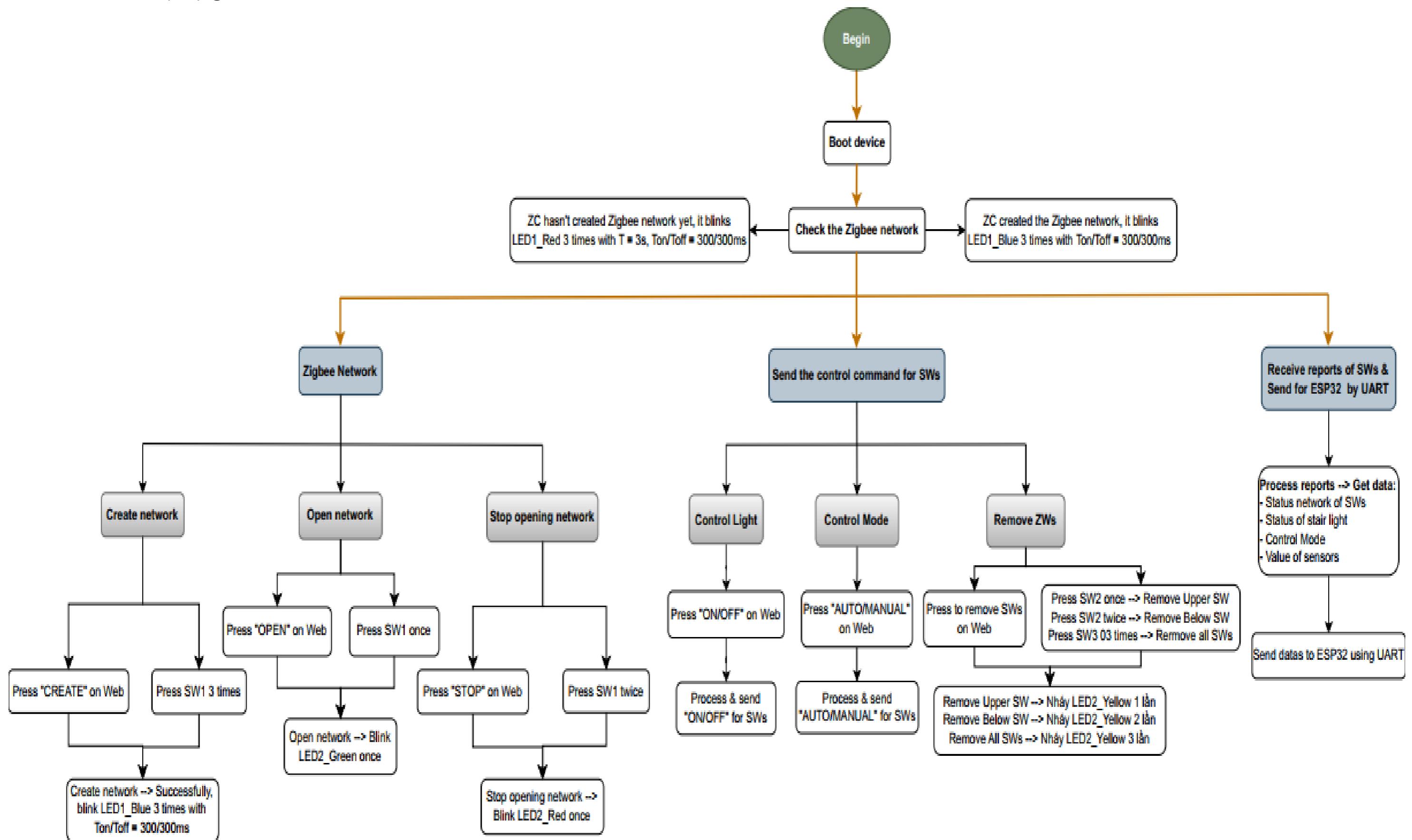
- ❖ **Sơ đồ thuật toán:** Được xây dựng cho từng module nhỏ sau khi đã phân tách từ sơ đồ hoạt động, nó thể hiện các bước logic để lập trình một module cụ thể. Sơ đồ thuật toán sẽ giúp lập trình viên có cái nhìn tổng quan trước về chức năng của toàn bộ module đó, cũng như khi ghép nối các loại modules vào có xảy ra vấn đề gì hay không, thay vì thực hiện coding trực tiếp từ những ý tưởng trong não bộ, điều này rất dễ xảy ra tình trạng xung đột giữa các module với nhau, hoặc code thiếu tính năng trong module.
- ❖ **Sơ đồ phân tầng:** Đây là loại sơ đồ gần nhất đối với chương trình code của người lập trình. Lúc này, người lập trình sẽ cần phải phân chia những module mình đang có vào từng tầng trong sơ đồ. Tương ứng với mỗi module, chúng ta sẽ ước tính ra những hàm, callbacks chính để phục vụ cho loại module đó. **Trong đồ án này**, sơ đồ phân tầng được chia làm 3 tầng: App, Mid và Driver. Tầng App sẽ là tầng sử dụng ứng dụng, chứa các thành phần như Main, và các module liên quan đến xử lý Zigbee. Tầng Mid xử lý các tác vụ liên quan tới nút bấm, led. Và tầng Driver sẽ phụ trách những thành phần liên quan đến hạ tầng phần cứng (*nếu có*).

4.2. Zigbee Coordinator

4.2.1. Vai trò

Zigbee Coordinator (ZC) là thiết bị đóng vai trò quản lý tất cả các thiết bị tồn tại trong mạng Zigbee, cho phép thiết bị gia nhập mạng, rời mạng, quản lý trạng thái từng thiết bị và phân phối bản tin giao tiếp giữa Web và các thiết bị trong mạng Zigbee.

4.2.2. Sơ đồ hoạt động của ZC

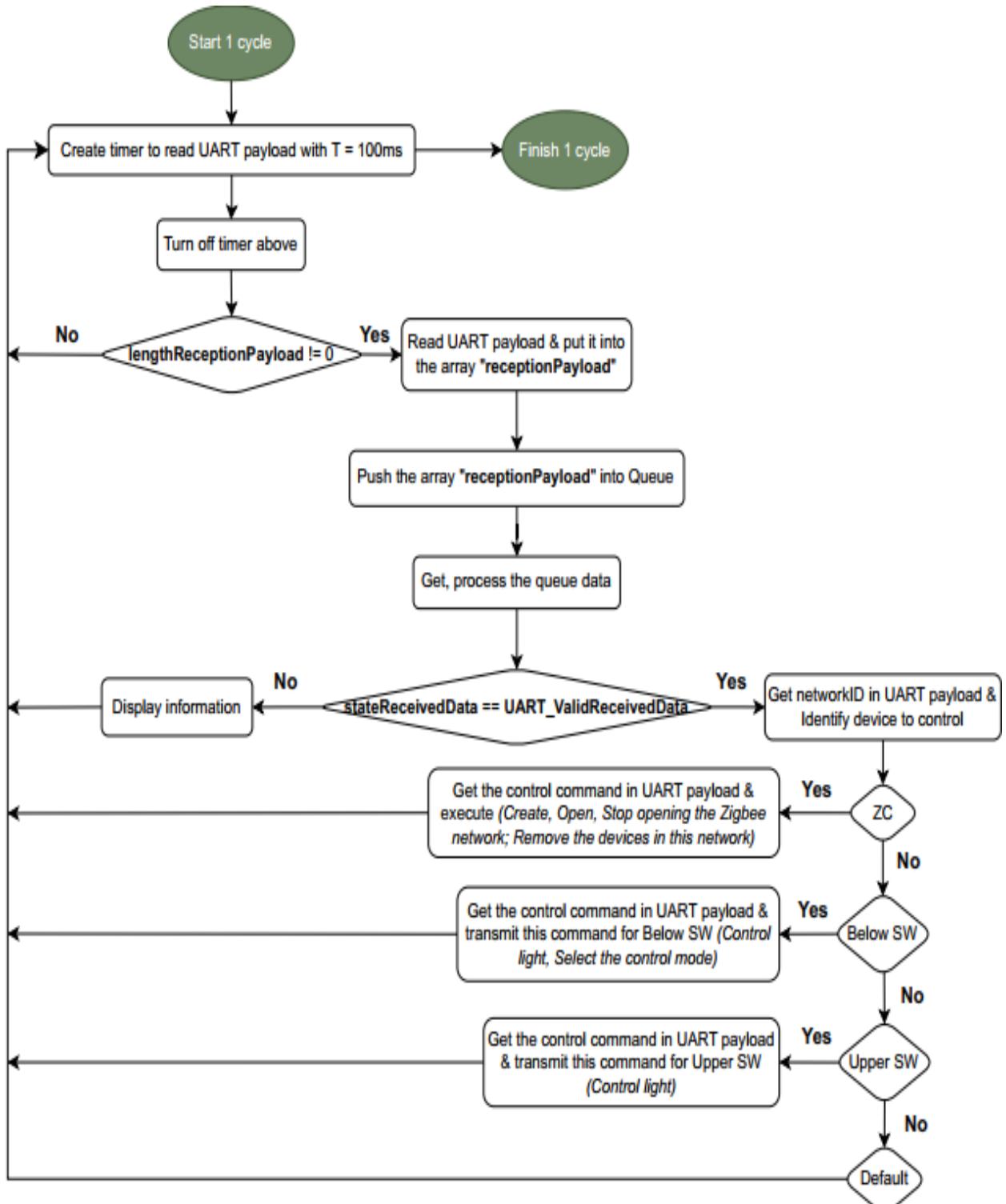


Hình 3.12: Sơ đồ hoạt động của ZC

4.2.3. Sơ đồ thuật toán của ZC

✚ ZC xử lý dữ liệu nhận được từ Web

❖ Sơ đồ thuật toán ZC xử lý bản tin nhận được từ Web

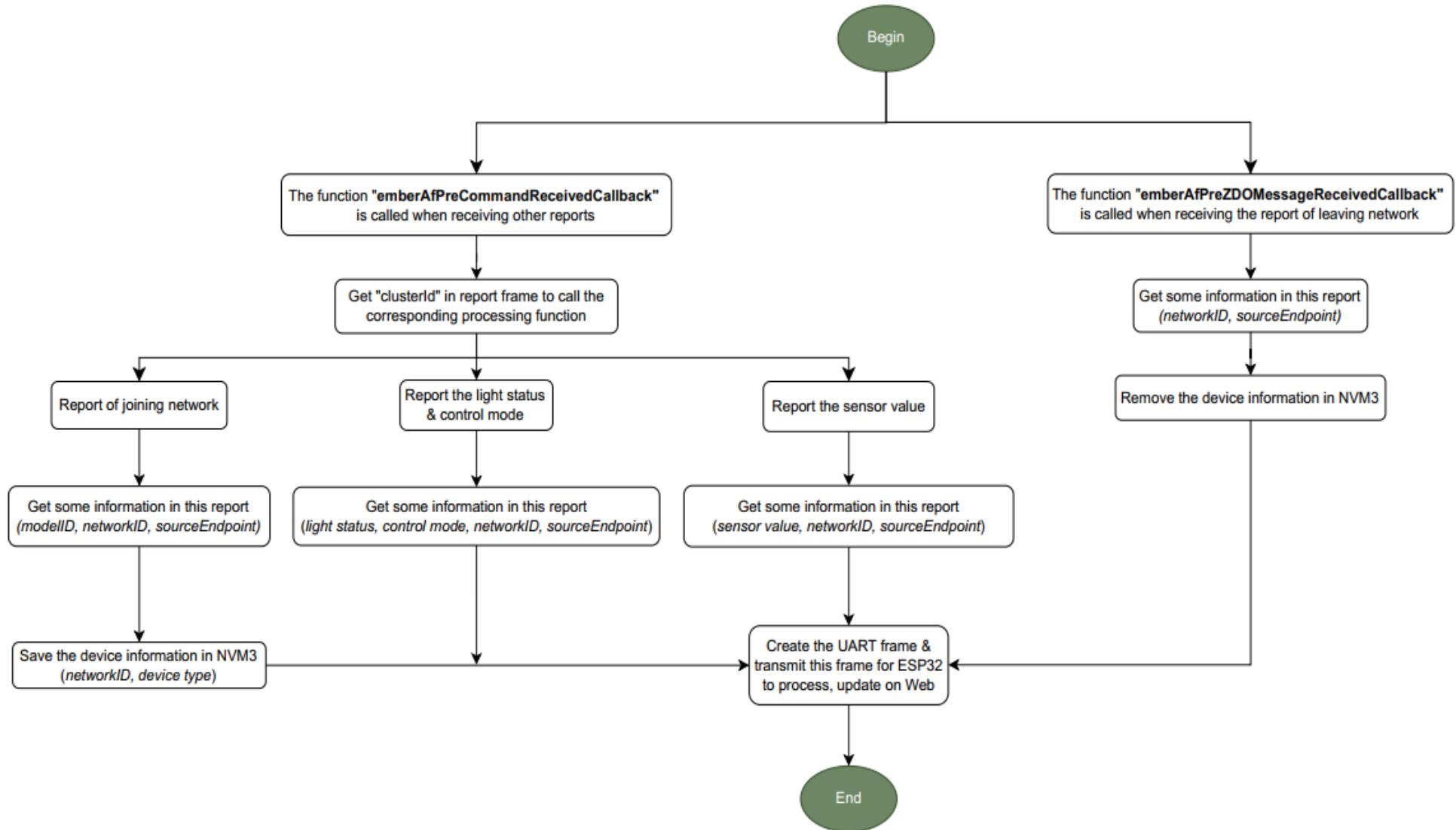


Hình 3.13: Sơ đồ thuật toán ZC xử lý bản tin nhận được từ Web

❖ Sơ đồ thuật toán lấy, xử lý dữ liệu hàng đợi

Xem tại [AlgorithmToProcessUARTdata](#)

