Thesis Presentation Script

# Chào quý Thầy/Cô (30s)

Xin kính chào quý thầy trong hội đồng, toàn thể quý thầy cô, và tất cả mọi người. Em tên là Nguyễn Vũ Minh Thành và em xin trình bày nội dung của khóa luận “THỰC HIỆN NỀN TẢNG IOT SỬ DỤNG HỆ THỐNG BACK-END GIAO TIẾP PHẦN CỨNG QUA WIFI KẾT HỢP KỸ THUẬT MẬT MÃ HÓA NHẸ”. Người hướng dẫn khoa học của em là TS. Lê Đức Hùng.

# mục lục trình bày (30s)

Đầu tiên, em xin giới thiệu về cấu trúc báo cáo.

Các nội dung chính sẽ được trình bày bao gồm: Giới thiệu đề tài, tổng quan về đề tài, nội dung thực hiện, kết quả đạt được, biểu diễn kết quả, kết luận và hướng mở rộng.

# giới thiệu đề tài(1p)

Tiếp theo, em xin giới thiệu về đề tài.

Các trạm viễn thông VNPT Tiền Giang yêu cầu xây dựng hệ thống giám sát các trạm BTS online. Vì vậy, nền tảng IoT được xây dựng để cung cấp bo mạch giám sát tại các trạm BTS, hệ thống back-end thu nhận dữ liệu từ phần cứng, và giao diện giám sát và điều khiển online. Tại các trạm BTS, hệ thống làm mát gồm hai máy lại cần được giám sát và điều khiển online. Các chỉ số cần giám sát bao gồm nhiệt độ tại trạm, dòng điện và điện thế của các máy lạnh. Thông qua Internet, giao diện web tương tác với phần cứng để điều khiển các relay tương ứng với việc bật/tắt các máy lạnh. Hệ thống IoT giám sát BTS đã có hoàn thành, tuy nhiên, hệ thống này cần đưa ra các kỹ thuật cỗi lõi, giao diện linh hoạt và cách triển khai hệ thống back-end và mật mã hóa một cách hợp lý. Trong đề tài này, nền tảng IoT được xây dựng để giải quyết các vấn đề trên thông qua việc đưa ra các kỹ thuật cỗi lõi, giao diện linh hoạt, và cách thức triển khai hệ thống back-end và mật mã hóa.

# Tổng quan (1p)

Tiếp theo, em xin chuyển sang phần tổng quan.

Dựa trên yêu cầu giám sát các trạm BTS, nền tảng IoT được xây dựng để đưa ra giải pháp trên bo mạch, máy chủ ảo (VPS) , và giao diện người dùng.

Mô hình hệ thống được bố trí theo mô hình IoT 4 lớp. Trong đó, sensing layer chứa MCU STM32 và các cảm biến giao tiếp với nó. Tiếp theo, network layer chứa ESP32 IoT gateway chuyển tiếp dữ liệu và các ứng dụng server của hệ thống back-end. Cụ thể, hệ thống back-end chứa API server là trung tâm thu nhận dữ liệu, Mongo database lưu trữ dữ liệu, và MQTT Broker luân chuyển dữ liệu giữa phần cứng và VPS. Trên application layer, Nginx web server host giao diện web. Cuối cùng, data processing layer, các tương tác trên hệ thống diễn ra thông qua giao thức giao tiếp và đồng bộ có sử dụng mật mã hóa nhẹ ChaCha20-Poly1305.

# Nội dung chính đã thực hiện và kết quả (5p)

## Nội dung chính (4p – 4 slide)

Trong phần tiếp theo, em xin trình bày nội dung thực hiện.

### Giao thức frame

Em sẽ bắt đầu với giao thức frame.

Giao thức frame được thiết kế để thiết lập tương tác giữa sensing và network layer. Trong cấu trúc frame, các header dùng để nhận diện phần đầu frame, các trailer dùng để nhận diện cuối frame, body chứa dữ liệu giao tiếp, và các CRC dùng để kiểm tra lỗi trên body. Dựa trên cấu trúc frame, quá trình phân giải frame diễn ra tuần tự theo 4 bước. Đó nhận diện header, thu thập các byte trong body, nhận diện trailer, nhận các byte CRC và tính toán CRC. Quá trình phân giải thành công khi việc nhận diện header, trailer, và tính toán CRC thành công và quá trình phân giải thất bại khi việc nhận diện header, trailer và tính toán CRC thất bại hoặc quá trình phân giải timeout. Với giao thức frame, giao tiếp sẽ trở nên đáng tin cậy cũng như có thể triển khai kỹ thuật phân giải frame và mật mã hóa.

### Cơ chế đồng bộ

Tiếp theo, em chuyển sang cơ chế đồng bộ.