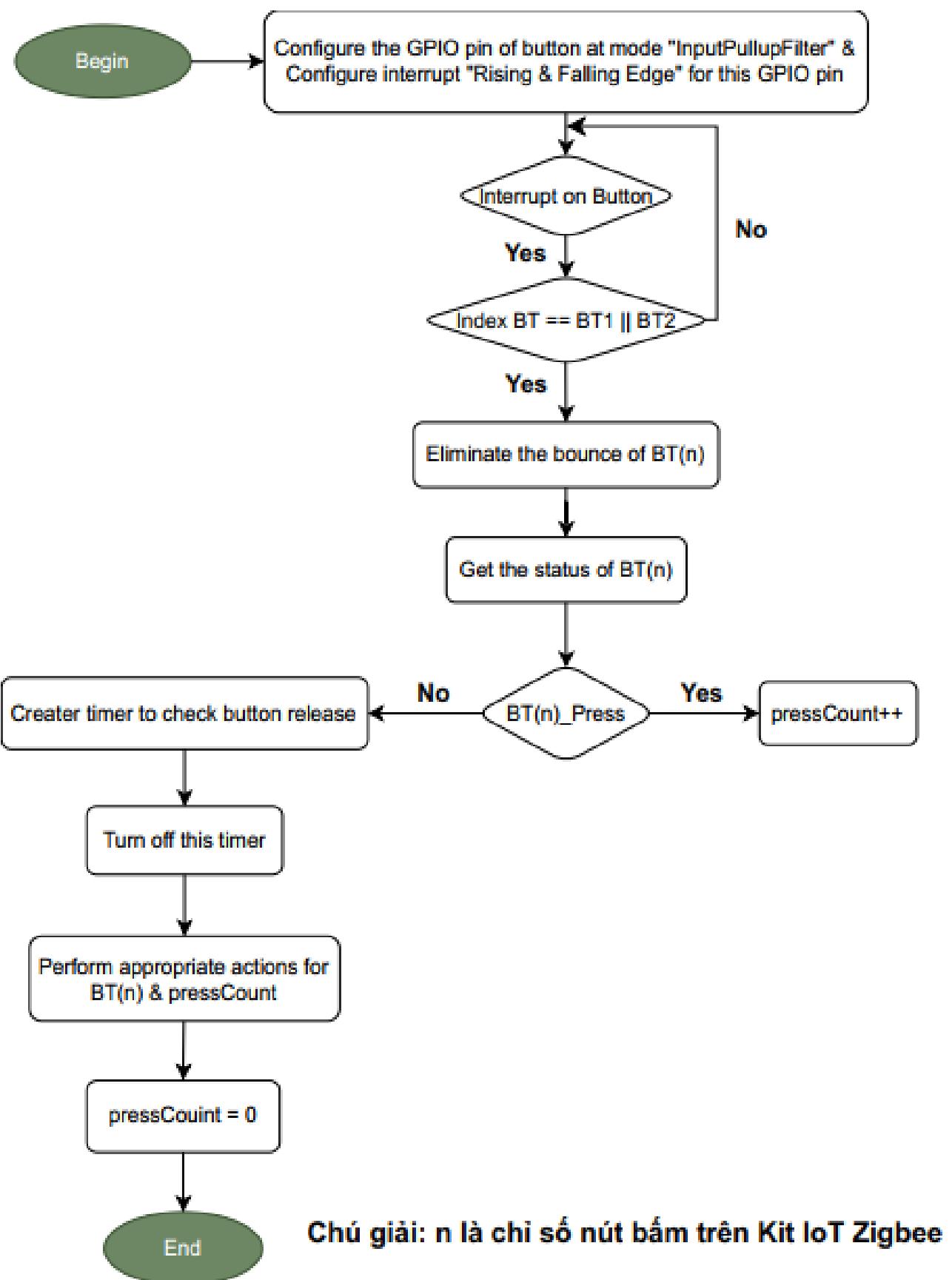
✚ ZC xử lý reports nhận được từ SWs và gửi cho Web



Hình 3.14: Sơ đồ thuật toán ZC xử lý report nhận được từ SWs và gửi cho Web

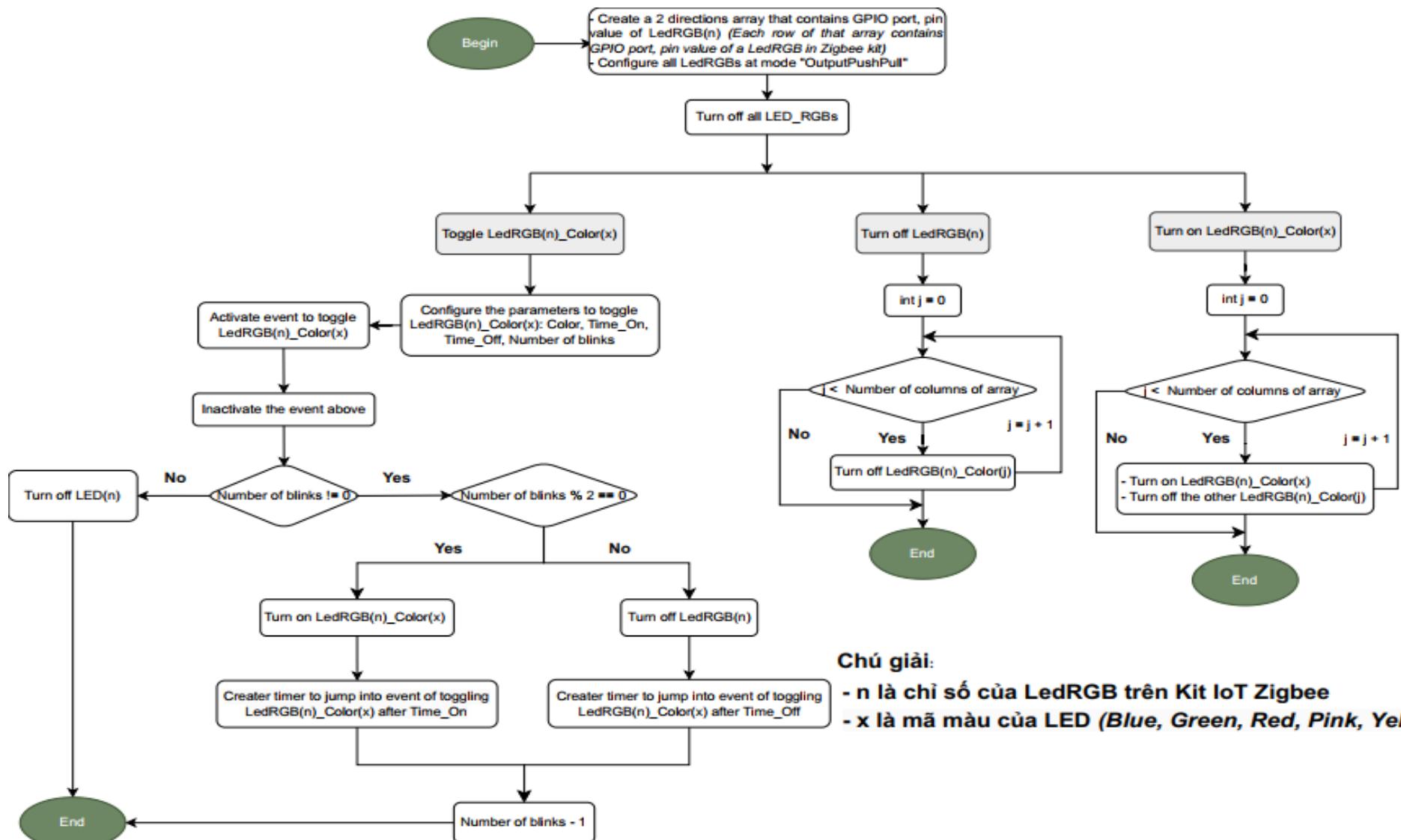
✚ Sơ đồ thuật toán khác

❖ Sơ đồ thuật toán xử lý nút bấm



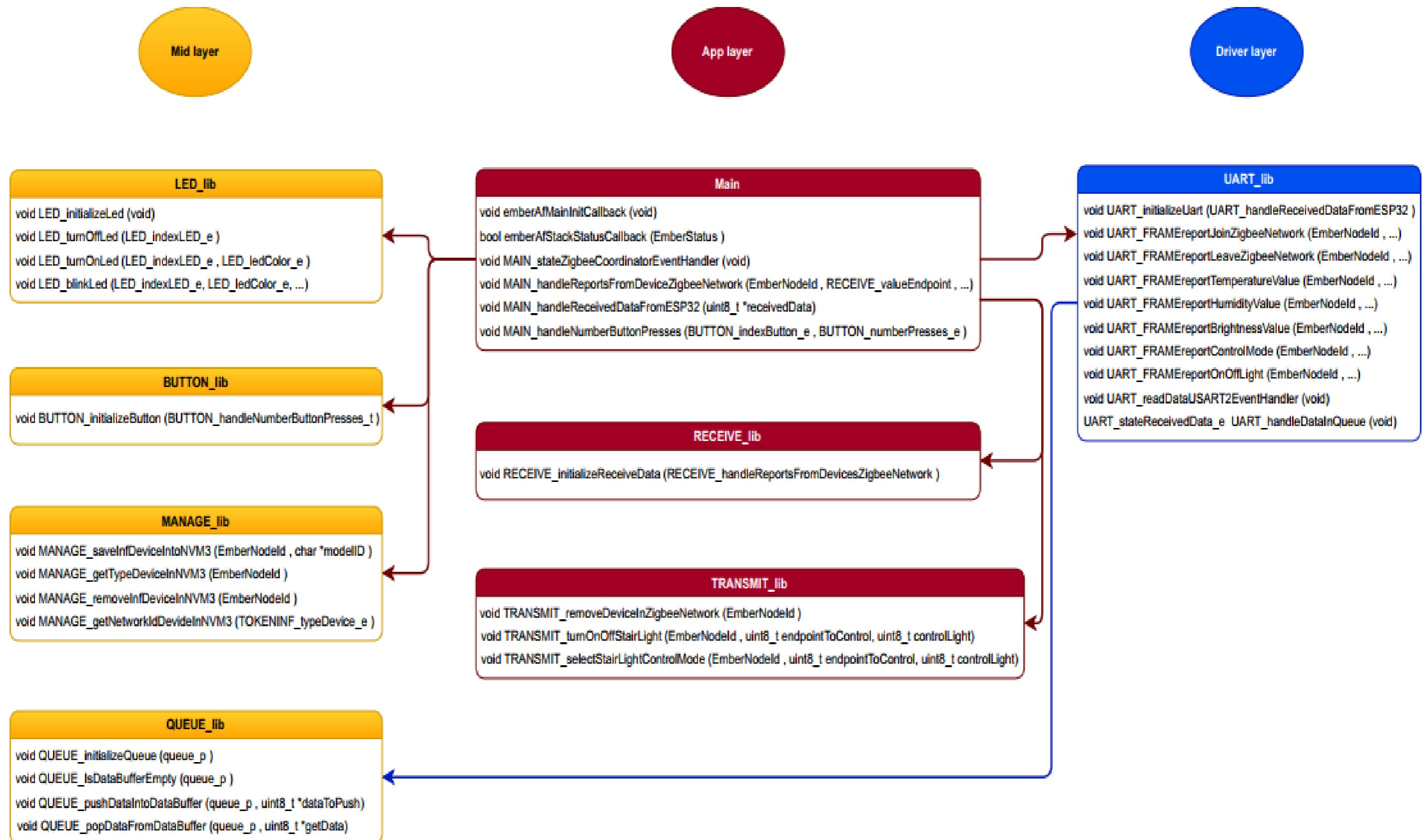
Hình 3.15: Sơ đồ thuật toán xử lý nút bấm

❖ Sơ đồ thuật toán điều khiển LED



Hình 3.16: Sơ đồ thuật toán điều khiển LED

4.2.4. Sơ đồ phân tầng của ZC



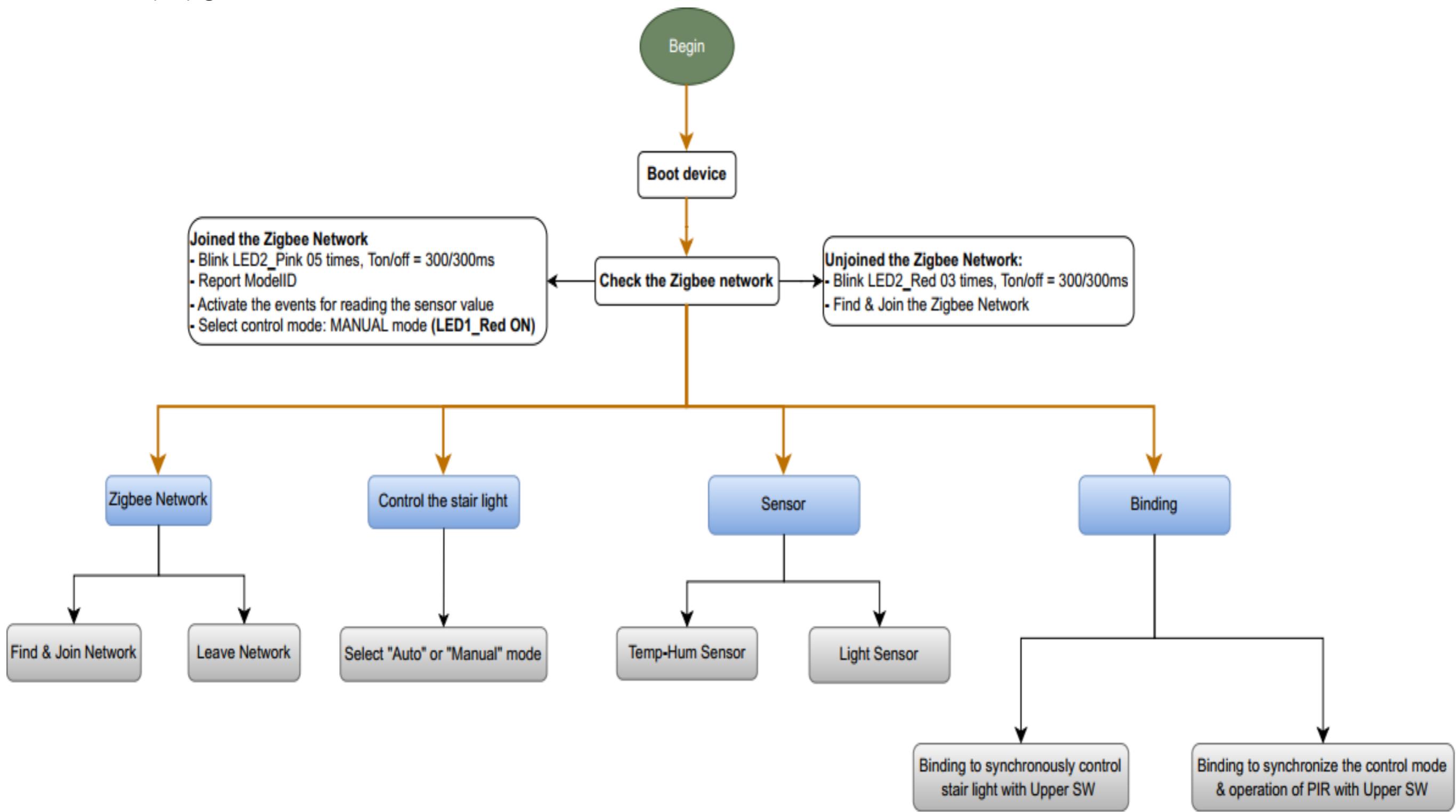
Hình 3.17: Sơ đồ phân tầng của ZC

4.3. Below Switch

4.3.1. Vai trò

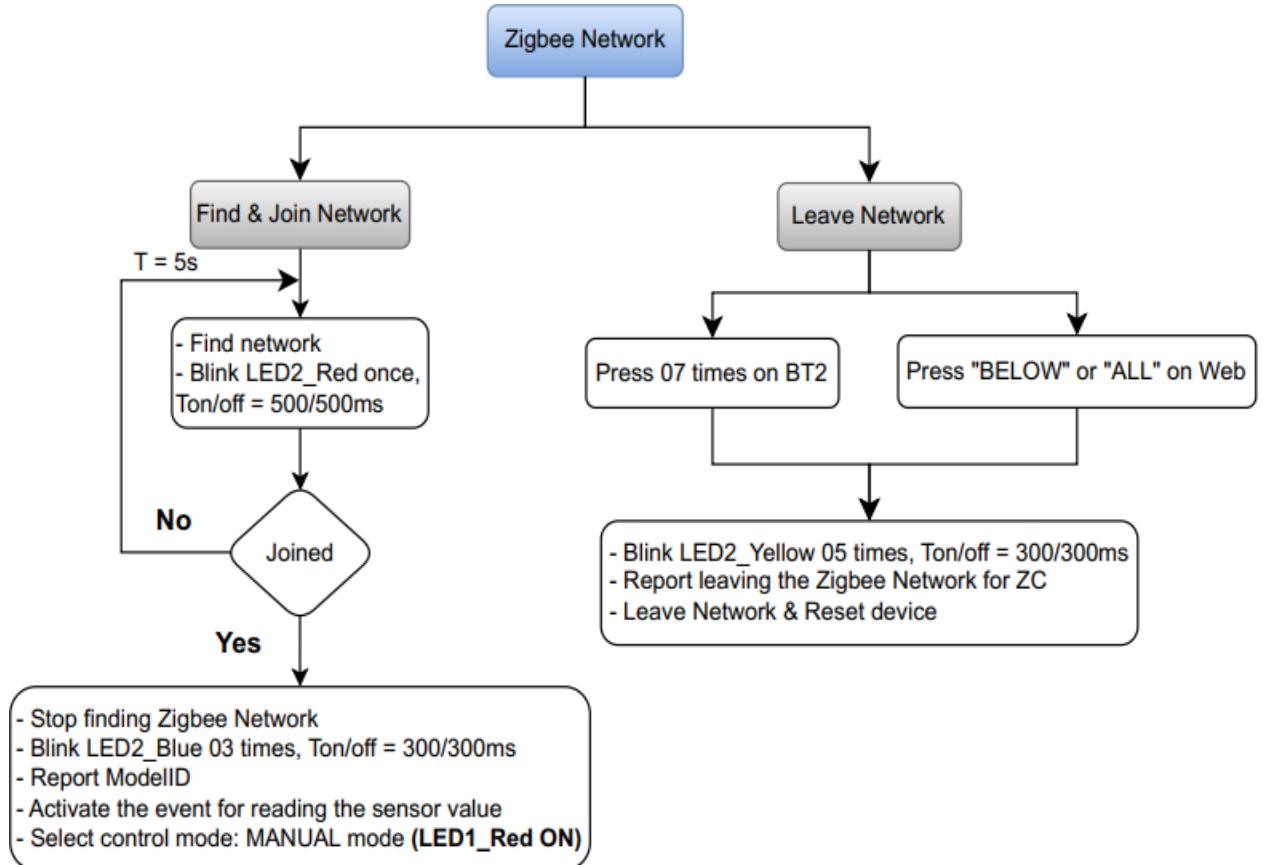
Below Switch (Zigbee Router) là thiết bị công tắc cầu thang được quản lý bởi thiết bị Zigbee Coordinator sử dụng để điều khiển đèn cầu thang với 3 chế độ điều khiển đồng bộ kết hợp với Upper Switch: Điều khiển bằng nút bấm cơ, Điều khiển sử dụng cảm biến PIR và Điều khiển trên WebLocal. Ngoài ra, để làm quen với WebLocal, trên thiết bị Below Switch, tôi sử dụng thêm tính năng thu thập thông số môi trường để report và hiển thị trên WebLocal.

4.3.2. Sơ đồ hoạt động của Below Switch



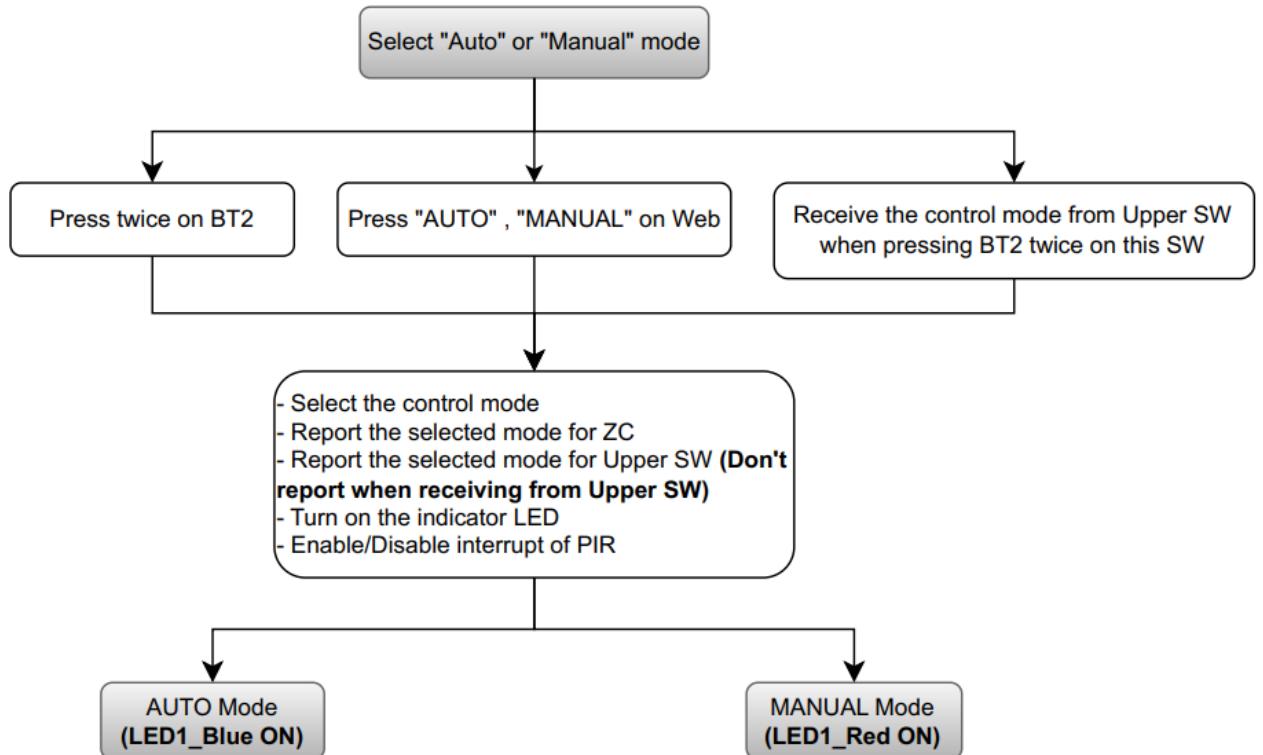
Hình 3.18: Sơ đồ hoạt động của Below Switch

❖ Sơ đồ hoạt động nhánh Zigbee Network của Below Switch



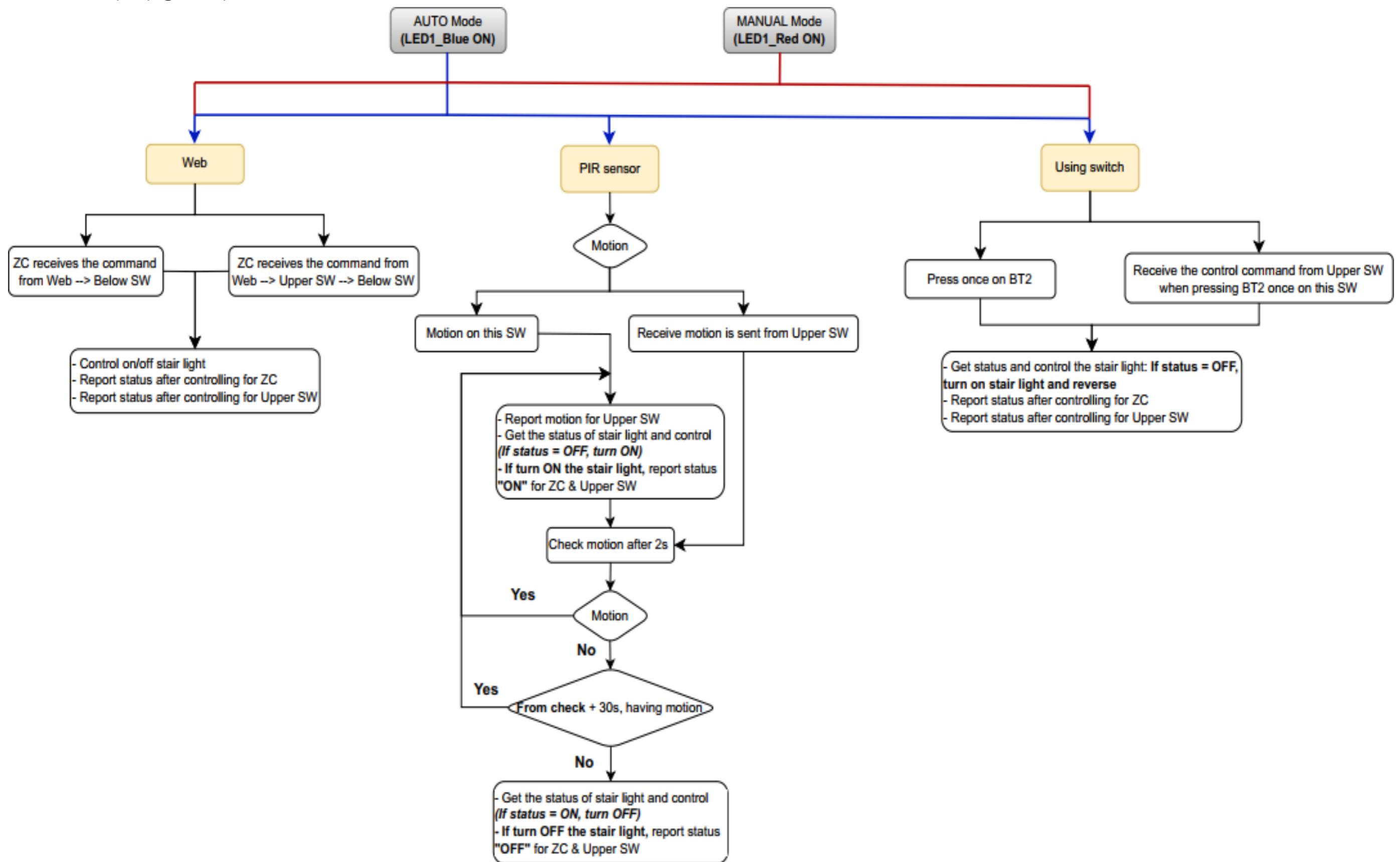
Hình 3.19: Sơ đồ hoạt động nhánh Zigbee Network của Below Switch

❖ Sơ đồ hoạt động lựa chọn chế độ “Auto”, “Manual” của Below SW



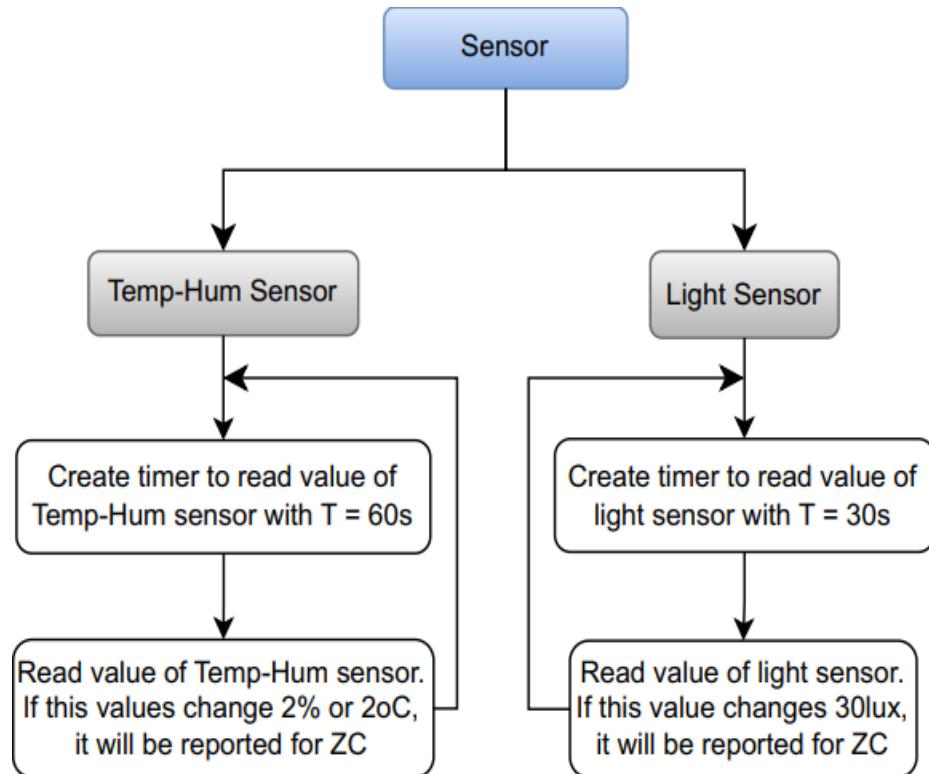
Hình 3.20: Sơ đồ hoạt động lựa chọn chế độ “Auto”, “Manual” của Below Switch

❖ Sơ đồ hoạt động chế độ “Auto”, “Manual” của Below SW



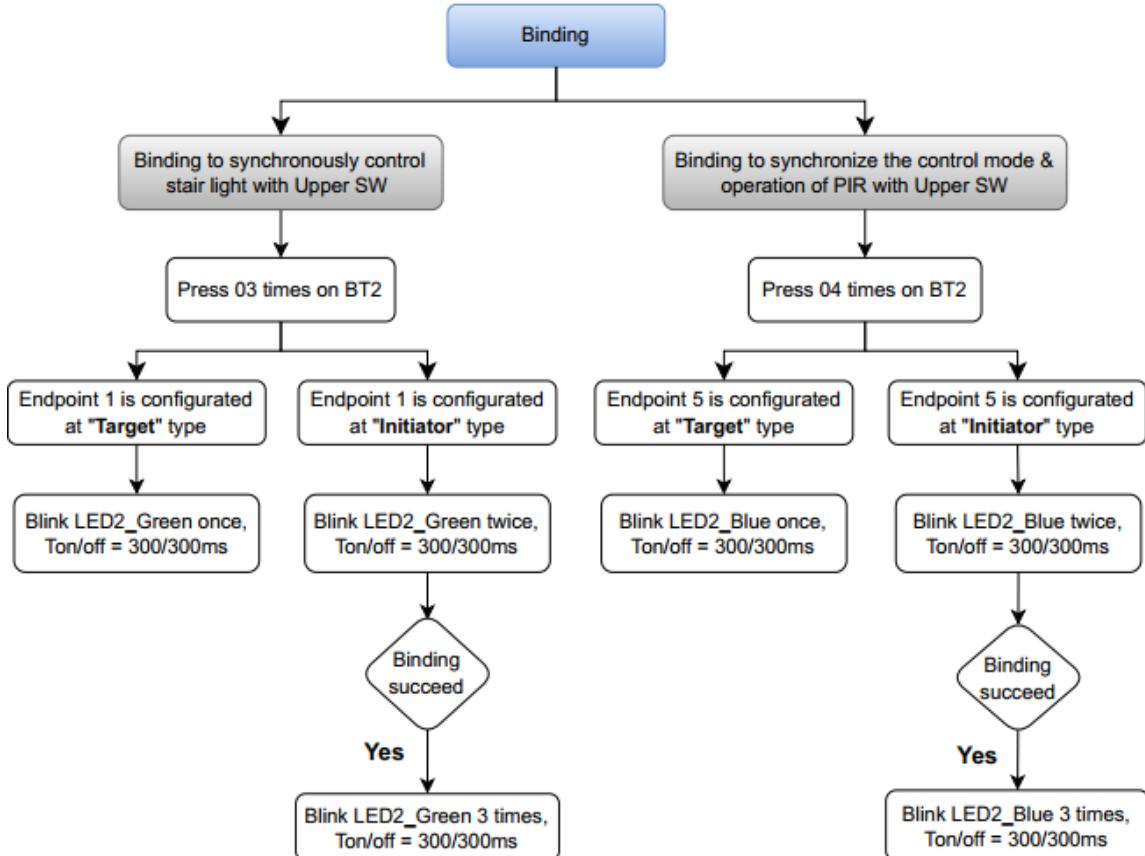
Hình 3.21: Sơ đồ hoạt động chế độ “Auto”, “Manual” của Below Switch