Cơ chế đồng bộ, đồng bộ thay đổi của các Virtual Storage trên database với phần cứng và giao diện. Hiện tại, nền tảng IoT cung cấp 32 Virtual Storage trên thiết bị và giao diện. Thông qua giao tiếp frame, thiết bị phần cứng và giao diện có thể thực hiện các tương tác để đọc và thay đổi dữ liệu của các Virtual Storage. Khi giá trị của một Virtual Storage thay đổi, sự kiện “change streams” trên MongoDB được kích hoạt và API server sẽ tiến hành đồng bộ các sự thay đổi đó với giao diện một cách trực tiếp và một cách gián tiếp với thiết bị phần cứng qua gateway.

### Giao diện web linh hoạt.

Em tiếp tục với giao diện web linh hoạt.

Trong các hệ thống IoT, cấu hình của các thiết bị luôn có thể thay đổi. Vì vậy, giao diện hiển thị cũng phải có tính linh hoạt và có thể tùy chỉnh để đáp ứng các thay đổi đó. Giao diện web cung cấp layout dạng lưới và các widget có sẵn. Trên mạng lưới đó, người dùng có thể kéo thả và thay đổi kích thước của các widget cũng như dữ liệu hiển thị của widget.

### Bo mạch và hệ thống back-end

Trong nội dung tiếp theo, đây là nội dung về bo mạch và hệ thống back-end.

Bo mạch và hệ thống back-end tương tác thông qua giao thức frame được đề cập. Trong đó, bo mạch phần cứng được sử dụng để xây dựng hệ thống IoT, là bo mạch phát triển IoT của PTN DESLab. Bo mạch này bao gồm vi điều khiển STM32F407VGT6 là thiết bị giao tiếp với cảm biến và ngoài và vi điều khiển ESP Wroom 32 là IoT gateway. Về hệ thống back-end, hệ thống bao gồm các ứng dụng server chạy trên VPS. Trong các ứng dụng server, MQTT Broker và API server là các ứng dụng được xây dựng trên framework NodeJS và database là ứng dụng mã nguồn mở của hệ quản trị cơ sở dữ liệu MongoDB.

### Mật mã hóa ChaCha20-Poly1305.

Cuối cùng, em trình bày về mật mã hóa ChaCha20-Poly1305.

ChaCha20-Poly1305 là mật mã hóa xác thực với dữ liệu liên kết. Đây là sự kết hợp của hai thuật toán độc lập. Đó là thuật toán mã hóa dòng tốc độ cao ChaCha20 và thuật toán xác thực tin nhắn tốc độ cao Poly1305. Thuật toán này được triển khai trên giao tiếp gateway-API server, cụ thể triển khai trên các byte data của giao thức frame. Quá trình mã hóa/giải mã xác thực với dữ liệu liên kết của thuật toán ChaCha20-Poly1305 diễn ra tuần tự theo 3 quá trình. Đó là thiết lập 512-bit ChaCha20 state, tạo 256-bit Poly1305 one-time key từ quá trình ChaCha20, sau đó là quá trình mã hóa/giải mã và tạo xác thực. Đối với quá trình mã hóa xác thực, quá trình tạo xác thực Poly1305 diễn ra sau quá trình mã hóa. Ngược lại, quá trình Poly1305 diễn ra trước quá trình giải mã đối với quá trình giải mã xác thực.

## Kết quả (1p)

Tiếp theo, em xin chuyển sang phần kết quả.

### Bo mạch

Trên bo mạch, kỹ thuật phân giải frame và cơ chế đồng bộ đã được cài đặt trên các vi điều khiển cũng như kỹ thuật mật mã hóa nhẹ ChaCha20-Poly1305 đã chạy trên ESP32 gateway.

### VPS

Trên VPS, hệ thống back-end được triển khai trên các port cụ thể thông qua kỹ thuật Docker Containerizing. Ngoài ra, kỹ thuật phân giải frame, đồng bộ, và mật mã hóa cũng đã được cài đặt trên API server. Cuối cùng, nội dung của giao diện web đã được host trên Nginx web server.

### Giao diện web

Về giao diện web, với các widget linh hoạt, dashboard giám sát và điều khiển đã được thiết kế cho bo mạch phát triển IoT của PTN DESLab và bo mạch của trạm BTS. Ngoài ra, giao diện web được host trên domain đáng tin cậy có giao thức HTTPS, bảo mật SSL. Cuối cùng, giao diện có cấu hình của một ứng dụng đa nền tảng thông qua chứng chỉ Progressive Web App.

# Biểu diễn kết quả (3p) (video demo)

Sau đây, em đến với phần biểu diễn kết quả.