❖ Sơ đồ hoạt động nhánh Sensor của Below Switch



Hình 3.22: Sơ đồ hoạt động nhánh Sensor của Below Switch

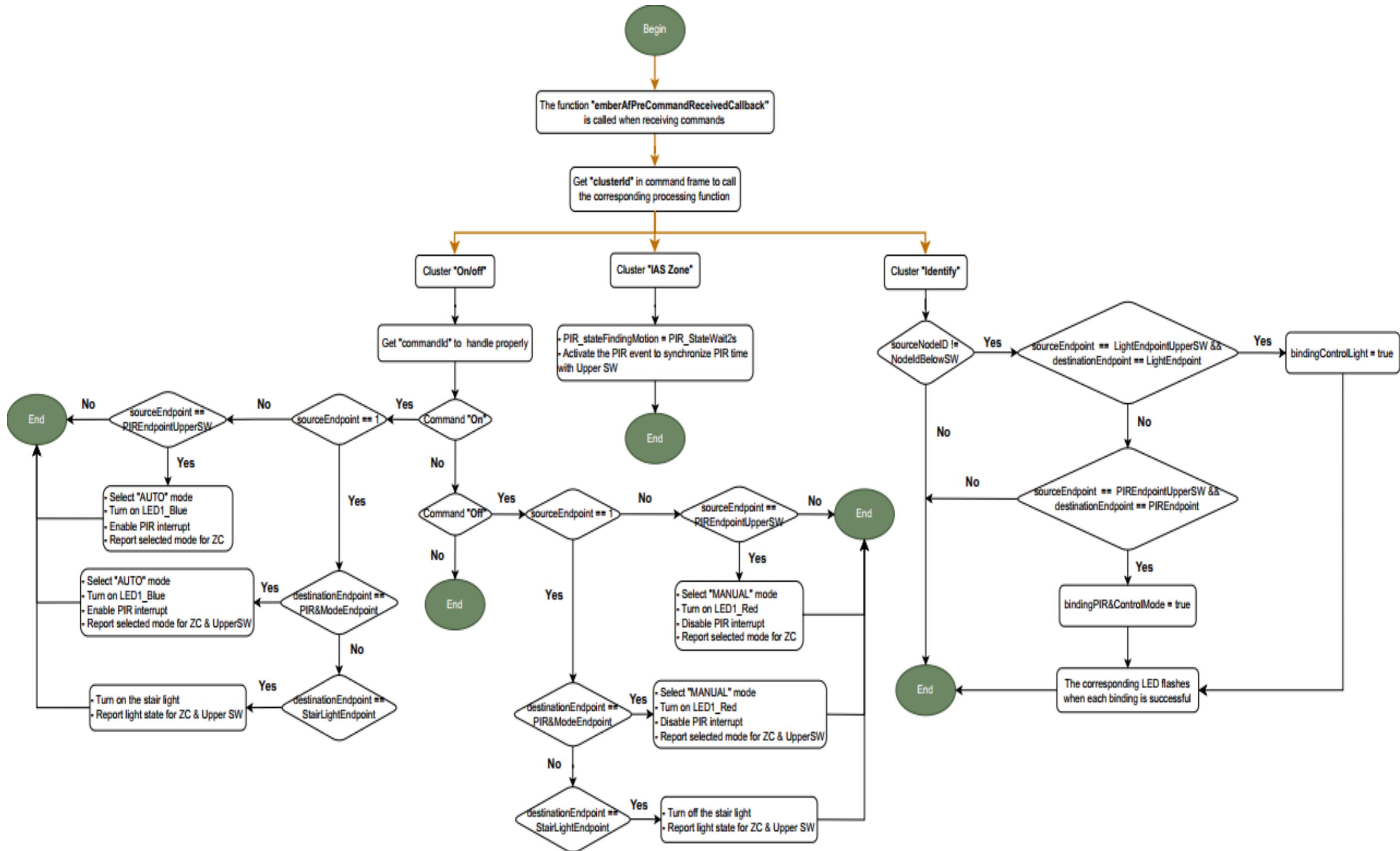
❖ Sơ đồ hoạt động nhánh Binding của Below Switch



Hình 3.23: Sơ đồ hoạt động nhánh Binding của Below Switch

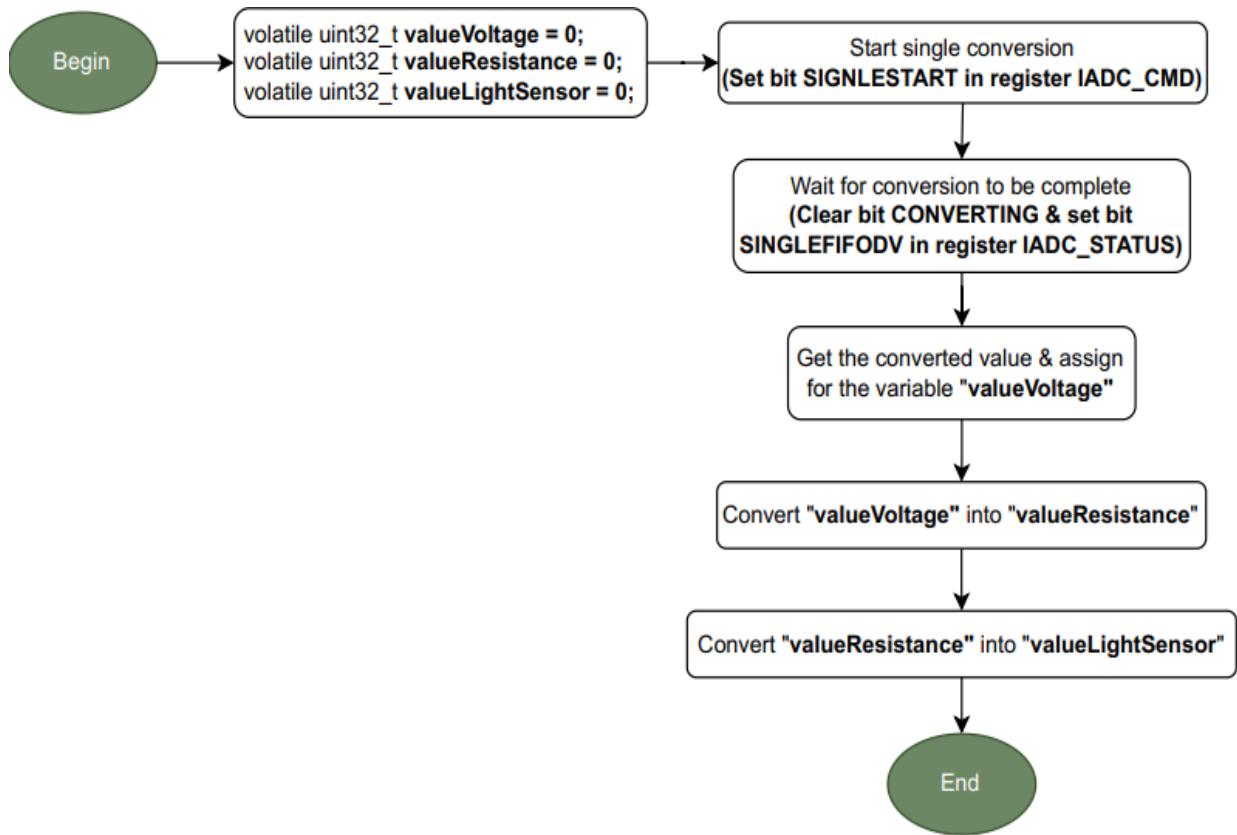
4.3.3. Sơ đồ thuật toán của Below Switch

❖ Sơ đồ thuật toán Below SW xử lý bản tin nhận được từ ZC và Upper SW



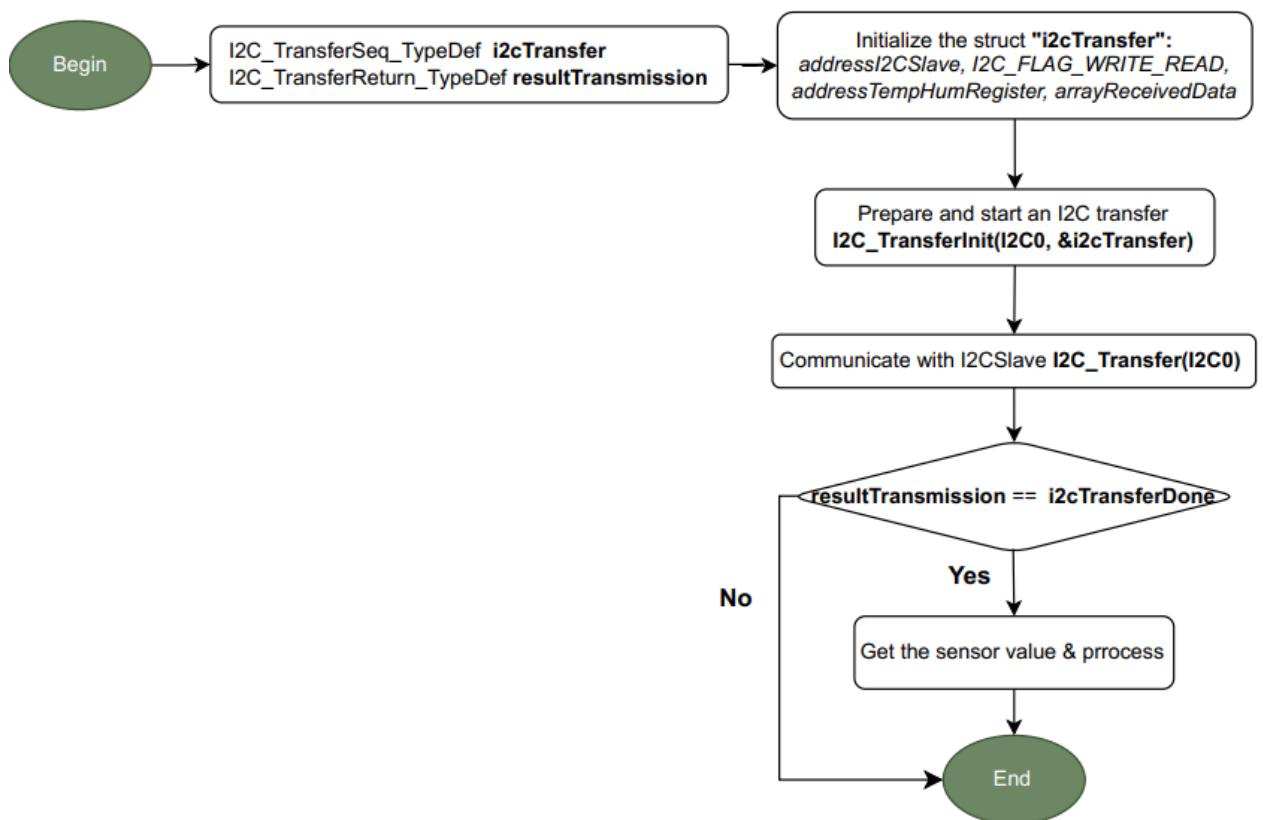
Hình 3.24: SĐTT Below SW xử lý bản tin nhận được từ ZC và Upper SW

❖ Sơ đồ thuật toán Hàm đọc dữ liệu cảm biến ánh sáng



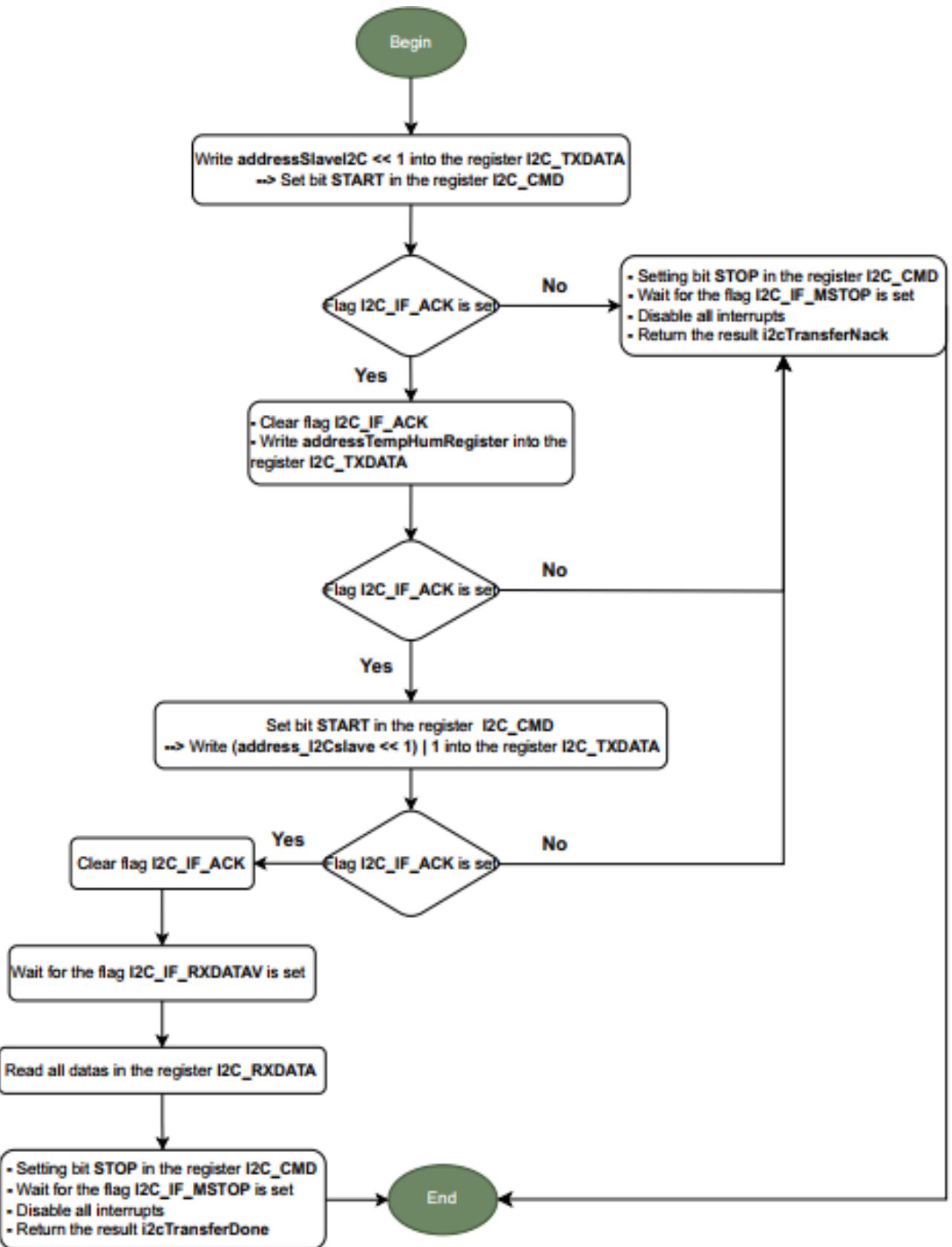
Hình 3.25: Sơ đồ thuật toán Hàm đọc dữ liệu cảm biến ánh sáng

❖ Sơ đồ thuật toán Hàm đọc dữ liệu cảm biến nhiệt độ, độ ẩm



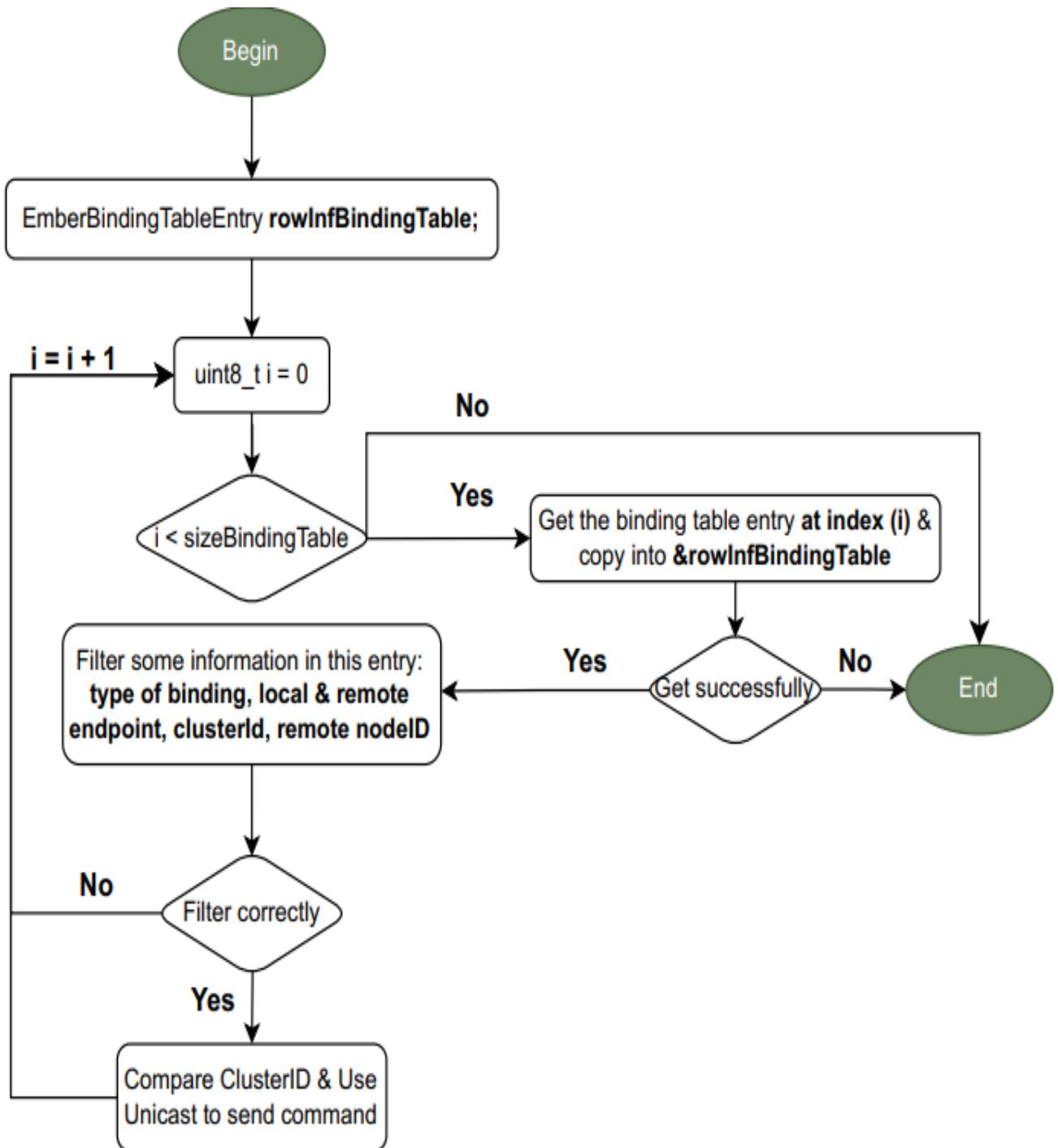
Hình 3.26: Sơ đồ thuật toán Hàm đọc dữ liệu cảm biến nhiệt độ, độ ẩm

❖ Sơ đồ thuật toán Hàm “I2C_Transfer” sử dụng để I2Cmaster – Kit Zigbee write & read với I2Cslave - Cảm biến nhiệt độ, độ ẩm



Hình 3.27: Sơ đồ thuật toán Hàm “I2C_Transfer”

❖ Sơ đồ thuật toán sử dụng Binding truyền dữ liệu



Hình 3.28: Sơ đồ thuật toán sử dụng Binding truyền dữ liệu

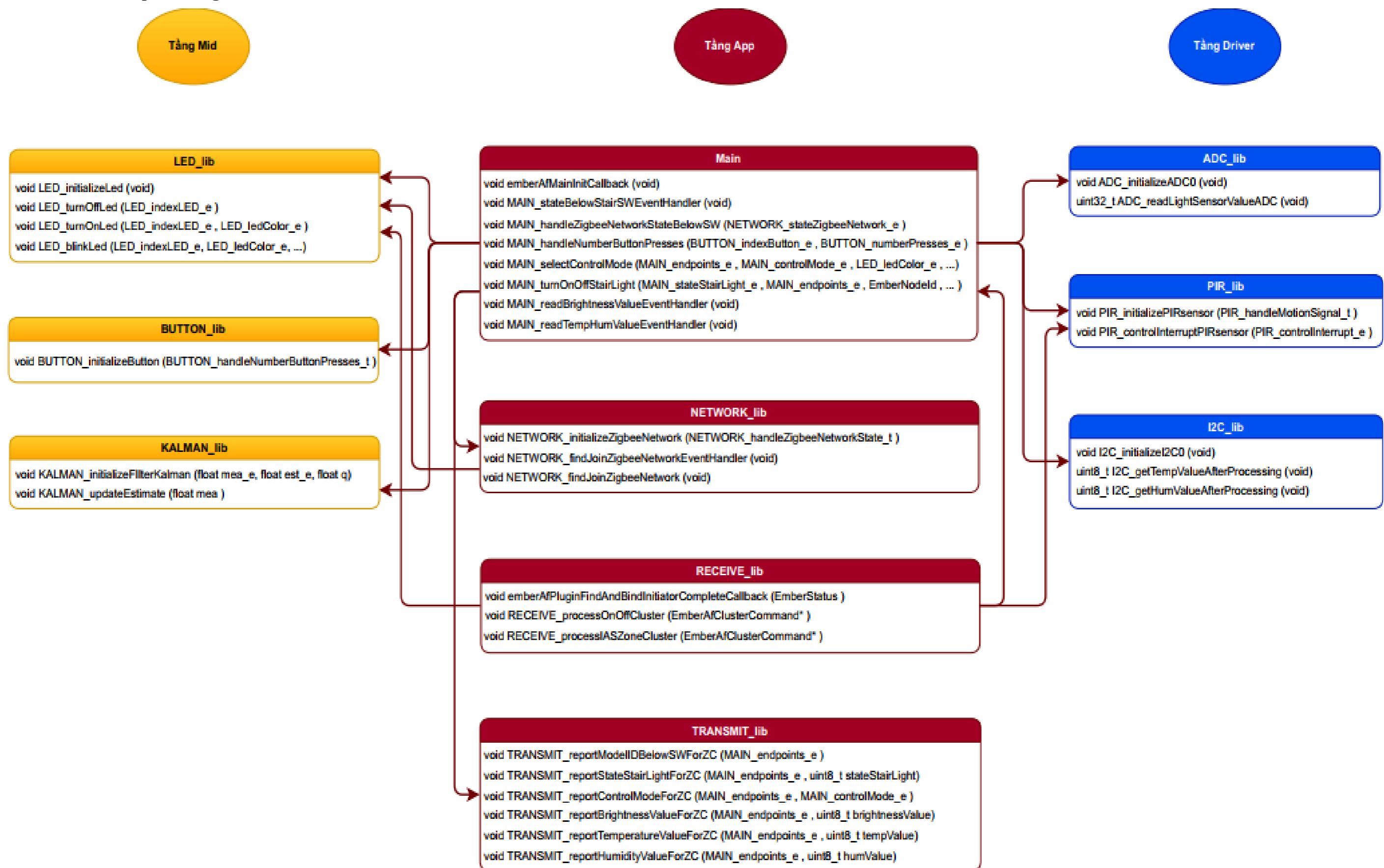
❖ Sơ đồ thuật toán xử lý nút bấm

Xem tại [AlgorithmToProcessButton](#)

❖ Sơ đồ thuật toán điều khiển LED

Xem tại [AlgorithmToControlLED](#)

4.3.4. Sơ đồ phân tầng của Below Switch



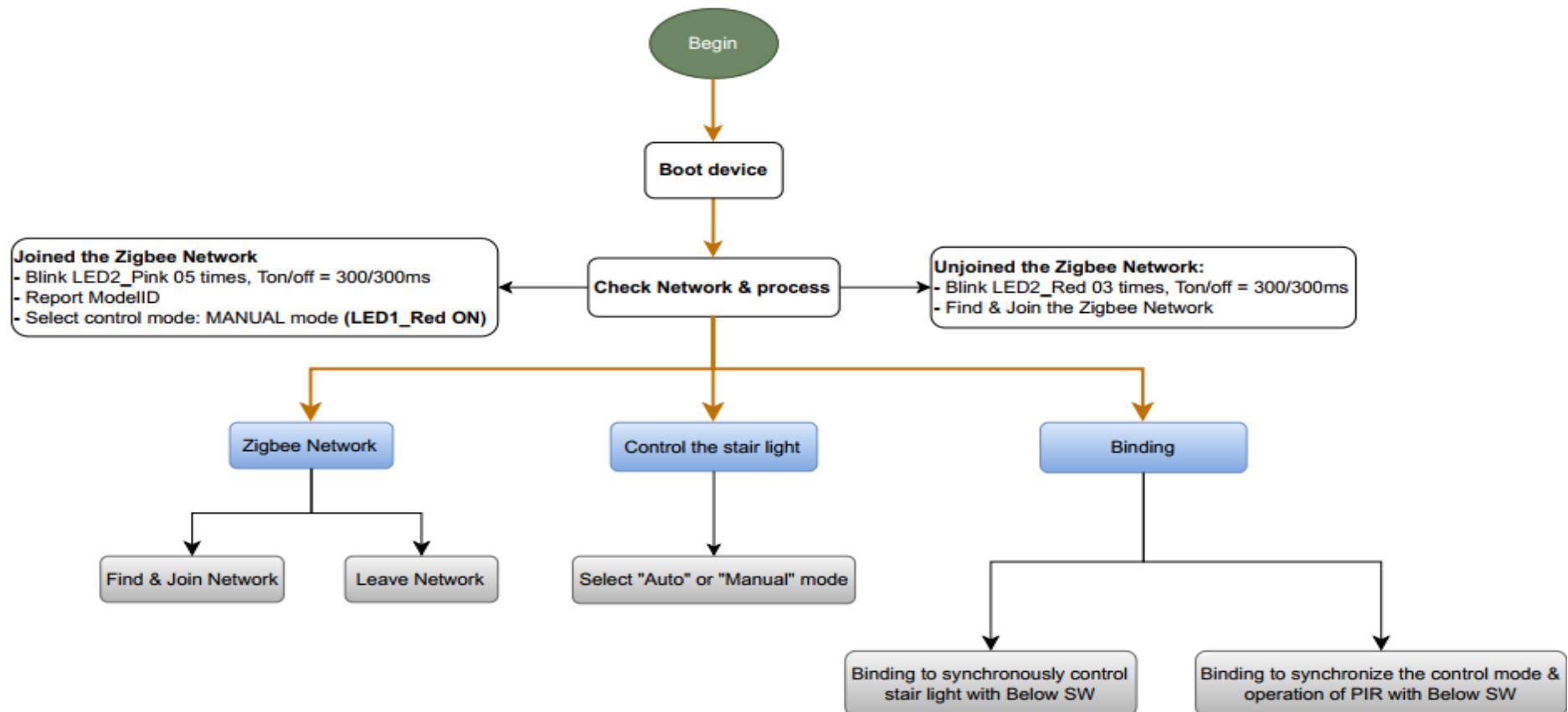
Hình 3.29: Sơ đồ phân tầng của Below Switch

4.4. Upper Switch

4.4.1. Vai trò

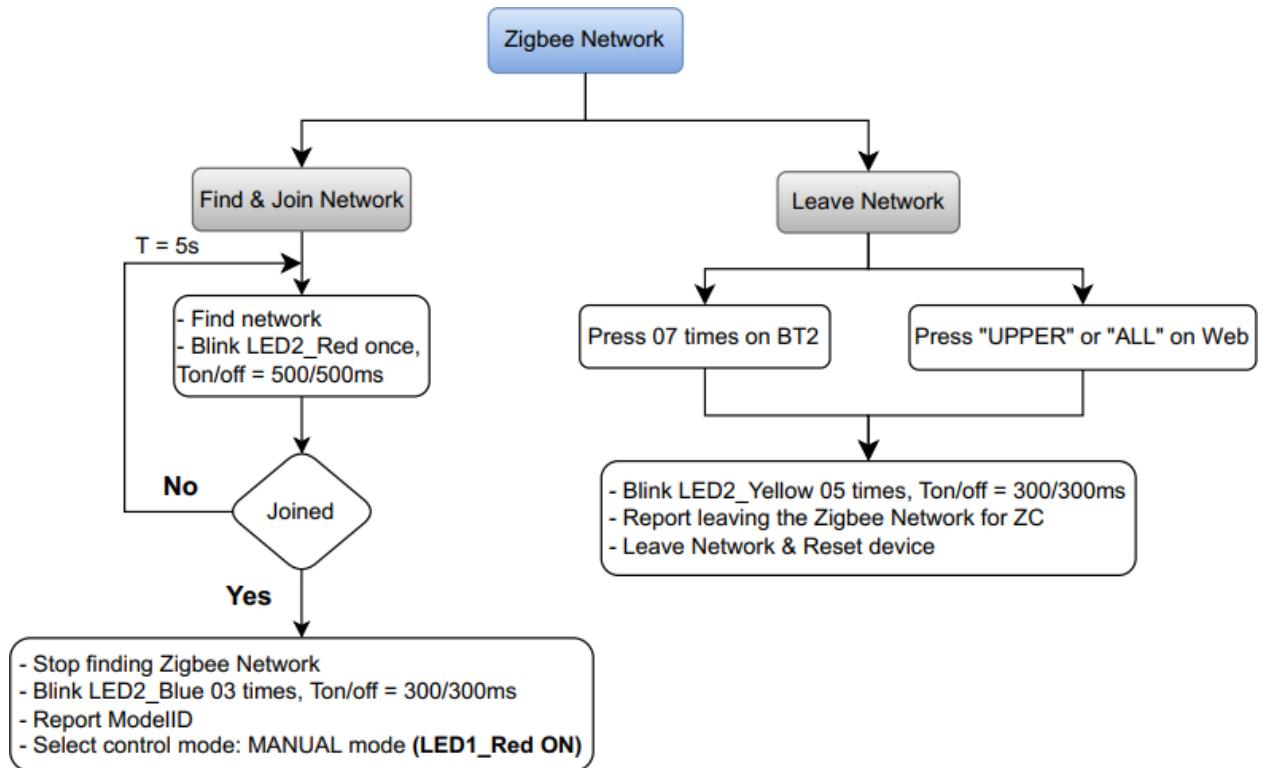
Upper Switch (*Zigbee Router*) là thiết bị công tắc cầu thang được quản lý bởi thiết bị Zigbee Coordinator sử dụng để điều khiển đèn cầu thang với 3 chế độ điều khiển đồng bộ kết hợp với Below Switch: Điều khiển bằng nút bấm cơ, Điều khiển sử dụng cảm biến PIR và Điều khiển trên WebLocal.