Phần demo của đề tài thực hiện trên bo mạch giám sát trạm BTS và bao gồm 3 phần. Đó là tổng quan về giao diện web, các quá trình giao tiếp đồng bộ, và mã hóa cũng như cơ chế phát hiện lỗi trên giao tiếp.

## Tổng quan giao diện web.

Về giao diện web, tương tác của giao diện là chủ yếu dựa trên hai tab Configuration và Device. Trong tab Configuration, tab cung cấp tương tác với dữ liệu của configuration của các bo mạch. Khi mở một configuration, giao diện hiển thị các thông tin chi tiết của configuration. Hơn nữa, trong tab Configuration, tab Device cung cấp tương tác với dữ liệu của các thiết bị; tab Virtual Storage cung cấp tương tác với dữ liệu của các Virtual Storage; và tab UI cung cấp tương tác với các UI dashboard. Trong mỗi UI dashboard, chế độ chỉnh sửa cung cấp các widget. Các widget này có thể kéo thả và được cấu hình dữ liệu Virtual Storage mà nó hiển thị.

Trong tab Device, danh sách các thiết bị được hiển thị và có thể mở một tab của một thiết bị.

## Quá trình giao tiếp, đồng bộ, và mã hóa.

Về quá trình giao tiếp, đồng bộ, và mã hóa, bo mạch sẽ tiến hành giao tiếp với API server để cập nhật các Virtual Storage trên database. Khi dữ liệu cập nhật, giao diện web sẽ nhận được dữ liệu đồng bộ real-time từ API server. Các dữ liệu đó bao gồm: nhiệt độ phòng, tick count, các chỉ số dòng điện, điện thế, và trạng thái relay của hai bóng đèn. Trên giao diện web, việc điểu khiển các relay sẽ thay đổi dữ liệu trên database và bo mạch sẽ nhận được các cập nhật đó thông qua cơ chế đồng bộ.

Tiến hành bật relay 1, đèn số 1 đã sáng. Tiến hành bật relay 2, đèn số 2 đã sáng. Điện thế của hai bóng đèn tăng lần lượt lên ? và ? Volt.

Trên giao diện debug của ESP32 IoT gateway, debug thể hiện các quá trình giao tiếp từ thiết bị phần cứng và API server. Các quá trình giao tiếp chạy lần lượt các giai đoạn kiểm tra header, kiểm tra trailer, và tính toán CRC một cách đúng đắn. Một quá trình giao tiếp diễn ra trong khoảng vài trăm mili giây và quá trình mã hóa/giải mã diễn ra trong khoảng vài trăm micro giây.

## Cơ chế phát hiện lỗi.

Về cơ chế phát hiện lỗi, chế độ lỗi của bo mạch sẽ được bật thông qua giao diện web. Trong chế độ này, dữ liệu của Virtual Storage 0, tương ứng với giá trị nhiệt độ, sẽ bị làm sai. Vì vậy, trên giao diện debug, các giao tiếp sẽ xuất hiện. Cụ thể, đây là lỗi kiểm tra ? thất bại. Tuy nhiên, các giao tiếp trên các Virtual Storage khác vẫn diễn ra bình thường. Cụ thể, các chỉ số dòng điện, điện thế, tick count vẫn được cập nhật. Hơn nữa, việc điều khiển các relay vẫn có thể hoạt động. Tiến hành tắt relay 1, relay 1 đã tắt. Tiến hành tắt relay2, relay2 đã tắt.

# Kết luận, hướng mở rộng (1p)

Cuối cùng, đây là nội dung kết luận và hướng mở rộng.

Về kết quả, đề tài đã đưa ra nền tảng IoT có mô hình cụ thể. Trong đó, kỹ thuật giao tiếp, đồng bộ, và mật mã hóa đã được triển khai trên phần cứng và API server. Hơn nữa, kỹ thuật triển khai hệ thống back-end được xác định rõ ràng và giao diện có sự linh hoạt.

Về hạn chế và giới hạn, nền tảng IoT chưa đưa ra nhiều tương tác cho các kiểu dữ liệu Virtual Storage, và dữ liệu người dùng cũng như chưa có cơ chế phát sinh sự kiện. Về giao diện, giao diện chưa đưa ra nhiều lựa chọn widget cho người dùng.

Cuối cùng, về hướng mở rộng, nền tảng IoT có thể xây dựng thêm các tương tác với kiểu dữ liệu Virtual Storage và dữ liệu người dùng. Hơn nữa, hệ thống có thể phát triển cơ chế sự kiện và tạo ra thêm các loại widget trên giao diện.

# Kết thúc trình bày

Đó là tất cả nội dung mà em muốn trình bày. Xin cảm ơn quý thầy trong hội đồng, toàn thể quý thầy/cô, và mọi người đã theo dõi.