4.4.2. Sơ đồ hoạt động của Upper Switch



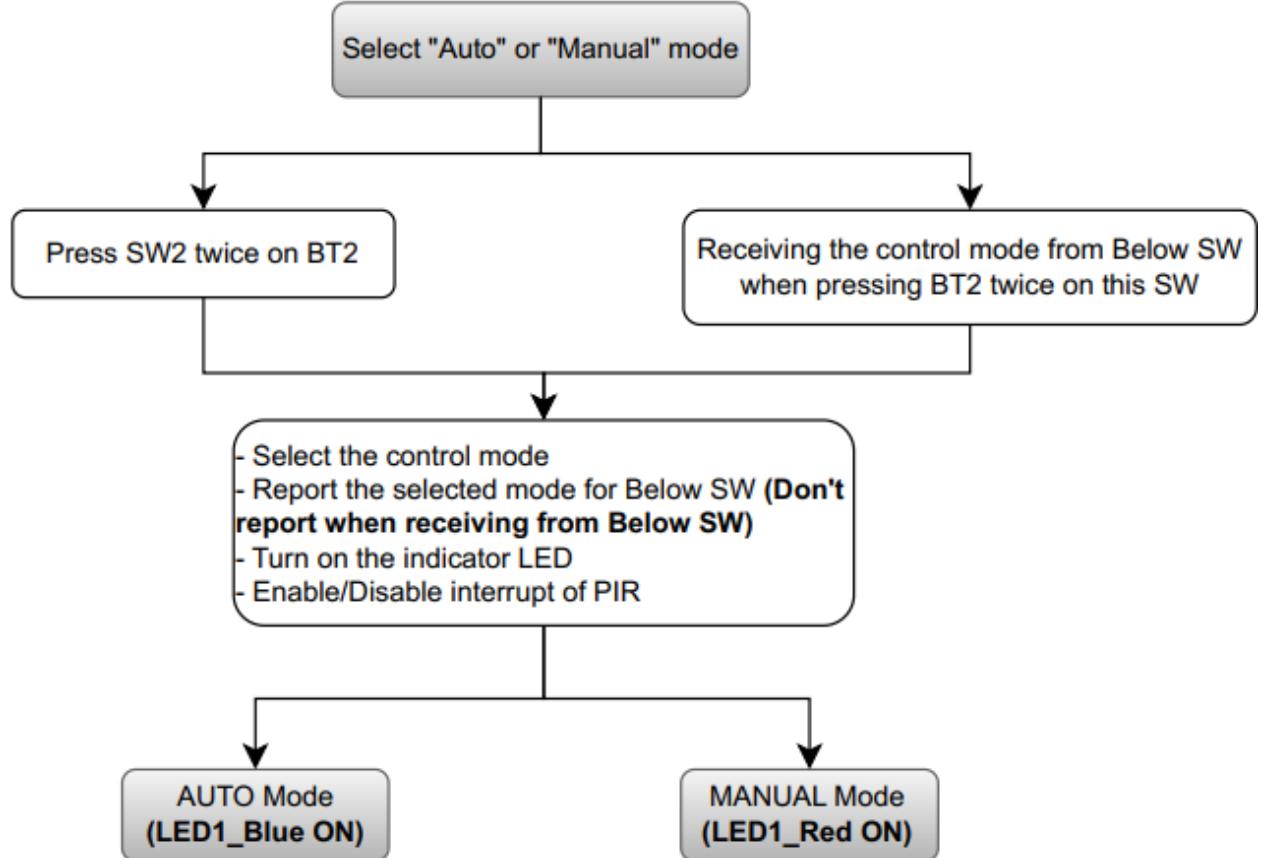
Hình 3.30: Sơ đồ hoạt động của Upper Switch

❖ Sơ đồ hoạt động nhánh Zigbee Network của Upper Switch



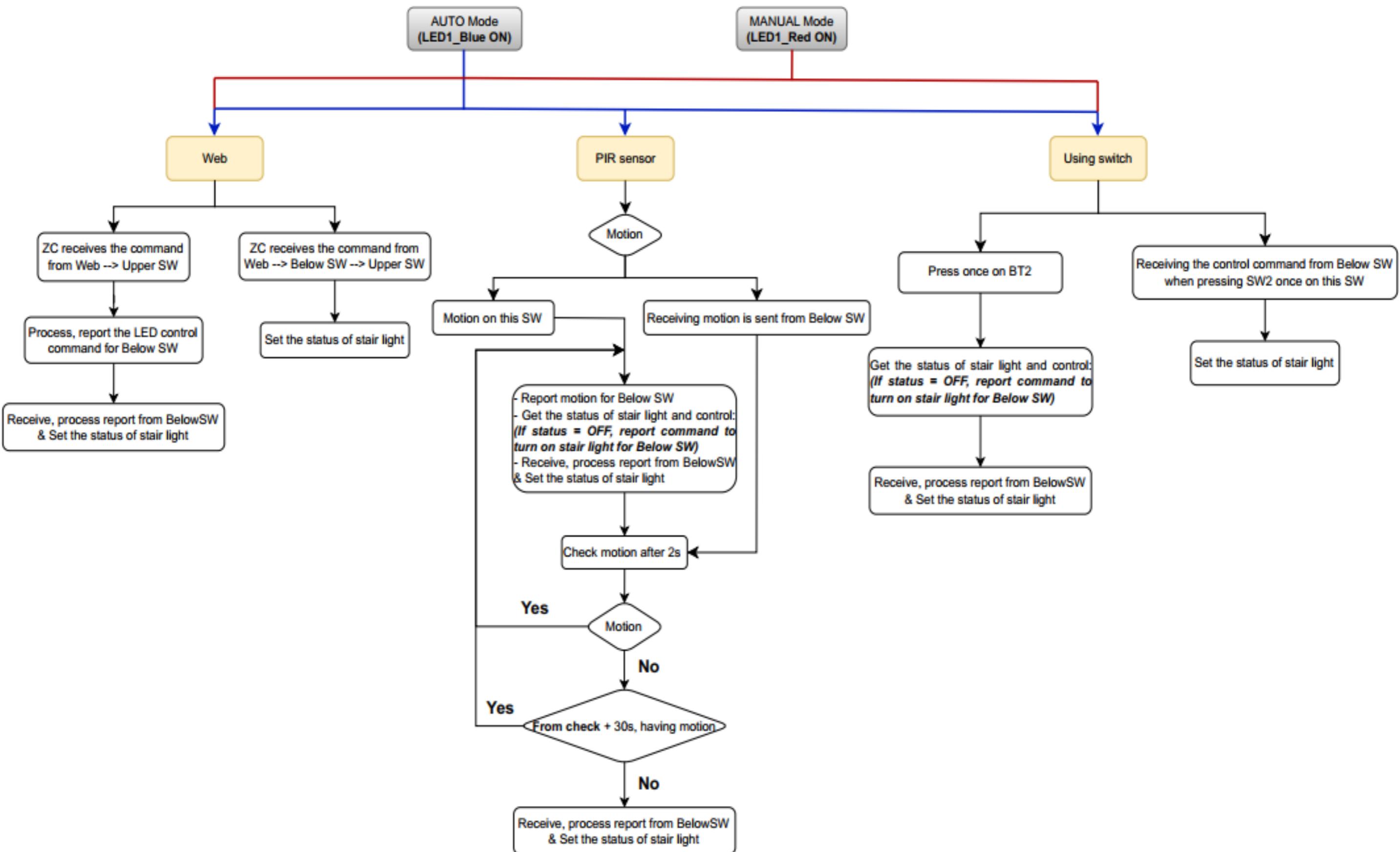
Hình 3.31: Sơ đồ hoạt động nhánh Zigbee Network của Upper Switch

❖ Sơ đồ hoạt động lựa chọn chế độ “Auto”, “Manual” của Upper SW



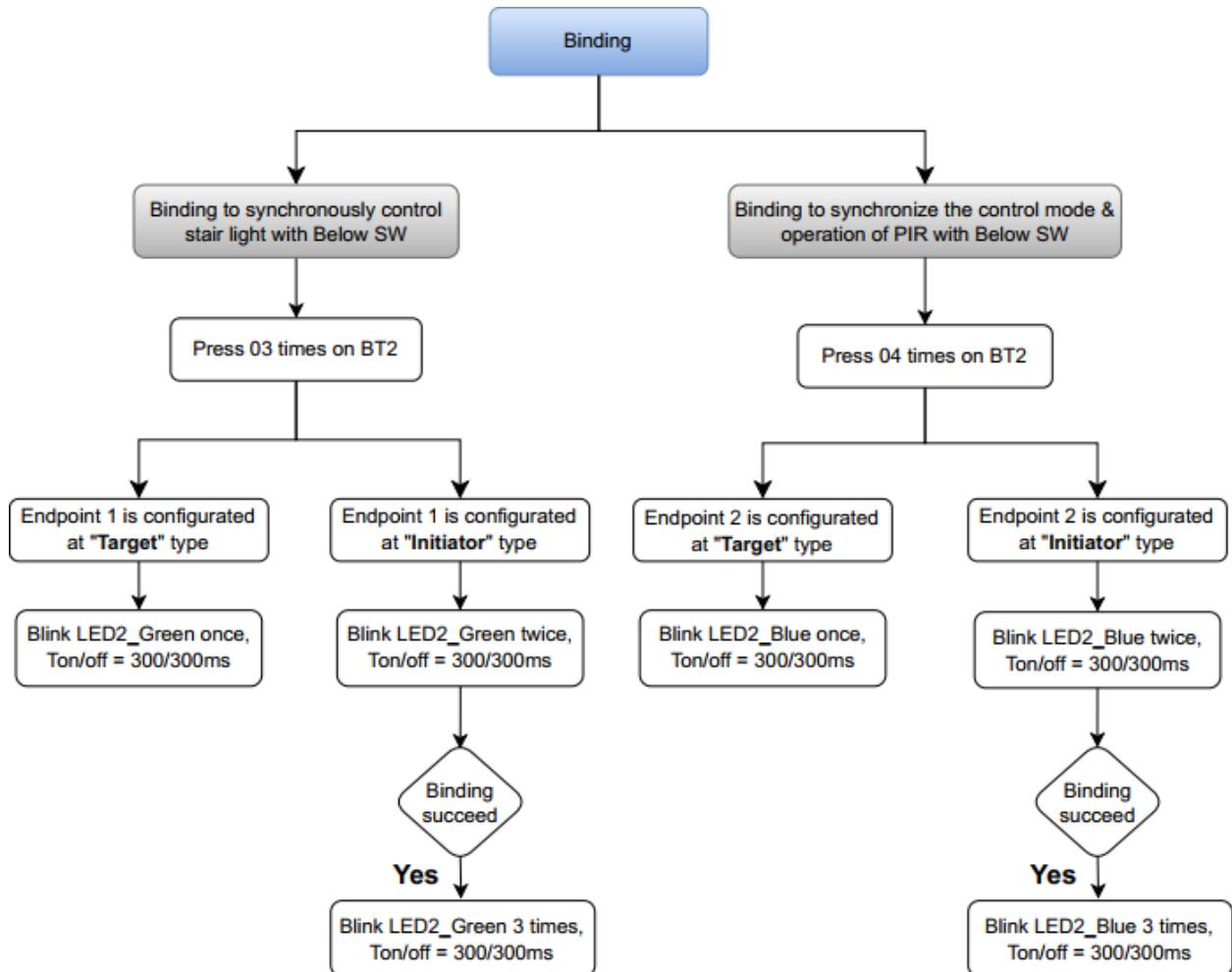
Hình 3.32: Sơ đồ hoạt động lựa chọn chế độ “Auto”, “Manual” của Upper Switch

❖ Sơ đồ hoạt động chế độ “Auto”, “Manual” của Upper SW



Hình 3.33: Sơ đồ hoạt động chế độ “Auto”, “Manual” của Upper Switch

❖ Sơ đồ hoạt động nhánh Binding của Upper Switch



Hình 3.34: Sơ đồ hoạt động nhánh Binding của Below Switch

4.4.3. Sơ đồ thuật toán của Upper Switch

❖ Sơ đồ thuật toán xử lý nút bấm

Xem tại [AlgorithmToProcessButton](#)

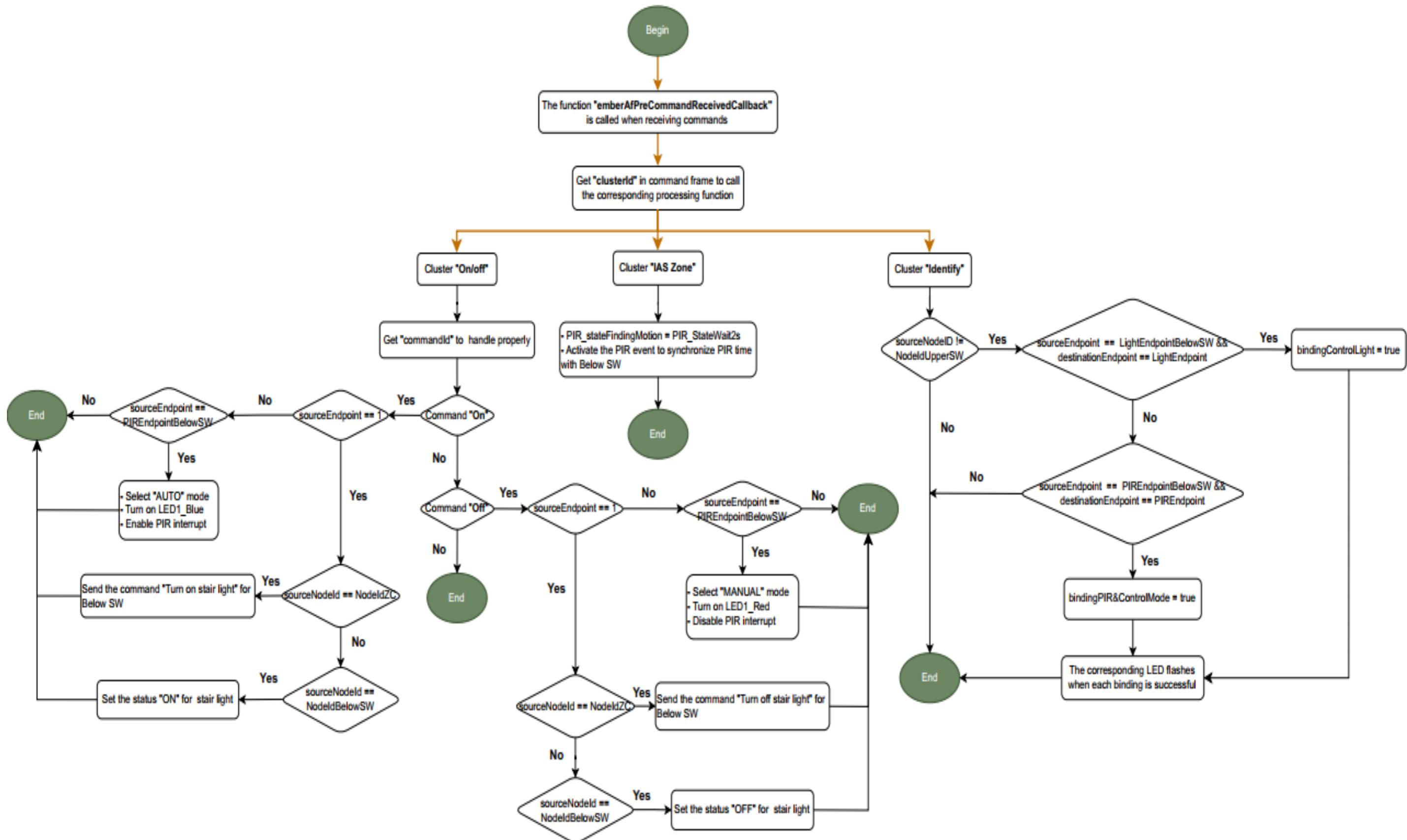
❖ Sơ đồ thuật toán điều khiển LED

Xem tại [AlgorithmToControlLED](#)

❖ Sơ đồ thuật toán sử dụng Binding truyền dữ liệu

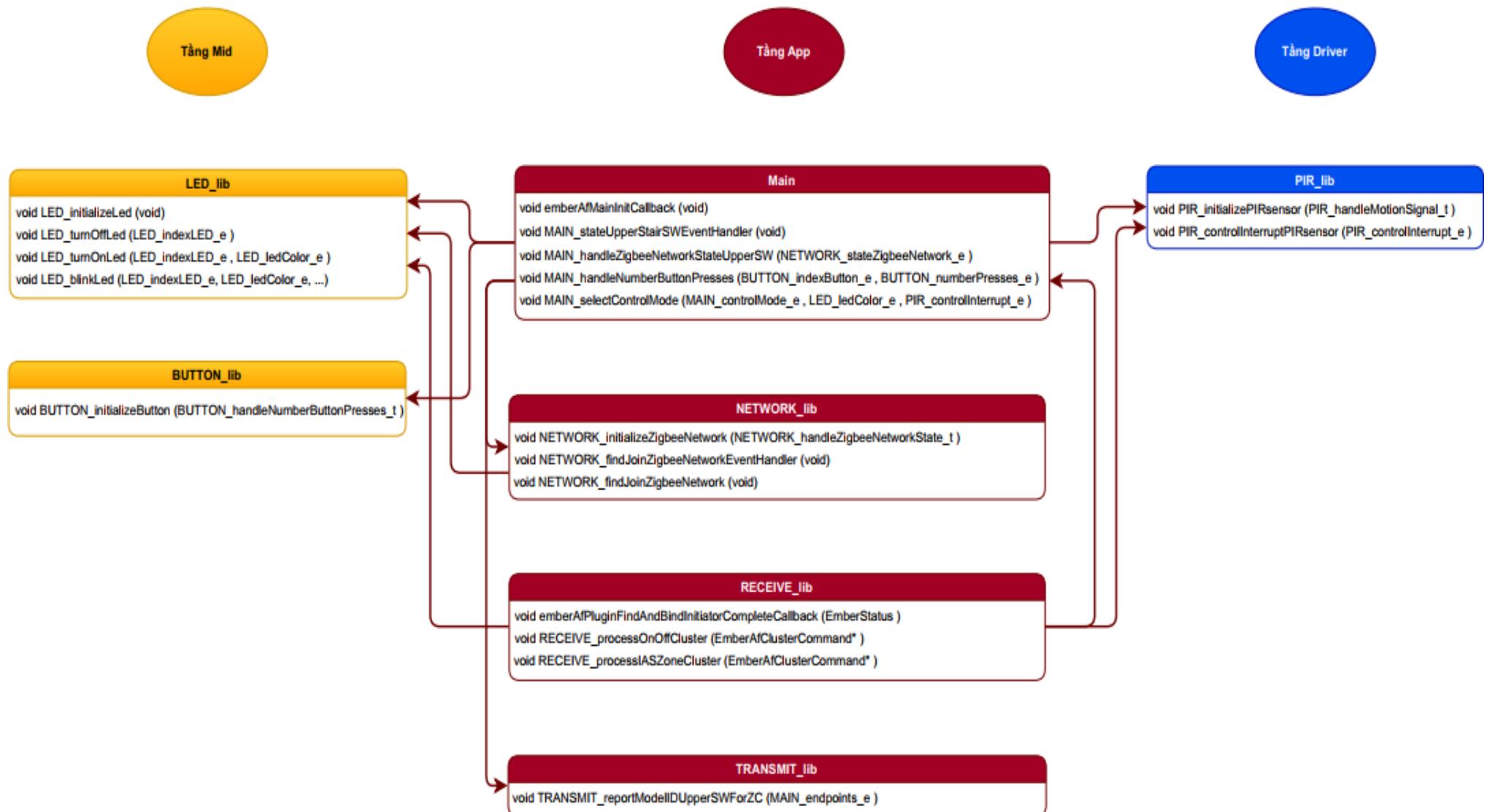
Xem tại [AlgorithmUsingBindingToTransmitData](#)

❖ Sơ đồ thuật toán Upper SW xử lý bản tin nhận được từ ZC và Below SW



Hình 3.35: SDTT Upper SW xử lý bản tin nhận được từ ZC và Below SW

4.4.4. Sơ đồ phân tầng của Upper Switch



Hình 3.36: Sơ đồ phân tầng của Upper Switch

4.5. Giao diện Web điều khiển & ESP32

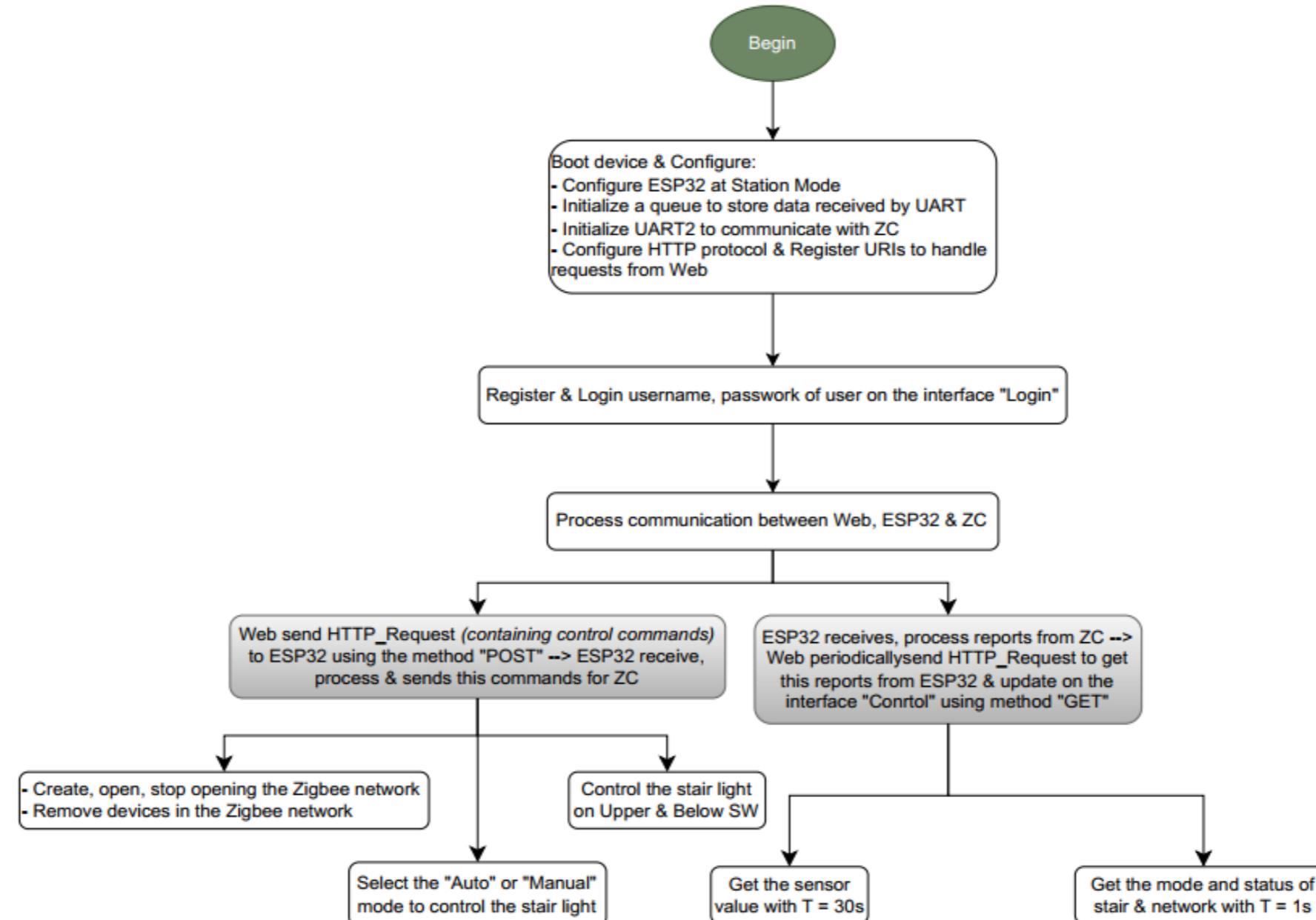
4.5.1. Vai trò

Giao diện Web điều khiển được sử dụng để giao tiếp với thiết bị công tắc cầu thang thông qua module ESP32. Việc truy cập vào “Giao diện Web điều khiển” được thực hiện bằng địa chỉ IP sau khi ESP32 kết nối Wi-Fi thành công. Tính năng điều khiển trên giao diện Web bao gồm:

- ❖ Tính năng mạng Zigbee: Tạo, mở, đóng và xóa thiết bị khỏi mạng.
- ❖ Tính năng điều khiển đèn cầu thang.
- ❖ Tính năng cập nhật, hiển thị trạng thái mạng của thiết bị & trạng thái đèn cầu thang.
- ❖ Tính năng cập nhật, hiển thị thông số môi trường: Nhiệt độ, độ ẩm, ánh sáng.

ESP32 (**Server**) là nơi lập trình các tính năng cho “Giao diện Web điều khiển”, lưu trữ và xử lý các yêu cầu của giao diện đó. ESP32 sẽ tiếp nhận các request từ “Giao diện Web điều khiển” và gửi response cho nó thông qua giao thức HTTP. Ngoài ra, ESP32 còn đóng vai trò trung gian trong giao tiếp giữa “Giao diện Web điều khiển” và thiết bị công tắc cầu thang.

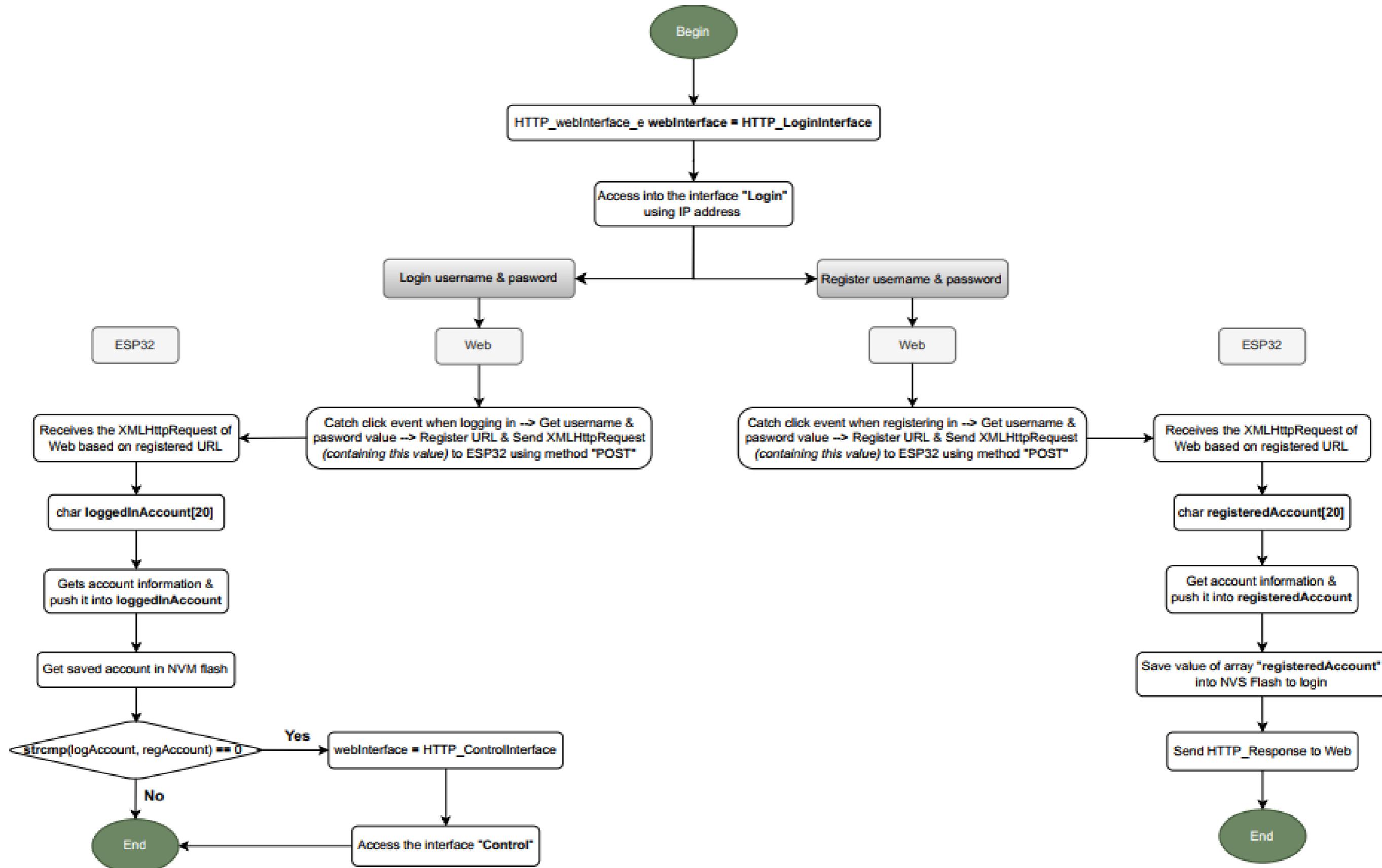
4.5.2. Sơ đồ hoạt động của giao diện Web điều khiển & ESP32



Hình 3.37: Sơ đồ hoạt động của giao diện Web điều khiển & ESP32

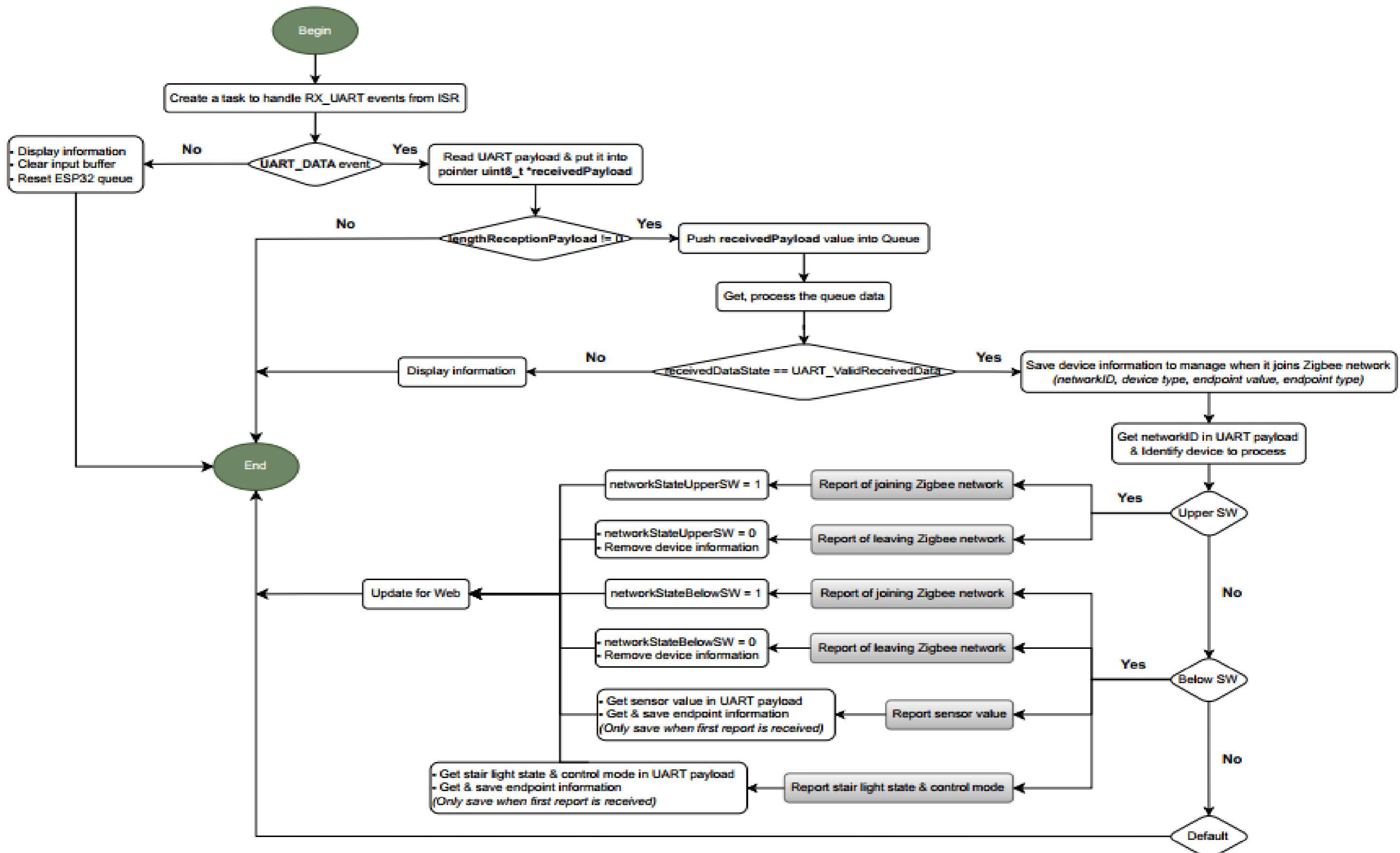
4.5.3. Sơ đồ thuật toán của giao diện Web điều khiển & ESP32

❖ Sơ đồ thuật toán đăng ký, đăng nhập tài khoản



Hình 3.38: Sơ đồ thuật toán đăng ký, đăng nhập tài khoản

❖ Sơ đồ thuật toán ESP32 xử lý bản tin nhận được từ ZC

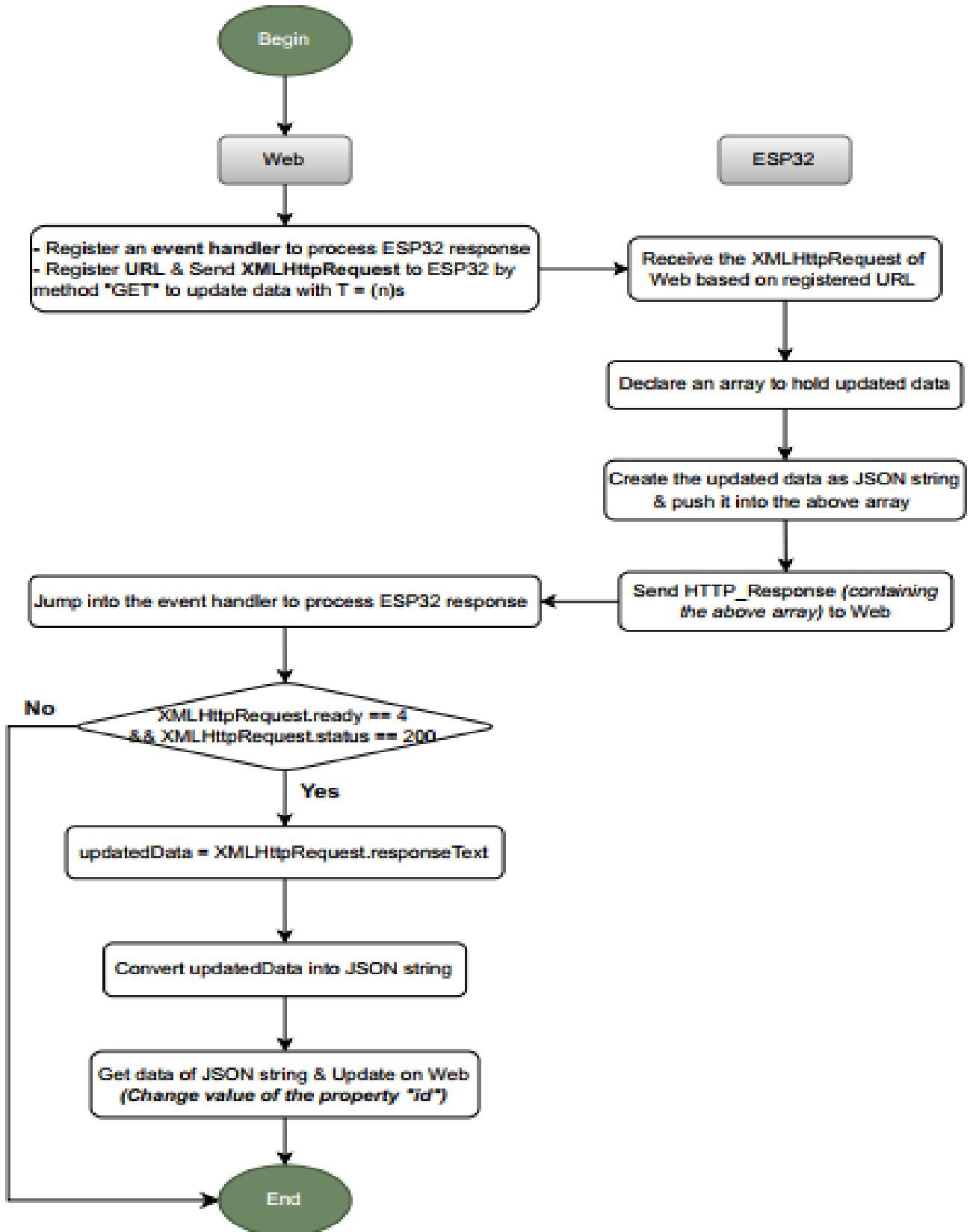


Hình 3.39: Sơ đồ thuật toán ESP32 xử lý bản tin nhận được từ ZC

❖ Sơ đồ thuật toán lấy, xử lý dữ liệu hàng đợi

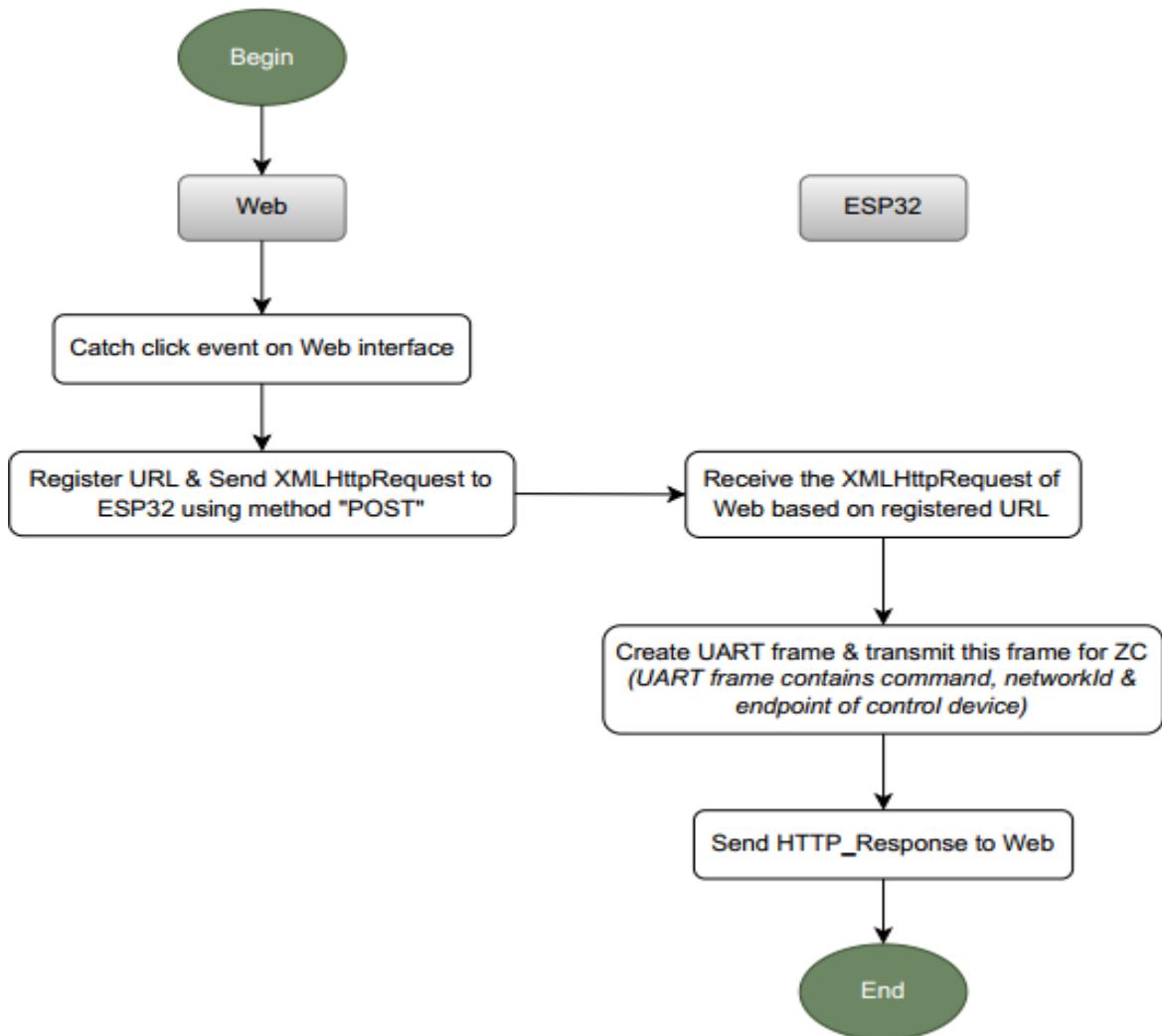
Xem tại [AlgorithmToProcessUARTdata](#)

❖ Sơ đồ thuật toán update dữ liệu lên Web



Hình 3.40: Sơ đồ thuật toán ESP32 xử lý bản tin nhận được từ ZC

❖ Sơ đồ thuật toán Web truyền lệnh điều khiển



Hình 3.41: Sơ đồ thuật toán Web truyền lệnh điều khiển

KHÓ KHĂN TRONG QUÁ TRÌNH HOÀN THIỆN ĐỒ ÁN

Thứ nhất, về thiết bị phần cứng: Do thời gian sử dụng dài kết hợp với điều kiện môi trường nên một số chức năng trên Kit IoT Zigbee không thể sử dụng được (*nút SW2 trên 01 Kit*) hoặc hoạt động sai (*cảm biến độ ẩm, cảm biến PIR, bắt số lần nháń/nháѣ trên SW1, SW2*) và các thiết bị hoạt động sai lại nằm trên các Kit khác nhau. Bên cạnh đó, nút SW1 (*kết nối với chân GPIO PD04*) sử dụng chung ngắt với cảm biến chuyển động PIR (*kết nối với chân GPIO PC04*) nên khó trong việc sử dụng đồng bộ hai thiết bị này trên bo mạch. Vì vậy, tôi đã phải thay đổi phương án nhiều lần, tập trung sử dụng nút SW2 trên Kit. Tất cả những điều đó làm ảnh hưởng đến trải nghiệm thực tế vì khó khăn hơn trong việc thực hiện các thao tác, thiết bị cảm biến hoạt động sai.

Thứ hai, về lập trình giao diện WebClient & Lập trình Kit ESP32

Lập trình giao diện WebClient : Đây là một ngôn ngữ lập trình hoàn toàn mới đối với tôi. Mặc dù tài liệu trên môi trường Internet về lập trình Web là rất nhiều, tuy nhiên nó lại không có tài liệu theo hướng bài bản. Do đó, việc xem những tài liệu về Web mất rất nhiều thời gian. Bên cạnh đó, kiến thức về lập trình Web là rất rộng với nhiều phương thức giao tiếp, nhiều kỹ thuật lập trình. Vì vậy, để kết hợp với yêu cầu đồ án tạo ra một giao diện WebClient theo ý tưởng của bản thân là một công việc khó đối với tôi.

Lập trình Kit ESP32: ESP32 cũng là một Kit mới đối với tôi, nó không được sử dụng trong chương trình học của Funix. Mặc dù, ESP32 có cả một thư viện về các ví dụ lập trình tính năng trên nó, nhưng để hiểu được ví dụ mà ứng dụng trong đồ án của bản thân cũng không hề dễ dàng: *Hiểu được các bước cấu hình, kết nối Wi-Fi như thế nào; Sử dụng NVS Flash trong việc lưu, đọc dữ liệu...* Tuy nhiên, với kiến thức nền tảng về VĐK và việc sử dụng một số công cụ lập trình được trang bị trong quá trình học tại Funix cũng đã giúp tôi làm quen với ESP32 nhanh hơn: *Sử dụng công cụ Visual Studio Code để lập trình ESP32; Sử dụng giao thức UART để giao tiếp với Kit IoT Zigbee...*

Thứ ba, về lập trình trên Kit IoT Zigbee: Mặc dù đã được trang bị các kiến thức nền tảng về Zigbee, tuy nhiên để thực hiện điều khiển đèn cầu thang sử dụng đồng bộ 3 phương pháp: Nút bấm cơ, WebClient và cảm biến chuyển động PIR, tôi cũng gặp rất nhiều khó khăn.

Thứ tư, về vẽ sơ đồ thuật toán: Trong thời gian học tại Funix, tôi chủ yếu thực hành theo hướng dẫn trong các Exercise, Lab, Assignment mà không thực hiện vẽ các loại sơ đồ trước khi lập trình. Vì vậy, việc hiểu các loại sơ đồ đối với tôi còn hạn chế và kỹ năng vẽ, trình bày kém.

Và cuối cùng, do hạn chế về khả năng và kinh nghiệm của bản thân.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Việc sử dụng “*Giải pháp IoT cho thiết bị công tắc cầu thang*” sẽ mang đến trải nghiệm tiện lợi, linh hoạt cho người sử dụng so với công tắc cầu thang truyền thống như tính năng tự động hóa việc bật/tắt, tính năng điều khiển từ xa bằng điện thoại di động và các thiết bị thông minh khác. Đây là một xu hướng không thể phủ nhận trong thế giới ngày nay, khi mà sự kết nối và tiện ích là những yếu tố quan trọng trong cuộc sống hiện đại. Thông qua đồ án này, tôi tạo ra “*Giải pháp IoT cho thiết bị công tắc cầu thang*” sử dụng đồng bộ 03 phương pháp để điều khiển đèn cầu thang (*nút bấm cơ, cảm biến chuyển động PIR và điều khiển từ xa bằng Web*).

Trong phạm vi đồ án, tôi tập trung vào việc phát triển sản phẩm và phát triển khả năng kết nối mạng để có thể điều khiển thông qua WebClient. Sau đồ án này, để tăng thêm tính năng của sản phẩm và nâng cao kiến thức của bản thân, tôi muốn tìm hiểu và làm thêm App điều khiển; cải thiện tính năng Login; sử dụng thêm các giao thức giao tiếp khác như MQTT, WebSocket; sử dụng hệ quản trị database MySQL...

DANH MỤC TÀI LIỆU THAM KHẢO

1. The Book “**ZigBee Wireless Networks and Transceivers**” of author **Shahin Farahani**.
2. The Book “**ZigBee Cluster Library Specification**” of **ZigBee Alliance**.
3. The Book “**Zigbee Application Framework Developer’s Guide for SDK 6.x and Lower** ” of **Silicon Labs**.
4. The Book “**EFR32xG21 Wireless Gecko Reference Manual** ” of **Silicon Labs**.
5. [**https://github.com/SiliconLabs/peripheral_examples/tree/master/series2**](https://github.com/SiliconLabs/peripheral_examples/tree/master/series2)
6. [**https://docs.silabs.com/zigbee/6.8/index**](https://docs.silabs.com/zigbee/6.8/index)
7. The Book “**ESP32 Technical Reference Manual**” of **Espressif Systems**.
8. [**https://docs.espressif.com/projects/esp-idf/en/v5.2.1/esp32/index.html**](https://docs.espressif.com/projects/esp-idf/en/v5.2.1/esp32/index.html)
9. Tra cứu các kiến thức về Web, Lập trình Web trên Internet.